InfiniDB® Multiple UM Configuration Guide

Release: 4.6
Document Version: 4.6-1

Contents

1 Introduction

This guide describes scale-out configuration for the InfiniDB User Module (UM) component. This guide is applicable when scaling front-end performance and/or configuring the front-end for High Availability (HA).

Please see the InfiniDB Installation Guide for installation of a multiple User Module configuration.

1.1 Audience

This guide is written for IT administrators who are responsible for administering a multiple User Module configuration of the InfiniDB System.

1.2 List of Documentation

The InfiniDB Database Platform documentation consists of several guides intended for different audiences. The documentation is described in the following table:

Document	Description
InfiniDB Administrator's Guide	Provides detailed steps for maintaining InfiniDB.
InfiniDB Apache Hadoop^{TM} Configuration Guide	Installation and Administration of an InfiniDB for Apache Hadoop system.
InfiniDB Concepts Guide	Introduction to the InfiniDB analytic database.
InfiniDB Minimum Recommended Technical Specifications	Lists the minimum recommended hardware and software specifications for implementing InfiniDB.
InfiniDB Installation Guide	Contains a summary of steps needed to perform an install of InfiniDB.
InfiniDB SQL Syntax Guide	Provides syntax native to InfiniDB.
Performance Tuning for the InfiniDB Analytics Database	Provides help for tuning the InfiniDB analytic database for parallelization and scalability.
InfiniDB Windows Installation and Administrator's Guide	Provides information for installing and maintaining InfiniDB for Windows.

1.3 Obtaining documentation

These guides reside on our http://www.infinidb.co website. Contact support@infinidb.co for any additional assistance.

1.4 Documentation feedback

We encourage feedback, comments, and suggestions so that we can improve our documentation. Send comments to support@infinidb.co along with the document name, version, comments, and page numbers.

1.5 Additional resources

If you need help installing, tuning, or querying your data with InfiniDB, you can contact *support@infinidb.co*.

2 Understanding Behavior of Multiple User Modules

InfiniDB allows configurations for multiple User Modules (UM). Behavior with a multiple UM system should be understood before using this type of configuration.

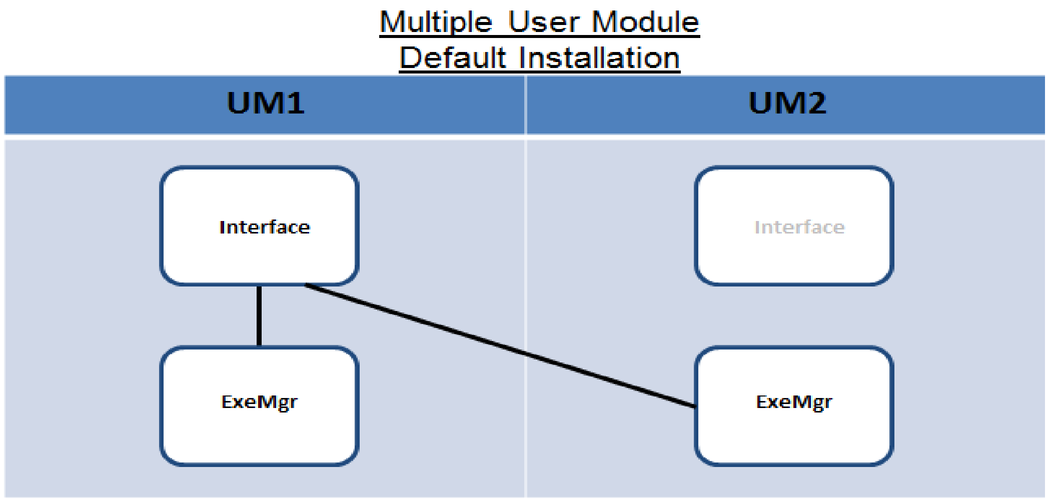
2.1 *Default Multi-UM Installation*

When a multi-UM system is first installed (either by initial installation or subsequently adding a UM), certain behavior exists for query execution, cross-engine joins and query statistics collection as follows:

2.1.1 Query Execution

Benefits for using the default installation setup for multiple UMs:

- Users/Tools still only need to use InfiniDB from UM1
- Automatic round-robin distribution/scale-out of queries (based on connection id) across all UMs as illustrated in the following graphic:



Connection Id based round-robin

NOTE: In this default setup, there will be a mysql instance running on UM 2..N. These mysql instances must NOT be used for connection to InfiniDB as there is no synchronization of schema between them. If you need to use more than one mysql instance please see the Multiple Front-End Configuration section.

