Orion SQL Syntax Help Index

This index lists the help topics available for the SQL syntax. Using the keyboard, tab to select the underlined topic you want to view, then press enter. Using the mouse, point to the underlined topic you want to view, and click the left mouse button. Use the scroll bar to see entries not currently visible in the help window.

To learn how to use help, press F1.

Overview

General Information

Structured Query Language (SQL) is the database management language declared by the American National Standards Institute (ANSI) as a standard.   SQL is the *lingua franca* of the computer database world.   SQL is used on mainframes, minicomputers and PCs.

Notational Conventions

While uppercase is used throughout this document to indicate SQL keywords (COMMIT, SELECT, etc), in actuality the Orion Database Administrator will accept keywords in either upper or lowercase.   The keyword 'COMMIT' may be spelled 'commit', 'Commit', 'COMMIT', etc.

The names of users, tables, indices and columns, and the content of character fields are, however, case sensitive.   If you used some combination of upper and lower case to create them, you have to use the same combination to access them.   All system names were created using upper case only.

Optional elements are indicated by color and enclosing them within square brackets as in **[option]**.   When a choice may be made among a list of possible optional elements, individual selections are separated by vertical bars as in **[option1 | option2 | option3]**.

When a pattern may be repeated, an ellipsis follows the pattern as in "ColumnName1 **[, ColumnName2] ...**".   This example implies a list of one or more column names separated by commas.

Statements

SQL is based on various uses of the following statements:

COMMIT WORK	Instructs the database administrator to make all changes to the database by the current transaction permanent.
CREATE INDEX	Creates an index on a database table.
CREATE TABLE	Creates a database table.
DELETE	Deletes records from a database table.
DROP INDEX	Deletes an index from a database table.
DROP TABLE	Deletes a database table.
INSERT	Creates records in a database table.
ROLLBACK WORK	Instructs the database administrator to forget all changes made to the database by the current transaction.
SELECT	Retrieves data from database tables.
UPDATE	Modifies the content of records within database tables.

Statement: COMMIT WORK

Syntax	**COMMIT WORK;**
	Changes made to the database are not made permanent until this statement is executed. Should the current user execute a **ROLLBACK WORK** or unexpectedly log off, all changes made by the user since logging in or executing a **COMMIT WORK** (whichever occurred most recently) will vanish.
	During database recovery only committed transactions are restored.
Comments	As queries are executed within a transaction and records are created, read, updated or deleted; the database applies various kinds of locks on the applicable records. These locks are released when the transaction is committed or rolled back. In order to minimize the conflict between transactions, be sure to minimize the amount of time these locks are in place by issuing a **COMMIT WORK** or **ROLLBACK WORK** whenever possible.
	Committing a transaction automatically starts a new transaction. **COMMIT WORK** is very fast since the changes have already been made, they are merely flagged as permanent.

Statement: CREATE INDEX

Syntax

```
CREATE [UNIQUE] INDEX
    [CreatorName.]IndexName
    ON [UserName.]TableName (
        ColumnName1 [ASC | DESC][,
        ColumnName2 [ASC | DESC]]
        ...
    );
```

This statement creates the index **IndexName** on the table **TableName**. While indices are never referenced explicitly in SQL (other than creating and dropping them), they are used extensively by the Orion Database Administrator to maximize system performance.

Phrase	Description
CREATE [UNIQUE] INDEX	Specify **UNIQUE** if you wish key values to be distinguishable, one from another, across the entire table upon which the index is constructed. If an attempt is made to create a record in the table which violates this unique constraint, an error condition will arise.
[CreatorName.]IndexName	**IndexName** becomes the name of the newly created index. Specify **CreatorName** if you wish the index to belong to a user different from the current user.
ON [UserName.]TableName	**TableName** identifies the table upon which the index is to be constructed. Specify **UserName** if the table belongs to a user different from the current user.
ColumnName1 [ASC | DESC]	**ColumnName** identifies a column in the table to be included in the index key. Specify **ASC** for ascending and **DESC** for descending. **ASC** is default. We recommend against the use of **DESC**, it is included in order to conform to the ANSI standard.
[, ColumnName2 [ASC | DESC]] ...	Additional columns may be included within the index key by creating a list of columns separated by commas. The index key will be constructed in the order in which columns appear in this list.

