Connecting to Notes Databases

Connecting to a Notes database with many forms and views can be relatively slow. This is because the Lotus Notes ODBC driver builds system catalogs derived from Notes form and view design documents in the Notes database during SQLConnect.

Lotus Notes ODBC Setup Dialog Box

1. Enter a name that identifies the data source in the Data Source Name text box. For example, add the name Employee to identify the ODBC connection to an employee database.
2. (Optional) Enter a description of the data source in the Description text box. For example, add the description "Hire date, salary history, and current review of all employees" to describe the contents of the employee database.
3. Enter the name of the Notes server that contains the Notes database you want to open in the Server text box. Leave the text box blank if the Notes database is on a local disk.
4. Enter the path and name of the .NSF file you want to open in the Database text box. For example,

   C:\PERSONNEL\EMPLOYEE.NSF

NotesSQL Options Setup

You can modify any of the following optional parameters. The values for these parameters affect run-time memory allocation:

Max Length of Text Fields

This parameter specifies the maximum number of characters the Notes driver allows in a string. This includes the limit on the number of characters returned from a Notes text field, as well as the limit on the length of a string to be inserted into a Notes document.

The maximum value allowed is 10,240 (10K).

The default value is 512.

Note: Notes has long text fields called rich text fields. The body of a Notes document is typically a single rich text field. If you want to retrieve all the data in such a field, be sure that the maximum string length you specify is high enough.

Max Number of Tables

This parameter specifies the maximum number of tables in a single query.

The maximum value allowed is 100.

The default value is 20.

Max Number of Subqueries

This parameter specifies the maximum number of nested subqueries in an SQL statement.

The maximum value allowed is 100.

The default value is 20.

SQL Statement Length

This parameter specifies the maximum length of an SQL statement passed to SQLPrepare/SQLExecDirect.

The maximum value allowed is 32,767 (32K).

The default value is 1,024.

Keep Temporary Indexes Until SQLDisconnect

This option button controls the saving of temporary indexes. Select the option to save temporary indexes until SQLDisconnect. Leave the option unselected to delete indexes at the end of each SELECT result.

See also

Hardware and Software Requirements
Adding, Modifying, and Deleting a Lotus Notes Data Source

Lotus Notes Data Source

A Lotus Notes data source specifies the Notes server and Notes database you want to open.

Using the Lotus Notes ODBC Driver

The following topics provide helpful information about using the Lotus Notes ODBC Driver:

Optimizing Lotus Notes ODBC Driver Performance

Mapping SQL Tables, Views, and Indexes To and From Notes

Notes View Column Definitions

Understanding the Universal Relation Table

Column Names

Connecting to Notes Databases

Using Reserved Words

Connecting to a Lotus Notes Data Source

If your Notes ID is password-protected, Notes prompts you for a password every time you try to connect to a remote database using the Lotus Notes ODBC driver. The password is not database-specific, and is not retained, so you will need to enter it more than once if you connect to more than one database.

Notes never prompts for your Notes ID. Your Notes ID is part of your workstation installation.

Adding, Modifying, and Deleting a Lotus Notes Data Source

Before you can query data with the Lotus Notes ODBC driver, you must add a data source for it. You can change or delete a data source at any time.

To add a Lotus Notes data source

1. In the Main group in the Program Manager window, double-click the Control Panel icon.
2. In the Control Panel window, double-click the ODBC icon.
   The Data Sources dialog box appears.
3. Choose Add.
   The Add Data Source dialog box appears.
4. Select Lotus Notes from the Installed ODBC Drivers list box.
5. Choose OK.
6. In the Lotus Notes ODBC Setup dialog box, enter information to set up the data source.

To modify a Lotus Notes data source

1. In the Main group in the Program Manager window, double-click the Control Panel icon.
2. In the Control Panel window, double-click the ODBC icon.
   The Data Sources dialog box appears.
3. Select the data source in the Data Sources (Drivers) list box.
4. Choose Setup.
5. In the Lotus Notes ODBC Setup dialog box, modify the information about the data source.

To delete a Lotus Notes data source

1. In the Main group in the Program Manager window, double-click the Control Panel icon.
2. In the Control Panel window, double-click the ODBC icon.
   The Data Sources dialog box appears.
3. Select the data source in the Data Sources (Drivers) list box.
4. Choose Delete, and then choose Yes to confirm the deletion.

See also

Installing the Lotus Notes ODBC Driver
Connecting to a Lotus Notes Data Source

Installing the Lotus Notes ODBC Driver

1. Double-click the Control Panel icon in the Main Group of the Program Manager window.
   The Control Panel window appears.
2. Double-click the ODBC icon.
   The Data Sources dialog box appears.
3. Choose Drivers.
   The Drivers dialog box appears.
4. Choose Add.
   The Add Drivers dialog box appears.
5. Enter the name of the drive and directory containing the Notes driver in the text box and choose OK; or choose Browse to select a drive and directory name.
   The Install Drivers dialog box appears.
6. In the Available ODBC Drivers list box, select Lotus Notes.
7. Choose OK.
   The Notes driver is installed.

To delete the Lotus Notes ODBC driver

1. Double-click the Control Panel icon in the Main group of the Program Manager window.
2. Double-click the ODBC icon.
   The Data Sources dialog box appears.
3. Choose Drivers.
   The Drivers dialog box appears.
4. In the Installed ODBC Drivers list box, select Lotus Notes.
5. Choose Delete.
   A message asks you to confirm that you want to remove the driver and all the data sources that use the driver.
6. Choose Yes.

Lotus Notes ODBC Driver

For All Users

The following topics discuss the Notes driver and how to install it.

Overview

Hardware and Software Requirements

Installing the Lotus Notes ODBC Driver

Adding, Modifying, and Deleting a Lotus Notes Data Source

Connecting to a Lotus Notes Data Source

Using the Lotus Notes ODBC Driver

For Advanced Users

The following topics discuss how to use the Notes driver directly.

Connection Strings

SQL Statements

Data Types

Error Messages

For Programmers

The following topics provide programming information on the Notes driver. These topics provide helpful information for application programmers who have knowledge of the Open Database Connectivity (ODBC) application programming interface (API).

