TS DataServer™ for MySQL
Reference Manual

Version 8.7.1

285 Davidson Ave. • Ste. 302 • Somerset, NJ 08873-4153
Telephone: 732-560-1377 • Outside NJ 800-524-0430
Fax: 732-560-1594

Internet address: **http://www.tbred.com**

Published by:
Thoroughbred Software International, Inc.
285 Davidson Ave., Suite 302
Somerset, New Jersey 08873-4153

Document Number: TQ8.7.1M102

INTRODUCTION

TS DataServer for MySQL is an integral part of the Thoroughbred Software network solution. With TS Environment 8.7.1+, data from MySQL tables can be made available across platforms for use in Dictionary-IV, Report-IV, Query-IV, Script-IV, and Thoroughbred Basic applications.

Operating System Support: Linux
For specific information, please contact your Thoroughbred Sales Representative.

This guide first describes the Client side of TS DataServer for MySQL, which is provided with every TS Environment installation. It then provides specific information for accessing MySQL.

Your applications may require special privileges to access the database. You should consider the following issues:

- Setting privileges to access the database.
- Designing or modifying site procedures.
- Setting procedures for server transaction processing.

Requirements

To run TS DataServer for MySQL your system must meet the following requirements:

- Network communications implemented with TCP/IP protocol
- TS Environment 8.7.1 or above
- MySQL version 5.1.49 or above

Server access is enabled by the addition of a number of configuration items. For more information see SERVER.MAP and IPLINPUT.

MySQL Configuration

To run TS DataServer for MySQL the following changes must be made to the default MySQL configuration:

- Use the InnoDB plugin:
  - Ignore the internal InnoDB storage engine
  - Set the location of the MySQL plugins directory
  - Load the InnoDB plugin
- Set the default storage engine to InnoDB
- Set the transaction isolation level
- Set the InnoDB transaction timeout value to 15 seconds.

These changes can be made by adding the following to the MySQL configuration file my.cnf:

```
[mysqld]
ignore_builtin_innodb
plugin_dir = /usr/lib/mysql/plugin
plugin_load = innodb=ha_innodb_plugin.so
        ; innodb_trx=ha_innodb_plugin.so
        ; innodb_locks=ha_innodb_plugin.so
        ; innodb_lock_waits=ha_innodb_plugin.so
        ; innodb_cmp=ha_innodb_plugin.so
        ; innodb_cmp_reset=ha_innodb_plugin.so
        ; innodb_cmpmem=ha_innodb_plugin.so
        ; innodb_cmpmem_reset=ha_innodb_plugin.so
default-storage-engine = InnoDB
transaction-isolation = READ-COMMITTED
innodb_lock_wait_timeout = 15
```

Note: The plugin_load option value must be written as a single line without spaces.

The MySQL service must be restarted in order for the above changes to take effect.

PRODUCT DESCRIPTION

TS DataServer for MySQL Components

The TS DataServer for MySQL consists of three components:

- A Client enabled TS Environment,
- A Startup Program, and
- A Server Process (tsmysql) for each connection.

Client Enabled TS Environment

The client side is enabled in the standard TS Environment. When configured for data server access, it detects requests for data that reside on the server. When this event occurs, it sends the I/O request to the server for processing. The server responds with the result set and processing continues.

Startup Program

The Startup Program (tsmysql) creates Server Processes for clients as needed. When the Startup Program detects a client attempting to connect, it tries to create a Server Process for the client. If a Server Process is created successfully, the client and server can perform client/server communications.

NOTE: A server system must have the Startup Program running before clients attempt to connect to it.

Server Process

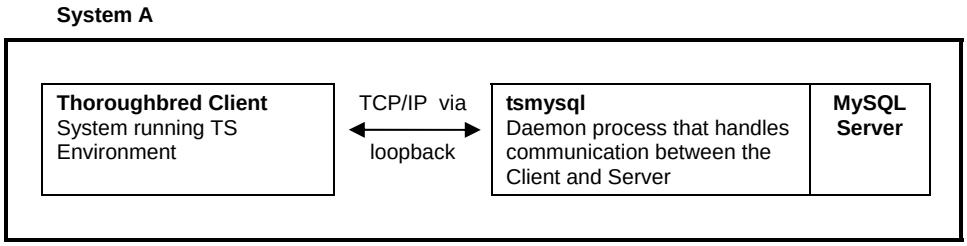
The Server Process (a child tsmysql) is triggered by requests from the client. Once a request is received, it is processed and the results returned to the client.

System Architecture

The following diagrams outline the core options for configuring the components of the Thoroughbred client.

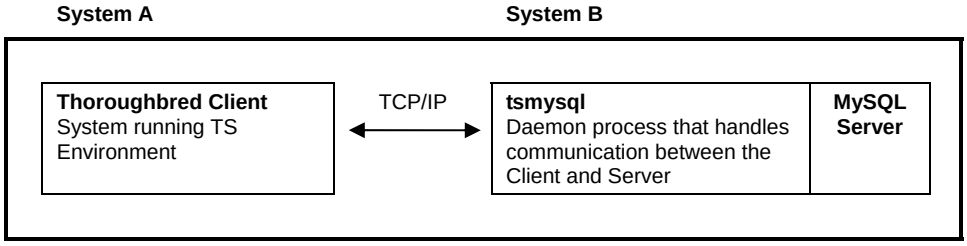
- Client and server on one system
- Client on one system - server on another

Client and Server Co-exist on One System



In the layout described by the above figure, the information is sent to the Server Process through the loopback TCP/IP address.

