# SQL Optimizer User's Guide

## MetaCube ROLAP Option

for Informix Dynamic Server

Microsoft Windows Environments

Version 4.1
December 1998
Part No. 000-5224

Published by INFORMIX® Press

Informix Corporation
4100 Bohannon Drive
Menlo Park, CA 94025-1032



Documentation Team:

# Table of Contents

# In This Guide

This book describes how to use the MetaCube SQL Optimizer software module. This software is designed to connect third-party query tools or custom query applications to the MetaCube analysis engine so that these applications can access a MetaCube data warehouse. The SQL Optimizer implements generation of optimized SQL statements for querying aggregate tables and sample tables in the MetaCube data warehouse.

With the ability to do this, performance during query processing can be noticeably improved for users of very large data warehouses. In addition, the MetaCube analysis engine provides powerful capabilities that simplify the presentation of meaningful, business-related reports for end users.

## About MetaCube

The MetaCube analysis engine retrieves data from a data warehouse in multidimensional terms. Retrieved data is formatted into reports where rows, columns, and pages of data reflect a user's view of the business data stored in the database. By manipulating data contained in flat two-dimensional result sets, MetaCube organizes information into tabular formats that present information in meaningful reports that correspond to an end user's view of a business.

The key to presenting data in this way is MetaCube metadata. Metadata is structured to provide a model of how users think about data. Metadata contains mappings to multiple dimension tables and dimension attribute tables that provide the information needed by the MetaCube analysis engine to organize retrieved data into business-related user views.

Metadata is created using MetaCube Warehouse Manager, a client application that allows a data warehouse administrator to enter information about the physical tables in a data warehouse. The information entered reflects the previously defined logical, multidimensional model of the data warehouse.

In addition to generating multidimensional reports from the data warehouse, the MetaCube analysis engine can perform subtotal and total calculations and embed these figures into a report. Subtotals and grand totals are calculated according to the orientation of data within the report. The MetaCube analysis engine also provides advanced calculations such as comparisons and rankings that cannot be performed using SQL. The MetaCube analysis engine enhances the information returned to end users by embedding the results of these calculations in a report.

MetaCube metadata contains information that reflects a predefined hierarchy for attributes for each dimension of the data warehouse. This dimension hierarchy enables drilling up, down, and across through data. The MetaCube analysis engine can retrieve more detailed information (when the user drills down) or more summarized information (when the user drills up).

Other analysis features enabled through MetaCube metadata include:

- balloon help for the end user that describes the dimensions and attributes available for querying.
- sorting capabilities for all data retrieved.
- error margins for all data retrieved from sample tables.

The MetaCube analysis engine also enforces a security mechanism for controlling user access to the data warehouse.

## About SQL Optimizer

The MetaCube SQL Optimizer is a library of routines that enables optimization of SQL generated by a query tool or custom query application other than MetaCube, so that the end user's query can run against a MetaCube data warehouse.

SQL Optimizer receives standard SQL, parses and analyzes it and, using MetaCube metadata, dimensionalizes it. SQL Optimizer then generates and submits a MetaCube-like query to the MetaCube analysis engine. In turn, the MetaCube analysis engine returns standard SQL that is "aware" of the aggregate and sample tables in the data warehouse. SQL Optimizer submits the optimized SQL, via the MetaCube the mcodbcdr and the passthrough drivers, to the MetaCube data warehouse. The results are returned to the query tool or application.

Using SQL Optimizer, a non-MetaCube query application can access a MetaCube data warehouse in the following ways:

- From any packaged SQL-generating query tool that uses an ODBC interface to access the database
- From a custom SQL-generating application using a set of MetaCube Data Interface function calls
- From a custom SQL-generating application using a set of MetaCube Data Interface function calls and native ESQL-C connectivity

## Database Connection

Using a packaged query tool that communicates through an ODBC driver, the connection to SQL Optimizer occurs as shown in Figure 1.