SQLGetInfo Return Values

ODBC API Functions

Hardware and Software Requirements

To query Notes data, you must have

- The Lotus Notes ODBC driver
- Notes database files
- A computer running MS-DOS 3.3 or later
- Microsoft Windows 3.1 or later
- The ODBC Driver Manager 1.0 (ODBC.DLL)

Notes version 3 Workstation software must be installed and be in the executable path.

Notes database files can reside on a server. You don't need to have local copies of these files, but you must have at least Reader access to them through Notes.

To add, modify, or delete drivers or data sources, you should have the ODBC Control Panel option installed on your computer.

For more information about Notes databases, refer to your Lotus Notes documentation.

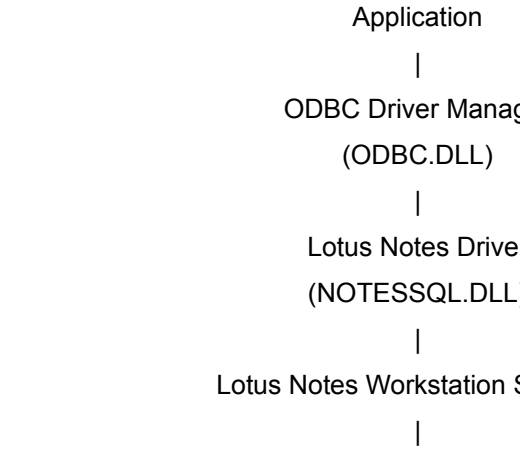
See also

Installing the Lotus Notes ODBC Driver

Overview

The Lotus Notes ODBC driver allows you to open and query a Notes database through the Open Database Connectivity (ODBC) interface.

The application/driver architecture is:



Note The Notes driver communicates exclusively with the Lotus Notes workstation software. Lotus Notes handles all network traffic and security.

See also

Hardware and Software Requirements
Installing the Lotus Notes ODBC Driver
Adding, Modifying, and Deleting a Lotus Notes Data Source
Connecting to a Lotus Notes Data Source
Using the Lotus Notes ODBC Driver

ODBC

ODBC (Open Database Connectivity) is an interface that allows applications to get to data in database management systems that use SQL. The interface allows a single application to connect to many different types of databases through a standard protocol. ODBC is implemented as a Driver Manager and multiple drivers. Each driver links the application to a specific database.

Using the Universal Relation Table

- A field name can exist in multiple forms with different data types in a Notes database. Therefore, you must specify field names in a SELECT clause that references the Universal Relation table. For example, you cannot enter SELECT *. You can only perform text operations on fields in the Universal Relation table because the data type for all fields is Character.
- The Universal Relation table can participate in a self-join, but cannot participate in any other kind of join.
- The Universal Relation table supports only SELECT and CREATE VIEW statements. You cannot perform an INSERT, DELETE, or UPDATE statement in a Universal Relation table.
- You cannot delete the Universal Relation table with the DROP TABLE statement.

Understanding the Universal Relation Table

The Universal Relation table contains the fields from all the forms in the Notes database. SQL tables created with the Notes driver are similar to SQL views rather than to traditional relational database tables.

For example, with the Notes driver, you can create a Notes form with the CREATE TABLE statement. However, the DROP TABLE statement deletes the Notes form, but does not delete any documents from the database. Using DROP TABLE with the Lotus Notes driver is like deleting an SQL view. The data remains in the database. You can view the data through other forms or views that use the same field names, or by referencing the Universal Relation table.

Using the Notes driver, if you create a new table with the same name as a previously deleted table, and use some of the field names from the deleted table, you could find data in the table before you insert any data. This is because the table is actually a view of existing data in the Universal Relation table. Fields in the Notes database contain a reference to the form used to create them. The Notes driver uses the form name stored with the field to identify the field when selecting from a form.

Using the Universal Relation Table

Connecting to a Lotus Notes Data Source

If your Notes ID is password-protected, Notes prompts you for a password every time you try to connect to a remote database using the Lotus Notes ODBC driver. The password is not database-specific, and is not retained, so you will need to enter it more than once if you connect to more than one database.

Notes never prompts for your Notes ID. Your Notes ID is part of your workstation installation.

Example: Comparing the Use of Forms and Views

The Notes Name and Address Book is a good database to use as an example to compare the use of forms or views in a database. The Name and Address Book database includes

- A form called Person
- A view called People with a sort key on LastName

The following statement is the most efficient way to find people in the Name and Address Book sorted by LastName:

```
SELECT LastName
FROM People
ORDER BY LastName
```

People is a Notes view. This query is efficient because the Lotus Notes ODBC driver can use the index already associated with the view People that lists LastName in the right order.

Now assume you want to list people sorted by their mailing addresses. You could use the following statement:

```
SELECT LastName, Mail_Address
FROM People
ORDER BY Mail_Address
```

Since the view People is not sorted on Mail_Address, the Notes driver uses the People index, generates a temporary database, and creates a temporary index on Mail_Address. This results in slower performance. This has a significant impact on performance when the database is on the server and you do not have designer access.

A more efficient way to achieve the same result is to issue the following statement:

```
SELECT LastName, Mail_Address
FROM Person
ORDER BY Mail_Address
```

Person is a Notes form. If there is no index on Mail_Address, the Notes driver generates a temporary index on Mail_Address but does not need to generate a temporary database. This statement is faster than the previous statement, which used ORDER BY on a view-based table. This statement is even faster if the user creates an index using the CREATE INDEX statement in the Notes driver or creates an index through Notes view creation.

Note You need Designer access to create a view if the database is on a server.

Reserved Words

Avoid using the reserved words listed in Appendix C of the *Microsoft ODBC Programmers Reference* as identifiers (table or column names).

If you must use a reserved word, enclose it in quotation marks (" "). For example, to use the reserved word DATE type "DATE".

Column Names

Do not use column names that contain characters other than alphabetic, numeric, dollar sign ($), or underscores (_) in SQL statements. However, if you enclose the column name in quotation marks (" ") any character is permitted.

Mapping SQL Tables, Views, and Indexes To and From Notes

- Each SQL table is derived from a Notes form.
- Each SQL index is derived from a Notes view in which all sorted columns map to fields in a single form, and which selects documents only through that form.
- Each SQL view is derived from a Notes view that selects documents through just one form.

Lotus Development Corporation

55 Cambridge Parkway

Cambridge, MA 02142

View Column Definitions

If a column in a view does not refer directly to a field, the Lotus Notes ODBC driver names the column in the SQLColumns result set starting with the character $ followed by a number, for example $2.