Client on System A and Server on System B



In the above figure the information is sent over the network to the server system. Once the requests reach the Server Process it creates SQL statements that are interpreted by the server system. The results are then passed back to the client through the Server Process.

CLIENT AND SERVER ENVIRONMENT

To set up TS DataServer for MySQL, the following files must be created or modified:

- SERVER.MAP
- IPLINPUT
- tsmysql.ini

SERVER.MAP

The SERVER.MAP text file establishes a relationship between a 2-character server ID used by Thoroughbred and the server system. This file is a simple text file.

NOTE: For UNIX Clients, this file should be placed in the **/usr/lib/basic** directory.

Thoroughbred uses the Server ID to reference the server system. Each line in the file represents an entry for one Server. The syntax for an entry is:

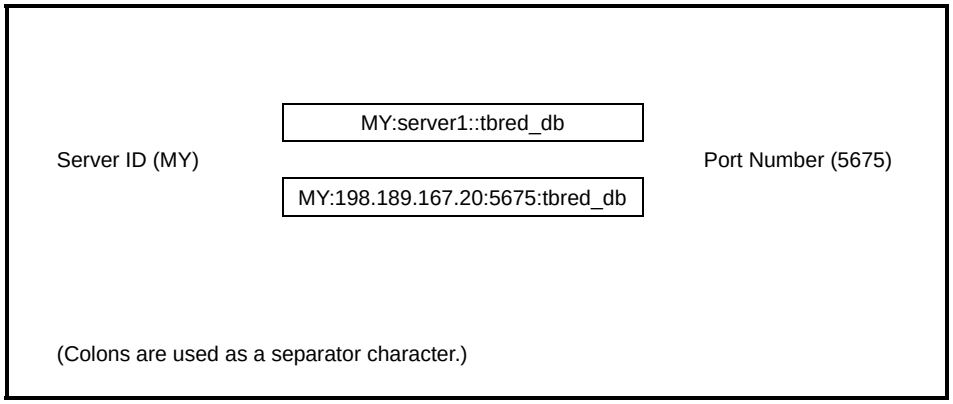
server-id*:[*host-name or TCP/IP-address*]:*TCP/IP-port* :*database

server-id	is a 2-character alphanumeric string used within Basic to reference a particular server system.
host-name	is the host name for the server system. For installations where the Thoroughbred client and database server environments reside on the same physical system (Windows Network Basic) this field can contain the loop back address or host name.
TCP/IP-address	is the TCP/IP address for the server system. For installations where the Thoroughbred client and database server environments reside on the same physical system (Windows Network Basic) this field can contain the loop back address or host name.
TCP/IP-port	port number is an optional parameter. It is used to override the default port number (5675), in the event of a conflict with another process using TCP/IP. The transport protocol layer of the TCP/IP process uses the port number to deliver the packet data to the appropriate application. In the event of a conflict on a particular system, add the port number to the SERVER.MAP file and execute the server process using the port number as an argument. For more information see Server Process in the following section.
database	this is the name of the database where Basic will access tables.

Example

The fields in a SERVER.MAP entry are separated by a colon (:). The following are examples of valid SERVER.MAP entries:

Server ID or TCP/IP Address



IPLINPUT

Once you establish the server ID you can make the required entries for the IPLINPUT file.

DEV

Each DEV line corresponds to a directory on the Server. The syntax for a DEV line is:

```
DEV device-name,1,,server-flag,,logon-cache,,server-id:log/pswd
```

device-name	is device ID used to reference logical disk directories. Valid values range from D0 through DZ.
server-flag	indicates that this will be a client to the configured server system. The value for the MySQL is 5.
logon-cache	configures a value for the maximum number of active logins to the MySQL database. For more information see Logon Cache in the Performance Tuning section.
server-id	is the two-character alphanumeric code assigned to this server. The server ID on the DEV line must match an entry in the SERVER.MAP file.
log/pswd	when using MySQL authentication, this is the MySQL login and password used to access the database. The id and password are optional. If omitted, the system prompts for the information when the client is executed.

PRM

```
PRM CREATWDBATTR
```

This parameter will create new tables using the attributes from the system dictionary. Mandatory fields (Entry Type 2 and 3 and the first key field) will be created with a NOT NULL constraint. In addition any fixed length fields (Entry Type 1 or 3) will be created with the CHAR data type so the field always contains data.

```
PRM UNIQUE-KEYS
```

This parameter will force all table creates to have unique secondary keys. This is accomplished by adding the Primary key to each sort definition. Some accommodations are made when the same data element name is used more than once in a sort definition. This is not supported by MySQL.

Setup Example

This environment consists of two MySQL databases on two different host systems.

Existing server-name TCP/IP address information:

```
acct:125.32.1.1::acct_db
dev:125.32.1.10::dev_db
```

The SERVER.MAP file may be set up in one of the following ways:

- Using host names
  S1:acct::acct_db
  S2:dev::dev_db

- Using TCP/IP addresses
  S1:125.32.1.1::acct_db
  S2:125.32.1.10::dev_db

- Using a combination of host names and TCP/IP addresses
  S1:125.32.1.1::acct_db
  S2:dev::dev_db

In this example the login information to the MySQL database is as follows:

acct: login id = mary
password = mjs

dev: login id = kurt
password = ktc

The IPLINPUT should reflect the following:

```
DEV D8,1,,5,,,,S1:mary/mjs

DEV D9,1,,5,,,,S2:kurt/ktc
```

After Basic is executed, there will now be a tsmysql process on each of the server systems (acct and dev).

tsmysql.ini

The tsmysql.ini contains control parameters for the tsmysql server process. The tsmysql.ini file is located in the /usr/lib/basic directory.

