Contents

1. SQL Language Elements ............................................................................. 8
  1.1 c-treeACE SQL Identifiers ........................................................................ 8
  Conventional Identifiers ........................................................................... 9
  Delimited Identifiers ............................................................................... 9
  1.2 Data Types ............................................................................................ 10
  Character Data Types .............................................................................. 10
  Exact Numeric Data Types ..................................................................... 11
  Approximate Numeric Data Types .......................................................... 13
  Date-Time Data Types .......................................................................... 13
  Bit String Data Types ............................................................................ 14
  1.3 Search Conditions .................................................................................. 15
  Logical Operators:  OR, AND, NOT ......................................................... 16
  Relational Operators ............................................................................ 16
  Basic Predicate ..................................................................................... 17
  Quantified Predicate ............................................................................. 17
  BETWEEN Predicate ............................................................................ 18
  NULL Predicate .................................................................................... 18
  LIKE Predicate ..................................................................................... 18
  CONTAINS Predicate ........................................................................... 19
  EXISTS Predicate ................................................................................ 19
  IN Predicate ........................................................................................ 20
  Outer Join Predicate ............................................................................ 20
  1.4 Expressions .......................................................................................... 20
  Concatenated Character Expressions ...................................................... 22
  Numeric Arithmetic Expressions ........................................................... 23
  Date Arithmetic Expressions ................................................................. 24
  Conditional Expressions ..................................................................... 25
  Scalar Sub-query Expressions ................................................................. 25
  Escape field name matching reserved words in c-tree expressions .......... 26
  1.5 Literals .................................................................................................. 26
  Numeric Literals .................................................................................... 27
  Character String Literals ....................................................................... 27
  Date-Time Literals ............................................................................... 27
  Date Literals ........................................................................................ 28
  Time Literals ........................................................................................ 29
  Timestamp Literals ............................................................................... 30
  1.6 Query Expressions ................................................................................ 30
  1.7 Inner Joins .......................................................................................... 37
1.8 Outer Joins ........................................................................... 39
1.9 Date-Time Format Strings ..................................................... 41
Date Format Strings ................................................................ 41
Time Format Strings ................................................................ 42
1.10 Functions ........................................................................... 43
Aggregate Functions .................................................................. 43
Scalar Functions ....................................................................... 49

2. SQL Statements .................................................................... 175
2.1 Syntax Conventions ............................................................... 175
Column Constraints .................................................................. 201
Table Constraints ..................................................................... 203
Storage Attributes ..................................................................... 205
SQL - Create [if not exists] and Drop [if exists] ......................... 206

3. c-treeACE SQL Reserved Words ............................................ 254
3.1 c-treeACE SQL Reserved Words ............................................ 254

4. Error Messages ..................................................................... 256
4.1 0, 10000 Series .................................................................... 257
4.2 11000 Series - c-treeACE SQL RDS Errors ......................... 258
4.3 15000 Series - c-treeACE SQL Flat-File Storage System ...... 259
4.4 160xx - Main Memory Storage System ................................. 260
4.5 16000 Series - Latte and Polka Subsystems ......................... 262
4.6 c-tree FSS Errors ................................................................. 264
   170xx ................................................................................. 264
   171xx ................................................................................. 268
   173xx ................................................................................. 272
   174xx ................................................................................. 272
   175xx ................................................................................. 276
   177xx ................................................................................. 279
   178xx ................................................................................. 283
   179xx ................................................................................. 288
   18000 Series ........................................................................ 291
4.7 -202xx - Sequence Support .................................................. 292
4.8 20000 Series - c-treeACE SQL ............................................ 293
4.9 21000 Series - c-treeDB FSS DB Layer ................................ 304
4.10 22000 Series - c-treeDB FSS DB Layer ............................... 310
4.11 23000 Series - c-treeACE SQL Data Exceptions ................ 315
4.12 250xx - c-treeACE SQL ODBC Integrator ......................... 315
4.13 251xx - c-treeACE SQL ODBC Driver .............................................318
4.14 26000 Series - c-treeACE SQL JDBC Driver .......................................320
4.15 30000 Series - c-treeACE SQL Network .............................................320
4.16 40000 Series - c-treeACE SQL Environment Errors ..........................323
4.17 50000 Series - c-treeACE SQL DHRSS Errors ....................................323

5. System Catalog Tables .............................................................................324
  5.1 Overview .................................................................................................324
  5.2 System Catalog Tables Definitions .........................................................324

6. System Limits ............................................................................................332
  6.1 Maximum Values for c-treeACE SQL Attributes ...................................332

7. Glossary .......................................................................................................334

8. Index ...........................................................................................................343
FairCom Typographical Conventions

Before you begin using this guide, be sure to review the relevant terms and typographical conventions used in the documentation.

The following formatted items identify special information.

<table>
<thead>
<tr>
<th>Formatting convention</th>
<th>Type of Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold</strong></td>
<td>Used to emphasize a point or for variable expressions such as parameters</td>
</tr>
<tr>
<td><strong>CAPITALS</strong></td>
<td>Names of keys on the keyboard. For example, SHIFT, CTRL, or ALT+F4</td>
</tr>
</tbody>
</table>

*FairCom Terminology*

<table>
<thead>
<tr>
<th>FunctionName()</th>
<th>FairCom technology term</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter</strong></td>
<td>c-treeACE Function Parameter</td>
</tr>
<tr>
<td>Code Example</td>
<td>Code example or Command line usage</td>
</tr>
<tr>
<td><strong>utility</strong></td>
<td>c-treeACE executable or utility</td>
</tr>
<tr>
<td><strong>filename</strong></td>
<td>c-treeACE file or path name</td>
</tr>
</tbody>
</table>

**CONFIGURATION KEYWORD**

| CTREE_ERR              | c-treeACE Error Code |
1. SQL Language Elements

This chapter describes language elements that are common to many c-treeACE SQL statements. Syntax diagrams in other chapters often refer to these language elements without detailed explanation. The major syntax elements described in the following sections are:

**Identifiers** - User-supplied names for elements such as tables, views, cursors, and columns. c-treeACE SQL statements use those names to refer to the elements.

**Data types** - Control how c-treeACE SQL stores column values.

**Query expressions** - Retrieve values from tables. Query expressions form the basis of other c-treeACE SQL statements and syntax elements.

**Search conditions** - Specify a condition that is true or false about a given row or group of rows. Query expressions and **UPDATE** statements specify search conditions to restrict the number of rows in the result table.

**Expressions** - A symbol or string of symbols used to represent or calculate a single value in a c-treeACE SQL statement. When c-treeACE SQL encounters an expression, it retrieves or calculates the value represented by the expression and uses that value when it executes the statement.

**Literals** - A type of c-treeACE SQL expression that specify a constant value. Some SQL constructs allow literals but prohibit other forms of expressions.

**Date-time format strings** - Control the output of date and time values. c-treeACE SQL interprets format strings and replaces them with formatted values.

**Functions** - A type of c-treeACE SQL expression that returns a value based on the argument supplied. Aggregate functions calculate a single value for a collection of rows in a result table. Scalar functions calculate a value based on another single value.

1.1 c-treeACE SQL Identifiers

SQL syntax requires users to supply names for elements such as tables, views, cursors, and columns when they define them. SQL statements then use those names to refer to the table, view, or other element. In syntax diagrams, c-treeACE SQL identifiers are shown in lowercase type.

The maximum length for c-treeACE SQL identifiers is 64 characters.

There are two types of c-treeACE SQL identifiers:

- **Conventional identifiers** (page 9, [http://docs.faircom.com/doc/sqlref/33346.htm](http://docs.faircom.com/doc/sqlref/33346.htm))
- **Delimited identifiers** ([http://docs.faircom.com/doc/sqlref/33347.htm](http://docs.faircom.com/doc/sqlref/33347.htm)) enclosed in double quotation marks
Conventional Identifiers

Unless they are delimited identifiers (refer to Delimited Identifiers (page 9)), c-treeACE SQL identifiers must:

- Begin with an uppercase or lowercase letter
- Contain only letters, digits, or the underscore character ( _ )
- Not be reserved words

Except for delimited identifiers, c-treeACE SQL does not distinguish between uppercase and lowercase letters in identifiers. By default all names are converted to lowercase, however, statements can refer to the names in mixed case. The following examples show some of the characteristics of conventional identifiers:

```
-- Names are case-insensitive:
CREATE TABLE Test (Column1 CHAR);
INSERT INTO TEST (COLUMN1) VALUES('1');
1 record inserted.
SELECT * FROM TEST;
COL
---
1
1 record selected
TABLE TEST;

<table>
<thead>
<tr>
<th>COLNAME</th>
<th>NULL ?</th>
<th>TYPE</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>column1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
-- Cannot use reserved words:
CREATE TABLE TABLE (COL1 CHAR);
CREATE TABLE TABLE (COL1 CHAR);
* error(-20003): Syntax error
```

Delimited Identifiers

Delimited identifiers are c-treeACE SQL identifiers enclosed in double quotation marks ("""). Enclosing a name in double quotation marks, by default, preserves the case of the name and allows it to be a reserved word and special characters. (Special characters are any characters other than letters, digits, or the underscore character.) Subsequent references to a delimited identifier must also use enclosing double quotation marks. To include a double-quoted character in a delimited identifier, precede it with another double quotation mark.

The following c-treeACE SQL example shows several ways to create and refer to delimited identifiers:

```
CREATE TABLE "delimited ids"
   ( """"" CHAR(10),
     "_uscore" CHAR(10),
     """quote" CHAR(10),
     " space" CHAR(10) );
INSERT INTO "delimited ids" (""""" VALUES('text string');
1 record inserted.
SELECT * FROM "delimited ids";
" _USCORE "QUOTE "SPACE
- --------- ------- -------
text string
1 record selected
```
1.2 Data Types

The SQL statements `CREATE TABLE` and `ALTER TABLE` specify data types for each column in the tables they define. This section describes the data types `ctreeACE SQL` supports for table columns.

There are several categories of `ctreeACE SQL` data types:

- Character
- Exact numeric
- Approximate numeric
- Date-time
- Bit String

All of the data types can store null values. A null value indicates that the value is not known and is distinct from all non-null values.

**Syntax**

\[
data_type ::\]
\[
\text{char_data_type} \quad | \quad \text{exact_numeric_data_type} \quad | \quad \text{approx_numeric_data_type} \quad | \quad \text{date_time_data_type} \quad | \quad \text{bit_string_data_type}
\]

**Character Data Types**

Refer to *Character String Literals* (page 27) for details on specifying values to be stored in character columns.

**Syntax**

\[
char_data_type ::
\]
\[
\{ \text{CHARACTER} \mid \text{CHAR} \} \mid (\text{length})
\]
\[
\} \quad | \quad \{ \text{CHARACTER VARYING} \mid \text{CHAR VARYING} \mid \text{VARCHAR} \} \mid (\text{length})
\]
\[
\} \quad | \quad \text{LVARCHAR} \mid \text{LONG VARCHAR}
\]

**Arguments**

- `{ \text{CHARACTER} \mid \text{CHAR} } [(\text{length})] [ \text{CHARACTER SET charset-name} ]`
  Type \text{CHARACTER} (abbreviated as \text{CHAR}) corresponds to a null terminated character string with the maximum length specified. The default length is 1. The maximum length is 8192.

- \text{LVARCHAR} \mid \text{LONG VARCHAR}
  Type \text{LONGVARCHAR} corresponds to an arbitrarily long character string with a maximum length of two gigabytes (2 Gb). The \text{LVARCHAR} data type is recommended for CLOB support. The arbitrary size and unstructured nature of \text{LONG} data types restrict where they can be used.
• LONG columns are allowed in select lists of query expressions and in INSERT and UPDATE statements.
• INSERT statements can store data from columns of any type except LONG VARBINARY into a LONG VARCHAR column, however, LONG VARCHAR data cannot be stored in any other type.
• CONTAINS predicates are the only predicates that allow LONG columns.
• Conditional expressions, arithmetic expressions, and functions cannot specify LONG columns.
• UPDATE statements cannot specify LONG columns.

- \{ CHARACTER VARYING | CHAR VARYING | VARCHAR \} [(length)]
  Type CHARACTER VARYING corresponds to a variable-length character string with the maximum length specified.
  - The default length for columns defined as CHARACTER VARYING is 1. The maximum length is 8192.

\section*{Exact Numeric Data Types}

Refer to \textit{Numeric Literals} (page 27) for details on specifying values to be stored in numeric columns.

\textbf{Syntax}

```
exact_numeric_data_type ::=
    TINYINT
    | SMALLINT
    | INTEGER
    | BIGINT
    | NUMERIC | NUMBER [ ( precision [ , scale ] ) ]
    | DECIMAL [(precision, scale)]
    | MONEY [(precision)]
    [ IDENTITY [ ( \pm seed , \pm increment ) ] ]
```

\textbf{Arguments}

- \textbf{TINYINT}
  Type TINYINT corresponds to an integer value stored in one byte. The range of TINYINT is -128 to 127.

- \textbf{SMALLINT}
  Type SMALLINT corresponds to an integer value of length two bytes. The range of SMALLINT is -32768 to +32767.

- \textbf{INTEGER}
  Type INTEGER corresponds to an integer of length four bytes. The range of values for INTEGER columns is \(-2^{31}\) to \(2^{31}-1\).

- \textbf{BIGINT}
  Type BIGINT corresponds to an integer of length eight bytes. The range of values for BIGINT columns is \(-2^{63}\) to \(2^{63}-1\).

- \[ IDENTITY [ (\pm seed , \pm increment ) ] \]
For TINYINT, SMALLINT, INTEGER and BIGINT column types, an optional auto-incrementing attribute can be defined with the IDENTITY option. This adds the column of the defined type to the table and automatically updates the value on each row insert. IDENTITY does not guarantee uniqueness of assigned values. IDENTITY can optionally specify seed and increment values. seed is the starting assignment value and is incremented by increment for each update.

```
CREATE TABLE t1 (name CHAR(10), id_num INTEGER IDENTITY (0, 1));
```

Only one IDENTITY column can be defined per table. IDENTITY columns cannot be specified on tables with only one column.

IDENTITY values assigned to aborted rows in a table are lost. Note that this can result in gaps in the numerical sequence order.

IDENTITY is not supported for NUMERIC, NUMBER, DECIMAL or MONEY column types. IDENTITY can not be added to an existing field via ALTER TABLE.

- **NUMERIC | NUMBER [ ( precision [ , scale ] ) ]**

Type NUMERIC corresponds to a number with the given precision (maximum number of digits) and scale (the number of digits to the right of the decimal point). By default, NUMERIC columns have a precision of 32 and scale of 0. If NUMERIC columns omit the scale, the default scale is 0.

The range of values for a NUMERIC type column is -n to +n where n is the largest number that can be represented with the specified precision and scale. If a value exceeds the precision of a NUMERIC column, c-treeACE SQL generates an overflow error. If a value exceeds the scale of a NUMERIC column, c-treeACE SQL rounds the value.

NUMERIC type columns cannot specify a negative scale or specify a scale larger than the precision.

The following example shows what values will fit in a column created with a precision of 3 and scale of 2:

```
insert into t4 values(33.33);
error(-20052): Overflow error
insert into t4 values(33.9);
error(-20052): Overflow error
insert into t4 values(3.3);
1 record inserted.
insert into t4 values(33);
error(-20052): Overflow error
insert into t4 values(3.33);
1 record inserted.
insert into t4 values(3.33333);
1 record inserted.
insert into t4 values(3.3555);
1 record inserted.
select * from t4;
   C1
  --
   3.30
   3.33
   3.33
   3.36
4 records selected
```

- **DECIMAL [(precision, scale)]**

Type DECIMAL is comparable to type NUMERIC.
• **MONEY [(precision)]**
  
  Type **MONEY** is equivalent to type **NUMERIC** with a fixed scale of 2.

---

**Approximate Numeric Data Types**

Refer to *Numeric Literals* (page 27) for details on specifying values to be stored in numeric columns.

**Syntax**

```
approx_numeric_data_type ::
    REAL
    |    DOUBLE [ PRECISION ]
    |    FLOAT [ (precision) ]
```

**Arguments**

- **REAL**
  Type **REAL** corresponds to a single precision floating point number equivalent to the C language float type.

- **DOUBLE PRECISION**
  Type **DOUBLE PRECISION** corresponds to a double precision floating point number equivalent to the C language double type.

  In V11 and later, **DOUBLE** can be used as an alias for **DOUBLE PRECISION**.

- **FLOAT [ (precision) ]**
  Type **FLOAT** corresponds to a double-precision floating point number of the given precision. By default, **FLOAT** columns have a precision of 8.

---

**Date-Time Data Types**

Refer to *Numeric Literals* (page 27) for details on specifying values to be stored in date-time columns. See *Date Format Strings* (page 41) for details on date strings and *Time Format Strings* (page 42) for details on using format strings to specify the output format of date-time columns.

**Syntax**

```
date_time_data_type ::
    DATE
    |    TIME
    |    TIMESTAMP
```

**Arguments**

- **DATE**
  Type **DATE** stores a date value as three parts: year, month, and day. The range for the parts is:
  - Year: 1700 to 9999 (in the specific case of 1700 the actual minimum date is 03/01/1700 -- 1 March 1700)
  - Month: 1 to 12
The c-tree date field stores date as an unsigned 4 bytes integer value representing the number of days since 02/28/1700. Hence the first valid date is 03/01/1700.

A check is done to limit the date range lower bound to 03/01/1700 in case of a c-tree Plus date type and 01/01/1900 for the c-treeACE SQL TIMESTAMP type.

- **TIME**
  Type TIME stores a time value as three parts: hours, minutes, and seconds. The range for the parts is:
  - Hours: 0 to 23
  - Minutes: 0 to 59
  - Seconds: 0 to 59

- **TIMESTAMP**
  Type TIMESTAMP combines the parts of DATE and TIME.
  - Year: 1900 to 9999
  - Month: 1 to 12
  - Day: Lower limit is 1. The upper limit is determined by the upper limit of the month.
  - Hours: 0 to 23
  - Minutes: 0 to 59
  - Seconds: 0 to 59

### Bit String Data Types

**Syntax**

```sql
bit_string_data_type ::=
  BIT
  | BINARY [(length)]
  | VARBINARY [(length)]
  | LVARBINARY | LONG VARBINARY
```

**Arguments**

- **BIT**
  Type BIT corresponds to a single bit value of 0 or 1.
  c-treeSQL statements can assign and compare values in BIT columns to and from columns of types BINARY, VARBINARY, TINYINT, SMALLINT, INTEGER, and Character types. However, in assignments from BINARY, VARBINARY, and LONG VARBINARY, the value of the first four bits must be 0001 or 0000.
  No arithmetic operations are allowed on BIT columns.

- **BINARY [(length)]**
  Type BINARY corresponds to a bit field of the specified length of bytes. The default length is 1 byte. The maximum length is 8192 bytes.
  In interactive SQL, INSERT statements must use a special format to store values in BINARY columns. They can specify the binary values as a bit string, hexadecimal string, or character string. INSERT statements must enclose binary values in single-quote marks, preceded by `b` for a bit string and `x` for a hexadecimal string:
Prefix | Suffix | Example (for same 2 byte data)
--- | --- | ---
bit string | b' | b'1010110100010000'
hex string | x' | x'ad10'
char string | | 'ad10'

c-treeACE SQL interprets a character string as the character representation of a hexadecimal string.
If the data inserted into a BINARY column is less than the length specified, c-treeACE SQL pads it with zeroes.

BINARY data can be assigned and compared to and from columns of type BIT, VARBINARY, and Character types. No arithmetic operations are allowed.

- **VARBINARY [(length)]**
  Type VARBINARY corresponds to a variable-length bit field with the maximum length specified. The default length is one and the maximum length is 8192. Otherwise, VARBINARY columns have the same characteristics as BINARY.

- **LVARBINARY | LONG VARBINARY**
  Type LONG VARBINARY corresponds to an arbitrarily long bit field with the maximum length of less than two gigabytes. The LVARBINARY data type is recommended for BLOB support.
The arbitrary size and unstructured nature of LONG data types restrict where they can be used.
- LONG columns are allowed in select lists of query expressions and in INSERT statements.
- INSERT statements can store data from columns of any type except LONG VARCHAR into a LONG VARBINARY column, but LONG VARBINARY data cannot be stored in any other type.
- CONTAINS predicates are the only predicates that allow LONG columns.
- Conditional expressions, arithmetic expressions, and functions cannot specify LONG columns.
- UPDATE statements cannot specify LONG columns.

### 1.3 Search Conditions

A search condition specifies a condition that is true or false about a given row or group of rows. Query expressions and UPDATE statements can specify a search condition. The search condition restricts the number of rows in the result table for the query expression or UPDATE statement.

Search conditions contain one or more predicates. The predicates that can be part of a search condition are described in the following subsections.

**Syntax**

```
search_condition ::=
    [NOT] predicate
    [ { AND | OR } { predicate | { search_condition } } ]
predicate ::= basic_predicate
    | quantified_predicate
```
Logical Operators: OR, AND, NOT

Logical operators combine multiple search conditions. c-treeACE SQL evaluates multiple search conditions in this order:

1. Search conditions enclosed in parentheses. (For nested search conditions in parentheses, c-treeACE SQL evaluates the innermost search condition first.)
2. Search conditions preceded by NOT
3. Search conditions combined by AND
4. Search conditions combined by OR

Examples

```sql
SELECT * 
FROM customer 
WHERE name = 'LEVIEN' OR name = 'SMITH' ;
SELECT * 
FROM customer 
WHERE city = 'PRINCETON' AND state = 'NJ' ;
SELECT * 
FROM customer 
WHERE NOT (name = 'LEVIEN' OR name = 'SMITH') ;
```

Relational Operators

Relational operators specify how c-treeACE SQL compares expressions in basic and quantified predicates.

Syntax

```
relop ::=
  = | <> | != | ^=
  | <
  | <=
  | >
  | >=
```

<table>
<thead>
<tr>
<th>Relational Operator</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>True if the two expressions are equal.</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>!=</td>
</tr>
<tr>
<td>&lt;</td>
<td>True if the first expression is less than the second expression.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>True if the first expression is less than or equal to the second expression</td>
</tr>
</tbody>
</table>
### Relational Operator

<table>
<thead>
<tr>
<th>Relational Operator</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>True if the first expression is greater than the second expression.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>True if the first expression is greater than or equal to the second expression.</td>
</tr>
</tbody>
</table>

Refer to *Basic Predicate* (page 17) and *Quantified Predicate* (page 17) for more information.

## Basic Predicate

**Description**

A basic predicate compares two values using a relational operator (see *Relational Operators* (page 16) for more information). If a basic predicate specifies a query expression, then the query expression must return a single value. Basic predicates often specify an inner join. See *Inner Joins* (page 37) for more detail.

If the value of any expression is null or the `query_expression` does not return any value, then the result of the predicate is set to false.

**Syntax**

```
basic_predicate ::
    expr relop { expr | (query_expression) }
```

## Quantified Predicate

**Description**

The quantified predicate compares a value with a collection of values using a relational operator (see *Relational Operators* (page 16) for more detail). A quantified predicate has the same form as a basic predicate with the `query_expression` being preceded by `ALL`, `ANY`, or `SOME` keyword. The result table returned by `query_expression` can contain only a single column.

When `ALL` is specified the predicate evaluates to true if the `query_expression` returns no values or the specified relationship is true for all the values returned.

When `SOME` or `ANY` is specified the predicate evaluates to true if the specified relationship is true for at least one value returned by the `query_expression`. There is no difference between the `SOME` and `ANY` keywords. The predicate evaluates to false if the `query_expression` returns no values or the specified relationship is false for all the values returned.

**Syntax**

```
quantified_predicate ::
    expr relop { ALL | ANY | SOME } (query_expression)
```

**Example**

```
10 < ANY ( SELECT COUNT(*)
            FROM order_tbl
            GROUP BY custid )
```
**BETWEEN Predicate**

**Description**

The BETWEEN predicate can be used to determine if a value is within a specified value range or not. The first expression specifies the lower bound of the range and the second expression specifies the upper bound of the range.

The predicate evaluates to true if the value is greater than or equal to the lower bound of the range, or less than or equal to the upper bound of the range.

**Syntax**

between_predicate ::

```
expr [ NOT ] BETWEEN expr AND expr
```

**Example**

```
salary BETWEEN 2000.00 AND 10000.00
```

**NULL Predicate**

**Description**

The NULL predicate can be used for testing null values of database table columns.

**Syntax**

null_predicate ::

```
column_name IS [ NOT ] NULL
```

**Example**

```
contact_name IS NOT NULL
```

**LIKE Predicate**

**Description**

The LIKE predicate searches for strings that have a certain pattern. The pattern is specified after the LIKE keyword in a scalar expression which can be as simple as a string constant or a complex expression containing concatenation operators, scalar functions, etc. The pattern can be specified by a string in which the underscore ( _ ) and percent sign ( % ) characters have special semantics.

The ESCAPE clause can be used to disable the special semantics given to characters ' _ ' and ' % '. The escape character specified must precede the special characters to disable their special semantics.

**Syntax**

like_predicate ::

```
column_name [ NOT ] LIKE expr

[ ESCAPE escape-character ]
```

```
Notes
- The column name specified in the LIKE predicate must refer to a character string column.
- A percent sign in the pattern matches zero or more characters of the column string.
- A underscore sign in the pattern matches any single character of the column string.

Examples
cust_name LIKE 'Computer'
cust_name LIKE '

item_name LIKE '

item_name LIKE left(part_name, 5) + '

In the first example, for all strings with the substring Computer, the predicate will evaluate to true. In the second example, for all strings which are exactly three characters long, the predicate will evaluate to true. In the third example, the backslash character ‘
’ has been specified as the escape character, which means that the special interpretation given to the character ‘_’ is disabled. The pattern will evaluate to TRUE if the column item_name has embedded underscore characters.

In the fourth example, the LIKE predicate evaluates to true if the first 5 characters of item_name match the first 5 characters of part_name.

CONTAINS Predicate

Description
The c-treeACE SQL CONTAINS predicate is an extension to the SQL standard providing search capabilities for LONG character and binary data.

Syntax
column_name [ NOT ] CONTAINS 'string'

Limitations
- column_name must be one of the following data types: Fixed or varying-length character type, LONG VARCHAR, BINARY, VARBINARY, or LONG VARBINARY.
- Unlike the LIKE clause, the wildcard characters ‘%’ and ‘_’ are not allowed with CONTAINS.
- CONTAINS does not take advantage of indices to perform the search.

EXISTS Predicate

Description
The EXISTS predicate can be used to check for the existence of specific rows. The query_expression returns rows rather than values. The predicate evaluates to true if the number of rows returned by the query_expression is non-zero. The EXISTS predicate can be used inside of a WHERE, HAVING, or JOIN statement.
See also *Search Conditions* (page 15).

**Syntax**

exists_predicate ::
    EXISTS ( query_expression )

**Example**

```sql
SELECT cm_custname FROM custmast WHERE EXISTS (SELECT co_ordrdate FROM custordr WHERE cm_custnumb = custordr.co_custnumb)
```

In this example, the predicate will evaluate to true if the specified customer has any orders.

### IN Predicate

**Description**

The **IN** predicate can be used to compare a value with a set of values. If an **IN** predicate specifies a query expression, then the result table it returns can contain only a single column.

**Syntax**

```sql
in_predicate ::
    expr [ NOT ] IN { ( query_expression ) | ( constant , constant [ , ... ] ) }
```

**Example**

```sql
address.state IN ('MA', 'NH')
```

### Outer Join Predicate

**Description**

An outer join predicate specifies two tables and returns a result table that contains all of the rows from one of the tables, even if there is no matching row in the other table. See *Outer Joins* (page 39) for more information.

**Syntax**

```sql
outer_join_predicate ::
    [ table_name. ] column = [ table_name. ] column (+) |
    [ table_name. ] column (+) = [ table_name. ] column
```

### 1.4 Expressions

An expression is a symbol or string of symbols used to represent or calculate a single value in a c-treeACE SQL statement. When you specify an expression in a statement, c-treeACE SQL retrieves or calculates the value represented by the expression and uses that value when it executes the statement.

Expressions are also called scalar expressions or value expressions.
Arguments

A column in a table.

You can qualify column names with the name of the table they belong to:

```
SELECT CUSTOMER.CUSTOMER_ID FROM CUSTOMERS
```

You must qualify a column name if it occurs in more than one table specified in the `FROM` clause:

```
SELECT CUSTOMER.CUSTOMER_ID
       FROM CUSTOMERS, ORDERS
```

Qualified column names are always allowed even when they are not required.

You can also qualify column names with an alias. Aliases are also called correlation names.

The `FROM` clause of a query expression can specify an optional alias after the table name (see `Query Expressions` (page 30) for more details). If you specify an alias, you must use it -not the table name - to qualify column names that refer to the table. Query expressions that join a table with itself must use aliases to distinguish between references to column names.

The following example shows a query expression that joins the table customer with itself. It uses the aliases `x` and `y` and returns information on customers in the same city as customer 'SMITH':

```
SELECT y.cust_no, y.name
       FROM customer x, customer y
       WHERE x.name = 'SMITH'
       AND y.city = x.city ;
```

**character-literal | numeric-literal | date-time-literal**

Literals that specify a constant value. See `Literals` (page 26) and subsequent pages for more details on all types of literals.

**aggregate-function | scalar function**

A c-treeACE SQL function. See `Functions` (page 43) for details.

**concatenated-char-expr**

An expression that concatenates multiple character expressions into a single character string. See `Concatenated Character Expressions` (page 22) for more details.

**numeric-arith-expr**
An expression that computes a value from numeric values. See *Numeric Arithmetic Expressions* (page 23) for more details.

**date-arith-expr**

An expression that computes a value from date-time values. See *Date Arithmetic Expressions* (page 24) for more details.

**conditional-expr**

An expression that evaluates a search condition or expression and returns one of multiple possible results depending on that evaluation. See *Conditional Expressions* (page 25) for more details.

**scalar-subquery-expr**

An expression which is a scalar sub-query. A scalar sub-query returns only one value. See *Scalar Sub-query Expressions* (page 25) for more details.

**( expr )**

An expression enclosed in parentheses. c-treeACE SQL evaluates expressions in parentheses first.

### Concatenated Character Expressions

**Description**

The || concatenation operator (two vertical bars) || or + (plus) concatenates the two character expressions it separates.

The concatenation operator is similar to the CONCAT scalar function. However, the concatenation operator allows easy concatenation of more than two character expressions, while the CONCAT scalar function requires nesting.

**Syntax**

```sql
classified-char-expr ::
{character-literal | character-expr } { || | + } { character-literal | character-expr} 
{ {character-literal | character-expr } { || | + } { character-literal | character-expr} } [ ... ]
```

**Arguments**

**character-literal**

A character literal. Refer to *Character String Literals* (page 27) for details on specifying character literals.

**character-expr**

Any expression that evaluates to a character string (refer to *Data Types* (page 10) for details of character data types), including column names and scalar functions that return a character string.

**Examples**

```sql
ISQL> SELECT 'Today''s date is ' || TO_CHAR(SYSDATE) FROM SYSCALCTABLE;
TODAY'S DATE IS 08/17/1998
--------------------------
```
Today's date is 08/17/1998
1 record selected

ISQL> SELECT 'Today''s date is ' + TO_CHAR(SYSDATE) FROM SYSCALCTABLE;
TODAY'S DATE IS 08/17/1998
--------------------------
Today's date is 08/17/1998
1 record selected

Numeric Arithmetic Expressions

Description
Numeric arithmetic expressions compute a value using addition, subtraction, multiplication, and division operations on numeric literals and expressions that evaluate to any numeric data type.

Syntax
numeric-arith-expr ::
[ + | - ] { numeric-literal | numeric-expr } [ { + | - | * | / } numeric-arith-expr ]

Arguments
[ + | - ]
Unary plus or minus operator

numeric-literal
A numeric literal. Refer to Numeric Literals (page 27) for details on specifying numeric literals.

numeric-expr
Any expression that evaluates to a numeric data type (Refer to Data Types (page 10) for details of numeric data types), including:
- Column names
- Subqueries that return a single value
- Aggregate functions
- CAST or CONVERT operations to numeric data types
- Other scalar functions that return a numeric data type

{ + | - | * | / }
Addition, subtraction, multiplication, or subtraction operator. c-treeACE SQL evaluates numeric arithmetic expressions in the following order:
- Unary plus or minus
- Expressions in parentheses
- Multiplication and division, from left to right
- Addition and subtraction, from left to right
Date Arithmetic Expressions

Description
Date arithmetic expressions compute the difference between date-time expressions in terms of days or milliseconds. c-treeACE SQL supports these forms of date arithmetic:

- Addition and subtraction of integers to and from date-time expressions
- Subtraction of a date-time expression from another

Syntax

```
date_arith_expr ::
   date_time_expr { + | - } int_expr
 | date_time_expr - date_time_expr
```

Arguments
date_time_expr
An expression that returns a value of type DATE or TIME or TIMESTAMP. A single date-time expression can not mix data types. All elements of the expression must be the same data type.

Date-time expressions can contain date-time literals, but they must be converted to DATE or TIME using the CAST, CONVERT, or TO_DATE functions (see the following examples as well as CAST function (SQL-92 compatible) (page 57) and CONVERT function (extension) (page 67)).

int_expr
An expression that returns an integer value. c-treeACE SQL interprets the integer differently depending on the data type of the date-time expression:

- For DATE expressions, integers represent days
- For TIME expressions, integers represent milliseconds
- For TIMESTAMP expressions, integers represent milliseconds

Examples
The following example manipulates DATE values using date arithmetic. c-treeACE SQL interprets integers as days and returns date differences in units of days:

```
SELECT C1, C2, C1-C2 FROM DTEST
C1          C2          C1-C2
1956-05-07  1952-09-29  1316
```

```
select sysdate,
       sysdate - 3,
       sysdate - cast ('9/29/52' as date)
from dtest;
sysdate         sysdate-3         sysdate-convert(date,9/29/52)
1995-03-24      1995-03-21        15516
```

The following example manipulates TIME values using date arithmetic. c-treeACE SQL interprets integers as milliseconds and returns time differences in milliseconds:

```
select systime,
       systime - 3000,
       systime - cast ('15:28:01' as time)
from dtest;
```
Conditional Expressions

Conditional expressions are a subset of scalar functions that generate different results depending on the value of their arguments. They provide some of the flexibility of traditional programming constructs to allow expressions to return alternate results depending on the value of their arguments.

The following scalar functions provide support for conditional expressions. Refer to *Scalar Functions* (page 49) for a description of each function.

**CASE**

CASE is the most general conditional expression. It specifies a series of search conditions and associated expressions. c-treeACE SQL returns the value specified by the first expression whose associated search condition evaluates as true. If none of the expressions evaluate as true, the CASE expression returns a null value (or the value of some other default expression if the CASE expression includes the ELSE clause).

All the other conditional expressions can also be expressed as CASE expressions.

**DECODE**

DECODE provides a subset of the functionality of CASE that is compatible with Oracle SQL syntax. DECODE is not SQL-92 compatible.

**NULLIF**

NULLIF substitutes a null value for an expression if it is equal to a second expression.

**COALESCE**

COALESCE specifies a series of expressions. c-treeACE SQL returns the first expression whose value is not null. If all the expressions evaluate as null, COALESCE returns a null value.

**IFNULL**

IFNULL substitutes a specified value if an expression evaluates as null. If the expression is not null, IFNULL returns the value of the expression.

Scalar Sub-query Expressions

**Description**

A scalar sub-query expression, denoted by grammar symbol `scalar_subquery_expr`, is a query_expression that returns only one value, that is, a value for one row and one column. See Query Expressions (page 30) for more information on query_expression.

Scalar sub-query expressions can be specified in select lists, search conditions and arguments of scalar functions.

**Examples**

- Scalar sub-query in a select list
SELECT e.ename, (SELECT d.dname
FROM dept d
WHERE d.deptno = e.deptno)
FROM emp e;

- Scalar sub-query in a search condition

SELECT ename
FROM emp
WHERE (SELECT MAX(deptno)
FROM dept) = deptno;

- Scalar sub-query as an argument to a scalar function.

SELECT e.ename, LEN ((SELECT d.dname
from dept d
WHERE d.deptno = e.deptno))
FROM emp e;

**Escape field name matching reserved words in c-tree expressions**

In V11.5 and later, it is possible to escape the field names by wrapping them in square brackets, thus identifying them directly as fields so they do not conflict with function names.

This is useful if you have a field name that is the same as a function name. For example, if you have a field named "year" and you want to set a filter on year being 2000, so the filter string would be "year = 2000". The parser would fail because it identified "year" as a function with the wrong syntax.

The new syntax would be "[year] = 2000", which the parser unambiguously interprets as a field named "year".

The parser allows the content of "[]" to be composed by any character excluding "]", space, tabs, newline, and linefeed.

### 1.5 Literals

Literals are a type of expression that specify a constant value (they are also called constants). You can specify literals wherever c-treeACE SQL syntax allows expressions. Some c-treeACE SQL constructs allow literals but prohibit other forms of expressions.

There are three types of literals:
- Numeric
- Character string
- Date-time

The following sections discuss each type of literal.
**Numeric Literals**

A numeric literal is a string of digits that c-treeACE SQL interprets as a decimal number. c-treeACE SQL allows the string to be in a variety of formats, including scientific notation.

**Syntax**

```
(+|-){[0-9][0-9]...} [.][0-9][0-9]...[E|e][+-][0-9][0-9]}
```

**Examples**

The following are all valid numeric strings:

123
123.456
-123.456
12.34E-04

**Character String Literals**

A character string literal is a string of characters enclosed in single quotation marks (').

To include a single quotation mark in a character-string literal, precede it with an additional single quotation mark. The following c-treeACE SQL examples demonstrate embedding quotation marks in character-string literals:

```sql
insert into quote values('unquoted literal');
insert into quote values('''single-quoted literal''');
insert into quote values("double-quoted literal");
insert into quote values('O''Hare');
select * from quote;
```

```
c1 unquoted literal
'single-quoted literal'
"double-quoted literal"
O'Hare
```

To insert a character-string literal that spans multiple lines, enclose each line in single quotation marks. The following c-treeACE SQL examples shows this syntax, as well as embedding quotation marks in one of the lines:

```sql
insert into quote2 values ('Here''s a very long character string
   'literal that will not fit on a single line.');
1 record inserted.
select * from quote2;
```

```
C1
--
Here's a very long character string literal that will not fit on a single line.
```

**Date-Time Literals**

SQL supports special formats for literals to be used in conjunction with date-time data types. Basic predicates and the VALUES clause of INSERT statements can specify date literals directly for comparison and insertion into tables. In other cases, you need to convert date literals to the appropriate date-time data type with the CAST, CONVERT, or TO_DATE scalar functions.

Enclose date-time literals in single quotation marks.
Date Literals

Date literals specify a day, month, and year. By default, c-treeACE SQL supports any of the following formats, enclosed in single quotation marks (’).

Syntax

date-literal ::= {d 'yyyy-mm-dd'}
| mm-dd-yyyy
| mm/dd/yyyy
| yyyy-mm-dd
| yyyy/mm/dd
| dd-mon-yyyy
| dd/month/yyyy

Arguments

{d 'yyyy-mm-dd'}

A date literal enclosed in an escape clause compatible with ODBC. Precede the literal string with an open brace ( { ) and a lowercase d. End the literal with a close brace. For example:

```
INSERT INTO DTEST VALUES ({d '1994-05-07'});
```

If you use the ODBC escape clause, you must specify the date using the format yyyy-mm-dd.

dd

The day of month as a 1- or 2-digit number (in the range 01-31).

mm

The month value as a 1- or 2-digit number (in the range 01-12).

mon

The first three characters of the name of the month (in the range 'JAN' to 'DEC').

yyy

The year as four-digit number. By default, c-treeACE SQL generates an Invalid date string error if the year is specified as anything other than digits.

Examples

The following c-treeACE SQL examples demonstrate some supported formats for date literals:

```
CREATE TABLE T2 (C1 DATE, C2 TIME);
INSERT INTO T2 (C1) VALUES('5/7/56');
INSERT INTO T2 (C1) VALUES('7/MAY/1956');
INSERT INTO T2 (C1) VALUES('1956/05/07');
INSERT INTO T2 (C1) VALUES({d '1956-05-07'});
INSERT INTO T2 (C1) VALUES('29-sep-1952');
SELECT C1 FROM T2;
```

<table>
<thead>
<tr>
<th>c1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1956-05-07</td>
</tr>
<tr>
<td>1956-05-07</td>
</tr>
<tr>
<td>1956-05-07</td>
</tr>
<tr>
<td>1956-05-07</td>
</tr>
<tr>
<td>1952-09-29</td>
</tr>
</tbody>
</table>
**Time Literals**

Time literals specify an hour, minute, second, and millisecond, using the following format, enclosed in single quotation marks (’):

**Syntax**

time-literal ::=  
  {t 'hh:mi:ss'}  
  |  hh:mi:ss

**Arguments**

{t 'hh:mi:ss'}

A time literal enclosed in an escape clause compatible with ODBC. Precede the literal string with an open brace ( { ) and a lowercase t. End the literal with a close brace. For example:

```
INSERT INTO TTEST VALUES ({t '23:22:12'})
```

If you use the ODBC escape clause, you must specify the time using the format **hh:mi:ss**.

**hh**

The hour value as a 1- or 2-digit number (in the range 00 to 23).

**mi**

The minute value as a 1- or 2-digit number (in the range 00 to 59).

**ss**

The seconds value as a 1- or 2-digit number (in the range 00 to 59).

**Examples**

The following c-treeACE SQL examples show some of the formats c-treeACE SQL will and will not accept for time literals:

```
INSERT INTO T2 (C2) VALUES('3');
error(-20234): Invalid time string

INSERT INTO T2 (C2) VALUES('8:30');
error(-20234): Invalid time string

INSERT INTO T2 (C2) VALUES('8:30:1');
INSERT INTO T2 (C2) VALUES('8:30:1');
error(-20234): Invalid time string

INSERT INTO T2 (C2) VALUES('8:30:00');
INSERT INTO T2 (C2) VALUES('8:30:01');
INSERT INTO T2 (C2) VALUES({t'8:30:01'});
SELECT C2 FROM T2;
c2
  08:30:01  
  08:30:00  
  08:30:01  
  08:30:01
```
**Timestamp Literals**

Timestamp literals specify a date and a time separated by a space, enclosed in single quotation marks ('):

**Syntax**

\{ts 'yyyy-mm-dd hh:mi:ss'}

|    | date-literal time-literal ' |

**Arguments**

\{ts 'yyyy-mm-dd hh:mi:ss'}

A timestamp literal enclosed in an escape clause compatible with ODBC. Precede the literal string with an open brace ( { ) and a lowercase ts. End the literal with a close brace. For example:

```sql
INSERT INTO DTEST
VALUES ((ts '1956-05-07 10:41:37'))
```

If you use the ODBC escape clause, you must specify the timestamp using the format `yyyy-mm-dd hh:mi:ss`.

**date-literal**

A date literal.

**time-literal**

A time literal.

**Example**

```sql
SELECT * FROM DTEST WHERE C1 = (ts '1956-05-07 10:41:37')
```

1.6 **Query Expressions**

**Description**

A query expression selects the specified column values from one or more rows contained in one or more tables specified in the **FROM** clause. The selection of rows is restricted by a search condition in the **WHERE** clause. The temporary table derived through the clauses of a **SELECT** statement is called a **result table**.

Query expressions form the basis of other SQL statements and syntax elements:

- **SELECT** statements are query expressions with optional **ORDER BY** and **FOR UPDATE** clauses.
- **CREATE VIEW** statements specify their result table as a query expression.
- **INSERT** statements can specify a query expression to add the rows of the result table to a table.
- **UPDATE** statements can specify a query expression that returns a single row to modify columns of a row.
- Some search conditions can specify query expressions. Basic predicates can specify query expressions, however, the result table can contain only a single value. Quantified and **IN**
predicates can specify query expressions, however, the result table can contain only a single column.

- The FROM clause of a query expression can itself specify a query expression, called a derived table.

**Syntax**

```sql
query_expression ::=     
  WITH [ RECURSIVE ] with_list     
  |  query_specification     
  |  query_expression  set_operator query_expression     
  |  ( query_expression )

set_operator ::=     
  {  UNION  [ ALL ]  |  INTERSECT  |  MINUS }

with_list ::=     
  with_list_element     
  |  with_list_element , with_list

with_list_element ::=     
  query_name tmptbl_column_commalist AS ( query_expr )

query_name ::=     
  tblname

tmptbl_column_commalist ::=     
  [ , [table.]column_name ]

query_specification ::=     
  SELECT [ALL | DISTINCT] [SKIP N] [TOP N]     
  {     
    *     
    |  { table_name | alias } . * [, { table_name | alias } . * ] ...     
    |  { { expr | NULL } [ AS ] column_title [ ' ] }     
    |  { [ ' ] column_title [ ' ] = ] { expr | NULL } }     
  }

FROM table_ref [ { ctree ORDERED } ] [, table_ref [ { ctree ORDERED } ] ] ...

WHERE search_condition

GROUP BY [table.]column_name     
[, [table.]column_name] ...

HAVING search_condition

table_ref ::=     
  table_name [ AS ] [ alias [ { column_alias , ... } ] ]     
  |  ( query_expression ) [ AS ] alias [ { column_alias , ... } ]     
  |  [ ] joined_table [ ]

joined_table ::=     
  table_ref CROSS JOIN table_ref
```
Arguments

WITH [ RECURSIVE ] with_list

RECURSIVE specifies query_expr in with_list_element is a recursive query.

SELECT [ ALL | DISTINCT ]

DISTINCT specifies that the result table omits duplicate rows. ALL is the default, and specifies that the result table includes all rows.

SELECT [ SKIP N ]

SKIP skips the leading rows in result sets. N specifies the number of rows to be skipped.

SELECT [ TOP N ]

TOP limits the number of rows in result sets. N specifies the number of rows to be returned. When used in conjunction with SKIP, rows are first skipped, then the TOP N rows remaining are returned.

SELECT * | { table_name | alias } . *

Specifies that the result table includes all columns from all tables named in the FROM clause. For instance, the following examples both specify all the columns in the customers table:

```
SELECT * FROM customers;
SELECT customers.* FROM customers;
```

The tablename.* syntax is useful when the select list refers to columns in multiple tables, and you want to specify all the columns in one of those tables:

```
SELECT CUSTOMERS.CUSTOMER_ID, CUSTOMERS.CUSTOMER_NAME, ORDERS.*
FROM CUSTOMERS, ORDERS ...
```

SELECT { { { expr | NULL } [ AS ] [ ''] column_title [''] ] } | { [ ''] column_title [''] = } { expr | NULL } } }

Specifies a list of expressions, called a select list, whose results will form columns of the result table. Typically, the expression is a column name from a table named in the FROM clause. The expression can also be any supported mathematical expression, scalar function, aggregate function, or scalar sub-query that returns one value.

The example for an expr which is a scalar sub-query is as follows.

```
SELECT e.ename,
    { SELECT d.dname FROM dept d where d.deptno = e.deptno }
FROM emp e;
```

The optional 'column_title' argument specifies a new heading for the associated column in the result table. Enclose the new title in single or double quotation marks if it contains spaces or other special characters:

```
SELECT order_value, order_value * .2 AS 'order "markup"' FROM orders;
```
You can qualify column names with the name of the table they belong to:

```
SELECT CUSTOMER.CUSTOMER_ID FROM CUSTOMERS
```

You must qualify a column name if it occurs in more than one table specified in the `FROM` clause:

```
SELECT CUSTOMERS.CUSTOMER_ID
FROM CUSTOMERS, ORDERS
```

Qualified column names are always allowed even when they are not required.

**Note:** You should not use `column_title` elsewhere in the query such as `WHERE`, `GROUP BY` and `HAVING` clauses.

### FROM table_ref ...

Specifies optional `FROM` clause. Without the `FROM` clause, the select list can contain all possible expressions that do not have references to columns of tables. For example, the select list can contain constants, scalar functions, aggregate functions, etc. Other clauses such as `WHERE` and `ORDER BY` and set operators such as `UNION` can also be used without the `FROM` clause.

**Examples**

**SELECT statement with constant expression:**

```
SELECT 10;
10
```

**SELECT statement with scalar function:**

```
SELECT abs(-10);
10
```

**SELECT statement with UNION set operator:**

```
SELECT 10 UNION SELECT 20;
10
20
```

If a `FROM` clause is specified, it is used to specify one or more table references. Each table reference resolves to one table (either a table stored in the database or a virtual table resulting from processing the table reference) whose rows the query expression uses to create the result table. There are three forms of table references:

- A direct reference to a table, view or synonym
- A derived table specified by a query expression in the `FROM` clause
- A joined table that combines rows and columns from multiple tables

The usage notes specific to each form of table reference follow.
If there are multiple table references, c-treeACE SQL joins the tables to form an intermediate result table that is used as the basis for evaluating all other clauses in the query expression. That intermediate result table is the Cartesian product of rows in the tables in the FROM clause, formed by concatenating every row of every table with all other rows in all tables.

**FROM table_name [ AS ] [ alias ]**

Explicitly names a table. The name listed in the FROM clause can be a table name, a view name, or a synonym.

*alias* is a name used to qualify column names in other parts of the query expression. Aliases are also called correlation names.

If you specify an alias, you must use it, and not the table name, to qualify column names that refer to the table. Query expressions that join a table with itself must use aliases to distinguish between references to column names.

For example, the following query expression joins the table customer with itself. It uses the aliases x and y and returns information on customers in the same city as customer ‘SMITH’:

```sql
SELECT y.cust_no, y.name
FROM customer x, customer y
WHERE x.name = 'SMITH'
  AND y.city = x.city ;
```

Similar to table aliases, the *column_alias* provides an alternative name to use in column references elsewhere in the query expression. If you specify column aliases, you must specify them for all the columns in *table_name*. Also, if you specify column aliases in the FROM clause, you must use them, and not the column names, in references to the columns.

**FROM ( query_expression ) [ AS ] alias [( column_alias [ , … ] )]**

Specifies a derived table through a query expression. With derived tables, you must specify an alias to identify the derived table.

Derived tables can also specify column aliases. Column aliases provides an alternative name to use in column references elsewhere in the query expression. If you specify column aliases, you must specify them for all the columns in the result table of the query expression. Also, if you specify column aliases in the FROM clause, you must use them, and not the column names, in references to the columns.

**FROM [ ( ] joined_table [ ) ]**

Combines data from two table references by specifying a join condition. The syntax currently allowed in the FROM clause supports only a subset of possible join conditions:

- **CROSS JOIN** specifies a Cartesian product of rows in the two tables
- **INNER JOIN** specifies an inner join using the supplied search condition
- **LEFT OUTER JOIN** specifies a left outer join using the supplied search condition

You can also specify these and other join conditions in the WHERE clause of a query expression. See *Inner Joins* (page 37) and *Outer Joins* (page 39) for further details on both ways of specifying joins.
The \texttt{ctree ORDERED} clause directs the c-treeACE SQL optimizer to join tables in a specified order. Use this clause when you want to override the SQL engine's join-order optimization. This is useful for special cases when you know in fact a particular join order results in better performance. Since this clause bypasses join-order optimization, carefully test queries that use it to make sure your specified join order is faster than relying on the optimizer. In addition, future c-treeACE SQL releases will further improve join ordering, outperforming manually created orders.

Note that the braces ( \{ and \} ) are part of the required syntax.

\begin{verbatim}
SELECT sc.tbl 'Table', sc.col 'Column',
       sc.coltype 'Data Type', sc.width 'Size'
FROM admin.syscolumns sc, admin.systables st
{ ctree ORDERED }
WHERE sc.tbl = st.tbl AND st.tbltype = 'S'
ORDER BY sc.tbl, sc.col;
\end{verbatim}

\textbf{WHERE search\_condition}

The \texttt{WHERE} clause specifies a \texttt{search\_condition} that applies conditions to restrict the number of rows in the result table. If the query expression does not specify a \texttt{WHERE} clause, the result table includes all the rows of the specified table reference in the \texttt{FROM} clause.

The \texttt{search\_condition} is applied to each row of the result table set of the \texttt{FROM} clause. Only rows that satisfy the conditions become part of the result table. If the result of the \texttt{search\_condition} is \texttt{NULL} for a row, the row is not selected.

Search conditions can specify different conditions for joining two or more tables. See \textit{Inner Joins} (page 37) and \textit{Outer Joins} (page 39) for more information.

Refer to \textit{Search Conditions} (page 15) for details on the different kinds of search conditions.

\begin{verbatim}
SELECT *
FROM customer
WHERE city = 'COLUMBIA' AND state = 'MO';
SELECT *
FROM customer
WHERE city IN (
    SELECT city
    FROM customer
    WHERE name = 'SMITH');
\end{verbatim}

\textbf{GROUP BY column\_name ...}

Specifies grouping of rows in the result table:

- For the first column specified in the \texttt{GROUP BY} clause, c-treeACE SQL arranges rows of the result table into groups whose rows all have the same values for the specified column.
- If a second \texttt{GROUP BY} column is specified, c-treeACE SQL then groups rows in each main group by values of the second column.
- SQL groups rows for values in additional \texttt{GROUP BY} columns in a similar fashion.
All columns named in the \texttt{GROUP BY} clause must also be in the select list of the query expression. Conversely, columns in the select list must also be in the \texttt{GROUP BY} clause or be part of an aggregate function.

\textbf{Note:} Prior to version 10.0 of c-treeACE SQL, rows in result tables were in ascending order of the \texttt{GROUP BY} columns. Subsequent use of a hashing scheme to group rows, results in the rows in the result table may not be in any specific order. Any explicit ordering must be defined by the SQL statement.

\textbf{HAVING search\_condition}

The \texttt{HAVING} clause allows conditions to be set on the groups returned by the \texttt{GROUP BY} clause. If the \texttt{HAVING} clause is used without the \texttt{GROUP BY} clause, the implicit group against which the search condition is evaluated is all the rows returned by the \texttt{WHERE} clause.

A condition of the \texttt{HAVING} clause can compare one aggregate function value with another aggregate function value or a constant.

\begin{verbatim}
-- select customer number and number of orders for all
-- customers who had more than 10 orders prior to
SELECT cust_no, count(*)
  FROM orders
  WHERE order_date < to_date ('3/31/1991')
  GROUP BY cust_no
  HAVING count (*) > 10 ;
\end{verbatim}

\textbf{UNION [ALL]}

Appends the result table from one query expression to the result table from another.

The two query expressions must have the same number of columns in their result table, and those columns must have the same or compatible data types.

The final result table contains the rows from the second query expression appended to the rows from the first. By default, the result table does not contain any duplicate rows from the second query expression. Specify \texttt{UNION ALL} to include duplicate rows in the result table.

\begin{verbatim}
-- Get a merged list of customers and suppliers.
  SELECT name, street, state, zip
  FROM customer
UNION
  SELECT name, street, state, zip
  FROM supplier ;
-- Get a list of customers and suppliers
-- with duplicate entries for those customers who are
-- also suppliers.
  SELECT name, street, state, zip
  FROM customer
UNION ALL
  SELECT name, street, state, zip
  FROM supplier ;
\end{verbatim}

\textbf{INTERSECT}

Limits rows in the final result table to those that exist in the result tables from both query expressions.
The two query expressions must have the same number of columns in their result table, and those columns must have the same or compatible data types.

-- Get a list of customers who are also suppliers.
SELECT name, street, state, zip
FROM customer
INTERSECT
SELECT name, street, state, zip
FROM supplier;

MINUS

Limits rows in the final result table to those that exist in the result table from the first query expression minus those that exist in the second. In other words, the MINUS operator returns rows that exist in the result table from the first query expression but that do not exist in the second.

The two query expressions must have the same number of columns in their result table, and those columns must have the same or compatible data types.

-- Get a list of suppliers who are not customers.
SELECT name, street, state, zip
FROM supplier;
MINUS
SELECT name, street, state, zip
FROM customer;

Authorization

The user executing a query expression must have any of the following privileges:

- DBA privilege
- SELECT permission on all the tables/views referred to in the query_expression.

1.7 Inner Joins

Description

Inner joins specify how the rows from one table reference are to be joined with the rows of another table reference. Inner joins usually specify a search condition that limits the number of rows from each table reference that become part of the result table generated by the inner join operation.

If an inner join does not specify a search condition, the result table from the join operation is the Cartesian product of rows in the tables, formed by concatenating every row of one table with every row of the other table. Cartesian products (also called cross products or cross joins) are not
practically useful, however, c-treeACE SQL logically processes all join operations by first forming
the Cartesian products of rows from tables participating in the join.

If specified, the search condition is applied to the Cartesian product of rows from the two tables. Only rows that satisfy the search condition become part of the result table generated by the join.

A query expression can specify inner joins in either its FROM clause or in its WHERE clause. For each formulation in the FROM clause, there is an equivalent syntax formulation in the WHERE clause. Currently, not all syntax specified by the SQL-92 standard is allowed in the FROM clause.

**Syntax**

```
from_clause_inner_join ::
  |  FROM table_ref CROSS JOIN table_ref
  |  FROM table_ref [ INNER ] JOIN table_ref ON search_condition

where_clause_inner_join ::
  FROM table_ref, table_ref WHERE search_condition
```

**Arguments**

**FROM table_ref CROSS JOIN table_ref**

Explicitly specifies that the join generates the Cartesian product of rows in the two table references. This syntax is equivalent to omitting the WHERE clause and a search condition. The following queries illustrate the results of a simple CROSS JOIN operation and an equivalent formulation that does not use the CROSS JOIN syntax:

```sql
SELECT * FROM T1; -- Contents of T1
  C1  C2
--  --
10  15
20  25
2 records selected
SELECT * FROM T2; -- Contents of T2
  C3  C4
--  --
10 BB
15 DD
2 records selected
SELECT * FROM T1 CROSS JOIN T2; -- Cartesian product
  C1  C2  C3  C4
--  --  --  --
10  15  10 BB
10  15  15 DD
20  25  10 BB
20  25  15 DD
4 records selected
SELECT * FROM T1, T2; -- Different formulation, same results
  C1  C2  C3  C4
--  --  --  --
10  15  10 BB
10  15  15 DD
20  25  10 BB
20  25  15 DD
4 records selected
```

**FROM table_ref [ INNER ] JOIN table_ref ON search_condition**

FROM table_ref, table_ref WHERE search_condition
These two equivalent syntax constructions both specify `search_condition` for restricting rows that will be in the result table generated by the join. In the first format, `INNER` is optional and has no effect. There is no difference between the `WHERE` form of inner joins and the `JOIN ON` form.

**Equi-joins**
An equi-join specifies that values in one table equal some corresponding column's values in the other:

```sql
-- For customers with orders, get their name and order info, :
SELECT customer.cust_no, customer.name,
       orders.order_no, orders.order_date
FROM customers INNER JOIN orders
     ON customer.cust_no = orders.cust_no ;
-- Different formulation, same results:
SELECT customer.cust_no, customer.name,
       orders.order_no, orders.order_date
FROM customers, orders
     WHERE customer.cust_no = orders.cust_no ;
```

**Self joins**
A self join, or auto join, joins a table with itself. If a `WHERE` clause specifies a self join, the `FROM` clause must use aliases to have two different references to the same table:

```sql
-- Get all the customers who are from the same city as customer SMITH:
SELECT y.cust_no, y.name
FROM customer AS x INNER JOIN customer AS y
     ON x.name = 'SMITH' AND y.city = x.city ;
-- Different formulation, same results:
SELECT y.cust_no, y.name
FROM customer x, customer y
     WHERE x.name = 'SMITH' AND y.city = x.city ;
```

### 1.8 Outer Joins

**Description**
An outer join between two tables returns more information than a corresponding inner join. An outer join returns a result table that contains all the rows from one of the tables even if there is no row in the other table that satisfies the join condition.

In a left outer join, the information from the table on the left is preserved: the result table contains all rows from the left table even if some rows do not have matching rows in the right table. Where there are no matching rows in the left table, c-treeACE SQL generates null values.

In a right outer join, the information from the table on the right is preserved: the result table contains all rows from the right table even if some rows do not have matching rows in the left table. Where there are no matching rows in the right table, c-treeACE SQL generates null values.

c-treeACE SQL supports two forms of syntax to support outer joins:

- In the `FROM` clause, specify one of the outer join clauses between two table names, followed by a search condition:
  ```sql
  LEFT OUTER JOIN
  RIGHT OUTER JOIN
  ```
The search condition can contain only the join condition between the specified tables.

- In the WHERE clause of a query expression, specify the outer join operator (+) after the column name of the table for which rows will not be preserved in the result table. Both sides of an outer-join search condition in a WHERE clause must be simple column references. This syntax allows both left and right outer joins.

**c-treeACE SQL does not support full (two-sided) outer joins.**

**Syntax**

```
from_clause_inner_join ::
    FROM table_ref LEFT OUTER JOIN table_ref ON search_condition
    | FROM table_ref RIGHT OUTER JOIN table_ref ON search_condition

where_clause_inner_join ::
    WHERE [table_name.]column (+) = [table_name.]column
    | WHERE [table_name.]column = [table_name.]column (+)
```

**Examples**

The following example shows a left outer join. It displays all the customers with their orders. Even if there is not a corresponding row in the orders table for each row in the customer table, NULL values are displayed for the `orders.order_no` and `orders.order_date` columns.

