Quasar 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

What is SQL?

Structured Query Language (SQL - pronounced "sequel") is the *lingua franca* of the computer database world. It was originally designed by D. D. Chamberlin and others of the IBM Research Laboratory in San Jose, California. Unlike other languages developed in a rather *ad hoc* manner, SQL has been designed to manipulate data on a very strict theoretical basis, which makes it stable and predictable. The fact that SQL has been minimally changed or enhanced in the years it has existed is proof of this fact. Of course, no one has to know relational algebra, or mathematics for that matter, to use SQL, but it is comforting to know that the language has a solid theoretical basis that will make it lasting and easy to use.

The power of the language allows software developers to write SQL commands that can replace dozens or even hundreds of lines of code. This not only greatly reduces development time, but also substantially reduces the expense of maintaining and enhancing systems. Value-added resellers (VAR) who use SQL can tailor a system to specific customer requirements quickly and at a low cost.

Notational Conventions

While uppercase is used throughout this document to indicate SQL keywords (COMMIT, SELECT, etc), in actuality the Quasar 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:

COMMENT ON COLUMN	Attaches a comment to a record in the COLUMNS **system catalog** table.
COMMENT ON TableName	Attaches comments to multiple records in the COLUMNS system catalog table which are associated with TableName.
COMMENT ON TABLE	Attaches a comment to a record in the TABLES system catalog table.
COMMENT ON USER	Attaches a comment to a record in the USERS system catalog table.
COMMIT	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.
GRANT CONNECT	Instructs the Database Administrator to allow an user access to the database.
GRANT Privilege	Instructs the Database Administrator to allow an user certain access to a specified table.
INSERT	Creates records in a database table.
REVOKE CONNECT	Instructs the Database Administrator to no longer allow an user access to the database.
REVOKE Privilege	Instructs the Database Administrator to reduce an user's access to a specified table.
ROLLBACK	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: COMMENT ON COLUMN

Syntax

```
COMMENT ON COLUMN
    [UserName.]TableName.ColumnName
    IS CharacterString;
```

CharacterString is written into the REMARK field of the record associated with **ColumnName** in the SYSTEM.COLUMNS table.

Phrase	Description
[UserName.]TableName.ColumnName	Identifies the record in the SYSTEM.COLUMNS table which is to receive the comment. If UserName is not specified, the current user is assumed. Only the system administrator may specify an **UserName** other than their own.
IS CharacterString	Provides the remark to be used. A comment may be removed by specifying 'NULL'.

Security The system administrator may apply a comment to any column of any table in the database. All other users may only apply comments to columns in their own tables.

Concurrency No locks are acquired.

Comments A remark in the SYSTEM.COLUMNS table provides a convenient method of documenting the purpose of the column.

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

Statement: COMMENT ON TableName

Syntax

```
COMMENT ON [UserName.]TableName (
    ColumnName1 IS CharacterString1[,
    ColumnName2 IS CharacterString2] ...
);
```

Each **CharacterString** is written into the REMARK field of the record associated with each **ColumnName** in the SYSTEM.COLUMNS table.

Phrase	Description
COMMENT ON [UserName.]TableName	Identifies the table whose columns are to receive the comment. If UserName is not specified, the current user is assumed. Only the system administrator may specify an **UserName** other than his own.
ColumnName1 IS CharacterString1	**CharacterString** is written into the REMARK field of the record which is associated with **ColumnName1** in the SYSTEM.COLUMNS table. A comment may be removed by specifying 'NULL'.
[, ColumnName2 IS CharacterString2] ...	Additional REMARK fields may be written within SYSTEM.COLUMNS table by creating a list of definitions separated by commas. Comments may be removed by specifying 'NULL'.

Security The system administrator may apply a comment to any column of any table in the database. All other users may only apply comments to columns in their own tables.

Concurrency No locks are acquired.

Comments A remark in the SYSTEM.COLUMNS table provides a convenient method of documenting the purpose of the column.

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

Statement: COMMENT ON TABLE

Syntax

```
COMMENT ON TABLE [UserName.]TableName
    IS CharacterString;
```

CharacterString is written into the REMARK field of the record associated with **TableName** in the SYSTEM.TABLES table.