CLIENT-SYSTEM-INFO

This flag will enable process and host information to be logged. For more information see Using the Debugging Facility.

COMMIT-COUNT=#

This entry is used to specify the commit count. The maximum commit count is 65535. If the specified commit count is greater than the maximum, the maximum value will be used. The default commit count is 1.

The commit count can be overridden by SET CTC in Basic.

DEBUG-LEVEL=#

This entry is used to specify the debugging level. For more information see Using the Debugging Facility.

LDA-CACHE=#

The entry is used to specify the logon cache count. For more information see Logon Cache.

NO-OPENLOCK

The flag will disable the use of the TS_TABLE_LOCK table.

UNIQUE-SORTS

This flag will enable tables to be created with unique secondary keys.

Start the Server Process (tsmysql)

The tsmysql program is a system daemon that is installed on the server system. Each time a Basic client executes, a child tsmysql process is started. It should be added to the appropriate startup procedure, but it must be executed subsequent to the execution of the database server program. Verify that the account that executes the startup procedure has any required MySQL shell values. Consult your MySQL DBA for the correct shell variable names and values.

This program is executed with the following syntax:

tsmysql [-dx] [port-num] [-s]

-dx	is the debugging level. For more information see Using the Debugging Facility.
Port-num	is the TCP/IP port number for the process. The default is 5675.
-s	causes process and host information to be logged. For more information see Using the Debugging Facility.

The tsmysql process requires /usr/lib/basic to exist on the system where it is executed. Once it is started, execute Thoroughbred Basic using the IPLINPUT file that contains the reference to the Server IDs.

DICTIONARY-IV FORMATS AND LINKS

You must have a Format and Link defined to access a MySQL table.

Format Definition

Although no changes are required to existing formats in order to use the client capabilities of the environment, you should be aware of the following information.

The data element names must not be MySQL reserved words. For more information see the appropriate MySQL documentation.

File Type

When Thoroughbred creates an MySQL table, data types are translated as follows:

```
Alphanumeric:  VARCHAR or CHAR
Text field:    TEXT
Numeric:       DECIMAL
Date:          DATETIME
```

The VARCHAR and CHAR data types only support characters above hex 20 to be included in the data. Any character below hex 20 is converted to a space.

The CHAR type will be used when the Basic PRM CREATWDBATTR is configured and the data element is set as a fixed length field either mandatory or optional (Entry Type 1 or 3).

When records are written to this table, any field of the CHAR type will be assigned a value of at least one space. This will insure that the column is not null. For more information see PRM CREATWDBATTR in the IPLINPUT topic.

Entry Type

All mandatory fields (Entry Type 2 and 3) will be set as NOT NULL fields in the MySQL tables when the Basic PRM CREATWDBATTR is set.

Data Elements – Naming

MySQL only allows alphanumeric characters, the currency symbol ($) and an underscore (_) in column names. The first character of the column must start with a letter. Any deviation from this convention results in table creation failure.

The tsmysql program translates the hyphen (-) that is acceptable in Thoroughbred Dictionary-IV Formats to an underscore (_).

The use of MySQL reserved words also results in table creation failure. For more information see your appropriate MySQL reference material.

Data Elements – Multiple Occurrences

MySQL does not support multiple occurrences for a single data element. For more information, see Multiple Occurrences in the Dictionary-IV Developer Guide.

The system creates separate columns in the MySQL table from a multiple occurrence field. The following is the naming convention for multiple occurrences.

`data-element-name_seq#`

seq# is the range from one to the number of defined occurrences.

Binary Numeric Fields

Fields defined as binary in the Format are converted to decimal values and stored in MySQL DECIMAL fields. The type of binary field determines the length of the field.

The following table displays the formulas for calculating the field width in the MySQL table.

Formulas for Calculating Field Widths

Numeric Type	Field Width	Formula for MySQL column Width L = Field Width
3,5	1 - 3	(L*2) + 1 + 1
	4 - 5	(L + 1)*2 = 1
	6 - 8	(L + 1)*2 + 1 + 1
4	1 - 2	(L*2) + 1
	3 - 4	(L + 1)*2
	5 - 7	(L + 1)*2 + 1
	8	(L + 2)*2
6	All	(L*2) + 1
7	10	16
	Odd Prec.	(((L*10) - 15)*2)/10
	Even Prec>	(((L*10) - 5)*2)/10 – 1
8	All	8
9	All	14
A	All	(L*2) - 1 + 1
B	All	(L*2) – 1
C	All	(L*2)
D	All	L + 1
E,F	All	L

In all cases: if the value contains a decimal, add one to the field width.