Comments

While indices may be created at anytime, we recommend that you create indices immediately after you create their base table; otherwise, CREATE INDEX has to read and rewrite all data which is already in the table.

The presence of suitable indices may greatly enhance system performance. Indices do, however, cause a moderate increase in the amount of time it takes to write a record to the table upon which the index is constructed. The database user should add indices judiciously.

ColumnName, **CreatorName**, **IndexName, TableName** and **UserName** are limited to 32 characters.

Statement: CREATE TABLE

Syntax

```
CREATE TABLE [UserName.]TableName (
        ColumnName1 DataType1 [NOT NULL [UNIQUE]][,
        ColumnName2 DataType2 [NOT NULL [UNIQUE]]]
        ...
        [UNIQUE (ColumnNameA[, ColumnNameB] ... )][,
        UNIQUE (ColumnNameA[, ColumnNameB] ... )]
        ...
    );
```

This statement creates the table **TableName** with the columns as specified.

Phrase	Description
CREATE TABLE [UserName.]TableName	**TableName** becomes the name of the newly created table. Specify **UserName** if you wish the table to belong to a user different from the current user.
ColumnName1 DataType1 [NOT NULL [UNIQUE]]	**ColumnName** identifies a column to be included in the table. **DataType** indicates the data type of the column. When a column is specified as **NOT NULL**, any attempt to insert or update a record which would result in a NULL value in this column will cause an error condition to arise. When a column is specified as **UNIQUE**, an UNIQUE index is automatically created for that column. This index will insure that all values for that column are distinguishable, one from another, across the entire table. Any attempt to insert or update a record which violates this unique constraint will cause an error condition to arise.
[, ColumnName2 DataType2 [NOT NULL UNIQUE]]] ...	Additional columns may be included within the table by creating a list of column definitions separated by commas.
[UNIQUE (ColumnNameA[, ColumnNameB] ...)]	The combination of **ColumnNameA, ColumnNameB, ...** are to be unique across all records in the table. An UNIQUE index which includes the named columns is automatically created. This index will insure that key values for these columns are distinguishable, one from another, across the entire table. Any attempt to insert or update a record which violates this unique constraint will cause an error condition to arise.
[, UNIQUE (ColumnNameN[, ColumnNameO] ...)]	Additional unique constraints may be added by creating a list of unique constraint definitions separated by commas. Each unique constraint is maintained by a separate UNIQUE index.

Comments **ColumnName**, **TableName** and **UserName** are limited to 32 characters.

Statement: DELETE

Syntax

DELETE FROM [UserName.]TableName
[WHERE SearchCondition];

This statement deletes records from the table **TableName**. **SearchCondition** specifies which records are to be deleted.

Phrase	Description
DELETE FROM [UserName.]TableName	**TableName** is the name of the table containing the records to be deleted. Specify **UserName** if the table belongs to a user different from the current user.
WHERE SearchCondition	**SearchCondition** specifies which records are to be deleted.

Comments

TableName and **UserName** are limited to 32 characters.

Statement: DROP INDEX

Syntax **DROP INDEX [CreatorName.]IndexName**
ON [UserName.]TableName;

This statement deletes the index **IndexName** on the table **TableName**.

Phrase	Description
DROP INDEX [CreatorName.]IndexName	
	IndexName is the name of the index to be deleted. Specify **CreatorName** if the index belongs to a user different from the current user.
ON [UserName.]TableName	
	TableName identifies the table upon which the index exists. Specify **UserName** if the table belongs to a user different from the current user.

Comments While indices may be dropped at anytime, dropping an index on a table causes the entire table to be rebuilt.

When a table is deleted, all indices associated with that table are automatically dropped.