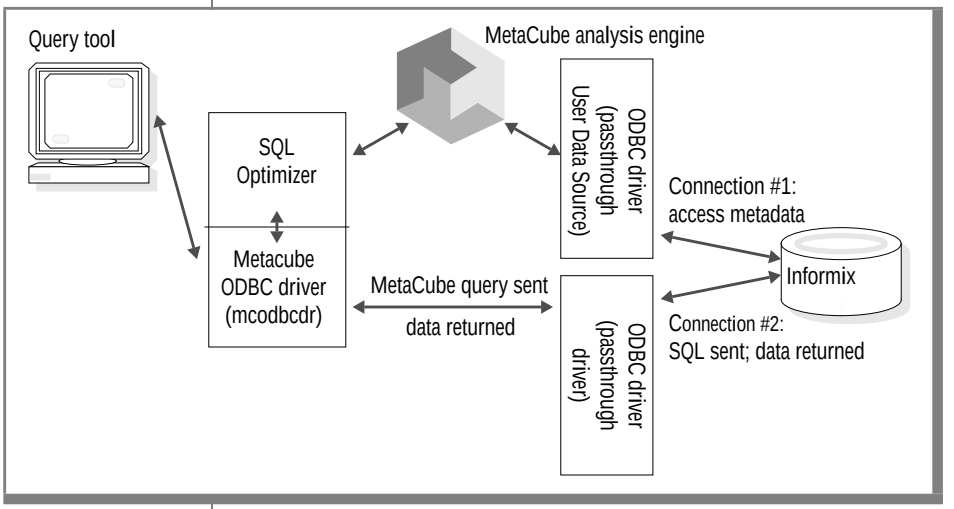


***Figure 1*** *Connections Using SQL Optimizer's ODBC Interface*

This architecture requires two ODBC connections to the Informix database where the data warehouse resides. One connection (the ODBC passthrough User Data Source) is used by the MetaCube analysis engine to access metadata. The second connection (the passthrough driver) is used by the query tool to send SQL and receive results, via the SQL Optimizer.

Using a custom query application that sends MetaCube Data Interface (MDI) function calls directly to the SQL Optimizer, an architecture using native database ODBC connectivity can be implemented, as shown in Figure 2.

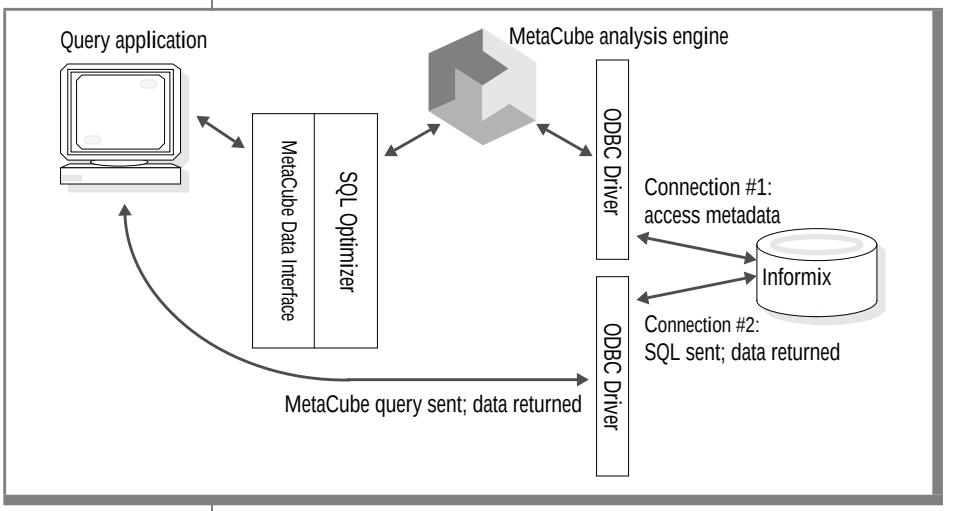


***Figure 2*** *Query Application Using MDI API Function Calls*

A third configuration option is also available when using a custom query application. In this case, the connection between the application and the Informix database is through native Informix connectivity, using ESQL-C. This implementation is shown in Figure 3.