If a view column definition contains one of the @functions listed below, the Notes driver returns no results for that column. These columns will display data when viewed from Notes, but this data is not available through an SQL query.

@All
@DeleteDocument
@DeleteFields
@DocChildren
@DocLevel
@DocNumber
@DocParentNumber
@DocSiblings
@Error
@IsCategory
@IsExpandable
@Unavailable

Optimizing Lotus Notes ODBC Driver Performance

Driver performance is affected by your hardware and software environment and other factors including:

- Access to the database
- The type of table you query
- The design of the Notes database
- The selection criteria you specify

Database Access

- Performance working with server based databases is affected by the type of network hardware and software you use and the amount of activity on the network. A query executed on a server based database that returns a small result set generally provides better performance than the same query on a local database.
- If you have designer access to the server based database the Notes driver can create a temporary view on the server to sort the data. If you don't have designer access and try to sort on a field without an index the Notes driver copies data to a temporary local Notes database and creates the view on the temporary copy.

Types of Notes Tables

The Notes driver can distinguish both Notes forms and views as tables. In addition the driver recognizes the Notes Universal Relation Table. However, Notes forms and views have very different properties that effect the performance of data access with the Lotus Notes ODBC driver.

> **Note** Notes documents created by a form with the Store form in document option selected in Notes form attributes do not appear in views that select documents using that form.

Notes views are indexed, and queries on views can use the indexes to improve performance. You can optimize performance by modifying existing Notes views or creating new Notes views (you need Designer access to create or modify views on a server) to sort data in Notes rather than sorting the data with the Lotus Notes driver.

Example

Comparing the Use of Forms and Views

Database Design Guidelines

Here are some guidelines for creating views in Notes to optimize performance from the Notes driver:

- Before you create an index you should consider the following information. An index on a table that is small compared to the size of the entire database speeds up access. If there is a single table for the database the index may slow down access time.

  If a WHERE clause uses the index, and the index selects a subset (20% or less) of the documents in the database, then the use of the index improves access time. An index on the whole database (SELECT @ALL) slows access.

- Design the Notes view so that columns appear in the following order:

  Categorized columns

  Sorted, uncategorized columns

  Non-sorted columns

- Avoid using the Store Form in Document option in Form Attributes. Documents created with this option do not appear in the table derived from the form.

You can retrieve data from forms that use the Store Form in Document option by creating a view and using either of the following formulas:

SELECT $TITLE = form name

SELECT@ALL

- Avoid creating more forms and views than necessary. Connection to a database with many forms and views can be relatively slow.
- Avoid using complex formulas in forms because Notes evaluates these formulas when you execute an INSERT or UPDATE statement.
- When you need to perform many inserts with the Notes driver change the Index Options in Design View Attributes to manual. If the Index option is automatic Notes updates the index every time you add or modify a document with an INSERT and UPDATE statement.
- The Notes driver is case sensitive. To avoid matching problems make form and view names uppercase.
- To make it easier to identify views created specifically for SQL access you might consider creating a coding scheme to identify them. For example, you could preface all the view names with the letters "SQL".

Query Guidelines

The following are guidelines to consider when you create a query.

- Try to use selection criteria only on categorized and sorted columns.
- Avoid selecting a table based on a Notes view and specifying a different sort order. When you specify a different sort order on an existing view Notes creates a copy of the data in a local database and a view to sort the data.
- When you use a WHERE or ORDER BYclause in a SELECT statement the query is much faster if there's an index on the columns specified in the WHERE or ORDER BYclause.

Using Notes Views as Indexes

The Notes driver is able to use Notes views as indexes that meet the following characteristics:

- The view selection formula is SELECT Form = "name". You can use SELECT @All but it is much slower.
- Each column in the view is defined as a simple reference to a field in that form (no formulas)
- At least one column is sorted.
- Each view name used as an index must be unique.

SQLGetInfo Return Values (Programming)

The following table lists the C language #defines for the fInfoType argument and the corresponding values returned by SQLGetInfo. You can retrieve this information by passing the C language #define to SQLGetInfo in the fInfoType argument.

***fInfoType* Value (#define)**	**Lotus Notes Driver Return Values**
SQL_ACCESSIBLE_PROCEDURES	No
SQL_ACCESSIBLE_TABLES	No
SQL_ACTIVE_CONNECTIONS	No Limit
SQL_ACTIVE_STATEMENTS	No Limit
SQL_CONCAT_NULL_BEHAVIOR	Result is NULL valued.
SQL_CONVERT_BIGINT	None
SQL_CONVERT_BINARY	None
SQL_CONVERT_BIT	None
SQL_CONVERT_CHAR	CHAR, VARCHAR, LONGVARCHAR, DATE, TIME, or TIMESTAMP
SQL_CONVERT_DATE	DATE, VARCHAR, LONGVARCHAR, or TIMESTAMP
SQL_CONVERT_DECIMAL	CHAR, NUMERIC, DECIMAL, INTEGER, SMALLINT, FLOAT, REAL, DOUBLE, VARCHAR, or LONGVARCHAR
SQL_CONVERT_DOUBLE	CHAR, NUMERIC, DECIMAL, INTEGER, SMALLINT, FLOAT, REAL, DOUBLE, VARCHAR, or LONGVARCHAR
SQL_CONVERT_FLOAT	CHAR, NUMERIC, DECIMAL, INTEGER, SMALLINT, FLOAT, REAL, DOUBLE, VARCHAR, or LONGVARCHAR
SQL_CONVERT_FUNCTIONS	Not supported
SQL_CONVERT_INTEGER	CHAR, NUMERIC, DECIMAL, INTEGER, SMALLINT, FLOAT, REAL, DOUBLE, VARCHAR, or LONGVARCHAR
SQL_CONVERT_LONGVARBINARY	None
SQL_CONVERT_LONGVARCHAR	CHAR, VARCHAR, LONGVARCHAR, DATE, TIME, or TIMESTAMP
SQL_CONVERT_NUMERIC	CHAR, NUMERIC, DECIMAL, INTEGER, SMALLINT, FLOAT, REAL, DOUBLE, VARCHAR, or LONGVARCHAR
SQL_CONVERT_REAL	CHAR, NUMERIC, DECIMAL, INTEGER, SMALLINT, FLOAT, REAL, DOUBLE, VARCHAR, or LONGVARCHAR
SQL_CONVERT_SMALLINT	CHAR, NUMERIC, DECIMAL, INTEGER, SMALLINT, FLOAT, REAL, DOUBLE, VARCHAR, or LONGVARCHAR
SQL_CONVERT_TIME	CHAR, VARCHAR, LONGVARCHAR,