CreatorName, IndexName, TableName and **UserName** are limited to 32 characters

Statement: DROP TABLE

Syntax	**DROP TABLE [UserName.]TableName;**
	This statement deletes the table **TableName**.

Phrase	Description
DROP TABLE [UserName.]TableName	
	TableName is the name of the table to be deleted. Specify **UserName** if the table belongs to a user different from the current user.

Comments	All indices associated with the table are automatically dropped.
	TableName and **UserName** are limited to 32 characters

Statement: INSERT

Syntax **INSERT INTO [UserName.]TableName**
[(ColumnName1[, ColumnName2] ...)]
VALUES (Value1[, Value2] ...);

-- or --

Syntax **INSERT INTO [UserName.]TableName**
[(ColumnName1[, ColumnName2] ...)]
SelectStatement;

This statement inserts records into the table TableName.

Phrase	Description
INSERT INTO [UserName.]TableName	**TableName** is the name of the table into which records are to be inserted. Specify **UserName** if the table belongs to a user different from the current user.
(ColumnName1[, ColumnName2] ...)	Identifies columns into which data is to be deposited. If any columns exist in the table which are not listed, they are set to the NULL value. A column name may not be repeated. Column names do not have to be in the same order as they are in the table itself.
VALUES (Value1[, Value2] ...)	Specifies the values to be deposited in the record. If a list of columns was supplied, there must be a match between the number of column names in the list and the number of values supplied. If a list of columns was not supplied, there must be a match between the total number of columns in the table and the number of values supplied. The data type of the value must be compatible with the data type of the column into which it is to be deposited. When the column allows the NULL value, the value may be 'NULL'.
SelectStatement	The **SelectStatement** generates a set of records to be inserted. If a list of columns was supplied, there must be a match between the number of column names in the list and the number of columns generated by the **SelectStatement**. If a list of columns was not supplied, there must be a match between the total number of columns in the table and the number of columns generated by the **SelectStatement**. The data types of the columns generated by the **SelectStatement** must be compatible with the data types of the columns into which they are to be deposited.

Comments **ColumnName**, **TableName** and **UserName** are limited to 32 characters.

Statement: ROLLBACK WORK

Syntax **ROLLBACK WORK;**

Changes made to the database are not made permanent until a **COMMIT WORK** statement is executed. **ROLLBACK WORK** instructs the database to purge all changes to the database by the current user since logging in or executing a **COMMIT WORK** (whichever occurred most recently).

Comments As queries are executed within a transaction and records are created, read, updated or deleted; the database applies various kinds of locks on the applicable records. These locks are released when the transaction is committed or rolled back. In order to minimize the conflict between transactions, be sure to minimize the amount of time these locks are in place by issuing a **COMMIT WORK** or **ROLLBACK WORK** whenever possible.

Rolling a transaction back automatically starts a new transaction. **ROLLBACK WORK** may take some time while the database administrator purges updates.

Statement: SELECT

Syntax

```
SELECT [ALL | DISTINCT]
    Expression1[, Expression2] ...
    FROM [UserName1.]TableName1 [CorrelationName1][,
        [UserName2.]TableName2 [CorrelationName2]]
        ...
    [WHERE SearchCondition]
    [GROUP BY ColumnSpecification1[,
        ColumnSpecification2]
        ...
        [HAVING SearchCondition]]
    [ORDER BY ColumnSpecificationA [ASC | DESC][,
        ColumnSpecificationB [ASC | DESC]]
        ...];
```

This statement generates a result table. There is one column in the result table for each **Expression** in the expression list of the SELECT statement. The values deposited in the columns of the result table are generated by evaluating the corresponding **Expression**.