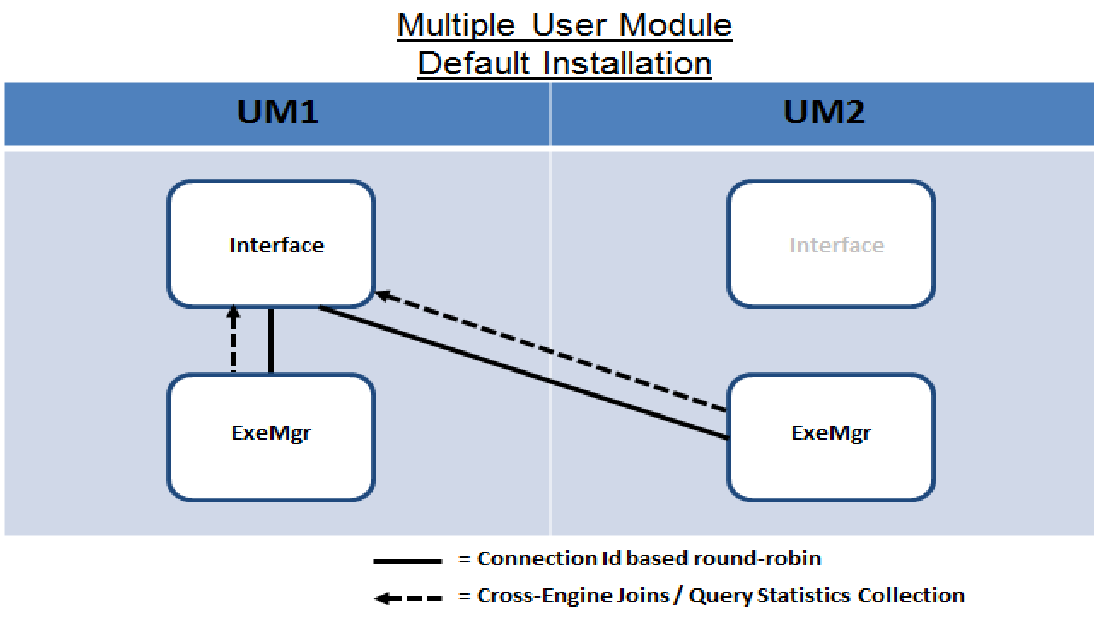
2.1.2 Cross-Engine Joins

Cross-Engine joins allow InfiniDB tables to be joined with non-InfiniDB tables (i.e., MyIsam tables) within a query. The connection information, as defined further in the InfniDB Administrator’s Guide, controls which UM the ExeMgr will be used to access the non-InfiniDB tables. By default, the cross-engine entry in the Calpont.XML file will have no connection information defined:

```
<CrossEngineSupport>
      <Host>unassigned</Host>
      <Port>3306</Port>
      <User>unassigned</User>
      <Password></Password>
</CrossEngineSupport>
```

If using this capability, you will need to populate this information by defining the connection information and restarting the system to distribute the Calpont.XML across all UMs.