	TIME, or TIMESTAMP
SQL_CONVERT_TIMESTAMP	CHAR, VARCHAR, LONGVARCHAR, DATE, TIME, or TIMESTAMP
SQL_CONVERT_TINYINT	None
SQL_CONVERT_VARBINARY	None
SQL_CONVERT_VARCHAR	CHAR, VARCHAR, LONGVARCHAR, DATE, TIME, or TIMESTAMP
SQL_CURSOR_COMMIT_BEHAVIOR	Preserved
SQL_CURSOR_ROLLBACK_BEHAVIOR	Close/Delete
SQL_DATA_SOURCE_NAME	Data source name from SQLConnect
SQL_DATA_SOURCE_READ_ONLY	NOTESSQL does not check the source to see if it is read only.
SQL_DATABASE_NAME	Database name found from SQLConnect
SQL_DBMS_NAME	Lotus Notes
SQL_DBMS_VER	Lotus Notes Version 3
SQL_DEFAULT_TXN_ISOLATION	SQL_TXN_REPEATABLE_READ
SQL_DRIVER_NAME	NOTESSQL.DLL
SQL_DRIVER_VER	Returns the version of the installed driver.
SQL_EXPRESSIONS_IN_ORDERBY	No
SQL_FETCH_DIRECTION	NEXT
SQL_IDENTIFIER_CASE	Names are case sensitive and can be mixed case.
SQL_IDENTIFIER_QUOTE_CHAR	Double quote (" ")
SQL_MAX_COLUMN_NAME_LEN	32
SQL_MAX_CURSOR_NAME_LEN	32
SQL_MAX_OWNER_NAME_LEN	0
SQL_MAX_PROCEDURE_NAME_LEN	0
SQL_MAX_QUALIFIER_NAME_LEN	0
SQL_MAX_TABLE_NAME_LEN	32
SQL_MULT_RESULT_SETS	Not supported
SQL_MULTIPLE_ACTIVE_TXN	Not supported
SQL_NUMERIC_FUNCTIONS	Not supported
SQL_ODBC_API_CONFORMANCE	Level1
SQL_ODBC_SAG_CLI_CONFORMANCE	SAG-compliant
SQL_ODBC_SQL_CONFORMANCE	Minimum
SQL_ODBC_SQL_OPT_IEF	No
SQL_ODBC_VER	Returns the version of the ODBC driver manager being used.
SQL_OUTER_JOINS	No

SQL_OWNER_TERM	Not supported
SQL_PROCEDURE_TERM	Not supported
SQL_PROCEDURES	Not supported
SQL_QUALIFIER_NAME_SEPARATOR	None
SQL_QUALIFIER_TERM	None
SQL_ROW_UPDATES	No
SQL_SCROLL_CONCURRENCY	SQL_SCCO_OPT_VALUES
SQL_SCROLL_OPTIONS	SQL_SO_FORWARD_ONLY
SQL_SEARCH_PATTERN_ESCAPE	Backlash (\ )
SQL_SERVER_NAME	Server name found from SQLConnect
SQL_STRING_FUNCTIONS	Not supported
SQL_SYSTEM_FUNCTIONS	Not supported
SQL_TABLE_TERM	FORM
SQL_TIMEDATE_FUNCTIONS	Not supported
SQL_TXN_CAPABLE	Transactions not supported
SQL_TXN_ISOLATION_OPTION	Not supported
SQL_USER_NAME	None

ODBC API Functions (Programming)

The Lotus Notes ODBC driver supports all Core and Level 1 functions with the following exceptions:

- SQLGetStmtOption
- SQLSetStmtOption
- SQLTransact

The following describes how particular functions are implemented with the Notes driver.

SQLCancel

The SQLSetStmtOption is not supported. The SQLCancel function (without the SQLSetStmtOption ASYNC_ENABLE option) is useful for interrupting an SQLPutData function when the return of a long data field requires additional SQLPutData functions.

Returns:

SQL_INVALID_HANDLE

SQLColumns

If a column in a view does not refer directly to a field, the Notes driver names the column in the result set starting with the character $ followed by a number, for example $2. A call to SQLColumns returns only the first 32 characters of the remarks field. The remarks field corresponds to the Help Description for a field set in the form using the Field Definition dialog box.

SQLConnect

The Notes driver builds system catalogs for the database at SQLConnect. These catalogs are derived from Notes form and view design documents. Therefore, connection to a database with many forms and views will seem slow.

SQLSetConnectOption

This function allows the following settings:

SQL_AUTOCOMMIT = 1

SQL_ACCESS_MODE = SQL_MODE_READ_WRITE

All other settings return an error. The SQLSetConnectOption only verifies that you asked for one of the settings shown above and does not affect the driver's state.

SQLTables

Valid table types are:

FORM

SYNONYM

TABLE

VIEW

UNIVERSAL

A Notes database contains a table that has the same name as the database referred to as the Universal Relation table. The Universal Relation table contains all the fields in the Notes database. SQL tables created with the Notes driver are similar to SQL views rather than to traditional relational database tables.

SQLTransact

SQLTransact is supported in a limited way. Since Notes does not provide any transaction mechanism, SQL_COMMIT is supported but SQL_ROLLBACK returns an error.

Summary of Supported ODBC SQL Grammar

The Lotus Notes ODBC driver supports the following ODBC SQL grammar:

Supported Statements

- ALTER TABLE
- CREATE INDEX
- CREATE TABLE
- CREATE VIEW
- DELETE searched
- DELETE positioned
- DROP INDEX
- DROP TABLE
- DROP VIEW
- INSERT
- SELECT
  - FROM
  - WHERE
  - FOR UPDATE
  - GROUP BY
  - HAVING
  - ORDER BY
- UPDATE searched
- UPDATE positioned

Supported Expressions, Functions, and Operators

- Numeric Operators
- Predicate Operators
- Column Functions

<*subquery*> is a SELECT statement with the restriction that only one column can be specified in the SELECT clause.

