Migrate a database from MySQL to IBM Informix Innovator-C Edition, Part 2: Step-by-step walk-through of the migration process

Skill Level: Intermediate

Sanjit Chakraborty (sanjitc@us.ibm.com)
Advisory Software Engineer
IBM

17 Feb 2011

IBM® Informix® Innovator-C Edition is free to download and deploy, and provides a strong, reliable, easy-to-administer foundation for your database applications. This tutorial is Part 2 of a series about migrating from MySQL to Informix Innovator-C Edition. Part 1 compared and contrasted MySQL and Informix Innovator-C, and examined their architectural differences. Now in Part 2, walk through a migration from MySQL to Informix, step by step. This tutorial provides a conversion methodology and discusses the processes for migrating both database objects and data. It includes a discussion of SQL differences and shows how to migrate tables, views, stored procedures, functions, triggers, and more.

Section 1. Before you start

About this series

Thinking about migrating from MySQL to Informix? If so, you're in the right spot. This migration series covers all the basics—the topics you'll need to understand before you start the migration process.

About this tutorial

This tutorial describes the database migration process from MySQL to Informix Innovator-C Edition.

Objectives

On completion of this tutorial, you should be able to:

- Understand how to install Informix Innovator-C Edition
- Know what you should consider when preparing to migrate
- Understand the Informix instance configuration process
- Migrate database objects and data

Prerequisites

This tutorial is geared toward database administrators (DBAs) familiar with basic database concepts. You should first read Part 1 to understand the differences between the products.

System requirements

You do not need a copy of Informix to complete this tutorial. However, you will get more out of the tutorial if you download the free trial version of Informix Innovator-C Edition to work along with this tutorial. (See Resources.)

Section 2. Migration considerations

Let's get started and find out how easy it is to migrate from MySQL to Informix Innovator-C Edition. A successful database migration depends on accurate planning, DBA resources, and the hardware infrastructure availability, such as disk space and CPU resources. Additional factors that can affect the success of a migration project include knowledge of the existing database objects, their sizes and relationships, and how the database and its objects are maintained on the source database server. A level of complexity is added if you take the migration as an opportunity to improve the existing design of the database (for example, by changing the storage layout). A good understanding of the various data movement methods for consistently and efficiently unloading, transforming, and loading the data is also a key factor to meet

a company's requirement for a database migration.

Here are five easy steps to consider when migrating from MySQL to Informix Innovator-C Edition:

Section 3. Get the Informix Innovator-C Edition

The Informix Innovator-C Edition provides small-to-mid-sized businesses the ability to use the most common database functionality at no cost. Comprehensive support is available as an optional purchase. Innovator-C integrates a full range of application program interfaces (APIs), including Java™ and Microsoft® .NET, and supports a broad range of integrated development environments (IDEs), such as Eclipse™, IBM Rational® Application Developer, and Visual Studio® .NET.

The Informix Innovator-C Edition can be installed on Linux®, UNIX®, Windows® or MacOS systems running with 32- or 64-bit hardware with one socket, up to four cores, 2GB of memory, and unlimited data storage. You can easily upgrade the Innovator-C Edition to the other Informix editions (for example, Choice Edition, Growth Edition, and Ultimate Edition) without modifying your database or applications.

See Resource for a link to download Informix Innovator-C Edition.

The installation process for Informix Innovator-C Edition is largely the same for UNIX and Linux. However, the installation process is a little different on Windows platform.

Installing Informix Innovator-C Edition on UNIX and Linux

Installation of Informix Innovator-C Edition on UNIX and Linux involves performing the following easy steps:

1. Log in to your system as the root user.
2. Execute the `ids_install` script to install the products. You can run the install process in GUI or console mode; the installation program runs in console mode by default, unless you choose GUI mode.
3. Follow the instructions in the installation application.
    a. Read and accept the license to proceed with the installation.
    b. Choose the **Typical** setup to install the product with all features.
    c. You can install into the default directory or choose a different directory.
    d. Select the products that you want to install, if that is an option.
    e. Optional: Choose whether to enable role separation for auditing procedures. **Note:** If you enable role separation, you cannot turn it off after the product is installed. To remove role separation, you must uninstall the database server and reinstall it without role separation.
    f. Optional: Select to create a demonstration database server instance.
    g. Verify that the installation summary accurately reflects your installation options. Go back to adjust the installation options as necessary.
4. Complete the installation and exit the installation application.