```
SELECT customer.cust_no, customer.name, orders.order_no, orders.order_date
FROM customers, orders
WHERE customer.cust_no = orders.cust_no (+);
```

The following series of examples illustrates the outer join syntax:

```
-- Left outer join
SELECT * FROM T1 LEFT OUTER JOIN T2 ON T1.C1 = T2.C3;
```

```
C1   C2   C3   C4
--   --   --   --
10   15   10   BB
20   25
2 records selected
```

```
-- Left outer join: different formulation, same results
SELECT * FROM T1, T2 WHERE T1.C1 = T2.C3 (+); 
```

```
C1   C2   C3   C4
--   --   --   --
10   15   10   BB
20   25
2 records selected
```
1.9 Date-Time Format Strings

The TO_CHAR scalar function supports a variety of format strings to control the output of date and time values. The format strings consist of keywords that c-treeACE SQL interprets and replaces with formatted values.

The format strings are case sensitive. For instance, c-treeACE SQL replaces 'DAY' with all uppercase letters, but follows the case of 'Day'.

Supply the format strings, enclosed in single quotation marks, as the second argument to the TO_CHAR function.

Example

```sql
SELECT C1 FROM T2;
C1
--
09/29/1952
1 record selected

SELECT TO_CHAR(C1, 'Day, Month ddth'), TO_CHAR(C2, 'HH12 a.m.') FROM T2;
TO_CHAR(C1,DAY, MONTH DDTH)  TO_CHAR(C2,HH12 A.M.)
---------------------------  ---------------------
Monday   , September 29th    02 p.m.
1 record selected
```

For details of the TO_CHAR function, see "TO_CHAR".

Date Format Strings

A date format string can contain any of the following format keywords along with other characters. The format keywords in the format string are replaced by corresponding values to get the result. The other characters are displayed as literals.

- **CC** The century as a two digit number.
YYYY  The year as a four digit number.
YYY   The last three digits of the year.
YY    The last two digits of the year.
Y     The last digit of the year.
Y,YYY The year as a four digit number with a comma after the first digit.
Q     The quarter of the year as one digit number (with values one, two, three, or four).
MM    The month value as two digit number (in the range 01-12).
MONTH The name of the month as a string of nine characters ('JANUARY' to 'DECEMBER').
MON   The first three characters of the name of the month (in the range 'JAN' to 'DEC').
WW    The week of year as a two digit number (in the range 01-52).
W     The week of month as a one digit number (in the range 1-5).
DDD   The day of year as a three digit number (in the range 001-365).
DD    The day of month as a two digit number (in the range 01-31).
D     The day of week as a one digit number (in the range 1-7, 1 for Sunday and 7 for Saturday).
DAY   The day of week as a nine character string (in the range ‘SUNDAY’ to ‘SATURDAY’).
DY    The day of week as a three character string (in the range ‘SUN’ to ‘SAT’).
J     The Julian day (number of days since DEC 31, 1899) as an eight digit number.
TH    When added to a format keyword that results in a number, this format keyword ('TH') is replaced by the string 'ST', 'ND', 'RD' or 'TH' depending on the last digit of the number.

Example

```
SELECT C1 FROM T2;
C1
--
09/29/1952
1 record selected
SELECT TO_CHAR(C1, 'Day, Month ddth'),
       TO_CHAR(C2, 'HH12 a.m.') FROM T2;
TO_CHAR(C1,DAY, MONTH DDTH)  TO_CHAR(C2,HH12 A.M.)
------------------------------- ---------------------
Monday   , September 29th    02 p.m.
1 record selected
```

Time Format Strings

A time format string can contain any of the following format keywords along with other characters. The format keywords in the format string are replaced by corresponding values to get the result. The other characters are displayed as literals.

AM     The string "AM" or "PM" depending on whether time corresponds to forenoon or afternoon.
PM     The string "AM" or "PM" depending on whether time corresponds to forenoon or afternoon.
A.M.   The string "A.M." or "P.M." depending on whether time corresponds to forenoon or afternoon.
P.M.  The string "A.M." or "P.M." depending on whether time corresponds to forenoon or afternoon.

HH12  The hour value as a two-digit number (in the range 00 to 11).

HH24  The hour value as a two-digit number (in the range 00 to 23).

HH    The hour value as a two-digit number (in the range 00 to 23).

MI     The minute value as a two-digit number (in the range 00 to 59).

SS     The seconds value as a two-digit number (in the range 00 to 59).

SSSSS  The seconds from midnight as a five-digit number (in the range 00000 to 86399).

Example

SELECT C1 FROM T2;
C1
--
09/29/1952
1 record selected

SELECT TO_CHAR(C1, 'Day, Month ddth'),
       TO_CHAR(C2, 'HH12 a.m.') FROM T2;
TO_CHAR(C1,DAY, MONTH DDTH)  TO_CHAR(C2,HH12 A.M.)
---------------------------  ---------------------
Monday   , September 29th    02 p.m.
1 record selected

1.10  Functions

Functions are a type of c-treeACE SQL expression that return a value based on the argument they are supplied. c-treeACE SQL supports two types of functions:

- **Aggregate functions** calculate a single value for a collection of rows in a result table (if the function is in a statement with a GROUP BY clause, it returns a value for each group in the result table). Aggregate functions are also called set or statistical functions. Aggregate functions cannot be nested.

- **Scalar functions** calculate a value based on another single value. Scalar functions are also called value functions. Scalar functions can be nested.

**Aggregate Functions**
**AVG**

**Syntax**

```sql
AVG ( { [ALL] expression } | { DISTINCT column_ref } )
```

**Description**

The aggregate function `AVG` computes the average of a collection of values. The keyword `DISTINCT` specifies that the duplicate values are to be eliminated before computing the average.

- Null values are eliminated before the average value is computed. If all the values are null, the result is null.
- The argument to the function must be of type `SMALLINT`, `INTEGER`, `NUMERIC`, `REAL` or `FLOAT`.
- The result is of type `NUMERIC`.

**Example**

```sql
SELECT AVG (salary)
FROM employee
WHERE deptno = 20;
```
COUNT

Syntax
COUNT ( { [ALL] expression } | { DISTINCT column_ref } | * )

Description
The aggregate function COUNT computes either the number of rows in a group of rows or the number of non-null values in a group of values.

- The keyword DISTINCT specifies that the duplicate values are to be eliminated before computing the count.
- If the argument to COUNT function is '*', then the function computes the count of the number of rows in group.

For fixed length files, an exact row count is stored in the file header and can be immediately returned. For variable length files, an index is required that returns the number of keys. The first index that is found from the following is chosen: RECBYT, ROWID, first unique index, first duplicate index. If no index is available, then a physical table scan is performed to count the actual number of rows in the table.

- If the argument to COUNT function is not '*', then null values are eliminated before the number of rows is computed.
- The argument column_ref or expression can be of any type.
- The result of the function is of INTEGER type. The result is never null.

Example
SELECT COUNT (*)
FROM orders
WHERE order_date = SYSDATE ;
COUNT_BIG

To resolve the problem of counting rows when they are more than $2^{31} - 1$, a new aggregate function COUNT_BIG has been added in V11.5 and later. This function is like COUNT (page 45) except that it returns a BIGINT instead of an INTEGER.
MAX

Syntax
MAX ( { [ALL] expression } | { DISTINCT column_ref } )

Description
The aggregate function MAX returns the maximum value in a group of values.

- The specification of DISTINCT has no effect on the result.
- The result of the function is of the same data type as that of the argument.
- The argument column_ref or expression can be of any type.
- The result is null if the result set is empty or contains only null values.

Example
SELECT order_date, product, MAX (qty)
FROM orders
GROUP BY order_date, product ;
**MIN**

Syntax

```sql
MIN ( { [ALL] expression } | { DISTINCT column_ref } )
```

Description

The aggregate function `MIN` returns the minimum value in a group of values.

- The specification of `DISTINCT` has no effect on the result.
- The argument `column_ref` or expression can be of any type.
- The result of the function is of the same data type as that of the argument.
- The result is null if the result set is empty or contains only null values.

Example

```sql
SELECT MIN (salary)
FROM employee
WHERE deptno = 20 ;
```
**SUM**

**Syntax**

```
SUM ( { [ALL] expression } | { DISTINCT column_ref } )
```

**Description**

The aggregate function `SUM` returns the sum of the values in a group. The keyword `DISTINCT` specifies that the duplicate values are to be eliminated before computing the sum.

- The argument `column_ref` or expression can be of any numeric type.
- The result of the function is of the same data type as that of the argument except that the result is of type `INTEGER` when the argument is of type `SMALLINT` or `TINYINT`.
- The result can have a null value.

**Example**

```
SELECT SUM (amount)
FROM orders
WHERE order_date = SYSDATE;
```

**Scalar Functions**

c-treeACE SQL does not provide a mechanism for calling a scalar function directly from a user-defined function. To use scalar functions, it is necessary to call the function from within a c-treeACE SQL statement. c-treeACE SQL defines the special table `SYSCALCTABLE` which has only one row for use in situations where all the data is in the inputs.
ABS function (ODBC compatible)

Syntax
ABS ( expression )

Description
The scalar function ABS computes the absolute value of expression.

Example
SELECT  ABS (MONTHS_BETWEEN (SYSDATE, order_date))
        FROM  orders
        WHERE   ABS (MONTHS_BETWEEN (SYSDATE, order_date)) > 3 ;

Notes
- The argument to the function must be of type TINYINT, SMALLINT, INTEGER, NUMERIC, REAL or FLOAT.
- The result is of type NUMERIC.
- If the argument expression evaluates to null, the result is null.
ACOS function (ODBC compatible)

Syntax
ACOS (expression)

Description
The scalar function ACOS returns the arccosine of expression.

Example
select acos (.5) 'Arccosine in radians' from syscalctable;
ARCCOSINE IN RAD
----------------
1.047197551196598
1 record selected

select acos (.5) * (180/ pi()) 'Arccosine in degrees' from syscalctable;
ARCCOSINE IN DEG
----------------
59.999999999999993
1 record selected

Notes
ACOS takes the ratio (expression) of two sides of a right triangle and returns the corresponding angle. The ratio is the length of the side adjacent to the angle divided by the length of the hypotenuse. The result is expressed in radians and is in the range -Pi/2 to Pi/2 radians. To convert degrees to radians, multiply degrees by Pi/180. To convert radians to degrees, multiply radians by 180/Pi.

- Expression must be in the range -1 to 1.
- Expression must evaluate to an approximate numeric data type.
**ADD_MONTHS function (extension)**

**Syntax**

```
ADD_MONTHS ( date_expression, integer_expression )
```

**Description**

The scalar function `ADD_MONTHS` adds to the date value specified by the `date_expression`, the given number of months specified by `integer_expression` and returns the resultant date value.

**Example**

```
SELECT *
FROM   customer
WHERE  ADD_MONTHS (start_date, 6) > SYSDATE ;
```

**Notes**

- The first argument must be of `DATE` type.
- The second argument to the function must be of numeric type.
- The result is of type `DATE`.
- If any of the arguments evaluate to null, the result is null.
ASCII function (ODBC compatible)

Syntax
ASCII ( char_expression )

Description
The scalar function ASCII returns the ASCII value of the first character of the given character expression.

Example
SELECT ASCII ( zip )
    FROM   customer ;

Notes
- The argument to the function must be of character type.
- The result is of type INTEGER.
- If the argument char_expression evaluates to null, the result is null.
**ASIN function (ODBC compatible)**

**Syntax**

ASIN ( expression )

**Description**

The scalar function **ASIN** returns the arcsine of **expression**.

**Example**

```sql
select asin (1) * (180/ pi()) 'Arcsine in degrees' from syscalctable;

ARCSINE IN DEGREES
--------------------
90.000000000000000
1 record selected

select asin (1) 'Arcsine in radians' from syscalctable;

ARCSINE IN RADIANS
-------------------
1.570796326794897
1 record selected
```

**Notes**

**ASIN** takes the ratio (**expression**) of two sides of a right triangle and returns the corresponding angle. The ratio is the length of the side opposite the angle divided by the length of the hypotenuse.

The result is expressed in radians and is in the range -Pi/2 to Pi/2 radians. To convert degrees to radians, multiply degrees by Pi/180. To convert radians to degrees, multiply radians by 180/Pi.

- Expression must be in the range -1 to 1.
- Expression must evaluate to an approximate numeric data type.
**ATAN function (ODBC compatible)**

**Syntax**

ATAN ( expression )

**Description**

The scalar function **ATAN** returns the arctangent of expression.

**Example**

```sql
select atan (1) * (180/ pi()) 'Arctangent in degrees' from syscalctable;
```

```
ARCTANGENT IN DE
----------------
45.000000000000000
1 record selected
```

```sql
select atan (1) 'Arctangent in radians' from syscalctable;
```

```
ARCTANGENT IN RA
----------------
0.785398163397448
1 record selected
```

**Notes**

**ATAN** takes the ratio (expression) of two sides of a right triangle and returns the corresponding angle. The ratio is the length of the side opposite the angle divided by the length of the side adjacent to the angle.

The result is expressed in radians and is in the range -\( \pi/2 \) to \( \pi/2 \) radians. To convert degrees to radians, multiply degrees by \( \pi/180 \). To convert radians to degrees, multiply radians by \( 180/\pi \).

- Expression must be in the range -1 to 1.
- Expression must evaluate to an approximate numeric data type.
**ATAN2 function (ODBC compatible)**

**Syntax**

```
ATAN2 ( expression1 , expression2 )
```

**Description**

The scalar function ATAN2 returns the arctangent of the x and y coordinates specified by `expression1` and `expression2`.

**Example**

```
select atan2 (1,1) * (180/ pi()) 'Arctangent in degrees' from syscalctable;
```

```
ARCTANGENT IN DE
----------------
45.000000000000000
1 record selected
```

```
select atan2 (1,1) 'Arctangent in radians' from syscalctable;
```

```
ARCTANGENT IN RA
----------------
0.785398163397448
1 record selected
```

**Notes**

ATAN2 takes the ratio of two sides of a right triangle and returns the corresponding angle. The ratio is the length of the side opposite the angle divided by the length of the side adjacent to the angle.

`expression1` and `expression2` specify the x and y coordinates of the end of the hypotenuse opposite the angle.

The result is expressed in radians and is in the range -Pi/2 to Pi/2 radians. To convert degrees to radians, multiply degrees by Pi/180. To convert radians to degrees, multiply radians by 180/Pi.

Both `expression1` and `expression2` must evaluate to approximate numeric data types.
CASE (SQL-92 Compatible)

Syntax

```
case-expr::
searched-case-expr | simple-case-expr

searched-case-expr::
CASE
  WHEN search_condition THEN { result-expr | NULL }
  [ ... ]
  [ ELSE expr | NULL ]
END

simple-case-expr::
CASE primary-expr
  WHEN expr THEN { result-expr | NULL }
  [ ... ]
  [ ELSE expr | NULL ]
END
```

Description

The `CASE` scalar function is a type of conditional expression. (See Conditional Expressions (page 25) for more details and a summary of all the conditional expressions.)

The general form of the `CASE` scalar function specifies a series of search conditions and associated result expressions. It is called a searched case expression. c-treeSQL returns the value specified by the first result expression whose associated search condition evaluates as true. If none of the search conditions evaluate as true, the `CASE` expression returns a null value (or the value of some other default expression if the `CASE` expression includes the `ELSE` clause).

`CASE` also supports syntax for a shorthand notation, called a simple case expression, for evaluating whether one expression is equal to a series of other expressions.

Notes

- This function is not allowed in a GROUP BY clause
- Arguments to this function cannot be query expressions

Arguments

**CASE**

The `CASE` keyword alone, not followed by `primary-expr`, specifies a searched case expression. It must be followed by one or more `WHEN-THEN` clauses each that specify a search condition and corresponding expression.

```
WHEN search_condition THEN { result-expr | NULL }
```

`WHEN` clause for searched case expressions. c-treeSQL evaluates search condition. If `search_condition` evaluates as true, `CASE` returns the value specified by `result-expr` (or null, if the clause specifies `THEN NULL`).

If `search_condition` evaluates as false, c-treeACE SQL evaluates the next `WHEN-THEN` clause, if any, or the `ELSE` clause, if it is specified.
CASE primary-expr
The CASE keyword followed by an expression specifies a simple case expression. In a simple case expression, one or more WHEN-THEN clauses specify two expressions.

A simple case expression can always be expressed as a searched case expression. Consider the following general simple case expression:

```sql
CASE primary-expr
    WHEN expr1 THEN result-expr1
    WHEN expr2 THEN result-expr2
    ELSE expr3
END
```

The preceding simple case expression is equivalent to the following searched case expression:

```sql
CASE
    WHEN primary-expr = expr1 THEN result-expr1
    WHEN primary-expr = expr2 THEN result-expr2
    ELSE expr3
END
```

WHEN expr THEN { result-expr | NULL }

WHEN clause for simple case expressions. c-treeSQL evaluates expr and compares it with primary-expr specified in the CASE clause. If they are equal, CASE returns the value specified by result-expr (or null, if the clause specifies THEN NULL).

If expr is not equal to primary-expr, c-treeSQL evaluates the next WHEN-THEN clause, if any, or the ELSE clause, if it is specified.

[ ELSE { expr | NULL } ]

In both searched case expressions and simple case expressions, the ELSE clause specifies an optional expression whose value c-treeACE SQL returns if none of the conditions specified in WHEN-THEN clauses were satisfied. If the CASE expression omits the ELSE clause, it is the same as specifying ELSE NULL.

Examples
The following example shows a searched case expression that assigns a label denoting tables as system tables if they begin with the letters sys. Note that this example can not be reformulated as a simple case expression, since it specifies a relational operator other than =.

```sql
SELECT tbl,
    CASE
        WHEN tbl like 'sys%' THEN 'System Table'
        ELSE 'Not System table'
    END
FROM systables;
```

<table>
<thead>
<tr>
<th>TBL</th>
<th>SEARCHED_CASE(TBLSY-----</th>
</tr>
</thead>
<tbody>
<tr>
<td>systblspaces</td>
<td>System Table</td>
</tr>
<tr>
<td>systables</td>
<td>System Table</td>
</tr>
<tr>
<td>syscolumns</td>
<td>System Table</td>
</tr>
<tr>
<td>sysindexes</td>
<td>System Table</td>
</tr>
<tr>
<td>sydbauth</td>
<td>System Table</td>
</tr>
<tr>
<td>systabauth</td>
<td>System Table</td>
</tr>
<tr>
<td>syscolauth</td>
<td>System Table</td>
</tr>
<tr>
<td>sysviews</td>
<td>System Table</td>
</tr>
</tbody>
</table>
The following example shows a searched `CASE` expression and an equivalent simple `CASE` expression.

- **Searched case expression:**
  ```sql
  SELECT tbl,
  CASE
    WHEN tbltype = 'S' THEN 'System Table'
  ELSE 'Not System table'
  END
  FROM systables;
  ```

- **Equivalent simple case expression:**
  ```sql
  SELECT tbl,
  CASE tbltype
    WHEN 'S' THEN 'System Table'
  ELSE 'Not System table'
  END
  FROM systables;
  ```
CAST function (SQL-92 compatible)

Syntax

CAST ( { expression | NULL } AS data_type [(length)] )

Description

The scalar function CAST converts an expression to another data type. The first argument is the expression to be converted. The second argument is the target data type.

The length option for the data_type argument specifies the length for conversions to character data types. If omitted, the default length is 30 bytes.

If the expression evaluates to null, the result of the function is null. Specifying NULL with the CAST function is useful for set operations such as UNION that require two tables to have the same structure. CAST NULL allows you to specify a column of the correct data type so a table with a similar structure to another, but with fewer columns, can be in a union operation with the other table.

The CAST function provides a data-type-conversion mechanism compatible with the SQL-92 standard.

Use the CONVERT function, enclosed in the ODBC escape clause {fn }, to specify ODBC-compliant syntax for data type conversion. See CONVERT function (ODBC compatible) (page 68) for more information.

Example

The following c-treeSQL example uses CAST to convert an integer field from a catalog table to a character data type:

```
SELECT CAST(fld AS CHAR(25)), fld FROM admin.syscalctable;
```

```
CONVERT(CHARACTER(25),FLD)          FLD
------------------------------------
100                                 100
```

1 record selected
CEILING function (ODBC compatible)

**Syntax**
```
CEILING ( expression )
```

**Description**
The scalar function CEILING returns the smallest integer greater than or equal to expression.

**Example**
```
SELECT CEILING (32.5) 'Ceiling'
    FROM ADMIN.SYSCALCTABLE;
```

**Notes**
- Expression must evaluate to a numeric data type.
CHAR function (ODBC compatible)

Syntax
CHAR ( integer_expression )

Description
The scalar function CHAR returns a character string with the first character having an ASCII value equal to the argument expression. CHAR is identical to CHR but provides ODBC-compatible syntax.

Example
```
SELECT *
  FROM   customer
  WHERE  SUBSTR (zip, 1, 1) = CHAR (53) ;
```

Notes
- The argument to the function must be of type INTEGER, TINYINT, or SMALLINT.
- The result is of type character.
- If the argument integer_expression evaluates to null, the result is null.
CHARTOROWID (extension)

Syntax

CHARTOROWID ( char_expression )

Description

The scalar function CHARTOROWID returns a ROWID contained in the input argument in character form.

Example

The following example shows the character-string format for a row identifier supplied as an argument to CHARTOROWID. In this example, the format for a row identifier is an integer (delimited as a character string by single quotes).

```sql
SELECT ROWID, FLD FROM SYSCALCTABLE;
ROWID  FLD
-----  ---
 0   100
1 record selected
```

CHARTOROWID requires single quotes around its argument

```sql
SELECT * FROM SYSCALCTABLE WHERE ROWID = CHARTOROWID ('0');
FLD
---
100
1 record selected
```

Notes

- The argument to the function must be of type character.
- The result is of internal ROWID type.
- If the argument char_expression evaluates to null, the result is null.
- The c-treeSQL statement execution returns error if the result of the input character expression does not contain a character string in the proper format for a row identifier.
CHR function (extension)

Syntax
CHR ( integer_expression )

Description
The scalar function CHR returns a character string with the first character having an ASCII value equal to the argument expression.

Example
SELECT *
FROM   customer
WHERE  SUBSTR (zip, 1, 1) = CHR (53) ;

Notes
- The argument to the function must be of type INTEGER, TINYINT, or SMALLINT.
- The result is of type character.
- If the argument integer_expression evaluates to null, the result is null.
COALESCE (SQL-92 compatible)

Syntax

COALESCE ( expression1, expression2 [ , ... ] )

Description

The `COALESCE` scalar function is a type of conditional expression. (See Conditional Expressions (page 25) for more information and a summary of all the conditional expressions.)

`COALESCE` specifies a series of expressions, and returns the first expression whose value is not null. If all the expressions evaluate as null, `COALESCE` returns a null value.

The `COALESCE` syntax is shorthand notation for a common case that can also be represented in a `CASE` expression. The following two formulations are equivalent:

```sql
COALESCE ( expression1 , expression2 , expression3 )
CASE
    WHEN expression1 IS NOT NULL THEN expression1
    WHEN expression2 IS NOT NULL THEN expression2
    ELSE expression3
```

Example

```sql
SELECT COALESCE(end_date, start_date) from job_hist;
```

Notes

- This function is not allowed in a `GROUP BY` clause
- Arguments to this function cannot be query expressions
CONCAT function (ODBC compatible)

Syntax
CONCAT ( char_expression , char_expression )

Description
The scalar function CONCAT returns a concatenated character string formed by concatenating argument one with argument two.

The CONCAT scalar function is similar to the concatenation operator. However, the concatenation operator allows easy concatenation of more than two character expressions by nesting the CONCAT function.

Example
SELECT name, empno, salary
FROM customer
WHERE project = CONCAT('US',proj_nam);

Notes
- Both the arguments must be of character type.
- If one of the arguments is a literal and the other one a field reference, concatenation is possible only if the literal is convertible to the character set of the field reference. Otherwise an error is returned.
- The result belongs to the character set of the arguments.
- If any of the argument expressions evaluates to null, the result is null.
- The trailing blanks for the first arguments are removed.
CONVERT function (extension)

Syntax

CONVERT ( [ 'data_type'(length) ] | data_type(length) ), expression

Description

The scalar function CONVERT converts an expression to another data type. The first argument is the target data type. The second argument is the expression to be converted to that type.

As indicated in the syntax, single quotes surrounding the data type are optional.

The length option for the data_type argument specifies the length for conversions to character types. If omitted, the default length is 30.

If the expression evaluates to null, the result of the function is null.

The CONVERT function syntax is similar to but not compatible with the ODBC CONVERT function. Enclose the function in the ODBC escape clause {fn }, to specify ODBC-compliant syntax. See CONVERT function (ODBC compatible) (page 68) for more information.

Examples

The following c-treeACE SQL examples convert an integer field from a catalog table to a character string:

```
SELECT CONVERT('CHAR', fld), fld FROM admin.syscalctable;
CONVERT(CHAR, FLD)                       FLD
-----------------               ---
100                        100
1 record selected
```

```
SELECT CONVERT('CHAR(35)', fld), fld FROM admin.syscalctable;
CONVERT(CHAR(35), FLD)                     FLD
-------------------               ---
100                         100
1 record selected
```

```
SELECT CONVERT(CHAR(50), fld), fld FROM systpe.syscalctable;
CONVERT(CHARACTER, FLD) FLD
---------------------- ---
100 100
1 record selected
```
CONVERT function (ODBC compatible)

Syntax

{fn CONVERT (expression , data_type ) }

data_type::

- SQL_BIGINT
- SQL_BINARY
- SQL_BIT
- SQL_CHAR
- SQL_DATE
- SQL_DECIMAL
- SQL_DOUBLE
- SQL_FLOAT
- SQL_INTEGER
- SQL_LONGVARBINARY
- SQL_LONGVARCHAR
- SQL_REAL
- SQL_SMALLINT
- SQL_TIME
- SQL_TIMESTAMP
- SQL_TINYINT
- SQL_VARBINARY
- SQL_VARCHAR

Description

The ODBC scalar function CONVERT converts an expression to another data type. The first argument is the expression to be converted. The second argument is the target data type. If the expression evaluates to null, the result of the function is null.

The ODBC CONVERT function provides ODBC-compliant syntax for data type conversion. You must enclose the function with the ODBC escape clause {fn } to use ODBC-compliant syntax.
COS function (ODBC compatible)

Syntax
COS (expression)

Description
The scalar function COS returns the cosine of expression.

Example
select cos(45 * pi()/180) 'Cosine of 45 degrees' from syscalctable;
COSINE OF 45 DEG
----------------
0.707106781186548
1 record selected

Notes
COS takes an angle (expression) and returns the ratio of two sides of a right triangle. The ratio is the length of the side adjacent to the angle divided by the length of the hypotenuse.

- Expression specifies an angle in radians
- Expression must evaluate to an approximate numeric data type.

To convert degrees to radians, multiply degrees by Pi/180. To convert radians to degrees, multiply radians by 180/Pi.
COT SQL Scaler Function

Syntax
COT(expression)

Description
The scalar function COT returns the cotangent of the expression. COT takes an angle expression and returns the ratio of two sides of a right angle triangle. The ratio is the length of the side adjacent to the angle to the length of the side opposite to the angle.

Example
SELECT COT(45 * PI()/180) 'COT OF 45 DEGREES' from admin.syscalctable;
COT OF 45 DEGREES
--------------------
1.000000000000000
1 record selected

Details
Expression specifies an angle in radians. Expression must evaluate to an approximate numeric data type. To convert degrees to radians, multiply degrees by PI/180. To convert radians to degrees, multiply radians by 180/PI.

Note: ODBC compatible


**CURDATE function (ODBC compatible)**

**Syntax**

```
CURDATE ()
```

**Description**

CURDATE returns the current date as a DATE value. This function takes no arguments.

c-treeSQL statements can refer to CURDATE anywhere they can refer to a DATE expression. For example,

```
INSERT INTO objects (object_owner, object_id, create_date)
VALUES (USER, 1001, CURDATE());
```
**CURRENT_DATE SQL function**

**Syntax**

`CURRENT_DATE`

**Description**

CURRENT_DATE returns the current date as a DATE value. A c-treeACE SQL statement can refer to CURRENT_DATE anywhere you can refer to a DATE expression.

**Example**

```sql
INSERT INTO objects(object_owner, object_id, create_date)
VALUES(user, 1002, CURRENT_DATE);
```

**Note:** SQL-99 compatible
CURRENT_TIMESTAMP function

Syntax
CURRENT_TIMESTAMP

Description
CURRENT_TIMESTAMP is a synonymous replacement for SYSTIMESTAMP.
CURRENT_USER  SQL Function

Syntax

CURRENT_USER

Description

CURRENT_USER returns a character string identifier for the database user as specified in the current connection. It returns a character in the database character set. If the current connection did not specify an user, CURRENT_USER returns the login name as determined by the host operating system.

c-treeACE SQL statements can refer to CURRENT_USER anywhere they can refer to a character string expression.

Example

ISQL>SELECT CURRENT_USER FROM admin.syscalctable ;
FRED
----------------- 
fred
1 record selected

Note: SQL-99 compatible
CURTIME function (ODBC compatible)

Syntax

CURTIME ()

Description

CURTIME returns the current time as a TIME value. This function takes no arguments.

c-treeSQL statements can refer to CURTIME anywhere they can refer to a TIME expression. For example,

```sql
INSERT INTO objects (object_owner, object_id, create_time)
VALUES (USER, 1001, CURTIME()) ;
```
Database (ODBC compatible)

Syntax

DATABASE ( )

Description

The scalar function DATABASE returns the name of the database corresponding to the current connection name. This function takes no arguments, and the trailing parentheses are optional.

Example

select database() from t2;
DATABASE
--------
steel
1 record selected
### DATALENGTH function

**Syntax**

```
DATALENGTH( expr )
```

**Description**

Returns the number of bytes used to represent any expression. `DATALENGTH` is especially useful with VARCHAR, VARBINARY, LVARCHAR and LVARBINARY data types because these data types can store variable length data.

**Arguments**

- `expr` - An expression of any type.

**Example**

```
SELECT DATALENGTH(tbl) FROM systables WHERE tbl='systables';
```

```
DATALENGTH(TBL)
--
9
```
DATEADD function

Syntax

DATEADD( interval, integer_exp, date_time_exp )

Description

This scalar function is the same as the scalar function TIMESTAMPADD. Refer to TIMESTAMPADD (page 158) for usage.
DATEDIFF function

Syntax
DATEDIFF( interval, date_time_exp1, date_time_exp2 )

Description
This scalar function is the same as the scalar function TIMESTAMPDIFF. Refer to TIMESTAMPDIFF (page 160) for usage.
DAYNAME function (ODBC compatible)

Syntax
DAYNAME ( date_expression )

Description
Returns a character string containing the name of the day (for example, Sunday, through Saturday) for the day portion of date_expression. The argument date_expression can be the name of a column, the result of another scalar function, or a date or timestamp literal.

Example
SELECT *
FROM orders
WHERE order_no = 342 and DAYNAME(order_date)='SATURDAY';

<table>
<thead>
<tr>
<th>ORDER_NO</th>
<th>ORDER_DATE</th>
<th>REFERENCE</th>
<th>CUST_NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>342</td>
<td>08/10/1991</td>
<td>tdfg/101</td>
<td>10001</td>
</tr>
</tbody>
</table>

1 record selected
DAYOFMONTH function (ODBC compatible)

Syntax
DAYOFMONTH ( date_expression )

Description
The scalar function `DAYOFMONTH` returns the day of the month in the argument as a short integer value in the range of 1 - 31.

Example
```
SELECT *
FROM  orders
WHERE DAYOFMONTH (order_date) = 14 ;
```

Notes
- The argument to the function must be of type `DATE`.
- The argument must be specified in the format `MM/DD/YYYY`.
- The result is of type `SHORT`.
- If the argument expression evaluates to null, the result is null.
DAYOFWEEK function (ODBC compatible)

Syntax

DAYOFWEEK ( date_expression )

Description

The scalar function DAYOFWEEK returns the day of the week in the argument as a short integer value in the range of 1 - 7.

Example

SELECT *
FROM  orders
WHERE DAYOFWEEK (order_date) = 2 ;

Notes

- The argument to the function must be of type DATE.
- The argument must be specified in the format MM/DD/YYYY.
- The result is of type SHORT.
- If the argument expression evaluates to null, the result is null.
DAYOFYEAR function (ODBC compatible)

Syntax
DAYOFYEAR ( date_expression )

Description
The scalar function DAYOFYEAR returns the day of the year in the argument as a short integer value in the range of 1 - 366.

Example
SELECT *
FROM orders
WHERE DAYOFYEAR (order_date) = 300 ;

Notes
- The argument to the function must be of type DATE.
- The argument must be specified in the format MM/DD/YYYY.
- The result is of type SHORT.
- If the argument expression evaluates to null, the result is null.
**DB_NAME (extension)**

**Syntax**

```
DB_NAME ( )
```

**Description**

The scalar function `DB_NAME` returns the name of the database corresponding to the current connection name. It provides compatibility with the Sybase SQL Server function `db_name`.

**Example**

```
SELECT DB_NAME() FROM T2;
```

```
DB_NAME
-------
ctreev4
```

1 record selected
DECODE function (extension)

Syntax

```sql
DECODE ( expression, search_expression, match_expression
          [, search_expression, match_expression …]
          [, default_expression ]
)
```

Description

The `DECODE` scalar function is a type of conditional expression. (Refer to Conditional Expressions (page 25) for a summary of all the conditional expressions.)

The scalar function `DECODE` compares the value of the first argument expression with each `search_expression` and if a match is found, returns the corresponding `match_expression`. If no match is found, then the function returns `default_expression`. If `default_expression` is not specified and no match is found, the function returns a null value.

`DECODE` provides a subset of the functionality of `CASE` that is compatible with Oracle SQL syntax. Use a simple case expression for SQL-compatible syntax (See CASE (SQL-92 Compatible) (page 57)).

Example

```sql
SELECT ename, DECODE (deptno,
       10, 'ACCOUNTS    ',
       20, 'RESEARCH    ',
       30, 'SALES       ',
       40, 'SUPPORT     ',
       'NOT ASSIGNED'
)
FROM employee ;
```

Notes

- The first argument expression can be of any type. The types of all `search_expressions` must be compatible with the type of the first argument.
- The `match_expressions` can be of any type. The types of all `match_expressions` must be compatible with the type of the first `match_expression`.
- The type of the `default_expression` must be compatible with the type of the first `match_expression`.
- The type of the result is the same as that of the first `match_expression`.
- If the first argument expression is null then the value of the `default_expression` is returned, if it is specified. Otherwise null is returned.
DEGREES function (ODBC compatible)

Syntax
DEGREES ( expression )

Description
The scalar function DEGREES returns the number of degrees in an angle specified in radians by expression.

Example
SELECT DEGREES(3.14159265359) 'Degrees in pi Radians'
FROM ADMIN.SYSCALCTABLE;

Notes
- Expression specifies an angle in radians.
- Expression must evaluate to a numeric data type.
DIFFERENCE function (ODBC compatible)

Syntax
DIFFERENCE ( string_exp1,string_exp2 )

Description
The scalar function DIFFERENCE returns an integer value that indicates the difference between the values returned by the SOUNDEX function for string_exp1 and string_exp2.

Example
SELECT DIFFERENCE(name,'Robets')
FROM customer
WHERE name = 'Roberts';

DIFFEREN

2
1 record selected

Notes
- The arguments of the function are of character types.
- The result is INTEGER.
- If the argument expression evaluates to null, the result is null.
EXP function (ODBC compatible)

Syntax
EXP ( expression )

Description
The scalar function EXP returns the exponential value of expression (e raised to the power of expression).

Example
SELECT EXP(1) FROM SYSCALCTABLE;
2.718282
--------
2.718281828459046
1 record selected

Notes
- Expression must evaluate to an approximate numeric data type.
**EXTRACT SQL Function**

**Syntax**

```
EXTRACT(date_time_field   FROM  date_time_expression)
```

```
date_time_field  ::
|           SECOND
|           MINUTE
|           HOUR
|           DAY
|           MONTH
|           YEAR
```

**Description**

The scalar function `EXTRACT` returns the a date and time field from a date and time expression.

**Example**

```
SELECT SYSDATE() ,EXTRACT(MONTH FROM SYSDATE()) 'month'
FROM  admin.syacalctable;
```

```
SYSDATE     MONTH
----------     ------
11/21/2004    11
```

**Details**

The `date_time_expression` can evaluate to a `TIME`, `DATE` or `TIMESTAMP` data type.

The `date_time_field` and `date_time_expression` must be compatible.

`date_time_expression` is a `TIME`, `DATE` or `TIMESTAMP` from which `EXTRACT` returns the specified `date_time_field`.

`date_time_field` keywords that specify the field to be extracted from the `date_time_expression`.

- `SECOND` specifies that second part has to be extracted from the `date_time_expression`.
- `MINUTE` specifies that minute part has to be extracted from the `date_time_expression`.
- `HOUR` specifies that hour part has to be extracted from the `date_time_expression`.
- `DAY` specifies that day part has to be extracted from the `date_time_expression`.
- `MONTH` specifies that month part has to be extracted from the `date_time_expression`.
- `YEAR` specifies that year part has to be extracted from the `date_time_expression`.

**Note:** SQL-99 and ODBC compatible
FLOOR function (ODBC compatible)

Syntax
FLOOR ( expression )

Description
The scalar function FLOOR returns the largest integer less than or equal to expression.

Example
SELECT FLOOR (32.5) 'Floor'
    FROM ADMIN.SYSCALCTABLE;

Notes
- Expression must evaluate to a numeric data type.
GETDATE function (ODBC compatible)

Syntax

GETDATE ( )

Description

Synonymous with NOW() (page 121).
GREATEST function (extension)

Syntax
GREATEST ( expression, expression, ... )

Description
The scalar function GREATEST returns the greatest value among the values of the given expressions.

Example
SELECT cust_no, name,
    GREATEST (ADD_MONTHS (start_date, 10), SYSDATE)
FROM customer;

Notes
- The first argument to the function can be of any type. The types of the subsequent arguments must be compatible with that of the first argument.
- The type of the result is the same as that of the first argument.
- If any of the argument expressions evaluates to null, the result is null.
**HOUR function (ODBC compatible)**

**Syntax**

```
HOUR ( time_expression )
```

**Description**

The scalar function HOUR returns the hour in the argument as a short integer value in the range of 0 - 23.

**Example**

```
SELECT *
FROM  arrivals
WHERE HOUR (in_time) < 12 ;
```

**Notes**

- The argument to the function must be of type TIME.
- The argument must be specified in the format hh:mm:ss.
- The result is of type SHORT.
- If the argument expression evaluates to null, the result is null.
IFNULL function (ODBC compatible)

Syntax

IFNULL( expr, value)

Description

The scalar function IFNULL returns value if expr is null. If expr is not null, IFNULL returns expr.

Example

select c1, ifnull(c1, 9999) from temp order by c1;
c1     ifnull(c1,9999)
  9999
  9999
  9999
  1     1
  3     3

Notes

The data type of value must be compatible with the data type of expr.
INITCAP function (extension)

Syntax

`INITCAP ( char_expression )`

Description

The scalar function `INITCAP` returns the result of the argument character expression after converting the first character to uppercase and the subsequent characters to lowercase.

Example

```
SELECT INITCAP (name)
FROM customer;
```

Notes

- The argument to the function must be of the character types.
- The result type is based on the argument type.
- If the argument expression evaluates to null, the result is null.
INSERT function (ODBC compatible)

Syntax
INSERT(string_exp1,start,length,string_exp2)

Description
The scalar function INSERT returns a character string where length characters have been deleted from string_exp1 beginning at start and string_exp2 has been inserted into string_exp1, beginning at start. The above operation will be performed only if both the arguments belong to the same character set (exceptions are shown below in the Notes section).

Example
SELECT INSERT(name,2,4,'xx')
FROM customer
WHERE name = 'Goldman';
INSERT(NAME,2,4,XX)

Gxxan
1 record selected

Notes
- The string_exp can be of fixed length or variable length character types.
- The start and length can be of the type INTEGER, SMALLINT, TINYINT or BIGINT.
- The string_exp2 has to belong to the string_exp1's character set.
- If one of the arguments is a literal and the other one a field reference, the operation is possible only if the literal is convertible to the character set of the field reference. Otherwise an error is returned.
- The result string is of the type string_exp1 and will belong to same character set as string_exp1.
- If any of the argument expression evaluates to a null, the result is a null.
- If start is negative or zero, the result string evaluates to a null.
- If length is negative, the result evaluates to a null.
INSTR function (extension)

Syntax
INSTR (char_expression, char_expression [, start_position [, occurrence]])

Description
The scalar function INSTR searches for the character string corresponding to the second argument in the character string corresponding to the first argument starting at start_position. If occurrence is specified, then INSTR searches for the nth occurrence where n is the value of the fourth argument.

The position (with respect to the start of string corresponding to the first argument) is returned if a search is successful. Zero is returned if no match can be found.

Example
SELECT cust_no, name
FROM customer
WHERE INSTR (LOWER (addr), 'heritage') > 0 ;

Notes
- The first and second arguments must be of character type.
- The third and fourth arguments, if specified, must be of type INTEGER.
- If one of the arguments is a literal and the other one a field reference, the operation is possible only if the literal is convertible to the character set of the field reference. Otherwise an error is returned.
- The values for specifying position in a character string starts from one. That is, the very first character in a string is at position one, the second character is at position two and so on.
- If the third argument is not specified, a default value of one is assumed.
- If the fourth argument is not specified, a default value of one is assumed.
- The result is of type INTEGER.
- If any of the argument expressions evaluates to null, the result is null.
ISNULL function

Syntax
ISNULL ( expr, value )

Description
This scalar function is the same as the scalar function IFNULL. Refer to IFNULL (page 94) for usage.
ISNUMERIC function

Syntax

ISNUMERIC( expr )

Description

The scalar function ISNUMERIC returns 1 if the input expression evaluates to an exact numeric or approximate numeric type; otherwise it returns 0. A return value of 1 guarantees that expr can be converted to one of these numeric types.

Arguments

- expr - Is an expression to be evaluated.

Example

SELECT ISNUMERIC(id), ISNUMERIC(tbl) FROM systables
WHERE tbl = 'systables';
ISNUMERIC(ID ISNUMERIC(TB
------------ ------------
1 0
LAST_DAY function (extension)

Syntax
LAST_DAY ( date_expression )

Description
The scalar function LAST_DAY returns the date corresponding to the last day of the month containing the argument date.

Example
SELECT *
FROM orders
WHERE LAST_DAY (order_date) + 1 = '08/01/1991' ;

Notes
- The argument to the function must be of type DATE.
- The result is of type DATE.
- If the argument expression evaluates to null, the result is null.
LAST_IDENT function (extension)

Syntax
LAST_IDENT ( )

Description
The scalar function LAST_IDENT() returns the last inserted IDENTITY value for the session.

Example
SELECT LAST_IDENT();

Notes
- The result is of type NUMERIC(32,0)
**LCASE function (ODBC compatible)**

**Syntax**

```
LCASE ( char_expression )
```

**Description**

The scalar function `LCASE` returns the result of the argument character expression after converting all the characters to lowercase. `LCASE` is the same as `LOWER` but provides ODBC-compatible syntax.

**Example**

```
SELECT *
FROM customer
WHERE LCASE (name) = 'smith' ;
```

**Notes**

- The argument to the function must be of character type.
- The result type is based on the argument type.
- If the argument expression evaluates to null, the result is null.
LEAST function (extension)

Syntax

LEAST ( expression, expression, ... )

Description

The scalar function LEAST returns the lowest value among the values of the given expressions.

Example

SELECT cust_no, name,
    LEAST (ADD_MONTHS (start_date, 10), SYSDATE)
FROM  customer ;

Notes

- The first argument to the function can be of any type. The types of the subsequent arguments must be compatible with that of the first argument.
- The type of the result is the same as that of the first argument.
- If any of the argument expressions evaluates to null, the result is null.
LEFT function (ODBC compatible)

Syntax

LEFT ( string_exp, count )

Description

The scalar function LEFT returns the leftmost count of characters of string_exp.

Example

SELECT LEFT(name,4)
FROM customer
WHERE name = 'Goldman';

LEFT(NAME,4)

Gold
1 record selected

Notes

- The string_exp can be of type fixed or variable length character type.
- The count can be of the type INTEGER, SMALLINT, BIGINT, or TINYINT.
- If any of the arguments of the expression evaluates to a null, the result would be null.
- If the count is negative, the result evaluates to a null.
LEN function (ODBC compatible)

Syntax
LEN ( char_expression )

Description
Synonymous with LENGTH() (page 106)
LENGTH function (ODBC compatible)

Syntax

```
LENGTH ( char_expression )
```

Description

The scalar function `LENGTH` returns the number of characters in `char_expression`, excluding trailing blanks.

Example

```
SELECT name 'LONG NAME'
FROM customer
WHERE LENGTH (name) > 5 ;
```

Notes

- The argument to the function must be of character type.
- The result is of type `INTEGER`.
- If the argument expression evaluates to null, the result is null.
LOCATE function (ODBC compatible)

Syntax
LOCATE( char-expr1 , char-expr2, [start-position] )

Description
The scalar function LOCATE returns the location of the first occurrence of char-expr1 in char-expr2. If the function includes the optional integer argument start-position, LOCATE begins searching char-expr2 at that position. If the function omits the start-position argument, LOCATE begins its search at the beginning of char-expr2.

LOCATE denotes the first character position of a character expression as one. If the search fails, LOCATE returns zero. If either character expression is null, LOCATE returns a null value.

Example
The following example uses two string literals as character expressions. LOCATE returns a value of six:

```sql
SELECT  LOCATE('this', 'test this test', 1) FROM TEST;
```

```
LOCATE(THIS,
------------
6
1 record selected
```
LOCALTIME SQL Function

Syntax
LOCALTIME([[time_precision]])

Description
The scalar function LOCALTIME returns current time as TIME value. This function takes an optional argument which decides the milliseconds precision in the result. The optional argument is an integer ranging between 0 and 3.

c-treeACE SQL statements can refer to LOCALTIME anywhere they can refer to a TIME expression.

Example
SELECT LOCALTIME(3) 'LOCALTIME' FROM admin.syscalctable;
LOCALTIME
-----------
12:34:000
1 record selected

Details
The time_precision value is always set as three irrespective of what the user has specified and milliseconds is always returned as '000'.
LOCALTIMESTAMP SQL Function

Syntax
LOCALTIMESTAMP( [timestamp_precision] )

Description
The scalar function LOCALTIMESTAMP returns current date and time as TIMESTAMP value. This function takes an optional argument which decides the milliseconds precision in the result. The optional argument timestamp_precision is an integer ranging between 0 and 9.

c-treeACE SQL statements can refer to LOCALTIMESTAMP anywhere they can refer to a TIMESTAMP expression.

Example
SELECT LOCALTIMESTAMP(3) 'localtimestamp' FROM admin.syscalctable ;
LOCALTIMESTAMP
--------------
12/05/2004  12:25:000
1 record selected

Details
The timestamp_precision value is always set as 3 irrespective of what the user has specified and milliseconds is always returned as '000'.
LOG SQL Scaler Function

Syntax
LOG ( expression )

Description
The scalar function LOG returns the natural logarithm of expression.

Example
SELECT LOG( 100 ) 'Natural Logarithm of 100' FROM admin.syscalctable ;
NATURAL LOGARITH
-------------
4.605170185988092
1 record selected

Details
Expression must evaluate to an approximate numeric data type.

Note: ODBC compatible
**LOG10 function (ODBC compatible)**

**Syntax**

```sql
LOG10 ( expression )
```

**Description**

The scalar function `LOG10` returns the base 10 logarithm of `expression`.

**Example**

```sql
SELECT LOG10 (100) 'Log base 10 of 100'
FROM ADMIN.SYSCALCTABLE;
```

**Notes**

- Expression must evaluate to an approximate numeric data type.
LOWER function (SQL-92 compatible)

Syntax
LOWER ( char_expression )

Description
The scalar function LOWER returns the result of the argument character expression after converting all the characters to lowercase.

Example
SELECT *
FROM customer
WHERE LOWER (name) = 'smith';

Notes
- The argument to the function must be of character type.
- The result type is based on the argument type.
- If the argument expression evaluates to null, the result is null.
LPAD function (extension)

Syntax
LPAD ( char_expression, length [, pad_expression] )

Description
The scalar function LPAD pads the character string corresponding to the first argument on the left with the character string corresponding to the third argument so that after the padding, the length of the result is length.

Example
SELECT LPAD (name, 30)
FROM customer ;
SELECT LPAD (name, 30, ' .')
FROM customer ;

Notes
- The first argument to the function must be of character type.
- The second argument to the function must be of type INTEGER.
- The third argument, if specified, must be of character type.
- If one of the arguments is a literal and the other one a field reference, the operation is possible only if the literal is convertible to the character set of the field reference. Otherwise an error is returned.
- If the third argument is not specified, the default value is a string of length one containing one blank.
- If L1 is the length of the first argument and L2 is the value of the second argument, then:
  - If L1 is less than L2, the number of characters padded is equal to L2 - L1.
  - If L1 is equal to L2, no characters are padded and the result string is the same as the first argument.
  - If L1 is greater than L2, the result string is equal to the first argument truncated to the first L2 characters.
- The result is of character type whose character set is same as that of the arguments.
- If the argument expression evaluates to null, the result is null.
LTRIM function (ODBC compatible)

Syntax
LTRIM ( char_expression [, char_set ] )

Description
The scalar function LTRIM removes all the leading characters in char_expression, that are present in char_set and returns the resultant string. Thus, the first character in the result is guaranteed to be not in char_set. If the char_set argument is omitted, the function removes the leading and trailing blanks from char_expression.

Example
SELECT name, LTRIM (addr, ' ') FROM customer;

Notes
- The first and second arguments to the function must be of character type.
- If one of the arguments is a literal and the other one a field reference, the operation is possible only if the literal is convertible to the character set of the field reference. Otherwise an error is returned.
- The result is of character type whose character set is same as that of the arguments.
- If the argument expression evaluates to null, the result is null.
MINUTE function (ODBC compatible)

Syntax

MINUTE ( time_expression )

Description

The scalar function MINUTE returns the minute value in the argument as a short integer in the range of 0 - 59.

Example

SELECT *
FROM  arrivals
WHERE MINUTE (in_time) > 10 ;

Notes

- The argument to the function must be of type TIME.
- The argument must be specified in the format HH:MI:SS.
- The result is of type SHORT.
- If the argument expression evaluates to null, the result is null.
MOD function (ODBC compatible)

Syntax
MOD ( expression1, expression2 )

Description
The scalar function MOD returns the remainder of expression1 divided by expression2.

Example
SELECT MOD (11, 4) 'Modulus'
    FROM ADMIN.SYSCALCTABLE;

Notes
- Both expression1 and expression2 must evaluate to exact numeric data types.
- If expression2 evaluates to zero, MOD returns zero.
MONTHNAME function (ODBC compatible)

Syntax
MONTHNAME ( date_expression )

Description
Returns a character string containing the name of the month (for example, January, through December) for the month portion of date_expression. Argument date_expression can be name of a column, the result of another scalar function, or a date or timestamp literal.

Example
SELECT * 
FROM orders 
WHERE order_no = 346 and MONTHNAME(order_date) = 'JUNE';

ORDER NO  ORDER DATE    REFERENCE    CUST NO
346  06/01/1991    87/rd        10002

1 record selected
MONTH function (ODBC compatible)

Syntax

MONTH (date_expression)

Description

The scalar function `MONTH` returns the month in the year specified by the argument as a short integer value in the range of 1 - 12.

Example

```
SELECT *
FROM orders
WHERE MONTH (order_date) = 6;
```

Notes

- The argument to the function must be of type `DATE`.
- The argument must be specified in the format `MM/DD/YYYY`.
- The result is of type `SHORT`.
- If the argument expression evaluates to null, the result is null.
MONTHS_BETWEEN function (extension)

Syntax
MONTHS_BETWEEN ( date_expression, date_expression )

Description
The scalar function MONTHS_BETWEEN computes the number of months between two date values corresponding to the first and second arguments.

Example
SELECT MONTHS_BETWEEN (SYSDATE, order_date)
FROM  orders
WHERE order_no = 1002 ;

Notes
- The first and the second arguments to the function must be of type DATE.
- The result is of type INTEGER.
- The result is negative if the date corresponding to the second argument is greater than that corresponding to the first argument.
- If any of the arguments expression evaluates to null, the result is null.
**NEXT_DAY function (extension)**

**Syntax**

NEXT_DAY ( date_expression, day_of_week )

**Description**

The scalar function **NEXT_DAY** returns the minimum date that is greater than the date corresponding to the first argument for which the day of the week is same as that specified by the second argument.

**Example**

SELECT NEXT_DAY (order_date, 'MONDAY')
FROM orders ;

**Notes**

- The first argument to the function must be of type **DATE**.
- The second argument to the function must be of type **NCHAR** or **NVARCHAR** for Unicode builds and type **CHAR** or **VARCHAR** for ANSI builds. The result of the second argument must be a valid day of week ('SUNDAY', 'MONDAY' etc.)
- The result is of type **DATE**.
- If any of the argument expressions evaluates to null, the result is null.
NOW function (ODBC compatible)

Syntax
NOW ( )

Description
NOW returns the current date and time as a TIMESTAMP value. This function takes no arguments.
NULLIF (SQL-92 compatible)

Syntax
NULLIF ( expression1, expression2 )

Description
The NULLIF scalar function is a type of conditional expression (See Conditional Expressions (page 25) for more information and a summary of all the conditional expressions).

The NULLIF scalar function returns a null value for expression1 if it is equal to expression2. It's useful for converting values to null from applications that use some other representation for missing or unknown data.

Notes
- This function is not allowed in a GROUP BY clause.
- Arguments to this function cannot be query expressions.
- The NULLIF expression is shorthand notation for a common case that can also be represented in a CASE expression, as follows:
  
  ```
  CASE
    WHEN expression1 = expression2 THEN NULL
    ELSE expression1
  END
  ```

Example
This example uses the NULLIF scalar function to insert a null value into an address column if the host-language variable contains a single space character.

INSERT INTO employee (add1) VALUES (NULLIF (:address1, ' '));
NVL function (extension)

Syntax
NVL ( expression, expression )

Description
The scalar function NVL returns the value of the first expression if the first expression value is not null. If the first expression value is null, the value of the second expression is returned.

The NVL function is not ODBC compatible. Use the IFNULL function for ODBC-compatible syntax.

Example
SELECT salary + NVL (comm, 0) 'TOTAL SALARY'
   FROM employee ;

Notes
- The first argument to the function can be of any type except for LONG data types.
- The type of the second argument must be compatible with that of the first argument.
- The type of the result is the same as the first argument.
**OBJECT_ID function (extension)**

**Syntax**

```sql
OBJECT_ID ('table_name')
```

**Description**

The scalar function `OBJECT_ID` returns the value of the id column in the `admin.systables`, plus one. This function provides compatibility with the Sybase SQL Server function `object_id`.

**Arguments**

- `table_name`

The name of the table for which `OBJECT_ID` returns an identification value.

**Example**

```sql
select id, object_id(tbl), tbl from admin.systables
where owner = 'admin';
```

<table>
<thead>
<tr>
<th>ID</th>
<th>OBJECT_ID(TB)</th>
<th>TBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>systblspaces</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>systables</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>syscolumns</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>sysindexes</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>systsfiles</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>syslogfiles</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>sysdbbackup</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>syslogbackup</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>sysdbsyncpt</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>sysdbuuid</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>syssyssvr</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>sysusrsrv</td>
</tr>
</tbody>
</table>

.`

`.```
**OCTET_LENGTH SQL Function**

**Syntax**

OCTET_LENGTH ( char_expression )

**Description**

The scalar function OCTET_LENGTH returns the number of bytes in the char_expression.

**Example**

```sql
SELECT NAME FROM customers
WHERE OCTET_LENGTH( name ) < 10 ;
```

<table>
<thead>
<tr>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>rathan</td>
</tr>
<tr>
<td>Dany</td>
</tr>
</tbody>
</table>

2 records selected

**Note:** SQL-99 and ODBC compatible
OVERLAY SQL Function

Syntax
OVERLAY ( char_exp1 PLACING char_exp2 FROM start_position
[ FOR length ] )

The argument length is optional. If length is not specified, the scalar function returns a character string where char_exp2 has been appended to the char_exp1 beginning at start_position.

Description
The scalar function OVERLAY returns a character string where length characters have been deleted from the char_exp1 beginning at start_position and char_exp2 has been inserted into char_exp1 beginning at start_position.

Example
SELECT OVERLAY( name PLACING 'Technologies' FROM 9 FOR 12 )
From customers
WHERE name = 'FairCom Corporation';
OVERLAY(NAME,8,9,TECHNOLOGIES)
--------------------
FairCom Technologies
1 record selected

Details
- The char_exp can be of fixed length or variable character types.
- The start_position and length can be of the type INTEGER, SMALLINT, TINYINT or BIGINT.
- The char_exp2 has to belong to char_exp1's character set.
- The result expression is of the type char_exp1 and will belong to the same character set as char_exp1.
- If start_position is negative or zero, the result expression evaluates to a null.
- If length is negative the result evaluates to a null.

Note: SQL-99 compatible
PI function (ODBC compatible)

Syntax
PI ( )

Description
The scalar function PI returns the constant value of pi as a floating point value.

Example
SELECT PI ()
    FROM ADMIN.SYSCALCTABLE;
POSITION SQL Function

Syntax
POSITION ( char_exp1 IN char_exp2 )

Description
The scalar function POSITION returns the first occurrence of char_exp1 in char_exp2.

Example
SELECT POSITION ( 'Fa' IN 'FairCom' )
FROM customers ;
POSITION
--------
1
1 record selected

Details
If any one of the expressions evaluates to be null, POSITION returns null.

Note: SQL-99 and ODBC compatible
POWER function (ODBC compatible)

Syntax

POWER ( expression1 , expression2 )

Description

The scalar function POWER returns expression1 raised to the power of expression2.

Example

SELECT POWER ( 3 , 2 ) '3 raised to the 2nd power'
FROM ADMIN.SYSCALCTABLE;

Notes

- Expression1 must evaluate to a numeric data type.
- Expression2 must evaluate to an exact numeric data type.
**PREFIX function (extension)**

**Syntax**

PREFIX(char_expression, start_position, char_expression)

**Description**

The scalar function PREFIX returns the substring of a character string starting from the position specified by start position, and ending before the specified character.

**Arguments**

**char_expression**

An expression that evaluates to a character string, typically a character-string literal or column name. If the expression evaluates to null, `PREFIX` returns null.

**start_position**

An expression that evaluates to an integer value. `PREFIX` searches the string specified in the first argument starting at that position. A value of one indicates the first character of the string.

**char_expression**

An expression that evaluates to a single character. `PREFIX` returns the substring that ends before that character. If `PREFIX` does not find the character, it returns the substring beginning with `start_position`, to the end of the string. If the expression evaluates to more than one character, `PREFIX` ignores all but the first character.

**Example**

```
SELECT C1, C2, PREFIX(C1, 1, '.') FROM T1;
C1       C2   PREFIX(C1,1,.)
---      --  ------------
test.pref .    test
pref.test s    pref
2 records selected

SELECT C1, C2, PREFIX(C1, 1, C2) FROM T1;
C1       C2   PREFIX(C1,1,C2)
---      --  ------------
test.pref .    test
pref.test s    pref.te
2 records selected

SELECT C1, C2, PREFIX(C1, 1, 'Q') FROM T1;
C1       C2   PREFIX(C1,1,'Q')
---      --  ------------
test.pref .    test.pref
pref.test s    pref.test
2 records selected
```

**Notes**

- If one of the arguments is a literal and the other one a field reference, the operation is possible only if the literal is convertible to the character set of the field reference. Otherwise an error is returned.
**QUARTER function (ODBC compatible)**

**Syntax**

```
QUARTER ( time_expression )
```

**Description**

The scalar function `QUARTER` returns the quarter in the year specified by the argument as a short integer value in the range of 1 - 4.

**Example**

```
SELECT *
FROM orders
WHERE QUARTER (order_date) = 3 ;
```

**Notes**

- The argument to the function must be of type `DATE`.
- The argument must be specified in the format `MM/DD/YYYY`.
- The result is of type `SHORT`.
- If the argument expression evaluates to null, the result is null.
RADIANS function (ODBC compatible)

Syntax
RADIANS ( expression )

Description
The scalar function RADIANS returns the number of radians in an angle specified in degrees by expression.

Example
SELECT RADIANS(180) 'Radians in 180 degrees'
FROM ADMIN.SYSCALCTABLE;

Notes
- Expression specifies an angle in degrees.
- Expression must evaluate to a numeric data type.
RAND function (ODBC compatible)

Syntax
RAND ( [ expression ] )

Description
The scalar function RAND returns a randomly-generated number, using expression as an optional seed value.

Example
SELECT RAND(3) 'Random number using 3 as seed value'
FROM ADMIN.SYSCALCTABLE;

Notes
- Expression must evaluate to an exact numeric data type.
REPEAT function (ODBC compatible)

Syntax
REPEAT ( string_exp,count )

Description
The scalar function REPEAT returns a character string composed of string_exp repeated count times.

Example
SELECT REPEAT(fld1,3)
FROM test100
WHERE fld1 = 'Afghanistan'

Results
REPEAT(FLD1,3)
AfghanistanAfghanistanAfghanistan
1 record selected

Notes
- The string expression can be any of the fixed length or variable length character types.
- The count can be of the type INTEGER, SMALLINT, BIGINT, or TINYINT.
- If any of the arguments of the expression evaluates to a null, the result would be null.
- If the count is negative or zero, the result evaluates to a null.
REPLACE function (ODBC compatible)

Syntax
REPLACE ( string_exp1, string_exp2, string_exp3 )

Description
The scalar function REPLACE replaces all occurrences of string_exp2 in string_exp1 with string_exp3. The resultant character string will have the same character set as that of the arguments.

Example
SELECT REPLACE ( name, 'mi', 'moo' ) 
FROM customer 
WHERE name = 'Smith';

REPLACE(NAME,MI,MOO)

Smooth
1 record selected

Notes
- string_exp can be any of the type fixed or variable length character types.
- If any of the arguments of the expression evaluates to null, the result is null.
- If one of the arguments is a literal and the other one a field reference, the operation is possible only if the literal is convertible to the character set of the field reference. Otherwise an error is returned.
- If the replacement string is not found in the search string, it returns the original string.
RIGHT function (ODBC compatible)

Syntax
RIGHT ( string_exp, count )

Description
The scalar function RIGHT returns the rightmost count of characters of string_exp.

Example
SELECT RIGHT(fld1,6)
FROM test100
WHERE fld1 = 'Afghanistan';
RIGHT(FLD1,6)

istan
1 record selected

Notes
- The string_exp can be any of the fixed or variable length Character types.
- The count can be of the type INTEGER, SMALLINT, BIGINT, or TINYINT.
- If any of the arguments of the expression evaluates to a null, the result would be null.
- If the count is negative, the result evaluates to a null.
### ROUND function

**Syntax**

ROUND( number, length [,operation_type] )

**Description**

The scalar function rounds `number` to `length` decimal places.

If value of `length` is positive, `number` is rounded to the number of decimal places specified by `length`.

If `length` is negative, `number` is rounded on the left side of the decimal point, as specified by the positive value of `length`.

**Arguments**

- `number` - This is an expression of an exact numeric or approximate numeric data type.
- `length` - This is the precision or number of digits to which `number` is to be rounded. The argument value must be of data type TINYINT, SMALLINT or INTEGER.
- `operation_type` - This is the type of operation - rounding or truncation - to perform. The argument value must be of type TINYINT, SMALLINT or INTEGER. If the argument is omitted or has a value of 0 (default), `number` is rounded. If a value other than 0 is specified, `number` is truncated.

**Example**

```sql
SELECT price, ROUND(price, 1), ROUND(price, -1)
FROM custorder;

PRICE ROUND(PRI ROUND(PRI
----- -------- -------
1546.56 1546.60 1550.00

SELECT price, ROUND(price, 1, 1), ROUND(price, -1, 1)
FROM custorder;

PRICE ROUND(PRI ROUND(PRI
----- -------- -------
1546.56 1546.50 1540.00
```
ROWID (extension)

Syntax
ROWID

Description
The scalar function ROWID returns the row identifier of the current row in a table. This function takes no arguments. The ROWID of a row is determined when the row is inserted into the table. Once assigned, the ROWID remains the same for the row until the row is deleted. At any given time, each row in a table is uniquely identified by its ROWID.

Selecting a row in a table using its ROWID is the most efficient way of selecting the row. For example:

```
SELECT *
FROM customers
WHERE ROWID = '10';
```
ROWIDTOCHAR (extension)

Syntax
ROWIDTOCHAR ( expression )

Description
The scalar function ROWIDTOCHAR returns the character form of a ROWID contained in the input argument.

Example
The following example uses ROWIDTOCHAR to convert a row identifier from its internal representation to a character string.