FOR UPDATE

The FOR UPDATE clause specifies a list of column names.

Syntax

<FOR UPDATE clause> is FOR UPDATE OF <*column* list>

See Conventions

Example

ORDER BY

The ORDER BY clause specifies how to sort the records retrieved based on a query.

Syntax

<ORDER BY clause> is ORDER BY <*column name*> | <*integer*> [ASC | DESC] [, ...]

See Conventions

Example

HAVING

The HAVING clause specifies a search condition for a group.

Syntax

<HAVING clause> is HAVING <*search condition*>

See Conventions

The search condition must include a Column function.

Example

GROUP BY

The GROUP BY clause groups the data from source tables by one or more column names and produces a single summary row for each column name specified.

Syntax

<GROUP BY clause> is GROUP BY <*column* list>

See Conventions

This list cannot include derived columns. All columns in the GROUP BY clause must also appear in the SELECT clause.

Example

Numeric Operators

The Lotus Notes ODBC driver supports the following numeric operators in expressions.

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division

Predicate Operators

The Lotus Notes ODBC driver supports the following predicate operators.

Operator	Meaning
<	Less Than
>	Greater Than
<=	Less Than or Equal
>=	Greater Than or Equal
=	Equal
<>	Not Equal
BETWEEN	Specifies a range of values between a lower and upper boundary.
IN	Specifies a member of a set of specified values or a member of a subquery.
LIKE	Use for matching a pattern. Wildcard characters in LIKE predicate: Use an underscore (_) to represent a single character. Use a percent symbol (%) to represent any number of characters. Use a backslash (\ ) as the escape character.
NOT	Use the NOT operator with another operator to specify a search condition that is false. For example: NOT IN, NOT LIKE, or NOT BETWEEN.
ANY	Use to compare a value to each value returned by a subquery. Can be prefaced by =, <>, >, >=, <, <=, =. =ANY is equivalent to IN. <>ANY is equivalent to NOT IN.
SOME	SOME is an alternate keyword for ANY.
ALL	Use to compare a value to each value returned by a subquery. Can be prefaced by =, <>, >, >=, <, <=

EXISTS	"True" if a subquery returned at least one record.

Column Functions

Column functions can be part of a SELECT clause. A column function takes an entire column of data as its argument and produces a single data item that summarizes the column. For example, the AVG column function takes a column of data and computes its average.

You can use a column function with a field name or in combination with a more complex expression. The Lotus Notes ODBC driver supports the following Column functions.

Function	Description
AVG	Returns the average values in a numeric field expression. For example, AVG(SALES) returns the average of all sales fields.
COUNT	Returns the number of values in any field expression.
MAX	Returns the highest value in any field expression. For example, MAX(SALES) returns the highest sales field value.
MIN	Returns the lowest value in any field expression. For example, MIN(SALES) returns the lowest sales field value.
SUM	Returns the total of values in numeric field expression. For example, SUM(SALES) returns the sum of all sales field values.

WHERE

The WHERE clause specifies the conditions that records must meet for retrieval.

Syntax

<WHERE clause> is WHERE [NOT] <*search condition*>[{ AND | OR } [NOT] <*search condition*> ...]

See Conventions

Example

Arguments

<*search condition*> is one of the following:

- <comparison predicate> is:
  <expression> <comparison operator> <expression>
- <between predicate> is:
  <*expression*> [NOT] BETWEEN <*expression*> AND <*expression*>
- <*in predicate*> is:

<*expression*> [NOT] IN { <*expression*> | <*value list*> }

- <*like predicate*> is:
  <*column name*> [NOT] LIKE <*search pattern*>
- <*quantified comparison predicate*> is:
  <*expression*> <*comparison operator*> { ANY | SOME } *subquery*
- <*all condition*> is:
  <*expression*> <*comparison operator*> [NOT] ALL (<*subquery*>)
- <*exists predicate*> is:
  WHERE [NOT] EXISTS (<*subquery*>)
- <IN comparison with subquery> is:
  WHERE <*expression*> [NOT] IN (<*subquery*>)
- <EXISTS predicate> is:
  WHERE [NOT] EXISTS (<*subquery*>)
- <*join condition*> is:
  <*column expression*> <*comparison operator*> <*column expression*>

These column expressions are generally qualified by table name or alias. Column expressions cannot include asterisks (*).

Restrictions

The WHERE clause does not support Notes rich text fields. Notes does not allow formulas on rich text fields. Since the Notes driver passes the search condition to Notes, and Notes is unable to evaluate the formula, no rows are returned.

FROM

The FROM clause specifies the table names or views that are the source tables for a query.

Syntax

<FROM clause> is FROM {<*table reference*> [<*alias*>] } [, ...]

See Conventions

Example

Arguments

<*table reference*> is the name of a table name or view name.
<*alias*> is an alias for the table name or view name. The alias is defined only for this query.

UPDATE Positioned

The positioned Update statement updates the last row fetched.

Syntax

UPDATE <*table name*> SET <*column name*> = <*value expression*> [, ...] WHERE CURRENT OF <*cursor name*>

See Conventions

Example

Arguments

<*table name*> is the name of the table to update.

<*column name*> is the name of a column in the table.

<*value expression*> is the new value for the column. This must be either a value expression or a dynamic parameter.

<*cursor name*> is the name of the cursor pointing to the row to update.

UPDATE Searched

The searched UPDATE statement updates values in selected rows of a table.

Syntax

UPDATE <*table name*> [<*alias*>] SET <*column name*> = <*expression*> [, ...][<where clause>]

See Conventions

Example

Arguments

<*table name*> is the name of the table to update. Cannot be used if <*alias*> is used.
<*alias*> is an optional alias for the table. Cannot be used if <*table name*> is used.
<*column name*> is the name of a column in the table.
<*expression*> is an expression that evaluates to a new value for the column.
<where clause> identifies which rows will be updated (see SELECT).

INSERT

The INSERT statement adds a row to a table or view. You can specify values explicitly or obtain them from a SELECT statement. The value you assign to a column must be compatible with the column's data type. If you do not specify a value a default is assigned if one is available.

Syntax

INSERT INTO <*table name*>[(<*column list*>)]{ VALUES (<*value list*>) | <query specification>}

See Conventions

Example

Arguments

<*table name*> is the name of the table to insert into.