SQL Dates

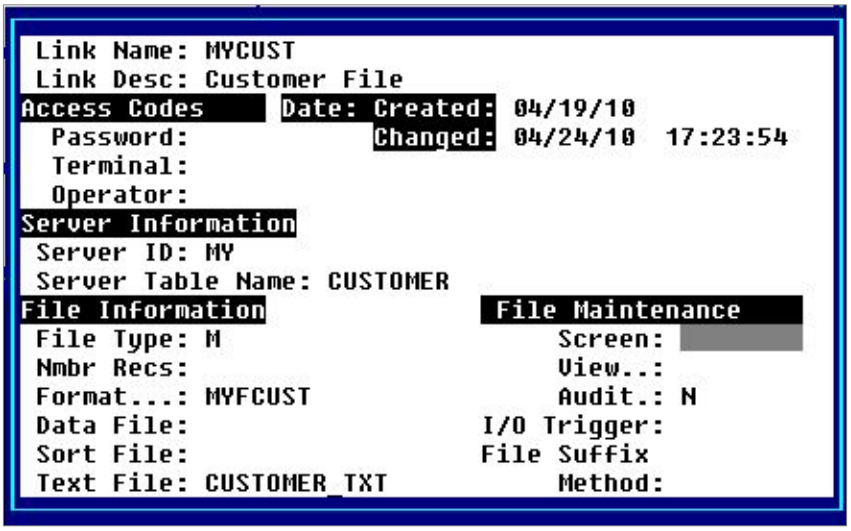
The system automatically translates dates stored in either of the SQL formats into the standard SQL DATETIME storage format as you update records.

Link Definition

The Link definition contains the configuration parameters necessary to describe a server table to Dictionary-IV. To access the MySQL tables, you must enter the following information in the Link definition:

- File Type: M
- Server ID
- Table Name

Below is a sample Link definition screen:



Server ID
The server ID is a unique identifier that references the host via the definition in the SERVER.MAP file. It will be validated against the list of server devices configured in the IPLINPUT file. Although no ID will be disallowed, a warning will be displayed.

Server Table Name
The Table Name is the name used to create a table on the server.

Text File
A Text File value is required if the Format contains Text fields. In the example above, the Text Field data will be maintained in an MySQL table named CUSTOMER_TXT. If the Text File name had a .TXT extension, the MySQL table name would be converted to use _TXT.

Both the table and text file names must conform to MySQL table naming conventions. For more information see the appropriate MySQL document to review the restrictions.

Links must be defined as a file type M (MSORT). While the physical file is not stored as an MSORT file, it will be processed as one in Dictionary-IV for sorts and text field storage. For more information see Using Sorts and Text Fields for MySQL Data.

For Dictionary-IV to detect that a Link references a server-based table, File Type, Server ID, and Table Name must all contain valid information. If they do not a local file will be used, referenced by the name entered in the Data File field.

The system supports file suffixes. The file suffix character (@) must be placed in the Table Name field for server files, rather than the Data File field. Where text fields are configured and a file suffix is used, the text file name must include the suffix character. For more information refer to the Dictionary-IV User Guide.

Sort definition occurs in the same way for server Links as for local Links and should be saved and updated with **F8** processing. Even though no physical rebuild takes place, this will properly update the Link header. For more information see the Dictionary-IV Developer Guide.

If sort definitions exist in the Link and the table is recreated, the indexes will be defined automatically on the server as part of the create process.

Under the current Dictionary-IV methodology, first build a Link header and create the table. Modify the Link header to define sorts. The next time the table is created, the indexes for all sorts will be defined on the server.

Once the Link header is configured, multiple and single record maintenance, CONNECTs, reports, queries, and recompiled scripts can access server tables. For more information about modifying Basic programs to support access to server tables see Using Basic.

While the data and sort file names may be populated during the Link header creation, neither is referenced during server access. However, the data file name is used to generate the text file name for text field storage. Therefore be sure to either enter the data file name for the automatic generation of a text file name or manually enter the field.

MySQL Tables

The following describes how to process MySQL tables from the TS Environment.

Accessing MySQL Tables

The following describes how to create MySQL tables and then access these tables from the TS Environment. For more information specific to Script-IV and Dictionary-IV see Script-IV/Dictionary-IV Tables.

Using Basic

Existing server files are accessed using the following syntax:

```
OPEN(CH,OPT="LINK")"link-name"
```

"LINK" specifies that the file to be opened on this channel is found in the system dictionary in the link-name record.

"link-name" specifies a valid link-name containing the necessary server references.

New server files may be created from the Basic environment as follows:

```
OPEN(CH,OPT="LINK|CREATE")"link-name"
```

"CREATE" is a parameter that causes Basic to generate the server commands necessary to build the file. The MySQL table definition is based on the format listed in the Link Definition. If the table already exists on the MySQL, the CREATE is ignored and the table opened.

A special OPEN syntax is used for access to text field data. This syntax is used internally by the Dictionary-IV system for accessing text field data. For more information see the Text Fields for MySQL section later in this manual.

```
OPEN(CH,OPT="LINK|TEXT")"link-name"
```

"TEXT" specifies that the text file from the link is to be opened.

The following code fragments show two ways of using the new OPEN syntax to access files. The examples assume a Link definition with information in the Server ID and Table Name fields and a file type of M.

Example 1

```
OPEN/READ server file into IOLIST (variables)

10       IOLIST CUST_CODE$,CUST_NAME$,ADDRESS$

100      CH=UNT;
         OPEN(CH,OPT="LINK") "UTCUST"
         READ(CH,KEY="0001")IOL=10
```

This example assumes a Format containing field separators.

Example 2

```
OPEN/READ server file into FORMAT

10       CH=UNT;
         OPEN(CH,OPT="LINK") "UTCUST";
         FORMAT INCLUDE #UTCUST;
         READ(CH,KEY="0001")#UTCUST
```

This example can be used on any format with or without field separators.