Phrase	Description
COMMENT ON TABLE [UserName.]TableName	Identifies the record in the SYSTEM.TABLES table which is to receive the comment. If UserName is not specified, the current user is assumed. Only the system administrator may specify an **UserName** other than his own.
IS CharacterString	Provides the remark to be used. A comment may be removed by specifying 'NULL'.

Security The system administrator may apply a comment to any table in the database. All other users may only apply comments to their own tables.

Concurrency No locks are acquired.

Comments A remark in the SYSTEM.TABLES table provides a convenient method of documenting the purpose of the table.

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

Statement: COMMENT ON USER

Syntax **COMMENT ON USER UserName IS CharacterString;**

CharacterString is written into the REMARK field of the record associated with **UserName** in the SYSTEM.USERS table.

Phrase	Description
COMMENT ON USER UserName	Identifies the record in the SYSTEM.USERS table which is to receive the comment.
IS CharacterString	Provides the remark to be used. A comment may be removed by specifying 'NULL'.

Security Only the system administrator may use this statement.

Concurrency No locks are acquired.

Comments A remark in the SYSTEM.USERS table provides a convenient method of documenting the role of the user.

UserName is limited to 32 characters.

Statement: COMMIT

Syntax	**COMMIT [WORK];**
	Changes made to the database are not made permanent until this statement is executed. Should the current user execute a **ROLLBACK** or unexpectedly log off, all changes made by the user since logging in or executing a **COMMIT** (whichever occurred most recently) will vanish.
	During database recovery only committed transactions are restored.
Security	No authorization is required.
Concurrency	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 tables. 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** or **ROLLBACK** whenever possible.
Comments	Committing a transaction automatically starts a new transaction. **COMMIT** 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 Quasar 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 an 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 an 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.

Security The system administrator and the owner of the table are always authorized to create an index. Other users may create an index if and only if they have been granted **INDEX** access on the table by either the system administrator or by the owner of the table. **INDEX** access is granted via the **GRANT Privilege** statement.

Concurrency An exclusive lock is acquired on the table upon which the index is constructed. The lock is released when the transaction is committed.

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 an 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.

Security The system administrator and the prospective owner of the table are always authorized to create a table. No user, other than the system administrator, may create a table which will belong to another user.

Concurrency As soon as the table is created, an exclusive lock is acquired on it. The lock is released when the transaction is committed.

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 an user different from the current user.
WHERE SearchCondition	**SearchCondition** specifies which records are to be deleted.

Security The system administrator and the owner of the table are always authorized to delete records. Other users may delete records if and only if they have been granted **DELETE** access on the table by either the system administrator or by the owner of the table. **DELETE** access is granted via the **GRANT Privilege** statement.

Concurrency An exclusive lock is acquired on the table from which records are to be deleted. If the **SearchCondition** contains any subqueries, shared locks are acquired on all tables identified in the **FROM** clauses of those subqueries. All locks are released when the transaction is committed.

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 an 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 an user different from the current user.

Security The system administrator and the owner of the table are always authorized to drop an index. Other users may drop an index if and only if they created the index and have been granted **INDEX** access on the table by either the system administrator or by the owner of the table. **INDEX** access is granted via the **GRANT Privilege** statement.

Concurrency An exclusive lock is acquired on the table from which the index is dropped. The lock is released when the transaction is committed.

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 an user different from the current user.

Security The system administrator and the owner of the table are always authorized to drop a table. No user, other than the system administrator, may drop a table which belongs to another user.

Concurrency An exclusive lock is acquired on the table to be dropped. The lock is released when the transaction is committed.

Comments All indices associated with the table are automatically dropped. All grants of privileges on the table are automatically revoked.

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

Statement: GRANT CONNECT

Syntax

GRANT CONNECT TO UserName1[, UserName2] ...
IDENTIFIED BY Password1[, Password2] ... ;

This statement grants users the privilege of logging on to the Database Administrator, executing queries and owning tables and indices.