Installing Informix Innovator-C Edition on Windows

Installation of Informix Innovator-C Edition on Windows involves performing the following easy steps:

1. Log in to your system as an administrator.
2. IBM Informix Innovator-C Edition for Windows includes a Windows launchpad, which starts automatically when the CD is inserted. The launchpad runs in GUI mode by default, unless you choose console mode. Alternatively, you can use setup.exe to install the product.

3. Follow the instructions in the installation process.

    a. Select the **Install Products** option from the launchpad, and click on **Next**.

    b. Select the products that you want to install.

    c. Read the license agreement. You must accept it to proceed.

    d. Choose the **Typical** option.

    e. In the user account information, enter your *informix* user password. If user *informix* does not exist on the system, the installation wizard creates one.

    f. Specify the installation directory, if you do not want to accept the default destination. The destination drive must be in NTFS format.

    g. Verify that the installation summary accurately reflects your installation options. Go back to adjust the installation options as necessary. Otherwise, select **Next** to begin installation.

Completing the installation program loads a configured database server with a typical setup. The *informix* user account, under which the database server runs, is assigned to the *Informix-Admin* group.

Section 4. Configure Informix instance

If you have limited or no experience with Informix, it is better to create and initialize a demonstration database server instance to make the migration process less complex. This tutorial assumes that the demonstration database server instance has been created as part of the installation process.

Allocate storage space

Whether you are using MySQL or Informix, you need storage space to keep all the tables, indexes, logs, and internal system data. MySQL uses different storage spaces depending on the data requirements. It has several storage engines (for example, ISAM, MYISAM, INNODB, CSV, ARCHIVE, FEDERATED, NDB, MERGE, BLACKHOLE, and so on) to access different storage spaces. It is sometimes

complicated and confusing to determine which storage engine is appropriate to your requirement.

Maintenance of storage space is simple on Informix; you do not need to worry about any storage engine. You can dynamically allocate storage space as needed. There is no need to restart the database server when allocating new storage space. By default, the demonstration database server creates three storage spaces (rootdbs, tempsbspace, and tempdbs). You can use the rootdbs storage space for migration purposes. You might need to allocate additional space, if your database is larger than the free space available in rootdbs. You can use the `onstat -d` command to verify the free space in rootdbs.

Let's take a look at the process of allocating additional storage space.

Informix provides two storage options:

- A raw device, which is a character-special device that allows the database server to use unbuffered disk access.
- A cooked file, which is a regular file that is managed by the operating system. While the database server controls the contents of the file, it must make I/O requests to the operating system.

Before a raw device or cooked file can allocate to the instance, the ownership and permission needs to change. For simplicity, in this tutorial we use cooked files for storage spaces. Perform the following steps to create a cooked file for use by Informix.

UNIX platform:

1. Change directory to storage space location:
   `cd` *`directory`*
2. Set up a file to use:
   `touch` *`filename`*
3. Set the file permissions to 660 (rw-rw----):
   `chmod 660` *`filename`*
4. Set the group of the file to informix:
   `chgrp informix` *`filename`*
5. Set the owner of the file to informix:
   `chown informix` *`filename`*

Windows platform:

1. Log in as a member of the *Informix-Admin* group.
2. Change directory to storage space location:
   `cd directory`
3. Set up a file to use:
   `copy nul filename`

Once the cooked file is ready, you can allocate it to rootdbs by adding a new to chunk. The example in Listing 1 adds a 2GB chunk to rootdbs with a 0KB offset.

Listing 1. Add a 2GB chunk to rootdbs with a 0KB offset

```
onspaces -a rootdbs -p filename -o 0 -s 2000000
```

You can create additional storage spaces, instead of allocating space to rootdbs, for better performance. The example in Listing 2 creates a 2GB dbspace named "dbspace1" with an offset of 0KB.

Listing 2. Create a 2GB dbspace named "dbspace1" with an offset of 0KB

```
onspaces -c -d dbspace1 -p filename -o 0 -s 2000000
```

Section 5. Database schema movement

In this step you need to identify the database objects that you need to move, such as tables, indexes, constraints, views, and triggers. For moving database object from MySQL to Informix, we will consider manually doing it to illustrate important points. Manually doing things should give you a better understanding of every detail in the process, and you can notice problems immediately if things go wrong. Although this tutorial focuses on manually moving schema and data, you can also consider using the IBM Migration Toolkit (MTK) for migration from MySQL to Informix Innovator-C Edition. (See Resources for more information on the IBM Migration Toolkit.)