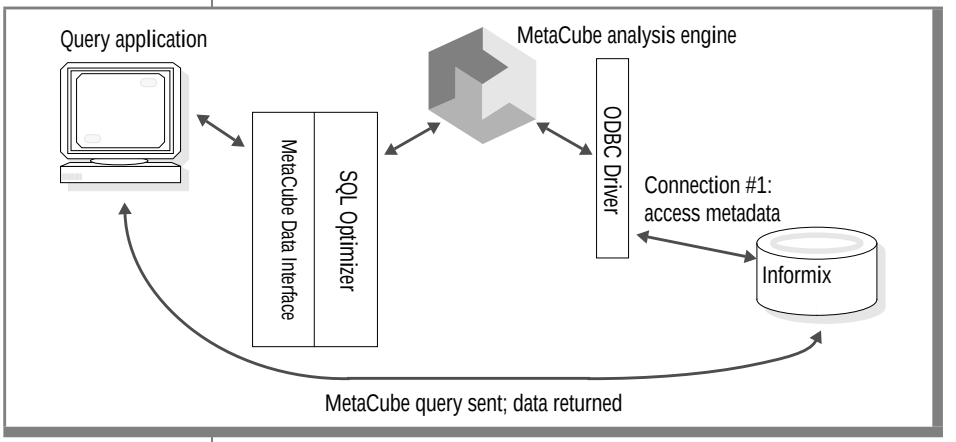


***Figure 3*** *Connections Using Informix Native Connectivity*

This architecture requires the standard ODBC connection used by the MetaCube analysis engine to access the database. The second connection, using Informix native connectivity through ESQL-C, is used to send SQL, obtained from the SQL Optimizer, and to receive results.

For these configurations, SQL Optimizer, the MetaCube analysis engine, and the ODBC drivers all reside on the same PC.

The MetaCube analysis engine can reside on a PC other than the one where the query application or tool resides. This PC is the middle tier in the MetaCube three-tier architecture. In this case:

- SQL Optimizer must reside on the same PC where the query tool resides. If the query application uses ODBC for connection to the database, that ODBC driver must also reside on the same PC.
- the MetaCube analysis engine and the ODBC driver it uses to access metadata must reside on the middle-tier PC.

For information on MetaCube's three-tier architecture, refer to the *MetaCube Data Warehouse Administrator's Guide*.

## ODBC Access to SQL Optimizer

The MetaCube installation program installs SQL Optimizer in the system area of the machine where it will run. When SQL Optimizer is installed, the MetaCube installation program makes entries in the machine's registry that are used when ODBC is configured to access the SQL Optimizer.

To access MetaCube SQL Optimizer through an ODBC connection, you must define an ODBC User Data Source for it using ODBC Administrator. The ODBC driver that is the basis for the configuration is mcodbcdr, the MetaCube ODBC driver. This ODBC driver provides its own setup procedures.

When defining the ODBC User Data Source for SQL Optimizer, you will need to provide the following MetaCube-related information:

- **Data Source Name**: any arbitrary name.

- **Metamodel Schema**: the schema / owner of the MetaCube data warehouse metadata tables. Usually, the owner name is "metacube." Note that the owner name must be followed by a period ( . ).
- **Passthrough Data Source**: name of the ODBC User Data Source used by the MetaCube analysis engine to access the data warehouse.
- **Passthrough Driver**: filename of the ODBC driver used by the query application to access the database. The default Passthrough Driver is the full pathname of the Informix CLI driver, ivinf709.dll, for accessing Informix databases; it typically resides in the system area.
- **DSS System Name**: name of the DSS System to access after connection to the database.
- **Log File**: the full pathname of the log file to be generated by SQL Optimizer. This file captures the SQL sent by the query application and the optimized SQL generated by SQL Optimizer. If you only specify a filename in this field, the log file is placed in the directory where the query application resides.
- **Enable Logging**: enables / disables the logging feature. When this box is checked, logging is enabled.
- **Database Type**: the supported database type is Informix.
- **Sampling**: enables / disables the use of sampling for queries. When this box is checked, Sampling is enabled and retrieved query results are statistically accurate sampled data. When this box is unchecked, the Sampling feature is disabled.
- **Confidence**: used to set a value between 1 and 100 that determines the margin of error for a sampled report. The value 100 has the effect of disabling the Sampling feature. A value less than 100, causes the MetaCube analysis engine to:
  - use Sampling when retrieving results, whenever the Sampling feature is enabled in the query application.
  - return results with a margin of error consistent with the setting; the higher the value of this setting, the wider the range for the margin of error.

- **Accuracy**: used to set a value between 1 and 100 that determines which sample table to use for retrieving results. The value 100 has the effect of disabling the Sampling feature. A value less than 100 causes the MetaCube analysis engine to:
  - use Sampling when retrieving results, whenever the Sampling feature is enabled in the query application.
  - return results that correlate to the Accuracy number. A high Accuracy value returns more data; a low Accuracy value returns less data.

  For more information on MetaCube's Sampling feature, see the *MetaCube Data Warehouse Administrator's Guide*.
- **Use MetaCube SQL OPT Server (MCServer)**: This option is used only for the Cognos Impromptu, Version 4.0, query tool. If this is the query tool in use, then click this button.