```
SELECT cust_no,
       SUBSTR (ROWIDTOCHAR (ROWID), 1, 8) 'PAGE NUMBER',
       SUBSTR (ROWIDTOCHAR (ROWID), 10, 4) 'LINE NUMBER',
       SUBSTR (ROWIDTOCHAR (ROWID), 15, 4) 'TABLE SPACE NUMBER'
FROM customer;
```

Notes
- The argument to the function must be a ROWID.
- The result is of type NCHAR for Unicode builds and CHAR for ANSI builds.
- If the argument expression evaluates to null, the result is null.
**RPAD function (extension)**

**Syntax**

```
RPAD ( char_expression, length [, pad_expression] )
```

**Description**

The scalar function `RPAD` pads the character string corresponding to the first argument on the right with the character string corresponding to the third argument so that after the padding, the length of the result would be equal to the value of the second argument length.

**Example**

```
SELECT RPAD (name, 30)
FROM customer;

SELECT RPAD (name, 30, '.')
FROM customer;
```

**Notes**

- The first argument to the function must be of character type.
- The second argument to the function must be of type `INTEGER`.
- The third argument, if specified, must be of character type.
- If one of the arguments is a literal and the other one a field reference, the operation is possible only if the literal is convertible to the character set of the field reference. Otherwise an error is returned.
- If L1 is the length of the first argument and L2 is the value of the second argument, then:
  - If L1 is less than L2, the number of characters padded is equal to L2 - L1.
  - If L1 is equal to L2, no characters are padded and the result string is the same as the first argument.
  - If L1 is greater than L2, the result string is equal to the first argument truncated to first L2 characters.
- The result is of character type whose character set is same as that of it's arguments.
- If the argument expression evaluates to null, the result is null.
**RTRIM function (ODBC compatible)**

**Syntax**

```
RTRIM ( char_expression [ , char_set ] )
```

**Description**

The scalar function `RTRIM` removes all the trailing characters in `char_expression`, that are present in `char_set` and returns the resultant string. Thus, the last character in the result is guaranteed to be not in `char_set`. If the `char_set` argument is omitted, the function removes the leading and trailing blanks from `char_expression`.

**Example**

```
SELECT RPAD ( RTRIM (addr, ' '), 30, '.')
FROM customer ;
```

**Notes**

- The first and second arguments to the function must be of character type.
- If one of the arguments is a literal and the other one a field reference, the operation is possible only if the literal is convertible to the character set of the field reference. Otherwise an error is returned.
- The result is of character type.
- If the argument expression evaluates to null, the result is null.
SECOND function (ODBC compatible)

Syntax
SECOND ( time_expression )

Description
The scalar function SECOND returns the seconds in the argument as a short integer value in the range of 0 - 59.

Example
SELECT *
FROM arrivals
WHERE SECOND (in_time) <= 40 ;

Notes
- The argument to the function must be of type TIME.
- The argument must be specified in the format HH:MI:SS.
- The result is of type SHORT.
- If the argument expression evaluates to null, the result is null.
SESSION_USER SQL Function

Syntax
SESSION_USER

Description
The scalar function SESSION_USER returns the value of the c-treeACE SQL session identifier. c-treeACE SQL returns the user.

Example
ISQL>SELECT SESSION_USER FROM admin.syscalctable;
FRED
-------
fred
1 record selected

Note: SQL-99 compatible
SIGN function (ODBC compatible)

Syntax
SIGN ( expression )

Description
The scalar function SIGN returns 1 if expression is positive, -1 if expression is negative, or zero if it is zero.

Example
SELECT SIGN(-14) 'Sign'
FROM ADMIN.SYSCALCTABLE;

Notes
- Expression must evaluate to a numeric data type.
**SIN function (ODBC compatible)**

**Syntax**

\[
\text{SIN ( expression )}
\]

**Description**

The scalar function \text{SIN} returns the sine of expression.

**Example**

```sql
select sin(45 * pi()/180) 'Sine of 45 degrees' from syscalctable;
```

```
SINE OF 45 DEGRE
----------------
0.707106781186547
1 record selected
```

**Notes**

\text{SIN} takes an angle (expression) and returns the ratio of two sides of a right triangle. The ratio is the length of the side opposite the angle divided by the length of the hypotenuse.

- Expression specifies an angle in radians.
- Expression must evaluate to an approximate numeric data type.

To convert degrees to radians, multiply degrees by \( \pi/180 \). To convert radians to degrees, multiply radians by \( 180/\pi \).
**SOUNDEX function (ODBC compatible)**

Syntax

```sql
SOUNDEX ( string_exp )
```

Description

The scalar function `SOUNDEX` returns a four-character soundex code for character strings that are composed of a contiguous sequence of valid single- or double-byte roman letters.

Example

```sql
SELECT SOUNDEX('Roberts')
FROM syscalctable;
```
**SPACE function (ODBC compatible)**

**Syntax**

```
SPACE ( count )
```

**Description**

The scalar function `SPACE` returns a character string consisting of count spaces.

**Example**

```
SELECT CONCAT(SPACE(3), name)
FROM customer
WHERE name = 'Roberts';
```

```
CONCAT (   ,NAME)
Roberts
```

1 record selected

**Notes**

- The count argument can be of type `INTEGER`, `SMALLINT`, `BIGINT`, or `TINYINT`.
- If count is null, the result is null.
- If count is negative, the result is null.
**SQRT function (ODBC compatible)**

**Syntax**

`SQRT(expression)`

**Description**

The scalar function `SQRT` returns the square root of the expression.

**Example**

```sql
SELECT SQRT(28) 'square root of 28'
FROM ADMIN.SYSCALCTABLE;
```

**Notes**

- The value of the expression must be positive.
- The expression must evaluate to an approximate numeric data type.
SUBSTR function (extension)

Syntax
SUBSTR ( char_expression, start_position [, length ] )

Description
The scalar function SUBSTR returns the substring of the character string corresponding to the first argument starting at start_position and length characters long. If the third argument length is not specified, substring starting at start_position up to the end of char_expression is returned.

Example
SELECT name, '(', SUBSTR (phone, 1, 3) , ')',
       SUBSTR (phone, 4, 3), '-',
       SUBSTR (phone, 7, 4)
FROM customer ;

Notes
- The first argument must be of character type.
- The second argument must be of type INTEGER.
- The third argument, if specified, must be of type INTEGER.
- The values for specifying position in the character string start from one. The very first character in a string is at position one, the second character is at position two and so on.
- The result is of character type.
- If any of the argument expressions evaluates to null, the result is null.
SUBSTRING SQL Function (SQL-99 compatible)

Syntax

```
SUBSTRING (char_exp FROM start_position [ FOR length ])
```

Description

The scalar function `SUBSTRING` returns the substring of a character string corresponding to the first argument starting at `start_position` and `length` characters long.

The third argument `length` is optional. If the `length` is not specified the substring starting at `start_position` up to the end of the `char_exp` is returned.

Example

```
SELECT SUBSTRING( 'FairCom Corporation' FROM 1 FOR 7)
From customers ;
FAIRCOM
------
FairCom
1 record selected
```

Details

- The first argument, `char_exp`, must be of character type.
- The second argument, `start_position`, must be of type `INTEGER`.
- The third argument, `length`, if specified, must be of type `INTEGER`.
- The values for specifying position in `char_exp` start from 1. The very first character in a string is at position 1, the second character is at position 2 and so on.
- The result is of character type.
- If any of the argument expressions evaluates to null, the result is null.

Note: SQL-99 compatible
SUBSTRING function (ODBC compatible)

Syntax

```
SUBSTRING ( char_expression, start_position [, length ] )
```

Description

The scalar function `SUBSTRING` returns the substring of the character string corresponding to the first argument starting at `start_position` and `length` characters long. If the third argument, `length`, is not specified, the substring starting at `start_position` up to the end of `char_expression` is returned. `SUBSTRING` is identical to `SUBSTR` and provides ODBC-compatible syntax.

Example

```
SELECT name, '(', SUBSTRING (phone, 1, 3) , ')',
       SUBSTRING (phone, 4, 3), ' - ',
       SUBSTRING (phone, 7, 4)
FROM customer ;
```

Notes

- The first argument must be of character type.
- The second argument must be of type `INTEGER`.
- The third argument, if specified, must be of type `INTEGER`.
- The values for specifying position in the character string start from one. The very first character in a string is at position one, the second character is at position two and so on.
- The result is of character type.
- If any of the argument expressions evaluates to null, the result is null.
**SUFFIX function (extension)**

**Syntax**

```sql
SUFFIX(char_expression1, start_position, char_expression2)
```

**Description**

The scalar function `SUFFIX` returns the substring of a character string starting after the position specified by `start_position` and the second `char_expression`, to the end of the string.

**Arguments**

- **char_expression1**
  An expression that evaluates to a character string, typically a character-string literal or column name. If the expression evaluates to null, `SUFFIX` returns null.

- **start_position**
  An expression that evaluates to an integer value. `SUFFIX` searches the string specified in the first argument starting at that position. A value of one indicates the first character of the string.

- **char_expression2**
  An expression that evaluates to a single character. `SUFFIX` returns the substring that begins with that character. If `SUFFIX` does not find the character after `start_position`, it returns null. If the expression evaluates to more than one character, `SUFFIX` ignores all but the first character.

**Example**

```
SELECT C1, C2, SUFFIX(C1, 6, '.') FROM T1;
```

```
<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>SUFFIX(C1, 6, .)</th>
</tr>
</thead>
<tbody>
<tr>
<td>test.</td>
<td>pref</td>
<td>.</td>
</tr>
<tr>
<td>pref</td>
<td>test</td>
<td>s</td>
</tr>
</tbody>
</table>
```

```
2 records selected
```

```
SELECT C1, C2, SUFFIX(C1, 1, C2) FROM T1;
```

```
<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>SUFFIX(C1, 1, C2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>test.</td>
<td>pref</td>
<td>pref</td>
</tr>
<tr>
<td>pref</td>
<td>test</td>
<td>s t</td>
</tr>
</tbody>
</table>
```

```
2 records selected
```

```
SELECT C1, C2, SUFFIX(C1, 6, '.') FROM T1;
```

```
<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>SUFFIX(C1, 6, .)</th>
</tr>
</thead>
<tbody>
<tr>
<td>test.</td>
<td>pref</td>
<td>.</td>
</tr>
<tr>
<td>pref</td>
<td>test</td>
<td>s</td>
</tr>
</tbody>
</table>
```

```
2 records selected
```

**NOTES**

- If one of the arguments is a literal and the other one a field reference, the operation is possible only if the literal is convertible to the character set of the field reference. Otherwise an error is returned.
SUSER_NAME function (extension)

Syntax

SUSER_NAME ( [user_id] )

Description

The scalar function SUSER_NAME returns the user login name for the user_id specified in the input argument. If no user_id is specified, SUSER_NAME returns the name of the current user.

This function provides compatibility with the Sybase SQL Server function suser_name. It is identical to the USER_NAME function.

Example

select suser_name() from admin.syscalctable;
SUSER_NAME
----------
searle
1 record selected

select suser_name(104) from admin.syscalctable;
SUSER_NAME(104)
----------
dbp
1 record selected

select id, tbl, owner from admin.systables
1 where owner = suser_name();

<table>
<thead>
<tr>
<th>ID</th>
<th>TBL</th>
<th>OWNER</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>test</td>
<td>searle</td>
</tr>
<tr>
<td>42</td>
<td>t2</td>
<td>searle</td>
</tr>
<tr>
<td>43</td>
<td>t1</td>
<td>searle</td>
</tr>
</tbody>
</table>

3 records selected
SYSDATE function (extension)

Syntax

SYSDATE [ ( ) ]

Description

SYSDATE returns the current date as a DATE value. This function takes no arguments, and the trailing parentheses are optional.

ctreeACE SQL statements can refer to SYSDATE anywhere they can refer to a DATE expression. For example,

```
INSERT INTO objects (object_owner, object_id, create_date)
VALUES (USER, 1001, SYSDATE) ;
```
SYSTIME function (extension)

Syntax

SYSTIME \( ( ) \)

Description

SYSTIME returns the current time as a TIME value. This function takes no arguments, and the trailing parentheses are optional.

c-treeACE SQL statements can refer to SYSTIME anywhere they can refer to a TIME expression. For example,

```
INSERT INTO objects (object_owner, object_id, create_time)
    VALUES (USER, 1001, SYSTIME) ;
```
**SYSTIMESTAMP function (extension)**

**Syntax**

SYSTIMESTAMP [ ( ) ]

**Description**

SYSTIMESTAMP returns the current date and time as a TIMESTAMP value. This function takes no arguments, and the trailing parentheses are optional.

The following c-treeSQL example shows the different formats for SYSDATE, SYSTIME, and SYSTIMESTAMP:

```
SELECT SYSDATE FROM test;
SYSDATE
-------
09/13/1994
1 record selected

SELECT SYSTIME FROM test;
SYSTIME
-------
14:44:07:000
1 record selected

SELECT SYSTIMESTAMP FROM test;
SYSTIMESTAMP
------------
1994-09-13 14:44:15:000
1 record selected
```
**TAN function (ODBC compatible)**

**Syntax**

```
TAN ( expression )
```

**Description**

The scalar function `TAN` returns the tangent of `expression`.

**Example**

```
select tan(45 * pi()/180) 'Tangent of 45 degrees' from syscalctable;
```

```
TANGENT OF 45 DE
----------------
1.000000000000000
1 record selected
```

**Notes**

`TAN` takes an angle (expression) and returns the ratio of two sides of a right triangle. The ratio is the length of the side opposite the angle divided by the length of the side adjacent to the angle.

- Expression specifies an angle in radians.
- Expression must evaluate to an approximate numeric data type.

To convert degrees to radians, multiply degrees by $\pi/180$. To convert radians to degrees, multiply radians by $180/\pi$. 
TIMESTAMPADD function (ODBC compatible)

Syntax
TIMESTAMPADD(interval, integer_exp, date_time_exp)

interval:
  SQL_TSI_FRAC_SECOND
  SQL_TSI_SECOND
  SQL_TSI_MINUTE
  SQL_TSI_HOUR
  SQL_TSI_DAY
  SQL_TSI_DAYOFYEAR
  SQL_TSI_WEEK
  SQL_TSI_MONTH
  SQL_TSI_QUARTER
  SQL_TSI_YEAR
  MILLISECOND
  SECOND
  MINUTE
  HOUR
  DAY
  DAYOFYEAR
  WEEK
  MONTH
  QUARTER
  YEAR

Description
Returns the timestamp calculated by adding integer_exp intervals of type interval to timestamp_exp.

Arguments
interval
Keywords that specify the interval to add to timestamp_exp. The SQL_TSI_FRAC_SECOND keyword specifies fractional seconds as milliseconds.

Keywords such as SECOND and HOUR are functionally equivalent to the keywords with the same name but having prefix SQL_TSI, i.e., the keywords such as SQL_TSI_SECOND and SQL_TSI_HOUR. The keyword MILLISECOND is the same as SQL_TSI_FRAC_SECOND and this interval means milliseconds. The keyword DAYOFYEAR or SQL_TSI_DAYOFYEAR indicates an interval value which ranges from 1 to 366 (See scalar function DAYOFYEAR()).

integer_exp
The number of interval values to add to timestamp_exp. integer_exp can be any expression that evaluates to an integer data type.

date_time_exp
A date-time expression from which TIMESTAMPADD calculates the return value. If date_time_exp is a date value and interval specifies fractional seconds, seconds, minutes, or hours, the time portion of timestamp_exp is set to 0 before calculating the resulting timestamp.
Example
The following example displays the current system time and uses the `TIMESTAMPADD` scalar function to add 8 hours to it.

```sql
> select systime, timestampadd(sql_tsi_hour, 8, systime) from syscalctable;
15:03:57:000    06/08/1999 23:03:57:000
-----------------
15:03:57:000,   1999-06-08 23:03:57:000
```
TIMESTAMPDIFF function (ODBC compatible)

Syntax
TIMESTAMPDIFF(interval, date_time_exp1, date_time_exp2)

interval:
- SQL_TSI_FRAC_SECOND
- SQL_TSI_SECOND
- SQL_TSI_MINUTE
- SQL_TSI_HOUR
- SQL_TSI_DAY
- SQL_TSI_WEEK
- SQL_TSI_MONTH
- SQL_TSI_QUARTER
- SQL_TSI_YEAR

Description
Returns an integer representing the number of intervals by which \textit{date_time_exp2} is greater than \textit{date_time_exp1}.

Arguments

interval
Keywords that specify the interval in which to express the difference between the two date-time arguments. The SQL_TSI_FRAC_SECOND keyword specifies fractional seconds in milliseconds.

date_time_exp1
A date-time expression which \textit{TIMESTAMPADD} subtracts from \textit{date_time_exp2}.

date_time_exp2
A date-time expression from which \textit{TIMESTAMPADD} subtracts \textit{date_time_exp1}.

Example
The following example displays difference in seconds between the current system time and one day later.

\begin{verbatim}
> select timestampdiff(sql_tsi_second, sysdate, sysdate + 1) from syscalctable;
86400
-----
86400
\end{verbatim}

Notes
If either date-time expression is a time value and interval specifies days, weeks, months, quarters, or years, the date portion of that expression is set to the current date before calculating the difference between the expressions.

If either date-time expression is a date value and interval specifies fractional seconds, seconds, minutes, or hours, the time portion of that expression is set to 0 before calculating the difference between the expressions.
**TO_CHAR function (extension)**

**Syntax**

```
TO_CHAR ( expression [ , format_string ] )
```

**Description**

The scalar function `TO_CHAR` converts the given expression to character form and returns the result. The primary use for `TO_CHAR` is to format the output of date-time expressions through the `format_string` argument.

**Arguments**

- **expression**
  Specifies the expression to be converted to character form. To use the `format_string` argument, `expression` must evaluate to a date or time value.

- **format_string**
  A date-time format string that specifies the format of the output. See *Date Format Strings* (page 41) and *Time Format Strings* (page 42) for more details. c-treeACE SQL ignores the format string if the expression argument does not evaluate to a date or time.

**Example**

```
SELECT C1 FROM T2;
C1
--
09/29/1952
1 record selected

SELECT TO_CHAR(C1, 'Day, Month ddth'),
       TO_CHAR(C2, 'HH12 a.m. ') FROM T2;
TO_CHAR(C1, DAY, MONTH DDTH)  TO_CHAR(C2, HH12 A.M.)
---------------------------  ---------------------
Monday   , September 29th    02 p.m.
1 record selected
```

**Notes**

- The first argument to the function can be of any type.
- The second argument, if specified, must be of type `NCHAR` or `NVARCHAR` for UNICODE builds and of type `CHAR` or `VARCHAR` for ANSI builds.
- The result is of character type.
- The format argument can be used only when the type of the first argument is `DATE`.
- If any of the argument expressions evaluates to null, the result is null.
**TO_DATE function (extension)**

**Syntax**

```sql
TO_DATE ( date_lit )
```

**Description**

The scalar function `TO_DATE` converts the given date literal to a date value.

**Example**

```sql
SELECT * FROM orders
WHERE order_date <= TO_DATE ('12/31/1991') ;
```

**Notes**

- The result is of type `DATE`.
- Supply the date literal in any valid format. See *Date Literals* (page 28) for valid formats of `DATE` literals.
TO_NUMBER function (extension)

Syntax
TO_NUMBER ( char_expression )

Description
The scalar function TO_NUMBER converts the given character expression to a number value.

Example
SELECT * FROM customer
     WHERE TO_NUMBER (SUBSTR (phone, 1, 3)) = 603 ;

Notes
• The argument to the function must be of type NCHAR or NVARCHAR for Unicode builds and of type CHAR or VARCHAR for ANSI builds.
• The result is of type NUMERIC.
• If any of the argument expressions evaluates to null, the result is null.
TO_TIME function (extension)

Syntax
TO_TIME ( time_lit )

Description
The scalar function TO_TIME converts the given time literal to a time value.

Example
SELECT * FROM orders
    WHERE order_date < TO_DATE ('05/15/1991')
    AND order_time < TO_TIME ('12:00:00') ;

Notes
- The result is of type TIME.
- Supply the time literal in any valid format. See Time Literals (page 29) for valid formats of TIME literals.
TO_TIMESTAMP function (extension)

Syntax

TO_TIMESTAMP ( timestamp_lit )

Description

The scalar function TO_TIMESTAMP converts the given timestamp literal to a timestamp value.

Example

SELECT * FROM orders
WHERE order_timestamp > TO_TIMESTAMP('4/18/95 10:41:19')

Notes

- The result is of type TIMESTAMP.
- Supply the timestamp literal in any valid format. See Timestamp Literals (page 30) for valid formats of TIMESTAMP literals.
TRANSLATE function (extension)

Syntax
TRANSLATE ( char_expression, from_set, to_set )

Description
The scalar function TRANSLATE translates each character in char_expression that is in from_set to the corresponding character in to_set. The translated character string is returned as the result. This function is similar to the Oracle TRANSLATE function.

Example
This example substitutes underscores for spaces in customer names.

SELECT TRANSLATE (customer_name, ' ', '_')
   "TRANSLATE Example" from customers;

TRANSLATE EXAMPLE
-----------------
Sports_Cars_Inc.__________________________________
Mighty_Bulldozer_Inc._____________________________
Ship_Shapers_Inc._________________________________
Tower_Construction_Inc.___________________________
Chemical_Construction_Inc.________________________
Aerospace_Enterprises_Inc.________________________
Medical_Enterprises_Inc.__________________________
Rail_Builders_Inc.________________________________
Luxury_Cars_Inc._________________________________
Office_Furniture_Inc.______________________________
10 records selected

Notes
- char_expression, from_set, and to_set can be any character expression.
- For each character in char_expression, TRANSLATE checks for the same character in from_set:
  - If it is in from_set, TRANSLATE translates it to the corresponding character in to_set (if the character is the nth character in from_set, the nth character in to_set).
  - If the character is not in from_set TRANSLATE does not change it.
- If from_set is longer than to_set, TRANSLATE does not change trailing characters in from_set that do not have a corresponding character in to_set.
- If either from_set or to_set is null, TRANSLATE does nothing.
TRIM SQL Function

Syntax
TRIM([trim_specification] [trim_characters] FROM char_expression)

trim specification ::
    LEADING
    | TRAILING
    | BOTH

Description
The function TRIM removes leading, trailing or both leading and trailing characters from the char_expression, that are present in trim_characters and returns the resultant string.

Example
SELECT TRIM ( LEADING 'x' FROM 'xxFairCom' )
FROM customers ;
FAIRCOM
------
FairCom
1 record selected

Details
- If trim_specification is not specified, the function TRIM removes both the leading and trailing trim_characters from the char_expression.
- If trim_characters are not specified, the function TRIM removes leading, trailing or both leading and trailing blank spaces from the char_expression.
- If both trim_specification and trim_characters are not specified, the function TRIM removes both leading and trailing blank spaces from the char_expression.

Note: SQL-99 compatible
TRUNCATE function

Syntax
TRUNCATE( number, length )

Description
The scalar function truncates number to length decimal places.
If value of length is positive, number is truncated to the length of places to the right of the decimal point.
If length is negative, number is truncated to positive value of length places to the left of decimal point.

Arguments
- number - This is an expression of an exact numeric or approximate numeric data type.
- length - Is the precision to which number is truncated.

Example
SELECT price, TRUNCATE(price, 1), TRUNCATE(price, -1)
FROM custorder;

PRICE  TRUNCATE( TRUNCATE
--------- ---------
1546.56 1546.50 1540.00
UCASE function (ODBC compatible)

Syntax

UCASE ( char_expression )

Description

The scalar function UCASE returns the result of the argument character expression after converting all the characters to uppercase. UCASE is identical to UPPER, but provides ODBC-compatible syntax.

Example

SELECT *
FROM customer
WHERE UCASE (name) = 'SMITH' ;

Notes

- The argument to the function must be of character type.
- The result is of character type.
- If the argument expression evaluates to null, the result is null.
**UPPER function (SQL-92 compatible)**

**Syntax**

```sql
UPPER ( char_expression )
```

**Description**

The scalar function `UPPER` returns the result of the argument character expression after converting all the characters to uppercase.

**Example**

```sql
SELECT *
FROM customer
WHERE UPPER (name) = 'SMITH' ;
```

**Notes**

- The argument to the function must be of character type.
- The result is of character type.
- If the argument expression evaluates to null, the result is null.
USER function (ODBC compatible)

Syntax
USER [ ( ) ]

Description
USER returns a character-string identifier for the database user, as specified in the current connection. If the current connection did not specify a user, USER returns the login name as determined by the host operating system. This function takes no arguments, and the trailing parentheses are optional.

c-treeSQL statements can refer to USER anywhere they can refer to a character string expression.

Example
The following interactive c-treeSQL example shows connecting to a database as the user fred. Queries on two system tables illustrate the USER scalar function and retrieve the names of any tables owned by the user fred:

```sql
% isql -u fred tstdb
ISQL> select user from admin.syscalctable;
FRED
----
fred
1 record selected
ISQL> select tbl, owner from adminadmin.systables where owner = user();
TBL     OWNER
---     ----
flab     fred
1 record selected
```
USER_NAME function (extension)

Syntax
USER_NAME ( [user_id] )

Description
The scalar function USER_NAME returns the user login name for the user_id specified in the input argument. If no user_id is specified, USER_NAME returns the name of the current user.

The scalar function USER_NAME is identical to SUSER_NAME.
WEEK function (ODBC compatible)

Syntax
WEEK ( time_expression )

Description
The scalar function WEEK returns the week of the year as a short integer value (range 1 - 53).

Example
SELECT * 
FROM orders 
WHERE WEEK (order_date) = 5 ;

Notes
• The argument to the function must be of type DATE.
• The argument must be specified in the format MM/DD/YYYY.
• The result is of type SHORT.
• If the argument expression evaluates to null, the result is null.
YEAR function (ODBC compatible)

Syntax
YEAR ( date_expression )

Description
The scalar function YEAR returns the year as a short integer value in the range of 0 - 9999.

Example
SELECT *  
FROM orders  
WHERE YEAR (order_date) = 1992 ;

Notes
- The argument to the function must be of type DATE.
- The argument must be specified in the format MM/DD/YYYY.
- The result is of type SHORT.
- If the argument expression evaluates to null, the result is null.
2. **SQL Statements**

This chapter provides detailed reference material on each SQL statement.

### 2.1 Syntax Conventions

Syntax diagrams appear in monospace type and use the following conventions:

<table>
<thead>
<tr>
<th>UPPERCASE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPERCASE</td>
<td>Uppercase type denotes reserved words. You must include reserved words in statements, but they can be upper or lower case.</td>
</tr>
<tr>
<td>lowercase</td>
<td>Lowercase type denotes either user-supplied elements or names of other syntax diagrams. User-supplied elements include names of tables, host-language variables, expressions, and literals. Syntax diagrams can refer to each other by name. If a diagram is named, the name appears in lowercase type above and to the left of the diagram, followed by a double-colon (for example, privilege ::). The name of that diagram appears in lowercase in diagrams that refer to it.</td>
</tr>
<tr>
<td>{}</td>
<td>Braces denote a choice among mandatory elements. They enclose a set of options, separated by vertical bars (</td>
</tr>
<tr>
<td>[]</td>
<td>Brackets denote an optional element or a choice among optional elements.</td>
</tr>
<tr>
<td></td>
<td>Vertical bars separate a set of options.</td>
</tr>
<tr>
<td>…</td>
<td>A horizontal ellipsis denotes that the preceding element can optionally be repeated any number of times.</td>
</tr>
<tr>
<td>( ), ;</td>
<td>Parentheses and other punctuation marks are required elements. Enter them as shown in syntax diagrams.</td>
</tr>
</tbody>
</table>
ALTER INDEX

Description
Alters an existing index by passing specific c-tree file attributes to change, or by renaming the index.

Syntax
ALTER INDEX [ owner_name. ] index_name [ON [ owner_name. ] table_name ]
{STORAGE_ATTRIBUTES 'attributes'
| RENAME TO new_index_name
};

(Support for altering an index by renaming it, RENAME TO new_index_name, was added in V11.)

Arguments
STORAGE_ATTRIBUTES 'attributes'
A quoted string specifying index attributes to modify. c-treeACE SQL recognizes the following attributes:

- 'HUGE' -- Recreate the index file as a c-tree HUGE file.
- 'partition=<rule>' -- Use this index with <rule> as a partition rule and create table as partitioned file. Only one index can be the partitioning index rule for a table at a time.

Examples
By default, c-treeACE SQL returns an error in response to an ALTER_INDEX statement:

ISQL> alter index t1_ix ON t1 RENAME TO t1_mod_ix;

Authorization
The user executing this statement must have any of the following privileges:

- DBA privilege.
- Ownership of the index.
- EXCLUSIVE file access is required for any ALTER operation.

<table>
<thead>
<tr>
<th>SQL Compliance</th>
<th>Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>Embedded SQL, interactive SQL, ODBC applications</td>
</tr>
<tr>
<td>Related Statements</td>
<td>CREATE INDEX, DROP INDEX</td>
</tr>
</tbody>
</table>
**ALTER SEQUENCE**

**Syntax**

```
ALTER SEQUENCE sequence_name
    [ INCREMENT BY increment_value ]
    [ MAXVALUE max_value | NOMAXVALUE ]
    [ MINVALUE min_value | NOMINVALUE ]
    [ CYCLE | NOCYCLE ]
```

**Description**

ALTER SEQUENCE is a SQL command used to alter one or more characteristics of a sequence.

- `sequence_name` is the name of the sequence that is to be modified.
- `increment_value` specifies a new increment by value to be associated with the sequence.
- `max_value` specifies a new maximum value to be associated with the sequence.
- `min_value` specifies a new minimum value to be associated with the sequence.
- `cycle` specifies if the sequence should cycle or not.

Note that it is possible to put the sequence into a state from which no further values can be returned through the use of the alter command. For example, if the new `max_value` is less than the current value of the sequence for an ascending sequence, or if the new `increment_value` would make the next value be outside of the sequence bounds.

**Return Values**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Returned By</th>
</tr>
</thead>
<tbody>
<tr>
<td>-20266</td>
<td>Sequence cannot be used here</td>
<td>DROP SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20267</td>
<td>Sequence not found</td>
<td>DROP SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20268</td>
<td>START-WITH/CURRENT-VALUE cannot be greater than MAXVALUE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20269</td>
<td>START-WITH/CURRENT-VALUE cannot be less than MINVALUE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20270</td>
<td>Invalid sequence MINVALUE specified</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20271</td>
<td>Invalid sequence INCREMENT specified</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20272</td>
<td>START-WITH cannot be altered in sequence</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20273</td>
<td>No options specified for ALTER SEQUENCE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20274</td>
<td>Sequence increment has exceeded MAXVALUE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20275</td>
<td>Sequence decrement has exceeded MINVALUE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>Error Code</td>
<td>Message</td>
<td>Returned By</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>-20276</td>
<td>Only SELECT and ALTER privileges are valid for sequences</td>
<td>CREATE SEQUENCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALTER SEQUENCE</td>
</tr>
</tbody>
</table>

See c-tree Plus Error Codes (http://docs.faircom.com/doc/ctreeplus/28320.htm) for a complete listing of valid c-tree Plus error values.

See Also
- Sequence Values
- DROP SEQUENCE (page 220)
- CREATE SEQUENCE (page 195)
ALTER TABLE

Description
Alters the schema of an existing table by adding new columns or modifying existing columns of the specified table.

Syntax
ALTER TABLE [ owner_name. ] table_name
{ ADD new_column_definition [, new_column_definition] ... 
| MODIFY (column_name [ new_column_type ] 
[ default_value ] [ NULL | NOT NULL ] 
| DROP [ COLUMN ] { column_name | { column_name , ... } } 
| ADD [ CONSTRAINT new_constraint_name ] table_constraint 
| RENAME CONSTRAINT constraint_name TO new_constraint_name 
| RENAME COLUMN column_name TO new_column_name 
| RENAME TO new_table_name
} ;

new_column_definition ::
new_column_name  column_type
[ [ NULL | NOT NULL ]
| default_value
| IDENTITY [(seed, increment)] ]

default_value ::
DEFAULT { literal | USER | NULL | UID | SYSDATE | SYSTIME | SYSTIMESTAMP }

table_constraint ::
 PRIMARY KEY { column [, ... ] }
| UNIQUE { column [, ... ] }
| FOREIGN KEY { column [, ... ] }
REFERENCES [ owner_name. ] table_name [ ( column [, ... ] ) ]
| CHECK { search_condition }
| REORG

Notes
• Column additions and modifications have the following restrictions:
  • A NOT NULL column can be added to a table only if the table does not contain any rows.
  • The type can be modified or the length of the column can be decreased only if all the rows contain null values for the column being modified.
  • An existing column can be made NOT NULL only if none of the rows contain a null value for that column.
• When a new column is added to an existing table, it is added after the existing columns for the table.
• Views that automatically refer to all the columns of a table (such as SELECT * FROM ...) need to be dropped and added to select any columns that have been added to the table after the view has been created.
• If you add a FOREIGN KEY constraint to an existing table, the table and column specified in the REFERENCES clause must exist before the constraint will work as expected. However, c-treeACE SQL does not check for the existence of the table and column specified in the REFERENCES clause. It is up to you to make sure they exist.
• When you drop or rename a column, the following database objects dependent on that column are also automatically dropped or updated:
  • Indexes that specify the column as one of the components
  • Update triggers that specify the column
  • Unique, foreign key, and check constraints that refer to the column
  • UPDATE and REFERENCES privileges that refer to the column
• However, columns referenced in check constraints, views, procedures, triggers and UDFs can not be modified from an ALTER TABLE event and need to be simultaneously addressed by a database administrator.
• ALTER TABLE DROP COLUMN statements cannot:
  • Specify a column created as a primary key, if columns in other tables refer to the column in a referential constraint
  • Name all the columns in a table
• When new columns are added IDENTITY can be specified for one of the columns only if the table does not already contain an identity column. A DEFAULT value can not be specified for and IDENTITY column.
• Existing rows are not updated to the default value, even if the data was initially missing. Only new inserted rows as assigned the new default value.
• (c-treeACE SQL V10.3 and later) Any table altered in the structure using ALTER TABLE or truncated using TRUNCATE will have the $DELFLD$ set to 4 bytes despite the size in the original table.
• RENAME CONSTRAINT is available in V11 and later.
• REORG is available in V11.5 and later.

Examples
To add columns to the tmp table:

```
ISQL> ALTER TABLE tmp
        ADD  mname CHAR(8) ;
ISQL> ALTER TABLE tmp
        ADD  (mname CHAR(8), nname CHAR(8)) ;
```

To add table-level constraints, consider the table tmp consisting of integer fields, *fld*, *fld1*, *fld2*, and *fld3*.

• To create a unique key on *fld*:

```
ISQL> ALTER TABLE tmp ADD UNIQUE(fld) ;
```

• To create a primary key on *fld1*:

```
ISQL> ALTER TABLE tmp ADD primary key(fld1) ;
```

• To create a foreign key constraint named *fri_cns* on *fld2*:

```
ISQL> ALTER TABLE tmp ADD constraint fri_cns foreign key(fld2) ;
```

• To create a check constraint, *chk_cns* on *fld3*:

```
ISQL> ALTER TABLE tmp ADD constraint chk_cns (fld2 > 10) ;
```

Authorization
The user executing this statement must have any of the following privileges:

• DBA privilege.
Ownership of the table.

ALTER privilege on the table.

**SQL Compliance**
- SQL-92, ODBC Core SQL grammar

**Environment**
- Embedded SQL, interactive SQL, ODBC applications

**Related Statements**
- CREATE TABLE, DROP TABLE
ALTER VIEW

Description
Creates a view with the specified name on existing tables and/or views.

Syntax
ALTER VIEW [ owner_name. ] view_name
[ ( column_name, column_name, ... ) ]
AS [ ( ) query_expression [ ] ]
[ WITH CHECK OPTION ] ;

Notes
- ALTER VIEW has the same syntax as CREATE VIEW. ALTER VIEW does not modify an existing view, but rather replaces the existing view with a complete new definition. However, it retains the permissions from the existing view for the new view. It is best practice, therefore, to use a script file to create a view that may be modified in the future. The script can then be modified and used with ALTER VIEW to effect the change without needing to reenter the permissions of the view.
- The owner_name is made the owner of the created view.
- The column names specified for the view are optional and provide an alias for the columns selected by the query specification. If the column names are not specified then the view will be created with the same column names as the tables and/or views it is based on.
- A view is deletable if deleting rows from that view is allowed. For a view to be deletable, the view definition has to satisfy the following conditions:
  - The first FROM clause contains only one table reference or one view reference.
  - There are no aggregate functions, DISTINCT clause, GROUP BY or HAVING clause in the view definition.
  - If the first FROM clause contains a view reference, then the view referred to is deletable.
- A view is updatable if updating rows from that view is allowed. For a view to be updatable, the view has to satisfy the following conditions:
  - The view is deletable (That is, it satisfies all the conditions specified above for deletability).
  - All the select expressions in the first SELECT clause of the view definition are simple column references.
  - If the first FROM clause contains a view reference, then the view referred to is updatable.
- A view is insertable if inserting rows into that view is allowed. For a view to be insertable, the view has to satisfy the following conditions:
  - The view is updatable (That is, it satisfies all the conditions specified above for updatability).
  - If the first FROM clause contains a table reference, then all NOT NULL columns of the table are selected in the first SELECT clause of the view definition.
  - If the first FROM clause contains a view reference, then the view referred to is insertable.
- The WITH CHECK OPTION clause can be specified only if the view is updatable.
• If WITH CHECK OPTION clause is specified when defining a view, then during any update or insert of a row on this view, it is checked that the updated/inserted row satisfies the view definition (That is, the row is selectable using the view).

Examples

```sql
ALTER VIEW ne_customers AS
    SELECT cust_no, name, street, city, state, zip
    FROM customer
    WHERE state IN ('NH', 'MA', 'NY', 'VT')
    WITH CHECK OPTION ;
ALTER VIEW order_count (cust_number, norders) AS
    SELECT cust_no, COUNT(*)
    FROM orders
    GROUP BY cust_no;
```

Authorization

The user executing this statement must have the following privileges:

• DBA or RESOURCE privilege.
• SELECT privilege on all the tables/views referred to in the view definition.

If owner_name is specified and is different from the name of the user executing the statement, then the user must have DBA privilege.

<table>
<thead>
<tr>
<th>SQL Compliance</th>
<th>SQL-92, ODBC Core SQL grammar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>Embedded SQL, interactive SQL, ODBC applications</td>
</tr>
<tr>
<td>Related Statements</td>
<td>Query Expressions, DROP VIEW</td>
</tr>
</tbody>
</table>
CALL

Description
Invokes a built-in c-treeACE SQL procedure or Java stored procedure (JSP).

Syntax
[ ? = ] CALL proc_name([parameter],[, ...]);

Arguments
[ ? = ]
A parameter marker for the return value of the procedure. Programs must determine if the procedure returns a value and use the parameter marker as a placeholder for a variable that will receive the return value.
CALL proc_name
The name of the procedure to invoke.
parameter
Literal or variable values to pass to the procedure.

Example
The following example shows invocation of the SQLPROC built-in procedure from interactive c-treeACE SQL. It invokes SQLPROC to retrieve information about another built-in procedure, SQLTABLES:

```sql
CALL SQLPROC('',0,'admin',6,'sqltables',9); -- specific procedure
```

<table>
<thead>
<tr>
<th>PROCEDURE_QUALIFIER</th>
<th>PROCEDURE_OWNER</th>
<th>PROCEDURE_NAME</th>
<th>NUM_INPUT</th>
<th>NUM_OUTPUT</th>
<th>NUM_RESULT</th>
<th>REMARKS</th>
<th>PROCEDURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>sqltables</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Returns info about a table</td>
<td>1</td>
</tr>
</tbody>
</table>

Authorization
- Users must have the DBA or EXECUTE privilege to invoke a stored procedure.
- Users invoking a stored procedure do not need privileges to database objects accessed by the procedure. When a user executes a stored procedure, c-treeACE SQL checks the privileges of the procedure owner, not the procedure user, on any objects that the procedure accesses. This enables a user to execute a procedure successfully even when that user does not have the privileges to directly access objects used by the procedure.

SQL Compliance
ODBC Extended SQL grammar, when enclosed in ODBC escape clause

Environment
Embedded SQL, interactive SQL, ODBC applications

Related Statements
CREATE PROCEDURE, DROP PROCEDURE
COMMIT WORK

Description
Used to commit a transaction explicitly after executing one or more c-treeACE SQL statements. Committing a transaction makes permanent any changes made by the c-treeACE SQL statements.

Syntax
COMMIT [ WORK ] ;

Notes
- The set of c-treeACE SQL statements executed prior to executing COMMIT WORK statement are executed as one atomic transaction that is recoverable, serializable and durable.
- On a system failure and/or the execution of the ROLLBACK WORK, the transaction is marked for abortion and the database is rolled back to its initial state.
- A commit operation makes any database modifications made by that transaction permanent.
- Once a commit operation is executed the database modifications cannot be rolled back.
- Once a commit operation is executed the transaction modifications are guaranteed to be durable irrespective of any transient system failures.
- The atomicity applies only to the database modification and not to any direct I/O performed to devices such as the terminal, printer and OS files by the application code.
- A commit operation releases any locks implicitly or explicitly acquired by the transaction on the database.
- Under certain circumstances, c-treeACE SQL marks a transaction for abort but does not actually roll it back. Without an explicit rollback, any subsequent updates will not take effect, since a COMMIT statement cause c-treeACE SQL to recognize the transaction as marked for abort, and instead implicitly rolls back the transaction. c-treeACE SQL marks a transaction for abort under these conditions:
  - Hardware or software system failures
  - Lock timeout errors

SQL Compliance
SQL-92

Environment
Embedded SQL and interactive

Related Statements
ROLLBACK WORK
CONNECT

Description
Establishes a connection to a database. Optionally, the CONNECT statement can also specify a name for the connection and a user-name/password for authentication.

Syntax
CONNECT TO connect_string
   [AS connection_name]
   [USER user_name]
   [USING password]
connect_string::=
   { DEFAULT
   | db_name
   | port@host_name:db_name }

Arguments
Arguments to CONNECT must either be string literals enclosed in quotation marks or character-string host variables.

connect_string
The string that specifies the database to connect to. If the CONNECT statement specifies DEFAULT, c-treeACE SQL tries to connect to the environment-defined database, if any. (How you define the default database varies between operating systems. On Unix, the value of the DB_NAME environment variable specifies the default connect string.)

The connect string can be a simple database name or a complete connect string. A complete connect string has the following components:

   port The port number, default 6597

   host_name Name or IP address of the system where the database resides.

   db_name Name of the database.

connection_name
The name for the connection for use in DISCONNECT and SET CONNECTION statements. If the CONNECT statement omits a connection name, the default is the name of the database. Connection names must be unique.

user_name
User name for authentication of the connection. c-treeACE SQL verifies the user name against a corresponding password before it connects to the database. If omitted, the default value depends on the environment. (On UNIX, the value of the DH_USER environment variable specifies the default user name. If DH_USER is not set, the value of the USER environment variable specifies the default user name.)

password
Password for authentication of the connection. c-treeACE SQL verifies the password against a corresponding user name before it connects to the database. If omitted, the default value depends on the environment.

(On Unix, the value of the \textit{DH\_PASSWD} environment variable specifies the default password.)

**Notes**
- Arguments to \texttt{CONNECT} must either be string literals enclosed in quotation marks or character-string host variables.
- An application can connect to more than one database at a time, with a maximum of 10 connections. However, the application can actually gain access to only one database at a time. The database name specified in the \texttt{CONNECT} statement becomes the active one.
- If an application executes an c-treeACE SQL statement before connecting to a database, c-treeACE SQL tries to connect to the database specified through the \texttt{DB\_NAME} environment variable, if that environment variable is defined. If the connection is successful, the c-treeACE SQL statement executes on that database.

**Authorization**
None.

**Examples**
The following examples illustrate the \texttt{CONNECT} statement:
- The first statement shown connects to the “salesdb” database on the local system.
- The second statement connects to the “custdb” database on the local system using a network protocol.
- The third statement connects to the “custdb” database on the local mode, using the local pipe IPC protocol.
- The fourth statement connects to the environment-defined database by default (if any).

\begin{verbatim}
CONNECT TO 'salesdb' AS 'sales_conn';
CONNECT TO '6597@localhost:custdb' AS 'cust_conn';
CONNECT TO '6597@testbox:custdb' AS 'cust_conn';
CONNECT TO DEFAULT;
\end{verbatim}

**SQL Compliance** SQL-92

**Environment** Embedded SQL and interactive

**Related Statements** DISCONNECT, SET CONNECTION
CREATE FUNCTION

Description

Creates a User Defined Scaler Function (UDF) for the specified table. User Defined Scalar Functions are an extension to the existing built-in scalar functions and return a single value each time one is invoked. These functions can be used in queries in the same way that system defined scalar functions are used. UDFs are written with Java source code. For more detail on creating and using triggers, see the c-treeACE SQL Guide to Using Stored Procedures and Triggers and User Defined Functions.

Syntax

```
CREATE FUNCTION [ IF NOT EXISTS ] [ owner_name.]function_name
    ( [parameter_decl , ... ] )
    RETURNS (data_type)
    [ IMPORT
        java_import_clause ]
    BEGIN
        java_snippet
    END
```

parameter_decl :
    [ IN ] parameter_name data_type Arguments

IF NOT EXISTS

This argument avoids failure by creating the function only if a function of the same name does not already exist.

owner_name

Specifies the owner of the user defined function. If the name is different from the user name of the user executing the statement, then the user must have DBA privileges.

function_name

Names the user defined function. DROP FUNCTION statements specify the function_name defined here. c-treeACE SQL also uses function_name in the name of the Java class it creates from the Java snippet.

RETURNS
    data_type

The return value of the function as defined by the data_type declaration.

IMPORT
    java_import_clause

Specifies standard Java classes to import. The IMPORT keyword must be upper case and on a separate line.

BEGIN
    java_snippet
END
The body of the function. The body contains the Java source code that implements the user defined function. The Java statements become a method in a class that c-treeACE SQL creates and submits to the Java compiler.

The BEGIN and END keywords must be upper case and on separate lines.

Notes
- User Defined Scalar Function are a type of c-treeACE SQL expression that return a value based on the argument(s) supplied. User Defined Scalar Function are invoked in exactly the same manner as built in scalar functions.
- User Defined Scalar Functions can be used in the SELECT list or in the WHERE clause. They can be used as parameters of other scalar functions or in any expression. The parameter passed to a user defined scalar function can be a literal, field reference or any expression.

Example
The following example creates a User Defined Scalar Function named ‘str_cat’ that takes two input arguments and returns the concatenated string.

```sql
CREATE FUNCTION str_cat(IN org_string VARCHAR(20), IN string_to_concat VARCHAR(20))
RETURNS VARCHAR(40)
IMPORT
  import java.math.*;
BEGIN
  String new_str = org_string + string_to_concat;
  return new_str;
END
```

Authorization
Users executing CREATE FUNCTION must have the DBA privilege or RESOURCE privilege. The owner or users with the DBA privilege can execute or drop any User Defined Scalar Function, and grant the EXECUTE privilege to other users.

When a User Defined Scalar Function is executed on behalf of a user with EXECUTE privilege on that User Defined Scalar Function, for the objects that are accessed by the User Defined Scalar Function, the User Defined Scalar Function owner’s privileges are checked and not the user’s. This enables a user to execute a User Defined Scalar Function successfully even when he does not have the privileges to directly access the objects that are accessed by the User Defined Scalar Function, so long as he has EXECUTE privilege on the User Defined Scalar Function.

SQL Compliance
SQL-93, ODBC Core SQL grammar

Environment
Embedded SQL, interactive SQL, ODBC and JDBC applications

Related Statements
DROP FUNCTION
CREATE INDEX

Description
Creates an index on the specified table using the specified columns of the table. An index improves the performance of c-treeACE SQL operations whose predicates are based on the indexed column. However, an index slows performance of INSERT, DELETE and UPDATE operations.

Syntax
```
CREATE [ UNIQUE ] INDEX [ IF NOT EXISTS ] index_name
    ON table_name
    ( {column_name [ASC | DESC]} [ , ... ] )
    [ STORAGE_ATTRIBUTES 'attributes' ];
```

Arguments
UNIQUE
A UNIQUE index will not allow the table to contain any rows with duplicate column values for the set of columns specified for that index.

IF NOT EXISTS
This argument avoids failure by creating the index only if an index of the same name does not already exist.

index_name
The name of the index has to be unique for the specified table.

table_name
The name of the table on which the index is being built.

column_name [ , ... ]
The columns on which searches and retrievals will be ordered. These columns are called the index key. When more than one column is specified in the CREATE INDEX statement a concatenated index is created.

ASC | DESC
The index can be ordered as either ascending (ASC) or descending (DESC) on each column of the concatenated index. The default is ASC.

STORAGE_ATTRIBUTES 'attributes'
A quoted string that specifies index attributes that are specific to c-treeACE SQL indexes. c-treeACE SQL interprets the following attribute strings:

- "PARTITION=<rule>" - Creates the index as the partition index to the table enabling multiple table partitions. This option can be specified after a table has been created, and the table will be rebuilt according to the partition rule in effect.

  <rule> is a c-treeACE conditional expression evaluating to a numeric value.
"PARTITION" - Creates the index as the partition index to the table enabling multiple table partitions. This option can be specified after a table has been created, and the table will be rebuilt according to the partition rule in effect. (This type of partitioned file currently require a server with your partition rules included at compile time. See the c-treeACE Programmer's Reference Guide (http://docs.faircom.com/doc/ctreeplus) for details.)

To combine STORAGE_ATTRIBUTE options, separate them with a semicolon (;).

Example
CREATE UNIQUE INDEX postdate_idx ON customer (invoice_date) STORAGE_ATTRIBUTES 'partition';

Authorization
The user executing this statement must have any of the following privileges:

- DBA privilege.
- Ownership of the index.
- INDEX privilege on the table.
- EXCLUSIVE file access is required for any this operation.

SQL Compliance
ODBC Core SQL grammar. Extensions: STORAGE_ATTRIBUTES,

Environment
Embedded SQL, interactive SQL, ODBC applications

Related Statements
CREATE TABLE, DROP INDEX, ALTER TABLE
CREATE PROCEDURE

Description

Creates a stored procedure. Stored procedures contain a Java code “snippet” that is processed by c-treeACE SQL into a Java class definition and stored in the database in text and compiled form. c-treeACE SQL applications invoke stored procedures through the SQL CALL statement or the procedure-calling mechanisms of ODBC and JDBC.

For more detail on creating and using stored procedures, see the c-treeACE SQL Guide to Java Stored Procedures and Triggers.

Syntax

CREATE PROCEDURE [ IF NOT EXISTS ] [ owner_name. ] procname
( [ parameter_decl [ , ... ] ] )
[ RESULT ( column_name data_type [ , ... ] ) ]
[ IMPORT java_import_clause ]
BEGIN
  java_snippet
END

parameter_decl ::=
{ IN | OUT | INOUT } parameter_name data_type

Arguments

IF NOT EXISTS

This argument avoids failure by creating the procedure only if a procedure of the same name does not already exist.

owner_name

Specifies the owner of the procedure. If the name is different from the user name of the user executing the statement, then the user must have DBA privileges.

procname

Names the stored procedure. DROP PROCEDURE statements specify the procedure name defined here. c-treeSQL also uses procname in the name of the Java class that it creates from the Java snippet.

IN | OUT | INOUT

Specifies whether the following parameter declaration is input, output, or both:

- Calling applications pass values for input parameters in the CALL statement or CALL escape sequence
- Stored procedures assign values to output parameters as part of their processing
- INOUT parameters both have a value passed in and receive a new value during procedure processing

parameter_name data_type
Names a parameter and associates an c-treeACE SQL data type with it. The data_type must be one of the supported data types described in Data Types (page 10).

RESULT ( column_name data_type [ , ... ] )

Specifies columns in the result set the procedure returns. If the CREATE PROCEDURE statement includes this clause, the Java snippet must explicitly insert rows into the result set using the c-treeACE SQL Java class SQLResultSet.

Note that the column_name argument is not used within the stored procedure body. Instead, methods of the c-treeACE SQL Java classes refer to columns in the result set by ordinal number, not by name. (Interactive c-treeACE SQL uses the column names as headers when it displays procedure result sets.)

IMPORT

    java_import_clause

Specifies standard Java classes to import. The IMPORT keyword must be upper case and on a separate line.

BEGIN

    java_snippet

END

The body of the stored procedure. The body is a sequence of Java statements between the BEGIN and END keywords. The Java statements become a method in a class c-treeACE SQL creates and submits to the Java compiler.

The BEGIN and END keywords must be upper case and on separate lines. You cannot follow the END keyword with a semicolon.

Example

CREATE PROCEDURE new_sal (  
    IN  deptnum   INTEGER, 
    IN  pct_incr  INTEGER, 
) 
RESULT (  
    empname CHAR(20), 
    oldsal   NUMERIC, 
    newsal   NUMERIC 
)
BEGIN
    StringBuffer ename = new StringBuffer (20) ;
    BigDecimal osal = new BigDecimal () ;
    BigDecimal nsal = new BigDecimal () ;

    SQLCursor empcursor = new SQLCursor (  
        "SELECT empname, sal, (sal * ( ? /100) + NVL (comm, 0)) total, 
        FROM emp WHERE deptnum = ? ",
    ) ;

    empcursor.setParam (1, pct_incr);
    empcursor.setParam (2, deptnum);
    empcursor.open () ;
    do 
    { 
        empcursor.fetch () ;
        if (empcursor.found ())
{  
  empcursor.getValue (1, ename);
  empcursor.getValue (2, osal);
  empcursor.getValue (3, nsal);
  
  SQLResultSet.set (1, ename);
  SQLResultSet.set (2, osal);
  SQLResultSet.set (3, nsal);
  SQLResultSet.insert ();
}
} while (empcursor.sound ()) ;

empcursor.close () ;

END

Authorization

- Users issuing the CREATE PROCEDURE statement must have the DBA privilege or RESOURCE privilege.
- The owner or users with the DBA privilege can execute or drop any stored procedure, and grant the EXECUTE privilege to other users.
- Users must have the DBA or EXECUTE privilege to invoke a stored procedure.
- Users invoking a stored procedure do not need privileges to database objects accessed by the procedure. When a user executes a stored procedure, c-treeACE SQL checks the privileges of the procedure owner, not the procedure user, on any objects that the procedure accesses. This enables a user to execute a procedure successfully even when that user does not have the privileges to directly access objects used by the procedure.

SQL Compliance

- SQL-93, ODBC Core SQL grammar

Environment

- Embedded SQL, interactive SQL, ODBC applications

Related Statements

- CALL, DROP PROCEDURE
CREATE SEQUENCE

Syntax
CREATE SEQUENCE [owner_name.]sequence_name
  [ START WITH start_value ]
  [ INCREMENT BY increment_value ]
  [ MAXVALUE max_value | NOMAXVALUE ]
  [ MINVALUE min_value | NOMINVALUE ]
  [ CYCLE | NOCYCLE ]

Description
The CREATE SEQUENCE command creates a new sequence in the current database.

- **sequence_name** is the name that is to be assigned to the sequence. The sequence_name in a database must be unique.
- **start_value** specifies a starting value for the sequence. The value of start_value must be between -9223372036854775808 and 9223372036854775807. The default is 1.
- **increment_value** specifies the value by which the sequence is incremented when sequence.NEXTVAL is called. Increment by can be positive or negative. The default is 1.
- **max_value** specifies the maximum value that can be returned by the sequence. If the sequence is a cycling sequence, then the next value after the max_value will be min_value for an ascending sequence and the next value after min_value will be max_value for an ascending sequence. Note that the value of increment_value may mean that the exact max_value or min_value will never be returned. The default is 9223372036854775807.
- **min_value** specifies the minimum value that can be returned by the sequence. The default min_value is -9223372036854775808.

When **CYCLE** is specified, a sequence will wrap from the max_value to the min_value for an ascending sequence or from the min_value to a max_value for a descending sequence. When **NOCYCLE** (the default) is specified, an error is returned when NEXTVAL (see Sequence Values) would exceed the max_value for an ascending sequence or the min_value for a descending sequence.

Example
The following command creates a sequence called myseq. It starts with a value of 5, increments by 5, and returns a maximum value of 50:

```
CREATE SEQUENCE myseq START WITH 5 INCREMENT BY 5 MAXVALUE 50;
```

Return Values

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Returned By</th>
</tr>
</thead>
<tbody>
<tr>
<td>-20265</td>
<td>Sequence with the same name already exists</td>
<td>CREATE SEQUENCE</td>
</tr>
<tr>
<td>-20268</td>
<td>START-WITH/CURRENT-VALUE cannot be greater than MAXVALUE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20269</td>
<td>START-WITH/CURRENT-VALUE cannot be less than MINVALUE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>Error Code</td>
<td>Message</td>
<td>Returned By</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>-20270</td>
<td>Invalid sequence MINVALUE specified</td>
<td>CREATE SEQUENCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20271</td>
<td>Invalid sequence INCREMENT specified</td>
<td>CREATE SEQUENCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20272</td>
<td>START-WITH cannot be altered in sequence</td>
<td>CREATE SEQUENCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20273</td>
<td>No options specified for ALTER SEQUENCE</td>
<td>CREATE SEQUENCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20274</td>
<td>Sequence increment has exceeded MAXVALUE</td>
<td>CREATE SEQUENCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20275</td>
<td>Sequence decrement has exceeded MINVALUE</td>
<td>CREATE SEQUENCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20276</td>
<td>Only SELECT and ALTER privileges are valid for</td>
<td>CREATE SEQUENCE</td>
</tr>
<tr>
<td></td>
<td>sequences</td>
<td>ALTER SEQUENCE</td>
</tr>
</tbody>
</table>

See c-tree Plus Error Codes (http://docs.faircom.com/doc/ctreeplus/28320.htm) for a complete listing of valid c-tree Plus error values.

**See Also**

- Sequence Values
- DROP SEQUENCE (page 220)
- ALTER SEQUENCE (page 177)
CREATE SYNONYM

Description

Creates a synonym for the table, view or synonym specified. A synonym is an alias that c-treeACE SQL statements can use instead of the name specified when the table, view, or synonym was created.

Syntax

CREATE [PUBLIC] SYNONYM synonym
    FOR [owner_name.] { table_name | view_name | synonym } ;

Arguments

PUBLIC

 Specifies that the synonym will be public: all users can refer to the name without qualifying it. By default, the synonym is private: other users must qualify the synonym by preceding it with the user name of the user who created it.

Users must have the DBA privilege to create public synonyms.

SYNONYM synonym

Name for the synonym.

FOR [owner_name.] { table_name | view_name | synonym }

Table, view, or synonym for which c-treeACE SQL creates the new synonym.

Example

CREATE SYNONYM customer FOR smith.customer ;
CREATE PUBLIC SYNONYM public_suppliers FOR smith.suppliers ;

Authorization

Users executing CREATE SYNONYM must have the DBA privilege or RESOURCE privilege. Users executing CREATE PUBLIC SYNONYM statement must have the DBA privilege.

SQL Compliance

Extension

Environment

Embedded SQL, interactive SQL, ODBC applications

Related Statements

DROP SYNONYM
CREATE TABLE

Description

Creates a table definition. A table definition consists of a list of column definitions that make up a table row. c-treeACE SQL provides two forms of the CREATE TABLE statement. The first form explicitly specifies column definitions. The second form, with the AS query_expression clause, implicitly defines the columns using the columns in the query expression.

Syntax

CREATE TABLE [ IF NOT EXISTS ] [ owner_name. ] table_name
( column_definition [ , ( column_definition | table_constraint ) ] ... )
[ STORAGE_ATTRIBUTES 'attributes' ]
;

CREATE TABLE [ IF NOT EXISTS ] [ owner_name. ] table_name
[ ( column_name [NULL | NOT NULL], ... ) ]
[ STORAGE_ATTRIBUTES 'attributes' ]
AS query_expression
;

column_definition ::=
  column_name data_type
  [ DEFAULT { literal | USER | NULL | UID | SYSDATE | SYSTIME | SYSTIMESTAMP } ]
  | [ IDENTITY [(seed, increment)] ]
  | [ column_constraint [ column_constraint ... ] ]

Arguments

IF NOT EXISTS

This argument avoids failure by creating the table only if a table of the same name does not already exist.

owner_name

Specifies the owner of the table. If the name is different from the user name of the user executing the statement, then the user must have DBA privileges.

table_name

Names the table definition. c-treeACE SQL defines the table in the database named in the last CONNECT statement.

column_name data_type

Names a column and associates a data type with it. The column names specified must be different than other column names in the table definition. The data_type must be one of the supported data types described in Data Types (page 10).

DEFAULT

Specifies an explicit default value for a column. The column takes on the value if an INSERT statement does not include a value for the column. If a column definition omits the DEFAULT clause, the default value is NULL.

The DEFAULT clause accepts the following arguments:
### SQL Statements

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>literal</strong></td>
<td>An integer, numeric or string constant.</td>
</tr>
<tr>
<td><strong>USER</strong></td>
<td>The name of the user issuing the INSERT or UPDATE statement on the table. Valid only for columns defined with character data types.</td>
</tr>
<tr>
<td><strong>NULL</strong></td>
<td>A null value.</td>
</tr>
<tr>
<td><strong>UID</strong></td>
<td>The user id of the user executing the INSERT or UPDATE statement on the table. Valid only for columns defined with character data types.</td>
</tr>
<tr>
<td><strong>SYSDATE</strong></td>
<td>The current date. Valid only for columns defined with DATE data types.</td>
</tr>
<tr>
<td><strong>SYSTIME</strong></td>
<td>The current time. Valid only for columns defined with TIME data types.</td>
</tr>
<tr>
<td><strong>SYSTIMESTAMP</strong></td>
<td>The current date and time. Valid only for columns defined with TIMESTAMP data types.</td>
</tr>
</tbody>
</table>

**IDENTITY**

Specifies that the column is an **IDENTITY** column. An **IDENTITY** column is one for which the data is not provided by the user but it is automatically generated by the DBMS. There can be only one identity column for a table. A default value can not be specified for an identity column. **IDENTITY** can only be specified on columns of type INTEGER, SMALLINT, TINYINT and BIGINT.

**seed**

Specifies a value that used for the identity column when the first row is inserted into the table. Specifying a seed value is optional. If a seed value is not specified, then it defaults to 1.

**increment**

Specifies a value by which the identity value of the previous row is incremented, which is then used for the identity column when the second and subsequent rows are inserted into the table. Specifying the increment value is optional. If it is not specified, then it defaults to 1.

Note: Although both the seed and increment values are optional, they must both be specified or both be left unspecified. It is not possible to specify only one of them.

**column_constraint**

Specifies a constraint that applies while inserting or updating a value in the associated column. For more information, see *Column Constraints* (page 201).

**table_constraint**

Specifies a constraint that applies while inserting or updating a row in the table. For more information, see *Table Constraints* (page 203).

**STORAGE_ATTRIBUTES 'attributes'**

Specifies a string describing underlying c-treeACE table storage attributes. For more information see *Storage Attributes* (page 205).

**AS query_expression**

Specifies a query expression to use for the data types and contents of the columns for the table. The types and lengths of the columns of the query expression result become the types and lengths of the respective columns in the table created. The rows in the resultant set of the query
expression are inserted into the table after creating the table. In this form of the `CREATE TABLE` statement, column names are optional.

If omitted, the names for the table columns are also derived from the query expression. For more information, see *Query Expressions* (page 30).

**Examples**

In the following example, the user issuing the `CREATE TABLE` statement must have `REFERENCES` privilege on the column “itemno” of the table “john.item”.