Example 3

```
OPEN/READ server file into record variable

10       CH=UNT;
         OPEN(CH,OPT="LINK") "UTCUST";
         READRECORD(CH,KEY="CUST0001)"REC$
```

This example shows access using a Format without field separators. The data in REC$ could be used to populate the format data area.

Under certain circumstances an error 167 may be returned when writing to a Link opened with OPT="LINK". When employing a Format directly or a Format that contains field separators, the system assigns that data to the data element name.

If this data is invalid for the particular data element name (invalid value for Yes/No fields, blank mandatory fields, or invalid data in a date field) an error 167 occurs. Printing the DNE system variable at the instance of the error will display the data element name for which the assignment failed.

ERASE

Server tables are deleted from within the Basic environment by specifying the Link name and OPT="LINK".

```
ERASE"UTCUST",OPT="LINK"
```

This deletes the table from the server referenced in the Link UTCUST. It also deletes the text field table if any text fields were defined.

INITFILE

Server tables may be cleared of all data by using the INITFILE directive. This directive is useful for clearing a table without the extra step of performing an erase.

```
INITFILE "UTCUST",OPT="LINK”
```

This clears the table on the server referenced in the Link UTCUST by doing a DROP followed by an ADD. It also clears the text field table if any fields are defined.

RENAME

The rename function allows a table name to be changed on the server from the client process.

```
RENAME "UTCUST","UTTEMP",OPT="LINK"
```

This renames the table referenced in Link UTCUST to the table UTTEMP. No changes are made to the original Link.

FID Function

When the FID is executed on a channel where a Link referencing a server table is opened, the resulting string contains (in addition to the standard FID information) the following:

Byte 1 Value is "S". Indicates that this is a channel opened to a link referencing a server table.

Byte 2 - 3 The two character Server ID from the Link.

The bytes that normally contain the file name now contain the Link name.

TCB Function

Error 150 has been added to the Basic error list to reflect errors resulting from server operations. To return the specific server error, an argument has been added to the TCB( ) function. The TCB(20) now returns the server error in the same way that the TCB(3) returns the operating system error. The value returned for the TCB(20) can be interpreted by referencing the server error guide.

DUMP Directive

Several of the DUMP options have been changed to be aware of server access.

DUMP CHANNELS now display the link name when a link has been OPENed with OPT="LINK".

DUMP IPLDEVS will specify that devices configured as a server will return "Access using server xx."

XFD Function
The following describes the current values returned for server tables:

Option = 0

Bytes	*Description*
1-2	Unused
3-10	N/A
11-35	Unused
36-38	N/A
38-41	N/A
42-47	Unused
48	"N"
49-53	Unused
54-59	N/A
60-65	N/A
66-71	N/A
72-85	Unused
86+	N/A

Option = 1
As documented in the Thoroughbred Basic Reference Manual.

Option = 3

Bytes	*Description*
1	05
2	83
3	Number of sorts (binary)
4	Current sort number (binary)
5-12	"NNY" (bytes 8-12 Unused)
13-14	N/A
15-16	COBOL only
17-42	Unused
45+	Sort definition parameters (For more information see the Thoroughbred Basic Reference Manual.)

Option = 6

Bytes	*Description*
1	06
2	00
3-6	00000000
7-8	Bytes per record (binary)
9+	Link name

Option = 10

Returns -1

FST Function
This function is not supported for server tables.

IND Function

MySQL does not support an IND function. IND for a channel opened to an SQL Server table will always return 1. Errors will be reported when trying to access SQL Server tables using IND.

NOTE: The string following OPT= can be a variable in any of the commands where it is specified. The value does not need to be hard-coded into the program.

Using Sorts

Sorts, also known as indexes, are handled in substantially the same manner for server tables as they are for local files. Created in the Link definition, they can be used for access in any operation that currently supports sorts. As with MSORT, if the Link references an existing file, the link definition can be updated to include sorts defined in the file without rebuilding the file. The differences occur in the way that sorting is applied to the server table.

Indexes do not need to physically exist in the server environment to use Link sorts anywhere in Dictionary-IV. As long as the sort definition is established in the Link definition, Dictionary-IV can use it to access the MySQL tables. Basic will expand the sort statement defined in the Link into the appropriate field names, rather than passing only the sort number.

MySQL sort statements will refer to these field (column) names, not index names, when defining an ORDER BY statement. So if the index does not exist, MySQL performs the sorting operation in-line. However, if a MySQL index exists for the columns in the ORDER BY statement, the index is used to access the data file. However, the Indexes will be used only if one of the columns has been defined with a NOT NULL constraint. For more information see the Performance Tuning section later in this manual.

If sort definitions exist in the Link and the table is recreated, the indexes will be defined automatically on the server as part of the create process.

Under the current Dictionary-IV methodology first build a Link header and create the table. Modify the Link header to define the sorts. Erase and recreate the table. The indexes for all sorts are now defined on the server.

MySQL does not support sorts on substrings of fields.

The sorts (indexes) are created using the following naming convention:

"I_" + *table-name* + " _ " + *Dictionary-IV sort number*

Example:

For sort number 2 on a Link with a table-name defined as TSISERVER, the index would be created as "I_TSISERVER_2".

Adding and deleting sorts is not supported from within the Basic client (ADDSORT/REMSORT). These functions are reserved for the database administrator.