Phrase	Description
SELECT [ALL | DISTINCT]	**ALL** is default. **DISTINCT** insures that all records in the result table are distinguishable, one from another. When **DISTINCT** is specified, duplicate records are eliminated from the result table.
Expression1[, Expression2]	This list specifies the values to be inserted into the columns of the result table. The data types of the columns of the result table are determined by the data types of the values resulting from the expressions in this list. You may substitute a single '*' in place of the list of expressions. '*' implies a list of all columns of all tables identified in the **FROM** clause. You may not use '*' if you use the **GROUP BY** clause.
FROM [UserName1.]TableName1 [CorrelationName1]	**TableName** is the name of the table from which records are to be read. Specify **UserName** if the table belongs to a user different from the current user. **CorrelationName** is effectively an alias for the **TableName** which it follows.
[, [UserName2.]TableName2 [CorrelationName2]] ...	Additional tables may be included by creating a list of tables separated by commas. This effectively creates a Cartesian product of all the tables in the list.
WHERE SearchCondition	**SearchCondition** specifies which records are to be read.
GROUP BY ColumnSpecification1	Rearranges the table(s) identified by the **FROM** clause into groups such that within any one group all rows have the same value for the **GROUP BY** column(s). The **SELECT** clause is then applied to these groups. Each group generates a single record in the result table. Please refer to a text book for a description of the "grouped

table".   "Grouped tables" are fully supported by the Orion Database Administrator.

[, ColumnSpecification2] ...

Additional columns may be included within the **GROUP BY** clause by creating a list of columns separated by commas.

Please refer to a text book for a description of the "grouped table".   "Grouped tables" are fully supported by the Orion Database Administrator.

HAVING SearchCondition

Specifies a restriction on the grouped table resulting from the GROUP BY clause by eliminating groups not meeting the **SearchCondition**.

Please refer to a text book for a description of the "grouped table".   "Grouped tables" are fully supported by the Orion Database Administrator.

ORDER BY ColumnSpecificationA [ASC | DESC]

Records in the result table will be sorted on the basis of the data in the columns specified by the **ORDER BY** clause.   Specify **ASC** for ascending and **DESC** for descending.   **ASC** is default.

ColumnSpecification must identify one of the **Expressions** within the list of expressions of the **SELECT STATEMENT**.   An integer may be used in place of the **ColumnSpecification**; when an integer is used it identifies which column in the result table is to be used to determine the order.

[, ColumnSpecificationB [ASC | DESC]] ...

Additional ordering **ColumnSpecifications** (or integers) may be included within the ORDER BY clause by creating a list of columns (or integers) separated by commas.

Comments **CorrelationName, TableName** and **UserName** are limited to 32 characters.

Statement: UPDATE

Syntax

```
UPDATE [UserName.]TableName
    SET ColumnName1 = Expression1[,
        ColumnName2 = Expression2]
        ...
    [WHERE SearchCondition];
```

This statement modifies records in the table **TableName**. **SearchCondition** specifies which records are to be modified.

Phrase	Description
UPDATE [UserName.]TableName	**TableName** is the name of the table in which records are to be modified. Specify **UserName** if the table belongs to a user different from the current user.
SET ColumnName1 = Expression1	**Expression** is evaluated and the result placed in the column identified by **ColumnName**. Columns not specifically identified are left unaffected.
[, ColumnName2 = Expression2] ...	Additional columns may be modified by creating a list of **ColumnNames** and **Expressions** separated by commas. A column name may not be repeated.
[WHERE SearchCondition]	**SearchCondition** specifies which records are to be modified.

Comments **ColumnName, TableName** and **UserName** are limited to 32 characters.

Data types

Data types are organized into three basic categories:

Approximate Numeric	This type is typically referred to as *floating point*.
Exact Numeric	This type is typically referred to as *fixed point*.
Character String	This type is used to store text.

Data Types: Approximate Numeric

This type is typically referred to as *floating point*.

Data type	Description
FLOAT	Floating point number with magnitude ranging from approximately 1.7976931348623158e+308 to 2.2250738585072014e-308.
FLOAT(p)	Floating point number with **p** significant digits, with magnitude ranging from approximately 1.7976931348623158e+308 to 2.2250738585072014e-308.
REAL	Floating point number with magnitude ranging from approximately 1.7976931348623158e+308 to 2.2250738585072014e-308.
DOUBLE PRECISION	Floating point number with magnitude ranging from approximately 1.7976931348623158e+308 to 2.2250738585072014e-308.