```sql
CREATE TABLE supplier_item (
  supp_no     INTEGER NOT NULL PRIMARY KEY,
  item_no     INTEGER NOT NULL REFERENCES john.item (itemno),
  qty         INTEGER
) ;
```

The following `CREATE TABLE` statement explicitly specifies a table owner, admin:

```sql
CREATE TABLE admin.account (
  account  integer,
  balance  money (12),
  info     char (84)
) ;
```

The following example shows the `AS query_expression` form of `CREATE TABLE` to create and load a table with a subset of the data in the `customer` table:

```sql
CREATE TABLE admin.dealer (name, street, city, state)
AS
  SELECT name, street, city, state
  FROM customer
  WHERE customer.state IN ('CA','NY', 'TX') ;
```

The following example includes a `NOT NULL` column constraint and `DEFAULT` clauses for column definitions:

```sql
CREATE TABLE emp (
  empno  integer NOT NULL,
  deptno  integer DEFAULT 10,
  join_date date DEFAULT NULL
) ;
```

**Authorization**

The user executing this statement must have either `DBA` or `RESOURCE` privilege. If the `CREATE TABLE` statement specifies a foreign key that references a table owned by a different user, the user must have the `REFERENCES` privilege on the corresponding columns of the referenced table.

The `AS query_expression` form of `CREATE TABLE` requires the user to have select privilege on all the tables and views named in the query expression.

**SQL Compliance**

- SQL-92, ODBC Minimum SQL grammar. Extensions: `AS query_expression`

**Environment**

- Embedded SQL, interactive SQL, ODBC applications

**Related Statements**

- DROP TABLE, ALTER TABLE, Query Expressions
See Also:

- Column Constraints (page 201)
- Table Constraints (page 203)
- Storage Attributes (page 205)
- \textit{SQL - Create [if not exists] and Drop [if exists]} (page 206)

\section*{Column Constraints}

\subsection*{Description}

Specifies a constraint for a column that restricts the values that the column can store. \texttt{INSERT}, \texttt{UPDATE}, or \texttt{DELETE} statements that violate the constraint fail. c-treeACE SQL returns a constraint violation error with \texttt{SQLCODE-20116}.

Column constraints are similar to table constraints but their definitions are associated with a single column.

\subsection*{Syntax}

\begin{verbatim}
column_constraint ::=  
  UNIQUE  
  |  NOT NULL [ PRIMARY KEY | UNIQUE ]  
  |  REFERENCES [ owner_name. ] table_name [ ( column_name ) ]  
  |  CHECK ( search_condition )
\end{verbatim}

\subsection*{Arguments}

\textbf{NOT NULL}

Restricts values in the column to values that are not null.

\textbf{NOT NULL PRIMARY KEY}

\texttt{PRIMARY KEY} is a valid abbreviation of \texttt{NOT NULL PRIMARY KEY}.

Defines the column as the primary key for the table. There can be only one primary key for a table. A column with the \texttt{NOT NULL PRIMARY KEY} constraint cannot contain null or duplicate values. Other tables can name primary keys as foreign keys in their \texttt{REFERENCES} clauses.

If other tables name primary keys in their \texttt{REFERENCES} clauses, c-treeACE SQL restricts operations on the table containing the primary key:

- \texttt{DROP TABLE} statements that delete the table fail
- \texttt{DELETE} and \texttt{UPDATE} statements that modify values in the column that match a foreign key’s value also fail

The following example shows the creation of a primary key column on the table supplier.

\begin{verbatim}
CREATE TABLE supplier (  
  supp_no INTEGER NOT NULL PRIMARY KEY,  
  name CHAR (30),  
  status SMALLINT,  
  city CHAR (20)  
) ;
\end{verbatim}
NOT NULL UNIQUE

Defines the column as a unique key that cannot contain null or duplicate values. Columns with NOT NULL UNIQUE constraints defined for them are also called candidate keys.

Other tables can name unique keys in their REFERENCES clauses. If they do, c-treeACE SQL restricts operations on the table containing the unique key:

- DROP TABLE statements that delete the table fail
- DELETE and UPDATE statements that modify values in the column that match a foreign key's value also fail

The following example creates a NOT NULL UNIQUE constraint to define the column ss_no as a unique key for the table employee:

```
CREATE TABLE employee (  
    empno        INTEGER NOT NULL PRIMARY KEY,  
    ss_no        INTEGER NOT NULL UNIQUE,  
    ename        CHAR (19),  
    sal          NUMERIC (10, 2),  
    deptno       INTEGER NOT NULL  
) ;
```

REFERENCES table_name [ (column_name) ]

Defines the column as a foreign key and specifies a matching primary or unique key in another table. The REFERENCES clause names the matching primary or unique key.

A foreign key and its matching primary or unique key specify a referential constraint: A value stored in the foreign key must either be null or be equal to some value in the matching unique or primary key.

You can omit the column_name argument if the table specified in the REFERENCES clause has a primary key and you want the primary key to be the matching key for the constraint.

The following example defines order_item.orditem_order_no as a foreign key that references the primary key orders.order_no.

```
CREATE TABLE orders (  
    order_no INTEGER NOT NULL PRIMARY KEY,  
    order_date DATE  
) ;
CREATE TABLE order_item (  
    orditem_order_no INTEGER REFERENCES orders ( order_no ),  
    orditem_quantity INTEGER  
) ;
```

Note that the second CREATE TABLE statement in the previous example could have omitted the column name order_no in the REFERENCES clause, since it refers to the primary key of table orders.

CHECK (search_condition)

Specifies a column-level check constraint. c-treeACE SQL restricts the form of the search condition. The search condition must not:

- Refer to any column other than the one with which it is defined
- Contain aggregate functions, subqueries, or parameter references
The following example creates a check constraint:

```sql
CREATE TABLE supplier (  
supp_no INTEGER NOT NULL,  
name CHAR (30),  
status SMALLINT,  
city CHAR (20) CHECK (supplier.city <> 'MOSCOW')
)
```

**Table Constraints**

**Description**

Specifies a constraint for a table that restricts the values that the table can store. INSERT, UPDATE, or DELETE statements that violate the constraint fail. c-treeACE SQL returns a Constraint violation error.

Table constraints have syntax and behavior similar to column constraints. Note the following differences:

- The syntax for table constraints is separated from column definitions by commas.
- Table constraints must follow the definition of columns they refer to.
- Table constraint definitions can include more than one column and c-treeACE SQL evaluates the constraint based on the combination of values stored in all the columns.

**Syntax**

```
table_constraint ::  
| PRIMARY KEY ( column [, ... ] )  
| UNIQUE ( column [, ... ] )  
| FOREIGN KEY ( column [, ... ] )  
| REFERENCES [ owner_name. ] table_name [ ( column [, ... ] ) ]  
| CHECK ( search_condition )
```

**Arguments**

**PRIMARY KEY ( column [, ... ] )**

Defines the column list as the primary key for the table. There can be at most one primary key for a table.

All the columns that make up a table-level primary key must be defined as NOT NULL, or the CREATE TABLE statement fails. The combination of values in the columns that make up the primary key must be unique for each row in the table.

Other tables can name primary keys in their REFERENCES clauses. If they do, c-treeACE SQL restricts operations on the table containing the primary key:

- DROP TABLE statements that delete the table fail
- DELETE and UPDATE statements that modify values in the combination of columns that match a foreign key's value also fail

The following example shows creation of a table-level primary key. Note that its definition is separated from the column definitions by a comma:

```sql
CREATE TABLE supplier_item (  
supp_no INTEGER NOT NULL,
```
SQL Statements

```
item_no   INTEGER NOT NULL,
qty       INTEGER NOT NULL DEFAULT 0,
             PRIMARY KEY (supp_no, item_no)
);

UNIQUE ( column [, ... ] )

Defines the column list as a unique, or candidate, key for the table. Unique key table-level constraints have the same rules as primary key table-level constraints, except that you can specify more than one UNIQUE table-level constraint in a table definition.

The following example shows creation of a table with two UNIQUE table-level constraints:

```
CREATE TABLE order_item (
    order_no    INTEGER NOT NULL,
    item_no     INTEGER NOT NULL,
    qty         INTEGER NOT NULL,
    price       MONEY NOT NULL,
    UNIQUE (order_no, item_no),
    UNIQUE (qty, price)
);

FOREIGN KEY ... REFERENCES

Defines the first column list as a foreign key and, in the REFERENCES clause, specifies a matching primary or unique key in another table.

A foreign key and its matching primary or unique key specify a referential constraint: The combination of values stored in the columns that make up a foreign key must either:

- Have at least one of the column values be null
- Be equal to some corresponding combination of values in the matching unique or primary key

You can omit the column list in the REFERENCES clause if the table specified in the REFERENCES clause has a primary key and you want the primary key to be the matching key for the constraint.

The following example defines the combination of columns student_courses.teacher and student_courses.course_title as a foreign key that references the primary key of the table courses. Note that the REFERENCES clause does not specify column names because the foreign key refers to the primary key of the courses table.

```
CREATE TABLE courses (     
    teacher        CHAR (20) NOT NULL,
    course_title   CHAR (30) NOT NULL,
    PRIMARY KEY (teacher, course_title)
);
CREATE TABLE student_courses (            
    student_id     INTEGER,
    teacher        CHAR (20),
    course_title   CHAR (30),
    FOREIGN KEY (teacher, course_title) REFERENCES courses
);
```

c-treeACE SQL evaluates the referential constraint to see if it satisfies the following search condition:

```
(student_courses.teacher IS NULL
  OR student_courses.course_title IS NULL)
OR
EXISTS (SELECT * FROM student_courses WHERE
  (student_courses.teacher = courses.teacher AND
```
student_courses.course_title = courses.course_title
}

INSERT, UPDATE or DELETE statements that cause the search condition to be false violate the constraint, fail, and generate an error.

**CHECK (search_condition)**

Specifies a table-level check constraint. The syntax for table-level and column level check constraints is identical. Table-level check constraints must be separated by commas from surrounding column definitions.

c-treeACE SQL restricts the form of the search condition. The search condition must not:

- Refer to any column other than columns that precede it in the table definition
- Contain aggregate functions, subqueries, or parameter references

The following example creates a table with two column-level check constraints and one table-level check constraint:

```sql
CREATE TABLE supplier (
  supp_no   INTEGER NOT NULL,
  name      CHAR (30),
  status    SMALLINT CHECK (supplier.status BETWEEN 1 AND 100),
  city      CHAR (20) CHECK (supplier.city IN ('NEW YORK', 'BOSTON', 'CHICAGO')),
  CHECK (supplier.city <> 'CHICAGO' OR supplier.status = 20)
) ;
```

**Storage Attributes**

A quoted string that specifies specific c-treeACE SQL table attributes. c-treeACE SQL supports the following **STORAGE_ATTRIBUTES** parameters:

- `CAMO` - A lightweight option for obscuring data. This option gives a small amount of protection from casual inspection of the c-tree data and index files. CAMO or “Camouflage” is an older, legacy method of hiding data, which is not a standards-conforming encryption scheme, such as AES. It is not intended as a replacement for Advanced Encryption or other security systems.
- `ENCR=crypt` - Advanced encryption options providing the highest level of security and protection for your data. Options include AES (Rijndael), Twofish, Blowfish, and DES. `crypt` can be one of the following:
  - AES16 (Rijndael)
  - AES24
  - AES32
  - DES8
  - DES16
  - DES24
  - BLF8 through BLF56 (Blowfish)
  - TWF16 (Twofish)
  - TWF24
• **TWF32**
  - 'RECBYT_IDX' - Creates the table with a *RECBYT* index. This index is required for physical backward traversal of variable length type files. The *RECBYT* index does impose a minimal amount of overhead, however, when inserting/updating/deleting records. This index is not required for usual operations, and as such, is off by default.
  - 'NORECBYT_IDX' - (Default) Creates the table without a *RECBYT* index.
  - 'ROWID_FLD' - (Default) The *ROWID* field is an auto-incrementing number which takes advantage of the c-tree serial segment index mode. This hidden field is a sequential number added to each record insert, and maintained with an associated index. This field is not required for proper SQL operation, however, is referenced directly by the *ROWID* related scalar functions. In the case of no *ROWID* field, the record *RECBYT* value is returned, which may not maintain uniqueness over time. As there is a limit of only one serial segment index per c-tree data file, this value is unavailable for other fields when a *ROWID* is present. Conversely, no *ROWID* is available when an existing c-tree data file with a serial segment field is imported into c-treeACE SQL. The IDENTITY field attribute should be used for auto-incrementing values as it is better performing with greater field type flexibility.
  - 'NOROWID_FLD' - Creates a table without the *ROWID* serial segment field and index.
  - 'PREIMG' - This option disables transaction logging for the table. This can be useful for temporary c-treeACE SQL tables that are not required to be recovered in the event of catastrophic failure, yet retain atomicity of transactions.
  - 'HOTALTER' - (Supported in V11.5 and later) Creates the the table with Hot Alter Table support.
  - 'NOHOTALTER' - (Supported in V11.5 and later) Create the table with Hot Alter Table support disabled.
  - 'HUGE' - (Default) Denotes that a table should be created as a c-tree *HUGE* file (64-bit file offset addressing).
  - 'NOTHUGE' - Denotes that a table should be created as a c-tree non-*HUGE* file (32-bit file offset addressing). These files have a limit of 2Gb or 4Gb depending on your OS platform.

**Note:** See the *SQL_OPTION NO_HUGEFILE* c-treeACE Server configuration option to reverse this behavior.

**Example**

To combine *STORAGE_ATTRIBUTE* options, separate them with a semicolon (;) as shown here:

```
CREATE TABLE small_preimage_table (name CHAR(10), age SMALLINT) STORAGE_ATTRIBUTES 'ENCR=TWF24;PREIMG'
```

**SQL - Create [if not exists] and Drop [if exists]**

The CREATE TABLE syntax has been expanded to avoid failure in case the table exists (during create) or does not exist (during drop). The feature is similar to syntaxes found in the MySql and Postgres dialects.

We now implement this feature for tables, indices, procedures, triggers, and functions:

```
CREATE TABLE [IF NOT EXISTS]...
```
DROP TABLE [IF EXISTS]...

CREATE INDEX [IF NOT EXISTS]...
DROP INDEX [IF EXISTS]...

CREATE PROCEDURE [IF NOT EXISTS]...
DROP PROCEDURE [IF EXISTS]...

CREATE TRIGGER [IF NOT EXISTS]...
DROP TRIGGER [IF EXISTS]...

CREATE FUNCTION [IF NOT EXISTS]...
DROP FUNCTION [IF EXISTS]...
CREATE TRIGGER

Description
Creates a trigger for the specified table. A trigger is a special type of stored procedure that helps insure referential integrity for a database.

Triggers contain Java source code which can use c-treeACE SQL Java classes to carry out database operations. Triggers are automatically activated when an INSERT, UPDATE, or DELETE statement affects the trigger’s target table. The Java source code details what actions the trigger takes when it is activated.

For more detail on creating and using triggers, see the c-treeACE SQL Guide to Using Stored Procedures and Triggers.

Syntax
```
CREATE  TRIGGER { IF NOT EXISTS } { owner_name. } trigname
{ BEFORE | AFTER } { INSERT
| DELETE
| UPDATE [ OF ( column_name [ , ... ] ) ]
ON table_name
[ REFERENCING { OLDROW | NEWROW } | NEWROW | OLDROW ]
[ FOR EACH { ROW | STATEMENT } ]
[ IMPORT
    java_import_clause ]
BEGIN
    java_snippet
END
```

Arguments
IF NOT EXISTS
This argument avoids failure by creating the trigger only if a trigger of the same name does not already exist.

owner_name
Specifies the owner of the trigger. If the name is different from the user name of the user executing the statement, then the user must have DBA privileges.

trigname
Names the trigger. DROP TRIGGER statements specify the trigger name defined here. c-treeACE SQL also uses trigname in the name of the Java class that it creates from the Java snippet.

BEFORE | AFTER
The trigger action time. Specifies whether the triggered action implemented by java_snippet executes before or after the triggering INSERT, UPDATE, or DELETE statement.

INSERT | DELETE | UPDATE [ OF column_name [ , ... ] ]
The statement that activates the trigger. Also called the trigger event.

If UPDATE is the trigger event, this clause can include an optional column list. Updates to any of the specified columns or use of a specified column in a search condition to update other values will activate the trigger. As long as a specified column is not used in either case then the trigger
will not be activated. If an UPDATE trigger does not include the optional column list, an update statement specifying any of the table columns will activate the trigger.

**ON table_name**
The name of the table for which the trigger is defined. The name of a view or a remote table. A triggering statement that specifies table_name causes the trigger to execute.

**REFERENCING { OLDROW [ , NEWROW ] | NEWROW [ , OLDROW ] }**
Allowed only if the trigger also specifies the FOR EACH ROW clause. The REFERENCING clause provides a mechanism for c-treeSQL to pass row values as input parameters to the stored procedure implemented by java_snippet. The code in java_snippet uses the getValue() method of the NEWROW and OLDROW objects to retrieve values of columns in rows affected by the trigger event and store them in procedure variables.

The meaning of the OLDROW and NEWROW arguments of REFERENCING clause depends on whether the trigger event is INSERT, UPDATE, or DELETE:

- **INSERT…REFERENCING NEWROW** means the triggered action can access values of columns of each row inserted. c-treeSQL passes the column values specified by the INSERT statement.
- **INSERT…REFERENCING OLDROW** is meaningless, since there are no existing values for a row being inserted. INSERT…REFERENCING OLDROW generates a syntax error.
- **UPDATE…REFERENCING NEWROW** means the triggered action can access the values of columns, after they are changed, of each row updated. SQL passes the column values specified by the UPDATE statement.
- **UPDATE…REFERENCING OLDROW** means the triggered action can access the values of columns, before they are changed, of each row updated. c-treeSQL passes the column values of the row as it exists in the database before the update operation.
- **DELETE…REFERENCING OLDROW** means the triggered action can access values of columns of each row deleted. c-treeSQL passes the column values of the row as it exists in the database before the delete operation.
- **DELETE…REFERENCING NEWROW** is meaningless, since there are no new existing to pass for a row being deleted. DELETE…REFERENCING OLDROW generates a syntax error.

UPDATE is the only triggering statement that allows both NEWROW and OLDROW in the REFERENCING clause.

Note that the trigger action time (BEFORE or AFTER) does not affect the meaning of the REFERENCING clause. For instance, BEFORE UPDATE…REFERENCING NEWROW still means the values of columns after they are updated will be available to the triggered action.

The REFERENCING clause generates an error if the trigger does not include the FOR EACH ROW clause.

**FOR EACH { ROW | STATEMENT }**
The frequency with which the triggered action implemented by java_snippet executes:
FOR EACH ROW means the triggered action executes once for each row being updated by the triggering statement. CREATE TRIGGER must include the FOR EACH ROW clause if it also includes a REFERENCING clause.

FOR EACH STATEMENT means the triggered action executes only once for the whole triggering statement. FOR EACH STATEMENT is the default.

IMPORT
java_import_clause

Specifies standard Java classes to import. The IMPORT keyword must be upper case and on a separate line.

BEGIN
java_snippet
END

The body of the trigger. Also called the triggered action. The body contains the Java source code that implements the actions to be completed when a triggering statement specifies the target table. The Java statements become a method in a class the c-treeACE SQL engine creates and submits to the Java compiler.

The BEGIN and END keywords must be upper case and on separate lines.

Notes

- Triggers can take action on their own table so that they invoke themselves. c-treeACE SQL limits such recursion to five levels.
- You can you have multiple triggers on the same table. c-treeACE SQL executes all triggers applicable to a given combination of table, trigger event, and action time. If more than one trigger is applicable for a particular combination, c-treeACE SQL executes the triggers in the order they were created. (You can determine the creation order from the triggerid column in the admin.systrigger system table. The higher the triggerid value, the later the trigger was created.) The interaction of multiple triggers on the same table can be confusing, so exercise care.
- The actions carried out by a trigger may fire another trigger. When this happens, the other trigger’s actions execute before the rest of the first trigger finishes executing. c-treeACE SQL limits such nesting to five levels.
- If a c-treeACE SQL statement both fires a trigger and violates a constraint (possible if the trigger action time is BEFORE), any actions taken by the trigger are rolled back and do not take effect.
- To modify an existing trigger, you must delete it and issue another CREATE TRIGGER statement. You can query the admin.systrigger and sysproctxt system tables for details of the trigger before deleting it.

Example

This example illustrates an update trigger on a table called BUG_INFO. If the STATUS or PRIORITY fields are modified, the trigger modifies the BUG_SUMMARY and BUG_STATUS tables appropriately based on some conditions.

CREATE TRIGGER BUG_UPDATE_TRIGGER
AFTER UPDATE OF STATUS, PRIORITY ON BUG_INFO
REFERENCING OLDROW, NEWROW
FOR EACH ROW
IMPORT
import java.sql.*;
BEGIN
try
{
    // column number of STATUS is 10
    String old_status, new_status;
    old_status = (String) OLDROW.getValue(10, CHAR);
    new_status = (String) NEWROW.getValue(10, CHAR);
    if (old_status.equals("OPEN") && new_status.equals("FIXED"))
    {
        // If STATUS has changed from OPEN to FIXED
        // increment the bugs_fixed_cnt by 1 in the 
        // row corresponding to current month 
        // and current year
        SQLIStatement update_stmt = new SQLIStatement("update BUG_STATUS set bugs_fixed_cnt = bugs_fixed_cnt + 1 
                                  where month = ? and year = ?");
        Integer current_month = 10;
        Integer current_year = 1997;
        update_stmt.setParam(1, current_month);
        update_stmt.setParam(2, current_year);
        update_stmt.execute();
        SQLIStatement insert_stmt = new SQLIStatement("insert into BUG_SUMMARY values (?, ?, ?)" );
        // Column number for bug_id, priority, reported_on and fixed_on 
        // are 1, 2, 5, 6
        String bug_id, priority;
        Date reported_on, fixed_on;
        bug_id = (String) NEWROW.getValue(1, CHAR);
        priority = (String) NEWROW.getValue(2, CHAR);
        reported_on = (Date) NEWROW.getValue(5, DATE);
        fixed_on = (Date) NEWROW.getValue(6, DATE);
        Integer turn_around_time = fixed_on - reported_on;
        insert_stmt.setParam(1, bug_id);
        insert_stmt.setParam(2, priority);
        insert_stmt.setParam(3, turn_around_time);
        insert_stmt.execute();
    }
    // If PRIORITY has changed to URGENT,
    // increment the bugs_escalated by 1 in the month field. 
    String old_priority, new_priority;
    new_priority = (String) NEWROW.getValue(2, CHAR);
    if (new_priority.equals("URGENT") && old_priority.equals("URGENT"))
    {
        // If PRIORITY has changed to URGENT 
        // Increment the bugs_escalated by 1 in the row corresponding to current month 
        // and current year
        SQLIStatement update_stmt = new SQLIStatement("update BUG_STATUS set bugs_escalated_cnt = bugs_escalated_cnt + 1 
                                              where month = ? and year = ?");
    }
}
Integer current_month = 10;
Integer current_year = 1997;
update_stmt.setParam(1, current_month);
update_stmt.setParam(2, current_year);
update_stmt.execute();
}
}
catch (SQLException e)
{
  // Log the exception message from e.
  SQLException sqle = new SQLException("UPDATE_BUG_TRIGGER failed");
  throw sqle;
}
END

Authorization

Users executing CREATE TRIGGER must have the DBA privilege or RESOURCE privilege.

SQL Compliance  SQL-93, ODBC Core SQL grammar

Environment  Embedded SQL, interactive SQL, ODBC applications

Related Statements  DROP TRIGGER
CREATE VIEW

Description
Creates a view with the specified name on existing tables and/or views.

Syntax
CREATE VIEW [ owner_name. ] view_name
[ ( column_name, column_name, ... ) ]
AS [ ( [ ) query_expression [ ] ]
[ WITH CHECK OPTION ] ;

Notes
- The owner_name is made the owner of the created view.
- The column names specified for the view are optional and provide an alias for the columns selected by the query specification. If the column names are not specified then the view will be created with the same column names as the tables and/or views it is based on.
- A view is deletable if deleting rows from that view is allowed. For a view to be deletable, the view definition has to satisfy the following conditions:
  - The first FROM clause contains only one table reference or one view reference.
  - There are no aggregate functions, DISTINCT clause, GROUP BY or HAVING clause in the view definition.
  - If the first FROM clause contains a view reference, then the view referred to is deletable.
- A view is updatable if updating rows from that view is allowed. For a view to be updatable, the view has to satisfy the following conditions:
  - The view is deletable (That is, it satisfies all the conditions specified above for deletability).
  - All the select expressions in the first SELECT clause of the view definition are simple column references.
  - If the first FROM clause contains a view reference, then the view referred to is updatable.
- A view is insertable if inserting rows into that view is allowed. For a view to be insertable, the view has to satisfy the following conditions:
  - The view is updatable (That is, it satisfies all the conditions specified above for updatability).
  - If the first FROM clause contains a table reference, then all NOT NULL columns of the table are selected in the first SELECT clause of the view definition.
  - If the first FROM clause contains a view reference, then the view referred to is insertable.
- The WITH CHECK OPTION clause can be specified only if the view is updatable.
- If WITH CHECK OPTION clause is specified when defining a view, then during any update or insert of a row on this view, it is checked that the updated/inserted row satisfies the view definition (That is, the row is selectable using the view).

Examples
CREATE VIEW ne_customers AS
  SELECT cust_no, name, street, city, state, zip
  FROM customer
WHERE state IN ('NH', 'MA', 'NY', 'VT')
WITH CHECK OPTION
;
CREATE VIEW order_count (cust_number, norders) AS
SELECT cust_no, COUNT(*)
FROM orders
GROUP BY cust_no;

Authorization

The user executing this statement must have the following privileges:

- DBA or RESOURCE privilege.
- SELECT privilege on all the tables/views referred to in the view definition.

If owner_name is specified and is different from the name of the user executing the statement, then the user must have DBA privilege.

SQL Compliance  SQL-92, ODBC Core SQL grammar
Environment  Embedded SQL, interactive SQL, ODBC applications
Related Statements  Query Expressions, DROP VIEW
DELETE

Description
Deletes zero, one or more rows from the specified table that satisfy the search condition specified in the WHERE clause. If the optional WHERE clause is not specified, then the DELETE statement deletes all rows of the specified table.

Syntax
DELETE FROM [owner_name.] { table_name | view_name } [ WHERE search_condition ];

Notes
• If the table has primary/candidate keys, and if there exists references from other tables to the rows to be deleted, the statement is rejected.
• While our DELETE doesn't directly support a TOP clause, you can do this with any indexed field with a subquery as such:
  DELETE FROM <table> WHERE <idxfield> <= (SELECT MAX(<idxfield>) FROM (SELECT TOP 1000 <idxfield> FROM <table>) x )

Example
DELETE FROM customer
  WHERE customer_name = 'RALPH' ;

Authorization
The user executing this statement must have any of the following privileges:
• DBA privilege.
• Ownership of the table.
• DELETE permission on the table.
If the target is a view, then the DELETE privilege is required on the target base table referred to in the view definition.

SQL Compliance
SQL-92, ODBC Extended SQL grammar

Environment
Embedded SQL, interactive SQL, ODBC applications

Related Statements
Search Conditions
**DISCONNECT**

**Description**
Terminates the connection between an application and a database environment.

**Syntax**
```sql
DISCONNECT
    { connection_name | ALL | CURRENT | DEFAULT }
connection_name::=
    { character_literal | host_variable }
```

**Notes**
- If a connection name is specified it should identify a dormant or current connection. If the `connection_name` specified is the current connection, the connection to the dormant DEFAULT database, if any, (this connection should have been previously achieved through CONNECT TO DEFAULT) is made the current connection; else no current connection exists.
- If `ALL` is specified, all established connections are disconnected. After the execution of this statement, a current connection does not exist.
- If `CURRENT` is specified, the current connection, if any, is disconnected. Here too, the connection to the dormant DEFAULT database, if any, (this connection should have been previously achieved through CONNECT TO DEFAULT) is made the current connection; else no current connection exists.
- If `DEFAULT` is specified, the DEFAULT connection, if any, is disconnected. If this connection happens to be the current connection, no current connection exists after the execution of this statement.

**Examples**
```sql
DISCONNECT 'conn_1';
DISCONNECT CURRENT;
DISCONNECT ALL;
DISCONNECT DEFAULT;
```

**Authorization**
None.

**SQL Compliance**
SQL-92

**Environment**
Embedded SQL and interactive

**Related Statements**
DISCONNECT, SET CONNECTION
DROP FUNCTION

**Description**
Deletes a User Defined Scaler Function (UDF) on the specified table.

**Syntax**
```
DROP FUNCTION [IF EXISTS] function_name
```

**Arguments**

**IF EXISTS**
This argument avoids failure by attempting to drop the function only if a function of that name already exists.

**function_name**
The user defined function to be deleted from the table.

**Example**
```
DROP FUNCTION str_cat;
```

**Authorization**
The user executing this statement must have any of the following privileges:
- DBA privilege
- Ownership of the index

**SQL Compliance**
ODBC Core SQL grammar

**Environment**
Embedded SQL, interactive SQL, ODBC and JDBC applications

**Related Statements**
CREATE FUNCTION
**DROP INDEX**

**Description**
Deletes an index on the specified table.

**Syntax**
```
DROP INDEX [IF EXISTS] [index_owner_name.]index_name
[ON [table_owner_name.]table_name]
```

**Arguments**

[IF EXISTS]
This argument avoids failure by attempting to drop the index only if an index of that name already exists.

**index_owner_name**
If `index_owner_name` is specified and is different from the name of the user executing the statement, then the user must have DBA privileges.

**table_name**
The `table_name` argument is optional. If specified, the `index_name` is verified to correspond to the table.

**Warning:** Do not DROP indexes on primary or foreign keys. This can lead to relational integrity issues. Use ALTER TABLE to perform any changes involving constraints.

**Example**
```
DROP INDEX custindex ON customer ;
```

**Authorization**
The user executing this statement must have any of the following privileges:

- DBA privilege
- Ownership of the index
- EXCLUSIVE file access is required for any this operation

**SQL Compliance**
ODBC Core SQL grammar

**Environment**
Embedded SQL, interactive SQL, ODBC applications

**Related Statements**
CREATE INDEX
DROP PROCEDURE

Description
Deletes a stored procedure.

Syntax
DROP PROCEDURE [IF EXISTS] [ owner_name. ] procedure_name ;

Arguments
IF EXISTS
This argument avoids failure by attempting to drop the procedure only if a procedure of that name already exists.

owner_name
Specifies the owner of the procedure.

procedure_name
Names of the stored procedure to delete.

Example
DROP PROCEDURE new_sal ;

Authorization
To drop a stored procedure, users must be the owner of the procedure or have the DBA privilege.

SQL Compliance
SQL-93, ODBC Core SQL grammar

Environment
Embedded SQL, interactive SQL, ODBC applications

Related Statements
CALL, CREATE PROCEDURE
DROP SEQUENCE

Syntax
DROP SEQUENCE sequence_name

Description
The DROP SEQUENCE command deletes the specified sequence.

- sequence_name is the name of the sequence that is to be deleted.

Example
To delete the sequence myseq, enter:
DROP SEQUENCE myseq;

Return Values

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Returned By</th>
</tr>
</thead>
<tbody>
<tr>
<td>-20266</td>
<td>Sequence cannot be used here</td>
<td>DROP SEQUENCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20267</td>
<td>Sequence not found</td>
<td>DROP SEQUENCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALTER SEQUENCE</td>
</tr>
</tbody>
</table>

See c-tree Plus Error Codes (http://docs.faircom.com/doc/ctreeplus/28320.htm) for a complete listing of valid c-tree Plus error values.

See Also
- Sequence Values
- CREATE SEQUENCE (page 195)
- ALTER SEQUENCE (page 177)
DROP SYNONYM

Description
Drops the specified synonym.

Syntax
DROP [PUBLIC] SYNONYM [owner_name.]synonym ;

Arguments
PUBLIC
Specifies that the synonym was created with the PUBLIC argument.

c-treeACE SQL generates the Base table not found error if DROP SYNONYM specifies PUBLIC and the synonym was not a public synonym. Conversely, the same error message occurs if DROP SYNONYM does not specify public and the synonym was created with the PUBLIC argument.

To drop a public synonym, you must have the DBA privilege.

owner_name
If owner_name is specified and is different from the name of the user executing the statement, then the user must have DBA privileges.

synonym
Name for the synonym.

Example
DROP SYNONYM customer ;
DROP PUBLIC SYNONYM public_suppliers ;

Authorization
Users executing DROP SYNONYM must have either the DBA privilege or be the owner of the synonym. Users executing DROP PUBLIC SYNONYM must have the DBA privilege.

SQL Compliance
Extension

Environment
Embedded SQL, interactive SQL, ODBC applications

Related Statements
CREATE SYNONYM
DROP TABLE

Description
Deletes the specified table.

Syntax
DROP TABLE  [IF EXISTS]  [owner_name.]table_name ;

Notes
- If owner_name is specified and is different from the name of the user executing the statement, then the user must have DBA privileges.
- When a table is dropped, the indexes on the table and the privileges associated with the table are dropped automatically.
- Views dependent on the dropped table are not automatically dropped, but become invalid.
- If the table is part of another table’s referential constraint (if the table is named in another table’s REFERENCES clause), the DROP TABLE statement fails. Use the ALTER TABLE statement to delete any referential constraints that refer to the table before issuing the DROP TABLE statement.
- IF EXISTS avoids failure by attempting to drop the table only if a table of that name already exists.

Example
DROP TABLE customer ;

Authorization
The user executing this statement must have any of the following privileges:
- DBA privilege.
- Ownership of the table.

SQL Compliance
SQL-92, ODBC Minimum SQL grammar

Environment
Embedded SQL, interactive SQL, ODBC applications

Related Statements
CREATE TABLE
**DROP TRIGGER**

**Description**
Deletes a trigger.

**Syntax**
```
DROP TRIGGER [IF EXISTS] [ owner_name. ] trigger_name;
```

**Arguments**

**IF EXISTS**
This argument avoids failure by attempting to drop the trigger only if a trigger of that name already exists.

**owner_name**
Specifies the owner of the trigger.

**trigger_name**
Names of the trigger to delete.

**Example**
```
DROP TRIGGER sal_check;
```

**Authorization**
- The **DBA** privilege entitles a user to drop any trigger.
- The owner of a trigger is given **EXECUTE** and **DROP** privilege on that trigger at creation time, by default.

**SQL Compliance**
- SQL-93, ODBC Core SQL grammar

**Environment**
- Embedded SQL, interactive SQL, ODBC applications

**Related Statements**
- CREATE TRIGGER
DROP VIEW

Description
Deletes the view from the database.

Syntax
DROP VIEW [owner_name.]view_name ;

Notes
- If owner_name is specified and is different from the name of the user executing the statement, then the user must have DBA privileges.
- When a view is dropped, other views that are dependent on this view are not dropped. The dependent views become invalid.

Example
DROP VIEW newcustomers ;

Authorization
The user executing this statement must have any of the following privileges:
- DBA privilege
- Ownership of the view

  SQL Compliance  SQL-92, ODBC Core SQL grammar
  Environment     Embedded SQL, interactive SQL, ODBC applications
  Related Statements  CREATE VIEW
EXPLAIN PLAN

Description
Generates a query execution plan for a SQL query.

Syntax
EXPLAIN PLAN [ INTO [ owner_name. ] table_name ] [ SET STATEMENT_ID = <stmt id> ] FOR <SQL statement>

Arguments

table name
The table into which the query plan is to be saved. Any valid table name may be specified. If the table does not exist, then it is created, provided the user has resource privilege. If the current owner is not the owner of the table specified he should have adequate privilege to write into it.

Note: The INTO table_name clause is optional. If it is not specified, then the query plan is stored in the default plan table admin.qep_tbl.

stmt_id
The user provided identifier for the query plan. Any character string (less than 32 characters) can be specified. The stmt_id is stored along with the query plan and is used to distinguish between the various Query plans stored in the Query plan table.

The SET STATEMENT_ID = <stmt id> clause is optional. If not specified, a unique ID of the form qep_stmtid_xx is generated internally.

Notes
For complex SQL operation, the query plan of the SQL query being executed has a very large impact on its performance. It is useful to have a mechanism wherein the execution plan of a query can be made available before actually executing the query. This aids in designing efficient queries.

c-treeACE SQL has the ability to dynamically generate and store query plans. The query plan can be generated and even viewed graphically with the c-treeACE SQL Explorer utility.

An internal stored procedure is used to retrieve the query plan.
access_get_qep( table_owner, table_name, stmt_id, max_nodeinfo_len )
- table_owner is the owner of the query plan table
- table_name is the name of the query plan table where the plan is stored
- stmt_id is the Statement Id provided by the user or the id internally generated when the execution plan was generated.
- max_nodeinfo_len (integer) - is the length of the information displayed per node in the resultset. By default this is set to 256. Set this value to 0 if the default value is to be used (8192).

The output of the procedure is a resultset consisting of a single column. Each record of the resultset corresponds to a node in the query plan. The ordering of the nodes in the resultset is, from bottom to top and left to right of the query execution plan tree.
### Authorization

If the table does not exist, then it is created, provided the user has resource privilege. If the current owner is not the owner of the table specified they should have adequate privilege to write into it.

<table>
<thead>
<tr>
<th>SQL Compliance</th>
<th>Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>Embedded SQL, interactive SQL, ODBC applications</td>
</tr>
</tbody>
</table>
GET DIAGNOSTICS

Description
Retrieves information about the execution of the previous c-treeACE SQL statement. GET DIAGNOSTICS extracts information from the c-treeACE SQL diagnostics area, a data structure that contains information about the execution status of the most recent c-treeACE SQL statement. There are two components to the diagnostics area:

- The header contains overall information about the last c-treeACE SQL statement as a whole
- The detail area contains information for a particular condition (an error, warning, or success condition) associated with execution of the last c-treeACE SQL statement. The diagnostics area can potentially contain multiple detail areas corresponding to multiple conditions generated by the c-treeACE SQL statement described by the header.

Note: The c-treeACE SQL diagnostics area currently supports only one detail area.

There are two forms of the GET DIAGNOSTICS statement, one that extracts header information (GET DIAGNOSTICS), and one that extracts detail information (GET DIAGNOSTICS EXCEPTION number).

Syntax
GET DIAGNOSTICS
:param = header_info_item [ , :param = header_info_item ] ...
GET DIAGNOSTICS EXCEPTION number
:param = detail_info_item [ , :param = detail_info_item ] ...

header_info_item ::
{ NUMBER | MORE | COMMAND_FUNCTION | DYNAMIC_FUNCTION | ROW_COUNT }

detail_info_item ::
{ CONDITION_NUMBER | RETURNED_SQLSTATE | CLASS_ORIGIN | SUBCLASS_ORIGIN | TABLE_NAME | COLUMN_NAME | MESSAGE_TEXT | MESSAGE_LENGTH }

Arguments
:parameter
A host-language variable to receive the information returned by the GET DIAGNOSTICS statement. The host-language program must declare parameter to be compatible with the SQL data type of the information item.

header_info_item
One of the following keywords, which returns associated information about the diagnostics area or the c-treeACE SQL statement:

**NUMBER**

The number of detail areas in the diagnostics area. Currently, **NUMBER** is always 1. **NUMBER** is type **NUMERIC** with a scale of 0.

**MORE**

Whether the diagnostics area contains information on all the conditions resulting from the statement. **MORE** is a one-character string with a value of Y (all conditions are detailed in the diagnostics area) or N (all conditions are not detailed).

**COMMAND_FUNCTION**

If the statement was a static c-treeACE SQL statement, contains the character-string code for the statement (as specified in the SQL-92 standard). If the statement was a dynamic statement, contains either the character string 'EXECUTE' or 'EXECUTE IMMEDIATE'.

**DYNAMIC_FUNCTION**

For dynamic c-treeACE SQL statements only (as indicated by 'EXECUTE' or 'EXECUTE IMMEDIATE' in the **COMMAND_FUNCTION** item), contains the character-string code for the statement (as specified in the SQL-92 standard).

**ROW_COUNT**

The number of rows affected by the c-treeACE SQL statement.

**EXCEPTION number**

Specifies that **GET DIAGNOSTICS** extracts detail information. number specifies which of multiple detail areas **GET DIAGNOSTICS** extracts. Currently, number must be the integer 1.

**detail_info_item**

One of the following keywords, which returns associated information about the particular error condition:

**CONDITION_NUMBER**

The sequence of this detail area in the diagnostics area. Currently, **CONDITION_NUMBER** is always 1.

**RETURNED_SQLSTATE**

The **SQLSTATE** value that corresponds to the condition. See the individual sections in *Error Messages* (page 256) for a list of **SQLSTATE** values.

**CLASS_ORIGIN**

Whether the **SQLSTATE** class code is defined by the SQL standard (indicated by the character string 'ISO 9075') or by c-treeACE SQL.

**SUBCLASS_ORIGIN**

Whether the **SQLSTATE** subclass code is defined by the c-treeACE SQL standard (indicated by the character string 'ISO 9075') or by c-treeACE SQL.

**TABLE_NAME**

If the error condition involves a table, the name of the table.

**COLUMN_NAME**

If the error condition involves a column, the name of the affected columns.

**MESSAGE_TEXT**

The associated message text for the error condition.

**MESSAGE_LENGTH**

The length in characters of the message in the **MESSAGE_TEXT** item.

**Notes**

The **GET DIAGNOSTICS** statement itself does not affect the contents of the diagnostics area. This means applications can issues multiple **GET DIAGNOSTICS** statements to retrieve different items of information about the same c-treeACE SQL statement.
Example
GET DIAGNOSTICS :num = NUMBER, :cmdfunc = COMMAND_FUNCTION
GET DIAGNOSTICS EXCEPTION :num
:ssstate = RETURNED_SQLSTATE, :msgtxt = MESSAGE_TEXT

Authorization
<table>
<thead>
<tr>
<th>SQL Compliance</th>
<th>SQL-92</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>Embedded SQL</td>
</tr>
</tbody>
</table>
GRANT

Description
Grants various privileges to the specified users or user group for the database. There are different forms of the GRANT statement for various purposes:

- To grant database-wide privileges, either system administration (DBA) or general creation (RESOURCE)
- To grant privileges on the specified tables or view
- To grant the privilege to execute the specified stored procedure

Syntax

```sql
GRANT { RESOURCE, DBA }
   TO {user_name } [ , { user_name } ] ... ;

GRANT { privilege [ , privilege ] ... | ALL [ PRIVILEGES ] }
   ON table_name
   TO { { user_name | usergroup_name } [ , {user_name | usergroup_name} ] ... | PUBLIC }
   [WITH GRANT OPTION] ;

GRANT EXECUTE ON procedure_name
   TO { { user_name | usergroup_name } [ , { user_name | usergroup_name } ] ... | PUBLIC } ;

privilege ::=
   { SELECT | INSERT | DELETE | ALTER | INDEX
   | UPDATE [ (column, column, ... ) ]
   | REFERENCES [ (column, column, ... ) ] }
```

Arguments

DBA
Allows the specified users to create, access, modify, or delete any database object, and to grant other users any privileges. This argument an only be applied to users.

RESOURCE
Allows the specified users to issue CREATE statements. The RESOURCE privilege does not allow users to issue DROP statements on database objects. Only the owner of the object and users with the DBA privilege can drop database objects. This argument an only be applied to users.

SELECT
Allows the specified users to read data in the table or view.

INSERT
Allows the specified users to add new rows to the table or view.

DELETE
Allows the specified users to delete rows in the table or view.

ALTER
Allows the specified users to modify the table or view.
INDEX
Allows the specified users to create an index on the table or view.

UPDATE [ (column, column, … ) ]
Allows the specified users to modify existing rows in the table or view. If followed by a column list, the users can modify values only in the columns named.

REFERENCES [ (column, column, … ) ]
Allows the specified users to refer to the table from other tables’ constraint definitions. If followed by a column list, constraint definitions can refer only to the columns named. For more detail on constraint definitions, see Column Constraints (page 201).

ALL
Grants all privileges for the table or view.

ON table_name
The table or view for which c-treeACE SQL grants the specified privileges.

EXECUTE ON procedure_name
Allows execution of the specified stored procedure.

TO user_name [ , user_name ] …
The list of users for which c-treeACE SQL grants the specified privileges.

TO PUBLIC
Grants the specified privileges to any user with access to the system.

WITH GRANT OPTION
Allows the specified users to grant their access rights or a subset of their rights to other users.

Examples
Note: You must commit, or turn on auto commit, to save your changes. For example, isql defaults to auto commit off, so if you grant a permission and exit, your changes are discarded.

GRANT RESOURCE TO user1;
GRANT SELECT ON custmaster TO odbc_group;
GRANT ALTER ON cust_view TO dbuser1 ;
GRANT SELECT ON newcustomers TO dbuser2 ;
GRANT EXECUTE ON sample_proc TO searle;
commit;

Authorization
The user granting DBA or RESOURCE privileges must have the DBA privilege.
The user granting privileges on a table must have any of the following privileges:

- DBA privilege
- Ownership of the table
- All the specified privileges on the table, granted with the WITH GRANT OPTION clause
| SQL Compliance                  | SQL-92, ODBC Core SQL grammar. Extensions: ALTER, INDEX, RESOURCE, DBA privileges |
| Environment                    | Embedded SQL, interactive SQL, ODBC applications |
| Related Statements             | REVOKE |
**INSERT**

**Description**

Inserts new rows into the specified table/view that will contain either the explicitly specified values or the values returned by the query expression.

**Syntax**

```sql
INSERT INTO [owner_name.] { table_name | view_name }
[ (column_name, column_name, ... ) ]
{ VALUES (value, value, ... ) | query_expression };
```

**Notes**

- If the optional list of column names is specified, then only the values for those columns need be supplied. The rest of the columns of the inserted row will contain NULL values, provided the table definition allows NULL values and there is no `DEFAULT` clause for the columns. If a `DEFAULT` clause is specified for a column and the column name is not present in the optional column list, then the column takes the default value.
- If the optional list is not specified then all the column values have to be either explicitly specified or returned by the query expression. The order of the values should be the same as the order in which the columns have been declared in the declaration of the table/view.
- Explicit specification of the column values provides for insertion of only one row at a time. The query expression option allows for insertion of multiple rows at a time.
- If the table contains a foreign key, and there does not exist a corresponding primary key that matches the values of the foreign key in the record being inserted, the insert operation is rejected.
- You can use `INSERT` statements with query expressions to transfer rows from one remote table to another.
- `IDENTITY` columns are populated automatically based on the seed and increment values that were specified for the `IDENTITY` column, values for the `IDENTITY` column can not be specified as part of the `INSERT` statement. An `IDENTITY` column can not be specified in the list of column names.

**Examples**

```sql
INSERT INTO customer (cust_no, name, street, city, state)
VALUES
(1001, 'RALPH', '#10 Columbia Street', 'New York', 'NY') ;
```

```sql
INSERT INTO neworders (order_no, product, qty)
SELECT order_no, product, qty
FROM orders
WHERE order_date = SYSDATE ;
```

**Authorization**

The user executing this statement must have any of the following privileges:

- DBA privilege.
- Ownership of the table.
- INSERT privilege on the table.

If a query_expression is specified, then the user must have any of the following privileges:
- DBA privilege.
- SELECT privilege on all the tables/views referred to in the query_expression.

**SQL Compliance**
- SQL-92, ODBC Core SQL grammar

**Environment**
- Embedded SQL, interactive SQL, ODBC applications

**Related Statements**
- Query Expressions
LOCK TABLE

Description
Explicitly locks the specified tables for shared or exclusive access.

Syntax
LOCK TABLE table_name [ , ... ]
   IN ( SHARE | EXCLUSIVE ) MODE
   [ NOWAIT ] ;

Notes
- Explicit locking can be used to improve the performance of a single transaction at the cost of decreasing the concurrency of the system and potentially blocking other transactions. It is more efficient to explicitly lock a table if you know ahead of time that the transaction would be updating a substantial part of a table. The efficiency is gained by decreased overhead of the implicit locking mechanism and any potential waits for acquiring page level locks for the table.
- Explicit locking can be used to minimize potential deadlocks in situations where a substantial part of a table is being modified by a transaction. The benefits of table locking should always be compared with the disadvantages of losing concurrency before a choice is made between explicit and implicit locking.
- The SHARE mode allows other transactions to read the table but does not allow modifications on the table.
- The EXCLUSIVE mode does not allow any other transactions to read and/or modify the table.
- If the lock request cannot be honored by the system (due to a conflict lock held by another transaction) then in the normal case the transaction is suspended until the specified lock can be acquired. The NOWAIT option provides an immediate return of control if the lock cannot be acquired.
- Locks that are acquired explicitly and/or implicitly are released only when the transaction is ended using either the COMMIT or the ROLLBACK WORK statement.

Example
LOCK TABLE custmaster
   IN EXCLUSIVE MODE ;

Authorization
The user executing this statement must have any of the following privileges:
- DBA, or INSERT, UPDATE, DELETE, or ALTER privilege.

SQL Compliance
SQL-92

Environment
Embedded SQL, interactive SQL

Related Statements
SELECT, INSERT, DELETE
**ORDER BY**

**Description**
The **ORDER BY** clause specifies the sorting of rows retrieved by the **SELECT** statement. c-treeACE SQL does not guarantee the sort order of rows unless the **SELECT** statement includes an **ORDER BY** clause.

**Syntax**

```sql
ORDER BY { expr | posn } [ ASC | DESC ]
   [ , { expr | posn } [ASC | DESC] , ... ]
```

**Notes**
- Ascending order is the default ordering. The descending order will be used only if the keyword **DESC** is specified for that column.
- Each **expr** is an expression of one or more columns of the tables specified in the **FROM** clause of the **SELECT** statement. Each **posn** is a number identifying the column position of the columns being selected by the **SELECT** statement.
- The selected rows are ordered on the basis of the first **expr** or **posn** and if the values are the same then the second **expr** or **posn** is used in the ordering.
- The **ORDER BY** clause if specified should follow all other clauses of the **SELECT** statement.
- An **ORDER BY** clause can appear in a subquery, however, subqueries with a combination of **TOP**, **GROUP BY** and outer references remain unsupported.
- An **ORDER BY** clause can appear in a **FOR UPDATE** query.

**Note:** This is a non-standard SQL feature as specified by SQL92. Also, full cursor update is not supported by c-treeACE SQL.

- A query expression followed by an optional **ORDER BY** clause can be specified. In such a case, if the query expression contains set operators, then the **ORDER BY** clause can specify column names or position from the first **SELECT** statement.

For example:

```sql
-- Get a merged list of customers and suppliers
-- sorted by their name.
(SELECT name, street, state, zip
 FROM customer
 UNION
 SELECT name, street, state, zip
 FROM supplier)
 ORDER BY 1 ;
```

```sql
(SELECT name, street, state, zip
 FROM customer
 UNION
 SELECT name, street, state, zip
 FROM supplier)
 ORDER BY customer.name;
```

- In V11 and later, a query projecting an LVARBINARY column using an **ORDER BY** clause that requires sorting in memory may result in projecting null values. Prior to this change,
LVARCHAR was the only supported long type. The logic has been updated to handle LVARBINARY.

Example

SELECT name, street, city, state, zip
FROM customer
ORDER BY name;
RENAME

Description
Renames the specified table name, view name or synonym to the new name specified.

Syntax
RENAME [owner_name.] oldname TO [owner_name.] newname ;

Arguments
[owner_name.]
Optional owner-name qualifier for the name. If the owner name is not the same as that of the current user, the current user must have the DBA privilege.
If specified, the owner name must be the same for oldname and newname. In other words, you cannot change the owner of a table, view, or synonym with RENAME.

oldname
Current name of the table, view, or synonym.

newname
New name for the table, view, or synonym.

Example
RENAME sitem TO supplier_item ;

Authorization
The user executing this statement must have any of the following privileges:

- DBA privilege
- Ownership of the table/view/synonym.
- ALTER privilege on the table/view.

SQL Compliance

<table>
<thead>
<tr>
<th>Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded SQL, interactive SQL, ODBC applications</td>
</tr>
</tbody>
</table>

Related Statements
CREATE TABLE, CREATE VIEW, CREATE SYNONYM
**REVOKE**

**Description**

Revokes various privileges to the specified users for the database. There are three forms of the `REVOKE` statement:

- The first form revokes database-wide privileges, either system administration (DBA) or general creation (RESOURCE)
- The second form revokes various privileges on specific tables and views
- The third form revokes the privilege to execute the specified stored procedure

**Syntax**

```sql
REVOKE { RESOURCE | DBA } FROM { {user_name | usergroup_name} [ , {user_name | usergroup_name} ] ... } ;
```

```sql
REVOKE [ GRANT OPTION FOR ] { privilege [ , privilege, ] ... | ALL [ PRIVILEGES ] } ON table_name FROM { {user_name | usergroup_name} [ , {user_name | usergroup_name} ] ... | PUBLIC } [ RESTRICT | CASCADE ] ;
```

```sql
REVOKE [ GRANT OPTION FOR ] EXECUTE ON procedure_name FROM { {user_name | usergroup_name} [ , {user_name | usergroup_name} ] ... | PUBLIC } [ RESTRICT | CASCADE ] ;
```

**Arguments**

**GRANT OPTION FOR**

Revolves the grant option for the privilege from the specified users. The actual privilege itself is not revoked. If specified with `RESTRICT`, and the privilege was passed on to other users, the `REVOKE` statement fails and generates an error. Otherwise, `GRANT OPTION FOR` implicitly revokes any rights the user may have in turn given to other users.

```sql
{ privilege [ , privilege, ] ... | ALL [ PRIVILEGES ] }
```

List of privileges to be revoked. See the description in `GRANT` (page 230) for details on specific privileges. Revoking RESOURCE and DBA rights can only be done by the administrator or a user with DBA rights.

If a user has been granted access to a table by more than one user then all the users have to perform a revoke for the user to lose his access to the table.

Using the keyword `ALL` revokes all the rights granted on the table/view.

**ON table_name**

The table or view for which c-treeACE SQL revokes the specified privileges.

**EXECUTE ON procedure_name**
Revoke the right to execute the specified stored procedure.

