MySQL 5.0 Data Dictionary

MySQL 5.0 New Features Series – Part 4

A MySQL® Technical White Paper
Trudy Pelzer
March, 2005

Table of Contents

Introduction

This book is for the long-time MySQL user who wants to know "what's new" in version 5. The short answer is "stored procedures, triggers, views, and information schema". The long answer is the MySQL 5.0 New Features series, and this book is the last in that series.

Conventions and Styles

Whenever I want to show actual code, such as something that comes directly from the screen of my `mysql` client program, I switch to a Courier font, which looks different from the regular text font. For example:

```
mysql> CREATE TABLE table1 (column1 INT);
Query OK, 0 rows affected (0.00 sec)
```

When the example is large and I want to draw attention to a particular line or phrase, I highlight it with a double underline and a small arrow on the right of the page. For example:

```
mysql> CREATE VIEW v AS
    -> SELECT column1 AS c /* view col name is c */     <--
    -> FROM table1;
Query OK, 0 rows affected (0.01 sec)
```

Sometimes I will leave out the `mysql>` and `->` prompts so that you can cut the examples and paste them into your copy of the `mysql` client program. (If you aren't reading the text of this book in a machine-readable form, try looking for the script on the mysql.com web site.)

All of the examples in this book were tested with the publicly-available alpha version of MySQL 5.0.3 on the SuSE Linux operating system (version 9.1). By the time you read this, the version number will be higher and the available operating systems will include Windows, Sparc, and HP-UX. So I'm confident that you'll be able to run every example on your own computer. But if not, well, as an experienced MySQL user you know that help and support is always available.

A Definition and an Example

Standard SQL (SQL:2003) provides a method of accessing database metadata through a schema called INFORMATION_SCHEMA. Until now, MySQL has provided metadata only through a series of SHOW commands. SHOW, however, has two disadvantages:

1. SHOW commands are non-standard; they are specific to MySQL.

2. SHOW commands require that you learn an entire set of commands to be able to access the metadata you need.

In contrast:

1. Use of INFORMATION_SCHEMA is standard SQL; thus alleviating some problems that may occur in porting applications from one DBMS to another. For example, Microsoft

SQL Server also supports INFORMATION_SCHEMA, while IBM DB2 supports a similar structure, albeit with different names.

2. The tables in INFORMATION_SCHEMA can be queried via a SELECT statement, just as regular tables can be queried; thus there is no need to learn a new set of commands to be able to access the metadata you need.

So MySQL AB made the decision to implement support for the INFORMATION_SCHEMA. Effective with MySQL 5.0.2, your MySQL installation will automatically contain a schema (usually called a database in MySQL parlance) called INFORMATION_SCHEMA; it contains a set of views that allow you to look at (but not change) the description of your database objects just as if the descriptions are regular SQL data. Here is an example:

```
mysql> SELECT table_name, table_type, engine
    -> FROM INFORMATION_SCHEMA.tables
    -> WHERE table_schema = 'tp'
    -> ORDER BY table_type ASC, table_name DESC;
+------------+------------+--------+
| table_name | table_type | engine |
+------------+------------+--------+
| t2         | BASE TABLE | MyISAM |
| t1         | BASE TABLE | InnoDB |
| v1         | VIEW       | NULL   |
+------------+------------+--------+
```

Terminology Notes

Metadata refers to data about the data. For example, the name of a table and the data type of a column is metadata. There are two other terms that are often used as synonyms for metadata:
1. Data dictionary
2. System catalog

I won't be using those terms in this book.

Using INFORMATION_SCHEMA

MySQL now has a new "database" named INFORMATION_SCHEMA. It is a virtual database only; there will never be a need to create a file by that name, and the MySQL server itself creates and populates the tables therein. It is not possible to USE INFORMATION_SCHEMA; nor is it possible to UPDATE, INSERT, DELETE, or even REFERENCE the INFORMATION_SCHEMA tables. The only action possible is SELECT.

Privileges

Accessing the INFORMATION_SCHEMA tables does not require a special privilege: the SELECT privilege on each table is automatically granted to every user.

Thus, there is no difference between the current (SHOW) privilege requirement and the SELECT requirement. In either case, you have to have some privilege on an object in order to see the metadata information about that object.

The Tables

In this section, I will describe the INFORMATION_SCHEMA tables and show you how they compare to the results produced by the equivalent MySQL SHOW command.

CHARACTER_SETS Table

The standard SQL INFORMATION_SCHEMA.CHARACTER_SETS table shows the character sets that are available to the current user and provides equivalent information to the MySQL-specific SHOW CHARACTER SET statement. CHARACTER_SETS has the following columns:

• CHARACTER_SET_NAME -- shows the name of a character set which the current user may use. This column provides the same information as the CHARSET column returned by SHOW CHARACTER SET.

• DEFAULT_COLLATE_NAME -- shows the name of the default collation for the character set. This column provides the same information as the DEFAULT COLLATION column returned by SHOW CHARACTER SET.

• DESCRIPTION -- shows a descriptive name for the character set that tells you the standard character set upon which this character set is based. This column provides the same information as the DESCRIPTION column returned by SHOW CHARACTER SET and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW CHARACTER SET.

• MAXLEN -- shows the number of bytes used to store each character that belongs to the character set. This column provides the same information as the MAXLEN column returned by SHOW CHARACTER SET and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW CHARACTER SET.