Processing existing MySQL Tables

To access existing MySQL tables you must build a Format and Link with the appropriate table information. Be sure to use the exact spelling for data element names and the table name.

MySQL supports many data types. Not all of the MySQL data types have a one to one Thoroughbred equivalent. Many of the data types can be derived using Basic code. Be aware of the following MySQL constructs:

- Variable length fields where data greater than 255 characters.
- Numeric fields exceeding Thoroughbred's maximum of 14 significant digits.
- Column names exceeding 20 characters in length.

You may be able to access a table that contains the above characteristics for reading and updating. By defining a format that contains a partial list of the fields defined in the table, data can be read and updated without affecting the fields not included in the format.

Text Fields for MySQL Data

Dictionary-IV treats server tables internally as MSORT for I/O processing. Therefore text field data associated with servers must be stored in a separate file as is the current convention with MSORT type files. This file, referenced as the text file in the Link header, must be stored on the server with the main table to maintain data integrity, to provide access to all clients, and for backing up.

If the Format defines one or more text fields then a separate table for the text data is automatically generated when the table for the record data is created. The record data is stored in the table referenced by the Table Name field and the text field data is stored in a table referenced by the Text File field. If present, a dot (.) in the file name is converted to an underscore (_).

The definition of the text table is derived from the record data and a number of preset fields that are used internally by Dictionary-IV. This does not appear as a Dictionary-IV Format but exists as an MySQL table. A logical Format called IDSV1 has been created to aid in the handling of this information. It is used by the system and is not associated with any specific Link.

The MySQL table layout is as follows:

```
KEY_FIELD       varchar (len=length of primary key)
TEXT_ID         varchar (len=1)
CREATE_DATE     datetime
CHANGE_DATE     datetime
TEXT_WIDTH      decimal (3,0)
TEXT_FIELD      text
```

KEY_FIELD
Contains the key to the record. This is similar to the current key structure of text field data, except there is no need for a sequence counter since the entire text field is stored in one record. This is possible because the text data type has a maximum storage capacity of 2 gigabytes.

TEXT_ID
Contains the text field identifier. (See Format definition).

CREATE_DATE
Specifies the creation date.

CHANGE_DATE
Specifies the date of the last change.

TEXT_WIDTH
Used by Dictionary-IV for compatibility to local MSORT processing.

TEXT_FIELD
Because this text data has been made available as a MySQL table, the data is comprised of purely printable characters (except line terminating $0A$). Therefore any color or attribute information is stripped prior to storage in the table. Graphic characters are converted to a standard ASCII character representation. Changes made from an MySQL application are available to the Thoroughbred application.

All file level access (create, erase, and clear) performed on the record table is also executed on the text table. However all record level access requires a distinct OPEN of the text table. This is consistent with the current MSORT text field operation.

During I/O processing a FF is implied in the first byte of the key. This byte does not appear in the data but is required for programming consistency. The byte is required for all key access and is returned in all key functions (KEY, FKY, LKY, and PKY).

The Text field data is automatically maintained when accessed via Dictionary-IV. No changes are required to CONNECTs, Scripts, Reports, or Queries that currently use Text fields.

PERFORMANCE TUNING

The following describes how to maximize the throughput using sorts and process cache.

Sorts

To maximize the throughput for a number of the Basic I/O operations that operate on server tables, you must carefully plan the design of the table. The client process creates generic MySQL tables. It makes no assumption about the data to be contained in the record other than what it determines from the Dictionary-IV Format. This is sufficient to describe the table to MySQL, but under some conditions, sorting, for example this may not be enough.

MySQL creates result sets that it generates from statements that specify ordering whether or not an index exists for the columns listed in the ORDER BY clause. Without an index this operation is very time consuming for large tables. Within the Basic environment this is multiplied since each access to the table may require an in-line sort. Basic saves the result set from a particular query and uses it if the subsequent query matches the one that generated the result. If it does not match, the result set must be regenerated the next time the query is executed.

The solution is to create an index for the columns to sort and define at least one column as NOT NULL (a mandatory field in Dictionary-IV). This must be done or the index is ignored for ORDER BY processing.

The create process translates mandatory fields and the first column of the primary key into columns with the NOT NULL constraint if PRM CREATWDBATTR is configured. If these columns are part of indices you need no further changes.

Degradation due to indices being ignored becomes evident in:

Basic
using LKY( ), PKY( ), and I/O with sorts

Dictionary-IV
File access with sorts (multi-record maintenance, view maintenance, sort order change) and backward reads in files (multi-record maintenance up arrow and page up)

Script-IV
READ PREVIOUS, LAST TO FIRST, sorts, etc.

Logon Cache

All access to the MySQL database occurs through one login to the server using multiple cursors. When a LOCK, EXTRACT, or WRITE is performed in Basic a new login is executed to preserve key pointers to the other tables opened on the original login (LOCKs, EXTRACTs, and WRITE will perform commits and will clear all record locks). Once the LOCK, EXTRACT, or WRITE is complete, this login is terminated. The logon cache count parameter in the IPLINPUT file is used to specify the number of login processes to retain.

For example, if the logon cache count is set to 5, a pool of up to 5 SQL login processes will remain open. The next LOCK, EXTRACT, or WRITE executed by Basic will attach itself to one of these 5 processes bypassing the overhead to login and establish a new process.

By configuring this value properly a balanced maximum can be maintained between excessive login processing and the number of active logins. A maximum of 20 logins can be cached.