```sql
FROM {user_name | usergroup_name} [ , {user_name | usergroup_name} ] ...
```

Revoke the specified rights on the table or view from the specified list of users.

### FROM PUBLIC

Revoke the specified rights on the table or view from any user with access to the system.

### RESTRICT | CASCADE

If the `REVOKE` statement specifies `RESTRICT`, c-treeACE SQL checks to see if the privilege being revoked was passed on to other users (possible only if the original privilege included the `WITH GRANT OPTION` clause). If so, the `REVOKE` statement fails and generates an error. If the privilege was not passed on, the `REVOKE` statement succeeds.

If the `REVOKE` statement specifies `CASCADE`, revoking the access right of a user also revokes the rights from all users who received the privilege as a result of that user giving the privilege to others.

If the `REVOKE` statement specifies neither `RESTRICT` nor `CASCADE`, the behavior is the same as for `CASCADE`.

### Example

```sql
REVOKE INSERT ON customer FROM dbuser1 ;
REVOKE ALTER ON cust_view FROM dbuser2 ;
```

### Authorization

The user revoking `DBA` or `RESOURCE` privileges must have the `DBA` privilege.

The user revoking privileges on a table must have any of the following privileges:

- `DBA` privilege
- Ownership of the table
- All the specified privileges on the table, granted with the `WITH GRANT OPTION` clause

### SQL Compliance

- SQL-92, ODBC Core SQL grammar. Extensions: ALTER, INDEX, RESOURCE, DBA privileges

### Environment

- Embedded SQL, interactive SQL, ODBC applications

### Related Statements

- GRANT
ROLLBACK WORK

Description
Ends the current transaction and undoes any database changes performed during the transaction.

Syntax
ROLLBACK [ WORK ] ;

Notes
Under certain circumstances, c-treeACE SQL marks a transaction for abort but does not actually roll it back. Without an explicit rollback, any subsequent updates will not take effect, since a COMMIT statement cause c-treeACE SQL to recognize the transaction as marked for abort, and instead implicitly rolls back the transaction. c-treeACE SQL marks a transaction for abort under these conditions:

- Hardware or software system failures
- Lock timeout errors

Authorization
None.

SQL Compliance
SQL-92.

Environment
Embedded SQL and interactive

Related Statements
COMMIT WORK
SELECT

Description
Selects the specified column values from one or more rows contained in the table(s) specified in the FROM clause. The selection of rows is restricted by the WHERE clause. The temporary table derived through the clauses of a select statement is called a result table.

The format of the SELECT statement is a query expression with optional ORDER BY and FOR UPDATE clauses. For more detail on query expressions, see Query Expressions (page 30).

Syntax
select_statement ::
    query_expression
    ORDER BY { expr | posn } [ ASC | DESC ]
    [ , { expr | posn } [ASC | DESC],... ]
    FOR UPDATE [ OF [table].column_name, ... ] [ NOWAIT ];
query_expression ::
    query_specification
    | query_expression set_operator query_expression
    | { query_expression }
set_operator ::= { UNION [ ALL ] | INTERSECT | MINUS }
query_specification ::= SELECT [ ALL | DISTINCT ] [ SKIP N ] [ TOP N ]
    { *
    | { table_name | alias } . * [ , { table_name | alias } . * ]...
    | { ( expr | NULL ) [ [ AS ] [ ' ] column_title [ ' ] ] }
    | { [ ' ] column_title [ ' ] = } { expr | NULL }
    } ,
    { { expr | NULL } [ [ AS ] [ ' ] column_title [ ' ] ] }
    | { [ ' ] column_title [ ' ] = } { expr | NULL }
    }...
FROM table_ref [ { ctree ORDERED } ] [ , table_ref [ { ctree ORDERED } ] ]...
WHERE search_condition
[ GROUP BY [table.]column_name
    [ , [table.]column_name ]...
[ HAVING search_condition ]
table_ref ::
    table_name [ AS ] [ alias [ { column_alias [ , ... ] } ] ]
    | ( query_expression ) [ AS ] alias [ { column_alias [ , ... ] } ]
    | [ ( ) joined_table ]
    | procedure_name(proc_arg [, ...]) [ AS ] [ alias ]
joined_table ::
    table_ref CROSS JOIN table_ref
    | table_ref [ INNER | LEFT [ OUTER ] ] JOIN table_ref ON search_condition
Arguments

query_expression

See Query Expressions (page 30).

expr

See Expressions (page 20)

expr can also be a scalar sub-query.

See Scalar Sub-query Expressions (page 25)

ORDER BY clause

See ORDER BY CLAUSE (page 236)

FOR UPDATE clause

NOWAIT

c-tree supports non-blocking locks with an info type of UPDATE_NOWAIT_SUPPORTED, which returns TRUE or FALSE based on whether the storage system supports NOWAIT or not. If the storage system supports NOWAIT, then SELECT . . . FOR UPDATE statements use a new fetch hint TPL_FH_WRITE_WITH_NOWAIT instead of TPL_FH_WRITE. If the row is already locked, instead of waiting (as the old logic would have done), the logic returns the ELCK_UPDATE_NOWAIT error and execution of the statement stops.

Note that the code does not return ELCK_UPDATE_NOWAIT; it returns c-tree error 42 mapped into -17042. The code does not have any check on ELCK_UPDATE_NOWAIT.

procedure_name

A procedure used in a table_ref must return a result_set and may not have any OUT or IN/OUT parameters. That is, only IN parameters are supported.

Note: Use of ORDER BY in a FOR UPDATE clause is a non-standard SQL feature as specified by SQL92. Also, full cursor update is not supported by c-treeACE SQL. See UPDATE (page 251).

Authorization

The user executing this statement must have any of the following privileges:

- DBA privilege
- SELECT permission on all the tables/views referred to in the query_expression.

SQL Compliance


Environment

Embedded SQL (within DECLARE), interactive SQL, ODBC applications

Related Statements

Query Expressions, DECLARE CURSOR, OPEN, FETCH, CLOSE
SET CONNECTION

Description
SET CONNECTION sets the database associated with the connection name as the current database.

Syntax
SET CONNECTION connection_name ;
connection_name::=
    { character_literal | host_variable | DEFAULT }

Notes
If DEFAULT is specified, there should exist a DEFAULT connection (this could have been previously achieved through a CONNECT TO DEFAULT statement). All c-treeACE SQL statements are executed for the current database.

Examples
SET CONNECTION 'salesdb';
SET CONNECTION DEFAULT;

Authorization
None.

SQL Compliance
SQL-92

Environment
Embedded SQL and interactive

Related Statements
DISCONNECT, SET CONNECTION
**SET IDENTITY_INSERT**

**Description**
Applies or removes the IDENTITY_INSERT property to the specified table.

**Syntax**
```
SET IDENTITY_INSERT tablename { ON | OFF }
```

**Arguments**
- `tablename` - The name of a table with an identity column.

**Notes:**
Only one table in a session can have the IDENTITY_INSERT property set to ON. If a table already has this property set to ON, and a SET IDENTITY_INSERT ON statement is issued for another table, an error will be returned.

If the value inserted is larger than the current identity value for the table, the server uses the new inserted value as the current identity value.

The setting of SET IDENTITY_INSERT is applied at execute time, not at parse time.

**Permissions**
User must own the table or have ALTER permission on the table.
SET TRANSACTION ISOLATION

Description
Explicitly sets the isolation level for a transaction. Isolation levels specify the degree to which one transaction can modify data or database objects being used by another concurrent transaction.

Syntax
SET TRANSACTION ISOLATION LEVEL  isolation_level ;

isolation_level ::
    READ UNCOMMITTED
    | READ COMMITTED
    | REPEATABLE READ
    | Serializable

Arguments
READ UNCOMMITTED
Allows dirty reads, non-repeatable reads, and phantoms (described below in Notes).

READ COMMITTED
Default. Prohibits dirty reads; allows non-repeatable reads and phantoms.

REPEATABLE READ
Prohibits dirty reads and non-repeatable reads; allows phantoms.

SERIALIZABLE
Prohibits dirty reads, non-repeatable reads, and phantoms (see the following notes). It guarantees that concurrent transactions will not affect each other; they behave as if they were executing serially, not concurrently.

Notes
SET TRANSACTION allows the user to choose the isolation level for future transactions. If a transaction is currently active, SET TRANSACTION generates an error.

The isolation level specifies the degree to which one transaction is isolated from the effects of concurrent access of the database by other transactions. The appropriate level of isolation depends on how a transaction needs to be isolated from effects of another transaction. Higher isolation levels provide greater data consistency to the user’s transaction but reduce access to data by concurrent transactions.

The isolation level SERIALIZABLE guarantees the highest consistency. The isolation level READ UNCOMMITTED guarantees the least consistency. Only READ COMMITTED and REPEATABLE READ are supported. The ANSI/ISO standard defines isolation levels in terms of the of the inconsistencies they allow, as detailed next:
Permitted Inconsistencies in Transactions

Dirty read
Allows the transaction to read a row that has been inserted or modified by another transaction, but not committed. If the other transaction rolls back its changes, the transaction will have read a row that never existed, in the sense that it was never committed.

Non-repeatable read
Allows the transaction to read a row that another transaction modifies or deletes before the next read operation. If the other transaction commits the change, the transaction will receive modified values, or discover the row is deleted, on subsequent read operations.

Phantom
Allows the transaction to read a range of rows that satisfies a given search condition. If another transaction adds rows before a second read operation using the same search condition, then the transaction receives a different collection of rows with the same search condition.

Authorization
None.

SQL Compliance
SQL-92.

Environment
Embedded SQL and interactive

Related Statements
COMMIT, ROLLBACK
SET SCHEMA

Description
SET SCHEMA specifies a new default qualifier for database object names (database objects include tables, indexes, views, synonyms, procedures, and triggers).

When you connect to a database with a particular user name, that name becomes the default qualifier for database object names. This means you do not have to qualify references to tables, for instance, that were created under the same user name. However, you must qualify references to all other tables with the user name of the user who created them.

SET SCHEMA allows you to change the user name that c-treeACE SQL uses as the default qualifier for database object names. The name specified in SET SCHEMA becomes the new default qualifier for object names.

Note: SET SCHEMA does not change your user name or affect authentication. It only changes the default qualifier.

Syntax
SET SCHEMA ' qualifier_name ' ;

Arguments
' qualifier_name ' 
The new qualifier name is enclosed in single quotation marks.

Notes
- SET SCHEMA does not check whether qualifier_name is a valid user name.
- Metadata for objects created without an explicit qualifier will show qualifier_name as the owner.
- SET SCHEMA does not start or end a transaction.

Examples
The following interactive SQL example shows changing the default qualifier through SET SCHEMA. The example:
- Invokes ISQL as the user admin, the owner of the system catalog tables
- Queries the systables catalog tables as admin
- Uses SET SCHEMA to change the default qualifier to fred
- Creates a table and queries systables to show that the newly-created table is owned by fred

ISQL> -- What is the user name for the current connection?
ISQL> select user() from syscalctable;
ADMIN
------
admin
1 record selected
ISQL> -- Show the name and owner of non-system tables:
ISQL> select tbl, owner from systables where tbltype <> 'S';
TBL       OWNER

Note: SET SCHEMA does not change your user name or affect authentication. It only changes the default qualifier.
---  -----  
t1     admin

test   admin

test   ctree

3 records selected

ISQL> set schema 'fred';
ISQL> create table freds_table (c1 int);
ISQL> create index freds_table_ix on freds_table (c1);
ISQL> select tbl, owner from systables where tbltype <> 'S';

*  
error(-20005): Table/View/Synonym not found

ISQL> -- Oops! Must now qualify references to the admin-owned tables:
ISQL> select tbl, owner from admin.systables where tbltype <> 'S';

---  -----  
t1     admin

test   admin

test   ctree

tfreds_table   fred

4 records selected

Authorization

None.

SQL Compliance  SQL-92

Environment  Embedded SQL and interactive

Related Statements  None
TRUNCATE TABLE

Description
Delete all the rows of a table in a single action.

(In V10.3 and later) Any table truncated using TRUNCATE will have the $DELFLD$ set to 4 bytes despite the size in the original table.

Syntax
```
TRUNCATE TABLE [ table_name ]
```

Arguments
- `table_name`
  Specifies a single table to truncate.

Authorization
The user must have the DBA privilege or SELECT privilege on all the tables in the database. To issue the TRUNCATE TABLE statement for a specific table, the user must be the owner or have UPDATE privilege on the table.

SQL Compliance
- Extension

Environment
- Embedded SQL, interactive SQL, ODBC applications
**UPDATE**

**Description**

Updates the columns of the specified table with the given values that satisfy the `search_condition`.

**Syntax**

```
UPDATE table_name
    SET assignment, assignment, ...
    [ WHERE search_condition ]
```

```
assignment ::
    column = { expr | NULL }
  | ( column, column, ... ) = ( expr, expr, ... )
  | ( column, column, ... ) = ( query_expression )
```

**Arguments**

If the optional `WHERE` clause is specified, then only rows that satisfy the `search_condition` are updated. If the `WHERE` clause is not specified then all rows of the table are updated.

The expressions in the `SET` clause are evaluated for each row of the table if they are dependent on the columns of the target table.

If a query expression is specified on the right hand side for an assignment, the number of expressions in the first `SELECT` clause of the query expression must be the same as the number of columns listed on the left hand side of the assignment.

If a query expression is specified on the right hand side for an assignment, the query expression must return one row.

Expressions other than the query expressions mentioned above can be scalar subqueries. A scalar sub-query returns one value. The expressions and search conditions can also contain scalar sub-queries. Refer to *Scalar Sub-query Expressions* (page 25) for more information on the scalar sub-queries.

If a table has check constraints and if the columns to be updated are part of a check expression, then the check expression is evaluated. If the result of evaluation is FALSE, the UPDATE statement fails.

If a table has primary/candidate keys and if the columns to be updated are part of the primary/candidate key, a check is made as to whether there exists any corresponding row in the referencing table. If so, the UPDATE operation fails.

IDENTITY columns cannot be updated and hence cannot be specified as one of the column names for assignment.

**Examples**

```
UPDATE orders
    SET qty = 12000
    WHERE order_no = 1001 ;
```

```
UPDATE orders
    SET (product) =
        (SELECT item_name
         FROM items
```
WHERE item_no = 2401
)
WHERE order_no = 1002 ;
UPDATE orders
SET (amount) = (2000 * 30)
WHERE order_no = 1004 ;
UPDATE orders
SET (product, amount) =
(SELECT item_name, price * 30
   FROM items
   WHERE item_no = 2401
 )
WHERE order_no = 1002 ;
UPDATE orders
SET status='delivered'
WHERE EXISTS (SELECT status
   FROM orders o,customers c
   WHERE o.cust_id=c.id AND c.name='FairCom');
UPDATE orders
SET product = (SELECT item_name
   FROM items
   WHERE item_no = 2401),
   amount = (SELECT price * 30
   FROM items
   WHERE item_no = 2401)
WHERE order_no = 1002 ;
UPDATE item
SET stock = stock + (SELECT SUM(order_item.quantity)
   FROM order_item
   WHERE order_item.order_no = 341
   AND order_item.item_no = item.item_no)
WHERE item_no = (SELECT MAX(item_no)
   FROM order_item
   WHERE order_no = 341);

Authorization
The user executing this statement must have:

- **DBA** privilege.
- **UPDATE** privilege on all the specified columns of the target table and **SELECT** privilege on all the other tables referred to in the statement.

**SQL Compliance**

SQL-92, ODBC Extended SQL grammar. Extensions: assignments of the form (column, column, ...) = (expr, expr, ...)

**Environment**

Embedded SQL, interactive SQL, ODBC applications

**Related Statements**

SELECT, OPEN, FETCH, search conditions, query expressions
UPDATE STATISTICS

Description
Queries system tables and updates table and column statistics:
- The number of rows in the table (the cardinality)
- The approximate number of occurrences of a value in each column

The optimizer uses the information from UPDATE STATISTICS to calculate a query strategy for a particular c-treeACE SQL statement.

Until a user, application, or c-treeACE SQL script issues an UPDATE STATISTICS statement, the optimizer bases query strategies on values it generates from various defaults. These values will not lead to the best performance, so it is good practice for database administrators to periodically update statistics.

UPDATE STATISTICS only works on tables that have indexes defined on them.

Syntax
UPDATE STATISTICS [ FOR table_name ]

Arguments
table_name
Specifies a single table on which to update statistics. The default is to update statistics on all tables in the database.

Authorization
To issue the UPDATE STATISTICS statement for all tables in the database, the user must have DBA privilege or SELECT privilege on all the tables in the database. To issue the UPDATE STATISTICS statement for a specific table, the user must be the owner or have SELECT privilege on the table.

SQL Compliance  Extension
Environment  Embedded SQL, interactive SQL, ODBC applications
Related Statements  SET DISPLAY COST ON (interactive SQL)
3. **c-treeACE SQL Reserved Words**

Frequently, an application developer chooses a logically apt column name, however, this name happens to collide with a c-treeACE SQL reserved word. Surrounding identifiers in double quotes ensures they are properly interpreted as a table name, column name, or other identifier.

**Example**