This *schema movement* includes extracting the Data Definition Language (DDL) definition of each object from a MySQL database. You can capture the schema of all database objects using the MySQL command shown in Listing 3.

Listing 3. MySQL - Capture the schema of all database objects

```
mysqldump --events --routines --skip-add-drop-table database
                        --no-data --skip-comments  > schema.sql
```

In this step you are capturing DDL statements only, not extracting any data from tables. You can use other `mysqldump` command options per your requirements. Once DDL extraction from MySQL is complete, you must identify whether any DDL statement requires modification due to differences in syntax between MySQL and Informix. There are different kinds of DDL statements and options available in MySQL and Informix. This tutorial compares some of the most commonly used DDL statements.

Creating a database

When creating a database in Informix, you need to define the transaction logging mode and the storage. Informix provides several choices of transaction logging; you can create a database with transaction logging enabled or disabled. This is set at database level. Informix uses logical logs to record Data Manipulation Language (DML) entries (INSERT, UPDATE, DELETE operations) for logged databases, as well as DDL statements and checkpoint activity for all databases. If you do not enable database logging, Informix cannot fully recover the database(s) in the event of a failure, and you cannot use transactions.

When creating a database, you can specify a name for a dbspace storage space in which the database is created. If you do not do this, by default, the database is created in the rootdbs storage space.

When creating the database in Informix, make sure that you prepare the database with the appropriate code set to enable Informix to store all the character-based data in the expected internal ASCII representation. Use the `DB_LOCALE` and `SERVER_LOCALE` environment variables for the appropriate setting. Table 1 shows an example and the syntax of `CREATE DATABASE` in MySQL and Informix. You can set the `DB_LOCALE` environment variable to appropriate locale before creating a database in Informix.

Table 1. Comparison of CREATE DATABASE statement syntax that supports the U.S. English format in UNIX platforms

MySQL	Informix
	`export DB_LOCALE=en_us.8859-1`

`CREATE DATABASE example`
`DEFAULT CHARACTER SET utf8`
`DEFAULT COLLATE utf8_general_ci;`
	`CREATE DATABASE example IN dbspace1;`

For performance considerations in data migration, initially create the database

without logging in Informix. This setting can save time while loading data into the database. After the data is loaded, you can enable logging, as shown in Listing 4.

Listing 4. Enable database logging in an Informix database

```
ontape -s -L 0 -B example -F
```

Selecting a database

After the database is created, you must select it as a current database for all subsequent SQL statements. Table 2 shows the SQL statements for making a database current for MySQL and Informix.

Table 2. Comparison of SQL statements for selecting a database

MySQL	Informix
`USE example;`	`DATABASE example;`

Dropping a database

Dropping a database removes the database and all database objects associated to it. In MySQL, a drop database operation removes all of the database files (.BAK, .DAT, .HSH, .ISD, .ISM, .ISM, .MRG, .MYD, .MYI, .db, and .frm) from your file system. Similarly, Informix frees all storage spaces uses by the database. Table 3 shows SQL statements to drop a database for MySQL and Informix.

Table 3. The drop database statement

MySQL	Informix
`DROP DATABASE example;`	`DROP DATABASE example;`

Data type mapping

Before you start working with other database objects, let's compare the differences between MySQL and Informix data types. In general, all MySQL data types can be mapped to Informix data types with very little changes.

Every column in the database table has an associated data type, which determines the values that column can contain. Informix supports both built-in data types and user-defined data types (UDTs), whereas MySQL only supports built-in data types. For migration purposes, you only need to consider Informix built-in data types.

Table 4 illustrates how MySQL data types are mapped to Informix data types. It also shows the optional mapping for that particular data type.