Phrase	Description
GRANT CONNECT TO UserName1	Identifies the user name to be granted connect privileges.
[, UserName2] ...	Additional users may be granted connect privileges by creating a list of user names separated by commas.
IDENTIFIED BY Password1	Identifies the password to be associated with the user name.
[, Password2] ...	If additional users were granted connect privileges, then each must be assigned a password by creating a list of passwords separated by commas.

Security Only the system administrator may use this statement.

Concurrency No locks are acquired.

Comments If **UserName** identifies a pre-existing user, the effect is that the user's password is changed to the new setting.

Password and **UserName** are limited to 32 characters

Statement: GRANT Privilege

Syntax

```
GRANT
    [ALL PRIVILEGES | Privilege1[, Privilege2] ... ]
    ON TableName
    TO [PUBLIC | UserName1[, UserName2] ... ] ;
```

This statement grants users certain access privileges to a specified table.

Phrase	Description
[ALL PRIVILEGES | Privilege1[, Privilege2] ...]	Identifies the access to be granted. Access is defined by creating a list of **Privilege** specifications separated by commas. The phrase **ALL PRIVILEGES** may be substituted for a list of all possible accesses.
ON TableName	Identifies the table upon which the access is to be granted.
TO [PUBLIC | UserName1[, UserName2] ...]	Identifies the users who are to receive the access. The phrase **PUBLIC** may be specified instead of a list of user names. **PUBLIC** implies all users are granted the specified access.

Security The system administrator and the owner of the table are always authorized to grant access to the table. No user, other than the system administrator, may grant access to a table which belongs to another user.

Concurrency No locks are acquired.

Comments Access granted by the system administrator is tracked separately from access granted by the owner of the table. All redundant access grants are combined; that is, granting the same access several times to the same user creates only one **SYSTEM.TABLE_AUTHORIZATION** record.

The effect of granting access may be reversed by the **REVOKE PRIVILEGE** statement.

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 an 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.

Security The system administrator and the owner of the table are always authorized to insert records. Other users may insert records if and only if they have been granted **INSERT** access on the table by either the system administrator or by the owner of the table. **INSERT** access is granted via the **GRANT Privilege** statement.

Concurrency An exclusive lock is acquired on the table into which records are to be inserted. If a **SelectStatement** is employed, shared locks are acquired on all tables identified in its **FROM** clause. If the **SelectStatement** contains any subqueries, shared locks are acquired on all tables identified in the **FROM** clauses of those subqueries. All locks are released when the transaction is committed.

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

Statement: REVOKE CONNECT

Syntax

REVOKE CONNECT FROM
    [PUBLIC | UserName1[, UserName2] ...];

This statement revokes from users the privilege of logging on to the Database Administrator, executing queries and owning tables and indices. The user is effectively removed from the database.

Phrase	Description
[PUBLIC | UserName1[, UserName2] ...]	
	Identifies the user name to be removed. Additional users may be removed by creating a list of user names separated by commas. The phrase **PUBLIC** may be specified instead of a list of user names. **PUBLIC** implies all users (other than the system administrator) are to be eliminated.

Security Only the system administrator may use this statement.

Concurrency Exclusive locks are acquired on all tables belonging to the user and all tables upon which the user has created an index. All locks are released when the transaction is committed.

Comments When an user's connect privilege is revoked: all that user's tables (and any associated indices) are automatically dropped; all indices created by that user are automatically dropped (even if created on another user's table); all privileges granted by that user to another user are automatically revoked.

UserName is limited to 32 characters

Statement: REVOKE Privilege

Syntax

```
REVOKE
    [ALL PRIVILEGES | Privilege1[, Privilege2] ... ]
    ON TableName
    FROM [PUBLIC | UserName1[, UserName2] ... ];
```

This statement reduces users' access privilege to a specified table.

Phrase	Description
[ALL PRIVILEGES | Privilege[, Privilege2] ...]	Identifies the access to be revoked. Access is defined by creating a list of **Privilege** specifications separated by commas. The phrase **ALL PRIVILEGES** may be substituted for the list of all possible accesses.
ON TableName	Identifies the table upon which the access is to be revoked.
FROM [PUBLIC | UserName1[, UserName2] ...]	Identifies the users who are to lose the access. The phrase **PUBLIC** may be substituted for the list of user names. **PUBLIC** implies all users (other than the owner of the table and the system administrator) are to lose the specified access.