Data Types: Exact Numeric

This type is typically referred to as *fixed point*.

Data type	Description
DEC	Signed decimal number with up to 19 digits of which 0 appear to the right of the decimal point.
DEC(p)	Signed decimal number with up to **p** digits of which 0 appear to the right of the decimal point; 1 <= **p** <= 19.
DEC(p, s)	Signed decimal number with up to **p** digits of which **s** appear to the right of the decimal point; 1 <= **p** <= 19 and 0 <= **s** <= **p**.
DECIMAL	Signed decimal number with up to 19 digits of which 0 appear to the right of the decimal point.
DECIMAL(p)	Signed decimal number with up to **p** digits of which 0 appear to the right of the decimal point; 1 <= **p** <= 19.
DECIMAL(p, s)	Signed decimal number with up to **p** digits of which **s** appear to the right of the decimal point; 1 <= **p** <= 19 and 0 <= **s** <= **p**.
INT	Whole number ranging from -2147483647 to 2147483647.
INTEGER	Whole number ranging from -2147483647 to 2147483647.
NUMERIC	Signed decimal number with up to 19 digits of which 0 appear to the right of the decimal point.
NUMERIC(p)	Signed decimal number with up to **p** digits of which 0 appear to the right of the decimal point; 1 <= **p** <= 19.
NUMERIC(p, s)	Signed decimal number with up to **p** digits of which **s** appear to the right of the decimal point; 1 <= **p** <= 19 and 0 <= **s** <= **p**.
SMALLINT	Whole number ranging from -2147483647 to 2147483647.

Data Types: Character String

This type is used to store text.

Data type	Description
CHAR	Character data, length assumed to be 1.
CHAR(n)	Character data, length specified by n where 1 <= n <= 2047.
CHARACTER	Character data, length assumed to be 1.
CHARACTER(n)	Character data, length specified by n where 1 <= n <= 2047.
VARCHAR	Character data, variable length where maximum length is 2047.
VARCHAR(n)	Character data, variable length where maximum length is specified by n where 1 <= n <= 2047.

Operators

The arithmetic operators have their usual meanings:

+	The value on the right is added to the value on the left.
-	The value on the right is subtracted from the value on the left.
*****	The value on the right is multiplied by the value on the left.
/	The value on the left is divided by the value on the right.

Built-in functions

Built-in functions act on several rows in a table together. Built-in functions may not be nested. SQL supports the following built-in functions:

AVG(Expression)	For each record selected, **Expression** is analyzed and a value obtained. **AVG** returns the average of these values. Only values which are not NULL are included.
COUNT(DISTINCT Expression)	For each record selected, **Expression** is analyzed and a value obtained. **COUNT(DISTINCT)** returns the number of these values which are distinguishable, one from another. Only values which are not NULL are included.
COUNT(*)	**COUNT(*)** returns the number of records selected.
MAX(Expression)	For each record selected, **Expression** is analyzed and a value obtained. **MAX** returns the maximum of these values. Only values which are not NULL are included.
MIN(Expression)	For each record selected, **Expression** is analyzed and a value obtained. **MIN** returns the minimum of these values. Only values which are not NULL are included.
SUM(Expression)	For each record selected, **Expression** is analyzed and a value obtained. **SUM** returns the sum of these values. Only values which are not NULL are included.

Expressions

Expressions can be:

- A column name
- A constant or literal value
- A built-in function
- An arithmetic combination of expressions

Constants can be:

- Integer (for example: 100, -5, +127)
- Decimal (for example: 100.0, -.001, 1., +1.5)
- Floating point (for example: 1E10, -2E-7, +3.14159E0)
- Character string (for example: 'SMITH' '-@k9-22', '-1', 'Orion')

Order of execution:

- Arithmetic expressions are evaluated before comparisons and logical operations.
- Arithmetic expressions are evaluated left to right except that multiplication and division are performed before addition and subtraction. Parentheses can be used to control the order of evaluation.