Table 4. Data type mapping

MySQL	Informix
TINYINT	SMALLINT
SMALLINT	SMALLINT
MEDIUMINT	INTEGER
INT	INTEGER
BIGINT	INT8
REAL	DOUBLE PRECISION
DOUBLE	DOUBLE PRECISION
FLOAT	DOUBLE PRECISION
DECIMAL(p,s) Where: s > 0 && p >= s s > 0 && p < s s < 0	DECIMAL(min(p,32), min(s,32)) DECIMAL(min(p,32), min(s,32)) DECIMAL(min(p,32),0)
NUMERIC(p,s) Where: s > 0 && p >= s s > 0 && p < s s < 0	DECIMAL(min(p,32), min(s,32)) DECIMAL(min(p,32), min(s,32)) DECIMAL(min(p,32),0)
TINYINT UNSIGNED	SMALLINT
SMALLINT UNSIGNED	INTEGER *optional*: SMALLINT
BIGINT UNSIGNED	DECIMAL(20,0) *optional*: INT8
REAL UNSIGNED	DOUBLE PRECISION
DOUBLE UNSIGNED	DECIMAL(p,s) *optional*: DOUBLE PRECESION
FLOAT UNSIGNED	DOUBLE PRECISION
DECIMAL UNSIGNED	DECIMAL(p,s)
NUMERIC UNSIGNED	DECIMAL(p,s)
DATE	DATE
TIME	DATETIME HOUR TO FRACTION
TIMESTAMP	DATETIME YEAR TO FRACTION
DATETIME	DATETIME YEAR TO FRACTION *optional*: DATE
YEAR	CHAR(4)

VARCHAR(l)	VARCHAR(l) *optional*: LVARCHAR *optional*: CLOB
TINYBLOB	BYTE *optional*: BLOB
BLOB	BLOB *optional*: BYTE
MEDIUMBLOB	BYTE *optional*: BLOB
LONGBLOB	BYTE *optional*: BLOB
TINYTEXT	TEXT
TEXT	TEXT
MEDIUMTEXT	TEXT
LONGTEXT	TEXT

Creating a table

In this section, get a high-level overview of the difference in the `CREATE TABLE` syntax of MySQL and Informix. The `CREATE TABLE` statements are quite simple, with few exceptions. Table 5 shows the `CREATE TABLE` statement syntax for MySQL and Informix. Notice that no major change is required.

Table 5. Comparison of the CREATE TABLE statement syntax

MySQL	Informix

`CREATE TABLE mytable (`
`  wk_id INT(11) UNSIGNED NOT NULL`
 	
`  user_id INT(11) UNSIGNED DEFAULT NULL,`
`  cnt INT(10) UNSIGNED DEFAULT 100,`
`  cat_desc CHAR(12) NOT NULL,`
`  status VARCHAR(10) DEFAULT NULL,`
`  PRIMARY KEY (wk_id)`
`) TYPE=MyISAM;`
	`CREATE TABLE mytable (`
	`  wk_id SERIAL NOT NULL,`
	
	`  user_id INT,`
	`  cnt INT DEFAULT 100,`
	`  cat_desc CHAR(12) NOT NULL,`
	`  status VARCHAR(10) DAEFAULT NULL,`
	`  PRIMARY KEY (wk_id)`
	`);`

The following initial syntax changes are required:

- Remove the `NULL` clause for columns that allow the NULL value.
- Replace the `AUTO_INCREMENT` clause with the SERIAL datatype.
- Remove any storage element clauses (for example, `TYPE`, `ENGINE`).

MySQL and Informix both support partitioned tables. However, partitioned table functionality is not available in Informix Innovator-C Edition.

A memory table in MySQL is a hashed table that is always stored in memory and created by the `ENGINE=MEMORY` modifier with a `CREATE TABLE` statement. Memory tables are fast, but are not crash-safe. When MySQL crashes or during a scheduled reboot, the data in a memory table gets lost. MySQL defines a memory table during table creation.

With Informix, the concept of memory table is little different. Informix internally determines if a table or index needs to be in memory for better performance. Informix allows tables and indexes to remain in the memory buffer as long as possible. When a free memory buffer is requested by other database operations, tables in the memory buffer are moved to disk. Fortunately, you never loose data for crashes or reboots in Informix.

MySQL and Informix both support temporary tables. Like Informix, MySQL implements temporary tables as regular database tables, which are created with the `CREATE TEMPORARY TABLE` statement. These tables are client-specific and remain in existence only for the duration of a single client session. Temporary tables are automatically deleted when the client that created them closes its connection with the database server. Table 6 shows the `CREATE TEMPORATY TABLE` statement syntax for MySQL and Informix.