***Tip:*** *This option is needed only for a two-tier architecture. If your configuration is a three-tier architecture, with the MetaCube analysis engine running on the middle tier, MCServer is not needed and it is not necessary to enable this option.*

## What SQL Optimizer Does

When SQL Optimizer receives SQL, it parses it, analyzes it, and translates it to conform syntactically to what the MetaCube analysis engine expects to receive from a query tool. That is, SQL Optimizer turns the SQL into a multidimensional query that is recognized by the MetaCube analysis engine. For example, certain SQL statements are constraints on data retrieved; SQL Optimizer recognizes these and translates the clauses into syntax that the MetaCube analysis engine recognizes as attribute or measure filters. SQL Optimizer temporarily defines these so-called filters, which are deleted after SQL generation is complete.

In turn, the MetaCube analysis engine refers to metadata to map the multidimensional request into SQL that is based on the actual physical tables in the data warehouse. This SQL is optimized to access aggregate tables and sample tables, where appropriate. The MetaCube analysis engine determines which aggregate tables and sample tables will facilitate the fastest retrieval of data.

## What the MetaCube ODBC Driver Does

The MetaCube ODBC driver, mcodbcdr, provides ODBC support. For the ODBC calls that involve SQL queries, the driver interacts with SQL Optimizer to optimize the SQL, which is then used to query the MetaCube data warehouse. The following ODBC calls are used for this:

- SQLExecDirect
- SQLPrepare

The MetaCube ODBC driver initializes SQL Optimizer during the connecting period. This is done within one of the following ODBC calls:

- SQLConnect
- SQLDriverConnect
- SQLBrowseConnect

SQLDisconnect terminates SQL Optimizer. SQLGetInfo provides some information related to the MetaCube ODBC driver.

The MetaCube ODBC driver hands all other ODBC calls directly over to the native database ODBC driver, called the PassThrough driver. The MetaCube ODBC driver supports the interface for all ODBC calls.

## SQL Support by SQL Optimizer

The information in this section will help you to design SQL queries that can be successfully optimized. Although no error will occur, unoptimized SQL will undoubtedly run against the large fact table rather than aggregate and/or sample tables when they exist in the data warehouse. Thus, the performance advantages inherent in the MetaCube data warehouse will not be realized.

The SQL submitted to the SQL Optimizer must be syntactically correct. SQL Optimizer does basic verification of all components of the SQL statements submitted to it. For example, it verifies the SELECT clause to ensure that column names are valid and unambiguous and join relationships are qualified and valid. It also verifies that all column names exist unambiguously in the tables listed in the FROM clause. The SELECT list itself must contain columns that are attributes, dimension element keys, or measures.

***Important:*** *SQL Optimizer supports single column keys.*

A simplified example of decision support SQL query is shown below:

```
SELECT T.A, SUM(F.M)
FROM T,F
WHERE T.C=F.C AND
    T.A=<attribute or dimension element filter condition>
GROUP BY T.A
HAVING SUM(F.M) <operator with measure filter condition>
```

In this example:

- `T.A` represents one or more attribute name.
- `F.M` represents one or more measure name contained in the fact table, to which an aggregate function (usually SUM) is applied.
- `T.C=F.C` represents one or more join between the fact table and the dimension or attribute table.

From this example, you can see that the WHERE clause is the basis for an attribute filter. The GROUP BY clause is required. The HAVING clause is the basis for a measure filter.

The WHERE clause may contain join relations and attribute or dimension element key column filters. For dimension elements, SQL Optimizer supports single-column values. Complex filters such as simple OR constructs that compare values in the same column are supported. An OR statement, such as “col_1 = 4 OR col_1 = 8”, is converted to an IN clause—“col_1 IN (‘4’, ‘8’)”. The use of more complex OR constructs causes the entire SQL statement to be passed through without change. Filter operators such as BETWEEN, LIKE, NOT LIKE, IN, IS NULL, IS NOT NULL, and all comparison operators are supported.

The SQL aggregate functions SUM, AVG, MAX, MIN, and COUNT are all supported by SQL Optimizer.

After the MetaCube analysis engine has generated its SQL, that SQL is returned to SQL Optimizer and verified. SQL Optimizer reorders the SELECT list, if necessary, since it may not have been returned from the MetaCube analysis engine in the same order as it was originally received by SQL Optimizer.