NOTE: Defining this connection information to point to UM1 will allow all users access to the non-InfiniDB tables regardless of which UM is processing the query:


Multiple User Module
Default Installation
UM1
UM2
Interface
Interface
ExeMgr
ExeMgr
= Connection Id based round-robin
= Cross-Engine Joins / Query Statistics Collection

Example Calpont.XML entry to point to UM1 (with ip address of 10.100.135.85):

```
<CrossEngineSupport>
      <Host>10.100.135.85</Host>
      <Port>3306</Port>
      <User>root</User>
      <Password>um_access</Password>
</CrossEngineSupport>
```

Additional Notes:

- Firewall access for the port must be resolved between the UMs. Contact your System Administrator for information on this.
- Grant access must be resolved as well. Please see the InfiniDB Administrator's Guide for more information.

2.1.3 Query Statistics Collection

Once enabled, query statistics are gathered and populated into the `querystats` table in the reserved `infinidb_querystats` database. Please see the Query Performance chapter in the InfiniDB Administrator's Guide for more information.

The CrossEngineSupport entry in Calpont.XML is used to determine where the `querystats` table is populated (see Cross-engine Join illustration above). As with query and cross-engine joins, the same benefits apply in that only one UM1 would need to be used to analyze statistics across the whole InfiniDB system.

2.2 Multiple Front-End Configuration

InfiniDB also has the capability to provide a concurrency scale out configuration. Please see The InfiniDB Architecture chapter in the InfiniDB Concepts Guide for more information.

Benefits with this configuration:

- Higher concurrency capability with multiple front-ends.
- Users/Tools may connect to InfiniDB on any UM.
- High Availability capable in case of the failure of one of the UMs.

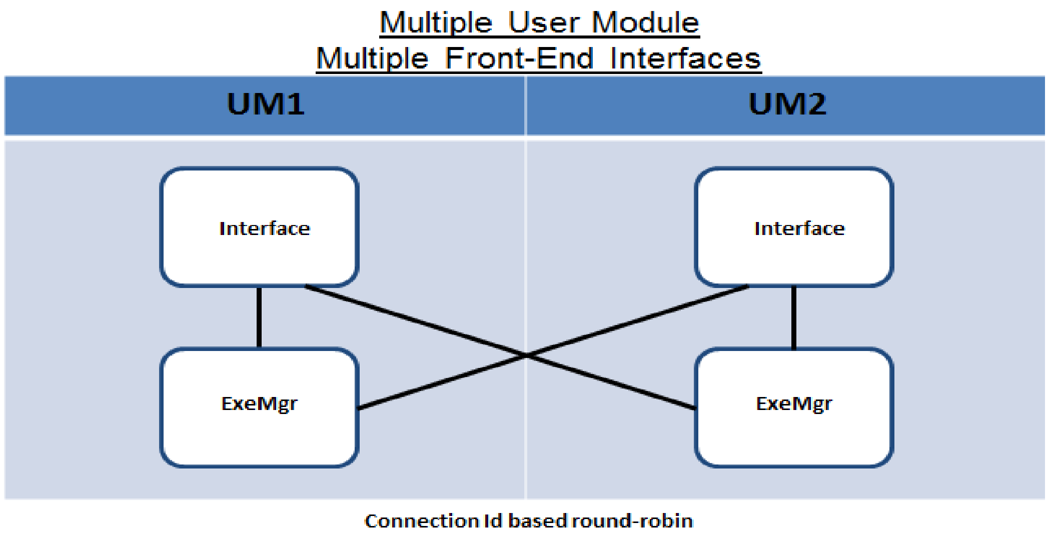
Disadvantage with this configuration:

- The front-end definitions for these User Modules must be kept in synch. Therefore, any DDL that has been executed on a UM must be replicated to the other UMs.
- Any changes to the MySQL configuration file (my.cnf) must manually be kept in sync across all UMs.

NOTE: The only difference between this configuration and the intial default configuration is the use of the front-end interface on all UMs instead of just one UM.