Security The system administrator and the owner of the table are always authorized to reduce an user's access to a table. No user, other than the system administrator, may revoke access to a table which belongs to another user.

Concurrency If **INDEX** access is revoked, and the user has created indices on the specified table: Exclusive locks are acquired on the table. The lock is released when the transaction is committed.

Comments When an user's **INDEX** privilege is revoked: all indices created on the specified table by the user are automatically dropped.

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

Statement: ROLLBACK

Syntax	**ROLLBACK [WORK];**
	Changes made to the database are not made permanent until a **COMMIT** statement is executed. **ROLLBACK** instructs the database to purge all changes to the database by the current user since logging in or executing a **COMMIT** (whichever occurred most recently).
Security	No authorization is required.
Concurrency	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 tables. 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** or **ROLLBACK** whenever possible.
Comments	Rolling a transaction back automatically starts a new transaction. **ROLLBACK** may take some time while the Database Administrator purges updates.

Statement: SELECT

Syntax

```
SELECT [ALL | DISTINCT]
    [ALL | 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 an 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 tables identified by the **FROM** clause into groups such that within any one group all rows have the same value for the **GROUP BY** columns. 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 Quasar 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 Quasar 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 Quasar 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.

Security The system administrator and the owner of the table are always authorized to select records. Other users may select records if and only if they have been granted **SELECT** access on the table by either the system administrator or by the owner of the table. **SELECT** access is granted via the **GRANT Privilege** statement.

Concurrency A shared lock is acquired on the tables from which records are to be selected. If the **SearchCondition** contains any subqueries, shared locks are acquired on all tables identified in the **FROM** clauses of those subqueries. These locks are released when the transaction is committed.

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 an 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.

Security The system administrator and the owner of the table are always authorized to update records. Other users may update records if and only if they have been granted **UPDATE** access on the table by either the system administrator or by the owner of the table. **UPDATE** access is granted via the **GRANT Privilege** statement.

Concurrency An exclusive lock is acquired on the table in which records are to be updated. If the **SearchCondition** contains any subqueries, shared locks are acquired on all tables identified in the **FROM** clauses of those subqueries. All locks are released when the transaction is committed.

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', 'Quasar')

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.

Privilege

Privileges to access data in a table may be granted to users by either the system administrator or the owner of the table. Privileges are granted with the **GRANT Privilege** statement. Privileges are revoked with the **REVOKE Privilege** statement.

Various types of access may be granted:

ALL PRIVILEGES

This is the equivalent of the combined access of **DELETE**, **INDEX**, **INSERT**, **SELECT** and **UPDATE**.

DELETE

The user receiving **DELETE** access may delete records from the table.

INDEX

The user receiving **INDEX** access may create and drop indices on the table.

INSERT

The user receiving **INSERT** access may insert records into the table.

SELECT

The user receiving **SELECT** access may select records from the table.

UPDATE

The user receiving **UPDATE** access may update records in the table.

UPDATE (ColumnName1[, ColumnName2] ...)

The user receiving **UPDATE** column access may update only the named columns in the table. This option is not available when you revoke **UPDATE** privileges.

Several privileges may be granted at one time by creating a list of privileges separated by commas, **ALL PRIVILEGES** may not, however, be used within a list of privileges.

Privileges may be revoked from users in a similar fashion. When **UPDATE** privilege is revoked, individual columns may not be specified: **UPDATE** privilege may only be revoked for the entire table.

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 six tables: **SYSTEM.COLUMNS**, **SYSTEM.COLUMN_AUTHORIZATION**, **SYSTEM.INDICES**, **SYSTEM.TABLES**, **SYSTEM.TABLE_AUTHORIZATION** and **SYSTEM.USERS**. All these tables belong to the system administrator. These tables are automatically maintained by the Quasar Database Administrator.