ORDER BY statements are withheld by SQL Optimizer and reinserted into the SQL that is returned from the MetaCube analysis engine.

Calculations contained in the received SQL are compared with existing MetaCube metadata constructs known as calculated measures. If an SQL calculation does not match an existing MetaCube calculated measure, SQL Optimizer generates a temporary calculated measure whose syntax is recognizable by the MetaCube analysis engine. The temporary calculated measure is deleted after the SQL generation process has completed. If a calculated measure cannot be formed, the entire SQL statement is passed through, as is.

***Important:*** *You should use Warehouse Manager to register frequently-used calculations as MetaCube calculated measures. The, SQL Optimizer can map the calculations quickly and does not need to create temporary ones.*

SQL table and column aliases are supported by SQL Optimizer. Measure labeling is also supported.

## The MetaCube Data Interface

You can incorporate into your custom query application direct API function calls to SQL Optimizer. SQL Optimizer API calls are described in this section.

***Important:*** *The MetaCube ODBC Driver conforms to ODBC API 2.5. Refer to Microsoft ODBC documentation for information about ODBC Driver APIs.*

**BOOLEAN MetaCubize(LPCTSTR inSql, LPCTSTR outSql, int* pMaxLen);**

### Description

MetaCubize( ) takes an input SQL select statement, *inSql*, and transforms it into a new SQL statement, *outSql*, that is aggregate table-aware and sample table-aware. The variable pMaxLen points to an integer that specifies the *outSql* buffer size.

### Return Value

If successful, MetaCubize( ) returns TRUE. Otherwise, it returns FALSE.

**BOOLEAN VBMetaCubize(LPCTSTR *inSql*, BSTR **pOutSql*, int* pMaxLen);**

### Description

VBMetaCubize( ) is the Visual Basic version of MetaCubize( ).

**BOOLEAN MCInitialize(LPCTSTR *connectString*,**
**LPCTSTR *dssName*,**
**LPCTSTR *login*,**
**LPCTSTR *password***
**int *connectDatabase***
**LPCTSTR *metaSchema*);**

### Description

MCInitialize( ) establishes connection to the MetaCube analysis engine and causes the engine to connect to an ODBC User Data Source specified by the parameters in the call. The parameters are:

- *connectString*: name of the User Data Source to be used for connection.
- *dssName*: DSS System name to which to connect.
- *login*: login ID of the database user.

- *password*: password of the database user.
- *connectDatabase*: type of database.
- *metaSchema*: schema / owner of metadata tables in the database.

### Return Value

If successful, MCInitialize( ) returns TRUE. Otherwise, it returns FALSE.

**BOOLEAN MCExecute(LPCTSTR *command*, LPCTSTR *param*, LPCTSTR *result*, int* pMaxLen);**

### Description

MCExecute( ) takes a command with parameters and executes it against the DSS System to which it is currently connected and puts the returning value, if any, in *result*. The maximum length of the result is specified by an integer pointed to by pMaxlen. The actual length of the result is the value pointed to by pMaxLen when MCExecute( ) returns TRUE.

The commands include the following:

- *connect*
- *disconnect*
- *help*
- *metacubize*

***Tip:*** *Metacubize operations can be done either through MetaCubize( ) or MCExecute( ) with the command "metacubize".*

### Return Value

If successful, MCExecute( ) returns TRUE. Otherwise, it returns FALSE.

**BOOLEAN VBMCExecute(LPCTSTR *command*, LPCTSTR *param*, BSTR *pResult, int* pMaxLen);**

### *Description*

VBMCExecute( ) is the Visual Basic version of MCExecute( ).

**BOOLEAN MCTerminate( );**

### *Description*

MCTerminate( ) terminates the current connection to the MetaCube analysis engine and causes the engine to disconnect from the ODBC User Data Source.

### *Return Value*

If successful, MCTerminate( ) returns TRUE. Otherwise, it returns FALSE.

**void SwitchSampling(BOOLEAN on, long *confidence*, long *accuracy*);**

### *Description*

SwitchSampling( ) switches the MetaCube Sampling option on and off according to the value of the variable *on*. When on is TRUE, SQL Optimizer uses *confidence* and *accuracy* to do Sampling. When on is FALSE, SQL Optimizer turns off Sampling.

### *Return Value*

None.

# Index

## O



## P



## S



## T



## U



## V



## W