2.2.1 Query Execution

When a multi-UM system is used with multiple-front ends, the same query behavior exists for query execution and the same automatic round-robin distribution exists across all UMs as illustrated in the following graphic:


Multiple User Module
Multiple Front-End Interfaces
UM1
UM2
Interface
Interface
ExeMgr
ExeMgr
Connection Id based round-robin

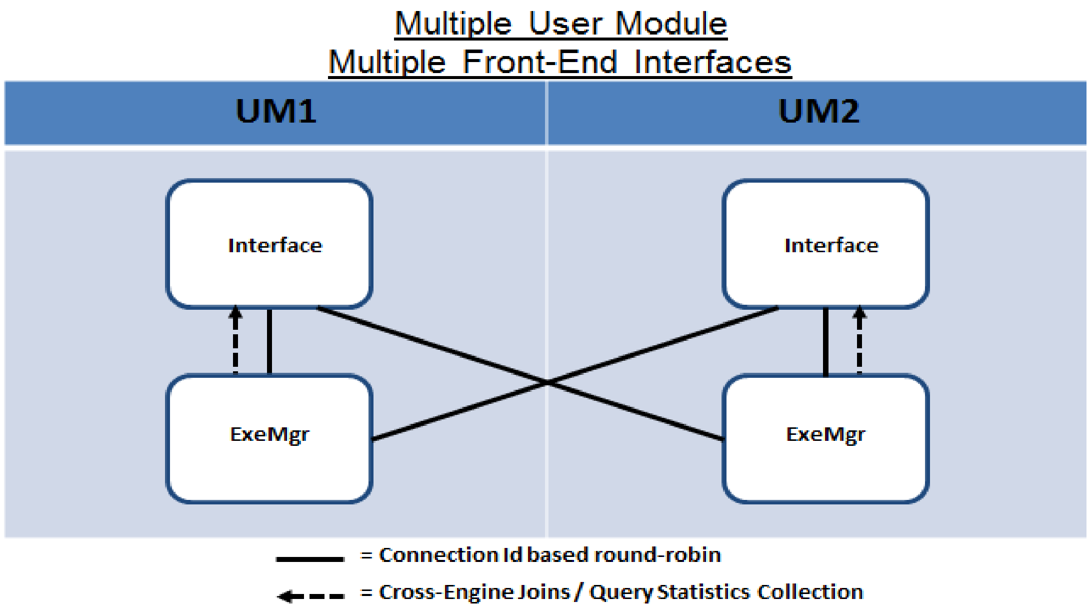
Again, there is nothing unique to using this configuration other than the high importance of keeping the front-ends for all UMs in synch. Please see “Keeping Multiple Front-End Interfaces in Synch” chapter later in this Guide.

2.2.2 Cross-Engine Joins

Cross-Engine Joins must still be set up regardless of using a single front-end or using multiple front-ends. If using this capability, you will need to populate this information by defining the connection information to point to the localhost and restarting the system to distribute the Calpont.XML across all UMs.

NOTES:

- Defining this connection information to point to localhost will allow all users access to the non-InfiniDB tables regardless of which UM is processing the cross-engine query and allow for HA capability in case of a failure of a UM.
- Any non-InfiniDB tables being used for cross-engine joins must be kept in synch for both definition and data. If following the procedure for keeping the multiple front-ends in synch, the non-InfiniDB tables will automatically synch as they reside in the `mysql/db` directory under the InfiniDB structure.
- The potential is greater for indeterminate results due to the front-ends not in synch.


Multiple User Module
Multiple Front-End Interfaces
UM1
UM2
Interface
Interface
ExeMgr
ExeMgr
= Connection Id based round-robin
= Cross-Engine Joins / Query Statistics Collection

Example Calpont.XML entry to point to localhost:

```
<CrossEngineSupport>
      <Host>127.0.0.1</Host>
      <Port>3306</Port>
      <User>root</User>
      <Password>um_access</Password>
</CrossEngineSupport>
```

2.2.3 Query Statistics Collection

As with the default installation, the CrossEngineSupport entry in Calpont.XML is used to determine where the `querystats` table is populated (see Cross-engine Join illustration above).

This leads to a disadvantage of using multiple front-end interfaces in that the statistics will be populated on each UM that the query originated from and the administrator would have to analyze statistics on each UM.