Here are two equivalent commands:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.CHARACTER_SETS
    -> WHERE CHARACTER_SET_NAME LIKE 'latin1%'\G
*************************** 1. row ***************************
  CHARACTER_SET_NAME: latin1
DEFAULT_COLLATE_NAME: latin1_swedish_ci
         DESCRIPTION: ISO 8859-1 West European
              MAXLEN: 1


mysql> SHOW CHARACTER SET LIKE 'latin1%'\G
*************************** 1. row ***************************
          Charset: latin1
      Description: ISO 8859-1 West European
Default collation: latin1_swedish_ci
           Maxlen: 1
```

COLLATIONS Table

The standard SQL INFORMATION_SCHEMA.COLLATIONS table shows the collations that are available to the current user and provides equivalent information to the MySQL-specific SHOW COLLATION statement. COLLATIONS has the following columns:

• COLLATION_NAME -- shows the name of a collation which the current user may use. This column provides the same information as the COLLATION column returned by SHOW COLLATION.

• CHARACTER_SET_NAME -- shows the name of an available character set to which the collation applies. This column provides the same information as the CHARSET column returned by SHOW COLLATION and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW COLLATION.

• ID -- shows a numeric identifier for the collation/character set combination. This column provides the same information as the ID column returned by SHOW COLLATION and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW COLLATION.

• IS_DEFAULT -- shows whether the collation is the default collation for the character set shown; values are either 'YES' or 'NO'. This column provides the same information as the DEFAULT column returned by SHOW COLLATION and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW COLLATION.

• IS_COMPILED -- indicates whether the collation is compiled into the MySQL server; values are either 'YES' or blank (meaning 'NO'). This column provides the same information as the COMPILED column returned by SHOW COLLATION and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW COLLATION.

• SORTLEN -- shows a value related to the amount of memory required to sort strings using the collation/character set combination. This column provides the same information as the SORTLEN column returned by SHOW COLLATION and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW COLLATION.

Here are two equivalent commands:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.COLLATIONS
    -> WHERE COLLATION_NAME LIKE 'latin1%'\G
*************************** 1. row ***************************
    COLLATION_NAME: latin1_german1_ci
CHARACTER_SET_NAME: latin1
                ID: 5
        IS_DEFAULT:
       IS_COMPILED:
           SORTLEN: 0
*************************** 2. row ***************************
    COLLATION_NAME: latin1_swedish_ci
CHARACTER_SET_NAME: latin1
                ID: 8
        IS_DEFAULT: Yes
       IS_COMPILED: Yes
           SORTLEN: 1
*************************** 3. row ***************************
...
```

```
mysql> SHOW COLLATION WHERE COLLATION LIKE 'latin1%'\G
*************************** 1. row ***************************
Collation: latin1_german1_ci
  Charset: latin1
       Id: 5
  Default:
 Compiled:
  Sortlen: 0
*************************** 2. row ***************************
Collation: latin1_swedish_ci
  Charset: latin1
       Id: 8
  Default: Yes
 Compiled: Yes
  Sortlen: 1
*************************** 3. row ***************************
...
```

COLLATION_CHARACTER_SET_APPLICABILITY Table

The standard SQL INFORMATION_SCHEMA.
COLLATION_CHARACTER_SET_APPLICABILITY table shows the character sets that are applicable for each collation that is available to the current user. This table has no direct MySQL SHOW equivalent; instead, the information provided comes from the first two columns returned by SHOW COLLATION. COLLATION_CHARACTER_SET_APPLICABILITY has the following columns:

• COLLATION_NAME -- shows the name of a collation which the current user may use. This column provides the same information as the COLLATION column returned by SHOW COLLATION.

• CHARACTER_SET_NAME -- shows the name of a character set, which the current user may use, to which the collation applies. This column provides the same information as the CHARSET column returned by SHOW COLLATION

Here are two, roughly equivalent, commands:

```
mysql> SELECT * FROM
    -> INFORMATION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY
    -> WHERE COLLATION_NAME LIKE 'latin1%'\G
*************************** 1. row ***************************
    COLLATION_NAME: latin1_german1_ci
CHARACTER_SET_NAME: latin1
*************************** 2. row ***************************
    COLLATION_NAME: latin1_swedish_ci
CHARACTER_SET_NAME: latin1
*************************** 3. row ***************************
...
```

```
mysql> SHOW COLLATION WHERE COLLATION LIKE 'latin1%'\G
*************************** 1. row ***************************
Collation: latin1_german1_ci
  Charset: latin1
       Id: 5
  Default:
 Compiled:
  Sortlen: 0
*************************** 2. row ***************************
Collation: latin1_swedish_ci
  Charset: latin1
       Id: 8
  Default: Yes
 Compiled: Yes
  Sortlen: 1
*************************** 3. row ***************************
...
```

COLUMN_PRIVILEGES Table

The standard SQL INFORMATION_SCHEMA.COLUMN_PRIVILEGES table provides information on every column privilege granted for the current database. This table has no direct MySQL SHOW equivalent; instead, the information provided comes from a combination of the results returned by SHOW GRANTS and SHOW FULL COLUMNS. COLUMN_PRIVILEGES has the following columns:

• GRANTEE -- shows the name of a user who has been granted a column privilege.