<*column list*> is an optional list of the columns to update. If you do not specify a column list, then values are assigned to columns in the order in which they appear in the definition of the table. You must separate column names with commas and enclose the entire list in parentheses.

<*value list*> is a list of values to insert into the table. You must separate values with commas and enclose the entire list in parentheses. Each value must be one of the following:

- A quoted string constant
- A numeric constant
- A dynamic parameter
- NULL

The INSERT statement must contain either a VALUES clause or a SELECT clause.

Input Validation Functions

When the Notes driver inserts a new record into a table, it evaluates three functions in the form design for every field (Notes does the same thing when you compose a document):

- Default value
- Input translation
- Input validation

The field definition can include any combination of these functions or none of them. These functions are evaluated in the following order:

1. The default value formula (if it exists) is evaluated to supply a value when the INSERT statement contains no data for the field.
2. The input translation formula is evaluated.

   Note The input translation formula could change the value of the inserted data.
3. The input validation formula is evaluated.

When an INSERT statement involves multiple records, and one or more of the insertions fails an input validation check, the return code from SQLExecute or SQLExecDirect is SQL_SUCCESS_WITH_INFO. This tells the user what failed and allows valid insertions to continue.

DROP VIEW

The DROP VIEW statement deletes a view from the current database.

Syntax

DROP VIEW <*view name*>

See Conventions

Example

Arguments

<*view name*> is the name of the view to delete.

Restrictions

- RESTRICT and CASCADE are not supported.
- Dependent objects and documents are not dropped.

DROP TABLE

The DROP TABLE statement deletes a table from the current database.

Note: The DROP TABLE statement does not delete data from the database. For more information, see Understanding the Universal Relation Table.

Syntax

DROP TABLE <base table name>

See Conventions

Example

Arguments

<*base table name*> is the name of the table to delete.

Restrictions

- RESTRICT and CASCADE are not supported.
- Dependent objects and documents are not dropped.

DROP INDEX

The DROP INDEX statement deletes an index from the current database.

Syntax

DROP INDEX <index name>

See Conventions

Example

Arguments

<*base table name*> is the (optional) name of the table on which the index was defined.
<*index name*> is the name of the index to delete.

Restrictions

- RESTRICT and CASCADE are not supported.
- Dependent objects and documents are not dropped.

DELETE Positioned

The positioned DELETE statement deletes the last row fetched (the current row).

Syntax

DELETE FROM <*table name*> WHERE CURRENT OF <*cursor name*>

See Conventions

Example

Arguments

<*table name*> is the name of the table or view from which to delete a row.
<*cursor name*> is the cursor pointing to the row being processed.

DELETE Searched

The searched DELETE statement deletes selected rows from a table.

Syntax

DELETE FROM <*table name*> [WHERE <where clause>]

See Conventions

Example

Arguments

<*table name*> is the name of the table where you want to delete data.

<where clause> specifies the rows to delete. If you do not specify a WHERE clause, all rows in the table are deleted.

SELECT

The SELECT statement selects rows and columns from tables either for display or as input to other SQL statements.

Syntax

SELECT <fullselect> [<ORDER BY clause> | <FOR UPDATE clause>]

See Conventions

Arguments

<fullselect> is one of the following:

- <SELECT clause>
- < FROM clause>
- [<WHERE clause>]
- [<GROUP BY clause>]
- [<HAVING clause>]

<SELECT clause> is SELECT {* | <*expression list*> }

<expression list> is a list of column names and other <expressions> whose values will appear in the result table (result set).

Restrictions

- The UNION operation is not supported.
- The DISTINCT clause is not supported in a Column function

CREATE VIEW

The CREATE VIEW statement defines a new view in the current database.

Syntax

CREATE VIEW <*view name*> [(<*column list*>)] AS <query specification>

See Conventions

Example

Arguments

<*view name*> is the name of a new view.
<*column list*> is a comma-delimited list of column names.

Restrictions

- Only single base tables (no view on view) are supported.
- Select columns cannot be Column functions.
- No GROUP BY or HAVING clauses are supported in a query specification.
- No nested queries are supported in a view definition.

CREATE TABLE

The CREATE TABLE statement defines a new table in the current database.

Syntax

CREATE TABLE <*base table name*> (<*column name*> <*datatype*> [DEFAULT <default value>] [CHECK <search condition>])

See Conventions

Example

Arguments

<*base table name*> is the name of the table to create.

<*column name*> is the name of a column.

<*datatype*> is one of the following:

- CHAR(n) or CHARACTER(n) where 1 <= n <= 254
- DECIMAL(p, s) where p (precision) is 1 <= p <= 15 and s (scale) is 0 <= s <= p
- NUMERIC(p, s) (same as DECIMAL)
- SMALLINT
- INTEGER
- REAL
- FLOAT
- DOUBLE PRECISION
- VARCHAR(n) or CHARACTER VARYING(n) (same as CHAR)
- DATE (format YYYY-MM-DD)
- TIME (format HH-MM-SS)
- TIMESTAMP (format YYYY-MM-DD HH:MM:SS)

Note The ODBC shorthand escape sequences {d 'yyyy-mm-dd'}, {t 'hh:mm:ss'} and {dt 'yyyy-mm-dd hh:mm:ss'} are not supported.

Note The millisecond portion of the TIMESTAMP value is not supported.

Restrictions

- NULL and NOT NULL are not supported.
- UNIQUE and PRIMARY KEY are not supported.
- The REFERENCES clause is not supported.
- A table constraint definition is not supported.

CREATE INDEX

The CREATE INDEX statement defines an index for a table.

Syntax

CREATE INDEX <*index name*> ON <*base table name*>(<*column name*> [ASC | DESC] [, <*column name*> [ASC | DESC]] ...)

See Conventions

Example

Arguments

<*index name*> is the name of the index to create.

<*base table name*> is the name of the table to index.

<*column name*> is the name of a column to include in the index. The entire list of column specifications is enclosed in parentheses and items are separated by commas. The column data type cannot be VARCHAR.

ASC | DESC specify the order of the index, either ascending (ASC) or descending (DESC). Ascending is the default sort order.

Restrictions

No unique index.