```sql
CREATE TABLE "version" ("character" VARCHAR(10), "hour" VARCHAR(10));
INSERT INTO "version" VALUES ("Cinderella", "midnight");
SELECT "character", "hour" FROM "version";
```

To ensure your application is always prepared for future versions of c-treeACE SQL it could be considered prudent to always quote identifiers, thus avoiding issues with future reserved words.

A complete list of Reserved Words follows.

### 3.1 **c-treeACE SQL Reserved Words**

<table>
<thead>
<tr>
<th>A</th>
<th>a, abs, acos, add, add_months, after, all, alter, an, and, any, array, as, asci, asin, atan, atan2, avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>before, begin, between, bigint, binary, bind, binding, bit, blob, both, by</td>
</tr>
<tr>
<td>C</td>
<td>call, cascade, case, cast, ceiling, char, char_length, character, character_length, chartorowid, check, chr, cleanup, close, clustered, coalesce, colgroup, collate, column, commit, complex, compress, concat, connect, constraint, contains, continue, convert, cos, col, count, create, cross, curdate, current, current_date, current_user, cursor, curtime, current_time, current_timestamp, cvar</td>
</tr>
<tr>
<td>D</td>
<td>database, datapages, date, dateadd, datediff, dayname, dayofmonth, dayofweek, dayofyear, db_name, dba, dec, decimal, declaration, declare, decode, default, definition, degrees, delete, desc, describe, descriptor, dhtype, difference, distinct, double, drop</td>
</tr>
<tr>
<td>E</td>
<td>each, else, end, escape, exclusive, exec, execute, exists, exit, exp, explain, explicit, extract</td>
</tr>
<tr>
<td>F</td>
<td>fetch, field, file, float, floor, for, foreign, found, from, full, function</td>
</tr>
<tr>
<td>G</td>
<td>go, goto, grant, greatest, group</td>
</tr>
<tr>
<td>H</td>
<td>hash, having, hour</td>
</tr>
<tr>
<td>I</td>
<td>identified, identity, ifnull, immediate, in, index, indexpages, indicator, initcap, inner, inout, input, insert, instr, int, integer, interface, intersect, into, is, isnull</td>
</tr>
<tr>
<td>J</td>
<td>join</td>
</tr>
<tr>
<td>K</td>
<td>key</td>
</tr>
<tr>
<td>L</td>
<td>large, last_day, lcase, least, leading, left, length, like, link, list, localtime, localtime, locatortime, locatortimestamp, locate, lock, log, log10, long, lower, lpad, ltrim, lvarchar, lvarchar</td>
</tr>
<tr>
<td>M</td>
<td>main, max, metadata_only, min, minus, minute, mod, mode, modify, money, month, monthname, months_between</td>
</tr>
<tr>
<td>N</td>
<td>national, natural, nchar, nclob, newrow, next_day, nocompress, not, now, nowait, null, nullif, nullvalue, number, numeric, nvarchar, nvl nvarchar</td>
</tr>
</tbody>
</table>
### c-treeACE SQL Reserved Words

<table>
<thead>
<tr>
<th>O</th>
<th>object, object_id, odbc_convert, odbcinfo, of, oldrow, on, open, option, or, order, out, outer, output</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>pctfree, percent, pi, placing, plan, position, power, precision, prefix, prepare, primary, privileges, procedure, public</td>
</tr>
<tr>
<td>Q</td>
<td>quarter</td>
</tr>
<tr>
<td>R</td>
<td>radians, rand, range, raw, real, record, recursive, references, referencing, rename, repeat, replace, resource, restrict, result, return, returns, revoke, right, rollback, row, rowid, rowidtochar, rownum, rpad, rtrim</td>
</tr>
<tr>
<td>S</td>
<td>schema, searched_case, second, section, select, service, set, session_user, share, short, sign, simple_case, sin, size, smallint, some, soundex, space, sql, sql_bigint, sql_binary, sql_bit, sql_char, sql_date, sql_decimal, sql_double, sql_float, sql_integer, sql_longvarbinary, sql_longvarchar, sql_numeric, sql_real, sql_smallint, sql_time, sql_timestamp, sql_tinyint, sql_tsi_day, sql_tsi_frac_second, sql_tsi_hour, sql_tsi_minute, sql_tsi_month, sql_tsi_quarter, sql_tsi_second, sql_tsi_week, sql_tsi_year, sql_varbinary, sql_varchar, sqlerror, sqlwarning, sort, start, statement, statement_id, statistics, storage_attributes, storage_manager, substr, substring, suffix, sum, synonym, sysdate, systime, systimestamp</td>
</tr>
<tr>
<td>T</td>
<td>table, tan, then, ties, time, timeout, timestamp, timestampadd, timestampdiff, tinyint, to, top, to_char, to_date, to_number, to_time, to_timestamp, tpe, trailing, transaction, translate, trigger, trim, truncate, type</td>
</tr>
<tr>
<td>U</td>
<td>ucase, uid, union, unique, unsigned, update, upper, user, user_id, user_name, using, uuid</td>
</tr>
<tr>
<td>V</td>
<td>values, varbinary, varchar, variables, varying, version, view</td>
</tr>
<tr>
<td>W</td>
<td>week, when, whenever, where, with, work</td>
</tr>
<tr>
<td>Y</td>
<td>year</td>
</tr>
</tbody>
</table>

All Rights Reserved
4. Error Messages

This appendix lists the error messages generated by the various components of c-treeACE SQL. In addition to the c-treeACE SQL-specific error codes, error conditions have an associated SQLSTATE value, a 5-character status parameter that indicates the condition status returned by the most recent c-treeACE SQL statement. The first two characters of SQLSTATE specify the class code and the last three characters specify the subclass code:

- Class codes beginning with A-H and 0-4 are reserved by the SQL standard. For those class codes only, subclass codes beginning with A-H and 0-4 are also reserved by the standard.
- Class codes beginning with I-Z and 5-9 are specific to database implementations such as c-treeACE SQL. All subclass codes in those classes are implementation defined.

Error Codes Used by c-treeACE SQL Components

The following sections list the c-treeACE SQL error messages and show the corresponding SQLSTATE values:

- 0, 10000 Series (page 257)
- 11000 Series - c-treeACE SQL RDS Errors (page 258)
- 15000 Series - c-treeACE SQL Flat-File Storage System (page 259)
- 160xx - Main Memory Storage System (page 260)
- 16xxx - Latte and Polka Subsystems (page 262)
- ctree FSS Errors:
  - 170xx (page 264)  174xx (page 272)  178xx (page 283)
  - 171xx (page 268)  175xx (page 276)  179xx (page 288)
  - 173xx (page 272)  177xx (page 279)  18xxx (page 291)
- 20000 Series - c-treeACE SQL (page 293)
- 21000 Series - c-treeDB FSS DB Layer (page 304)
- 22000 Series - c-treeDB FSS DB Layer (page 310)
- 23000 Series - c-treeACE SQL Data Exceptions (page 315)
- 250xx - c-treeACE SQL ODBC Integrator (page 315)
- 251xx - c-treeACE SQL ODBC Driver (page 318)
- 26000 Series - c-treeACE SQL JDBC Driver (page 320)
- 30000 Series - c-treeACE SQL Network (page 320)
- 40000 Series - c-treeACE SQL Environment Errors (page 323)
- 50000 Series - c-treeACE SQL DHRSS errors (page 323)
## 4.1 0, 10000 Series

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000</td>
<td>Successful completion</td>
<td>***status okay</td>
</tr>
<tr>
<td>10002</td>
<td>02503</td>
<td>No data</td>
<td>Tuple not found for the specified TID.</td>
</tr>
<tr>
<td>10012</td>
<td>n0n12</td>
<td>Flag</td>
<td>ETPL_SCAN_EOP flag is set.</td>
</tr>
<tr>
<td>10013</td>
<td>02514</td>
<td>No data</td>
<td>No more records to be fetched.</td>
</tr>
<tr>
<td>100</td>
<td>02000</td>
<td>No data</td>
<td>**SQL not found.</td>
</tr>
<tr>
<td>10100</td>
<td>2150b</td>
<td>Cardinality violation</td>
<td>Too many fields specified.</td>
</tr>
<tr>
<td>10101</td>
<td>02701</td>
<td>No data</td>
<td>No more records exist.</td>
</tr>
<tr>
<td>10102</td>
<td>5050c</td>
<td>c-treeACE SQL RDS error</td>
<td>Duplicate record specified.</td>
</tr>
<tr>
<td>10104</td>
<td>22505</td>
<td>Data exception</td>
<td>Field size is too high.</td>
</tr>
<tr>
<td>10106</td>
<td>m0m06</td>
<td>c-treeACE SQL RSS error</td>
<td>Specified index method is not supported.</td>
</tr>
<tr>
<td>10107</td>
<td>n0n07</td>
<td>Flag</td>
<td>EIX_SCAN_EOP flag is set.</td>
</tr>
<tr>
<td>10108</td>
<td>2350i</td>
<td>Integrity constraint</td>
<td>Duplicate primary/index key value.</td>
</tr>
<tr>
<td>10301</td>
<td>m030a</td>
<td>c-treeACE SQL RSS error</td>
<td>Table is locked and LCKF_NOWAIT.</td>
</tr>
<tr>
<td>10309</td>
<td>m030a</td>
<td>c-treeACE SQL RSS error</td>
<td>Row locked and NOWAIT specified.</td>
</tr>
<tr>
<td>10400</td>
<td>22501</td>
<td>Data exception</td>
<td>Invalid file size for alter log statement.</td>
</tr>
<tr>
<td>10920</td>
<td>22521</td>
<td>Data exception</td>
<td>Already existing value specified.</td>
</tr>
</tbody>
</table>
## 4.2 11000 Series - c-treeACE SQL RDS Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>11100</td>
<td>5050b</td>
<td>c-treeACE SQL RDS error</td>
<td>Invalid transaction ID.</td>
</tr>
<tr>
<td>11102</td>
<td>5050d</td>
<td>c-treeACE SQL RDS error</td>
<td>TDS area specified is not found.</td>
</tr>
<tr>
<td>11103</td>
<td>50504</td>
<td>c-treeACE SQL RDS error</td>
<td>TDS not found for binding.</td>
</tr>
<tr>
<td>11104</td>
<td>50505</td>
<td>c-treeACE SQL RDS error</td>
<td>Transaction aborted.</td>
</tr>
<tr>
<td>11105</td>
<td>50506</td>
<td>c-treeACE SQL RDS error</td>
<td>Active Transaction error.</td>
</tr>
<tr>
<td>11109</td>
<td>50510</td>
<td>c-treeACE SQL RDS error</td>
<td>Invalid Transaction handle.</td>
</tr>
<tr>
<td>11111</td>
<td>50912</td>
<td>c-treeACE SQL RDS error</td>
<td>Invalid isolation level.</td>
</tr>
<tr>
<td>11112</td>
<td>50913</td>
<td>c-treeACE SQL RDS error</td>
<td>Isolation level cannot be changed within active transaction.</td>
</tr>
<tr>
<td>11300</td>
<td>m0m00</td>
<td>c-treeACE SQL RSS error</td>
<td>Specified INFO type is not supported.</td>
</tr>
<tr>
<td>11301</td>
<td>m0m01</td>
<td>c-treeACE SQL RSS error</td>
<td>Specified index type is not supported.</td>
</tr>
</tbody>
</table>
## 15000 Series - c-treeACE SQL Flat-File Storage System

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>15001</td>
<td>60601</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- File IO error</td>
</tr>
<tr>
<td>15002</td>
<td>60602</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- No more records</td>
</tr>
<tr>
<td>15003</td>
<td>42603</td>
<td>Access violation error</td>
<td>FF- Table already exists</td>
</tr>
<tr>
<td>15004</td>
<td>22604</td>
<td>Data exception</td>
<td>FF- Invalid record number</td>
</tr>
<tr>
<td>15005</td>
<td>60605</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- Record deleted</td>
</tr>
<tr>
<td>15006</td>
<td>60606</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- Invalid type</td>
</tr>
<tr>
<td>15007</td>
<td>60607</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- Duplicate value</td>
</tr>
<tr>
<td>15008</td>
<td>08608</td>
<td>Connection exception</td>
<td>FF- Database exists</td>
</tr>
<tr>
<td>15009</td>
<td>08609</td>
<td>Connection exception</td>
<td>FF- No database found</td>
</tr>
<tr>
<td>15010</td>
<td>60610</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- Version mismatch</td>
</tr>
<tr>
<td>15011</td>
<td>60611</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- Virtual file cache exceeded</td>
</tr>
<tr>
<td>15012</td>
<td>60612</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- Physical file open error</td>
</tr>
<tr>
<td>15013</td>
<td>60613</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- Corrupt virtual file handle</td>
</tr>
<tr>
<td>15014</td>
<td>22614</td>
<td>Data exception</td>
<td>FF- Overflow error</td>
</tr>
<tr>
<td>15021</td>
<td>60615</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- dbm_calls not implemented</td>
</tr>
<tr>
<td>15024</td>
<td>60616</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- Data exceeds max field length supported</td>
</tr>
<tr>
<td>15026</td>
<td>60616</td>
<td>c-treeACE SQL FF errors</td>
<td>FF- Max columns per table exceeded</td>
</tr>
</tbody>
</table>
## 4.4 160xx - Main Memory Storage System

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>16001</td>
<td>22701</td>
<td>Data exception</td>
<td>MM- No data block</td>
</tr>
<tr>
<td>16002</td>
<td>70702</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Bad swap block</td>
</tr>
<tr>
<td>16003</td>
<td>70703</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- No cache block</td>
</tr>
<tr>
<td>16004</td>
<td>22704</td>
<td>Data exception</td>
<td>MM- Invalid row number</td>
</tr>
<tr>
<td>16005</td>
<td>70705</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Invalid cache block</td>
</tr>
<tr>
<td>16006</td>
<td>70706</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Bad swap file</td>
</tr>
<tr>
<td>16007</td>
<td>70707</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Row too big</td>
</tr>
<tr>
<td>16008</td>
<td>70708</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Array initialized</td>
</tr>
<tr>
<td>16009</td>
<td>70709</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Invalid chunk number</td>
</tr>
<tr>
<td>16010</td>
<td>70710</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Can't create table</td>
</tr>
<tr>
<td>16011</td>
<td>70711</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Can't alter table</td>
</tr>
<tr>
<td>16012</td>
<td>70712</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Can't drop table</td>
</tr>
<tr>
<td>16020</td>
<td>70713</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- TPL ctor error</td>
</tr>
<tr>
<td>16021</td>
<td>70714</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Insertion error</td>
</tr>
<tr>
<td>16022</td>
<td>70715</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Deletion error</td>
</tr>
<tr>
<td>16023</td>
<td>70716</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Updation error</td>
</tr>
<tr>
<td>16024</td>
<td>70717</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Fetching error</td>
</tr>
<tr>
<td>16025</td>
<td>70718</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Sorting error</td>
</tr>
<tr>
<td>16026</td>
<td>70719</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Printing error</td>
</tr>
<tr>
<td>16027</td>
<td>70720</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- TPLSCAN ctor error</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>16028</td>
<td>70721</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Scan fetching error</td>
</tr>
<tr>
<td>16030</td>
<td>70722</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Can't create index</td>
</tr>
<tr>
<td>16031</td>
<td>70723</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Can't drop index</td>
</tr>
<tr>
<td>16032</td>
<td>70724</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- IXSCAN ctor error</td>
</tr>
<tr>
<td>16033</td>
<td>70725</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- IX ctor error</td>
</tr>
<tr>
<td>16034</td>
<td>70726</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- IX deletion error</td>
</tr>
<tr>
<td>16035</td>
<td>70727</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- IX appending error</td>
</tr>
<tr>
<td>16036</td>
<td>70728</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- IX insertion error</td>
</tr>
<tr>
<td>16037</td>
<td>70729</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- IX scan fetching error</td>
</tr>
<tr>
<td>16040</td>
<td>70730</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Begin transaction</td>
</tr>
<tr>
<td>16041</td>
<td>70731</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Commit transaction</td>
</tr>
<tr>
<td>16042</td>
<td>40000</td>
<td>Transaction rollback</td>
<td>***MM- Rollback transaction</td>
</tr>
<tr>
<td>16043</td>
<td>70732</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Mark point</td>
</tr>
<tr>
<td>16044</td>
<td>70733</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Rollback savepoint</td>
</tr>
<tr>
<td>16045</td>
<td>70734</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- Set &amp; Get isolation</td>
</tr>
<tr>
<td>16050</td>
<td>70735</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- TID to char</td>
</tr>
<tr>
<td>16051</td>
<td>70736</td>
<td>c-treeACE SQL MM errors</td>
<td>MM- char to TID</td>
</tr>
</tbody>
</table>
## 4.5 16000 Series - Latte and Polka Subsystems

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>16100</td>
<td>70800</td>
<td>ctree Latte error</td>
<td>0</td>
</tr>
<tr>
<td>16101</td>
<td>70801</td>
<td>ctree Latte error</td>
<td>1</td>
</tr>
<tr>
<td>16102</td>
<td>70802</td>
<td>ctree Latte error</td>
<td>2</td>
</tr>
<tr>
<td>16103</td>
<td>70803</td>
<td>ctree Latte error</td>
<td>3</td>
</tr>
<tr>
<td>16104</td>
<td>70804</td>
<td>ctree Latte error</td>
<td>4</td>
</tr>
<tr>
<td>16105</td>
<td>70805</td>
<td>ctree Latte error</td>
<td>5</td>
</tr>
<tr>
<td>16106</td>
<td>70806</td>
<td>ctree Latte error</td>
<td>6</td>
</tr>
<tr>
<td>16107</td>
<td>70807</td>
<td>ctree Latte error</td>
<td>7</td>
</tr>
<tr>
<td>16108</td>
<td>70808</td>
<td>ctree Latte error</td>
<td>8</td>
</tr>
<tr>
<td>16109</td>
<td>70809</td>
<td>ctree Latte error</td>
<td>9</td>
</tr>
<tr>
<td>16110</td>
<td>70810</td>
<td>ctree Latte error</td>
<td>10</td>
</tr>
<tr>
<td>16111</td>
<td>70811</td>
<td>ctree Latte error</td>
<td>11</td>
</tr>
<tr>
<td>16112</td>
<td>70812</td>
<td>ctree Latte error</td>
<td>12</td>
</tr>
<tr>
<td>16113</td>
<td>70813</td>
<td>ctree Latte error</td>
<td>13</td>
</tr>
<tr>
<td>16114</td>
<td>70814</td>
<td>ctree Latte error</td>
<td>14</td>
</tr>
<tr>
<td>16300</td>
<td>70815</td>
<td>ctree Latte error</td>
<td>15</td>
</tr>
<tr>
<td>16301</td>
<td>70816</td>
<td>ctree Latte error</td>
<td>16</td>
</tr>
<tr>
<td>16302</td>
<td>70817</td>
<td>ctree Latte error</td>
<td>17</td>
</tr>
<tr>
<td>16303</td>
<td>70818</td>
<td>ctree Latte error</td>
<td>18</td>
</tr>
<tr>
<td>16304</td>
<td>70819</td>
<td>ctree Latte error</td>
<td>19</td>
</tr>
<tr>
<td>16305</td>
<td>70820</td>
<td>ctree Latte error</td>
<td>20</td>
</tr>
<tr>
<td>16306</td>
<td>70821</td>
<td>ctree Latte error</td>
<td>21</td>
</tr>
<tr>
<td>16307</td>
<td>70822</td>
<td>ctree Latte error</td>
<td>22</td>
</tr>
<tr>
<td>16308</td>
<td>70823</td>
<td>ctree Latte error</td>
<td>23</td>
</tr>
<tr>
<td>16309</td>
<td>70824</td>
<td>ctree Latte error</td>
<td>24</td>
</tr>
<tr>
<td>16310</td>
<td>70825</td>
<td>ctree Latte error</td>
<td>25</td>
</tr>
<tr>
<td>16311</td>
<td>70826</td>
<td>ctree Latte error</td>
<td>26</td>
</tr>
<tr>
<td>16312</td>
<td>70827</td>
<td>ctree Latte error</td>
<td>27</td>
</tr>
<tr>
<td>16313</td>
<td>70828</td>
<td>ctree Latte error</td>
<td>28</td>
</tr>
<tr>
<td>16314</td>
<td>70829</td>
<td>ctree Latte error</td>
<td>29</td>
</tr>
<tr>
<td>16315</td>
<td>70830</td>
<td>ctree Latte error</td>
<td>30</td>
</tr>
<tr>
<td>16316</td>
<td>70831</td>
<td>ctree Latte error</td>
<td>31</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>16317</td>
<td>70832</td>
<td>ctree Latte error</td>
<td>32</td>
</tr>
<tr>
<td>16318</td>
<td>70833</td>
<td>ctree Latte error</td>
<td>33</td>
</tr>
<tr>
<td>16319</td>
<td>70834</td>
<td>ctree Latte error</td>
<td>34</td>
</tr>
<tr>
<td>16320</td>
<td>70835</td>
<td>ctree Latte error</td>
<td>35</td>
</tr>
<tr>
<td>16321</td>
<td>70836</td>
<td>ctree Latte error</td>
<td>36</td>
</tr>
<tr>
<td>16500</td>
<td>70837</td>
<td>ctree Polka error</td>
<td>37</td>
</tr>
<tr>
<td>16501</td>
<td>70838</td>
<td>ctree Polka error</td>
<td>38</td>
</tr>
<tr>
<td>16502</td>
<td>70839</td>
<td>ctree Polka error</td>
<td>39</td>
</tr>
<tr>
<td>16503</td>
<td>70840</td>
<td>ctree Polka error</td>
<td>40</td>
</tr>
<tr>
<td>16504</td>
<td>70841</td>
<td>ctree Polka error</td>
<td>41</td>
</tr>
<tr>
<td>16505</td>
<td>70842</td>
<td>ctree Polka error</td>
<td>42</td>
</tr>
<tr>
<td>16506</td>
<td>70843</td>
<td>ctree Polka error</td>
<td>43</td>
</tr>
<tr>
<td>16507</td>
<td>70844</td>
<td>ctree Polka error</td>
<td>44</td>
</tr>
<tr>
<td>16508</td>
<td>70845</td>
<td>ctree Polka error</td>
<td>45</td>
</tr>
<tr>
<td>16509</td>
<td>70846</td>
<td>ctree Polka error</td>
<td>46</td>
</tr>
<tr>
<td>16510</td>
<td>70847</td>
<td>ctree Polka error</td>
<td>47</td>
</tr>
<tr>
<td>16511</td>
<td>70848</td>
<td>ctree Polka error</td>
<td>48</td>
</tr>
<tr>
<td>16512</td>
<td>70849</td>
<td>ctree Polka error</td>
<td>49</td>
</tr>
</tbody>
</table>
## 4.6 c-tree FSS Errors

### 170xx

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>17001</td>
<td>00001</td>
<td>FSS ctree error</td>
<td>CT - General error</td>
</tr>
<tr>
<td>17002</td>
<td>00002</td>
<td>FSS ctree error</td>
<td>CT - Key value already exists in index %s</td>
</tr>
<tr>
<td>17003</td>
<td>00003</td>
<td>FSS ctree error</td>
<td>CT - Could not delete since pointers don't match</td>
</tr>
<tr>
<td>17004</td>
<td>00004</td>
<td>FSS ctree error</td>
<td>CT - Could not find key to delete</td>
</tr>
<tr>
<td>17005</td>
<td>00005</td>
<td>FSS ctree error</td>
<td>CT - Cannot call delete w/o verification with duplicate keys</td>
</tr>
<tr>
<td>17006</td>
<td>00006</td>
<td>FSS ctree error</td>
<td>CT - c-tree(...) jump table error</td>
</tr>
<tr>
<td>17007</td>
<td>00007</td>
<td>FSS ctree error</td>
<td>CT - Terminate user</td>
</tr>
<tr>
<td>17008</td>
<td>00008</td>
<td>FSS ctree error</td>
<td>CT - File open failed due to conflicting file access modes (exclusive/shared)</td>
</tr>
<tr>
<td>17009</td>
<td>00009</td>
<td>FSS ctree error</td>
<td>CT - File open or create failed due to device access error</td>
</tr>
<tr>
<td>17010</td>
<td>00010</td>
<td>FSS ctree error</td>
<td>CT - INTREE parameters require too much space</td>
</tr>
<tr>
<td>17011</td>
<td>00011</td>
<td>FSS ctree error</td>
<td>CT - Bad INTREE parameters</td>
</tr>
<tr>
<td>17012</td>
<td>00012</td>
<td>FSS ctree error</td>
<td>CT - Could not open file - not there or locked</td>
</tr>
<tr>
<td>17013</td>
<td>00013</td>
<td>FSS ctree error</td>
<td>CT - Unknown file type</td>
</tr>
<tr>
<td>17014</td>
<td>00014</td>
<td>FSS ctree error</td>
<td>CT - File corrupt at open</td>
</tr>
<tr>
<td>17015</td>
<td>00015</td>
<td>FSS ctree error</td>
<td>CT - File has been compacted</td>
</tr>
<tr>
<td>17016</td>
<td>00016</td>
<td>FSS ctree error</td>
<td>CT - Could not create index file</td>
</tr>
<tr>
<td>17017</td>
<td>00017</td>
<td>FSS ctree error</td>
<td>CT - Could not create data file</td>
</tr>
<tr>
<td>17018</td>
<td>00018</td>
<td>FSS ctree error</td>
<td>CT - Tried to create existing index file</td>
</tr>
<tr>
<td>17019</td>
<td>00019</td>
<td>FSS ctree error</td>
<td>CT - Tried to create existing data file</td>
</tr>
<tr>
<td>17020</td>
<td>00020</td>
<td>FSS ctree error</td>
<td>CT - Key length too large for node size</td>
</tr>
<tr>
<td>17021</td>
<td>00021</td>
<td>FSS ctree error</td>
<td>CT - Record length too small</td>
</tr>
<tr>
<td>17022</td>
<td>00022</td>
<td>FSS ctree error</td>
<td>CT - File number out of range</td>
</tr>
<tr>
<td>17023</td>
<td>00023</td>
<td>FSS ctree error</td>
<td>CT - Illegal index member info</td>
</tr>
<tr>
<td>17024</td>
<td>00024</td>
<td>FSS ctree error</td>
<td>CT - Could not close file</td>
</tr>
<tr>
<td>17025</td>
<td>00025</td>
<td>FSS ctree error</td>
<td>CT - Bad link in deleted node list. REBUILD</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>17026</td>
<td>00026</td>
<td>FSS ctree error</td>
<td>CT - File number not active</td>
</tr>
<tr>
<td>17027</td>
<td>00027</td>
<td>FSS ctree error</td>
<td>CT - drn before beginning of data records</td>
</tr>
<tr>
<td>17028</td>
<td>00028</td>
<td>FSS ctree error</td>
<td>CT - Zero drn in ADDKEY</td>
</tr>
<tr>
<td>17029</td>
<td>00029</td>
<td>FSS ctree error</td>
<td>CT - Zero drn in data file routine</td>
</tr>
<tr>
<td>17030</td>
<td>00030</td>
<td>FSS ctree error</td>
<td>CT - drn exceeds logical end of file</td>
</tr>
<tr>
<td>17031</td>
<td>00031</td>
<td>FSS ctree error</td>
<td>CT - Flag not set on record in delete chain</td>
</tr>
<tr>
<td>17032</td>
<td>00032</td>
<td>FSS ctree error</td>
<td>CT - Attempt to delete record twice in a row</td>
</tr>
<tr>
<td>17033</td>
<td>00033</td>
<td>FSS ctree error</td>
<td>CT - Attempt to use NULL pointer in read/write</td>
</tr>
<tr>
<td>17034</td>
<td>00034</td>
<td>FSS ctree error</td>
<td>CT - Predecessor repeat attempts exhausted</td>
</tr>
<tr>
<td>17035</td>
<td>00035</td>
<td>FSS ctree error</td>
<td>CT - Seek error - check sysiocod value</td>
</tr>
<tr>
<td>17036</td>
<td>00036</td>
<td>FSS ctree error</td>
<td>CT - Read error - check sysiocod error</td>
</tr>
<tr>
<td>17037</td>
<td>00037</td>
<td>FSS ctree error</td>
<td>CT - Write error - check sysiocod error</td>
</tr>
<tr>
<td>17038</td>
<td>00038</td>
<td>FSS ctree error</td>
<td>CT - Could not convert virtual open to actual</td>
</tr>
<tr>
<td>17039</td>
<td>00039</td>
<td>FSS ctree error</td>
<td>CT - No more records available</td>
</tr>
<tr>
<td>17040</td>
<td>00040</td>
<td>FSS ctree error</td>
<td>CT - Index node size too large</td>
</tr>
<tr>
<td>17041</td>
<td>00041</td>
<td>FSS ctree error</td>
<td>CT - Could not unlock data record</td>
</tr>
<tr>
<td>17042</td>
<td>00042</td>
<td>FSS ctree error</td>
<td>CT - Could not obtain data record lock</td>
</tr>
<tr>
<td>17043</td>
<td>00043</td>
<td>FSS ctree error</td>
<td>CT - Version incompatibility</td>
</tr>
<tr>
<td>17044</td>
<td>00044</td>
<td>FSS ctree error</td>
<td>CT - Data file serial number overflow</td>
</tr>
<tr>
<td>17045</td>
<td>00045</td>
<td>FSS ctree error</td>
<td>CT - Key length exceeds MAXLEN parameter</td>
</tr>
<tr>
<td>17046</td>
<td>00046</td>
<td>FSS ctree error</td>
<td>CT - File number already in use</td>
</tr>
<tr>
<td>17047</td>
<td>00047</td>
<td>FSS ctree error</td>
<td>CT - c-tree has not been initialized</td>
</tr>
<tr>
<td>17048</td>
<td>00048</td>
<td>FSS ctree error</td>
<td>CT - Operation incompatible with type of file</td>
</tr>
<tr>
<td>17049</td>
<td>00049</td>
<td>FSS ctree error</td>
<td>CT - Could not save file</td>
</tr>
<tr>
<td>17050</td>
<td>00050</td>
<td>FSS ctree error</td>
<td>CT - Could not lock node</td>
</tr>
<tr>
<td>17051</td>
<td>00051</td>
<td>FSS ctree error</td>
<td>CT - Could not unlock node</td>
</tr>
<tr>
<td>17052</td>
<td>00052</td>
<td>FSS ctree error</td>
<td>CT - Variable length keys disabled OR invalid key type</td>
</tr>
<tr>
<td>17053</td>
<td>00053</td>
<td>FSS ctree error</td>
<td>CT - File mode inconsistent with c-tree config</td>
</tr>
<tr>
<td>17054</td>
<td>00054</td>
<td>FSS ctree error</td>
<td>CT - Attempt to write a read only file</td>
</tr>
<tr>
<td>17055</td>
<td>00055</td>
<td>FSS ctree error</td>
<td>CT - File deletion failed</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>---------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>17056</td>
<td>00056</td>
<td>FSS ctree error</td>
<td>CT - File must be opened exclusive for delete</td>
</tr>
<tr>
<td>17057</td>
<td>00057</td>
<td>FSS ctree error</td>
<td>CT - Proper lock is not held (CHECKLOCK/READ)</td>
</tr>
<tr>
<td>17058</td>
<td>00058</td>
<td>FSS ctree error</td>
<td>CT - LOADKEY called with incorrect key number. You cannot continue</td>
</tr>
<tr>
<td>17059</td>
<td>00059</td>
<td>FSS ctree error</td>
<td>CT - LOADKEY called with key out of order You may skip this key &amp; continue</td>
</tr>
<tr>
<td>17060</td>
<td>00060</td>
<td>FSS ctree error</td>
<td>CT - Percent out of range</td>
</tr>
<tr>
<td>17061</td>
<td>00061</td>
<td>FSS ctree error</td>
<td>CT - NULL fcib detected during I/O</td>
</tr>
<tr>
<td>17062</td>
<td>00062</td>
<td>FSS ctree error</td>
<td>CT - File must be opened exclusively</td>
</tr>
<tr>
<td>17063</td>
<td>00063</td>
<td>FSS ctree error</td>
<td>CT - Start file / log file serial number error</td>
</tr>
<tr>
<td>17064</td>
<td>00064</td>
<td>FSS ctree error</td>
<td>CT - Checkpoint past end of log file</td>
</tr>
<tr>
<td>17065</td>
<td>00065</td>
<td>FSS ctree error</td>
<td>CT - Not enough memory during tran processing</td>
</tr>
<tr>
<td>17066</td>
<td>00066</td>
<td>FSS ctree error</td>
<td>CT - Log file entry failed to find checkpoint</td>
</tr>
<tr>
<td>17067</td>
<td>00067</td>
<td>FSS ctree error</td>
<td>CT - Could not rename file</td>
</tr>
<tr>
<td>17068</td>
<td>00068</td>
<td>FSS ctree error</td>
<td>CT - Could not allocate memory for control list</td>
</tr>
<tr>
<td>17069</td>
<td>00069</td>
<td>FSS ctree error</td>
<td>CT - Node does not belong to index</td>
</tr>
<tr>
<td>17070</td>
<td>00070</td>
<td>FSS ctree error</td>
<td>CT - Transaction already pending</td>
</tr>
<tr>
<td>17071</td>
<td>00071</td>
<td>FSS ctree error</td>
<td>CT - No active transaction</td>
</tr>
<tr>
<td>17072</td>
<td>00072</td>
<td>FSS ctree error</td>
<td>CT - No space for shadow buffer</td>
</tr>
<tr>
<td>17073</td>
<td>00073</td>
<td>FSS ctree error</td>
<td>CT - LOGFIL encountered during shadow only</td>
</tr>
<tr>
<td>17074</td>
<td>00074</td>
<td>FSS ctree error</td>
<td>CT - Recovery - two active tran for user</td>
</tr>
<tr>
<td>17075</td>
<td>00075</td>
<td>FSS ctree error</td>
<td>CT - Recovery - bad tran owner</td>
</tr>
<tr>
<td>17076</td>
<td>00076</td>
<td>FSS ctree error</td>
<td>CT - Recovery - bad tran type</td>
</tr>
<tr>
<td>17077</td>
<td>00077</td>
<td>FSS ctree error</td>
<td>CT - Recovery - file name too long</td>
</tr>
<tr>
<td>17078</td>
<td>00078</td>
<td>FSS ctree error</td>
<td>CT - Transaction abandoned - too many log extents or dynamic dump wait exhausted</td>
</tr>
<tr>
<td>17079</td>
<td>00079</td>
<td>FSS ctree error</td>
<td>CT - Could not log file opn/cre/cls/del</td>
</tr>
<tr>
<td>17080</td>
<td>00080</td>
<td>FSS ctree error</td>
<td>CT - NULL target or bad keyno</td>
</tr>
<tr>
<td>17081</td>
<td>00081</td>
<td>FSS ctree error</td>
<td>CT - Transaction allocation error</td>
</tr>
<tr>
<td>17082</td>
<td>00082</td>
<td>FSS ctree error</td>
<td>CT - User allocation error</td>
</tr>
<tr>
<td>17083</td>
<td>00083</td>
<td>FSS ctree error</td>
<td>CT - ISAM allocation error</td>
</tr>
<tr>
<td>17084</td>
<td>00084</td>
<td>FSS ctree error</td>
<td>CT - Maximum users exceeded</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17085</td>
<td>00085</td>
<td>FSS ctree error</td>
<td>CT - Reduce lock to read lock after update</td>
</tr>
<tr>
<td>17086</td>
<td>00086</td>
<td>FSS ctree error</td>
<td>CT - Dead lock detected</td>
</tr>
<tr>
<td>17087</td>
<td>00087</td>
<td>FSS ctree error</td>
<td>CT - System not quiet - files in use</td>
</tr>
<tr>
<td>17088</td>
<td>00088</td>
<td>FSS ctree error</td>
<td>CT - Linked list memory allocation error</td>
</tr>
<tr>
<td>17089</td>
<td>00089</td>
<td>FSS ctree error</td>
<td>CT - Memory allocation during tran processing</td>
</tr>
<tr>
<td>17090</td>
<td>00090</td>
<td>FSS ctree error</td>
<td>CT - Could not create queue</td>
</tr>
<tr>
<td>17091</td>
<td>00091</td>
<td>FSS ctree error</td>
<td>CT - Queue write error</td>
</tr>
<tr>
<td>17092</td>
<td>00092</td>
<td>FSS ctree error</td>
<td>CT - Queue memory error during write</td>
</tr>
<tr>
<td>17093</td>
<td>00093</td>
<td>FSS ctree error</td>
<td>CT - Queue read error</td>
</tr>
<tr>
<td>17094</td>
<td>00094</td>
<td>FSS ctree error</td>
<td>CT - Pending error - cannot save or commit tran</td>
</tr>
<tr>
<td>17095</td>
<td>00095</td>
<td>FSS ctree error</td>
<td>CT - Could not start task</td>
</tr>
<tr>
<td>17096</td>
<td>00096</td>
<td>FSS ctree error</td>
<td>CT - Start-file/log open error</td>
</tr>
<tr>
<td>17097</td>
<td>00097</td>
<td>FSS ctree error</td>
<td>CT - Bad user handle</td>
</tr>
<tr>
<td>17098</td>
<td>00098</td>
<td>FSS ctree error</td>
<td>CT - Bad transaction mode</td>
</tr>
<tr>
<td>17099</td>
<td>00099</td>
<td>FSS ctree error</td>
<td>CT - Transaction type / filmod conflict</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>17100</td>
<td>00100</td>
<td>FSS ctree error</td>
<td>CT - No current record for isam datno</td>
</tr>
<tr>
<td>17101</td>
<td>00101</td>
<td>FSS ctree error</td>
<td>CT - Could not find isam keyno request</td>
</tr>
<tr>
<td>17102</td>
<td>00102</td>
<td>FSS ctree error</td>
<td>CT - Could not open ISAM parameter file</td>
</tr>
<tr>
<td>17103</td>
<td>00103</td>
<td>FSS ctree error</td>
<td>CT - Could not read first 5 parameters in ISAM parameter file</td>
</tr>
<tr>
<td>17104</td>
<td>00104</td>
<td>FSS ctree error</td>
<td>CT - Too many files in ISAM parameter file</td>
</tr>
<tr>
<td>17105</td>
<td>00105</td>
<td>FSS ctree error</td>
<td>CT - Could not undo ISAM update. Rebuild Files</td>
</tr>
<tr>
<td>17106</td>
<td>00106</td>
<td>FSS ctree error</td>
<td>CT - Could not read data file record in ISAM parameter file</td>
</tr>
<tr>
<td>17107</td>
<td>00107</td>
<td>FSS ctree error</td>
<td>CT - Too many keys for data file in ISAM parameter file</td>
</tr>
<tr>
<td>17108</td>
<td>00108</td>
<td>FSS ctree error</td>
<td>CT - Incorrect keyno for index member in parameter file</td>
</tr>
<tr>
<td>17109</td>
<td>00109</td>
<td>FSS ctree error</td>
<td>CT - Too many key segments defined in ISAM parameter file</td>
</tr>
<tr>
<td>17110</td>
<td>00110</td>
<td>FSS ctree error</td>
<td>CT - Could not read segment record in ISAM parameter file</td>
</tr>
<tr>
<td>17111</td>
<td>00111</td>
<td>FSS ctree error</td>
<td>CT - Could not read index file record in ISAM parameter file</td>
</tr>
<tr>
<td>17112</td>
<td>00112</td>
<td>FSS ctree error</td>
<td>CT - LKISAM(ENABLE) found pending locks</td>
</tr>
<tr>
<td>17113</td>
<td>00113</td>
<td>FSS ctree error</td>
<td>CT - No memory for user lock table</td>
</tr>
<tr>
<td>17114</td>
<td>00114</td>
<td>FSS ctree error</td>
<td>CT - 1st byte of data record equals delete flag or bad variable length record mark</td>
</tr>
<tr>
<td>17115</td>
<td>00115</td>
<td>FSS ctree error</td>
<td>CT - Key segments do not match key length</td>
</tr>
<tr>
<td>17116</td>
<td>00116</td>
<td>FSS ctree error</td>
<td>CT - Bad mode parameter</td>
</tr>
<tr>
<td>17117</td>
<td>00117</td>
<td>FSS ctree error</td>
<td>CT - Could not read index member record</td>
</tr>
<tr>
<td>17118</td>
<td>00118</td>
<td>FSS ctree error</td>
<td>CT - NXTSET called before FRSSET for keyno</td>
</tr>
<tr>
<td>17119</td>
<td>00119</td>
<td>FSS ctree error</td>
<td>CT - FRSSET called for index with wrong keytyp</td>
</tr>
<tr>
<td>17120</td>
<td>00120</td>
<td>FSS ctree error</td>
<td>CT - Data record length exceeds rebuild max</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17121</td>
<td>00121</td>
<td>FSS ctree error</td>
<td>CT - Tried to update data with ctSAMKBUFhdr on</td>
</tr>
<tr>
<td>17122</td>
<td>00122</td>
<td>FSS ctree error</td>
<td>CT - Attempt to change fixed vs variable len</td>
</tr>
<tr>
<td>17123</td>
<td>00123</td>
<td>FSS ctree error</td>
<td>CT - Variable-length header has bad record mark</td>
</tr>
<tr>
<td>17124</td>
<td>00124</td>
<td>FSS ctree error</td>
<td>CT - # of indices does not match (OPNIFIL)</td>
</tr>
<tr>
<td>17125</td>
<td>00125</td>
<td>FSS ctree error</td>
<td>CT - c-tree already initialized</td>
</tr>
<tr>
<td>17126</td>
<td>00126</td>
<td>FSS ctree error</td>
<td>CT - Bad directory path get</td>
</tr>
<tr>
<td>17127</td>
<td>00127</td>
<td>FSS ctree error</td>
<td>CT - Could not send request</td>
</tr>
<tr>
<td>17128</td>
<td>00128</td>
<td>FSS ctree error</td>
<td>CT - Could not receive answer</td>
</tr>
<tr>
<td>17129</td>
<td>00129</td>
<td>FSS ctree error</td>
<td>CT - c-tree not initialized</td>
</tr>
<tr>
<td>17130</td>
<td>00130</td>
<td>FSS ctree error</td>
<td>CT - Null file name pointer in OPNIFIL</td>
</tr>
<tr>
<td>17131</td>
<td>00131</td>
<td>FSS ctree error</td>
<td>CT - File name length exceeds message size</td>
</tr>
<tr>
<td>17132</td>
<td>00132</td>
<td>FSS ctree error</td>
<td>CT - No room for application message buffer</td>
</tr>
<tr>
<td>17133</td>
<td>00133</td>
<td>FSS ctree error</td>
<td>CT - Server is not active</td>
</tr>
<tr>
<td>17134</td>
<td>00134</td>
<td>FSS ctree error</td>
<td>CT - Could not get servers message ID</td>
</tr>
<tr>
<td>17135</td>
<td>00135</td>
<td>FSS ctree error</td>
<td>CT - Could not allocate application ID</td>
</tr>
<tr>
<td>17136</td>
<td>00136</td>
<td>FSS ctree error</td>
<td>CT - Could not get application message status</td>
</tr>
<tr>
<td>17137</td>
<td>00137</td>
<td>FSS ctree error</td>
<td>CT - Could not set message appl message size</td>
</tr>
<tr>
<td>17138</td>
<td>00138</td>
<td>FSS ctree error</td>
<td>CT - Could not get rid of application message</td>
</tr>
<tr>
<td>17139</td>
<td>00139</td>
<td>FSS ctree error</td>
<td>CT - Badly formed file name</td>
</tr>
<tr>
<td>17140</td>
<td>00140</td>
<td>FSS ctree error</td>
<td>CT - Variable record length too long</td>
</tr>
<tr>
<td>17141</td>
<td>00141</td>
<td>FSS ctree error</td>
<td>CT - Required message size exceeds maximum</td>
</tr>
<tr>
<td>17142</td>
<td>00142</td>
<td>FSS ctree error</td>
<td>CT - Application MAXLEN &gt; server's MAXLEN</td>
</tr>
<tr>
<td>17143</td>
<td>00143</td>
<td>FSS ctree error</td>
<td>CT - Communications handler not installed</td>
</tr>
<tr>
<td>17144</td>
<td>00144</td>
<td>FSS ctree error</td>
<td>CT - Application could not id output queue</td>
</tr>
<tr>
<td>17145</td>
<td>00145</td>
<td>FSS ctree error</td>
<td>CT - Could not find COMM software</td>
</tr>
<tr>
<td>17146</td>
<td>00146</td>
<td>FSS ctree error</td>
<td>CT - Could not update free space info</td>
</tr>
<tr>
<td>17147</td>
<td>00147</td>
<td>FSS ctree error</td>
<td>CT - Space to be reused is not marked deleted</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>17148</td>
<td>00148</td>
<td>FSS ctree error</td>
<td>CT - WRTVREC cannot fit record at recbyt</td>
</tr>
<tr>
<td>17149</td>
<td>00149</td>
<td>FSS ctree error</td>
<td>CT - Varlen less than minimum in ADDVREC</td>
</tr>
<tr>
<td>17150</td>
<td>00150</td>
<td>FSS ctree error</td>
<td>CT - Server is shutting down</td>
</tr>
<tr>
<td>17151</td>
<td>00151</td>
<td>FSS ctree error</td>
<td>CT - Could not shut down. transactions pending</td>
</tr>
<tr>
<td>17152</td>
<td>00152</td>
<td>FSS ctree error</td>
<td>CT - Could not extend logfile</td>
</tr>
<tr>
<td>17153</td>
<td>00153</td>
<td>FSS ctree error</td>
<td>CT - Buffer too small</td>
</tr>
<tr>
<td>17154</td>
<td>00154</td>
<td>FSS ctree error</td>
<td>CT - Zero length record in REDVREC</td>
</tr>
<tr>
<td>17155</td>
<td>00155</td>
<td>FSS ctree error</td>
<td>CT - Native system failure</td>
</tr>
<tr>
<td>17156</td>
<td>00156</td>
<td>FSS ctree error</td>
<td>CT - timeout error</td>
</tr>
<tr>
<td>17158</td>
<td>00158</td>
<td>FSS ctree error</td>
<td>CT - REDVREC record not marked active</td>
</tr>
<tr>
<td>17159</td>
<td>00159</td>
<td>FSS ctree error</td>
<td>CT - Zero recbyt value</td>
</tr>
<tr>
<td>17160</td>
<td>00160</td>
<td>FSS ctree error</td>
<td>CT - Multi-user interference - index information updated by the time user got to actual data record</td>
</tr>
<tr>
<td>17161</td>
<td>00161</td>
<td>FSS ctree error</td>
<td>CT - User appears inactive</td>
</tr>
<tr>
<td>17162</td>
<td>00162</td>
<td>FSS ctree error</td>
<td>CT - Server has gone away</td>
</tr>
<tr>
<td>17163</td>
<td>00163</td>
<td>FSS ctree error</td>
<td>CT - No more room in server lock table</td>
</tr>
<tr>
<td>17164</td>
<td>00164</td>
<td>FSS ctree error</td>
<td>CT - File number out of range</td>
</tr>
<tr>
<td>17165</td>
<td>00165</td>
<td>FSS ctree error</td>
<td>CT - No file control block available</td>
</tr>
<tr>
<td>17166</td>
<td>00166</td>
<td>FSS ctree error</td>
<td>CT - No more ct file control blocks in server</td>
</tr>
<tr>
<td>17167</td>
<td>00167</td>
<td>FSS ctree error</td>
<td>CT - Could not read request</td>
</tr>
<tr>
<td>17168</td>
<td>00168</td>
<td>FSS ctree error</td>
<td>CT - Could not send answer</td>
</tr>
<tr>
<td>17169</td>
<td>00169</td>
<td>FSS ctree error</td>
<td>CT - Create file already opened (in recovery)</td>
</tr>
<tr>
<td>17170</td>
<td>00170</td>
<td>FSS ctree error</td>
<td>CT - Bad function number</td>
</tr>
<tr>
<td>17171</td>
<td>00171</td>
<td>FSS ctree error</td>
<td>CT - Application message size exceeds server size</td>
</tr>
<tr>
<td>17172</td>
<td>00172</td>
<td>FSS ctree error</td>
<td>CT - Could not allocate server message buffer</td>
</tr>
<tr>
<td>17173</td>
<td>00173</td>
<td>FSS ctree error</td>
<td>CT - Could not identify server</td>
</tr>
<tr>
<td>17174</td>
<td>00174</td>
<td>FSS ctree error</td>
<td>CT - Could not get server message ID</td>
</tr>
<tr>
<td>17175</td>
<td>00175</td>
<td>FSS ctree error</td>
<td>CT - Server could not allocate user message area</td>
</tr>
<tr>
<td>17176</td>
<td>00176</td>
<td>FSS ctree error</td>
<td>CT - Could not get server msg status</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17177</td>
<td>00177</td>
<td>FSS ctree error</td>
<td>CT - Could not set message server message size</td>
</tr>
<tr>
<td>17178</td>
<td>00178</td>
<td>FSS ctree error</td>
<td>CT - Unexpected file# assigned to [si] in rcv</td>
</tr>
<tr>
<td>17179</td>
<td>00179</td>
<td>FSS ctree error</td>
<td>CT - Server is at full user capacity</td>
</tr>
<tr>
<td>17180</td>
<td>00180</td>
<td>FSS ctree error</td>
<td>CT - Could not read symbolic key name</td>
</tr>
<tr>
<td>17181</td>
<td>00181</td>
<td>FSS ctree error</td>
<td>CT - Could not get mem for key symb name</td>
</tr>
<tr>
<td>17182</td>
<td>00182</td>
<td>FSS ctree error</td>
<td>CT - No room for sort key. increase MAXFIL</td>
</tr>
<tr>
<td>17183</td>
<td>00183</td>
<td>FSS ctree error</td>
<td>CT - Could not read file field number values</td>
</tr>
<tr>
<td>17184</td>
<td>00184</td>
<td>FSS ctree error</td>
<td>CT - Attempt to reallocate set space</td>
</tr>
<tr>
<td>17185</td>
<td>00185</td>
<td>FSS ctree error</td>
<td>CT - Not enough memory for additional sets-batches</td>
</tr>
<tr>
<td>17186</td>
<td>00186</td>
<td>FSS ctree error</td>
<td>CT - Set number out of range</td>
</tr>
<tr>
<td>17187</td>
<td>00187</td>
<td>FSS ctree error</td>
<td>CT - Null buffer in rtread.c</td>
</tr>
<tr>
<td>17188</td>
<td>00188</td>
<td>FSS ctree error</td>
<td>CT - Null target buffer in rtread.c</td>
</tr>
<tr>
<td>17189</td>
<td>00189</td>
<td>FSS ctree error</td>
<td>CT - Join_to skip</td>
</tr>
<tr>
<td>17190</td>
<td>00190</td>
<td>FSS ctree error</td>
<td>CT - Join_to error</td>
</tr>
<tr>
<td>17191</td>
<td>00191</td>
<td>FSS ctree error</td>
<td>CT - Join_to null fill</td>
</tr>
<tr>
<td>17192</td>
<td>00192</td>
<td>FSS ctree error</td>
<td>CT - Detail_for skip</td>
</tr>
<tr>
<td>17193</td>
<td>00193</td>
<td>FSS ctree error</td>
<td>CT - Detail_for error</td>
</tr>
<tr>
<td>17194</td>
<td>00194</td>
<td>FSS ctree error</td>
<td>CT - Detail_for null fill</td>
</tr>
<tr>
<td>17195</td>
<td>00195</td>
<td>FSS ctree error</td>
<td>CT - Could not get mem for dat symb name</td>
</tr>
<tr>
<td>17196</td>
<td>00196</td>
<td>FSS ctree error</td>
<td>CT - Exceeded RETRY_LIMIT in RTREAD.C</td>
</tr>
<tr>
<td>17197</td>
<td>00197</td>
<td>FSS ctree error</td>
<td>CT - Could not get memory for IFIL block</td>
</tr>
<tr>
<td>17198</td>
<td>00198</td>
<td>FSS ctree error</td>
<td>CT - Improper IFIL block</td>
</tr>
<tr>
<td>17199</td>
<td>00199</td>
<td>FSS ctree error</td>
<td>CT - Schema not defined for data file</td>
</tr>
</tbody>
</table>
# Error Messages

## 173xx

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>17370</td>
<td>00370</td>
<td>FSS ctree error</td>
<td>CT - Sort base - errors + 101 thru 126 see CTSORT.C or CTERRC.H for error listing</td>
</tr>
</tbody>
</table>

## 174xx

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>17400</td>
<td>00400</td>
<td>FSS ctree error</td>
<td>CT - Resource already enabled</td>
</tr>
<tr>
<td>17401</td>
<td>00401</td>
<td>FSS ctree error</td>
<td>CT - Resources not enabled</td>
</tr>
<tr>
<td>17402</td>
<td>00402</td>
<td>FSS ctree error</td>
<td>CT - File must be exclusive to enable res</td>
</tr>
<tr>
<td>17403</td>
<td>00403</td>
<td>FSS ctree error</td>
<td>CT - Empty resource id</td>
</tr>
<tr>
<td>17404</td>
<td>00404</td>
<td>FSS ctree error</td>
<td>CT - Output buffer to small</td>
</tr>
<tr>
<td>17405</td>
<td>00405</td>
<td>FSS ctree error</td>
<td>CT - Resource id already added</td>
</tr>
<tr>
<td>17406</td>
<td>00406</td>
<td>FSS ctree error</td>
<td>CT - Bad resource search mode</td>
</tr>
<tr>
<td>17407</td>
<td>00407</td>
<td>FSS ctree error</td>
<td>CT - Attempt to get non-resource info</td>
</tr>
<tr>
<td>17408</td>
<td>00408</td>
<td>FSS ctree error</td>
<td>CT - Resource not found</td>
</tr>
<tr>
<td>17409</td>
<td>00409</td>
<td>FSS ctree error</td>
<td>CT - Not in use</td>
</tr>
<tr>
<td>17410</td>
<td>00410</td>
<td>FSS ctree error</td>
<td>CT - User not active</td>
</tr>
<tr>
<td>17411</td>
<td>00411</td>
<td>FSS ctree error</td>
<td>CT - Not a superfile</td>
</tr>
<tr>
<td>17412</td>
<td>00412</td>
<td>FSS ctree error</td>
<td>CT - WRL to WXL commit promote pending(CIL)</td>
</tr>
<tr>
<td>17413</td>
<td>00413</td>
<td>FSS ctree error</td>
<td>CT - Superfile host not opened</td>
</tr>
<tr>
<td>17414</td>
<td>00414</td>
<td>FSS ctree error</td>
<td>CT - Cannot nest superfiles</td>
</tr>
<tr>
<td>17415</td>
<td>00415</td>
<td>FSS ctree error</td>
<td>CT - Illegal ADDKEY to superfile</td>
</tr>
<tr>
<td>17416</td>
<td>00416</td>
<td>FSS ctree error</td>
<td>CT - Illegal DELBLD to superfile</td>
</tr>
<tr>
<td>17417</td>
<td>00417</td>
<td>FSS ctree error</td>
<td>CT - Cache page size error</td>
</tr>
<tr>
<td>17418</td>
<td>00418</td>
<td>FSS ctree error</td>
<td>CT - Max name inconsistency</td>
</tr>
<tr>
<td>17419</td>
<td>00419</td>
<td>FSS ctree error</td>
<td>CT - Host superfile does not support recovery</td>
</tr>
<tr>
<td>17420</td>
<td>00420</td>
<td>FSS ctree error</td>
<td>CT - Key update with pending transaction on index %s</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17421</td>
<td>00421</td>
<td>FSS ctree error</td>
<td>CT - Filter not supported yet</td>
</tr>
<tr>
<td>17422</td>
<td>00422</td>
<td>FSS ctree error</td>
<td>CT - Other functions not sup</td>
</tr>
<tr>
<td>17423</td>
<td>00423</td>
<td>FSS ctree error</td>
<td>CT - Incomplete</td>
</tr>
<tr>
<td>17424</td>
<td>00424</td>
<td>FSS ctree error</td>
<td>CT - Add list err</td>
</tr>
<tr>
<td>17425</td>
<td>00425</td>
<td>FSS ctree error</td>
<td>CT - Batch in progress</td>
</tr>
<tr>
<td>17426</td>
<td>00426</td>
<td>FSS ctree error</td>
<td>CT - No batch active</td>
</tr>
<tr>
<td>17427</td>
<td>00427</td>
<td>FSS ctree error</td>
<td>CT - Status info already returned</td>
</tr>
<tr>
<td>17428</td>
<td>00428</td>
<td>FSS ctree error</td>
<td>CT - No more info, batch cancelled</td>
</tr>
<tr>
<td>17429</td>
<td>00429</td>
<td>FSS ctree error</td>
<td>CT - Bufsz too small for record</td>
</tr>
<tr>
<td>17430</td>
<td>00430</td>
<td>FSS ctree error</td>
<td>CT - Request is empty or inconsistent</td>
</tr>
<tr>
<td>17431</td>
<td>00431</td>
<td>FSS ctree error</td>
<td>CT - Aggregate/serialization lock denied</td>
</tr>
<tr>
<td>17432</td>
<td>00432</td>
<td>FSS ctree error</td>
<td>CT - Fixed-length string requires len in DODA</td>
</tr>
<tr>
<td>17433</td>
<td>00433</td>
<td>FSS ctree error</td>
<td>CT - Segment def inconsistent with schema</td>
</tr>
<tr>
<td>17434</td>
<td>00434</td>
<td>FSS ctree error</td>
<td>CT - Very long def block not supported</td>
</tr>
<tr>
<td>17435</td>
<td>00435</td>
<td>FSS ctree error</td>
<td>CT - File def memory error</td>
</tr>
<tr>
<td>17436</td>
<td>00436</td>
<td>FSS ctree error</td>
<td>CT - Bad def number</td>
</tr>
<tr>
<td>17437</td>
<td>00437</td>
<td>FSS ctree error</td>
<td>CT - defptr NULL during GETDEFBLK</td>
</tr>
<tr>
<td>17438</td>
<td>00438</td>
<td>FSS ctree error</td>
<td>CT - Requested def block is empty</td>
</tr>
<tr>
<td>17439</td>
<td>00439</td>
<td>FSS ctree error</td>
<td>CT - No conversion routine for Definition Block</td>
</tr>
<tr>
<td>17440</td>
<td>00440</td>
<td>FSS ctree error</td>
<td>CT - Dynamic dump already in progress</td>
</tr>
<tr>
<td>17441</td>
<td>00441</td>
<td>FSS ctree error</td>
<td>CT - No memory for dynamic dump file buffer</td>
</tr>
<tr>
<td>17442</td>
<td>00442</td>
<td>FSS ctree error</td>
<td>CT - One or more files not available for dump</td>
</tr>
<tr>
<td>17443</td>
<td>00443</td>
<td>FSS ctree error</td>
<td>CT - File length discrepancy</td>
</tr>
<tr>
<td>17444</td>
<td>00444</td>
<td>FSS ctree error</td>
<td>CT - Could not create file during dump rcv</td>
</tr>
<tr>
<td>17445</td>
<td>00445</td>
<td>FSS ctree error</td>
<td>CT - Not enough data to assemble key value</td>
</tr>
<tr>
<td>17446</td>
<td>00446</td>
<td>FSS ctree error</td>
<td>CT - Bad mode - parameter out of range</td>
</tr>
<tr>
<td>17447</td>
<td>00447</td>
<td>FSS ctree error</td>
<td>CT - Only the file's owner can perform op</td>
</tr>
<tr>
<td>17448</td>
<td>00448</td>
<td>FSS ctree error</td>
<td>CT - Permission to set file definition denied</td>
</tr>
<tr>
<td>17449</td>
<td>00449</td>
<td>FSS ctree error</td>
<td>CT - ADMIN has opened file. Cannot delete file</td>
</tr>
<tr>
<td>17450</td>
<td>00450</td>
<td>FSS ctree error</td>
<td>CT - Invalid user id</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17451</td>
<td>00451</td>
<td>FSS ctree error</td>
<td>CT - Invalid password</td>
</tr>
<tr>
<td>17452</td>
<td>00452</td>
<td>FSS ctree error</td>
<td>CT - Server could not process user/acct info</td>
</tr>
<tr>
<td>17453</td>
<td>00453</td>
<td>FSS ctree error</td>
<td>CT - No such server</td>
</tr>
<tr>
<td>17454</td>
<td>00454</td>
<td>FSS ctree error</td>
<td>CT - Service not supported</td>
</tr>
<tr>
<td>17455</td>
<td>00455</td>
<td>FSS ctree error</td>
<td>CT - User does not belong to group</td>
</tr>
<tr>
<td>17456</td>
<td>00456</td>
<td>FSS ctree error</td>
<td>CT - Group access denied</td>
</tr>
<tr>
<td>17457</td>
<td>00457</td>
<td>FSS ctree error</td>
<td>CT - File password invalid</td>
</tr>
<tr>
<td>17458</td>
<td>00458</td>
<td>FSS ctree error</td>
<td>CT - Write permission not granted</td>
</tr>
<tr>
<td>17459</td>
<td>00459</td>
<td>FSS ctree error</td>
<td>CT - File delete permission denied</td>
</tr>
<tr>
<td>17460</td>
<td>00460</td>
<td>FSS ctree error</td>
<td>CT - Resource not enabled</td>
</tr>
<tr>
<td>17461</td>
<td>00461</td>
<td>FSS ctree error</td>
<td>CT - Bad permission flag</td>
</tr>
<tr>
<td>17462</td>
<td>00462</td>
<td>FSS ctree error</td>
<td>CT - No directory found in superfile</td>
</tr>
<tr>
<td>17463</td>
<td>00463</td>
<td>FSS ctree error</td>
<td>CT - File id uniqueness error</td>
</tr>
<tr>
<td>17464</td>
<td>00464</td>
<td>FSS ctree error</td>
<td>CT - ISAM level logon not performed</td>
</tr>
<tr>
<td>17465</td>
<td>00465</td>
<td>FSS ctree error</td>
<td>CT - Incremental Index - dnumidx &lt; 1</td>
</tr>
<tr>
<td>17466</td>
<td>00466</td>
<td>FSS ctree error</td>
<td>CT - Incremental Index - dfilno not a ISAM file</td>
</tr>
<tr>
<td>17467</td>
<td>00467</td>
<td>FSS ctree error</td>
<td>CT - Incremental Index - aidxnam NULL for 1st</td>
</tr>
<tr>
<td>17468</td>
<td>00468</td>
<td>FSS ctree error</td>
<td>CT - Incremental Index - active tran not allowed</td>
</tr>
<tr>
<td>17469</td>
<td>00469</td>
<td>FSS ctree error</td>
<td>CT - Negative I/O request</td>
</tr>
<tr>
<td>17470</td>
<td>00470</td>
<td>FSS ctree error</td>
<td>CT - Guest logons disabled</td>
</tr>
<tr>
<td>17471</td>
<td>00471</td>
<td>FSS ctree error</td>
<td>CT - error deleting sortwork file</td>
</tr>
<tr>
<td>17472</td>
<td>00472</td>
<td>FSS ctree error</td>
<td>CT - error creating unique name</td>
</tr>
<tr>
<td>17473</td>
<td>00473</td>
<td>FSS ctree error</td>
<td>CT - error opening first dummy file</td>
</tr>
<tr>
<td>17474</td>
<td>00474</td>
<td>FSS ctree error</td>
<td>CT - too few handles available min 3</td>
</tr>
<tr>
<td>17475</td>
<td>00475</td>
<td>FSS ctree error</td>
<td>CT - error closing dummy file</td>
</tr>
<tr>
<td>17476</td>
<td>00476</td>
<td>FSS ctree error</td>
<td>CT - error unlinking dummy file</td>
</tr>
<tr>
<td>17477</td>
<td>00477</td>
<td>FSS ctree error</td>
<td>CT - error getting first data area</td>
</tr>
<tr>
<td>17478</td>
<td>00478</td>
<td>FSS ctree error</td>
<td>CT - sinit phase not previously performed-srelease</td>
</tr>
<tr>
<td>17479</td>
<td>00479</td>
<td>FSS ctree error</td>
<td>CT - sreturn phase already started</td>
</tr>
<tr>
<td>17480</td>
<td>00480</td>
<td>FSS ctree error</td>
<td>CT - no records in data buffers</td>
</tr>
<tr>
<td>17481</td>
<td>00481</td>
<td>FSS ctree error</td>
<td>CT - sint phase not previously performed-sreturn</td>
</tr>
<tr>
<td>17482</td>
<td>00482</td>
<td>FSS ctree error</td>
<td>CT - not enough memory</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>17483</td>
<td>00483</td>
<td>FSS ctree error</td>
<td>CT - no valid record pointers in merge buffers</td>
</tr>
<tr>
<td>17484</td>
<td>00484</td>
<td>FSS ctree error</td>
<td>CT - error opening sortwork file</td>
</tr>
<tr>
<td>17485</td>
<td>00485</td>
<td>FSS ctree error</td>
<td>CT - error creating sortwork.00x file</td>
</tr>
<tr>
<td>17486</td>
<td>00486</td>
<td>FSS ctree error</td>
<td>CT - no records fit in output buffer</td>
</tr>
<tr>
<td>17487</td>
<td>00487</td>
<td>FSS ctree error</td>
<td>CT - error reading sortwork file</td>
</tr>
<tr>
<td>17488</td>
<td>00488</td>
<td>FSS ctree error</td>
<td>CT - bytes in buf &lt;&gt; merge buf size</td>
</tr>
<tr>
<td>17489</td>
<td>00489</td>
<td>FSS ctree error</td>
<td>CT - error adjusting file pointer</td>
</tr>
<tr>
<td>17490</td>
<td>00490</td>
<td>FSS ctree error</td>
<td>CT - error closing sortwork.00x</td>
</tr>
<tr>
<td>17491</td>
<td>00491</td>
<td>FSS ctree error</td>
<td>CT - error closing sortwork file</td>
</tr>
<tr>
<td>17492</td>
<td>00492</td>
<td>FSS ctree error</td>
<td>CT - error deleting sortwork file</td>
</tr>
<tr>
<td>17493</td>
<td>00493</td>
<td>FSS ctree error</td>
<td>CT - error renaming sortwork.00x</td>
</tr>
<tr>
<td>17494</td>
<td>00494</td>
<td>FSS ctree error</td>
<td>CT - error closing output file</td>
</tr>
<tr>
<td>17495</td>
<td>00495</td>
<td>FSS ctree error</td>
<td>CT - error creating output file</td>
</tr>
<tr>
<td>17496</td>
<td>00496</td>
<td>FSS ctree error</td>
<td>CT - insufficient disk space</td>
</tr>
<tr>
<td>17498</td>
<td>00498</td>
<td>FSS ctree error</td>
<td>CT - Old log file found during log create</td>
</tr>
<tr>
<td>17499</td>
<td>00499</td>
<td>FSS ctree error</td>
<td>CT - Mismatch between receive log &amp; file ID</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17500</td>
<td>00500</td>
<td>FSS ctree error</td>
<td>CT - Server could not init SQL engine</td>
</tr>
<tr>
<td>17501</td>
<td>00501</td>
<td>FSS ctree error</td>
<td>CT - Could not init SQL for a user</td>
</tr>
<tr>
<td>17502</td>
<td>00502</td>
<td>FSS ctree error</td>
<td>CT - Could not access SQL master info</td>
</tr>
<tr>
<td>17503</td>
<td>00503</td>
<td>FSS ctree error</td>
<td>CT - Could not continue SQL request</td>
</tr>
<tr>
<td>17504</td>
<td>00504</td>
<td>FSS ctree error</td>
<td>CT - Server does not support SQL</td>
</tr>
<tr>
<td>17505</td>
<td>00505</td>
<td>FSS ctree error</td>
<td>CT - User profile does not enable SQL</td>
</tr>
<tr>
<td>17506</td>
<td>00506</td>
<td>FSS ctree error</td>
<td>CT - Could open save-restore file</td>
</tr>
<tr>
<td>17507</td>
<td>00507</td>
<td>FSS ctree error</td>
<td>CT - Could not process save-restore file</td>
</tr>
<tr>
<td>17508</td>
<td>00508</td>
<td>FSS ctree error</td>
<td>CT - Save restore inconsistency</td>
</tr>
<tr>
<td>17509</td>
<td>00509</td>
<td>FSS ctree error</td>
<td>CT - Duplicate server</td>
</tr>
<tr>
<td>17510</td>
<td>00510</td>
<td>FSS ctree error</td>
<td>CT - Active chkpnt at start of roll-forward</td>
</tr>
<tr>
<td>17511</td>
<td>00511</td>
<td>FSS ctree error</td>
<td>CT - Index nodes form illegal loop - rebuild</td>
</tr>
<tr>
<td>17512</td>
<td>00512</td>
<td>FSS ctree error</td>
<td>CT - Data file loop detected</td>
</tr>
<tr>
<td>17513</td>
<td>00513</td>
<td>FSS ctree error</td>
<td>CT - FPUTFGET does not support CTSBLDX ()</td>
</tr>
<tr>
<td>17514</td>
<td>00514</td>
<td>FSS ctree error</td>
<td>CT - Queue has been closed</td>
</tr>
<tr>
<td>17515</td>
<td>00515</td>
<td>FSS ctree error</td>
<td>CT - Cannot convert old IFIL structure</td>
</tr>
<tr>
<td>17516</td>
<td>00516</td>
<td>FSS ctree error</td>
<td>CT - ctNOGLOBALs not allocated</td>
</tr>
<tr>
<td>17517</td>
<td>00517</td>
<td>FSS ctree error</td>
<td>CT - 'regid' is not registered</td>
</tr>
<tr>
<td>17518</td>
<td>00518</td>
<td>FSS ctree error</td>
<td>CT - 'regid' is already registered</td>
</tr>
<tr>
<td>17519</td>
<td>00519</td>
<td>FSS ctree error</td>
<td>CT - Index logical EOF error</td>
</tr>
<tr>
<td>17520</td>
<td>00520</td>
<td>FSS ctree error</td>
<td>CT - Attempt to update index with inconsistent tran#</td>
</tr>
<tr>
<td>17521</td>
<td>00521</td>
<td>FSS ctree error</td>
<td>CT - Could not allocate memory for the streettalk login message buffer</td>
</tr>
<tr>
<td>17522</td>
<td>00522</td>
<td>FSS ctree error</td>
<td>CT - UserID in INTISAM does not match current login ID</td>
</tr>
<tr>
<td>17527</td>
<td>00527</td>
<td>FSS ctree error</td>
<td>CT - Index must be rebuilt - see CTSTATUS.FCS</td>
</tr>
<tr>
<td>17528</td>
<td>00528</td>
<td>FSS ctree error</td>
<td>CT - Key segment length error</td>
</tr>
<tr>
<td>17529</td>
<td>00529</td>
<td>FSS ctree error</td>
<td>CT - System checkpoints terminated</td>
</tr>
<tr>
<td>17530</td>
<td>00530</td>
<td>FSS ctree error</td>
<td>CT - Client does not match server</td>
</tr>
<tr>
<td>17531</td>
<td>00531</td>
<td>FSS ctree error</td>
<td>CT - Index reorg entry error</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>17532</td>
<td>00532</td>
<td>FSS ctree error</td>
<td>CT - TRANSAV called with AUTOSAVE on</td>
</tr>
<tr>
<td>17533</td>
<td>00533</td>
<td>FSS ctree error</td>
<td>CT - File header high-water-mark overflow</td>
</tr>
<tr>
<td>17534</td>
<td>00534</td>
<td>FSS ctree error</td>
<td>CT - Transaction # overflow</td>
</tr>
<tr>
<td>17535</td>
<td>00535</td>
<td>FSS ctree error</td>
<td>CT - ctree not registered. Call REGCTREE</td>
</tr>
<tr>
<td>17536</td>
<td>00536</td>
<td>FSS ctree error</td>
<td>CT - Only automatic REGCTREEs allowed</td>
</tr>
<tr>
<td>17538</td>
<td>00538</td>
<td>FSS ctree error</td>
<td>CT - Client-side bad function array type</td>
</tr>
<tr>
<td>17539</td>
<td>00539</td>
<td>FSS ctree error</td>
<td>CT - sysiocod when file does not appear to contain any valid information</td>
</tr>
<tr>
<td>17540</td>
<td>00540</td>
<td>FSS ctree error</td>
<td>CT - Null parameter</td>
</tr>
<tr>
<td>17541</td>
<td>00541</td>
<td>FSS ctree error</td>
<td>CT - Transaction log cannot be written</td>
</tr>
<tr>
<td>17542</td>
<td>00542</td>
<td>FSS ctree error</td>
<td>CT - Could not create mirror file</td>
</tr>
<tr>
<td>17543</td>
<td>00543</td>
<td>FSS ctree error</td>
<td>CT - Could not open mirror file</td>
</tr>
<tr>
<td>17544</td>
<td>00544</td>
<td>FSS ctree error</td>
<td>CT - Could not close mirror file</td>
</tr>
<tr>
<td>17545</td>
<td>00545</td>
<td>FSS ctree error</td>
<td>CT - Could not delete mirror file</td>
</tr>
<tr>
<td>17546</td>
<td>00546</td>
<td>FSS ctree error</td>
<td>CT - Could not write to mirror file</td>
</tr>
<tr>
<td>17547</td>
<td>00547</td>
<td>FSS ctree error</td>
<td>CT - Could not save mirror file</td>
</tr>
<tr>
<td>17548</td>
<td>00548</td>
<td>FSS ctree error</td>
<td>CT - Could not read from mirror</td>
</tr>
<tr>
<td>17549</td>
<td>00549</td>
<td>FSS ctree error</td>
<td>CT - Mismatch between mirror headers</td>
</tr>
<tr>
<td>17550</td>
<td>00550</td>
<td>FSS ctree error</td>
<td>CT - Attempt to open primary w/o mirror</td>
</tr>
<tr>
<td>17551</td>
<td>00551</td>
<td>FSS ctree error</td>
<td>CT - File already opened without mirror</td>
</tr>
<tr>
<td>17552</td>
<td>00552</td>
<td>FSS ctree error</td>
<td>CT - Segmented file cannot be mirrored</td>
</tr>
<tr>
<td>17555</td>
<td>00555</td>
<td>FSS ctree error</td>
<td>CT - Could not read primary, switching</td>
</tr>
<tr>
<td>17556</td>
<td>00556</td>
<td>FSS ctree error</td>
<td>CT - Could not write primary, switching</td>
</tr>
<tr>
<td>17557</td>
<td>00557</td>
<td>FSS ctree error</td>
<td>CT - Could not write mirror, suspend mir</td>
</tr>
<tr>
<td>17558</td>
<td>00558</td>
<td>FSS ctree error</td>
<td>CT - Could not save primary, switching</td>
</tr>
<tr>
<td>17559</td>
<td>00559</td>
<td>FSS ctree error</td>
<td>CT - Could not save mirror, suspend mir</td>
</tr>
<tr>
<td>17560</td>
<td>00560</td>
<td>FSS ctree error</td>
<td>CT - Only one of each monitor at a time</td>
</tr>
<tr>
<td>17561</td>
<td>00561</td>
<td>FSS ctree error</td>
<td>CT - SYSMON - dynamic dump begins</td>
</tr>
<tr>
<td>17562</td>
<td>00562</td>
<td>FSS ctree error</td>
<td>CT - SYSMON - dynamic dump ends</td>
</tr>
<tr>
<td>17563</td>
<td>00563</td>
<td>FSS ctree error</td>
<td>CT - SYSMON - dynamic dump ends (errors)</td>
</tr>
<tr>
<td>17570</td>
<td>00570</td>
<td>FSS ctree error</td>
<td>CT - Incomplete compression</td>
</tr>
<tr>
<td>17571</td>
<td>00571</td>
<td>FSS ctree error</td>
<td>CT - Index rebuild required</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17572</td>
<td>00572</td>
<td>FSS ctree error</td>
<td>CT - Incomplete compression &amp; index rebuild required</td>
</tr>
<tr>
<td>17573</td>
<td>00573</td>
<td>FSS ctree error</td>
<td>CT - Primary/mirror out-of-sync. Copy good file over bad</td>
</tr>
<tr>
<td>17574</td>
<td>00574</td>
<td>FSS ctree error</td>
<td>CT - Incomplete compression &amp; primary/mirror out-of-sync</td>
</tr>
<tr>
<td>17575</td>
<td>00575</td>
<td>FSS ctree error</td>
<td>CT - Incomplete compression required &amp; primary/mirror out-of-sync</td>
</tr>
<tr>
<td>17576</td>
<td>00576</td>
<td>FSS ctree error</td>
<td>CT - Incomplete compression &amp; index rebuild required &amp; primary/mirror out-of-sync</td>
</tr>
<tr>
<td>17577</td>
<td>00577</td>
<td>FSS ctree error</td>
<td>CT - Update refused during protected dmp</td>
</tr>
<tr>
<td>17579</td>
<td>00579</td>
<td>FSS ctree error</td>
<td>CT - Logon interval error</td>
</tr>
<tr>
<td>17580</td>
<td>00580</td>
<td>FSS ctree error</td>
<td>CT - Could not perform file security op</td>
</tr>
<tr>
<td>17581</td>
<td>00581</td>
<td>FSS ctree error</td>
<td>CT - Group does not exist</td>
</tr>
<tr>
<td>17582</td>
<td>00582</td>
<td>FSS ctree error</td>
<td>CT - Key length too small</td>
</tr>
<tr>
<td>17583</td>
<td>00583</td>
<td>FSS ctree error</td>
<td>CT - File extension exceeds diskfull limit</td>
</tr>
<tr>
<td>17584</td>
<td>00584</td>
<td>FSS ctree error</td>
<td>CT - Exceeded failed logon limit</td>
</tr>
<tr>
<td>17585</td>
<td>00585</td>
<td>FSS ctree error</td>
<td>CT - Logon date exception</td>
</tr>
<tr>
<td>17586</td>
<td>00586</td>
<td>FSS ctree error</td>
<td>CT - Diff file name, matching file ID</td>
</tr>
<tr>
<td>17587</td>
<td>00587</td>
<td>FSS ctree error</td>
<td>CT - Close/delete deferred - pending tran</td>
</tr>
<tr>
<td>17588</td>
<td>00588</td>
<td>FSS ctree error</td>
<td>CT - Attempt to close or delete file with pending (active) tran</td>
</tr>
<tr>
<td>17589</td>
<td>00589</td>
<td>FSS ctree error</td>
<td>CT - Member of ADMIN group required</td>
</tr>
<tr>
<td>17590</td>
<td>00590</td>
<td>FSS ctree error</td>
<td>CT - Could not find ISAM context ID</td>
</tr>
<tr>
<td>17591</td>
<td>00591</td>
<td>FSS ctree error</td>
<td>CT - Old context ID. Call CLSICON()</td>
</tr>
<tr>
<td>17592</td>
<td>00592</td>
<td>FSS ctree error</td>
<td>CT - Context ID exists</td>
</tr>
<tr>
<td>17593</td>
<td>00593</td>
<td>FSS ctree error</td>
<td>CT - Non-ADMIN user blocked from log on</td>
</tr>
<tr>
<td>17595</td>
<td>00595</td>
<td>FSS ctree error</td>
<td>CT - varlen too small in PUTCRES</td>
</tr>
<tr>
<td>17596</td>
<td>00596</td>
<td>FSS ctree error</td>
<td>CT - Missing information</td>
</tr>
<tr>
<td>17597</td>
<td>00597</td>
<td>FSS ctree error</td>
<td>CT - Could not initialize expression</td>
</tr>
<tr>
<td>17598</td>
<td>00598</td>
<td>FSS ctree error</td>
<td>CT - Could not evaluate conditional expression</td>
</tr>
<tr>
<td>17599</td>
<td>00599</td>
<td>FSS ctree error</td>
<td>CT - Dynamic dump extent error</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17700</td>
<td>00700</td>
<td>FSS ctree error</td>
<td>CT - Could not open key segment definition</td>
</tr>
<tr>
<td>17701</td>
<td>00701</td>
<td>FSS ctree error</td>
<td>CT - Could not process comp options</td>
</tr>
<tr>
<td>17702</td>
<td>00702</td>
<td>FSS ctree error</td>
<td>CT - Could not process comp attributes</td>
</tr>
<tr>
<td>17703</td>
<td>00703</td>
<td>FSS ctree error</td>
<td>CT - Invalid key segment handle</td>
</tr>
<tr>
<td>17704</td>
<td>00704</td>
<td>FSS ctree error</td>
<td>CT - Invalid source type</td>
</tr>
<tr>
<td>17705</td>
<td>00705</td>
<td>FSS ctree error</td>
<td>CT - Segment definition already exists</td>
</tr>
<tr>
<td>17706</td>
<td>00706</td>
<td>FSS ctree error</td>
<td>CT - No segment data produced</td>
</tr>
<tr>
<td>17707</td>
<td>00707</td>
<td>FSS ctree error</td>
<td>CT - No segment definition</td>
</tr>
<tr>
<td>17708</td>
<td>00708</td>
<td>FSS ctree error</td>
<td>CT - Multibyte names not supported</td>
</tr>
<tr>
<td>17709</td>
<td>00709</td>
<td>FSS ctree error</td>
<td>CT - Badly formed multibyte name</td>
</tr>
<tr>
<td>17710</td>
<td>00710</td>
<td>FSS ctree error</td>
<td>CT - Multibyte variant not supported</td>
</tr>
<tr>
<td>17711</td>
<td>00711</td>
<td>FSS ctree error</td>
<td>CT - Cannot call UPDCIDX while DROPIDX pending</td>
</tr>
<tr>
<td>17712</td>
<td>00712</td>
<td>FSS ctree error</td>
<td>CT - Partition number out of range - low</td>
</tr>
<tr>
<td>17713</td>
<td>00713</td>
<td>FSS ctree error</td>
<td>CT - File is not partition host</td>
</tr>
<tr>
<td>17714</td>
<td>00714</td>
<td>FSS ctree error</td>
<td>CT - File is not partition member</td>
</tr>
<tr>
<td>17715</td>
<td>00715</td>
<td>FSS ctree error</td>
<td>CT - Raw partition does not exist</td>
</tr>
<tr>
<td>17716</td>
<td>00716</td>
<td>FSS ctree error</td>
<td>CT - Bad value for partition key</td>
</tr>
<tr>
<td>17717</td>
<td>00717</td>
<td>FSS ctree error</td>
<td>CT - could not overload partition #</td>
</tr>
<tr>
<td>17718</td>
<td>00718</td>
<td>FSS ctree error</td>
<td>CT - Partition member in use</td>
</tr>
<tr>
<td>17719</td>
<td>00719</td>
<td>FSS ctree error</td>
<td>CT - Partition member pending purge</td>
</tr>
<tr>
<td>17720</td>
<td>00720</td>
<td>FSS ctree error</td>
<td>CT - Partition member purged</td>
</tr>
<tr>
<td>17721</td>
<td>00721</td>
<td>FSS ctree error</td>
<td>CT - Partition member archived</td>
</tr>
<tr>
<td>17722</td>
<td>00722</td>
<td>FSS ctree error</td>
<td>CT - Bad partition host list</td>
</tr>
<tr>
<td>17723</td>
<td>00723</td>
<td>FSS ctree error</td>
<td>CT - Must retry operation</td>
</tr>
<tr>
<td>17724</td>
<td>00724</td>
<td>FSS ctree error</td>
<td>CT - Bad current ISAM position for host</td>
</tr>
<tr>
<td>17725</td>
<td>00725</td>
<td>FSS ctree error</td>
<td>CT - PARTRES version error</td>
</tr>
<tr>
<td>17726</td>
<td>00726</td>
<td>FSS ctree error</td>
<td>CT - Duplicate error caused by purged part</td>
</tr>
<tr>
<td>17727</td>
<td>00727</td>
<td>FSS ctree error</td>
<td>CT - Bad partition file mode settings</td>
</tr>
<tr>
<td>17728</td>
<td>00728</td>
<td>FSS ctree error</td>
<td>CT - Partition support not enabled</td>
</tr>
<tr>
<td>17729</td>
<td>00729</td>
<td>FSS ctree error</td>
<td>CT - Purged unique global keys encountered</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>17730</td>
<td>00730</td>
<td>FSS ctree error</td>
<td>CT - Partition number out of range - high</td>
</tr>
<tr>
<td>17731</td>
<td>00731</td>
<td>FSS ctree error</td>
<td>CT - Illegal operation with primary key</td>
</tr>
<tr>
<td>17732</td>
<td>00732</td>
<td>FSS ctree error</td>
<td>CT - Illegal file mode / x8mode value</td>
</tr>
<tr>
<td>17733</td>
<td>00733</td>
<td>FSS ctree error</td>
<td>CT - Cannot open this member except RO</td>
</tr>
<tr>
<td>17734</td>
<td>00734</td>
<td>FSS ctree error</td>
<td>CT - Extended header (V7 file) required</td>
</tr>
<tr>
<td>17735</td>
<td>00735</td>
<td>FSS ctree error</td>
<td>CT - Client does not support UNIFRMAT server</td>
</tr>
<tr>
<td>17736</td>
<td>00736</td>
<td>FSS ctree error</td>
<td>CT - Async handle overflow</td>
</tr>
<tr>
<td>17737</td>
<td>00737</td>
<td>FSS ctree error</td>
<td>CT - Partition member characteristics do not match</td>
</tr>
<tr>
<td>17738</td>
<td>00738</td>
<td>FSS ctree error</td>
<td>CT - cThrdInit must be called first</td>
</tr>
<tr>
<td>17739</td>
<td>00739</td>
<td>FSS ctree error</td>
<td>CT - Record does not pass filter</td>
</tr>
<tr>
<td>17740</td>
<td>00740</td>
<td>FSS ctree error</td>
<td>CT - Dynamic dump refused SF member create</td>
</tr>
<tr>
<td>17741</td>
<td>00741</td>
<td>FSS ctree error</td>
<td>CT - Dynamic dump refused SF member delete</td>
</tr>
<tr>
<td>17742</td>
<td>00742</td>
<td>FSS ctree error</td>
<td>CT - Superfile member idx / host 6BTRAN !=</td>
</tr>
<tr>
<td>17743</td>
<td>00743</td>
<td>FSS ctree error</td>
<td>CT - Superfile host directory index null</td>
</tr>
<tr>
<td>17744</td>
<td>00744</td>
<td>FSS ctree error</td>
<td>CT - File requires unavailable feature</td>
</tr>
<tr>
<td>17745</td>
<td>00745</td>
<td>FSS ctree error</td>
<td>CT - 6BTRAN file required</td>
</tr>
<tr>
<td>17746</td>
<td>00746</td>
<td>FSS ctree error</td>
<td>CT - Could not allocate I/O channel(s)</td>
</tr>
<tr>
<td>17747</td>
<td>00747</td>
<td>FSS ctree error</td>
<td>CT - Partition rule syntax error</td>
</tr>
<tr>
<td>17748</td>
<td>00748</td>
<td>FSS ctree error</td>
<td>CT - Read failed external cause - check sysiocod</td>
</tr>
<tr>
<td>17749</td>
<td>00749</td>
<td>FSS ctree error</td>
<td>CT - Bad parameter value</td>
</tr>
<tr>
<td>17750</td>
<td>00750</td>
<td>FSS ctree error</td>
<td>CT - Different ICU version, rebuild index</td>
</tr>
<tr>
<td>17751</td>
<td>00751</td>
<td>FSS ctree error</td>
<td>CT - Checkpoint memory inconsistency</td>
</tr>
<tr>
<td>17752</td>
<td>00752</td>
<td>FSS ctree error</td>
<td>CT - More than one log set in transaction</td>
</tr>
<tr>
<td>17753</td>
<td>00753</td>
<td>FSS ctree error</td>
<td>CT - Cannot RST/CLR past special savepoint</td>
</tr>
<tr>
<td>17754</td>
<td>00754</td>
<td>FSS ctree error</td>
<td>CT - Only Q creator can perform operation</td>
</tr>
<tr>
<td>17755</td>
<td>00755</td>
<td>FSS ctree error</td>
<td>CT - A system queue is required</td>
</tr>
<tr>
<td>17756</td>
<td>00756</td>
<td>FSS ctree error</td>
<td>CT - Server is not activated</td>
</tr>
<tr>
<td>17757</td>
<td>00757</td>
<td>FSS ctree error</td>
<td>CT - Must uninit c-tree (STPUSR)</td>
</tr>
<tr>
<td>17758</td>
<td>00758</td>
<td>FSS ctree error</td>
<td>CT - Only notifications to queue</td>
</tr>
<tr>
<td>17759</td>
<td>00759</td>
<td>FSS ctree error</td>
<td>CT - wrong queue instance</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>17760</td>
<td>00760</td>
<td>FSS ctree error</td>
<td>CT - SYMON interrupted / cancelled</td>
</tr>
<tr>
<td>17761</td>
<td>00761</td>
<td>FSS ctree error</td>
<td>CT - No active SYMON</td>
</tr>
<tr>
<td>17762</td>
<td>00762</td>
<td>FSS ctree error</td>
<td>CT - No range defined for index</td>
</tr>
<tr>
<td>17763</td>
<td>00763</td>
<td>FSS ctree error</td>
<td>CT - Segment out of range</td>
</tr>
<tr>
<td>17764</td>
<td>00764</td>
<td>FSS ctree error</td>
<td>CT - Range defined but no FRS/LST RNG</td>
</tr>
<tr>
<td>17765</td>
<td>00765</td>
<td>FSS ctree error</td>
<td>CT - Comm I/O has been blocked-ctcomioblk has been set by user</td>
</tr>
<tr>
<td>17766</td>
<td>00766</td>
<td>FSS ctree error</td>
<td>CT - Insufficient file handles and could not find file to close for virtual processing</td>
</tr>
<tr>
<td>17767</td>
<td>00767</td>
<td>FSS ctree error</td>
<td>CT - Fixed length record offset not aligned</td>
</tr>
<tr>
<td>17768</td>
<td>00768</td>
<td>FSS ctree error</td>
<td>CT - Commit lock error</td>
</tr>
<tr>
<td>17769</td>
<td>00769</td>
<td>FSS ctree error</td>
<td>CT - Unexpected CMTLOK unlock failure</td>
</tr>
<tr>
<td>17770</td>
<td>00770</td>
<td>FSS ctree error</td>
<td>CT - Cannot turn off file's tran support in middle of transaction if file updated</td>
</tr>
<tr>
<td>17771</td>
<td>00771</td>
<td>FSS ctree error</td>
<td>CT - Compatibility option not enabled</td>
</tr>
<tr>
<td>17772</td>
<td>00772</td>
<td>FSS ctree error</td>
<td>CT - Platform does not support compatibility option</td>
</tr>
<tr>
<td>17773</td>
<td>00773</td>
<td>FSS ctree error</td>
<td>CT - Superfile host &amp; member must have same REPLICATION attribute</td>
</tr>
<tr>
<td>17774</td>
<td>00774</td>
<td>FSS ctree error</td>
<td>CT - Superfile members must all be closed</td>
</tr>
<tr>
<td>17775</td>
<td>00775</td>
<td>FSS ctree error</td>
<td>CT - No UNQKEY support for REPLICATION</td>
</tr>
<tr>
<td>17776</td>
<td>00776</td>
<td>FSS ctree error</td>
<td>CT - Operation not supported for SUPERFILES</td>
</tr>
<tr>
<td>17777</td>
<td>00777</td>
<td>FSS ctree error</td>
<td>CT - Cannot unregister another client's replication instance</td>
</tr>
<tr>
<td>17778</td>
<td>00778</td>
<td>FSS ctree error</td>
<td>CT - The specified replication instance name is not registered</td>
</tr>
<tr>
<td>17779</td>
<td>00779</td>
<td>FSS ctree error</td>
<td>CT - The specified replication instance name is already registered</td>
</tr>
<tr>
<td>17780</td>
<td>00780</td>
<td>FSS ctree error</td>
<td>CT - Cannot attach to replication instance with active connection</td>
</tr>
<tr>
<td>17781</td>
<td>00781</td>
<td>FSS ctree error</td>
<td>CT - This replication connection already has a registered instance name</td>
</tr>
<tr>
<td>17782</td>
<td>00782</td>
<td>FSS ctree error</td>
<td>CT - Decompression error</td>
</tr>
<tr>
<td>17783</td>
<td>00783</td>
<td>FSS ctree error</td>
<td>CT - Incremental index</td>
</tr>
<tr>
<td>17784</td>
<td>00784</td>
<td>FSS ctree error</td>
<td>CT - LDAP initialization failed</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>17785</td>
<td>00785</td>
<td>FSS ctree error</td>
<td>CT - Error connecting to LDAP server</td>
</tr>
<tr>
<td>17786</td>
<td>00786</td>
<td>FSS ctree error</td>
<td>CT - Error binding to LDAP server</td>
</tr>
<tr>
<td>17787</td>
<td>00787</td>
<td>FSS ctree error</td>
<td>CT - LDAP user authentication failed</td>
</tr>
<tr>
<td>17788</td>
<td>00788</td>
<td>FSS ctree error</td>
<td>CT - Error checking user's LDAP groups</td>
</tr>
<tr>
<td>17789</td>
<td>00789</td>
<td>FSS ctree error</td>
<td>CT - Member of c-tree login group required</td>
</tr>
<tr>
<td>17790</td>
<td>00790</td>
<td>FSS ctree error</td>
<td>CT - Strict serializer must be in transaction to access record</td>
</tr>
<tr>
<td>17791</td>
<td>00791</td>
<td>FSS ctree error</td>
<td>CT - Strict serializer cannot keep locks</td>
</tr>
<tr>
<td>17792</td>
<td>00792</td>
<td>FSS ctree error</td>
<td>CT - External server shutdown disabled</td>
</tr>
<tr>
<td>17793</td>
<td>00793</td>
<td>FSS ctree error</td>
<td>CT - Partial record rewrite has keys spanning partial record and remaining region of record</td>
</tr>
<tr>
<td>17794</td>
<td>00794</td>
<td>FSS ctree error</td>
<td>CT - Tried to update data with missing key buffer contents</td>
</tr>
<tr>
<td>17796</td>
<td>00796</td>
<td>FSS ctree error</td>
<td>CT - File copy failed</td>
</tr>
<tr>
<td>17797</td>
<td>00797</td>
<td>FSS ctree error</td>
<td>CT - Immediate dump restore failed</td>
</tr>
<tr>
<td>17798</td>
<td>00798</td>
<td>FSS ctree error</td>
<td>CT - File is blocked, retry later</td>
</tr>
<tr>
<td>17799</td>
<td>00799</td>
<td>FSS ctree error</td>
<td>CT - File block cleared</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17800</td>
<td>00800</td>
<td>FSS ctree error</td>
<td>CT - TRANDEP file operation pending</td>
</tr>
<tr>
<td>17801</td>
<td>00801</td>
<td>FSS ctree error</td>
<td>CT - Operation terminated by file block request</td>
</tr>
<tr>
<td>17802</td>
<td>00802</td>
<td>FSS ctree error</td>
<td>CT - File block not supported for file type</td>
</tr>
<tr>
<td>17803</td>
<td>00803</td>
<td>FSS ctree error</td>
<td>CT - Existing lock not replaced</td>
</tr>
<tr>
<td>17804</td>
<td>00804</td>
<td>FSS ctree error</td>
<td>CT - Cannot override configuration option that was specified in settings file</td>
</tr>
<tr>
<td>17805</td>
<td>00805</td>
<td>FSS ctree error</td>
<td>CT - No memory for function stack allocation</td>
</tr>
<tr>
<td>17806</td>
<td>00806</td>
<td>FSS ctree error</td>
<td>CT - Failed to connect to kernel engine</td>
</tr>
<tr>
<td>17807</td>
<td>00807</td>
<td>FSS ctree error</td>
<td>CT - ctDBGstack limit exceeded</td>
</tr>
<tr>
<td>17808</td>
<td>00808</td>
<td>FSS ctree error</td>
<td>CT - Request timed out</td>
</tr>
<tr>
<td>17809</td>
<td>00809</td>
<td>FSS ctree error</td>
<td>CT - Response timed out</td>
</tr>
<tr>
<td>17810</td>
<td>00810</td>
<td>FSS ctree error</td>
<td>CT - Status log write failure</td>
</tr>
<tr>
<td>17811</td>
<td>00811</td>
<td>FSS ctree error</td>
<td>CT - Update only from internal API</td>
</tr>
<tr>
<td>17812</td>
<td>00812</td>
<td>FSS ctree error</td>
<td>CT - Unexpected zero node size in header</td>
</tr>
<tr>
<td>17813</td>
<td>00813</td>
<td>FSS ctree error</td>
<td>CT - Unexpected state for user block object</td>
</tr>
<tr>
<td>17814</td>
<td>00814</td>
<td>FSS ctree error</td>
<td>CT - Thread did not have ctFBActive set</td>
</tr>
<tr>
<td>17815</td>
<td>00815</td>
<td>FSS ctree error</td>
<td>CT - File block already in progress for file</td>
</tr>
<tr>
<td>17816</td>
<td>00816</td>
<td>FSS ctree error</td>
<td>CT - Only one ctQUIET process at a time</td>
</tr>
<tr>
<td>17817</td>
<td>00817</td>
<td>FSS ctree error</td>
<td>CT - Transaction aborted by quiesce operation</td>
</tr>
<tr>
<td>17818</td>
<td>00818</td>
<td>FSS ctree error</td>
<td>CT - Quiesce / file block conflict</td>
</tr>
<tr>
<td>17819</td>
<td>00819</td>
<td>FSS ctree error</td>
<td>CT - Unexpected failure to block thread</td>
</tr>
<tr>
<td>17820</td>
<td>00820</td>
<td>FSS ctree error</td>
<td>CT - Trying to quiesce; leave core</td>
</tr>
<tr>
<td>17821</td>
<td>00821</td>
<td>FSS ctree error</td>
<td>CT - Transaction abandoned - MAX_USER_LOGS</td>
</tr>
<tr>
<td>17822</td>
<td>00822</td>
<td>FSS ctree error</td>
<td>CT - Abort request would be suspended</td>
</tr>
<tr>
<td>17824</td>
<td>00824</td>
<td>FSS ctree error</td>
<td>CT - Problem impersonating thread</td>
</tr>
<tr>
<td>17825</td>
<td>00825</td>
<td>FSS ctree error</td>
<td>CT - ctQUIET called with files opened</td>
</tr>
<tr>
<td>17826</td>
<td>00826</td>
<td>FSS ctree error</td>
<td>CT - Improper ctQT actions - see sysicod</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17827</td>
<td>00827</td>
<td>FSS ctree error</td>
<td>CT - User block timed out</td>
</tr>
<tr>
<td>17830</td>
<td>00830</td>
<td>FSS ctree error</td>
<td>CT - Transaction abandoned - TRAN_TIMEOUT</td>
</tr>
<tr>
<td>17831</td>
<td>00831</td>
<td>FSS ctree error</td>
<td>CT - Abort request would be suspended</td>
</tr>
<tr>
<td>17833</td>
<td>00833</td>
<td>FSS ctree error</td>
<td>CT - The connection attempt has been rejected because it would exceed the maximum number of concurrent client machines allowed</td>
</tr>
<tr>
<td>17834</td>
<td>00834</td>
<td>FSS ctree error</td>
<td>CT - The connection attempt has been rejected because it would exceed the maximum number of concurrent connections allowed from this client machine</td>
</tr>
<tr>
<td>17835</td>
<td>00835</td>
<td>FSS ctree error</td>
<td>CT - The transaction was aborted before the requested operation was processed</td>
</tr>
<tr>
<td>17836</td>
<td>00836</td>
<td>FSS ctree error</td>
<td>CT - The connection attempt has been rejected because only connections from the local system are allowed</td>
</tr>
<tr>
<td>17838</td>
<td>00838</td>
<td>FSS ctree error</td>
<td>CT - clnleaf(CLNIDXX) failed to clean node</td>
</tr>
<tr>
<td>17839</td>
<td>00839</td>
<td>FSS ctree error</td>
<td>CT - index file requires key level lock cleaning</td>
</tr>
<tr>
<td>17841</td>
<td>00841</td>
<td>FSS ctree error</td>
<td>CT - A connection attempt using the shared memory protocol failed due to an incompatibility between client and server</td>
</tr>
<tr>
<td>17842</td>
<td>00842</td>
<td>FSS ctree error</td>
<td>CT - Could not clear all threads from core, abort ctFILBLK attempt</td>
</tr>
<tr>
<td>17843</td>
<td>00843</td>
<td>FSS ctree error</td>
<td>CT - Could not clear all threads from core, abort ctQUIET attempt</td>
</tr>
<tr>
<td>17845</td>
<td>00845</td>
<td>FSS ctree error</td>
<td>CT - Index does not have distinct attribute</td>
</tr>
<tr>
<td>17846</td>
<td>00846</td>
<td>FSS ctree error</td>
<td>CT - ctDISTINCTset update in progress</td>
</tr>
<tr>
<td>17847</td>
<td>00847</td>
<td>FSS ctree error</td>
<td>CT - Auto recovery interrupted</td>
</tr>
<tr>
<td>17850</td>
<td>00850</td>
<td>FSS ctree error</td>
<td>CT - Transactional replication - Failed to start c-tree remote client subsystem - see CTSTATUS.FCS for details</td>
</tr>
<tr>
<td>17851</td>
<td>00851</td>
<td>FSS ctree error</td>
<td>CT - Transactional replication - To open replication mapping file - see CTSTATUS.FCS for details</td>
</tr>
<tr>
<td>17852</td>
<td>00852</td>
<td>FSS ctree error</td>
<td>CT - Transactional replication - Failed to open replica for writing because the file does not meet the requirements for replication - see CTSTATUS.FCS for details</td>
</tr>
<tr>
<td>17853</td>
<td>00853</td>
<td>FSS ctree error</td>
<td>CT - Transactional replication - Failed to set node name for connection to master server - see CTSTATUS.FCS for details</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17854</td>
<td>00854</td>
<td>FSS ctree error</td>
<td>CT - Transactional replication - Attempted to update replica without enabling transactional replication for the connection</td>
</tr>
<tr>
<td>17855</td>
<td>00855</td>
<td>FSS ctree error</td>
<td>CT - Transactional replication - Failed to establish a savepoint because savepoint number is out of sync with master</td>
</tr>
<tr>
<td>17856</td>
<td>00856</td>
<td>FSS ctree error</td>
<td>CT - Transactional replication - Failed to read record for update - local record differs from master record</td>
</tr>
<tr>
<td>17857</td>
<td>00857</td>
<td>FSS ctree error</td>
<td>CT - The superfile host is open in exclusive mode</td>
</tr>
<tr>
<td>17858</td>
<td>00858</td>
<td>FSS ctree error</td>
<td>CT - Transactional replication - This feature is not supported for replicas</td>
</tr>
<tr>
<td>17859</td>
<td>00859</td>
<td>FSS ctree error</td>
<td>CT - The client's structure definition for the file FAIRCOM.FCS!USER.dat is out of date. Update client library</td>
</tr>
<tr>
<td>17860</td>
<td>00860</td>
<td>FSS ctree error</td>
<td>CT - The server's structure definition for the file FAIRCOM.FCS!USER.dat is out of date. Update c-tree Server.</td>
</tr>
<tr>
<td>17861</td>
<td>00861</td>
<td>FSS ctree error</td>
<td>CT - After recovery, a key level lock for an undone transaction discovered in optional diagnostic scan</td>
</tr>
<tr>
<td>17862</td>
<td>00862</td>
<td>FSS ctree error</td>
<td>CT - LOCK_CACHE - System pagesize query failed</td>
</tr>
<tr>
<td>17863</td>
<td>00863</td>
<td>FSS ctree error</td>
<td>CT - The request to impersonate the specified connection was denied because the target connection does not allow impersonation.</td>
</tr>
<tr>
<td>17864</td>
<td>00864</td>
<td>FSS ctree error</td>
<td>CT - The request to impersonate the specified connection was denied because the target connection does not allow impersonation by the specified connection.</td>
</tr>
<tr>
<td>17865</td>
<td>00865</td>
<td>FSS ctree error</td>
<td>CT - The request to impersonate the specified connection was denied because the target connection is already being impersonated.</td>
</tr>
<tr>
<td>17866</td>
<td>00866</td>
<td>FSS ctree error</td>
<td>CT - The request to impersonate the specified connection was denied because the target connection is executing a database operation or is blocked.</td>
</tr>
<tr>
<td>17867</td>
<td>00867</td>
<td>FSS ctree error</td>
<td>CT - Failed to load the filter callback library. See CTSTATUS.FCS for details.</td>
</tr>
<tr>
<td>17868</td>
<td>00868</td>
<td>FSS ctree error</td>
<td>CT - Failed to resolve the filter callback function in the filter callback DLL. See CTSTATUS.FCS for details.</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17869</td>
<td>00869</td>
<td>FSS ctree error</td>
<td>CT - A user-defined callback function terminated the rebuild operation.</td>
</tr>
<tr>
<td>17870</td>
<td>00870</td>
<td>FSS ctree error</td>
<td>CT - The filter callback DLL version is not compatible with the c-tree Server's filter callback version. See CTSTATUS.FCS for details.</td>
</tr>
<tr>
<td>17871</td>
<td>00871</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - the source file could not be opened for reading.</td>
</tr>
<tr>
<td>17872</td>
<td>00872</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - the destination file could not be opened for writing.</td>
</tr>
<tr>
<td>17873</td>
<td>00873</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - the source file could not be read.</td>
</tr>
<tr>
<td>17874</td>
<td>00874</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - the destination file could not be written.</td>
</tr>
<tr>
<td>17875</td>
<td>00875</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - bound database call not supported.</td>
</tr>
<tr>
<td>17876</td>
<td>00876</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - invalid file transfer block size.</td>
</tr>
<tr>
<td>17877</td>
<td>00877</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - NULL or empty source file name.</td>
</tr>
<tr>
<td>17878</td>
<td>00878</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - NULL or empty destination file name.</td>
</tr>
<tr>
<td>17879</td>
<td>00879</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - incompatible file transfer structure</td>
</tr>
<tr>
<td>17880</td>
<td>00880</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - the destination file exists and the caller did not specify that the destination file is to be overwritten.</td>
</tr>
<tr>
<td>17881</td>
<td>00881</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - the server does not support the transactional replication feature.</td>
</tr>
<tr>
<td>17882</td>
<td>00882</td>
<td>FSS ctree error</td>
<td>CT - File transfer failed - the server is not configured as a local server. Use the REPL_MAPPINGS option to configure the server as a local server.</td>
</tr>
<tr>
<td>17883</td>
<td>00883</td>
<td>FSS ctree error</td>
<td>CT - An attempt was made to open a file multiple times by the same user with different user file numbers but this support is disabled</td>