• TABLE_CATALOG -- always shows NULL, since MySQL does not support the concept of a database catalog.

• TABLE_SCHEMA -- shows the name of the schema (i.e., the database) in which the table that contains the column resides.

• TABLE_NAME -- shows the name of the table that contains the column on which a privilege has been granted.

• COLUMN_NAME -- shows the name of the column on which the column privilege has been granted.

• PRIVILEGE_TYPE -- shows the type of privilege that was granted; either 'SELECT', 'INSERT', 'UPDATE', or 'REFERENCES'.

• IS_GRANTABLE -- shows whether the privilege was granted WITH GRANT OPTION; either 'YES' or 'NO'.

Here is an example:

```
mysql> SELECT * FROM
    -> INFORMATION_SCHEMA.COLUMN_PRIVILEGES\G
*************************** 1. row ***************************
       GRANTEE: 'peter'@'%'
 TABLE_CATALOG: NULL
  TABLE_SCHEMA: tp
    TABLE_NAME: t1
   COLUMN_NAME: col1
PRIVILEGE_TYPE: UPDATE
  IS_GRANTABLE: NO
*************************** 2. row ***************************
       GRANTEE: 'trudy'@'%'
 TABLE_CATALOG: NULL
  TABLE_SCHEMA: tp
    TABLE_NAME: t2
   COLUMN_NAME: col1
PRIVILEGE_TYPE: SELECT
  IS_GRANTABLE: YES
```

COLUMNS Table

The standard SQL INFORMATION_SCHEMA.COLUMNS table shows the columns that are available to the current user and provides equivalent information to the MySQL-specific SHOW COLUMNS statement. COLUMNS has the following columns:

• TABLE_CATALOG -- always shows NULL, since MySQL does not support the concept of a database catalog. SHOW COLUMNS has no equivalent column.

• TABLE_SCHEMA -- shows the name of the schema (i.e., the database) in which the table that contains an available column resides. SHOW COLUMNS has no equivalent column.

• TABLE_NAME -- shows the name of the table to which an available column belongs. SHOW COLUMNS has no equivalent column.

• COLUMN_NAME -- shows the name of a column that may be accessed by the current user (i.e., for which the current user has been granted a privilege). This column provides the same information as the FIELD column returned by SHOW COLUMNS.

• ORDINAL_POSITION -- shows the ordinal position of the column in the table to which it belongs. SHOW COLUMNS has no equivalent column.

• COLUMN_DEFAULT -- shows the column's default value. If this column is blank, the column has no defined default value. This column provides the same information as the DEFAULT column returned by SHOW COLUMNS.

• IS_NULLABLE -- shows whether the column may accept NULL values; either 'YES' or 'NO'. This column provides the same information as the NULL column returned by SHOW COLUMNS.

• DATA_TYPE -- shows the column's defined data type (keyword only, not the entire definition). This column provides some of the same information as the TYPE column returned by SHOW COLUMNS.

• CHARACTER_MAXIMUM_LENGTH -- shows the column's defined maximum length in characters. This column provides some of the same information as the TYPE column returned by SHOW COLUMNS.

• CHARACTER_OCTET_LENGTH -- shows the column's defined maximum length in octets. SHOW COLUMNS has no equivalent column.

• NUMERIC_PRECISION -- shows, for a column with a numeric data type, the column's defined precision; otherwise NULL. This column provides some of the same information as the TYPE column returned by SHOW COLUMNS.

• NUMERIC_SCALE -- shows, for a column with a numeric data type, the column's defined scale; otherwise NULL. This column provides some of the same information as the TYPE column returned by SHOW COLUMNS.

• CHARACTER_SET_NAME -- shows, for a column with a character string data type, the column's default character set; otherwise NULL. SHOW COLUMNS has no equivalent column, but a column's default character set can be derived from the first part of the value shown in the SHOW COLUMNS COLLATION column -- e.g., if COLLATION shows "latin1_swedish_ci", the character set is the name shown prior to the first underscore ("latin1") and the rest of the value is the collation name.

• COLLATION_NAME -- shows, for a column with a character string data type, the column's default collation; otherwise NULL. SHOW COLUMNS has no equivalent column, but the column's default collation can be derived from the final part of the value shown in the SHOW COLUMNS COLLATION column.

• COLUMN_TYPE -- shows the column's defined data type in full. This column provides the same information as the TYPE column returned by SHOW COLUMNS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW COLUMNS.

• COLUMN_KEY -- shows whether the column is indexed; either 'PRI' if the column is part of a PRIMARY KEY, 'UNI' if the column is part of a UNIQUE key, 'MUL' if the column is part of an index key that allows duplicates, or blank if the column is not indexed. This column provides the same information as the KEY column returned by SHOW COLUMNS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW COLUMNS.

• EXTRA -- shows any additional column definition information, e.g., whether the column was defined with the AUTO_INCREMENT attribute; otherwise left blank. This column provides the same information as the EXTRA column returned by SHOW COLUMNS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW COLUMNS.

• PRIVILEGES -- shows the privileges available to the current user on the column. This column provides the same information as the PRIVILEGES column returned by SHOW COLUMNS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW COLUMNS.