Exceptions to ODBC SQL Grammar

The Lotus Notes ODBC driver supports most SQL statements and clauses in the ODBC Minimum and Core grammar. The following table describes exceptions:

Grammar	Limitation
ALTER TABLE	The following keywords are not supported: NULL NOT NULL PRIMARY KEY REFERENCES No table constraint definition
CREATE INDEX	The UNIQUE keyword is not supported
CREATE TABLE	The following keywords are not supported: NULL NOT NULL UNIQUE PRIMARY KEY REFERENCES No table constraint definition
CREATE VIEW	Only single base table (no view on view) SELECT columns cannot be Column functions No GROUP BY clause or HAVING clause in a query specification No nested queries in a view definition
RESTRICT or CASCADE	Not supported in DROP INDEX, DROP VIEW, and DROP TABLE statements. Dependent objects and document are not dropped.
GRANT and REVOKE	Not Supported. All access control is handled implicitly by Notes.
UNION	Not supported in the SELECT statement
DISTINCT	Not supported in select clause or in a Column function
DATE, TIME, and TIMESTAMP ODBC escape clauses	The ODBC shorthand escape sequences {d 'yyyy-mm-dd'}, {t 'hh:mm:ss'} and {dt 'yyyy-mm-dd hh:mm:ss'} are not supported.

Rich Text Fields

The Lotus Notes driver returns only the text part(s) of a rich text field. The Notes driver cannot create a rich text field.

The WHERE clause does not support Notes rich text fields. Notes does not allow formulas on rich text fields. Since the Notes driver passes the search condition to Notes, and Notes is unable to evaluate the formula, no rows are returned.

List Fields (multi-valued fields)

The Lotus Notes driver supports multiple values in fields. The Notes driver contains information about which fields can have multi-values and the underlying data type for these values.

The Notes driver does not support list fields in DDL statements (CREATE TABLE, DROP TABLE, ALTER TABLE, CREATE INDEX, CREATE VIEW, or DROP VIEW).

Use the following notation in DML statements (SELECT, INSERT, UPDATE searched, UPDATE positioned, DELETE searched, DELETE positioned) to specify a list of values:

'string;string;string'

Text fields

The Notes driver returns all the data in multi-valued fields composed of text as a single string, with items separated by semicolons. For example:

'a;b;c'

Note: An extra semicolon can appear in the result if, when the information was entered in Notes, any character other than a semicolon were used as the list separator and the list contains an item which includes a semicolon.

The Notes driver accepts a semicolon-delimited list of strings for insertion in a multi-valued field composed of text. For example:

'a;b;c'

This value creates a list in the Notes document if the field allows multiple values. If the field does not allow multiple values, the value appears in Notes as a single string. Data retrieved through the Notes driver produces the same result in either case.

Numeric and date fields

The Notes driver returns only the first value in the list if the multi-valued fields are numeric or date fields.

The Notes driver accepts only a single numeric value for insertion in a multi-valued field that is numeric.

SQL Statements

The Lotus Notes driver supports most ODBC Minimum and Core level SQL syntax. Restrictions and exceptions are described for each statement.

Summary of Supported ODBC SQL Grammar

Exceptions to ODBC SQL Grammar

See also

Using the Lotus Notes ODBC Driver

Connection Strings

The following keywords are supported in an SQLDriverConnect call:

Keyword	Description
DSN	The name of the data source
Database	The name of the Notes database, with a path if necessary
Server	The name of the Notes server where the database is located If the database is on the local workstation leave the field blank.

For example, to connect to the Personnel data source in the directory C:\PERSONNEL on server HR_1, use the following connection string:

DSN=Personnel; Database=C:\Personnel\employee.nsf; Server=HR_1

Data Types

The following table shows how Lotus Notes data types are mapped to ODBC SQL data types. In addition to the SQL data types Notes supports two additional data types:

- List fields
- Rich text fields

SQL to Notes Data Type Mapping

ODBC SQL Data Type	Lotus Notes Data Type
SQL_CHAR	Text
SQL_VARCHAR	Text
SQL_LONGVARCHAR	Notes Rich Text
SQL_DECIMAL	Number
SQL_NUMERIC	Number
SQL_SMALLINT	Number
SQL_INTEGER	Number
SQL_REAL	Number
SQL_FLOAT	Number
SQL_DOUBLE	Number
SQL_DATE	Time
SQL_TIME	Time
SQL_TIMESTAMP	Time

Note SQLGetTypeInfo returns ODBC SQL data types. All conversions in Appendix D of the *Microsoft ODBC SDK Programmer's Reference* are supported for the ODBC SQL data types listed above.

Notes to SQL data type mapping

Lotus Notes data type	ODBC SQL data type
Number	SQL_FLOAT
Time	Depending on format this can be SQL_TIME, SQL_DATE, or SQL_TIMESTAMP
Text	SQL_VARCHAR
Keyword	SQL_VARCHAR
Multi-value list	SQL_VARCHAR
Rich text field	Text portion only, as SQL_LONGVARCHAR
Section	Not supported

Error Messages

Error messages can originate in one of three layers:

- The Driver Manager layer traps any incorrect sequence of ODBC API calls and other invalid values.
- The Lotus Notes ODBC and SQL engine layer traps any error for processing an SQL statement.
- The Lotus Notes internal layer traps error messages returned from a Notes API call.

Error messages have the following format:

[vendor][ODBC-component][data source]message-text

The prefixes in brackets ([ ]) identify the location of the error.

The following table shows the format of error messages returned by the Driver Manager, Lotus Notes ODBC and SQL Engine layer, and Lotus Notes internal layer:

Error message	**Error location**
[Microsoft][ODBC DLL]message-text	Driver Manager (ODBC.DLL)
[Casahl/Lotus][ODBC Lotus Notes]message-text	Lotus Notes Driver (NOTESSQL.DLL)
[Casahl/Lotus][ODBC Lotus Notes][Lotus Notes Server]Notes API error:message-text	Notes API DLLs

Errors occurring in the Notes driver or the Driver Manager

The driver returns an error message with the SQLSTATE when the error occurs in the ODBC Lotus Notes driver or the Driver Manager. When the error occurs in the ODBC Lotus Notes internal layer, the [vendor] and [ODBC-component] prefixes identify the vendor and name of the ODBC component that received the error from the data source.

Errors occuring in the data source

For errors that occur in the data source, the Notes driver returns an error message with SQLSTATE S1000 returned by the Notes API Call. When the error occurs in the Driver Manager, the Notes driver, or the SQL engine layer, the data source is not given.