Table 6. Comparison of the CREATE TEMPORARY TABLE statement syntax

MySQL	Informix
`CREATE TEMPORARY TABLE temp_sales SELECT * FROM sales;`	`CREATE TEMP TABLE temp_sales (col1 INT, col2 CHAR(8), ...);` or `SELECT * FROM sales INTO temp_sales;`

Alter a table

`ALTER TABLE` is a statement that is used to modify one or more properties of a table. The syntax of the `ALTER TABLE` statement for MySQL and Informix is quite similar, although there are some differences. You need to modify the MySQL statements so that the statements can run successfully on Informix. Table 7 shows some `ALTER TABLE` statement syntax for MySQL and Informix.

Table 7. Comparison of the ALTER TABLE statement syntax

Action	Syntax for MySQL	Syntax for Informix

Add a new column	`ALTER ONLINE TABLE t1 ADD COLUMN c3 INT;`	`ALTER TABLE t1 ADD COLUMN c3 INT;`
Modify an existing column	`ALTER TABLE t1 MODIFY c2 VARCHAR(20);`	`ALTER TABLE t1 MODIFY (c2 VARCHAR(20));`
Drop a column	`ALTER TABLE t1 DROP COLUMN c2;`	`ALTER TABLE t1 DROP (c2);`
Alter the next serial value of a column to 1000	`ALTER TABLE t1 AUTO_INCREMENT = 1000`	`ALTER TABLE t1 MODIFY (c4 SERIAL(1000));`
Drop a constraints	`ALTER TABLE t1 DROP FOREIGN KEY fk_symbol;`	`ALTER TABLE t1 DROP CONSTRAINT fk_symbol;`

Creating constraints

You can have unique, not null, primary key, foreign key, and check constraints on a table. MySQL does not have check constraints. In the generated DDL script, depending on the constraint type, a constraint can be defined in the table at a column level or a table level, or it can be applied with an ALTER TABLE statement. Table 8 illustrates the syntax differences of some common constraint definitions.

Table 8. Comparison of constraints

Constraints	Syntax for MySQL	Syntax for Informix

Primary key constraints	`CREATE TABLE t1 ( c1 INT NOT NULL PRIMA c2 VARCHAR(30));`
	
	`OR`
	
	`ALTER TABLE t1 ADD PRIMARY`
		`CREATE TABLE t1 ( c1 INT NOT NULL PRIMARY KEY, c2 VARCHAR(30));`
		
		`OR`
		
		`ALTER TABLE t1 ADD CONSTRAINT PRIMARY KEY (c);`

Unique constraints	`CREATE TABLE t1 ( c1 INT NOT NULL, c2 VARCHAR(30), UNIQUE KEY (C1));`
	
	`OR`
	
	`ALTER TABLE t1 ADD UNIQUE`
		`CREATE TABLE t1 ( c1 INT NOT NULL UNIQUE, c2 VARCHAR(30));`
		
		`OR`
		
		`ALTER TABLE t1 ADD CONSTRAINT UNIQUE (a);`

Foreign key constraints	`CREATE TABLE t1 ( c1 INT NOT NULL PRIMA c2 VARCHAR(30));`	`CREATE TABLE t1 ( c1 INT NOT NULL PRIMARY KEY, c2 VARCHAR(30));`

```
CREATE TABLE t2 (
    c1 INT,
    FOREIGN KEY (c1)
    REFERENCES t1 (c1));

OR

ALTER TABLE t2 ADD  FOREIG
      REFERENCES t1 (c1) ;
```

```
CREATE TABLE t2 (
    c1 INT,
    FOREIGN KEY (c1)
    REFERENCES t1 (c1));

OR

ALTER TABLE t2 ADD  FOREIGN KEY (c1)
      REFERENCES t1 (c1) ;
```

Creating an index

The basic concept of an index is the same on MySQL and Informix. In addition to MySQL's index types and attributes, Informix provides additional index types and attributes. Informix supports different index types for example, B-Tree indexes, R-Tree indexes, functional indexes, and indexes that Informix DataBlade modules provide for user-defined data such as spatial or time series data.

Informix also provides a large variety of index attributes, such as DISTINCT, UNIQUE, CLUSTER, FILLFACTOR, ONLINE, and more. The general `CREATE INDEX` statement is the same on MySQL and Informix. A couple of exceptions are:

- The HASH index type for MySQL MEMORY and NDB storage engines needs to be changed to a normal index under Informix.
- You cannot add an index using an `ALTER TABLE` statement under Informix.