• COLUMN_COMMENT -- shows the comment, if any, stored for this column; otherwise left blank. This column provides the same information as the COMMENT column returned by SHOW COLUMNS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW COLUMNS.

Here are two, roughly equivalent, commands:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.COLUMNS
    -> WHERE TABLE_NAME='t1'\G
*************************** 1. row ***************************
           TABLE_CATALOG: NULL
            TABLE_SCHEMA: tp
              TABLE_NAME: t1
             COLUMN_NAME: col1
        ORDINAL_POSITION: 1
          COLUMN_DEFAULT:
             IS_NULLABLE: NO
               DATA_TYPE: int
CHARACTER_MAXIMUM_LENGTH: 11
  CHARACTER_OCTET_LENGTH: 11
       NUMERIC_PRECISION: 11
           NUMERIC_SCALE: 0
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
             COLUMN_TYPE: int(11)
              COLUMN_KEY: PRI
                   EXTRA:
              PRIVILEGES: select,insert,update,references
          COLUMN_COMMENT:
*************************** 2. row ***************************
           TABLE_CATALOG: NULL
            TABLE_SCHEMA: tp
              TABLE_NAME: t1
             COLUMN_NAME: col2
        ORDINAL_POSITION: 2
          COLUMN_DEFAULT: hello
             IS_NULLABLE: YES
               DATA_TYPE: char
CHARACTER_MAXIMUM_LENGTH: 10
  CHARACTER_OCTET_LENGTH: 10
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      CHARACTER_SET_NAME: latin1
          COLLATION_NAME: latin1_swedish_ci
             COLUMN_TYPE: char(10)
              COLUMN_KEY:
                   EXTRA:
              PRIVILEGES: select,insert,update,references
          COLUMN_COMMENT:
```

```
mysql> SHOW FULL COLUMNS FROM t1\G
*************************** 1. row ***************************
     Field: col1
      Type: int(11)
 Collation: NULL
      Null: NO
       Key: PRI
   Default:
     Extra:
Privileges: select,insert,update,references
   Comment:
*************************** 2. row ***************************
     Field: col2
      Type: char(10)
 Collation: latin1_swedish_ci
      Null: YES
       Key:
   Default: hello
     Extra:
Privileges: select,insert,update,references
   Comment:
```

KEY_COLUMN_USAGE Table

The standard SQL INFORMATION_SCHEMA.KEY_COLUMN_USAGE table shows the columns, available to the current user, that are operated on by some constraint and/or that are part of an index key. This table has no direct MySQL SHOW equivalent. KEY_COLUMN_USAGE has the following columns:

• CONSTRAINT_CATALOG -- always shows NULL, since MySQL does not support the concept of a database catalog.

• CONSTRAINT_SCHEMA -- shows the name of the schema (i.e., the database) in which an available constraint or index resides.

• CONSTRAINT_NAME -- shows the name of a constraint or index on a table that the current user may access.

• TABLE_CATALOG -- as with CONSTRAINT_CATALOG, always shows NULL.

• TABLE_SCHEMA -- shows the name of the schema in which the table that is operated on by the available constraint or index resides.

• TABLE_NAME -- shows the name of the available table that is operated on by the constraint or index.

• COLUMN_NAME -- shows the name of a column that is either all, or part of, the constraint or index key.

• ORDINAL_POSITION -- shows the ordinal position of the column within the constraint or index (not its position in the table to which it belongs).

• POSITION_IN_UNIQUE_CONSTRAINT -- shows, for a foreign key column, the ordinal position of the referenced column within the referenced unique index; otherwise NULL.

Here is an example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
    -> WHERE TABLE_NAME='t1'\G
*************************** 1. row ***************************
           CONSTRAINT_CATALOG: NULL
            CONSTRAINT_SCHEMA: tp
              CONSTRAINT_NAME: PRIMARY
                TABLE_CATALOG: NULL
                 TABLE_SCHEMA: tp
                   TABLE_NAME: t1
                  COLUMN_NAME: col1
             ORDINAL_POSITION: 1
POSITION_IN_UNIQUE_CONSTRAINT: NULL
```

ROUTINES Table

The standard SQL INFORMATION_SCHEMA.ROUTINES table shows the stored procedures -- both functions and procedures -- that may be executed by the current user. This table has no SHOW equivalent; instead, it provides equivalent information to the MySQL-specific mysql.proc table. ROUTINES has the following columns:

• SPECIFIC_NAME -- shows the name of a stored procedure (also known as a routine) that may be executed by the current user. This column provides the same information as the SPECIFIC_NAME column of the mysql.proc table.

• ROUTINE_CATALOG -- always shows NULL, since MySQL does not support the concept of a database catalog. The mysql.proc table has no equivalent column.

• ROUTINE_SCHEMA -- shows the name of the schema (i.e., the database) in which the stored procedure resides. This column provides the same information as the DB column of the mysql.proc table.

• ROUTINE_NAME -- as with SPECIFIC_NAME, shows the name of the available stored procedure. This column provides the same information as the NAME column of the mysql.proc table.

• ROUTINE_TYPE -- shows whether the stored procedure is a 'PROCEDURE' or a 'FUNCTION'. This column provides the same information as the TYPE column of the mysql.proc table.

• DTD_IDENTIFIER -- shows, for a function, the complete data type definition of the value the function will return; otherwise NULL. This column provides the same information as the DATA TYPE DESCRIPTOR column of the mysql.proc table.