</tr>
<tr>
<td>17884</td>
<td>00884</td>
<td>FSS ctree error</td>
<td>CT - User file number mismatch during lock or unlock operations.</td>
</tr>
<tr>
<td>17885</td>
<td>00885</td>
<td>FSS ctree error</td>
<td>CT - API call to perform a checkpoint took no action because a checkpoint was already in progress.</td>
</tr>
<tr>
<td>17886</td>
<td>00886</td>
<td>FSS ctree error</td>
<td>CT - Unexpected CHG_UFLOCK failure - call FairCom.</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>17887</td>
<td>00887</td>
<td>FSS ctree error</td>
<td>CT - Promoting a secondary lock from read to write is not supported.</td>
</tr>
<tr>
<td>17888</td>
<td>00888</td>
<td>FSS ctree error</td>
<td>CT - The specified task ID does not correspond to an active client connection.</td>
</tr>
<tr>
<td>17889</td>
<td>00889</td>
<td>FSS ctree error</td>
<td>CT - The specified connection is a member of a connection group.</td>
</tr>
<tr>
<td>17890</td>
<td>00890</td>
<td>FSS ctree error</td>
<td>CT - An attempt to get the name of the host system failed.</td>
</tr>
<tr>
<td>17891</td>
<td>00891</td>
<td>FSS ctree error</td>
<td>CT - An attempt to get the IP addresses associated with the host system failed.</td>
</tr>
<tr>
<td>17892</td>
<td>00892</td>
<td>FSS ctree error</td>
<td>CT - Sequential, index based retrieval when the index failed - ctfSAMKBUFhdr has turned off.</td>
</tr>
<tr>
<td>17893</td>
<td>00893</td>
<td>FSS ctree error</td>
<td>CT - Sequential, index based retrieval when the index failed - partial record read or changing current ISAM record location.</td>
</tr>
<tr>
<td>17894</td>
<td>00894</td>
<td>FSS ctree error</td>
<td>CT - Sequential, index based retrieval when the index failed - conditional index constraint or NUL key results in no key value</td>
</tr>
<tr>
<td>17895</td>
<td>00895</td>
<td>FSS ctree error</td>
<td>CT - Partition index segment definition error.</td>
</tr>
<tr>
<td>17896</td>
<td>00896</td>
<td>FSS ctree error</td>
<td>CT - Covering index segment definitions do not map to partition index - Call FairCom.</td>
</tr>
<tr>
<td>17897</td>
<td>00897</td>
<td>FSS ctree error</td>
<td>CT - Covering index segment map (ptmap) is empty.</td>
</tr>
<tr>
<td>17898</td>
<td>00898</td>
<td>FSS ctree error</td>
<td>CT - Abandoned ctQUIET blocks attempts to validate user logons.</td>
</tr>
<tr>
<td>17899</td>
<td>00899</td>
<td>FSS ctree error</td>
<td>CT - Cannot perform ISAM Context operations on partition member.</td>
</tr>
</tbody>
</table>
## Error Messages

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>17900</td>
<td>00900</td>
<td>FSS ctree error</td>
<td>CT - A sequence having the specified name already exists.</td>
</tr>
<tr>
<td>17901</td>
<td>00901</td>
<td>FSS ctree error</td>
<td>CT - An invalid sequence name was specified.</td>
</tr>
<tr>
<td>17902</td>
<td>00902</td>
<td>FSS ctree error</td>
<td>CT - The specified sequence handle is invalid.</td>
</tr>
<tr>
<td>17903</td>
<td>00903</td>
<td>FSS ctree error</td>
<td>CT - Invalid combination of sequence type options.</td>
</tr>
<tr>
<td>17904</td>
<td>00904</td>
<td>FSS ctree error</td>
<td>CT - The initial value specified for the sequence is out of range.</td>
</tr>
<tr>
<td>17905</td>
<td>00905</td>
<td>FSS ctree error</td>
<td>CT - The current value specified for the sequence is out of range.</td>
</tr>
<tr>
<td>17906</td>
<td>00906</td>
<td>FSS ctree error</td>
<td>CT - The limit value specified for the sequence is out of range.</td>
</tr>
<tr>
<td>17907</td>
<td>00907</td>
<td>FSS ctree error</td>
<td>CT - The increment value specified for the sequence is out of range.</td>
</tr>
<tr>
<td>17908</td>
<td>00908</td>
<td>FSS ctree error</td>
<td>CT - Maximum number of partial key distinct counts supported by the file is greater than the system support limit.</td>
</tr>
<tr>
<td>17909</td>
<td>00909</td>
<td>FSS ctree error</td>
<td>CT - The structure version specified for the input and output structures is not supported by this version of the code.</td>
</tr>
<tr>
<td>17910</td>
<td>00910</td>
<td>FSS ctree error</td>
<td>CT - A user-defined callback function terminated the index verify operation.</td>
</tr>
<tr>
<td>17911</td>
<td>00911</td>
<td>FSS ctree error</td>
<td>CT - File contains a Direct Access Resource (DAR) that is not supported.</td>
</tr>
<tr>
<td>17912</td>
<td>00912</td>
<td>FSS ctree error</td>
<td>CT - Special resource can only be added by system routine.</td>
</tr>
<tr>
<td>17913</td>
<td>00913</td>
<td>FSS ctree error</td>
<td>CT - Duplicate DAR not allowed.</td>
</tr>
<tr>
<td>17914</td>
<td>00914</td>
<td>FSS ctree error</td>
<td>CT - Requested DAR does not exist.</td>
</tr>
<tr>
<td>17915</td>
<td>00915</td>
<td>FSS ctree error</td>
<td>CT - Log file requires augmented log entries that are not supported.</td>
</tr>
<tr>
<td>17916</td>
<td>00916</td>
<td>FSS ctree error</td>
<td>CT - Mismatch between header DAR count, and DARs found in resource chain.</td>
</tr>
<tr>
<td>17917</td>
<td>00917</td>
<td>FSS ctree error</td>
<td>CT - Reached limit on the number of instances of a particular DAR type for a single file.</td>
</tr>
<tr>
<td>17918</td>
<td>00918</td>
<td>FSS ctree error</td>
<td>CT - DAR attribute word is marked deleted.</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>17919</td>
<td>00919</td>
<td>FSS ctree error</td>
<td>CT - Low-level operations are not allowed on replicated files.</td>
</tr>
<tr>
<td>17920</td>
<td>00920</td>
<td>FSS ctree error</td>
<td>CT - The file exists but could not be accessed- check for permission or sharing restrictions.</td>
</tr>
<tr>
<td>17921</td>
<td>00921</td>
<td>FSS ctree error</td>
<td>CT - Attempt to change IDfield during an ISAM rewrite operation.</td>
</tr>
<tr>
<td>17922</td>
<td>00922</td>
<td>FSS ctree error</td>
<td>CT - Attempt an ISAM rewrite operation without current ISAM IDfield buffer contents.</td>
</tr>
<tr>
<td>17923</td>
<td>00923</td>
<td>FSS ctree error</td>
<td>CT - An internal error has occurred; similar to a catastrophic err. See CTSTATUS.FCS for details.</td>
</tr>
<tr>
<td>17924</td>
<td>00924</td>
<td>FSS ctree error</td>
<td>CT - Only the super administrator user account (named ADMIN) can perform this operation.</td>
</tr>
<tr>
<td>17925</td>
<td>00925</td>
<td>FSS ctree error</td>
<td>CT - The c-tree client is attempting to use features of the SECURITY API function that this c-tree Server does not support. Update your c-tree Server.</td>
</tr>
<tr>
<td>17926</td>
<td>00926</td>
<td>FSS ctree error</td>
<td>CT - The transaction history log scan terminated because the user-specified limit on the number of logs to scan was reached.</td>
</tr>
<tr>
<td>17938</td>
<td>00938</td>
<td>FSS ctree error</td>
<td>CT - Could not decompress the data record.</td>
</tr>
<tr>
<td>17941</td>
<td>00941</td>
<td>FSS ctree error</td>
<td>CT - This c-tree Server requires a secure logon that your c-tree client library does not support. Update your c-tree client library.</td>
</tr>
<tr>
<td>17942</td>
<td>00942</td>
<td>FSS ctree error</td>
<td>CT - This c-tree client uses a secure logon that your c-tree Server does not support. Update your c-tree Server.</td>
</tr>
<tr>
<td>17959</td>
<td>00959</td>
<td>FSS ctree error</td>
<td>CT - This c-tree client uses a different secure logon version than your c-tree Server. Update your c-tree client library.</td>
</tr>
<tr>
<td>17965</td>
<td>00965</td>
<td>FSS ctree error</td>
<td>CT - c-tree Server was not able to decrypt the buffer sent to it by the client. This is an unexpected error. Contact FairCom support.</td>
</tr>
<tr>
<td>17967</td>
<td>00967</td>
<td>FSS ctree error</td>
<td>CT - Logon is denied because this user account has reached its maximum number of concurrent logons.</td>
</tr>
<tr>
<td>17968</td>
<td>00968</td>
<td>FSS ctree error</td>
<td>CT - Logon is denied because one of the groups for this user account has reached its maximum number of concurrent logons.</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>17970</td>
<td>00970</td>
<td>FSS ctree error</td>
<td>CT - The connection attempt has been rejected because it would exceed the maximum number of concurrent ISAM connections allowed.</td>
</tr>
<tr>
<td>17971</td>
<td>00971</td>
<td>FSS ctree error</td>
<td>CT - The connection attempt has been rejected because it would exceed the maximum number of concurrent SQL connections allowed.</td>
</tr>
<tr>
<td>17979</td>
<td>00979</td>
<td>FSS ctree error</td>
<td>CT - c-tree is not licensed to use this feature.</td>
</tr>
<tr>
<td>17980</td>
<td>00980</td>
<td>FSS ctree error</td>
<td>CT - Secure key exchange failed.</td>
</tr>
<tr>
<td>17981</td>
<td>00981</td>
<td>FSS ctree error</td>
<td>CT - Could not load the specified DLL or shared library.</td>
</tr>
<tr>
<td>17982</td>
<td>00982</td>
<td>FSS ctree error</td>
<td>CT - Could not resolve the specified function name in the specified DLL or shared library.</td>
</tr>
<tr>
<td>17984</td>
<td>00984</td>
<td>FSS ctree error</td>
<td>CT - Logon is denied because the number of distinct user accounts that are allowed to be connected at one time has been reached.</td>
</tr>
<tr>
<td>17985</td>
<td>00985</td>
<td>FSS ctree error</td>
<td>CT - Logon is denied because this user account requires LDAP authentication, but c-tree Server has not enabled LDAP authentication.</td>
</tr>
<tr>
<td>17990</td>
<td>00990</td>
<td>FSS ctree error</td>
<td>CT - This operation is not supported for a dynamic partitioned file.</td>
</tr>
<tr>
<td>17991</td>
<td>00991</td>
<td>FSS ctree error</td>
<td>CT - No partition members have been associated with this dynamic partition host file.</td>
</tr>
<tr>
<td>17993</td>
<td>00993</td>
<td>FSS ctree error</td>
<td>CT - The table, index, record, or field definitions of the partition member differ from those of the dynamic partition host file.</td>
</tr>
</tbody>
</table>
### 18000 Series

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>18001</td>
<td>01001</td>
<td>FSS ctree error</td>
<td>CT - Could not update the specified key value because this index does not allow an ISAM record update to change the key value.</td>
</tr>
<tr>
<td>18002</td>
<td>01002</td>
<td>FSS ctree error</td>
<td>CT - Identity field overflow/underflow error.</td>
</tr>
<tr>
<td>18003</td>
<td>01003</td>
<td>FSS ctree error</td>
<td>CT - Identity field invalid extended attribute error.</td>
</tr>
<tr>
<td>18006</td>
<td>01005</td>
<td>FSS ctree error</td>
<td>CT - Inconsistent co-file lock attribute for file.</td>
</tr>
<tr>
<td>18007</td>
<td>01006</td>
<td>FSS ctree error</td>
<td>CT - A superfile can only have one open instance at a time per connection.</td>
</tr>
<tr>
<td>18008</td>
<td>01007</td>
<td>FSS ctree error</td>
<td>CT - A partitioned file can only have one open instance at a time per connection.</td>
</tr>
<tr>
<td>18024</td>
<td>01024</td>
<td>FSS ctree error</td>
<td>CT - Could not obtain data record lock because the table is locked, or a request to lock the table is pending.</td>
</tr>
<tr>
<td>18025</td>
<td>01025</td>
<td>FSS ctree error</td>
<td>CT - Could not obtain table lock because the table is locked, or a request to lock the table is pending, or a conflicting data record read or write lock exists.</td>
</tr>
<tr>
<td>18026</td>
<td>01026</td>
<td>FSS ctree error</td>
<td>CT - Could not update the table because the table is locked.</td>
</tr>
<tr>
<td>18030</td>
<td>01030</td>
<td>FSS ctree error</td>
<td>CT - Maximum concurrent active connections exceeded.</td>
</tr>
<tr>
<td>18104</td>
<td>01104</td>
<td>FSS ctree error</td>
<td>CT - SSL connection attempt failed.</td>
</tr>
</tbody>
</table>
# 4.7 -202xx - Sequence Support

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Returned By</th>
</tr>
</thead>
<tbody>
<tr>
<td>-20265</td>
<td>Sequence with the same name already exists</td>
<td>CREATE SEQUENCE</td>
</tr>
<tr>
<td>-20266</td>
<td>Sequence cannot be used here</td>
<td>DROP SEQUENCE</td>
</tr>
<tr>
<td>-20267</td>
<td>Sequence not found</td>
<td>DROP SEQUENCE</td>
</tr>
<tr>
<td>-20268</td>
<td>START-WITH/CURRENT-VALUE cannot be greater than MAXVALUE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20269</td>
<td>START-WITH/CURRENT-VALUE cannot be less than MINVALUE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20270</td>
<td>Invalid sequence MINVALUE specified</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20271</td>
<td>Invalid sequence INCREMENT specified</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20272</td>
<td>START-WITH cannot be altered in sequence</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20273</td>
<td>No options specified for ALTER SEQUENCE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20274</td>
<td>Sequence increment has exceeded MAXVALUE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20275</td>
<td>Sequence decrement has exceeded MINVALUE</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
<tr>
<td>-20276</td>
<td>Only SELECT and ALTER privileges are valid for sequences</td>
<td>CREATE SEQUENCE ALTER SEQUENCE</td>
</tr>
</tbody>
</table>
## Error Messages

### 4.8 20000 Series - c-treeACE SQL

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>20000</td>
<td>50501</td>
<td>c-treeACE SQL RDS error</td>
<td>SQL internal error</td>
</tr>
<tr>
<td>20001</td>
<td>50502</td>
<td>c-treeACE SQL RDS error</td>
<td>Memory allocation failure</td>
</tr>
<tr>
<td>20002</td>
<td>50503</td>
<td>c-treeACE SQL RDS error</td>
<td>Open database failed</td>
</tr>
<tr>
<td>20003</td>
<td>2a504</td>
<td>Syntax error</td>
<td>Syntax error</td>
</tr>
<tr>
<td>20004</td>
<td>28505</td>
<td>Invalid auth specs</td>
<td>User %s not found</td>
</tr>
<tr>
<td>20005</td>
<td>22506</td>
<td>Data exception</td>
<td>Table/View/Synonym %s not found</td>
</tr>
<tr>
<td>20006</td>
<td>22507</td>
<td>Data exception</td>
<td>Column %s%s%s not found/specified</td>
</tr>
<tr>
<td>20007</td>
<td>22508</td>
<td>Data exception</td>
<td>No columns in table</td>
</tr>
<tr>
<td>20008</td>
<td>22509</td>
<td>Data exception</td>
<td>Inconsistent types</td>
</tr>
<tr>
<td>20009</td>
<td>22510</td>
<td>Data exception</td>
<td>Column %s ambiguously specified</td>
</tr>
<tr>
<td>20010</td>
<td>22511</td>
<td>Data exception</td>
<td>Duplicate column %s specification</td>
</tr>
<tr>
<td>20011</td>
<td>22512</td>
<td>Data exception</td>
<td>Invalid length at column %s</td>
</tr>
<tr>
<td>20012</td>
<td>22513</td>
<td>Data exception</td>
<td>Invalid precision at column %s</td>
</tr>
<tr>
<td>20013</td>
<td>22514</td>
<td>Data exception</td>
<td>Invalid scale at column %s</td>
</tr>
<tr>
<td>20014</td>
<td>22515</td>
<td>Data exception</td>
<td>Missing input parameters</td>
</tr>
<tr>
<td>20015</td>
<td>22516</td>
<td>Data exception</td>
<td>Subquery returns multiple rows</td>
</tr>
<tr>
<td>20016</td>
<td>22517</td>
<td>Data exception</td>
<td>Null value supplied for a mandatory (not null) column %s</td>
</tr>
<tr>
<td>20017</td>
<td>22518</td>
<td>Data exception</td>
<td>Too many values specified</td>
</tr>
<tr>
<td>20018</td>
<td>22519</td>
<td>Data exception</td>
<td>Too few values specified</td>
</tr>
<tr>
<td>20019</td>
<td>50520</td>
<td>c-treeACE SQL RDS error</td>
<td>Cannot modify table %s referred to in subquery</td>
</tr>
<tr>
<td>20020</td>
<td>42521</td>
<td>Access rule violation</td>
<td>Bad column %s specification for group by clause</td>
</tr>
<tr>
<td>20021</td>
<td>42522</td>
<td>Access rule violation</td>
<td>Non-group-by expression in having clause</td>
</tr>
<tr>
<td>20022</td>
<td>42523</td>
<td>Access rule violation</td>
<td>Non-group-by expression in select clause</td>
</tr>
<tr>
<td>20023</td>
<td>42524</td>
<td>Access rule violation</td>
<td>Aggregate function not allowed here</td>
</tr>
<tr>
<td>20024</td>
<td>0a000</td>
<td>feature not supported</td>
<td>Sorry, operation not yet implemented</td>
</tr>
<tr>
<td>20025</td>
<td>42526</td>
<td>Access rule violation</td>
<td>Aggregate functions nested</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>20026</td>
<td>50527</td>
<td>c-treeACE SQL RDS error</td>
<td>Too many table references</td>
</tr>
<tr>
<td>20027</td>
<td>42528</td>
<td>Access rule violation</td>
<td>Bad field specification in order by clause</td>
</tr>
<tr>
<td>20028</td>
<td>50529</td>
<td>c-treeACE SQL RDS error</td>
<td>Index with the same name %s already exists</td>
</tr>
<tr>
<td>20029</td>
<td>50530</td>
<td>c-treeACE SQL RDS error</td>
<td>Index referenced %s not found</td>
</tr>
<tr>
<td>20030</td>
<td>22531</td>
<td>Data exception</td>
<td>Table space with same name already exists</td>
</tr>
<tr>
<td>20031</td>
<td>50532</td>
<td>c-treeACE SQL RDS error</td>
<td>Cluster with same name already exists</td>
</tr>
<tr>
<td>20032</td>
<td>50533</td>
<td>c-treeACE SQL RDS error</td>
<td>No cluster with this name</td>
</tr>
<tr>
<td>20033</td>
<td>22534</td>
<td>Data exception</td>
<td>Tablespace not found</td>
</tr>
<tr>
<td>20034</td>
<td>50535</td>
<td>c-treeACE SQL RDS error</td>
<td>Bad free percentage specification</td>
</tr>
<tr>
<td>20035</td>
<td>50536</td>
<td>c-treeACE SQL RDS error</td>
<td>At least column spec or null clause should be specified</td>
</tr>
<tr>
<td>20036</td>
<td>07537</td>
<td>dynamic SQL error</td>
<td>Statement not prepared</td>
</tr>
<tr>
<td>20037</td>
<td>24538</td>
<td>Invalid cursor state</td>
<td>Executing select statement</td>
</tr>
<tr>
<td>20038</td>
<td>24539</td>
<td>Invalid cursor state</td>
<td>Cursor not closed</td>
</tr>
<tr>
<td>20039</td>
<td>24540</td>
<td>Invalid cursor state</td>
<td>Open for non-Select statement</td>
</tr>
<tr>
<td>20040</td>
<td>24541</td>
<td>Invalid cursor state</td>
<td>Cursor not opened</td>
</tr>
<tr>
<td>20041</td>
<td>22542</td>
<td>Data exception</td>
<td>Table/View/Synonym %s already exists</td>
</tr>
<tr>
<td>20042</td>
<td>2a543</td>
<td>Syntax error</td>
<td>Distinct specified more than once in query</td>
</tr>
<tr>
<td>20043</td>
<td>50544</td>
<td>c-treeACE SQL RDS error</td>
<td>Tuple size too high</td>
</tr>
<tr>
<td>20044</td>
<td>50545</td>
<td>c-treeACE SQL RDS error</td>
<td>Array size too high</td>
</tr>
<tr>
<td>20045</td>
<td>08546</td>
<td>Connection exception</td>
<td>File does not exist or not accessible</td>
</tr>
<tr>
<td>20046</td>
<td>50547</td>
<td>c-treeACE SQL RDS error</td>
<td>Field value not null for some tuples</td>
</tr>
<tr>
<td>20047</td>
<td>42548</td>
<td>Access rule violation</td>
<td>Granting to self not allowed</td>
</tr>
<tr>
<td>20048</td>
<td>42549</td>
<td>Access rule violation</td>
<td>Revoking for self not allowed</td>
</tr>
<tr>
<td>20049</td>
<td>22550</td>
<td>Data exception</td>
<td>Keyword %s used for a name</td>
</tr>
<tr>
<td>20050</td>
<td>21551</td>
<td>Cardinality violation</td>
<td>Too many fields specified</td>
</tr>
<tr>
<td>20051</td>
<td>21552</td>
<td>Cardinality violation</td>
<td>Too many indexes on the table %s</td>
</tr>
<tr>
<td>20052</td>
<td>22553</td>
<td>Data exception</td>
<td>Overflow/Underflow error</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>20053</td>
<td>08554</td>
<td>Connection exception</td>
<td>Database not opened</td>
</tr>
<tr>
<td>20054</td>
<td>08555</td>
<td>Connection exception</td>
<td>Database not specified or improperly specified</td>
</tr>
<tr>
<td>20055</td>
<td>08556</td>
<td>Connection exception</td>
<td>Database not specified or Database not started</td>
</tr>
<tr>
<td>20056</td>
<td>28557</td>
<td>Invalid auth specs</td>
<td>No DBA access rights</td>
</tr>
<tr>
<td>20057</td>
<td>28558</td>
<td>Invalid auth specs</td>
<td>User %s has no RESOURCE privileges</td>
</tr>
<tr>
<td>20058</td>
<td>40559</td>
<td>Transaction</td>
<td>Rollback</td>
</tr>
<tr>
<td>20059</td>
<td>22560</td>
<td>Data exception</td>
<td>No files in the table space</td>
</tr>
<tr>
<td>20060</td>
<td>22561</td>
<td>Data exception</td>
<td>Table %s not empty</td>
</tr>
<tr>
<td>20061</td>
<td>22562</td>
<td>Data exception</td>
<td>Input parameter size too high</td>
</tr>
<tr>
<td>20062</td>
<td>42563</td>
<td>Syntax error</td>
<td>Full pathname not specified</td>
</tr>
<tr>
<td>20063</td>
<td>50564</td>
<td>c-treeACE SQL RDS error</td>
<td>Duplicate file specification</td>
</tr>
<tr>
<td>20064</td>
<td>08565</td>
<td>Connection exception</td>
<td>Invalid attach type</td>
</tr>
<tr>
<td>20065</td>
<td>26000</td>
<td>Invalid SQL statement name</td>
<td>Invalid statement type</td>
</tr>
<tr>
<td>20066</td>
<td>33567</td>
<td>Invalid SQL descriptor name</td>
<td>Invalid sqlda</td>
</tr>
<tr>
<td>20067</td>
<td>08568</td>
<td>Connection exception</td>
<td>More than one database can't be attached locally</td>
</tr>
<tr>
<td>20068</td>
<td>42569</td>
<td>Syntax error</td>
<td>Bad arguments</td>
</tr>
<tr>
<td>20069</td>
<td>33570</td>
<td>Invalid SQL descriptor name</td>
<td>SQLDA size not enough</td>
</tr>
<tr>
<td>20070</td>
<td>33571</td>
<td>Invalid SQL descriptor name</td>
<td>SQLDA buffer length too high</td>
</tr>
<tr>
<td>20071</td>
<td>42572</td>
<td>Access rule violation</td>
<td>Specified operation not allowed on the view %s</td>
</tr>
<tr>
<td>20072</td>
<td>50573</td>
<td>c-treeACE SQL RDS error</td>
<td>Server is not allocated</td>
</tr>
<tr>
<td>20073</td>
<td>2a574</td>
<td>Access rule violation</td>
<td>View query specification for view too long</td>
</tr>
<tr>
<td>20074</td>
<td>2a575</td>
<td>Access rule violation</td>
<td>View column list must be specified as expressions are given</td>
</tr>
<tr>
<td>20075</td>
<td>21576</td>
<td>Cardinality violation</td>
<td>Number of columns in column list is less than in select list</td>
</tr>
<tr>
<td>20076</td>
<td>21577</td>
<td>Cardinality violation</td>
<td>Number of columns in column list is more than in select list</td>
</tr>
<tr>
<td>20077</td>
<td>42578</td>
<td>Access rule violation</td>
<td>Check option specified for non-insertable view</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>20078</td>
<td>42579</td>
<td>Access rule violation</td>
<td>Given SQL statement is not allowed on the view %s</td>
</tr>
<tr>
<td>20079</td>
<td>50580</td>
<td>c-treeACE SQL RDS error</td>
<td>More Tables cannot be created.</td>
</tr>
<tr>
<td>20080</td>
<td>44581</td>
<td>Check option violation</td>
<td>View check option violation</td>
</tr>
<tr>
<td>20081</td>
<td>22582</td>
<td>Data exception</td>
<td>No of expressions projected on either side of set-op don't match</td>
</tr>
<tr>
<td>20082</td>
<td>42582</td>
<td>Access rule violation</td>
<td>Column names not allowed in order by clause for this statement</td>
</tr>
<tr>
<td>20083</td>
<td>42583</td>
<td>Access rule violation</td>
<td>Outerjoin specified on a complex predicate</td>
</tr>
<tr>
<td>20084</td>
<td>42585</td>
<td>Access rule violation</td>
<td>Outerjoin specified on a sub_query</td>
</tr>
<tr>
<td>20085</td>
<td>42586</td>
<td>Access rule violation</td>
<td>Invalid Outerjoin specification</td>
</tr>
<tr>
<td>20086</td>
<td>42587</td>
<td>Access rule violation</td>
<td>Duplicate table constraint specification</td>
</tr>
<tr>
<td>20087</td>
<td>21588</td>
<td>Cardinality violation</td>
<td>Column count mismatch</td>
</tr>
<tr>
<td>20088</td>
<td>28589</td>
<td>Invalid auth specs</td>
<td>Invalid user name %s</td>
</tr>
<tr>
<td>20089</td>
<td>22590</td>
<td>Data exception</td>
<td>System date retrieval failed</td>
</tr>
<tr>
<td>20090</td>
<td>42591</td>
<td>Access rule violation</td>
<td>Table columnlist must be specified as expressions are given</td>
</tr>
<tr>
<td>20091</td>
<td>2a592</td>
<td>Access rule violation</td>
<td>Query statement too long.</td>
</tr>
<tr>
<td>20092</td>
<td>2d593</td>
<td>Invalid transaction</td>
<td>termination</td>
</tr>
<tr>
<td>20093</td>
<td>22594</td>
<td>Data exception</td>
<td>Synonym already exists</td>
</tr>
<tr>
<td>20094</td>
<td>hz595</td>
<td>Remote database access</td>
<td>Database link %s with same name already exists</td>
</tr>
<tr>
<td>20095</td>
<td>hz596</td>
<td>Remote database access</td>
<td>Database link %s not found</td>
</tr>
<tr>
<td>20096</td>
<td>08597</td>
<td>Connection exception</td>
<td>Connect String not specified/incorrect</td>
</tr>
<tr>
<td>20097</td>
<td>hz598</td>
<td>Remote database access</td>
<td>Specified operation not allowed on a remote table</td>
</tr>
<tr>
<td>20098</td>
<td>22599</td>
<td>Data exception</td>
<td>More than one row selected by the query</td>
</tr>
<tr>
<td>20099</td>
<td>24000</td>
<td>Invalid cursor state</td>
<td>Cursor not positioned on a valid row</td>
</tr>
<tr>
<td>20100</td>
<td>4250a</td>
<td>Access rule violation</td>
<td>Subquery not allowed here</td>
</tr>
<tr>
<td>20101</td>
<td>2350b</td>
<td>Integrity constraint</td>
<td>No references for the table %s</td>
</tr>
<tr>
<td>20102</td>
<td>2350c</td>
<td>Integrity constraint</td>
<td>Primary key column defined null</td>
</tr>
<tr>
<td>20103</td>
<td>2350d</td>
<td>Integrity constraint</td>
<td>No matching key defined for the referenced table</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>20104</td>
<td>2350e</td>
<td>Integrity constraint</td>
<td>Keys in reference constraint incompatible</td>
</tr>
<tr>
<td>20105</td>
<td>5050f</td>
<td>c-treeACE SQL RDS error</td>
<td>Statement not allowed in readonly isolation level</td>
</tr>
<tr>
<td>20106</td>
<td>2150g</td>
<td>Cardinality violation</td>
<td>Invalid ROWID</td>
</tr>
<tr>
<td>20107</td>
<td>hz50h</td>
<td>Remote database access</td>
<td>Remote Database not started</td>
</tr>
<tr>
<td>20108</td>
<td>0850i</td>
<td>Connection exception</td>
<td>Remote Network Server not started.</td>
</tr>
<tr>
<td>20109</td>
<td>hz50j</td>
<td>Remote database access</td>
<td>Remote Database Name not valid</td>
</tr>
<tr>
<td>20110</td>
<td>0850k</td>
<td>Connection exception</td>
<td>TCP/IP Remote HostName is unknown.</td>
</tr>
<tr>
<td>20114</td>
<td>33002</td>
<td>Invalid SQL descriptor name</td>
<td>Fetched Value NULL &amp; indicator var not defined</td>
</tr>
<tr>
<td>20115</td>
<td>5050l</td>
<td>c-treeACE SQL RDS error</td>
<td>References to the table/record present</td>
</tr>
<tr>
<td>20116</td>
<td>2350m</td>
<td>Integrity constraint</td>
<td>Constraint violation</td>
</tr>
<tr>
<td>20117</td>
<td>2350n</td>
<td>Integrity constraint</td>
<td>Table definition not complete</td>
</tr>
<tr>
<td>20118</td>
<td>4250o</td>
<td>Access rule violation</td>
<td>Duplicate constraint name %s</td>
</tr>
<tr>
<td>20119</td>
<td>2350p</td>
<td>Integrity constraint</td>
<td>Constraint name not found</td>
</tr>
<tr>
<td>20120</td>
<td>22000</td>
<td>Data exception</td>
<td>**use of reserved word %s</td>
</tr>
<tr>
<td>20121</td>
<td>5050q</td>
<td>c-treeACE SQL RDS error</td>
<td>Permission denied</td>
</tr>
<tr>
<td>20122</td>
<td>5050r</td>
<td>c-treeACE SQL RDS error</td>
<td>Procedure %s not found</td>
</tr>
<tr>
<td>20123</td>
<td>5050s</td>
<td>c-treeACE SQL RDS error</td>
<td>Invalid arguments to procedure</td>
</tr>
<tr>
<td>20124</td>
<td>5050t</td>
<td>c-treeACE SQL RDS error</td>
<td>Query conditionally terminated</td>
</tr>
<tr>
<td>20125</td>
<td>0750u</td>
<td>Dynamic SQL-error</td>
<td>Number of open cursors exceeds limit</td>
</tr>
<tr>
<td>20126</td>
<td>34000</td>
<td>Invalid cursor name</td>
<td>***Invalid cursor name</td>
</tr>
<tr>
<td>20127</td>
<td>07001</td>
<td>Dynamic SQL-error</td>
<td>Bad parameter specification for the statement</td>
</tr>
<tr>
<td>20128</td>
<td>2250x</td>
<td>Data Exception</td>
<td>Numeric value out of range.</td>
</tr>
<tr>
<td>20129</td>
<td>2250y</td>
<td>Data Exception</td>
<td>Data truncated.</td>
</tr>
<tr>
<td>20130</td>
<td>40000</td>
<td>c-treeACE SQL RDS error</td>
<td>Rolled back transaction.</td>
</tr>
<tr>
<td>20131</td>
<td>50522</td>
<td>c-treeACE SQL RDS error</td>
<td>Value for Parameter marker is Missing</td>
</tr>
<tr>
<td>20132</td>
<td>5050u</td>
<td>c-treeACE SQL RDS error</td>
<td>Revoke failed because of restrict</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>20133</td>
<td>0a000</td>
<td>Feature not supported</td>
<td>Sorry, feature not supported in this Edition.</td>
</tr>
<tr>
<td>20134</td>
<td>5050v</td>
<td>c-treeACE SQL RDS error</td>
<td>Invalid long datatype column references</td>
</tr>
<tr>
<td>20135</td>
<td>5050x</td>
<td>c-treeACE SQL RDS error</td>
<td>Contains operator is not supported in this context</td>
</tr>
<tr>
<td>20136</td>
<td>5050z</td>
<td>c-treeACE SQL RDS error</td>
<td>Contains operator is not supported for this datatype</td>
</tr>
<tr>
<td>20137</td>
<td>50514</td>
<td>c-treeACE SQL RDS error</td>
<td>Index is not defined or does not support CONTAINS</td>
</tr>
<tr>
<td>20138</td>
<td>50513</td>
<td>c-treeACE SQL RDS error</td>
<td>Index on long fields requires that it can push down only CONTAINS</td>
</tr>
<tr>
<td>20139</td>
<td>50540</td>
<td>c-treeACE SQL RDS error</td>
<td>Index on long fields not supported</td>
</tr>
<tr>
<td>20140</td>
<td>50512</td>
<td>c-treeACE SQL RDS error</td>
<td>Procedure %s already exists</td>
</tr>
<tr>
<td>20141</td>
<td>85001</td>
<td>c-treeACE SQL Stored procedure/function Compilation</td>
<td>Error in Stored Procedure/function Compilation.</td>
</tr>
<tr>
<td>20142</td>
<td>86001</td>
<td>c-treeACE SQL Stored procedure/function Execution</td>
<td>Error in Stored Procedure/function Execution - %s.</td>
</tr>
<tr>
<td>20143</td>
<td>86002</td>
<td>c-treeACE SQL Stored procedure Execution</td>
<td>Too many recursions in call procedure.</td>
</tr>
<tr>
<td>20144</td>
<td>86003</td>
<td>c-treeACE SQL Stored procedure Execution</td>
<td>Null value fetched.</td>
</tr>
<tr>
<td>20145</td>
<td>86004</td>
<td>c-treeACE SQL Stored procedure Execution</td>
<td>Invalid field reference.</td>
</tr>
<tr>
<td>20146</td>
<td>86005</td>
<td>c-treeACE SQL Triggers</td>
<td>Trigger with this name %s already exists.</td>
</tr>
<tr>
<td>20147</td>
<td>86006</td>
<td>c-treeACE SQL Triggers</td>
<td>Trigger with this name %s does not exist.</td>
</tr>
<tr>
<td>20148</td>
<td>86007</td>
<td>c-treeACE SQL Triggers</td>
<td>Trigger Execution Failed.</td>
</tr>
<tr>
<td>20150</td>
<td>50551</td>
<td>c-treeACE SQL RDS error</td>
<td>View manager ID is not found.</td>
</tr>
<tr>
<td>20151</td>
<td>50515</td>
<td>c-treeACE SQL RDS error</td>
<td>Cannot drop all columns; use DROP TABLE instead</td>
</tr>
<tr>
<td>20152</td>
<td>50516</td>
<td>c-treeACE SQL RDS error</td>
<td>Cannot preallocate memory for n cache items</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>20153</td>
<td>50517</td>
<td>c-treeACE SQL RDS error</td>
<td>Tree not present in cache ; search failed</td>
</tr>
<tr>
<td>20154</td>
<td>50518</td>
<td>c-treeACE SQL RDS error</td>
<td>Statement cache insert failed</td>
</tr>
<tr>
<td>20155</td>
<td>50519</td>
<td>c-treeACE SQL RDS error</td>
<td>Environment variable used for the creation of multi component index is not set properly.</td>
</tr>
<tr>
<td>20156</td>
<td>50521</td>
<td>c-treeACE SQL RDS error</td>
<td>No SQL statement</td>
</tr>
<tr>
<td>20157</td>
<td>50523</td>
<td>c-treeACE SQL RDS error</td>
<td>Invalid Java SP/T method sequence.</td>
</tr>
<tr>
<td>20158</td>
<td>50524</td>
<td>c-treeACE SQL RDS error</td>
<td>Can't create triggers on system tables.</td>
</tr>
<tr>
<td>20159</td>
<td>28590</td>
<td>Invalid auth specs</td>
<td>Can't REVOKE privileges, you did not grant.</td>
</tr>
<tr>
<td>20160</td>
<td>28591</td>
<td>Feature not supported</td>
<td>%s SP/T Feature not supported.</td>
</tr>
<tr>
<td>20161</td>
<td>28592</td>
<td>c-treeACE SQL Stored procedure Execution</td>
<td>Not all OUT/INOUT parameters are registered.</td>
</tr>
<tr>
<td>20162</td>
<td>28593</td>
<td>c-treeACE SQL Stored procedure Execution</td>
<td>IN parameter Can't be registered.</td>
</tr>
<tr>
<td>20163</td>
<td>28594</td>
<td>c-treeACE SQL Stored procedure Execution</td>
<td>'Scale' for registerOutParam not implemented.</td>
</tr>
<tr>
<td>20164</td>
<td>2a565</td>
<td>Syntax error</td>
<td>Escape character must be character string of length 1</td>
</tr>
<tr>
<td>20165</td>
<td>2a566</td>
<td>Syntax error</td>
<td>Missing or illegal character following the escape character</td>
</tr>
<tr>
<td>20166</td>
<td>50547</td>
<td>c-treeACE SQL RDS error</td>
<td>Column nullability and Default value do not match</td>
</tr>
<tr>
<td>20169</td>
<td>2a568</td>
<td>Query Execution Plan</td>
<td>Invalid plan table definition</td>
</tr>
<tr>
<td>20170</td>
<td>2a569</td>
<td>Query Execution Plan</td>
<td>No RESOURCE privileges to create plan table</td>
</tr>
<tr>
<td>20171</td>
<td>2a570</td>
<td>Query Execution Plan</td>
<td>No DBA access rights to create plan table</td>
</tr>
<tr>
<td>20172</td>
<td>2a571</td>
<td>Query Execution Plan</td>
<td>Plan table access denied</td>
</tr>
<tr>
<td>20173</td>
<td>2a572</td>
<td>Query Execution Plan</td>
<td>Statement ID not specified</td>
</tr>
<tr>
<td>20174</td>
<td>24543</td>
<td>c-treeACE SQL RDS error</td>
<td>Batch Update has resulted in resultset.</td>
</tr>
<tr>
<td>20175</td>
<td>42588</td>
<td>Access rule violation</td>
<td>duplicate or conflicting NULL and/or NOT NULL specifications</td>
</tr>
<tr>
<td>20176</td>
<td>2a573</td>
<td>Query Execution Plan</td>
<td>Duplicate Statement ID specified</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>20177</td>
<td>42589</td>
<td>Access rule violation</td>
<td>Cannot define PRIMARY KEY constraint on nullable column</td>
</tr>
<tr>
<td>20178</td>
<td>0A500</td>
<td>Feature not supported</td>
<td>PERCENT and WITH TIES options not supported in TOP clause</td>
</tr>
<tr>
<td>20179</td>
<td>0A501</td>
<td>Data exception</td>
<td>View %s not found</td>
</tr>
<tr>
<td>20181</td>
<td>0A503</td>
<td>Data exception</td>
<td>Synonym %s not found</td>
</tr>
<tr>
<td>20182</td>
<td>50525</td>
<td>c-treeACE SQL RDS error</td>
<td>Function already exists</td>
</tr>
<tr>
<td>20183</td>
<td>50526</td>
<td>c-treeACE SQL RDS error</td>
<td>Function not found</td>
</tr>
<tr>
<td>20184</td>
<td>22543</td>
<td>Data exception</td>
<td>Table %s not found</td>
</tr>
<tr>
<td>20185</td>
<td>22544</td>
<td>Data exception</td>
<td>Literal length exceeds maximum limit</td>
</tr>
<tr>
<td>20186</td>
<td>22585</td>
<td>Data Exception</td>
<td>Character column expected for LIKE predicate</td>
</tr>
<tr>
<td>20187</td>
<td>08006</td>
<td>Connection exception</td>
<td>JVM Initialization failed</td>
</tr>
<tr>
<td>20188</td>
<td>22543</td>
<td>Data exception</td>
<td>Multiple IDENTITY/DEFAULT clause specified or DEFAULT specified for IDENTITY column</td>
</tr>
<tr>
<td>20189</td>
<td>22543</td>
<td>Data exception</td>
<td>Insert/Update of an IDENTITY column not allowed</td>
</tr>
<tr>
<td>20190</td>
<td>22543</td>
<td>Data exception</td>
<td>Invalid IDENTITY Seed/Increment value specified</td>
</tr>
<tr>
<td>20191</td>
<td>22543</td>
<td>Data exception</td>
<td>Identity allowed only on columns of type Integer, Smallint, Tinyint and Bigint</td>
</tr>
<tr>
<td>20192</td>
<td>22543</td>
<td>Data exception</td>
<td>IDENTITY cannot be specified if the table has only one column</td>
</tr>
<tr>
<td>20193</td>
<td>50530</td>
<td>c-treeACE SQL RDS error</td>
<td>Ambiguous Index reference %s, specify table name</td>
</tr>
<tr>
<td>20194</td>
<td>21588</td>
<td>Syntax error</td>
<td>Search condition required for recursive query</td>
</tr>
<tr>
<td>20195</td>
<td>22566</td>
<td>Invalid auth specs</td>
<td>User/Group %s already exists</td>
</tr>
<tr>
<td>20196</td>
<td>22567</td>
<td>Invalid auth specs</td>
<td>User/Group %s not found</td>
</tr>
<tr>
<td>20197</td>
<td>22568</td>
<td>Invalid auth specs</td>
<td>User %s already exists in User Group %s</td>
</tr>
<tr>
<td>20198</td>
<td>22569</td>
<td>Invalid auth specs</td>
<td>User Group %s not found</td>
</tr>
<tr>
<td>20199</td>
<td>22570</td>
<td>Invalid auth specs</td>
<td>User %s not found in User Group %s</td>
</tr>
<tr>
<td>20200</td>
<td>22571</td>
<td>Invalid auth specs</td>
<td>Cannot add a User Group to another User Group</td>
</tr>
<tr>
<td>20201</td>
<td>21589</td>
<td>SQL Stored procedure</td>
<td>Procedure %s has OUT/INOUT parameters</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>--------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>20202</td>
<td>21590</td>
<td>SQL Stored procedure</td>
<td>Procedure %s has no resultset</td>
</tr>
<tr>
<td>20203</td>
<td>21591</td>
<td>SQL Stored procedure</td>
<td>Scrollable cursor not supported on call statement</td>
</tr>
<tr>
<td>20211</td>
<td>22800</td>
<td>Data exception</td>
<td>Remote procedure call error</td>
</tr>
<tr>
<td>20212</td>
<td>08801</td>
<td>Connection exception</td>
<td>SQL client bind to daemon failed</td>
</tr>
<tr>
<td>20213</td>
<td>08802</td>
<td>Connection exception</td>
<td>SQL client bind to SQL server failed</td>
</tr>
<tr>
<td>20214</td>
<td>08803</td>
<td>Connection exception</td>
<td>SQL NETWORK service entry is not available</td>
</tr>
<tr>
<td>20215</td>
<td>08804</td>
<td>Connection exception</td>
<td>Invalid TCP/IP hostname</td>
</tr>
<tr>
<td>20216</td>
<td>08805</td>
<td>Remote database access</td>
<td>Invalid remote database name</td>
</tr>
<tr>
<td>20217</td>
<td>08806</td>
<td>Connection exception</td>
<td>Network error on server</td>
</tr>
<tr>
<td>20218</td>
<td>08807</td>
<td>Connection exception</td>
<td>Invalid protocol</td>
</tr>
<tr>
<td>20219</td>
<td>2e000</td>
<td>Invalid connection name</td>
<td>***Invalid connection name</td>
</tr>
<tr>
<td>20220</td>
<td>08809</td>
<td>Connection exception</td>
<td>Duplicate connection name</td>
</tr>
<tr>
<td>20221</td>
<td>08810</td>
<td>Connection exception</td>
<td>No active connection</td>
</tr>
<tr>
<td>20222</td>
<td>08811</td>
<td>Connection exception</td>
<td>No environment defined database</td>
</tr>
<tr>
<td>20223</td>
<td>08812</td>
<td>Connection exception</td>
<td>Multiple local connections</td>
</tr>
<tr>
<td>20224</td>
<td>08813</td>
<td>Connection exception</td>
<td>Invalid protocol in connect_string</td>
</tr>
<tr>
<td>20225</td>
<td>08814</td>
<td>Connection exception</td>
<td>Exceeding permissible number of connections</td>
</tr>
<tr>
<td>20226</td>
<td>80815</td>
<td>c-treeACE SQL SNW errors</td>
<td>Bad database handle</td>
</tr>
<tr>
<td>20227</td>
<td>08816</td>
<td>Connection exception</td>
<td>Invalid host name in connect string</td>
</tr>
<tr>
<td>20228</td>
<td>28817</td>
<td>Invalid auth specs</td>
<td>Access denied (Authorization failed)</td>
</tr>
<tr>
<td>20229</td>
<td>22818</td>
<td>Data exception</td>
<td>Invalid date value</td>
</tr>
<tr>
<td>20230</td>
<td>22819</td>
<td>Data exception</td>
<td>Invalid date string</td>
</tr>
<tr>
<td>20231</td>
<td>22820</td>
<td>Data exception</td>
<td>Invalid number strings</td>
</tr>
<tr>
<td>20232</td>
<td>22821</td>
<td>Data exception</td>
<td>Invalid number string</td>
</tr>
<tr>
<td>20233</td>
<td>22822</td>
<td>Data exception</td>
<td>Invalid time value</td>
</tr>
<tr>
<td>20234</td>
<td>22523</td>
<td>Data exception</td>
<td>Invalid time string</td>
</tr>
<tr>
<td>20235</td>
<td>22007</td>
<td>Data exception</td>
<td>Invalid time stamp string</td>
</tr>
<tr>
<td>20236</td>
<td>22012</td>
<td>Data exception</td>
<td>Division by zero attempted</td>
</tr>
<tr>
<td>20238</td>
<td>22615</td>
<td>Data exception</td>
<td>Error in format type</td>
</tr>
<tr>
<td>20239</td>
<td>2c000</td>
<td>Invalid character set name</td>
<td>Invalid character set name specified.</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>20240</td>
<td>5050y</td>
<td>c-treeACE SQL RDS errors</td>
<td>Invalid collation name specified.</td>
</tr>
<tr>
<td>20241</td>
<td>08815</td>
<td>Connection Exception</td>
<td>Service in use.</td>
</tr>
<tr>
<td>20242</td>
<td>22008</td>
<td>Data exception</td>
<td>Invalid timestamp</td>
</tr>
<tr>
<td>20243</td>
<td>2c001</td>
<td>Illegal Buffer Size</td>
<td>Illegal Buffer Size specified.</td>
</tr>
<tr>
<td>20244</td>
<td>2c002</td>
<td>Invalid Coercion</td>
<td>Invalid Coercion Value Specified.</td>
</tr>
<tr>
<td>20245</td>
<td>2c003</td>
<td>Missing Dboption</td>
<td>Value for specified dboption not found.</td>
</tr>
<tr>
<td>20246</td>
<td>2c004</td>
<td>Bad Dboption</td>
<td>Illegal Value specified for a dboption.</td>
</tr>
<tr>
<td>20247</td>
<td>2c005</td>
<td>Invalid characters in String</td>
<td>String contains Invalid characters.</td>
</tr>
<tr>
<td>20248</td>
<td>22601</td>
<td>Data Exception</td>
<td>Identifier too long.</td>
</tr>
<tr>
<td>20249</td>
<td>22602</td>
<td>Data Exception</td>
<td>Identifier incompatible character set.</td>
</tr>
<tr>
<td>20250</td>
<td>22603</td>
<td>Data Exception</td>
<td>Invalid identifier.</td>
</tr>
<tr>
<td>20251</td>
<td>22604</td>
<td>Data Exception</td>
<td>Quoted string not properly terminated.</td>
</tr>
<tr>
<td>20252</td>
<td>22605</td>
<td>Data Exception</td>
<td>Missing double quote in identifier.</td>
</tr>
<tr>
<td>20253</td>
<td>22606</td>
<td>Invalid length</td>
<td>Storage Attribute too long.</td>
</tr>
<tr>
<td>20254</td>
<td>21000</td>
<td>Invalid Handle</td>
<td>Invalid Environment Handle</td>
</tr>
<tr>
<td>20255</td>
<td>08818</td>
<td>Connection Exception</td>
<td>Destination IP address validation failed; the destination IP address specified by the client is not authorized to access the server.</td>
</tr>
<tr>
<td>20256</td>
<td>08817</td>
<td>Connection Exception</td>
<td>Source IP address validation failed; client machine is not authorized to access the server.</td>
</tr>
<tr>
<td>20257</td>
<td>08557</td>
<td>Connection exception</td>
<td>Length of the database name exceeds the maximum limit</td>
</tr>
<tr>
<td>20258</td>
<td>5051a</td>
<td>c-treeACE SQL RDS error</td>
<td>Invalid lob datatype column references.</td>
</tr>
<tr>
<td>20259</td>
<td>50912</td>
<td>c-treeACE SQL RDS error</td>
<td>Index on lob types requires CONTAINS operator.</td>
</tr>
<tr>
<td>20260</td>
<td>42523</td>
<td>Access rule violation</td>
<td>Non-group-by expression in order by clause</td>
</tr>
<tr>
<td>20261</td>
<td>42528</td>
<td>Access rule violation</td>
<td>Order by field not found in SELECT DISTINCT list</td>
</tr>
<tr>
<td>20262</td>
<td>42528</td>
<td>Access rule violation</td>
<td>Order by in subquery without top or skip</td>
</tr>
<tr>
<td>20263</td>
<td>08819</td>
<td>Connection exception</td>
<td>Client Server mismatch; both must be either Non-Commercial Trial or Pro</td>
</tr>
<tr>
<td>20264</td>
<td>2350p</td>
<td>Integrity constraint</td>
<td>Invalid constraint info from storage system</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>20300</td>
<td>90901</td>
<td>c-treeACE SQL/DBS</td>
<td>Column group column doesn't exist</td>
</tr>
<tr>
<td>20301</td>
<td>90902</td>
<td>c-treeACE SQL/DBS</td>
<td>Column group column already specified</td>
</tr>
<tr>
<td>20302</td>
<td>90903</td>
<td>c-treeACE SQL/DBS</td>
<td>Column group name already specified</td>
</tr>
<tr>
<td>20303</td>
<td>90904</td>
<td>c-treeACE SQL/DBS</td>
<td>Column groups haven't covered all columns</td>
</tr>
<tr>
<td>20304</td>
<td>90905</td>
<td>c-treeACE SQL/DBS</td>
<td>Column groups are not implemented in c-treeACE SQL storage</td>
</tr>
</tbody>
</table>
## 4.9 21000 Series - c-treeDB FSS DB Layer

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>21001</td>
<td>4001</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Not enough memory</td>
</tr>
<tr>
<td>21002</td>
<td>4002</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Handle is NULL</td>
</tr>
<tr>
<td>21003</td>
<td>4003</td>
<td>FSS dblayer error</td>
<td>c-treeDB - No session handle</td>
</tr>
<tr>
<td>21004</td>
<td>4004</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid argument</td>
</tr>
<tr>
<td>21005</td>
<td>4005</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Index out of range</td>
</tr>
<tr>
<td>21006</td>
<td>4006</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Argument is too small</td>
</tr>
<tr>
<td>21007</td>
<td>4007</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Null argument not valid</td>
</tr>
<tr>
<td>21008</td>
<td>4008</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Null argument not valid</td>
</tr>
<tr>
<td>21009</td>
<td>4009</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Table is not open</td>
</tr>
<tr>
<td>21010</td>
<td>4010</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Not a database handle</td>
</tr>
<tr>
<td>21011</td>
<td>4011</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Handle is active</td>
</tr>
<tr>
<td>21012</td>
<td>4012</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Handle is not active</td>
</tr>
<tr>
<td>21013</td>
<td>4013</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Handle is not a table</td>
</tr>
<tr>
<td>21014</td>
<td>4014</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Unknown field name</td>
</tr>
<tr>
<td>21015</td>
<td>4015</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Can't perform delete</td>
</tr>
<tr>
<td>21016</td>
<td>4016</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Field already exists</td>
</tr>
<tr>
<td>21017</td>
<td>4017</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Not a field handle</td>
</tr>
<tr>
<td>21018</td>
<td>4018</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Not an index handle</td>
</tr>
<tr>
<td>21019</td>
<td>4019</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid field/key type</td>
</tr>
<tr>
<td>21020</td>
<td>4020</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Not a segment handle</td>
</tr>
<tr>
<td>21021</td>
<td>4021</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Database already exists</td>
</tr>
<tr>
<td>21022</td>
<td>4022</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Table already exists</td>
</tr>
<tr>
<td>21023</td>
<td>4023</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Table does not exist</td>
</tr>
<tr>
<td>21024</td>
<td>4024</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Not a record handle</td>
</tr>
<tr>
<td>21025</td>
<td>4025</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Internal error</td>
</tr>
<tr>
<td>21026</td>
<td>4026</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid find mode</td>
</tr>
<tr>
<td>21027</td>
<td>4027</td>
<td>FSS dblayer error</td>
<td>c-treeDB - No data in record</td>
</tr>
<tr>
<td>21028</td>
<td>4028</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid date or time format</td>
</tr>
<tr>
<td>21029</td>
<td>4029</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid date</td>
</tr>
<tr>
<td>21030</td>
<td>4030</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid date day</td>
</tr>
<tr>
<td>21031</td>
<td>4031</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid date month</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>21032</td>
<td>4032</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid date year</td>
</tr>
<tr>
<td>21033</td>
<td>4033</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid packed CTIME</td>
</tr>
<tr>
<td>21034</td>
<td>4034</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid hour</td>
</tr>
<tr>
<td>21035</td>
<td>4035</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid minute</td>
</tr>
<tr>
<td>21036</td>
<td>4036</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid second</td>
</tr>
<tr>
<td>21037</td>
<td>4037</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid time AM or PM value</td>
</tr>
<tr>
<td>21038</td>
<td>4038</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Operation causes Overflow</td>
</tr>
<tr>
<td>21039</td>
<td>4039</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Operation causes underflow</td>
</tr>
<tr>
<td>21040</td>
<td>4040</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Division by zero error</td>
</tr>
<tr>
<td>21041</td>
<td>4041</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid timestamp</td>
</tr>
<tr>
<td>21042</td>
<td>4042</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Can't perform type conversion</td>
</tr>
<tr>
<td>21043</td>
<td>4043</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Argument is too big</td>
</tr>
<tr>
<td>21044</td>
<td>4044</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Path does not exist</td>
</tr>
<tr>
<td>21045</td>
<td>4045</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Unknown index number</td>
</tr>
<tr>
<td>21046</td>
<td>4046</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Not found</td>
</tr>
<tr>
<td>21047</td>
<td>4047</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid segment mode</td>
</tr>
<tr>
<td>21048</td>
<td>4048</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Table has no indices</td>
</tr>
<tr>
<td>21049</td>
<td>4049</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid segment number</td>
</tr>
<tr>
<td>21050</td>
<td>4050</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid ISAM context handle</td>
</tr>
<tr>
<td>21051</td>
<td>4051</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Index name already in use</td>
</tr>
<tr>
<td>21052</td>
<td>4052</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Partial field data read</td>
</tr>
<tr>
<td>21053</td>
<td>4053</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid index name</td>
</tr>
<tr>
<td>21054</td>
<td>4054</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Feature not supported</td>
</tr>
<tr>
<td>21055</td>
<td>4055</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid lock mode</td>
</tr>
<tr>
<td>21056</td>
<td>4056</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Record not locked</td>
</tr>
<tr>
<td>21057</td>
<td>4057</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Record is not locked for writes</td>
</tr>
<tr>
<td>21058</td>
<td>4058</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Dictionary handle not allocated</td>
</tr>
<tr>
<td>21059</td>
<td>4059</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Not yet implemented</td>
</tr>
<tr>
<td>21060</td>
<td>4060</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid number</td>
</tr>
<tr>
<td>21061</td>
<td>4061</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid number precision</td>
</tr>
<tr>
<td>21062</td>
<td>4062</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid number scale</td>
</tr>
<tr>
<td>21063</td>
<td>4063</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Record buffer not large enough</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>21064</td>
<td>4064</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Recbyt index can't be deleted</td>
</tr>
<tr>
<td>21065</td>
<td>4065</td>
<td>FSS dblayer error</td>
<td>c-treeDB - rowid index can't be deleted</td>
</tr>
<tr>
<td>21066</td>
<td>4066</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Table has no rowid index</td>
</tr>
<tr>
<td>21067</td>
<td>4067</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Cannot create a file (dupdb)</td>
</tr>
<tr>
<td>21068</td>
<td>4068</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Cannot copy a file (dupdb)</td>
</tr>
<tr>
<td>21069</td>
<td>4069</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Can't restore lock</td>
</tr>
<tr>
<td>21070</td>
<td>4070</td>
<td>FSS dblayer error</td>
<td>c-treeDB - invalid isolation level</td>
</tr>
<tr>
<td>21071</td>
<td>4071</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Cond. expr evaluates to false</td>
</tr>
<tr>
<td>21072</td>
<td>4072</td>
<td>FSS dblayer error</td>
<td>c-treeDB - conditional expr. parser error</td>
</tr>
<tr>
<td>21073</td>
<td>4073</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid type mix-up</td>
</tr>
<tr>
<td>21074</td>
<td>4074</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Unknown field name</td>
</tr>
<tr>
<td>21075</td>
<td>4075</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Internal yacc error</td>
</tr>
<tr>
<td>21076</td>
<td>4076</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Memory allocation failed</td>
</tr>
<tr>
<td>21077</td>
<td>4077</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Stack overflow</td>
</tr>
<tr>
<td>21078</td>
<td>4078</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Stack Underflow</td>
</tr>
<tr>
<td>21079</td>
<td>4079</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid execution node</td>
</tr>
<tr>
<td>21080</td>
<td>4080</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Division by zero</td>
</tr>
<tr>
<td>21081</td>
<td>4081</td>
<td>FSS dblayer error</td>
<td>c-treeDB - No record schema</td>
</tr>
<tr>
<td>21082</td>
<td>4082</td>
<td>FSS dblayer error</td>
<td>c-treeDB - No record buffer</td>
</tr>
<tr>
<td>21083</td>
<td>4083</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Not enough data</td>
</tr>
<tr>
<td>21084</td>
<td>4084</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid session type</td>
</tr>
<tr>
<td>21085</td>
<td>4085</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid alter table action</td>
</tr>
<tr>
<td>21086</td>
<td>4086</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Records are different</td>
</tr>
<tr>
<td>21087</td>
<td>4087</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid operator</td>
</tr>
<tr>
<td>21088</td>
<td>4088</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Table was open read only</td>
</tr>
<tr>
<td>21089</td>
<td>4089</td>
<td>FSS dblayer error</td>
<td>c-treeDB - CALLBACK not implemented</td>
</tr>
<tr>
<td>21090</td>
<td>4090</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid handle type</td>
</tr>
<tr>
<td>21091</td>
<td>4091</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid callback type</td>
</tr>
<tr>
<td>21092</td>
<td>4092</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Can't move segment</td>
</tr>
<tr>
<td>21093</td>
<td>4093</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Index name duplicated in database</td>
</tr>
<tr>
<td>21094</td>
<td>4094</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Database does not exist or not found</td>
</tr>
<tr>
<td>21095</td>
<td>4095</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Index has no segments</td>
</tr>
<tr>
<td>21096</td>
<td>4096</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Batch operation not active</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>21097</td>
<td>4097</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Batch operation is already active</td>
</tr>
<tr>
<td>21098</td>
<td>4098</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid batch mode</td>
</tr>
<tr>
<td>21099</td>
<td>4099</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Not a resource handle</td>
</tr>
<tr>
<td>21100</td>
<td>4100</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid session attach mode</td>
</tr>
<tr>
<td>21101</td>
<td>4101</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Session not attached</td>
</tr>
<tr>
<td>21102</td>
<td>4102</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid binary field length</td>
</tr>
<tr>
<td>21103</td>
<td>4103</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Router table error</td>
</tr>
<tr>
<td>21104</td>
<td>4104</td>
<td>FSS dblayer error</td>
<td>c-treeDB - invalid router table name</td>
</tr>
<tr>
<td>21105</td>
<td>4105</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid number of fields in router</td>
</tr>
<tr>
<td>21106</td>
<td>4106</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid router field mapping</td>
</tr>
<tr>
<td>21107</td>
<td>4107</td>
<td>FSS dblayer error</td>
<td>c-treeDB - The database is not SQL</td>
</tr>
<tr>
<td>21108</td>
<td>4108</td>
<td>FSS dblayer error</td>
<td>c-treeDB - The record buffer contains data causing field offset calculation going after the record size</td>
</tr>
<tr>
<td>21109</td>
<td>4109</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 1</td>
</tr>
<tr>
<td>21110</td>
<td>4110</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 2</td>
</tr>
<tr>
<td>21111</td>
<td>4111</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 3</td>
</tr>
<tr>
<td>21112</td>
<td>4112</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 4</td>
</tr>
<tr>
<td>21113</td>
<td>4113</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 5</td>
</tr>
<tr>
<td>21114</td>
<td>4114</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 6</td>
</tr>
<tr>
<td>21115</td>
<td>4115</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 7</td>
</tr>
<tr>
<td>21116</td>
<td>4116</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 8</td>
</tr>
<tr>
<td>21117</td>
<td>4117</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 9</td>
</tr>
<tr>
<td>21118</td>
<td>4118</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 10</td>
</tr>
<tr>
<td>21119</td>
<td>4119</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 11</td>
</tr>
<tr>
<td>21120</td>
<td>4120</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 12</td>
</tr>
<tr>
<td>21121</td>
<td>4121</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 13</td>
</tr>
<tr>
<td>21122</td>
<td>4122</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 14</td>
</tr>
<tr>
<td>21123</td>
<td>4123</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 15</td>
</tr>
<tr>
<td>21124</td>
<td>4124</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 16</td>
</tr>
<tr>
<td>21125</td>
<td>4125</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 17</td>
</tr>
<tr>
<td>21126</td>
<td>4126</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 18</td>
</tr>
<tr>
<td>21127</td>
<td>4127</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 19</td>
</tr>
<tr>
<td>21128</td>
<td>4128</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Callback error 20</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>21129</td>
<td>4129</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Cannot properly check or update dictionary UID</td>
</tr>
<tr>
<td>21130</td>
<td>4130</td>
<td>FSS dblayer error</td>
<td>c-treeDB - No more Resource for Vtable info available</td>
</tr>
<tr>
<td>21131</td>
<td>4131</td>
<td>FSS dblayer error</td>
<td>c-treeDB - The table has Vtable defined in the dictionary</td>
</tr>
<tr>
<td>21132</td>
<td>4132</td>
<td>FSS dblayer error</td>
<td>c-treeDB - The VTable type in the dictionary mismatches the one in the resource</td>
</tr>
<tr>
<td>21133</td>
<td>4133</td>
<td>FSS dblayer error</td>
<td>c-treeDB - An Identity field has been already defined</td>
</tr>
<tr>
<td>21134</td>
<td>4134</td>
<td>FSS dblayer error</td>
<td>c-treeDB - An Identity field has not been defined</td>
</tr>
<tr>
<td>21135</td>
<td>4135</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Field is null</td>
</tr>
<tr>
<td>21136</td>
<td>4136</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Invalid use of reserved name</td>
</tr>
<tr>
<td>21137</td>
<td>4137</td>
<td>FSS dblayer error</td>
<td>Feature not supported in Dictionary</td>
</tr>
<tr>
<td>21138</td>
<td>4138</td>
<td>FSS dblayer error</td>
<td>Invalid file name</td>
</tr>
<tr>
<td>21139</td>
<td>4139</td>
<td>FSS dblayer error</td>
<td>Operation not supported on Virtual field</td>
</tr>
<tr>
<td>21140</td>
<td>4140</td>
<td>FSS dblayer error</td>
<td>Operation not supported on tables with Virtual fields</td>
</tr>
<tr>
<td>21141</td>
<td>4141</td>
<td>FSS dblayer error</td>
<td>Cannot add VFields without setting vclear function</td>
</tr>
<tr>
<td>21142</td>
<td>4142</td>
<td>FSS dblayer error</td>
<td>Attempting a getfield operation on VField without having a get function</td>
</tr>
<tr>
<td>21143</td>
<td>4143</td>
<td>FSS dblayer error</td>
<td>Attempting a setfield operation on VField without having a set function</td>
</tr>
<tr>
<td>21144</td>
<td>4144</td>
<td>FSS dblayer error</td>
<td>CTNUMBER has invalid length</td>
</tr>
<tr>
<td>21145</td>
<td>4145</td>
<td>FSS dblayer error</td>
<td>Invalid millisecond</td>
</tr>
<tr>
<td>21146</td>
<td>4146</td>
<td>FSS dblayer error</td>
<td>Unable to compute legacy segment offset.</td>
</tr>
<tr>
<td>21147</td>
<td>4147</td>
<td>FSS dblayer error</td>
<td>Millisecond precision not supported by c-tree data type in use.</td>
</tr>
<tr>
<td>21991</td>
<td>4991</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 1st index</td>
</tr>
<tr>
<td>21992</td>
<td>4992</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 2nd index</td>
</tr>
<tr>
<td>21993</td>
<td>4993</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 3rd index</td>
</tr>
<tr>
<td>21994</td>
<td>4994</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 4th index</td>
</tr>
<tr>
<td>21995</td>
<td>4995</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 5th index</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>---------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>21996</td>
<td>4996</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 6th index</td>
</tr>
<tr>
<td>21997</td>
<td>4997</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 7th index</td>
</tr>
<tr>
<td>21998</td>
<td>4998</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 8th index</td>
</tr>
<tr>
<td>21999</td>
<td>4999</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 9th index</td>
</tr>
</tbody>
</table>
## 4.10 22000 Series - c-treeDB FSS DB Layer

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>22000</td>
<td>5000</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 10th index</td>
</tr>
<tr>
<td>22001</td>
<td>5001</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 11st index</td>
</tr>
<tr>
<td>22002</td>
<td>5002</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 12nd index</td>
</tr>
<tr>
<td>22003</td>
<td>5003</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 13rd index</td>
</tr>
<tr>
<td>22004</td>
<td>5004</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 14th index</td>
</tr>
<tr>
<td>22005</td>
<td>5005</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 15th index</td>
</tr>
<tr>
<td>22006</td>
<td>5006</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 16th index</td>
</tr>
<tr>
<td>22007</td>
<td>5007</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 17th index</td>
</tr>
<tr>
<td>22008</td>
<td>5008</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 18th index</td>
</tr>
<tr>
<td>22009</td>
<td>5009</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 19th index</td>
</tr>
<tr>
<td>22010</td>
<td>5010</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 20th index</td>
</tr>
<tr>
<td>22011</td>
<td>5011</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 21st index</td>
</tr>
<tr>
<td>22012</td>
<td>5012</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 22nd index</td>
</tr>
<tr>
<td>22013</td>
<td>5013</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 23rd index</td>
</tr>
<tr>
<td>22014</td>
<td>5014</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 24th index</td>
</tr>
<tr>
<td>22015</td>
<td>5015</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 25th index</td>
</tr>
<tr>
<td>22016</td>
<td>5016</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 26th index</td>
</tr>
<tr>
<td>22017</td>
<td>5017</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 27th index</td>
</tr>
<tr>
<td>22018</td>
<td>5018</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 28th index</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>22019</td>
<td>5019</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 29th index</td>
</tr>
<tr>
<td>22020</td>
<td>5020</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 30th index</td>
</tr>
<tr>
<td>22021</td>
<td>5021</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 31st index</td>
</tr>
<tr>
<td>22022</td>
<td>5022</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 32nd index</td>
</tr>
<tr>
<td>22023</td>
<td>5023</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 33rd index</td>
</tr>
<tr>
<td>22024</td>
<td>5024</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 34th index</td>
</tr>
<tr>
<td>22025</td>
<td>5025</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 35th index</td>
</tr>
<tr>
<td>22026</td>
<td>5026</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 36th index</td>
</tr>
<tr>
<td>22027</td>
<td>5027</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 37th index</td>
</tr>
<tr>
<td>22028</td>
<td>5028</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 38th index</td>
</tr>
<tr>
<td>22029</td>
<td>5029</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 39th index</td>
</tr>
<tr>
<td>22030</td>
<td>5030</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 40th index</td>
</tr>
<tr>
<td>22031</td>
<td>5031</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 41st index</td>
</tr>
<tr>
<td>22032</td>
<td>5032</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 42nd index</td>
</tr>
<tr>
<td>22033</td>
<td>5033</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 43rd index</td>
</tr>
<tr>
<td>22034</td>
<td>5034</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 44th index</td>
</tr>
<tr>
<td>22035</td>
<td>5035</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 45th index</td>
</tr>
<tr>
<td>22036</td>
<td>5036</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 46th index</td>
</tr>
<tr>
<td>22037</td>
<td>5037</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 47th index</td>
</tr>
<tr>
<td>22038</td>
<td>5038</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 48th index</td>
</tr>
<tr>
<td>22039</td>
<td>5039</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 49th index</td>
</tr>
<tr>
<td>22040</td>
<td>5040</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 50th index</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>22041</td>
<td>5041</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 51st index</td>
</tr>
<tr>
<td>22042</td>
<td>5042</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 52nd index</td>
</tr>
<tr>
<td>22043</td>
<td>5043</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 53rd index</td>
</tr>
<tr>
<td>22044</td>
<td>5044</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 54th index</td>
</tr>
<tr>
<td>22045</td>
<td>5045</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 55th index</td>
</tr>
<tr>
<td>22046</td>
<td>5046</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 56th index</td>
</tr>
<tr>
<td>22047</td>
<td>5047</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 57th index</td>
</tr>
<tr>
<td>22048</td>
<td>5048</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 58th index</td>
</tr>
<tr>
<td>22049</td>
<td>5049</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 59th index</td>
</tr>
<tr>
<td>22050</td>
<td>5050</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 60th index</td>
</tr>
<tr>
<td>22051</td>
<td>5051</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 61st index</td>
</tr>
<tr>
<td>22052</td>
<td>5052</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 62nd index</td>
</tr>
<tr>
<td>22053</td>
<td>5053</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 63rd index</td>
</tr>
<tr>
<td>22054</td>
<td>5054</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 64th index</td>
</tr>
<tr>
<td>22055</td>
<td>5055</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 65th index</td>
</tr>
<tr>
<td>22056</td>
<td>5056</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 66th index</td>
</tr>
<tr>
<td>22057</td>
<td>5057</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 67th index</td>
</tr>
<tr>
<td>22058</td>
<td>5058</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 68th index</td>
</tr>
<tr>
<td>22059</td>
<td>5059</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 69th index</td>
</tr>
<tr>
<td>22060</td>
<td>5060</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 70th index</td>
</tr>
<tr>
<td>22061</td>
<td>5061</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 71st index</td>
</tr>
<tr>
<td>22062</td>
<td>5062</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 72nd index</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>22063</td>
<td>5063</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 73rd index</td>
</tr>
<tr>
<td>22064</td>
<td>5064</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 74th index</td>
</tr>
<tr>
<td>22065</td>
<td>5065</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 75th index</td>
</tr>
<tr>
<td>22066</td>
<td>5066</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 76th index</td>
</tr>
<tr>
<td>22067</td>
<td>5067</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 77th index</td>
</tr>
<tr>
<td>22068</td>
<td>5068</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 78th index</td>
</tr>
<tr>
<td>22069</td>
<td>5069</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 79th index</td>
</tr>
<tr>
<td>22070</td>
<td>5070</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 80th index</td>
</tr>
<tr>
<td>22071</td>
<td>5071</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 81st index</td>
</tr>
<tr>
<td>22072</td>
<td>5072</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 82nd index</td>
</tr>
<tr>
<td>22073</td>
<td>5073</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 83rd index</td>
</tr>
<tr>
<td>22074</td>
<td>5074</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 84th index</td>
</tr>
<tr>
<td>22075</td>
<td>5075</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 85th index</td>
</tr>
<tr>
<td>22076</td>
<td>5076</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 86th index</td>
</tr>
<tr>
<td>22077</td>
<td>5077</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 87th index</td>
</tr>
<tr>
<td>22078</td>
<td>5078</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 88th index</td>
</tr>
<tr>
<td>22079</td>
<td>5079</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 89th index</td>
</tr>
<tr>
<td>22080</td>
<td>5080</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 90th index</td>
</tr>
<tr>
<td>22081</td>
<td>5081</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 91st index</td>
</tr>
<tr>
<td>22082</td>
<td>5082</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 92nd index</td>
</tr>
<tr>
<td>22083</td>
<td>5083</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 93rd index</td>
</tr>
<tr>
<td>22084</td>
<td>5084</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 94th index</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>22085</td>
<td>5085</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 95th index</td>
</tr>
<tr>
<td>22086</td>
<td>5086</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 96th index</td>
</tr>
<tr>
<td>22087</td>
<td>5087</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 97th index</td>
</tr>
<tr>
<td>22088</td>
<td>5088</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 98th index</td>
</tr>
<tr>
<td>22089</td>
<td>5089</td>
<td>FSS dblayer error</td>
<td>c-treeDB - Key value already exists for 99th index</td>
</tr>
</tbody>
</table>
### 4.11 23000 Series - c-treeACE SQL Data Exceptions

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>23000</td>
<td>22563</td>
<td>c-treeACE SQL Data exception</td>
<td>Table create returned invalid table ID</td>
</tr>
<tr>
<td>23001</td>
<td>22564</td>
<td>c-treeACE SQL Data exception</td>
<td>Index create returned invalid index ID</td>
</tr>
<tr>
<td>23002</td>
<td>22565</td>
<td>c-treeACE SQL Data exception</td>
<td>Invalid index handle</td>
</tr>
</tbody>
</table>

### 4.12 250xx - c-treeACE SQL ODBC Integrator

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>25001</td>
<td>i0i01</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Operation is valid only on a linked table</td>
</tr>
<tr>
<td>25002</td>
<td>i0i02</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Operation not allowed on a linked table</td>
</tr>
<tr>
<td>25003</td>
<td>i0i03</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Copy object exists</td>
</tr>
<tr>
<td>25004</td>
<td>i0i04</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Unknown copy object</td>
</tr>
<tr>
<td>25005</td>
<td>i0i05</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Dropping table failed</td>
</tr>
<tr>
<td>25006</td>
<td>i0i06</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Bad copy SQL statement</td>
</tr>
<tr>
<td>25007</td>
<td>i0i07</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Unknown data type</td>
</tr>
<tr>
<td>25008</td>
<td>i0i08</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Bad insert statement</td>
</tr>
<tr>
<td>25009</td>
<td>i0i09</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Fetch operation failed</td>
</tr>
<tr>
<td>25010</td>
<td>i0i10</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Insert operation failed</td>
</tr>
<tr>
<td>25011</td>
<td>i0i11</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Operation not started</td>
</tr>
<tr>
<td>25012</td>
<td>i0i12</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Operation marked for abort</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>25013</td>
<td>i0i13</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Commit operation failed</td>
</tr>
<tr>
<td>25014</td>
<td>i0i14</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Create table failed</td>
</tr>
<tr>
<td>25015</td>
<td>i0i15</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Bad sync SQL statement</td>
</tr>
<tr>
<td>25016</td>
<td>i0i16</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Sync object exists</td>
</tr>
<tr>
<td>25017</td>
<td>i0i17</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Create sync object failed</td>
</tr>
<tr>
<td>25018</td>
<td>i0i18</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Create copy object failed</td>
</tr>
<tr>
<td>25019</td>
<td>i0i19</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Unknown sync object</td>
</tr>
<tr>
<td>25020</td>
<td>i0i20</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Illegal column name</td>
</tr>
<tr>
<td>25021</td>
<td>i0i21</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Duplicate column name</td>
</tr>
<tr>
<td>25022</td>
<td>i0i22</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Install failure</td>
</tr>
<tr>
<td>25023</td>
<td>i0i23</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Invalid sync mode</td>
</tr>
<tr>
<td>25024</td>
<td>i0i24</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Download or snapshot table missing</td>
</tr>
<tr>
<td>25025</td>
<td>i0i25</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Upload table missing</td>
</tr>
<tr>
<td>25026</td>
<td>i0i26</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Update operation failed</td>
</tr>
<tr>
<td>25027</td>
<td>i0i27</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Delete operation failed</td>
</tr>
<tr>
<td>25028</td>
<td>i0i28</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Close cursor failed</td>
</tr>
<tr>
<td>25029</td>
<td>i0i29</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>No primary key</td>
</tr>
<tr>
<td>25030</td>
<td>i0i30</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Missing row</td>
</tr>
<tr>
<td>25031</td>
<td>i0i31</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Bad primary key</td>
</tr>
<tr>
<td>25032</td>
<td>i0i32</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Update contention</td>
</tr>
<tr>
<td>25033</td>
<td>i0i33</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Link table failed</td>
</tr>
<tr>
<td>25034</td>
<td>i0i34</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Unlink table failed</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>25035</td>
<td>i0i35</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Link data source failed</td>
</tr>
<tr>
<td>25036</td>
<td>i0i36</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Unlink data source failed</td>
</tr>
<tr>
<td>25037</td>
<td>i0i37</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Integrator internal error</td>
</tr>
<tr>
<td>25038</td>
<td>i0i38</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Operation already started</td>
</tr>
<tr>
<td>25039</td>
<td>i0i39</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Opening of copy SQL statement failed</td>
</tr>
<tr>
<td>25040</td>
<td>i0i40</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Sync object failed</td>
</tr>
<tr>
<td>25041</td>
<td>i0i41</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Dropping copy object failed</td>
</tr>
<tr>
<td>25042</td>
<td>i0i42</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Closing copy SQL statement failed</td>
</tr>
<tr>
<td>25043</td>
<td>i0i43</td>
<td>c-treeACE SQL ODBC integrator</td>
<td>Failure to update metadata timestamp</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>25101</td>
<td>j0j01</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLAllocEnv failed</td>
</tr>
<tr>
<td>25102</td>
<td>j0j02</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLAllocConnect failed</td>
</tr>
<tr>
<td>25103</td>
<td>j0j03</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLConnect failed</td>
</tr>
<tr>
<td>25104</td>
<td>j0j04</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLGetConnectOption failed</td>
</tr>
<tr>
<td>25105</td>
<td>j0j05</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLSetConnectOption failed</td>
</tr>
<tr>
<td>25106</td>
<td>j0j06</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>Failed to map statement handle to UUID</td>
</tr>
<tr>
<td>25107</td>
<td>j0j07</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLSetParam failed</td>
</tr>
<tr>
<td>25108</td>
<td>j0j08</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLDisconnect failed</td>
</tr>
<tr>
<td>25109</td>
<td>j0j09</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLExecute failed</td>
</tr>
<tr>
<td>25110</td>
<td>j0j10</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLRowCount failed</td>
</tr>
<tr>
<td>25111</td>
<td>j0j11</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLSetParam failed</td>
</tr>
<tr>
<td>25112</td>
<td>j0j12</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLBindCol failed</td>
</tr>
<tr>
<td>25113</td>
<td>j0j13</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLPrepare failed</td>
</tr>
<tr>
<td>25114</td>
<td>j0j14</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLResultCols failed</td>
</tr>
<tr>
<td>25115</td>
<td>j0j15</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLDescribeCol failed</td>
</tr>
<tr>
<td>25116</td>
<td>j0j16</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLFreeStmt failed</td>
</tr>
<tr>
<td>25117</td>
<td>j0j17</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLFetch failed</td>
</tr>
<tr>
<td>25118</td>
<td>j0j18</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLTransact failed</td>
</tr>
<tr>
<td>25119</td>
<td>j0j19</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLAllocStmt failed</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>25120</td>
<td>j0j20</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLTables failed</td>
</tr>
<tr>
<td>25121</td>
<td>j0j21</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLColumns failed</td>
</tr>
<tr>
<td>25122</td>
<td>j0j22</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLStatistics failed</td>
</tr>
<tr>
<td>25123</td>
<td>j0j23</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>ODBC Driver interface mismatch</td>
</tr>
<tr>
<td>25124</td>
<td>j0j24</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>ODBC Driver metadata exceeds storage limits</td>
</tr>
<tr>
<td>25125</td>
<td>j0j25</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>SQLGetInfo failed</td>
</tr>
<tr>
<td>25126</td>
<td>j0j26</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>Operation not allowed on the read-only database</td>
</tr>
<tr>
<td>25127</td>
<td>j0j27</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>Cannot update views-with-check-option on remote tables</td>
</tr>
<tr>
<td>25128</td>
<td>j0j28</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>Query terminated as max row limit exceeded for a remote table</td>
</tr>
<tr>
<td>25131</td>
<td>j0j29</td>
<td>c-treeACE SQL ODBC trans layer</td>
<td>Unable to read column info from remote table</td>
</tr>
</tbody>
</table>
# 4.14 26000 Series - c-treeACE SQL JDBC Driver

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>26039</td>
<td>50581</td>
<td>c-treeACE SQL RDS error</td>
<td>Statement_id exceeds maximum length supported %s</td>
</tr>
<tr>
<td>26000</td>
<td>26000</td>
<td>Access rule violation</td>
<td>ctree options conflict</td>
</tr>
<tr>
<td>26001</td>
<td>26001</td>
<td>Feature not supported</td>
<td>Support for this future feature is not enabled</td>
</tr>
</tbody>
</table>

# 4.15 30000 Series - c-treeACE SQL Network

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>30001</td>
<td>5050w</td>
<td>c-treeACE SQL RDS errors</td>
<td>Query aborted on user request</td>
</tr>
<tr>
<td>30002</td>
<td>k0k02</td>
<td>c-treeACE SQL network interface</td>
<td>Invalid network handle</td>
</tr>
<tr>
<td>30003</td>
<td>k0k03</td>
<td>c-treeACE SQL network interface</td>
<td>Invalid sqlnetwork INTERFACE</td>
</tr>
<tr>
<td>30004</td>
<td>k0k04</td>
<td>c-treeACE SQL network interface</td>
<td>Invalid sqlnetwork INTERFACE procedure</td>
</tr>
<tr>
<td>30005</td>
<td>k0k05</td>
<td>c-treeACE SQL network interface</td>
<td>INTERFACE is already attached</td>
</tr>
<tr>
<td>30006</td>
<td>k0k06</td>
<td>c-treeACE SQL network interface</td>
<td>INTERFACE entry not found</td>
</tr>
<tr>
<td>30007</td>
<td>k0k07</td>
<td>c-treeACE SQL network interface</td>
<td>INTERFACE is already registered</td>
</tr>
<tr>
<td>30008</td>
<td>k0k08</td>
<td>c-treeACE SQL network interface</td>
<td>Mismatch in packet header size and total argument size</td>
</tr>
<tr>
<td>30009</td>
<td>k0k09</td>
<td>c-treeACE SQL network interface</td>
<td>Invalid server ID</td>
</tr>
<tr>
<td>30010</td>
<td>k0k10</td>
<td>c-treeACE SQL network interface</td>
<td>Reply does not match the request</td>
</tr>
<tr>
<td>30011</td>
<td>k0k02</td>
<td>c-treeACE SQL network interface</td>
<td>Memory allocation failure</td>
</tr>
<tr>
<td>30012</td>
<td>S1000</td>
<td>c-treeACE SQL Query Timeout</td>
<td>Query has timed out</td>
</tr>
<tr>
<td>Error Code</td>
<td>SQLSTATE Value</td>
<td>Class Condition</td>
<td>Subclass Message</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>30031</td>
<td>k0k11</td>
<td>c-treeACE SQL network interface</td>
<td>Error in transmission of packet</td>
</tr>
<tr>
<td>30032</td>
<td>k0k12</td>
<td>c-treeACE SQL network interface</td>
<td>Error in reception of packet</td>
</tr>
<tr>
<td>30033</td>
<td>k0k13</td>
<td>c-treeACE SQL network interface</td>
<td>No packet received</td>
</tr>
<tr>
<td>30034</td>
<td>k0k14</td>
<td>c-treeACE SQL network interface</td>
<td>Connection reset</td>
</tr>
<tr>
<td>30051</td>
<td>k0k15</td>
<td>c-treeACE SQL network interface</td>
<td>Network handle is inprocess handle</td>
</tr>
<tr>
<td>30061</td>
<td>k0k16</td>
<td>c-treeACE SQL network interface</td>
<td>Could not connect to SQL network daemon</td>
</tr>
<tr>
<td>30062</td>
<td>k0k17</td>
<td>c-treeACE SQL network interface</td>
<td>Error in number of arguments</td>
</tr>
<tr>
<td>30063</td>
<td>k0k18</td>
<td>c-treeACE SQL network interface</td>
<td>Requested INTERFACE not registered</td>
</tr>
<tr>
<td>30064</td>
<td>k0k19</td>
<td>c-treeACE SQL network interface</td>
<td>Invalid INTERFACE procedure id</td>
</tr>
<tr>
<td>30065</td>
<td>k0k20</td>
<td>c-treeACE SQL network interface</td>
<td>Requested server executable not found</td>
</tr>
<tr>
<td>30066</td>
<td>k0k21</td>
<td>c-treeACE SQL network interface</td>
<td>Invalid configuration information</td>
</tr>
<tr>
<td>30067</td>
<td>k0k22</td>
<td>c-treeACE SQL network interface</td>
<td>INTERFACE not supported</td>
</tr>
<tr>
<td>30091</td>
<td>k0k23</td>
<td>c-treeACE SQL network interface</td>
<td>invalid service name</td>
</tr>
<tr>
<td>30092</td>
<td>k0k24</td>
<td>c-treeACE SQL network interface</td>
<td>Invalid host</td>
</tr>
<tr>
<td>30093</td>
<td>k0k25</td>
<td>c-treeACE SQL network interface</td>
<td>Error in TCP/IP accept call</td>
</tr>
<tr>
<td>30094</td>
<td>k0k26</td>
<td>c-treeACE SQL network interface</td>
<td>Error in TCP/IP connect call</td>
</tr>
<tr>
<td>30095</td>
<td>k0k27</td>
<td>c-treeACE SQL network interface</td>
<td>Error in TCP/IP bind call</td>
</tr>
<tr>
<td>30096</td>
<td>k0k28</td>
<td>c-treeACE SQL network interface</td>
<td>Error in creating socket</td>
</tr>
<tr>
<td>30097</td>
<td>k0k29</td>
<td>c-treeACE SQL network interface</td>
<td>Error in setting socket option</td>
</tr>
<tr>
<td>30101</td>
<td>k0k30</td>
<td>c-treeACE SQL network interface</td>
<td>Interrupt occurred</td>
</tr>
<tr>
<td>30102</td>
<td>k0k31</td>
<td>c-treeACE SQL network interface</td>
<td>Client/Server not WideChar Compatible</td>
</tr>
</tbody>
</table>
### 4.16 40000 Series - c-treeACE SQL Environment Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>40001</td>
<td>L0L01</td>
<td>c-treeACE SQL environment error</td>
<td>Error in reading configuration</td>
</tr>
</tbody>
</table>

### 4.17 50000 Series - c-treeACE SQL DHRSS Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>SQLSTATE Value</th>
<th>Class Condition</th>
<th>Subclass Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>50000</td>
<td>60614</td>
<td>c-treeACE SQL DHRSS errors</td>
<td>DHRSS - Improper call to DFLT SS</td>
</tr>
</tbody>
</table>
5. System Catalog Tables

5.1 Overview

c-treeACE SQL maintains a set of system tables for storing information about table spaces, tables, columns, indexes, constraints, and privileges.

c-treeACE SQL data definition statements and GRANT and REVOKE statements update system catalog tables. Users have read access to the system catalog tables. The database administrator has update access to the tables, but should avoid modifying them directly.

There are two types of tables in the system catalog: base tables and extended tables. Base tables store the information on the table spaces, tables, columns, and indexes that make up the database. There are four system tables:

- systables
- systblspaces
- syscolumns
- sysindexes

The rest of the system catalog tables are extended tables. They contain detailed information on database objects and statistical information.

The owner of the system tables is admin. If you connect to a c-treeACE SQL environment with a User ID other than admin, you need to qualify references to the tables in c-treeACE SQL queries. For example:

```
SELECT * FROM ADMIN.SYSTABLES
```

The following table shows details of the columns in each system table. Here is the c-treeACE SQL query that generated the data for this table. You can modify it to generate a similar list that includes user-created tables by omitting the line `and st.tbltype = 'S'`.

```
select sc.tbl 'Table', sc.col 'Column',
    sc.coltype 'Data Type', sc.width 'Size'
from admin.syscolumns sc, admin.systables st
where sc.tbl = st.tbl
    and st.tbltype = 'S'
order by sc.tbl, sc.id
```

5.2 System Catalog Tables Definitions

The following table lists all the tables in the system catalog. It gives a brief description of their purpose and lists the column definitions for every table.

System Catalog Table Definitions
<table>
<thead>
<tr>
<th>Table</th>
<th>Purpose</th>
<th>Column</th>
<th>Data Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_chk_constrs</td>
<td>Contains the CHECK clause for each check constraint specified on a user table.</td>
<td>chkclause</td>
<td>varchar</td>
<td>8192</td>
</tr>
<tr>
<td></td>
<td></td>
<td>chkseq</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cnstname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>owner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td>sys_chkcol_usage</td>
<td>Contains one entry for each column on which the check constraint is specified</td>
<td>cnstname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>colname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>owner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td>sys_keycol_usage</td>
<td>Contains one entry for each column on which primary or foreign key is specified</td>
<td>cnstname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>colname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>colposition</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>owner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td>sys_ref_constrs</td>
<td>Contains one entry for each referential constraint specified on a user table</td>
<td>cnstname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>deleterule</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>owner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>refcnstname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>refowner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reftblname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td>sys_tbl_constrs</td>
<td>Contains one entry for each table constraint.</td>
<td>cnstname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cnstrtype</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>idxname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>owner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td>sysattachtbls</td>
<td>Contains one entry for each table link.</td>
<td>db_link</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>linkowner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>remtbl</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>remowner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tbl</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>owner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td>sysbigintstat</td>
<td>Contains one row for each bigint column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rssid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val1</td>
<td>bigint</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>bigint</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val10</td>
<td>bigint</td>
<td>8</td>
</tr>
<tr>
<td>syiscalctable</td>
<td>Contains exactly one row with a single column with a value of 100.</td>
<td>fld</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td>syscharstat</td>
<td>Contains one row for each char column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rssid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val1</td>
<td>varchar</td>
<td>8192</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>varchar</td>
<td>8192</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val10</td>
<td>varchar</td>
<td>8192</td>
</tr>
<tr>
<td>Table</td>
<td>Purpose</td>
<td>Column</td>
<td>Data Type</td>
<td>Size</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>------</td>
</tr>
<tr>
<td>syscolauth</td>
<td>Contains the update privileges held by users on individual columns of tables in the database.</td>
<td>col, grantee, grantor, ref, tbl, tblowner, upd</td>
<td>varchar, varchar, varchar, varchar, varchar, varchar</td>
<td>64, 64, 64, 1, 64, 1</td>
</tr>
<tr>
<td>syscolstat</td>
<td>Provides mapping information between syscolumns and sys*stat tables.</td>
<td>colid, coltype, rssid, tblid</td>
<td>integer, varchar, integer, integer</td>
<td>4, 12, 4, 4</td>
</tr>
<tr>
<td>syscolumns</td>
<td>Contains one row for each column of every table in the database.</td>
<td>charset, col, collation, coltype, dflt_value, id, nullflag, owner, scale, tbl, width</td>
<td>varchar, varchar, varchar, varchar, varchar, varchar, varchar, integer, varchar, integer</td>
<td>64, 64, 64, 12, 250, 4, 1, 64, 4, 64, 4</td>
</tr>
<tr>
<td>sysdatatypes</td>
<td>Contains information on each data type supported by the database.</td>
<td>autoincr, casesensitive, createparams, datatype, dtypename, literalprefix, literalsuffix, localtypename, nullable, odbcmoney, searchable, typeprecision, unsignedattr</td>
<td>smallint, smallint, varchar, smallint, smallint, varchar, smallint, varchar, varchar, varchar, integer, smallint</td>
<td>2, 2, 64, 2, 64, 2, 2, 1, 1, 2, 2, 2</td>
</tr>
<tr>
<td>sysdatestat</td>
<td>Contains one row for each date column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid, rssid, tblid, val1, ..., val10</td>
<td>integer, integer, integer, date, date</td>
<td>4, 4, 4, 4, 4</td>
</tr>
<tr>
<td>sysdblinks</td>
<td>Contains one entry for each data source link.</td>
<td>host, password, username, db_link, owner</td>
<td>varchar, varchar, varchar, varchar, varchar</td>
<td>128, 30, 64, 64, 64</td>
</tr>
<tr>
<td>sysdbauth</td>
<td>Contains the database-wide privileges held by users.</td>
<td>dba_acc, grantee, res_acc</td>
<td>varchar, varchar, varchar</td>
<td>1, 64, 1</td>
</tr>
<tr>
<td>Table</td>
<td>Purpose</td>
<td>Column</td>
<td>Data Type</td>
<td>Size</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td>sysfloatstat</td>
<td>Contains one row for each float column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid rssid tblid val1 ⋯ val10</td>
<td>integer integer integer float float</td>
<td>4 4 4 8 8</td>
</tr>
<tr>
<td>sysgroupauth</td>
<td>Contains privileges held by user groups</td>
<td>grantor granteetbowner tbl ins del upd sel exedidxalt</td>
<td>[n]varchar [n]varchar [n]varchar [n]varchar [n]varchar [n]varchar [n]varchar [n]varchar [n]varchar [n]varchar [n]varchar</td>
<td>64 64 64 64 64 64 64 64 64 64 64</td>
</tr>
<tr>
<td>sysgroups</td>
<td>Contains list of user groups</td>
<td>groupname gid</td>
<td>[n]varchar integer</td>
<td>64 4</td>
</tr>
<tr>
<td>sysgroupusers</td>
<td>Contains list of users in each group</td>
<td>username groupname</td>
<td>[n]varchar [n]varchar</td>
<td>64 64</td>
</tr>
<tr>
<td>sysidentity</td>
<td>Contains one row for each identity field in the database. Used for SQL IDENTITY support.</td>
<td>owner tbl colid seed incr</td>
<td>varchar varchar integer integer integer</td>
<td>64 64 4 4 4</td>
</tr>
<tr>
<td>sysidxstat</td>
<td>Contains statistics for each index in the database.</td>
<td>idxid nleaf nlevels recsz rssid tblid</td>
<td>integer integer smallint integer integer integer</td>
<td>4 4 2 4 4 4</td>
</tr>
<tr>
<td>sysindexes</td>
<td>Contains one row for each component of an index in the database. For an index with n components, there will be n rows in this table.</td>
<td>colname id idxcompress idxmethod idxname idxorder idxowner idxsegid idxseq idxtype rssid tbl tblowner</td>
<td>varchar integer varchar varchar varchar varchar varchar varchar varchar varchar varchar varchar</td>
<td>64 4 1 1 64 1 64 1 4 4 4 64 64</td>
</tr>
<tr>
<td>Table</td>
<td>Purpose</td>
<td>Column</td>
<td>Data Type</td>
<td>Size</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>-------</td>
</tr>
<tr>
<td>sysintstat</td>
<td>Contains one row for each integer column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid rssid tblid val1 … val10</td>
<td>integer integer integer integer integer</td>
<td>4 4 4 4 4</td>
</tr>
<tr>
<td>sysmoneystat</td>
<td>Contains one row for each money column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid rssid tblid val1 … val10</td>
<td>integer integer integer money money</td>
<td>4 4 4 32 32</td>
</tr>
<tr>
<td>sysncharstat</td>
<td>Contains one row for each national char() column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid rssid tblid val1 … val10</td>
<td>integer integer integer nchar nchar</td>
<td>4 4 4 8192 8192</td>
</tr>
<tr>
<td>sysnumstat</td>
<td>Contains one row for each numeric column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid rssid tblid val1 … val10</td>
<td>integer integer integer numeric numeric</td>
<td>4 4 4 64 64</td>
</tr>
<tr>
<td>sysnvarcharstat</td>
<td>Contains one row for each national varchar column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid rssid tblid val1 … val10</td>
<td>integer integer integer varchar varchar</td>
<td>4 4 4 8192 8192</td>
</tr>
<tr>
<td>sysprocbin</td>
<td>Contains one or more rows for each stored procedure and trigger in the database, which contain the compiled Java bytecode for their procedure or trigger.</td>
<td>id proc_bin proc_type rssid seq</td>
<td>integer varbinary char integer integer</td>
<td>4 8192 2 4 4</td>
</tr>
<tr>
<td>sysproccolumns</td>
<td>Contains one row for each column of a stored procedure’s result set.</td>
<td>argtype col datatype dflt_value id nullflag proc_id rssid scale width</td>
<td>varchar varchar varchar integer character integer integer integer integer</td>
<td>64 64 64 250 4 1 4 4 4</td>
</tr>
<tr>
<td>Table</td>
<td>Purpose</td>
<td>Column</td>
<td>Data Type</td>
<td>Size</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------</td>
<td>----------------------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>sysprocedures</td>
<td>Contains one row for each stored procedure in the database.</td>
<td>creator</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>has_resultset</td>
<td>character</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>has_return_val</td>
<td>character</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>owner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>proc_id</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>proc_name</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>proc_type</td>
<td>varchar</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rssid</td>
<td>integer</td>
<td>64</td>
</tr>
<tr>
<td>sysproctext</td>
<td>Contains one or more rows for each stored procedure and trigger in the database, which contain the Java source code for their procedure or trigger.</td>
<td>id</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>proc_text</td>
<td>varchar</td>
<td>8192</td>
</tr>
<tr>
<td></td>
<td></td>
<td>proc_type</td>
<td>character</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rssid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>seq</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td>sysrealstat</td>
<td>Contains one row for each real column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rssid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val1</td>
<td>real</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>real</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val10</td>
<td>real</td>
<td>4</td>
</tr>
<tr>
<td>syssmintstat</td>
<td>Contains one row for each smallint column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rssid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val1</td>
<td>smallint</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>smallint</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val10</td>
<td>smallint</td>
<td>2</td>
</tr>
<tr>
<td>syssynonyms</td>
<td>Contains one entry for each synonym in the database.</td>
<td>ispublic</td>
<td>smallint</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>screator</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sowner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sremdb</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stbl</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stblowner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td>systabauth</td>
<td>Contains privileges held by users for tables, views, and procedures.</td>
<td>alt</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>del</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>exe</td>
<td>character</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>grantee</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>grantor</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ins</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ndx</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ref</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sel</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tbl</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblowner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>upd</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td>Table</td>
<td>Purpose</td>
<td>Column</td>
<td>Data Type</td>
<td>Size</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------</td>
<td>-------------------------</td>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td>systables</td>
<td>Contains one row for each table in the database.</td>
<td>creator</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>has_ccnstrs</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>has_fcnstrs</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>has_pcnstrs</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>has_ucnstrs</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>id</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>owner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rssid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>segid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tbl</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tbl_status</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblpctfree</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tbltype</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td>systblspaces</td>
<td>No longer used.</td>
<td>id</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tname</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td>systblistat</td>
<td>Contains table statistics for each user table.</td>
<td>card</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>npages</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pagesz</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>recsz</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rssid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td>systimestat</td>
<td>Contains one row for each time column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rssid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val1</td>
<td>time</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>time</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val10</td>
<td>time</td>
<td>4</td>
</tr>
<tr>
<td>systinyintstat</td>
<td>Contains one row for each tinyint column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rssid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val1</td>
<td>time</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>time</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>val10</td>
<td>time</td>
<td>4</td>
</tr>
<tr>
<td>systrigcols</td>
<td>Contains one row for each column specified in each UPDATE trigger in the database.</td>
<td>colid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>owner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>triggername</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td>systrigger</td>
<td>Contains one row for each trigger in the database.</td>
<td>owner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>refers_to_new</td>
<td>character</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>refers_to_old</td>
<td>character</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rssid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>statement_or_row</td>
<td>character</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tbl</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tblowner</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>trigger_event</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>trigger_time</td>
<td>varchar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>triggerid</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>triggername</td>
<td>varchar</td>
<td>64</td>
</tr>
<tr>
<td>Table</td>
<td>Purpose</td>
<td>Column</td>
<td>Data Type</td>
<td>Size</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------</td>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td>systsstat</td>
<td>Contains one row for each timestamp column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid, rssid, tblid, val1, ... val10</td>
<td>integer, integer, integer, timestamp</td>
<td>4, 4, 4, 8</td>
</tr>
<tr>
<td>sysvarcharstat</td>
<td>Contains one row for each varchar column. Used by the optimizer, each row contains a sampling of values in the column.</td>
<td>colid, rssid, tblid, val1, ... val10</td>
<td>integer, integer, integer, varchar</td>
<td>4, 4, 4, 8192</td>
</tr>
<tr>
<td>sysviews</td>
<td>Contains information on each view in the database.</td>
<td>creator, owner, seq, viewname, viewtext</td>
<td>varchar, varchar, integer, varchar</td>
<td>64, 64, 4, 64, 8192</td>
</tr>
</tbody>
</table>
# 6. System Limits

## 6.1 Maximum Values for c-treeACE SQL Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of procedure arguments in a c-treeACE SQL CALL statement</td>
<td>50</td>
</tr>
<tr>
<td>Maximum length of a c-treeACE SQL statement</td>
<td>32MB (prior to V10.2, the limit was 35,000 characters)</td>
</tr>
<tr>
<td>Maximum length of a column: standard data types.</td>
<td>8192</td>
</tr>
<tr>
<td>Maximum length of a column: VARBINARY and VARCHAR specifying the character set designated as NATIONAL CHARACTER</td>
<td>32752</td>
</tr>
<tr>
<td>Maximum length of default value specification</td>
<td>250</td>
</tr>
<tr>
<td>Maximum length for an identifier</td>
<td>64</td>
</tr>
<tr>
<td>Maximum length of a connect string</td>
<td>100</td>
</tr>
<tr>
<td>Maximum length for a user-name in a connect string</td>
<td>64</td>
</tr>
<tr>
<td>Maximum number of database connections (Remote server connections. This feature is not currently supported with c-treeACE SQL.)</td>
<td>10</td>
</tr>
<tr>
<td>Maximum length of an error message</td>
<td>511</td>
</tr>
<tr>
<td>Maximum number of columns in a table</td>
<td>2000</td>
</tr>
<tr>
<td>Maximum number of index components for a table, for all indexes on that table</td>
<td>200</td>
</tr>
<tr>
<td>Maximum length of a CHECK constraint clause</td>
<td>20,000</td>
</tr>
<tr>
<td>Maximum number of check constraints in a table</td>
<td>4000</td>
</tr>
<tr>
<td>Maximum number of foreign constraints in a table</td>
<td>4000</td>
</tr>
<tr>
<td>Maximum number of nesting levels in a c-treeACE SQL statement</td>
<td>25</td>
</tr>
<tr>
<td>Maximum number of table references in a c-treeACE SQL statement</td>
<td>250</td>
</tr>
<tr>
<td>Maximum size of input parameters for a c-treeACE SQL statement</td>
<td>512</td>
</tr>
<tr>
<td>Maximum number of outer references in a c-treeACE SQL statement</td>
<td>25</td>
</tr>
<tr>
<td>Maximum nesting level for view references</td>
<td>25</td>
</tr>
<tr>
<td>Maximum recursion level for stored procedures</td>
<td>10</td>
</tr>
</tbody>
</table>
7. **Glossary**

**add [an ODBC data source]**
Make a data source available to ODBC through the Add operation of the ODBC Administrator utility. Adding a data source tells ODBC where a specific database resides and which ODBC driver to use to access it. Adding a data source also invokes a setup dialog box for the particular driver so you can provide other details the driver needs to connect to the database.

**admin**
The user name for the database administrator in a c-treeACE SQL environment. The admin user is the owner of all system tables. Users must qualify references to system tables as `admin.tablename`.

**alias**
A temporary name for a table or column specified in the `FROM` clause of a c-treeACE SQL query expression. Also called correlation name. Derived tables and search conditions that join a table with itself must specify an alias. Once a query specifies an alias, references to the table or column must use the alias and not the underlying table or column name.

**applet**
A special kind of Java program whose compiled class files a Java-enabled browser can download from the Internet and run.

**ASCII**

**bytecode**
Machine-independent code generated by the Java compiler and executed by the Java interpreter.

**candidate key**
Another term for unique key.

**cardinality**
Number of rows in a result table.

**Cartesian product**
Also called cross-product. In a query expression, the result table generated when a `FROM` clause lists more than one table but specifies no join conditions. In such a case, the result table is formed by concatenating every row of every table with all other rows in all tables. Typically, Cartesian products are not useful and are slow to process.

**client**
Generally, in client/server systems, the part of the system that sends requests to servers and processes the results of those requests.
client character set
A client character set is a character set used by a client application for representing c-treeACE SQL statements and data.

clipboard
A temporary storage area for text and graphics that you want to copy from one location or application to another. Most Windows applications support the clipboard through cut, copy, and paste operations.

collation
The rules used to control how character strings in a character set compare with each other. Each character set specifies a collating sequence that defines relative values of each character for comparing, merging and sorting character strings.

column alias
An alias specified for a column. See alias.

constraint
Part of a c-treeACE SQL table definition that restricts the values that can be stored in a table. When you insert, delete, or update column values, the constraint checks the new values against the conditions specified by the constraint. If the value violates the constraint, it generates an error. Along with triggers, constraints enforce referential integrity by insuring that a value stored in the foreign key of a table must either be null or be equal to some value in the matching unique or primary key of another table.

correlation name
Another term for alias.

cross product
Another term for Cartesian product.

data dictionary
Another term for system catalog.

data source
See ODBC data source.

database administrator
The user name admin is the database administrator.

delete [an ODBC data source]
Remove information about an ODBC data source through the Delete operation of the ODBC Administrator utility. Deleting a data source does not delete the database it corresponds to, but removes information about the database’s location from the registry (on Windows NT) or the odbc.ini file (on Unix).

delimited identifiers
Names in c-treeACE SQL statements enclosed in double quotation marks ("""). Enclosing a name in double quotation marks preserves the case of the name and allows it to include reserved words.
and special characters. Subsequent references to a delimited identifier must also use enclosing double quotation marks.

**derived table**
A virtual table specified as a query expression in the FROM clause of another query expression.

**driver manager**
See JDBC driver manager and ODBC driver manager.

**foreign key**
A column or columns in a table whose values must either be null or equal to some value in a corresponding column (called the primary key) in another table. Use the REFERENCES clause in the c-treeACE SQL CREATE TABLE statement to create foreign keys.

**form of use**
The storage format for characters in a character set. Some character sets, such as ASCII, require one byte (octet) for each character. Others, such as Unicode, use two bytes, and are called multi-octet character sets.

**input parameter**
In a stored procedure specification, an argument that an application must pass when it calls the stored procedure. In an c-treeACE SQL statement, a parameter marker in the statement string that acts as a placeholder for a value that will be substituted when the statement executes.

**interface**
In Java, a definition of a set of methods that one or more objects will implement. Interfaces declare only methods and constants, not variables. Interfaces provide multiple-inheritance capabilities.

**Java snippet**
See snippet.

**JDBC**
Java Database Connectivity: a part of the Java language that allows applications to embed standard SQL statements and access any database that implements a JDBC driver.

**JDBC driver**
Database-specific software that receives calls from the JDBC driver manager, translates them into a form that the database can process, and returns data to the application.

**JDBC driver manager**
A Java class that implements methods to route calls from a JDBC application to the appropriate JDBC driver for a particular JDBC URL.

**join**
A relational operation that combines data from two tables.

**manager**
A main component of c-treeACE SQL. c-treeACE SQL includes several managers, including the c-treeACE SQL statement manager, parser, and optimizer.
metadata
Data that details the structure of tables and indexes in the proprietary storage system. The c-treeACE SQL engine stores metadata in the system catalog.

octet
A group of 8 bits. Synonymous with byte, and often used in descriptions of character-set encoding format.

ODBC Administrator
On Windows NT, a Microsoft-supplied utility to add and delete ODBC data sources and drivers. The Administrator maintains information in the system registry.

ODBC application
Any program that calls ODBC functions and uses them to issue c-treeACE SQL statements. Many vendors have added ODBC capabilities to their existing Windows-based tools.

ODBC data source
In ODBC terminology, the technical information needed to access data—the driver name, network address, network software, and so on. Before applications can access a database through ODBC, you use the ODBC Administrator (or, on Unix, edit the odbc.ini file) to add a data source for that database. More than one data source name can refer to the same database, and deleting a data source does not delete the associated database.

ODBC driver
Vendor-supplied software that processes ODBC function calls for a specific data source. The driver connects to the data source, translates the standard c-treeACE SQL statements into syntax the data source can process, and returns data to the application. c-treeACE SQL includes an ODBC driver that provides access to c-treeACE SQL.

ODBC driver manager
On Windows NT, a Microsoft-supplied program that routes calls from an application to the appropriate ODBC driver for a data source.

optimizer
Within c-treeACE SQL the manager that analyzes costs and statistics associated with the statement and converts the relational algebra tree to the most efficient form for execution. The optimizer stores the trees for later use.

output parameter
In a stored procedure specification, an argument in which the stored procedure returns a value after it executes.

package
A group of related Java classes and interfaces, like a class library in C++. The Java development environment includes many packages of classes that procedures can import. The Java runtime system automatically imports the java.lang package. Stored procedures must explicitly import other classes by specifying them in the IMPORT clause of a CREATE PROCEDURE statement.

parameter marker
A question mark (?) in a procedure call or c-treeACE SQL statement string that acts as a placeholder for an input or output parameter supplied at runtime when the procedure executes. The CALL statement (or corresponding ODBC or JDBC escape clause) uses parameter markers to pass parameters to stored procedures. The SQLStatement, SQLPStatement, and SQLCursor objects use them within procedures.

postfix notation
Notation in which the numbers precede the operation. For example, 2 + 2 is expressed as 2 2 +, and 10 - 3 * 4 would be 10 3 4 * -. If a storage manager supports processing of expressions, c-treeACE SQL passes them to the storage manager using postfix notation.

primary key
A subset of the fields in a table, characterized by the constraint that no two records in a table may have the same primary key value, and that no fields of the primary key may have a null value. Primary keys are specified in a CREATE TABLE statement.

procedure body
In a stored procedure, the Java code between the BEGIN and END keywords of a CREATE PROCEDURE statement.

procedure result set
In a stored procedure, a set of data rows returned to the calling application. The number and data types of columns in the procedure result set are specified in the RESULT clause of the CREATE PROCEDURE statement. The procedure can transfer data from an c-treeACE SQL result set to the procedure result set or it can store data generated internally. A stored procedure can have only one procedure result set.

procedure specification
In a CREATE PROCEDURE statement, the clauses preceding the procedure body that specify the procedure name, any input and output parameters, any result set columns, and any Java packages to import.

procedure variable
A Java variable declared within the body of a stored procedure, as compared to a procedure input parameter or output parameter, which are declared outside the procedure body and are visible to the application that calls the stored procedure.

query expression
The fundamental element in c-treeACE SQL syntax. Query expressions specify a result table derived from some combination of rows from the tables or views identified in the FROM clause of the expression. Query expressions are the basis of SELECT, CREATE VIEW, and INSERT statements, and can be used in some expressions and search conditions.

referential integrity
The condition where the value stored in a database table’s foreign key must either be null or be equal to some value in another table’s the matching unique or primary key. c-treeACE SQL provides two mechanisms to enforce referential integrity: constraints specified as part of CREATE
TABLE statements prevent updates that violate referential integrity, and triggers specified in CREATE TRIGGER statements execute a stored procedure to enforce referential integrity.

**repertoire**

The set of characters allowed in a character set.

**result set**

In a stored procedure, either an c-treeACE SQL result set or a procedure result set. More generally, another term for result table.

**result table**

A temporary table of values derived from columns and rows of one or more tables that meet conditions specified by a query expression.

**row identifier**

Another term for tuple identifier.

**search condition**

c-treeACE SQL syntax element that specifies a condition that is true or false about a given row or group of rows. Query expressions and UPDATE statements can specify a search condition. The search condition restricts the number of rows in the result table for the query expression or UPDATE statement. Search conditions contain one or more predicates. Search conditions follow the WHERE or HAVING keywords in c-treeACE SQL statements.

**selectivity**

The fraction of a table’s rows returned by a query.

**server**

Generally, in client/server systems, the part of the system that receives requests from clients and responds with results to those requests.

**snippet**

In a stored procedure, the sequence of Java statements between the BEGIN and END keywords in the CREATE PROCEDURE (or CREATE TRIGGER) statement. The Java statements become a method in a class c-treeACE SQL creates and submits to the Java compiler.

**SQL diagnostics area**

A data structure that contains information about the execution status (success, error or warning conditions) of the most recent c-treeACE SQL statement. The SQL-92 standard specified the diagnostics area as a standardized alternative to widely varying implementations of the SQLCA. c-treeACE SQL supports both the SQLCA and the c-treeACE SQL diagnostics area. The GET DIAGNOSTICS statement returns information about the diagnostics area to an application, including the value of the SQLSTATE status parameter.

**SQL engine**

The core component of the c-treeACE SQL environment. c-treeACE SQL receives requests from applications, processes them, and returns results. c-treeACE SQL calls c-treeDB to convey requests to c-treeACE.
**SQL result set**

In a stored procedure, the set of data rows generated by an c-treeACE SQL statement (SELECT and, in some cases, CALL).

**SQLCA**

c-treeACE SQL Communications area: A data structure that contains information about the execution status (success, error or warning conditions) of the most recent c-treeSQL statement. The SQLCA includes an SQLCODE field. The SQLCA provides the same information as the c-treeACE SQL diagnostics area, however, is not compliant with the SQL-92 standard. c-treeACE SQL supports both the SQLCA and the c-treeACE SQL diagnostics area.

**SQLCODE**

An integer status parameter whose value indicates the condition status returned by the most recent c-treeACE SQL statement. An SQLCODE value of zero means success, a positive value means warning, and a negative value means an error status. SQLCODE is superseded by SQLSTATE in the SQL-92 standard. Applications declare either SQLSTATE or SQLCODE, or both. c-treeACE SQL returns the status to SQLSTATE or SQLCODE after execution of each c-treeACE SQL statement.

**SQLSTATE**

A 5-character status parameter whose value indicates the condition status returned by the most recent c-treeACE SQL statement. SQLSTATE is specified by the SQL-92 standard as a replacement for the SQLCODE status parameter (which was part of SQL-89). SQLSTATE defines many more specific error conditions than SQLCODE, which allows applications to implement more portable error handling. Applications declare either SQLSTATE or SQLCODE, or both. c-treeACE SQL returns the status to SQLSTATE or SQLCODE after execution of each c-treeACE SQL statement.

**storage environment**

The combination of storage systems which have implemented the storage interfaces. One possible combination of storage systems in an implementation is the flat-file and main-memory storage system, with a proprietary database containing user data.

**stored procedure**

A snippet of Java source code embedded in an CREATE PROCEDURE statement. The source code can use all standard Java features as well as use c-treeACE SQL supplied Java classes for processing any number of c-treeACE SQL statements.

**system catalog**

Tables created by c-treeACE SQL that store information about tables, columns, and indexes that make up the database. c-treeACE SQL creates and manages the system catalog.

**system tables**

Another term for system catalog.

**tid**

Another term for tuple identifier.

**transaction**
A group of operations whose changes can be made permanent or undone only as a unit to protect against data corruption.

**trigger**
A special type of stored procedure that helps insure referential integrity for a database. Like stored procedures, triggers also contain Java source code (embedded in a `CREATE TRIGGER` statement) and use c-treeACE SQL Java classes. However, triggers are automatically invoked (“fired”) by certain c-treeACE SQL operations (an insert, update, or delete operation) on the trigger’s target table.

**trigger action time**
The `BEFORE` or `AFTER` keywords in a `CREATE TRIGGER` statement. The trigger action time specifies whether the actions implemented by the trigger execute before or after the triggering `INSERT`, `UPDATE`, or `DELETE` statement.

**trigger event**
The statement that causes a trigger to execute. Trigger events can be c-treeACE SQL `INSERT`, `UPDATE`, or `DELETE` statements that affect the table for which a trigger is defined.

**triggered action**
The Java code within the `BEGIN END` clause of a `CREATE TRIGGER` statement. The code implements actions to be completed when a triggering statement specifies the target table.

**tuple identifier**
A unique identifier for a tuple (row) in a table. A tuple identifier for the tuple that was inserted is returned after an insert operation. c-treeACE SQL passes a tuple identifier to the delete, update, and fetch stubs to indicate which tuple is affected. The c-treeACE SQL scalar function `ROWID` and related functions return tuple identifiers to applications.

**unicode**
A superset of the ASCII character set that uses two bytes for each character rather than ASCII’s 7-bit representation. Able to handle 65,536 character combinations instead of ASCII’s 128, Unicode includes alphabets for many of the world’s languages. The first 128 codes of Unicode are identical to ASCII, with a second-byte value of zero.

**unique key**
A column or columns in a table whose value (or combination of values) must be unique. Use the `UNIQUE` clause of the SQL `CREATE TABLE` statement to create unique keys. Unique keys are also called candidate keys.

**URL**
In general, a Universal Resource Locator used to specify protocols and locations of items on the Internet. In JDBC, a database connection string in the form `jdbc:subprotocol:subname`. The c-treeACE SQL JDBC Driver format for database URLs is `jdbc:ctree:T:host_name:db_name`.

**utility class**
A set of utility functions that a storage manager uses to assemble and disassemble data elements passed through the storage interfaces.
**view**
A virtual table that recreates the result table specified by a `SELECT` statement. No data is stored in a view, but other queries can refer to it as if it were a table containing data corresponding to the result table it specifies.

**virtual machine**
The Java specification for a hardware-independent and portable language environment. Java language compilers generate code that can execute on a virtual machine. Implementations of the Java virtual machine for specific hardware and software platforms allow the same compiled code to execute without modification.

**virtual table**
A table of values that is not physically stored in a database, but instead derived from columns and rows of other tables. c-treeACE SQL generates virtual tables in its processing of query expressions: the `FROM`, `WHERE`, `GROUP BY` and `HAVING` clauses each generate a virtual table based on their input.
8. Index

0
0, 10000 Series ................................................. 257

1
11000 Series - c-treeACE SQL RDS Errors ........... 258
15000 Series - c-treeACE SQL Flat-File Storage
System .................................................. 259
16000 Series - Latte and Polka Subsystems .......... 262
160xx - Main Memory Storage System ............... 260
170xx ................................................. 264
171xx ................................................. 268
173xx ................................................. 272
174xx ................................................. 272
175xx ................................................. 276
177xx ................................................. 279
178xx ................................................. 283
179xx ................................................. 288
18000 Series ............................................. 291

2
20000 Series - c-treeACE SQL ............................ 293
-202xx - Sequence Support ................................ 292
21000 Series - c-treeDB FSS DB Layer ............... 304
22000 Series - c-treeDB FSS DB Layer ............... 310
23000 Series - c-treeACE SQL Data Exceptions ..... 315
250xx - c-treeACE SQL ODBC Integrator .......... 315
251xx - c-treeACE SQL ODBC Driver ............... 318
26000 Series - c-treeACE SQL JDBC Driver .......... 320

3
30000 Series - c-treeACE SQL Network ................. 320

4
40000 Series - c-treeACE SQL Environment
Errors ................................................. 323

5
50000 Series - c-treeACE SQL DHRSS Errors ....... 323

A
ABS function (ODBC compatible) ...................... 50
ABS scalar function .................................. 50
ACOS function (ODBC compatible) .................. 51
ACOS scalar function ................................ 51
ADD_MONTHS function (extension) ................. 52
ADD_MONTHS scalar function ........................ 52
Aggregate functions
AVG ...................................................... 44
COUNT .................................................. 45
MAX ..................................................... 47
MIN ..................................................... 48
SUM ..................................................... 49
Aggregate Functions .................................. 43

Aliases ..................................................... 20
ALTER INDEX .......................................... 176
ALTER INDEX statement .............................. 176
ALTER SEQUENCE .................................. 177
ALTER TABLE ......................................... 179
ALTER TABLE statement ............................. 179
ALTER VIEW .......................................... 182
Approximate numeric data type ...................... 13
Approximate Numeric Data Types ................... 13
ASCII function (ODBC compatible) ................. 53
ASCII scalar function ................................ 53
ASIN function (ODBC compatible) .................. 54
ASIN scalar function ................................ 54
ATAN function (ODBC compatible) ................. 55
ATAN scalar function ................................ 55
ATAN2 function (ODBC compatible) ............... 56
ATAN2 scalar function ................................ 56
AVG ...................................................... 44
AVG aggregate function .............................. 44

B
Basic predicate ........................................... 17
Basic Predicate ......................................... 17
Between predicate ....................................... 18
BETWEEN Predicate ..................................... 18
BIGINT data type ....................................... 11
BINARY data type ...................................... 14
BIT data type ........................................... 14
Bit String Data Types ................................ 14
BLOB ....................................................... 14

C
CALL ....................................................... 184
CALL statement ......................................... 184
Cartesian product ....................................... 37
CASE (SQL-92 Compatible) ........................... 57
CASE scalar function .................................. 57
CAST function (SQL-92 compatible) ............... 60
CAST scalar function .................................. 60
CEILING function (ODBC compatible) ............. 61
CEILING scalar function .............................. 61
CHAR function (ODBC compatible) ................. 62
CHAR scalar function .................................. 62
Character Data Types ................................ 10
Character expressions
concatenated ........................................... 22
Character string literals .............................. 27
Character String Literals ............................. 27
CHARTOROWID (extension) ......................... 63
CHARTOROWID scalar function ..................... 63
CHR function (extension) ............................. 64
CHR scalar function .................................. 64
CLOB ....................................................... 10
COALESCE (SQL-92 compatible) .................... 65
COALESCE scalar function ............................ 65
Column Constraints ................................... 201
Identifiers

HOUR function (ODBC compatible) ........................................... 93
HOUR scalar function .............................................................. 93

I
Identifiers ................................................................. 8
  conventional ............................................................ 9
  delimited .............................................................. 9
  SQL ................................................................. 8

J
Joining tables
  Cartesian product .................................................... 37
  equi-joins .................................................................. 37
  inner joins ........................................................... 37
  outer joins .......................................................... 39
  self joins ............................................................ 37

L
LAST_DAY function (extension) ........................................... 100
LAST_DAY scalar function ................................................. 100
LAST_IDENT function (extension) ..................................... 101
LCASE function (ODBC compatible) .................................. 102
LCASE scalar function .................................................... 102
LEAST function (extension) ............................................. 103
LEAST scalar function .................................................... 103
LEFT function (ODBC compatible) .................................... 104
LEFT scalar function ...................................................... 104
LEN function (ODBC compatible) ...................................... 105
LENGTH function (ODBC compatible) ............................... 106
LENGTH scalar function .................................................. 106
Like predicate ............................................................. 18
LIKE Predicate ............................................................ 18
Literals ........................................................................ 26
  character string ....................................................... 27
  date-time ................................................................ 27
  numeric ................................................................... 27
  timestamp ............................................................... 30
LOCALTIME SQL Function .................................................. 108
LOCALTIMESTAMP SQL Function .................................... 109
LOCATE function (ODBC compatible) ............................... 107
LOCATE scalar function .................................................. 107
LOCK TABLE ............................................................. 235
LOG SQL Scalar Function .................................................. 110
LOG10 function (ODBC compatible) ................................. 111
LOG10 scalar function ..................................................... 111
Logical operators ......................................................... 16
Logical Operators ..........................................................
REVOKE .........................................................239
REVOKE statement ........................................239
RIGHT function (ODBC compatible) ..................136
RIGHT scalar function ..................................136
ROLLBACK WORK ........................................241
ROLLBACK WORK statement .............................241
ROUND function ........................................137
ROWID (extension) .......................................138
ROWID scalar function ................................138
ROWIDTOCHAR (extension) ..............................139
ROWIDTOCHAR scalar function .........................139
RPAD function (extension) ...............................140
RPAD scalar function ....................................140
RTRIM function (ODBC compatible) ..................141
RTRIM scalar function ..................................141

S
Scalar functions ..........................................50
ABS ........................................................50
ACOS ....................................................51
ADD_MONTHS ..........................................52
ASCII ....................................................53
ASIN .....................................................54
ATAN ....................................................55
ATAN2 ..................................................56
CASE ....................................................57
CAST ....................................................60
CEILING ...............................................61
CHAR .....................................................62
CHARTOROWID (extension) ............................63
CHR ........................................................64
COALESCE ..............................................65
CONCAT ................................................66
CONVERT ...............................................67
COS ........................................................69
CURDATE ...............................................71
CURTIME ...............................................75
DATABASE ..............................................76
DAYNAME ..............................................80
DAYOFMONTH ..........................................81
DAYOFWEEK ...........................................82
DAYOFYEAR ..........................................83
DB_NAME ..............................................84
DECODE ...............................................85
DEGREES ...............................................86
DIFFERENCE ..........................................87
EXP ......................................................88
FLOOR ....................................................90
GREATEST .............................................92
HOUR .....................................................93
IFNULL ................................................94
INITCAP ...............................................95
INSERT ...............................................96
INSTR ..................................................97
LAST_DAY .............................................100
LCASE ...................................................102

LEAST ....................................................103
LEFT .....................................................104
LENGTH .................................................106
LOCATE ...............................................107
LOG10 ..................................................111
LOWER ..................................................112
LPAD ....................................................113
LTRIM ....................................................114
MINUTE ..................................................115
MOD .....................................................116
MONTH ...................................................118
MONTHNAME .........................................117
MONTHS_BETWEEN ....................................119
NEXT_DAY .............................................120
NOW .....................................................121
NULLIF ...................................................122
NVL .......................................................123
OBJECT_ID ...........................................124
PI 127 ....................................................129
POWER ..................................................129
PREFIX ..................................................130
QUARTER ..............................................131
RADIANS ..............................................132
RAND ....................................................133
REPLACE ..............................................135
RIGHT ...................................................136
ROWID ...................................................138
ROWIDTOCHAR .......................................139
RPAD ....................................................140
RTRIM ....................................................141
SECOND ................................................142
SIGN .....................................................144
SIN .......................................................145
SOUNDEX .............................................146
SPACE ..................................................147
SQRT ....................................................148
SUBSTR ................................................149
SUBSTRING ..........................................151
SUFFIX ..................................................152
SUSER_NAME ..........................................153
SYSDATE ...............................................154
SYSTIME ..............................................155
SYSTIMESTAMP ......................................156
TAN ......................................................157
TIMESTAMP ...........................................158
TIMESTAMPADD .......................................158
TIMESTAMPDIFF ......................................160
TO_CHAR ..............................................161
TO_DATE ..............................................162
TO_NUMBER ..........................................163
TO_TIME ..............................................164
TO_TIMESTAMP ......................................165
TRANSLATE ..........................................166
UCASE ..................................................169
UPPER ..................................................170
USER .....................................................171
USER_NAME ..........................................172
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO_DATE scalar function</td>
<td>162</td>
</tr>
<tr>
<td>TO_NUMBER function (extension)</td>
<td>163</td>
</tr>
<tr>
<td>TO_NUMBER scalar function</td>
<td>163</td>
</tr>
<tr>
<td>TO_TIME function (extension)</td>
<td>164</td>
</tr>
<tr>
<td>TO_TIME scalar function</td>
<td>164</td>
</tr>
<tr>
<td>TO_TIMESTAMP function (extension)</td>
<td>165</td>
</tr>
<tr>
<td>TO_TIMESTAMP scalar function</td>
<td>165</td>
</tr>
<tr>
<td>TRANSLATE function (extension)</td>
<td>166</td>
</tr>
<tr>
<td>TRANSLATE scalar function</td>
<td>166</td>
</tr>
<tr>
<td>TRIM SQL Function</td>
<td>167</td>
</tr>
<tr>
<td>TRUNCATE function</td>
<td>168</td>
</tr>
<tr>
<td>TRUNCATE TABLE</td>
<td>250</td>
</tr>
<tr>
<td>UCASE function (ODBC compatible)</td>
<td>169</td>
</tr>
<tr>
<td>UCASE scalar function</td>
<td>169</td>
</tr>
<tr>
<td>UPDATE</td>
<td>251</td>
</tr>
<tr>
<td>UPDATE statement</td>
<td>251</td>
</tr>
<tr>
<td>UPDATE STATISTICS</td>
<td>253</td>
</tr>
<tr>
<td>UPDATE STATISTICS statement</td>
<td>253</td>
</tr>
<tr>
<td>UPPER function (SQL-92 compatible)</td>
<td>170</td>
</tr>
<tr>
<td>UPPER scalar function</td>
<td>170</td>
</tr>
<tr>
<td>User Defined Scalar Function</td>
<td>188, 217</td>
</tr>
<tr>
<td>USER function (ODBC compatible)</td>
<td>171</td>
</tr>
<tr>
<td>USER scalar function</td>
<td>171</td>
</tr>
<tr>
<td>USER_NAME function (extension)</td>
<td>172</td>
</tr>
<tr>
<td>USER_NAME scalar function</td>
<td>172</td>
</tr>
<tr>
<td>VARBINARY data type</td>
<td>14</td>
</tr>
<tr>
<td>WEEK function (ODBC compatible)</td>
<td>173</td>
</tr>
<tr>
<td>WEEK scalar function</td>
<td>173</td>
</tr>
<tr>
<td>YEAR function (ODBC compatible)</td>
<td>174</td>
</tr>
<tr>
<td>YEAR scalar function</td>
<td>174</td>
</tr>
</tbody>
</table>