Table 9 shows some `CREATE INDEX` statement syntax for MySQL and Informix.

Table 9. Comparison of the CREATE INDEX statement syntax

MySQL	Informix

`CREATE INDEX idx1 ON tab1(id);`
`OR`
`CREATE INDEX idx1 ON tab1(id) USING BTRE`
`OR`
`ALTER TABLE tab1 ADD INDEX idx1 (c);`
	`CREATE INDEX idx1 ON tab1(id);`
	`OR`
	`CREATE INDEX idx1 ON tab1(id) USING BTREE;`

Creating a view

A view is a virtual table defined by a SELECT statement. There are not many differences in CREATE VIEW syntax between MySQL and Informix. Table 10 shows the common differences.

Table 10. Comparison of the CREATE VIEW statement syntax

MySQL	Informix
`CREATE VIEW user1.view_name AS SELECT * FROM tab1 AS s;`	`CREATE VIEW user1.view_name AS SELECT * FROM tab1;`

There is no difference in retrieving data from a view. A view can be treated as a table, and you can use all of the SELECT statements that you can use against a table.

Dropping a view

You don't need to consider any changes in the DROP VIEW statement, as MySQL and Informix use similar syntax. Table 11 shows some DROP VIEW statement syntax for MySQL and Informix.

Table 11. Comparison of the DROP VIEW statement syntax

MySQL	Informix
`DROP VIEW user1.view_name;`	`CREATE VIEW user1.view_name;`

Creating, altering, or dropping an event

MySQL events are tasks that run according to a schedule. When you create an event, you are creating a named database object containing one or more SQL statements to be executed at one or more regular intervals, beginning and ending at a specific date and time. In Informix, you need to use a Scheduler task to perform a specific event/action at specific times.

Creating or dropping a procedure or function

In Informix, although you can use a CREATE PROCEDURE statement to write and register a procedure routine that returns one or more values, it is recommended that you use the CREATE FUNCTION statement instead. You must also use the CREATE FUNCTION statement to register an external function. Use the CREATE PROCEDURE statement to write and register a procedure or to register an external procedure. Table 12 shows some simple CREATE and DROP FUNCTION statement syntax for MySQL and Informix.

Table 12. Comparison of the CREATE/DROP FUNCTION statement syntax

Action	Syntax for MySQL	Syntax for Informix

Create a Function	`CREATE FUNCTION delete_ord RE`
	`BEGIN;`
	`DECLARE item_cnt INT;`
	`SELECT count(*) INTO it`
	`WHERE order_num = o_`
	`DELETE FROM orders`
	`WHERE order_num = o_`
	`RETURN o_num, item_cnt;`
	`END;`
		`CREATE FUNCTION delete_order(o_num INT)`
		`RETURNING INT, I`
		`DEFINE item_cnt INT;`
		`SELECT count(*) INTO item_cnt FROM it`
		`WHERE order_num = o_num;`
		`DELETE FROM orders`
		`WHERE order_num = o_num;`
		`RETURN o_num, item_cnt;`
		`END FUNCTION;`

Drop a Function	`DROP FUNCTION delete_order;`	`DROP FUNCTION delete_order(o_num INT);`

In Informix, if the function name is not unique within the database, you must specify enough parameter (argument) information to disambiguate the function name.

Creating a trigger

A trigger is a database object that, unless disabled, automatically executes a specified set of SQL statements, called the trigger action, when a specified trigger event occurs. Syntactically there are some differences between the MySQL and Informix `CREATE TRIGGER` statements. Because of the complexity in the `CREATE TRIGGER` syntax, you must consider the differences on a case-by-case basis.

The trigger event that initiates the trigger action can be an `INSERT` statement, a `DELETE` statement, an `UPDATE` statement, or (for triggers on Informix tables only) a `SELECT` statement. The `MERGE` statement can also be the triggering event for an `UPDATE`, `DELETE`, or `INSERT` trigger.

In Informix, there are three trigger action clauses: `BEFORE`, `AFTER` and `FOR EACH ROW`.