• ROUTINE_BODY -- shows the language in which the stored procedure is written; currently always 'SQL'. The mysql.proc table has no equivalent column.

• ROUTINE_DEFINITION -- shows as much of the routine body as is possible in the allotted space. This column provides the same information as the BODY column of the mysql.proc table.

• EXTERNAL_NAME -- always shows NULL since, currently, all MySQL stored procedures are internal routines: they are stored in the database. The mysql.proc table has no equivalent column.

• EXTERNAL_LANGUAGE -- as with EXTERNAL_NAME, always shows NULL. This column provides the same information as the LANGUAGE column of the mysql.proc table.

• PARAMETER_STYLE -- shows the routine's parameter style; always 'SQL'. The mysql.proc table has no equivalent column.

• IS_DETERMINISTIC -- shows whether the routine is deterministic; either 'YES' or 'NO'. This column provides the same information as the IS_DETERMINISTIC column of the mysql.proc table.

• SQL_DATA_ACCESS -- shows the routine's defined sql-data-access clause value; either 'NO SQL', 'CONTAINS SQL', 'READS SQL DATA', or 'MODIFIES SQL DATA'. This column provides the same information as the SQL_DATA_ACCESS column of the mysql.proc table.

• SQL_PATH -- as with EXTERNAL_NAME, always shows NULL. The mysql.proc table has no equivalent column.

• SECURITY_TYPE -- shows whether the routine's defined security_type is 'DEFINER' or 'INVOKER'. This column provides the same information as the SECURITY_TYPE column of the mysql.proc table and is not part of the standard SQL definition. It was added to provide complete equivalence with the mysql.proc table.

• CREATED -- shows the timestamp of the time the routine was created. This column provides the same information as the CREATED column of the mysql.proc table.

• LAST_ALTERED -- shows the timestamp of the time the routine was last altered. This column provides the same information as the MODIFIED column of the mysql.proc table.

• SQL_MODE -- shows the sql_mode setting at the time the routine was created. This column provides the same information as the SQL_MODE column of the mysql.proc table and is not part of the standard SQL definition. It was added to ensure that a stored procedure always runs using the same sql_mode setting and to provide complete equivalence with the mysql.proc table.

• ROUTINE_COMMENT -- shows the comment, if any, defined for the routine; otherwise left blank. This column provides the same information as the COMMENT column of the mysql.proc table and is not part of the standard SQL definition. It was added to provide complete equivalence with the mysql.proc table.

• DEFINER -- shows the user who created the routine. This column provides the same information as the DEFINER column of the mysql.proc table and is not part of the standard SQL definition. It was added to provide complete equivalence with the mysql.proc table.

Here are two equivalent commands:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.ROUTINES
    -> WHERE SPECIFIC_NAME='curdemo'\G
*************************** 1. row ***************************
     SPECIFIC_NAME: curdemo
   ROUTINE_CATALOG: NULL
    ROUTINE_SCHEMA: tp
      ROUTINE_NAME: curdemo
      ROUTINE_TYPE: PROCEDURE
    DTD_IDENTIFIER: NULL
      ROUTINE_BODY: SQL
ROUTINE_DEFINITION: BEGIN
  DECLARE done INT DEFAULT 0;
  ...
  CLOSE cur2;
END
     EXTERNAL_NAME: NULL
 EXTERNAL_LANGUAGE: NULL
   PARAMETER_STYLE: SQL
  IS_DETERMINISTIC: NO
   SQL_DATA_ACCESS: CONTAINS SQL
         SQL_PATH: NULL
     SECURITY_TYPE: DEFINER
           CREATED: 2005-02-03 13:24:59
      LAST_ALTERED: 2005-02-03 13:24:59
          SQL_MODE:
   ROUTINE_COMMENT:
           DEFINER: root@localhost