If this disadvantage is great enough, then it is recommended to have both cross-engine and query statistics collection point to a single UM (i.e.UM1). If a failure occurs, then manual modification of the Calpont.XML entry would be need to point to another UM (i.e.UM2).

3 Keeping Multiple Front-End Interfaces in Synch

As referenced earlier, if using multiple front-end interfaces, the front-end definitions for these User Modules must be kept in synch. Therefore, any DDL and non-InfiniDB table creation/population that have been executed on a UM must be replicated to the other UMs. The information below describes options to effectively perform this replication.

When in a configuration with multiple front-end interfaces, it is InfiniDB's recommendation that all DDL and non-InfiniDB table creation/population be executed on only one UM, preferably UM1. By doing this, it is easier to keep all UMs in synch by using one of the two options below.

Note: In addition to these options below, any changes made to the MySQL Configuration file (my.cnf) must be made across all UMs.

3.1 Option 1: Front-End Directory Copy

Note: This is the preferred option to use.

All front-end schema information resides in the following directory under the InfiniDB system:

```
mysql/db
```

This schema information must be kept in synch across all online UMs. In order to keep them in synch, the ***rsync*** command may be used to synch up the source UM to all other target UMs.

Here is an example rsync command used to sync up 2 UM servers with a root installation (excluding any MySQL .err and .pid files). This would be executed on the source UM where DDL statements have been executed:

```
rsync -vuopg -e ssh --delete --exclude=*err --exclude=*pid -r /
usr/local/Calpont/mysql/db root@infiniDB_UM2:/usr/local/Calpont/ mysql/
```

If a more automated mechanism is desired, this command can be placed within a script. Please see "Appendix A - Script Example for rsync" or an example of this command within a script.

3.2 Option 2: Manual Execution of Commands

The following areas of syntax must be kept in synch on all UMs:

- DDL
- All Non-InfiniDB table DDL, DML and import

3.2.1 DDL

The CREATE and DROP commands can be used on all UMs needing to be synched up.

- For any CREATE TABLE executed on a UM, the other UMs would have to be synched up using the 'schema sync only' comment. The same CREATE TABLE command would need to be executed on all other UMs with the comment option.

```
CREATE newtable (column_info) ENGINE=INFINIDB COMMENT='schema sync only';
```

- For any DROP TABLE executed on a UM, the other UMs would have to be synched up using the RESTRICT option. This DROP TABLE command would need to be executed on all other UMs.

```
DROP TABLE droppedtable RESTRICT;
```

- For any ALTER TABLE or RENAME TABLE command, the combination of the above two statements of DROP and CREATE would need to be executed on all other UMs.

```
DROP TABLE alteredtable RESTRICT;
CREATE alteredtable (column_info) ENGINE=INFINIDB COMMENT='schema sync
only';
```

- For any CREATE PROCEDURE or DROP PROCEDURE command, the same command would need to be executed on all other UMs.

- For any CREATE DATABASE or DROP DATABASE command, the same command would need to be executed on all other UMs.

3.2.2 Non-InfiniDB tables

All actions against non-InfiniDB tables on one UM must be kept in synch on all UMs if using this manual method. This included the following actions:

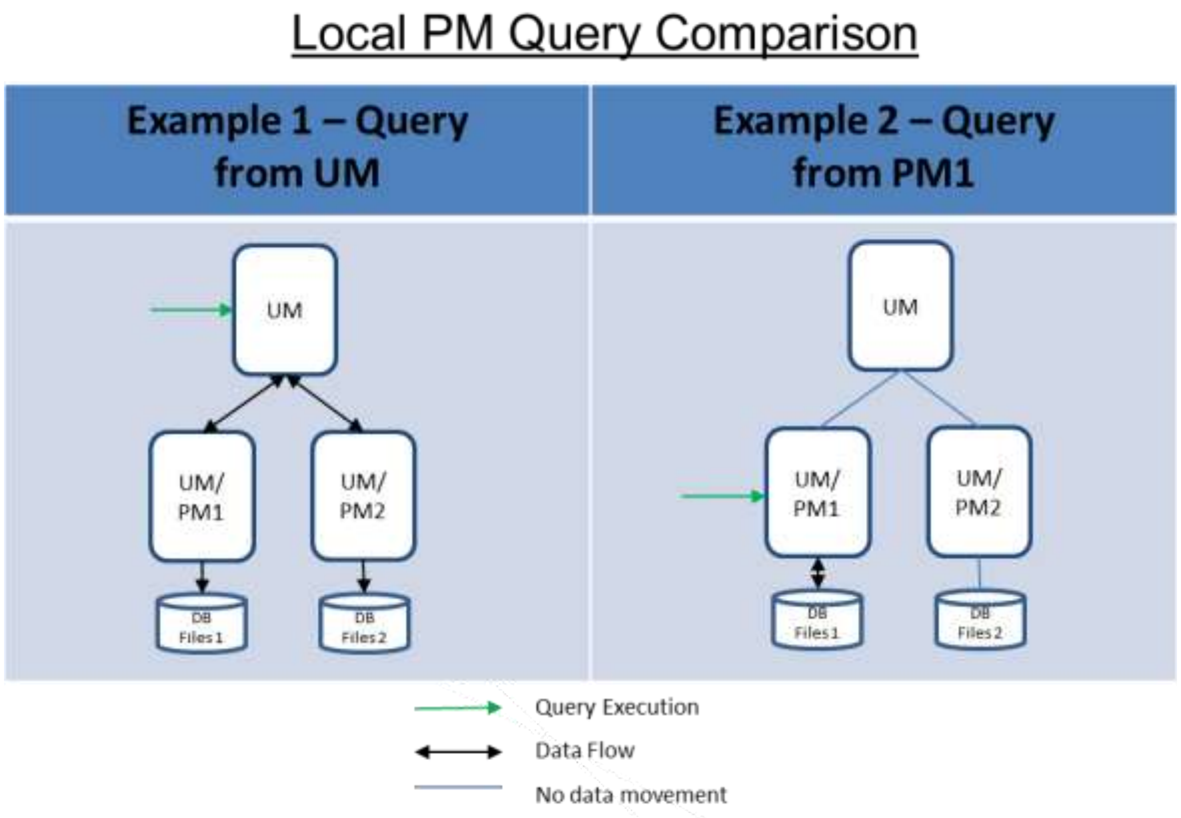
- DDL -Table creation/modification
- DML - All INSERTS, UPDATES, DELETE, LOAD DATA INFILE commands
- Import - Any imports into non-InfiniDB tables must be replicated across all UMs.

3.3 Option 3: Replication across User Modules

Replication across User Modules may be set up where there is one replication master and one or more replication slaves. Please see the "Replication across UMs" section below for more information.

4 Local PM Query

InfiniDB provides the ability to query data from a single PM and for the query results to remain on the PM without having to use the network bandwidth between UM and PM. This allows local PM query and the results to be used as input for another query or cpimport. The following diagram illustrates the comparison between a standard query from the UM server and a query that originates from a PM server:


Local PM Query Comparison
Example 1 – Query from UM
Example 2 – Query from PM1
UM
UM/ PM1
UM/ PM2
DB Files 1
DB Files 2
Query Execution
Data Flow
No data movement

In Example 1, a query that originates from the UM follows the standard path of obtaining data from all PMs to satisfy the query. For Example 2, a query that originates from a PM only obtains the data specific to that PM.

In order to utilize this type of processing, the following key areas of InfiniDB must be set up:

- Installation
- Replication across UMs (both standalone UMs and UMs as part of the local PM) to ensure synched front-end data
- Configuration of local PM settings

4.1 Installation of Local PM Query

Local PM query can be set up on InfiniDB PMs at installation/upgrade time in the `postConfigure` utility with the `–lq` option. When this option is used, the PM is changed to a UM/PM combination server. A user would then log into a specific UM/PM combination server as needed.

```
postConfigure -lq
```

Note: It is important to note that with this option, replication on all UMs (both standalone UMs and UMs as part of the PM) will be automatically enabled. See" Replication across UMs" section below.