USING THE DEBUGGING FACILITY

SQL Output

A debugging facility is available from the tsmysql process. Execute this program with the following syntax to activate the debugger.

tsmysql -dn

n Currently the only debug level is 1. This sends the generated SQL statements to the log file (/usr/lib/basic/tsmysql.debug). The output in the log file contains a separate line for each generated SQL statement prefixed by the child tsmysql process id.

While tsmysql is active, output for all clients (located on the same server) is sent to one tsmysql.debug file.

If the tsmysql.debug file is copied, output will continue to be placed in the renamed file until the parent tsmysql process is terminated. A subsequent Basic session would then use a new tsmysql.debug.

Process Identification

A method is available to associate the child process tsmysql process with the client process. Execute this program with the following syntax to enable this feature.

tsmysql -s

By including the –s argument on the tsmysql command, each time the client logs into the server, information is sent to the tsmysql.pid file. This file is located in /usr/lib/basic. The format of the information is as follows;

Line 1 parent tsmysql process

Line 2 – n [child tsmysql process] [client host name] [client basic process]

GENERATED SQL CODE

This section shows the resulting SQL syntax for Thoroughbred functions (Italic typeface). They are presented in alphabetical order.

SQL Syntax for Thoroughbred Functions

CLOSE (channel)

COMMIT
This is performed via a direct MySQL call, not as a literal SQL statement.

ERASE

DROP TABLE table-name
If text fields are configured: DROP TABLE text table-name

EXTRACT[RECORD] (channel)

SELECT columns FROM table-name WHERE key-conditions >= current-key
ORDER BY sort-information
Get key from first record or current SELECT
SELECT columns FROM table-name WHERE key-conditions = current-key
ORDER BY sort-information FOR UPDATE

EXTRACT[RECORD] (channel,KEY=)

SELECT columns FROM table-name WHERE key-conditions = current-key
ORDER BY sort-information FOR UPDATE

FKY( )

SELECT columns FROM table-name WHERE key-conditions
ORDER BY sort-information
Get the first record.
Extract the key information from the record.

INITFILE

TRUNCATE TABLE table-name
If text fields are configured: TRUNCATE TABLE text-table-name

KEY( )

If a record is not extracted
SELECT columns FROM table-name WHERE key > key of last record read
    ORDER BY sort-information
Otherwise: Extract the key information from the record.

LKY( )

SELECT columns FROM table-name ORDER BY sort-information DESC
Get the first record.
Extract the key information from the record.

LOCK (channel)

SELECT 1 FROM table-name WHERE 1=2

OPEN (channel,OPT="LINK|CREATE")

CREATE TABLE table-name
If text fields are configured: CREATE TABLE text-table-name

For sort0:
CREATE UNIQUE INDEX I_table-name_0

For sortn:
CREATE INDEX I_table-name_n

OPEN (channel, OPT="LINK")

SELECT 1 FROM table-name WHERE 1=2

OPEN (channel, OPT="LINK|TEXT")

SELECT 1 FROM text-table-name WHERE 1=2

PKY( )

If a record is not extracted
SELECT columns FROM table-name WHERE key-conditions
    ORDER BY sort-information DESC
Otherwise: Extract the key information from the record.

PREAD[RECORD] (channel)

If a SELECT is in memory, get the next record
Else
SELECT columns FROM table-name WHERE key-conditions < current key
    ORDER BY sort-information DESC

READ[RECORD] (channel,KEY=)
FIND[RECORD](channel,KEY=)

If a SELECT is in memory, get the next record
Else
SELECT columns FROM table-name WHERE key-conditions >= current key
    ORDER BY sort-information

REMOVE (channel,KEY=)

SELECT columns FROM table-name WHERE key-conditions
    FOR UPDATE NOWAIT
DELETE FROM table-name WHERE key-conditions

RENAME

RENAME TABLE table-name TO new-table-name

UNLOCK (channel)

No SQL equivalent. Data will be committed and Basic's internal lock on the channel will be released.

WRITE[RECORD] (channel,KEY=)

Get key value from the record data
SELECT columns FROM table-name WHERE key-conditions = current-key
    FOR UPDATE
If the record exists:
UPDATE table-name SET column = value,...
Else:
INSERT INTO table-name column ... VALUES (value...)
fi
COMMIT
This is performed via a direct MySQL call, not as a literal SQL statement.

Multiple Select Statements

The following Basic commands may produce multiple select statements when sequentially accessing tables with multiple key fields defined in the Format:

READ[RECORD]
FIND[RECORD]
PREAD[RECORD]
EXTRACT[RECORD]
KEY( )
PKY( )
FKY( )

Unsupported Basic Directives

There is no equivalent SQL code for the following Basic directives:

ADD
ADDSORT
FILE
REMSORT

These items are not supported when applied to server tables.

TECHNICAL REFERENCE

The following provides information specific to Script-IV, Dictionary-IV, Thoroughbred Basic, and MySQL views.

Script-IV/Dictionary-IV

The following information is specific to Script-IV and Dictionary-IV

DATA-FILE IS, SORT-FILE IS

This feature will be supported but the object referenced by this command must be a Link rather than a local data file. The Link can then reference either a server table or a local file. The file type set in this Link must be "M".

Access Subset of Columns

By defining a format that contains a subset of the columns that are defined in a table, access to tables that contain unsupported data types (BIGINT,IMAGE,NTEXT, etc.) is possible. All of the possible I/O directives are supported with no effect on the unreferenced fields.