mysql> SELECT * FROM mysql.proc
    -> WHERE SPECIFIC_NAME='curdemo'\G
*************************** 1. row ***************************
             db: tp
           name: curdemo
           type: PROCEDURE
   specific_name: curdemo
       language: SQL
 sql_data_access: CONTAINS_SQL
is_deterministic: NO
   security_type: DEFINER
      param_list:
         returns:
            body: BEGIN
  DECLARE done INT DEFAULT 0;
  ...
  CLOSE cur2;
END
         definer: root@localhost
         created: 2005-02-03 13:24:59
        modified: 2005-02-03 13:24:59
        sql_mode:
         comment:
```

SCHEMA_PRIVILEGES Table

The INFORMATION_SCHEMA.SCHEMA_PRIVILEGES table shows the schema privileges that have been granted on all databases. SCHEMA_PRIVILEGES is not a standard SQL table, nor does this table have a SHOW equivalent; instead, it provides equivalent information to the MySQL-specific mysql.db table. SCHEMA_PRIVILEGES has the following columns:

• GRANTEE -- shows the user to whom a schema privilege has been granted. This column provides the same information as the USER column of the mysql.db table.

• TABLE_CATALOG -- always shows NULL, since MySQL does not support the concept of a database catalog. The mysql.db table has no equivalent column.

• TABLE_SCHEMA -- shows the name of a schema (i.e., a database) on which schema privileges have been granted. This column provides the same information as the DB column of the mysql.db table.

• PRIVILEGE_TYPE -- shows the privilege granted. This column provides the same information as the *_PRIV columns of the mysql.db table.

• IS_GRANTABLE -- shows whether the privilege was granted WITH GRANT OPTION; either 'YES' or 'NO'. The mysql.db table has no equivalent column.

Here are two equivalent commands:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.SCHEMA_PRIVILEGES\G
*************************** 1. row ***************************
       GRANTEE: ''@'%'
 TABLE_CATALOG: NULL
  TABLE_SCHEMA: test
PRIVILEGE_TYPE: SELECT
  IS_GRANTABLE: NO
*************************** 2. row ***************************
       GRANTEE: ''@'%'
 TABLE_CATALOG: NULL
  TABLE_SCHEMA: test
PRIVILEGE_TYPE: INSERT
  IS_GRANTABLE: NO
*************************** 3. row ***************************
...
*************************** 12. row ***************************
       GRANTEE: ''@'%'
 TABLE_CATALOG: NULL
  TABLE_SCHEMA: test
PRIVILEGE_TYPE: LOCK TABLES
  IS_GRANTABLE: NO
*************************** 13. row ***************************
       GRANTEE: ''@'%'
 TABLE_CATALOG: NULL
  TABLE_SCHEMA: test
PRIVILEGE_TYPE: CREATE VIEW
  IS_GRANTABLE: NO
*************************** 14. row ***************************
       GRANTEE: ''@'%'
 TABLE_CATALOG: NULL
  TABLE_SCHEMA: test
PRIVILEGE_TYPE: SHOW VIEW
  IS_GRANTABLE: NO


mysql> SELECT * FROM mysql.db\G
*************************** 1. row ***************************
                 Host: %
                   Db: test
                 User:
          Select_priv: Y
          Insert_priv: Y
          Update_priv: Y
          Delete_priv: Y
          Create_priv: Y
            Drop_priv: Y
           Grant_priv: N
      References_priv: Y
           Index_priv: Y
           Alter_priv: Y
Create_tmp_table_priv: Y
     Lock_tables_priv: Y
     Create_view_priv: Y
       Show_view_priv: Y
```

SCHEMATA Table

The standard SQL INFORMATION_SCHEMA.SCHEMATA table shows the schemas that may be accessed by the current user. The MySQL-specific SHOW equivalent is SHOW DATABASES. SCHEMATA has the following columns:

• CATALOG_NAME -- always shows NULL, since MySQL does not support the concept of a database catalog. SHOW DATABASES has no equivalent column.

• SCHEMA_NAME -- shows the name of a schema (i.e., a database) which the current user may use. This column provides the same information as the DATABASE column returned by SHOW DATABASES.

• DEFAULT_CHARACTER_SET_NAME -- shows the name of the database's default character set. SHOW DATABASES has no equivalent column.

• SQL_PATH -- always shows NULL, since the MySQL server does not use this value to find the database files. SHOW DATABASES has no equivalent column.

Here are two, roughly equivalent, commands:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.SCHEMATA\G
*************************** 1. row ***************************
              CATALOG_NAME: NULL
               SCHEMA_NAME: information_schema
DEFAULT_CHARACTER_SET_NAME: utf8
                 SQL_PATH: NULL
*************************** 2. row ***************************
              CATALOG_NAME: NULL
               SCHEMA_NAME: tp
DEFAULT_CHARACTER_SET_NAME: latin1
                 SQL_PATH: NULL


mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| tp                 |
+--------------------+
```

STATISTICS Table

The INFORMATION_SCHEMA.STATISTICS table shows the indexes on tables that may be accessed by the current user. STATISTICS is not a standard SQL table, since the SQL Standard does not deal with any physical database constructs such as indexes; the MySQL-specific SHOW equivalent is SHOW INDEX. STATISTICS has the following columns:

• TABLE_CATALOG -- always shows NULL, since MySQL does not support the concept of a database catalog. SHOW INDEX has no equivalent column.

• TABLE_SCHEMA -- shows the name of a schema (i.e., a database) in which an indexed table available to the current user resides. SHOW INDEX has no equivalent column.

• TABLE_NAME -- shows the name of an indexed table which the current user may access (i.e., on which the current user has been granted a privilege). This column provides the same information as the TABLE column returned by SHOW INDEX.

• NON_UNIQUE -- shows whether the index may contain duplicate values; '0' (zero) if it cannot, '1' (one) if it can. This column provides the same information as the NON_UNIQUE column returned by SHOW INDEX.

• INDEX_SCHEMA -- shows the name of the schema in which the index on this table resides. SHOW INDEX has no equivalent column.

• INDEX_NAME -- shows the name of an index, on this table, which the current user may access. This column provides the same information as the KEY_NAME column returned by SHOW INDEX.

• SEQ_IN_INDEX -- shows the ordinal position of the indexed column within the index key. This column provides the same information as the SEQ_IN_INDEX column returned by SHOW INDEX.

• COLUMN_NAME -- shows the name of a column that comprises some, or all, of the index key. This column provides the same information as the COLUMN_NAME column returned by SHOW INDEX.

• COLLATION -- shows how this column is sorted in the index; either 'A' for ascending or NULL for unsorted columns. This column provides the same information as the COLLATION column returned by SHOW INDEX.

• CARDINALITY -- shows the number of unique values in the index. This column provides the same information as the CARDINALITY column returned by SHOW INDEX.

• SUB_PART -- shows the number of indexed characters if the index is a prefix index; otherwise NULL. This column provides the same information as the SUB_PART column returned by SHOW INDEX.

• PACKED -- shows how the index key is packed; NULL if the key is not packed. This column provides the same information as the PACKED column returned by SHOW INDEX.

• NULLABLE -- shows whether this indexed column may contain NULL values; values are either 'YES', 'NO', or (for versions prior to 5.0.3) a blank (which means 'NO'). This column provides the same information as the NON_UNIQUE column returned by SHOW INDEX.

• INDEX_TYPE -- shows the index type; either 'BTREE', 'FULLTEXT', 'HASH', or 'RTREE'. This column provides the same information as the INDEX_TYPE column returned by SHOW INDEX.

• COMMENT -- shows the comment, if any, stored for this index; otherwise left blank. This column provides the same information as the COMMENT column returned by SHOW INDEX.

Here are two equivalent commands:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.STATISTICS
    -> WHERE TABLE_SCHEMA='tp'\G
*************************** 1. row ***************************
TABLE_CATALOG: NULL
 TABLE_SCHEMA: tp
   TABLE_NAME: t1
   NON_UNIQUE: 0
 INDEX_SCHEMA: tp
   INDEX_NAME: PRIMARY
 SEQ_IN_INDEX: 1
  COLUMN_NAME: col1
    COLLATION: A
  CARDINALITY: 0
     SUB_PART: NULL
       PACKED: NULL
     NULLABLE:
   INDEX_TYPE: BTREE
      COMMENT:


mysql> SHOW INDEX FROM t1\G
*************************** 1. row ***************************
       Table: t1
  Non_unique: 0
    Key_name: PRIMARY
Seq_in_index: 1
 Column_name: col1
   Collation: A
 Cardinality: 0
    Sub_part: NULL
      Packed: NULL
        Null:
  Index_type: BTREE
     Comment:
```

TABLE_CONSTRAINTS Table

The standard SQL INFORMATION_SCHEMA.TABLE_CONSTRAINTS table shows the tables, available to the current user, which are operated on by some constraint and/or index. This table has no direct SHOW equivalent. TABLE_CONSTRAINTS has the following columns:

• CONSTRAINT_CATALOG -- always shows NULL, since MySQL does not support the concept of a database catalog.

• CONSTRAINT_SCHEMA -- shows the name of the schema (i.e., the database) in which an available constraint or index resides.

• CONSTRAINT_NAME -- shows the name of the constraint or index that operates on a table that the current user may access.

• TABLE_SCHEMA -- shows the name of the schema in which the table that is operated on by this constraint or index resides.

• TABLE_NAME -- shows the name of the available table that is operated on by this constraint or index.

• CONSTRAINT_TYPE -- shows the type of the constraint; either 'PRIMARY KEY', 'FOREIGN KEY', or 'UNIQUE'. This column provides basically the same information as the KEY_NAME column returned by SHOW INDEX, when NON_UNIQUE is '0' (zero).

Here is an example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    -> WHERE TABLE_NAME='t1'\G
*************************** 1. row ***************************
CONSTRAINT_CATALOG: NULL
 CONSTRAINT_SCHEMA: tp
   CONSTRAINT_NAME: PRIMARY
      TABLE_SCHEMA: tp
        TABLE_NAME: t1
   CONSTRAINT_TYPE: PRIMARY KEY
```

TABLE_PRIVILEGES Table

The standard SQL INFORMATION_SCHEMA.TABLE_PRIVILEGES table provides information on every table privilege granted for the current database. This table has no direct MySQL SHOW equivalent; instead, the information provided comes from a combination of the results returned by SHOW GRANTS and SHOW FULL COLUMNS. TABLE_PRIVILEGES has the following columns:

• GRANTEE -- shows the name of a user who has been granted a table privilege.

• TABLE_CATALOG -- always shows NULL, since MySQL does not support the concept of a database catalog.

•TABLE_SCHEMA -- shows the name of the schema (i.e., the database) in which the table for which a privilege has been granted resides.

• TABLE_NAME -- shows the name of the table on which the privilege has been granted.

• PRIVILEGE_TYPE -- shows the type of privilege that was granted: either 'SELECT', 'INSERT', 'UPDATE', 'DELETE', 'REFERENCES', 'ALTER', 'INDEX', 'DROP', or 'CREATE VIEW'.

• IS_GRANTABLE -- shows whether the privilege was granted WITH GRANT OPTION; either 'YES' or 'NO'.

Here is an example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TABLE_PRIVILEGES\G
*************************** 1. row ***************************
       GRANTEE: 'TRUDY'@'%'
 TABLE_CATALOG: NULL
  TABLE_SCHEMA: tp
    TABLE_NAME: T1
PRIVILEGE_TYPE: SELECT
  IS_GRANTABLE: NO
*************************** 2. row ***************************
       GRANTEE: 'TRUDY'@'%'
 TABLE_CATALOG: NULL
  TABLE_SCHEMA: tp
    TABLE_NAME: T1
PRIVILEGE_TYPE: INSERT
  IS_GRANTABLE: NO
*************************** 3. row ***************************
       GRANTEE: 'TRUDY'@'%'
 TABLE_CATALOG: NULL
  TABLE_SCHEMA: tp
    TABLE_NAME: T1
PRIVILEGE_TYPE: UPDATE
  IS_GRANTABLE: NO
...
```

TABLES Table

The standard SQL INFORMATION_SCHEMA.TABLES table shows the base tables and views that are available to the current user and provides equivalent information to the MySQL-specific SHOW TABLE STATUS statement. TABLES has the following columns:

• TABLE_CATALOG -- always shows NULL, since MySQL does not support the concept of a database catalog. SHOW TABLE STATUS has no equivalent column.