Search Conditions

A search condition can be a simple condition or a logical combination of conditions.   If the value of any expression is NULL then the condition evaluates to UNKNOWN:

Simple conditions:

Expression1 = Expression2

Evaluates to TRUE if and only if **Expression1** has a value equal to that of **Expression2**, otherwise the condition evaluates to FALSE.

Expression1 < Expression2

Evaluates to TRUE if and only if **Expression1** has a value less than that of **Expression2**, otherwise the condition evaluates to FALSE.

Expression1 <= Expression2

Evaluates to TRUE if and only if **Expression1** has a value less than or equal to that of **Expression2**, otherwise the condition evaluates to FALSE.

Expression1 > Expression2

Evaluates to TRUE if and only if **Expression1** has a value greater than that of **Expression2**, otherwise the condition evaluates to FALSE.

Expression1 >= Expression2

Evaluates to TRUE if and only if **Expression1** has a value greater than or equal to that of **Expression2**, otherwise the condition evaluates to FALSE.

Expression1 <> Expression2

Evaluates to TRUE if and only if **Expression1** has a value which is not equal to that of **Expression2**, otherwise the condition evaluates to FALSE.

Expression1 [NOT] BETWEEN Expression2 AND Expression3

Same as [NOT] ((**Expression2** <= **Expression1**) AND (**Expression1** <= **Expression3**).

Expression1 [NOT] IN (Value1[, Value2] ...)

Same as [NOT] ((**Expression1** = **Value1**)**[OR (Expression1 = Value2)] ...** .

Expression1 [NOT] IN (Subquery)

TRUE if **Expression1** is [not] equal to any value returned by **Subquery**.

ColumnName [NOT] LIKE Pattern

Only available for character types: [not] TRUE if the string in the specified column matches Pattern.   In **Pattern**, '_' matches any single character, '%' matches any character sequence.

ColumnName IS [NOT] NULL

True if the value of ColumnName is [not] NULL.

[NOT] EXISTS (Subquery)

[Not] TRUE if **Subquery** returns at least one record.

Expression1 [NOT] IN (Subquery)

[Not] TRUE if **Subquery** returns at least one value which is equal to **Expression1**.

Expression1 = [ANY | ALL | SOME] (Subquery)

Expression1 < [ANY | ALL | SOME] (Subquery)
Expression1 <= [ANY | ALL | SOME] (Subquery)
Expression1 > [ANY | ALL | SOME] (Subquery)
Expression1 >= [ANY | ALL | SOME] (Subquery)
Expression1 <> [ANY | ALL | SOME] (Subquery)

Please refer to a text book for a description of the "quantified" predicate. While supported, we recommend against its use.

Logical combination of conditions:

NOT Condition

Evaluates to TRUE if and only if **Condition** is FALSE. Evaluates to FALSE if and only if **Condition** is TRUE.

Condition1 AND Condition2

Evaluates to TRUE if and only if both **Condition1** and **Condition2** are TRUE.

Condition1 OR Condition2

Evaluates to TRUE if either **Condition1** or **Condition2** is TRUE or both are TRUE. Evaluates to TRUE even if one condition is UNKNOWN.

System Catalog

The system catalog is composed of four tables: SYSTEM.COLUMNS, SYSTEM.INDICES, SYSTEM.TABLES and SYSTEM.USERS. All these tables belong to the user "SYSTEM". These tables are maintained by the Orion Database Administrator.

Unless otherwise indicated, you must not modify the SYSTEM.COLUMNS, SYSTEM.INDICES, SYSTEM.TABLES and SYSTEM.USERS tables.

COLUMNS	This table contains information about all columns of all tables in the database.
INDICES	This table contains information about all indices in the database. When more than one column is included in an index, there is a separate record for each column included.
TABLES	This table contains information about all tables in the database.
USERS	This table contains information about all users known to the database.

System Catalog: SYSTEM.COLUMNS

This table contains information about all columns of all tables in the database.   You may only modify the REMARK field of records in this table.