ALTER TABLE

The ALTER TABLE statement adds one or more columns to a table.

Syntax

> ALTER TABLE <base table name>{ ADD <column name> <data type>| ADD (<column name> <data type> [, <column name><data type>]... }
>
> See Conventions

Example

Arguments

<*base table name*> is the table to be altered.

<*column name*> is the name of the column to be added.

<*datatype*> is one of the following:

- CHAR(n) or CHARACTER(n) where 1 <= n <= 254
- DECIMAL(p, s) where p (precision) is 1 <= p <= 15 and s (scale) is 0 <= s <= p
- NUMERIC(p, s) (same as DECIMAL)
- SMALLINT
- INTEGER
- REAL
- FLOAT
- DOUBLE PRECISION
- VARCHAR(n) or CHARACTER VARYING(n) (same as CHAR)
- DATE (format YYYY-MM-DD)
- TIME (format HH-MM-SS)
- TIMESTAMP (format YYYY-MM-DD HH:MM:SS)

Note The ODBC shorthand escape sequences {d 'yyyy-mm-dd'}, {t 'hh:mm:ss'} and {dt 'yyyy-mm-dd hh:mm:ss'} are not supported.

Note The millisecond portion of the TIMESTAMP value is not supported.

Restrictions

- NULL or NOT NULL is not supported.
- PRIMARY KEY is not supported.
- The REFERENCES clause is not supported.
- A table constraint definition is not supported.

The following table lists the conventions used to describe the syntax for SQL statements.

Convention	Description
argument	Information that the application must provide.
SQLTransact	Syntax that must be entered exactly as shown, including function names.
[ ]	Optional items or, if in bold text, brackets that must be included in the syntax.
|	Separates two mutually exclusive choices in a syntax line.
{ }	Delimits mutually exclusive choices in a syntax line.
...	Arguments that can be repeated several times.
user input	In examples, indicates information that the user must provide.

A table name can be up to 32 characters long and must not be the same as the name of another table or view in the database. If you enclose the table name in quotation marks (" "), it can contain any characters including blanks. Otherwise, table names can consist only of letters, digits, underscores (_) and dollar signs ($), must begin with a letter, and cannot be the same as any SQL reserved word.

A column name can be up to 32 characters long and must not be the same as the name of another column in the table. Column names can consist of letters, digits, underscores (_) and dollar signs ($), and must begin with a letter. Don't use an SQL reserved word as a column name.

An index name can be up to 32 characters long and must not be the same as the name of another index created by this user on this table. If you enclose the index name in quotation marks (" "), it can contain any characters including blanks. Otherwise, index names can consist only of letters, digits, underscores (_) and dollar signs ($), and must begin with a letter. Don't use an SQL reserved word as an index name.

A view name can be up to 32 characters long and must not be the same as the name of another table or view in the database. If you enclose the view name in quotation marks (" "), it can contain any characters including blanks. Otherwise, view names can consist only of letters, digits, underscores (_) and dollar signs ($), and must begin with a letter. Don't use an SQL reserved word as a view name.

<*expression*> is any combination of constants, Column functions, and column names.

<*comparison operator*> is one of the following:

Operator	**Meaning**
<	Less Than
>	Greater Than
<=	Less Than or Equal
>=	Greater Than or Equal
=	Equal
<>	Not Equal

<*search pattern*> is either a string constant, the keyword USER, or a dynamic parameter. String constants must be enclosed in single (' ') or double (" ") quotation marks and can include wildcard characters.

```
UPDATE INVENTORY SET QTY = 100.00 WHERE CURRENT OF SQL_CUR_0
```

```
UPDATE INVENTORY SET UNITCOST=UNITCOST * 1.2
UPDATE STAFF SET COMMISSION = (COMMISSION * 1.25) WHERE HIREDATE < '1982-07-05'
```

```
SELECT LOCATION, AVG(UNITCOST) FROM INVENTORY GROUP BY LOCATION ORDER BY
LOCATION DESC
```

```
SELECT LOCATION, AVG(UNITCOST) FROM INVENTORY GROUP BY LOCATION HAVING
AVG(UNITCOST) > 600
```

```
SELECT LOCATION, AVG(UNITCOST) FROM INVENTORY GROUP BY LOCATION
```

```
Select * FROM INVENTORY WHERE LOCATION='New York' FOR UPDATE OF ON_HAND
```

```
SELECT * FROM CUSTOMER WHERE CITY='New York

SELECT PART_NO, DESCRIPT, ON_HAND, LOCATION, UNITCOST FROM INVENTORY WHERE
ON_HAND > 50

SELECT PART_NO, DESCRIPT, LOCATION, ON_HAND, UNITCOST FROM INVENTORY WHERE
LOCATION = 'Los Angeles' OR LOCATION = 'New York' AND ON_HAND < 20 AND UNITCOST < 1000
```

```
SELECT * FROM CUSTOMER
```

```
INSERT INTO STAFF VALUES ('000001', 'Zambini', 'Rick', '1980-02-15', 'Los Angeles', '000000', 6000, 5.0)

INSERT INTO NEWSALTAB SELECT SALARY, LASTNAME FROM STAFF,SALES WHERE STAFF.STAFF_NO = SALES.STAFF_NO AND SALARY BETWEEN 5500 AND 6000
```

DROP VIEW NYCUST

```
DROP TABLE NEWCUST
```

```
DROP INDEX CUSTNDX
```

```
DELETE FROM INVENTORY WHERE CURRENT OF SQL_CUR_0
```

```
DELETE FROM NYCUST WHERE COMPANY='Interior Designs'
```

```
CREATE VIEW NYCUST (COMPANY,ADDRESS) AS SELECT COMPANY,ADDRESS FROM
CUSTOMER WHERE STATE='NY'
```

```
CREATE TABLE STAFF
        (STAFF_NO CHAR(6),
         LASTNAME CHAR(15),
         FIRSTNAME CHAR(10),
         HIREDATE DATE,
         LOCATION CHAR(15),
         SUPERVISOR CHAR(6),
         SALARY FLOAT,
         COMMISSION FLOAT)
```

```
CREATE INDEX CUSTNDX ON CUSTOMER (LASTNAME ASC)
```

```
ALTER TABLE CUSTOMER ADD COUNTRY VARCHAR(20)
```