- The `BEFORE` actions are executed once for each triggering event, before the database server performs the triggering DML operation.
- The `AFTER` actions are also executed once for each triggering DML event, after the operation on the table is complete, in the context of the triggering statement.
- The `FOR EACH ROW` actions are executed for each row that is inserted, updated, deleted, or selected in the DML operation, after the DML operation is executed on each row, but before the database server writes the values into the log and into the table.

Table 13 provides a simple example of CREATE TRIGGER syntax. This example inserts a row into backup_table1 for every row that is inserted into table1. The values that are inserted into col1 of backup_table1 are an exact copy of the value for table1.

Table 13. Comparison of the CREATE TRIGGER statement syntax

MySQL	Informix

CREATE TABLE *table1* (col1 INT);
CREATE TABLE *backup_table1* (col1 INT);
CREATE TRIGGER *before_trig*
BEFORE INSERT ON *table1*
FOR EACH ROW
BEGIN
(
INSERT INTO *backup_table1*
SET col1 = NEW.col1
)
END;
	CREATE TABLE *table1* (col1 INT);
	CREATE TABLE *backup_table1* (col1 INT);
	CREATE TRIGGER *before_trig*
	INSERT ON *table1*
	REFERENCING NEW AS new
	FOR EACH ROW
	(
	INSERT INTO *backup_table1* (col1)
	VALUES (new.col1)
	);

Granting and revoking privileges

The GRANT statement provides access privileges to users. In Informix, use the GRANT statement to grant privileges on a database, table, view, or procedure, or to grant a role to a user or another role. Similarly, use the REVOKE statement to revoke privileges on a database or database object, or to revoke a role from a user or from another role. Table 14 provides a summary of all Informix privileges available for a particular database object. If not all, most of these privileges are supported by MySQL.

Table 14. Database objects and their privileges

SQL object	Privileges
Database	Connect, Resource, DBA
Table	Select, Update, Insert, Delete, Index, Alter, References
Database	Connect, Resource, DBA
Column	Select, Update, References
View	Select, Insert, Delete, Update
Sequence	Select, Alter
UDT	Usage, Under
Routine	Execute
Language	Usage

In MySQL, there is no concept of role. In Informix, a role is a classification of access privileges that the DBA assigns. For example, 'development' could be a role that involves specific access privileges. After a role is created with the `CREATE ROLE` statement, the DBA can use the `GRANT` statement to assign access privileges to the role and to assign the role to individual users (or to other roles). Thus users with similar work tasks can hold the same set of access privileges that their work tasks require.

Unlike MySQL, Informix supports all standard SQL privileges. For example, MySQL does not support the `UNDER` privilege. The major difference is MySQL associates privileges with the combination of a host name and user name, while Informix does the same with a user name only. In Informix, there is no concept of create user at the database level. The user gets created at the operating system level. Before a user can access any Informix database object(s), the user must have proper permission (CONNECT/RESOURCE/DBA) to accesses the database (database-level privileges). As a database administrator or a particular database object owner, you can provide appropriate permission to an individual user, a group, a role, or everyone (PUBLIC).

The concept of `GRANT` and `REVOKE` privileges is the same on MySQL and Informix. To use `GRANT`, you must have the `GRANT OPTION` privilege, and you must have the privileges that you are granting. Table 15 shows some common `GRANT` and `REVOKE` statement syntax for MySQL and Informix.

Table 15. Comparison of the GRANT and REVOKE statement syntax

Privilege	Syntax for MySQL	Syntax for Informix
Grant	`CREATE USER 'tom'@'localho` `  IDENTIFIED BY 'mypass';`	`DATABASE db1;` `GRANT RESOURCE TO tom;`
	`GRANT ALL ON db1.tab1` `  TO 'tom'@'localhost';`	`GRANT ALL ON 'db1'.tab1 TO 'tom';`
	`GRANT SELECT ON db1.tab2` `  TO 'tom'@'localhost';`	`GRANT SELECT ON 'db1'.tab2 TO 'tom';`
	`GRANT UPDATE(col1,col2) ON` `  TO 'tom'@'localhost';`	`GRANT UPDATE(col1,col2)` `  ON 'db1'.tab3 TO 'tom';`
	`GRANT EXECUTE ON db1.proc1` `  TO 'tom'@'localhost';`	`GRANT EXECUTE ON 'db1'.proc1 TO 'tom';`