Column	Type/Description
SEQUENCE_NUMBER	**SMALLINT** Indicates the position of the column within the table.
LENGTH	**SMALLINT** Indicates the length of character strings or the precision of numeric values.
SCALE	**SMALLINT** Indicates the scale of numeric values.
DATA_TYPE	**VARCHAR(32)** Indicates **data type**.
NOT_NULL	**CHARACTER(5)** Indicates whether NULL values are allowed. TRUE indicates NULL is not allowed.
USER_NAME	**VARCHAR(32)** Indicates the user name of the owner of the table.
TABLE_NAME	**VARCHAR(32)** Indicates the name of the table.
COLUMN_NAME	**VARCHAR(32)** Indicates the name of the column.
REMARK	**VARCHAR(128)** A comment.

System Catalog SYSTEM.INDICES

This table contains information about all indices in the database. When more than one column is included in an index, there is a separate record for each column included. You may only modify the REMARK field of records in this table.

The INDEX_NAME of indices created to enforce unique constraints have a '~' as their first character.

Column	Description
USER_NAME	**VARCHAR(32)** Indicates the user name of the owner of the table upon which the index is constructed.
TABLE_NAME	**VARCHAR(32)** Indicates the name of the table upon which the index is constructed.
INDEX_NAME	**VARCHAR(32)** Indicates the name of the index.
SEGMENT_NUMBER	**SMALLINT** Indicates the position of the column within the index key.
NUMBER_OF_SEGMENTS	**SMALLINT** Indicates the number of columns included in the index key.
DESCENDING	**CHARACTER(5)** Indicates whether the index is marked as ascending or descending. ASC indicates ascending while DESC indicates descending.
UNIQUE	**CHARACTER(5)** Indicates whether duplicate values are allowed. TRUE indicates all values must be distinguishable, one from another. FALSE indicates duplicate values are allowed.
CREATOR_NAME	**VARCHAR(32)** Indicates the user name of the creator of the index.
COLUMN_NAME	**VARCHAR(32)** Indicates the name of the column which makes up this segment of the index key.
REMARK	**VARCHAR(128)** A comment.

System Catalog SYSTEM.TABLES

This table contains information about all tables in the database. You may only modify the REMARK field of records in this table.

The TABLE_NAME of temporary tables used during the execution of a query have a '~' as their first character.

Column	Description
TABLE_ID	**SMALLINT** Used internally to identify the MSDOS files used in the table.
NUMBER_OF_COLUMNS	**SMALLINT** Indicates the number of columns in the table.
ISAM_CONTROL_BLOCK	**BINARY** Used internally to save the ISAM control block for the table.
USER_NAME	Indicates the name of the user who owns the table.
TABLE_NAME	**VARCHAR(32)** Indicates the name of the table.
REMARK	**VARCHAR(128)** A comment.

System Catalog SYSTEM.USERS

This table contains information about all users known to the database.

You may manage user access by modifying the SYSTEM.USERS table.   Use Caution: deleting all user records will permanently lock you out of the database.   Do not delete the user "SYSTEM".

Column	Description
USER_NAME	**VARCHAR(32)** Indicates a user name.
USER_PASSWORD	**VARCHAR(32)** Indicates a user password.
REMARK	**VARCHAR(128)** A comment.

current user

The current user is the user who logged into the database to execute the query.

key

A key is the set of columns within a table used to construct an index on that table.

key value

The database administrator creates a key value by concatenating the values of all columns defined in an index. The columns are concatenated in the order in which they were specified when the index was created.

result table

All SELECT queries generate a table containing the chosen records. This is called the *result table*. Its contents are made available to you one at a time through one of the Orion SQL API fetch Statements.

scroll bar

A bar that appears at the right and/or bottom edge of a window whose contents aren't completely visible. Each scroll bar contains two scroll arrows and a scroll box, which allow you to scroll within the window or list box.

transaction journal

A transaction journal is a pair of files ('or_log.dat' and 'or_log.idx') written by the database manager. The transaction journal contains a record of every event which caused a change to the database.