Basic

Soft Includes

When an OPEN is executed on a server Link, the Format referenced in the Link header is soft included. This means that the Format data area is defined in memory. However, the Format name will not appear in the FMTNL list and will not interfere with any hard includes that take place. The soft include is executed internally as a FORMAT INCLUDE #format-name, OPT=”NONE” so that any data defined in a Format data area at the time of an OPEN of a server Link that references this Format is preserved.

If however a READ or a WRITE is executed using an IOLIST, the Format data area will be modified since the I/O operation uses the Format data area to parse the data into the proper variable values.

Enhanced Access to Tables

Where possible, sequential access to tables may provide faster access times during READ operations. This is possible because a particular I/O statement will generate a result set on the server. Subsequent READ operations will simply move through the result set and return the data to Basic. The area where this is most obvious is when a table is sorted by Basic in a way that no index exists in MySQL. The initial READ will force the sort to take place, but all READs that follow will get the data from the result set without the need to sort again. If any other I/O statement had occurred between READs another result set would have been created forcing a re-sort on the next READ statements. The following is NOT considered a sequential READ:

K$ = KEY(CH); READ (CH,KEY=K$)

This code generates two separate result sets. Each READ that specifies a specific key generates a new result set. A sequential READ looks like:

READ (CH,ERC=2)

Reestablish a Connection

To reestablish a connection, use the following command:

ENABLE disk-no,”RECONNECT”

Where *disk-no* is the DEV line designation to the MySQL server.

SET CTC and CTC()

These functions allow setting and retrieving the current Commit Count from Basic.

A = CTC(“DQ”)

This retrieves the current commit count for database connection associated with DEV DQ.

SET CTC “DQ”,1000

This sets the commit count to 1000 for the database connection associated with DEV DQ.

Tables

MySQL Views

Where possible the use of an MySQL view as the table in Link definition should be avoided. Views will not support all of the SQL statements that are generated by Basic statements. The most common of these will be the SELECT FOR UPDATE which is the EXTRACT directive in Basic. Therefore it is possible that server errors (ERR=150) will occur accessing Views which will interrupt normal program functioning.

Also for the View data to be returned in a sorted order, the sort must be explicitly defined in the definition of the view in MySQL. Merely defining key fields will not produce sorted results. A sort must be defined in the link header and then used in all I/O statements.

ERROR MESSAGES

The following describes client error messages.

Client/Server Basic Error Messages

150

Explanation: Server system error. For more information refer to TCB(20) to get the server error number. Also see Virtual Error Table below.

171

Explanation: Undefined Link.

Thoroughbred Basic Startup Messages

No match for server id in SERVER.MAP

Explanation: The two-character server id defined in the IPLINPUT file cannot be located in the SERVER.MAP file.

Action: Verify contents of both files.

Could not resolve Network address for Thoroughbred foreign file server

Explanation: System does not understand the Network address configured in the SERVER.MAP file.

Action: Verify that the address or host name in the SERVER.MAP file is a valid value for the system. Also verify that the /etc/hosts file is accessible. This file must be readable for all users. On systems that use DNS, be sure the system is configured.

Invalid Thoroughbred foreign file server type

Explanation: The system found an invalid value in the server type parameter in the IPLINPUT server device line.

Action: Verify that the server device line is defined properly with a Server type of 5.

Could not create client-end connection to Thoroughbred foreign file server

Explanation: This can occur if TCP/IP has not been started or configured on the client system.

Action: Verify the client machine can be reached over the network.

Could not connect to Thoroughbred foreign file server

Explanation: Thoroughbred Basic could not establish communication to tsmysql.

Action: Verify that tsmysql is started.

Thoroughbred foreign file server could not connect to foreign database

Explanation: tsmysql could not establish communication to the server.

Action: Verify that the server processes are started and that the account starting tsmysql is has shell variables set appropriately.

MySQL Error Messages

When a Basic error 150 occurs, the MySQL Error number will be returned in the TCB(20). Refer to the MySQL documentation for explanation for each error.

TROUBLESHOOTING

The following provides troubleshooting information specific to MySQL.

Link Error

Error 0 occurs opening the Link and no locks seem to be active in Basic or from an SQL application.

Since the Link header is read internally by Basic when opening the Link, this record needs to be accessible at open time. If it is extracted through Dictionary-IV, for example, the error 0 occurs. Setting the PRM READONLY parameter in the IPLINPUT file will clear this situation.

Server Table Error

Errors opening server tables.

- Verify that the DEV line for the server where the table resides is present in the appropriate IPLINPUT file.
- Verify the user has MySQL credentials for the Database and Table.
- Verify the Link has a Table name. If the associated Format has text fields defined, verify that the Link also has a Text File Name.

Unexpected Sorting Sequence or Errors Accessing SQL Tables

- If a Link header has been configured to access a server table defined on the server as a view, unexpected results ranging from incorrectly sorted data to tsmssql process failures may result. It is suggested that access to the server be limited to objects defined as tables. SQL does not support certain operations on views such as SELECT ...FOR UPDATE. This may also occur if a table has no index and the fields in the sort are defined as NULL. This can be resolved by adding a index and/or changing the field constraint to NOT NULL.
- Be sure that the primary key index is present.

Some Access Using Sorts is Very Slow

Sorts that contain references to substrings of fields may not perform as well as sorts on full fields. This is because MySQL has no facility to define indexes on portions of fields. For more information see the Performance Tuning section of this manual.