Unless otherwise indicated, you must not modify the system catalog tables.

COLUMNS	This table contains information about all columns of all tables in the database.
COLUMN_AUTHORIZATION	This table contains one record for each grant of UPDATE access on a specific column of a table. Redundant grants are combined into a single record.
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.
TABLE_AUTHORIZATION	This table contains one record for each grant of access to a table. Redundant grants are combined into a single record.
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. While you may modify the **REMARK** field of records in this table, the preferred method is to use the **COMMENT ON COLUMN** statement. Modifying any other field will probably cause the system to irrecoverably crash.

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.COLUMN_AUTHORIZATION

This table contains one record for each grant of **UPDATE** access on a specific column of a table. While you may modify the **REMARK** field of records in this table, modifying any other field will probably cause the system to irrecoverably crash.

Column	Type/Description
GRANTOR	**VARCHAR(32)** Indicates the user name of the user granting access: the grantor will be either "SYSTEM" (the system administrator) or the owner of the table.
GRANTEE	**VARCHAR(32)** Indicates the user name of the user receiving the grant of access.
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. While you may modify the **REMARK** field of records in this table, modifying any other field will probably cause the system to irrecoverably crash.

The INDEX_NAME of indices created by the Database Administrator 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.
DECENDING	**CHARACTER(5)** Indicates whether the index is marked as ascending or descending. FALSE indicates ascending while TRUE 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. While you may modify the **REMARK** field of records in this table, the preferred method is to use the **COMMENT ON TABLE** statement. Modifying any other field will probably cause the system to irrecoverably crash.

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	**VARCHAR(32)** 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.TABLE_AUTHORIZATION

This table contains one record for each grant of access. While you may modify the **REMARK** field of records in this table, modifying any other field will probably cause the system to irrecoverably crash.

Column	Type/Description
GRANTOR	**VARCHAR(32)** Indicates the user name of the user granting access: the grantor will be either SYSTEM (the system administrator) or the owner of the table.
GRANTEE	**VARCHAR(32)** Indicates the user name of the user receiving the grant of access.
USER_NAME	**VARCHAR(32)** Indicates the user name of the owner of the table.
TABLE_NAME	**VARCHAR(32)** Indicates the name of the table.
UPDATE_COLUMNS	**CHAR** '*' indicates GRANTEE is allowed to update specific columns (as specified in **SYSTEM.COLUMN_AUTHORIZATION**) of the table, '' indicates no grant of this type of access.
DELETE_AUTHORIZATION	**CHAR** 'Y' indicates GRANTEE is allowed to delete records from the table, '' indicates no grant of this type of access.
INDEX_AUTHORIZATION	**CHAR** 'Y' indicates GRANTEE is allowed to create and drop indices on the table, '' indicates no grant of this type of access.
INSERT_AUTHORIZATION	**CHAR** 'Y' indicates GRANTEE is allowed to insert records into the table, '' indicates no grant of this type of access.
SELECT_AUTHORIZATION	**CHAR** 'Y' indicates GRANTEE is allowed to select records from the table, '' indicates no grant of this type of access.
UPDATE_AUTHORIZATION	**CHAR** 'Y' indicates GRANTEE is allowed to update records in the table, '' indicates no grant of this type of access.
REMARK	**VARCHAR(128)** A comment.

System Catalog SYSTEM.USERS

This table contains information about all users known to the database. While you may modify the **REMARK** field of records in this table, the preferred method is to use the **COMMENT ON USER** statement. Modifying any other field will probably cause the system to irrecoverably crash.

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

current user

For any query, the current user is the user who logged onto the database and executed the query exclusive lock

When an exclusive lock is obtained by one transaction on a table, other concurrent transactions attempting to acquire either an exclusive lock or a shared lock on the same table are aborted and rolled back.

key

A key is a 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 Quasar 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.

shared lock

When a shared lock is obtained by one transaction on a table, other concurrent transactions attempting to acquire an exclusive lock on the same table are aborted and rolled back.   Other concurrent transactions attempting to acquire a shared lock on the same table are allowed to do so.

system administrator

The user whose user name is SYSTEM.   This user has absolute authority over all tables in the database.

transaction journal

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