4.2 Replication across UMs

As referenced earlier, if using multiple front-end interfaces, the front-end definitions for these User Modules must be kept in synch. Therefore, any DDL and non-InfiniDB table creation/population that have been executed on a UM must be replicated to the other UMs.

Automated replication can be set up on InfiniDB UMs where one UM is the replication master (typically UM1) and all other UMs (standalone UMs or UM/PM combination servers) are replication slaves. This is performed at installation/upgrade time in the postConfigure utility with the `–rep` option.

```
postConfigure -rep
```

Note: It is important to note that if local PM Query is desired along with replication, then the `postConfigure –lq` option should be used. See "Installation of Local PM Query" section below.

4.3 Configuration of Local PM Settings

InfiniDB has the ability to query data from just a single PM instead of the whole database thru the UM. In order to accomplish this, the `infinidb_local_query` variable in the my.cnf configuration file is used and maybe set as a default for the instance or set at the session level. If the `postConfigure` utility has been executed with the `–lq` option (Local PM Query), this variable has automatically be set on for each PM. You may choose to modify this variable either at the instance level or session level. Please see the InfiniDB Administrator's Guide for modification at the instance level or the InfiniDB SQL Syntax Guide for setting at the session level.

As a reminder, with this variable set to on, only data from the local PM will be returned.

4.4 *Local PM Query Examples*

Example #1 - SELECT from a single table on local PM to import back on local PM:

With the infinidb_local_query variable set to 1 (default with local PM Query):

```
idbmysql -e 'select * from source_schema.source_table;' -N |
/usr/local/Calpont/bin/cpimport target_schema target_table -s
'\t' -n1
```

Example #2 - SELECT involving a join between a fact table on the PM node and dimension table across all the nodes to import back on local PM:

With the infinidb_local_query variable set to 0 (default with local PM Query):

Create a script (i.e., extract_query_script.sql in our example) similar to the following:

```
set infinidb_local_query=0;
select fact.column1, dim.column2 from fact join dim using (key)
where idbPm(fact.key) = idbLocalPm();
```

The `infinidb_local_query` is set to 0 to allow query across all PMs.

The query is structured so that the UM process on the PM node gets the fact table data locally from the PM node (as indicated by the use of the `idbLocalPm()` function), while the dimension table data is extracted from all the PM nodes.

Then you can execute the script to pipe it directly into cpimport:

```
idbmysql source_schema -N < extract_query_script.sql |
/usr/local/Calpont/bin/cpimport target_schema target_table -s
'\t' -n1
```

5 Appendix A - Script Example for rsync

The following is an example of placing the rsync command within a script for more automated means of executing the rsync command. The rsync command would be replicated for each UM needing to be kept in synch.

```
#!/usr/bin/expect
#
set timeout 600
set PASSWORD serverpwd
set COMMAND1 "rsync -vuopg -e ssh --delete --exclude=*err --exclude=*pid -r
/usr/ local/Calpont/mysql/db root@infinidb_UM2:/usr/local/Calpont/mysql/"
set DEBUG 0
log_user $DEBUG
spawn -noecho /bin/bash
expect -re "# "
#
# send command
#
send "$COMMAND1\n"
expect {
   -re "Host key verification failed" { send_user "FAILED: Host key verification
         failed\n" ; exit -1}
   -re "service not known" { send_user " FAILED: Invalid Host\n" ; exit -1}
   -re "ssh: connect to host" { send_user " FAILED: Invalid Host\n" ; exit -1 }
   -re "authenticity" { send "yes\n"
      expect {
      -re "word: " { send "$PASSWORD\n" } abort
      }
   }
-re "word: " { send "$PASSWORD\n" } abort
}
expect {
   -re "# " {} abort
   -re "Permission denied" { send_user " FAILED: Invalid password\n" ; exit -1 }
   -re "(y or n)" { send "y\n"
      expect -re "# " { exit }
      }
   }
exit
```