Revoke	`REVOKE ALL PRIVILEGES ON`
	`db1.tab1`
	`FROM 'tom'@'localhost';`
		`REVOKE ALL ON 'db1'.tab1`
		`FROM 'tom';`

Section 6. Table data movement

Once the database schema is extracted from MySQL, the next thing you need to consider is extracting data from each individual table in MySQL. You can use `mysqldump` or the `OUTFILE` option with `SELECT` statement. For simplicity, we used the `OUTFILE` option with `SELECT` statement for this tutorial. Whatever method you choose for data extraction, make sure you consider the following items for smooth data movement between MySQL and Informix:

- Unload data in MySQL is in text format.
- Create an individual unload (outfile) file for each table in MySQL.
- Each line in an unload file represents one single row in My SQL.
- Each column/field is separated by a pipe (|) character in MySQL.
- Each Line is terminated by '/n' or valid line terminator character in MySQL.

You can extract data from a MySQL table using SQL commands similar to those shown in Listing 5.

Listing 5. Unload data from a MySQL table in text format

```
SELECT * INTO OUTFILE '/tmp/table1.txt'
  FIELDS TERMINATED BY '|'
  LINES TERMINATED BY '\n'
  FROM table1;
```

Section 7. Migrate to Informix Innovator-C Edition

You already transformed DDL statements from MySQL to Informix syntax (in the "Database schema movement" section). In this section, see how you can execute the schema file with DDL statements on Informix Innovator-C Edition to create necessary database objects. Use the command in Listing 6 to create database objects from the schema file (schema.sql):

Listing 6. Create database objects from a schema file

```
dbschema - schema.sql
```

After all database objects are created on Informix, the next process is to load data previously extracted from MySQL into Informix tables. You must load data to one Informix table at a time, using SQL commands similar to those shown in Listing 7.

Listing 7. Load data from a text file to an Informix table

```
LOAD FROM '/tmp/table1.txt'
  INSERT INTO table1;
```

Section 8. Conclusion

This tutorial has focused on a step-by-step migration approach to speed up the migration process from MySQL to Informix Innovator-C Edition. It is our goal that you start considering the use of this powerful database, which, until recently, was largely found within the domain of large corporations. Start using Informix Innovator-C Edition, and leverage the advantages of a world-class database to meet your own database needs.

Resources

Learn

- "Migrate a database from MySQL to IBM Informix Innovator-C Edition, Part 1: Comparing MySQL to IBM Informix Innovator-C Edition" (developerWorks, February 2011): Gain insight into these products' respective advantages and disadvantages for your business.
- Informix Innovator-C Edition: Learn more about Informix Innovator-C Edition
- IBM Informix Dynamic Server v11.70 Information Center: Learn more about Informix. Find the information that you need to use Informix products and features.
- "MySQL Restrictions and Limitations" (Oracle, February 2011): Understand the restrictions that apply to the use of MySQL features, such as subqueries or views.
- "IBM Migration Toolkit support for migrating data from MySQL to DB2 and Informix" (developerWorks, July 2008): Learn how this toolkit can help you migrate DDL and DML statements, and see how to map data types.
- developerWorks on Twitter: Follow us.
- developerWorks on-demand demos: Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- developerWorks Information Management zone: Learn more about Information Management. Find technical documentation, how-to articles, education, downloads, product information, and more.
- Stay current with developerWorks technical events and webcasts.

Get products and technologies

- Informix Innovator-C Edition: Download Informix Innovator-C Edition.
- IBM Migration Toolkit: Download the IBM Migration Toolkit—an easy-to-use tool that allows you to migrate your data from a wide variety of source databases to either DB2 or Informix, regardless of platform
- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.

Discuss

- Participate in the discussion forum for this content.

- Participate in developerWorks blogs and get involved in the developerWorks community; with your personal profile and custom home page, you can tailor developerWorks to your interests and interact with other developerWorks users.

About the author

Sanjit Chakraborty

Sanjit Chakraborty is a member of the Down System and Diagnostics Team for IBM Informix Technical Support, which is responsible for handling critical customer situations and developing support tools for use by the Technical Support Organization. Sanjit has worked more than 15 years in the information technology industry in various roles. He is an IBM Certified System Administrator for Informix and DB2, and a designated archiving subject matter expert. Sanjit developed several Informix features and Down System Support tools. He is also an author and technical reviewer of many technical articles, tutorials, and training course materials on various Informix topics.