• TABLE_SCHEMA -- shows the name of the schema (i.e., the database) in which an available table resides. SHOW TABLE STATUS has no equivalent column.

• TABLE_NAME -- shows the name of a table which the current user may access (i.e., on which the current user has been granted a privilege). This column provides the same information as the NAME column returned by SHOW TABLE STATUS.

• TABLE_TYPE -- shows whether this table is a 'BASE TABLE', a 'TEMPORARY' table, or a 'VIEW'. This column provides the same information as the TYPE column returned by SHOW TABLE.

• ENGINE -- shows the storage engine used for this table. This column provides the same information as the ENGINE column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• VERSION -- shows the version number of this table's .frm file. This column provides the same information as the VERSION column returned by SHOW TABLE STATUS and is not part of the

standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• ROW_FORMAT -- shows this table's row storage format; either 'FIXED', 'DYNAMIC', or 'COMPRESSED'. This column provides the same information as the ROW_FORMAT column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• TABLE_ROWS -- shows the number of rows in this table. This column provides the same information as the ROWS column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• AVG_ROW_LENGTH -- shows the average length of this table's rows. This column provides the same information as the AVG_ROW_LENGTH column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• DATA_LENGTH -- shows the length of this table's data file. This column provides the same information as the DATA_LENGTH column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• MAX_DATA_LENGTH -- shows the maximum length of this table's data file. This column provides the same information as the MAX_DATA_LENGTH column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• INDEX_LENGTH -- shows the length of the index file associated with this table. This column provides the same information as the INDEX_LENGTH column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• DATA_FREE -- shows the number of allocated unused bytes for this table. This column provides the same information as the DATA_FREE column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• AUTO_INCREMENT -- shows the next AUTO_INCREMENT value where applicable; otherwise NULL. This column provides the same information as the AUTO_INCREMENT column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• CREATE_TIME -- shows the timestamp of the time this table was created. This column provides the same information as the CREATE_TIME column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• UPDATE_TIME -- shows the timestamp of the time this table's data file was last updated. This column provides the same information as the UPDATE_TIME column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• CHECK_TIME -- shows the timestamp of the time this table was last checked; NULL if the table has never been checked. This column provides the same information as the CHECK_TIME

column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• TABLE_COLLATION -- shows this table's default character set and collation combination. This column provides the same information as the COLLATION column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• CHECKSUM -- shows the live checksum value for the table if any; otherwise NULL. This column provides the same information as the CHECKSUM column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• CREATE_OPTIONS -- shows any additional options used in the table's definition; otherwise left blank. This column provides the same information as the CREATE_OPTIONS column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

• TABLE_COMMENT -- shows the comment, if any, stored for this table; otherwise left blank. This column provides the same information as the COMMENT column returned by SHOW TABLE STATUS and is not part of the standard SQL definition. It was added to provide complete equivalence with SHOW TABLE STATUS.

Here are two equivalent commands:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TABLES
    -> WHERE TABLE_NAME='t1'\G
*************************** 1. row ***************************
  TABLE_CATALOG: NULL
   TABLE_SCHEMA: tp
     TABLE_NAME: t1
     TABLE_TYPE: BASE TABLE
         ENGINE: MyISAM
        VERSION: 9
     ROW_FORMAT: Fixed
     TABLE_ROWS: 1
 AVG_ROW_LENGTH: 15
    DATA_LENGTH: 15
MAX_DATA_LENGTH: 64424509439
   INDEX_LENGTH: 2048
      DATA_FREE: 0
 AUTO_INCREMENT: NULL
    CREATE_TIME: 2005-02-08 15:43:05
    UPDATE_TIME: 2005-02-09 11:35:13
     CHECK_TIME: NULL
TABLE_COLLATION: latin1_swedish_ci
       CHECKSUM: NULL
 CREATE_OPTIONS:
  TABLE_COMMENT:
```

```
mysql> SHOW TABLE STATUS LIKE 't1%'\G
*************************** 1. row ***************************
           Name: t1
         Engine: MyISAM
        Version: 9
     Row_format: Fixed
           Rows: 1
 Avg_row_length: 15
    Data_length: 15
Max_data_length: 64424509439
   Index_length: 2048
      Data_free: 0
 Auto_increment: NULL
    Create_time: 2005-02-08 15:43:05
    Update_time: 2005-02-09 11:35:13
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
```

USER_PRIVILEGES Table

The INFORMATION_SCHEMA.USER_PRIVILEGES table provides information on every user who has privileges on the current database. USER_PRIVILEGES is not a standard SQL table, nor does this table have a SHOW equivalent; instead, it provides equivalent information to the MySQL-specific mysql.user table. USER_PRIVILEGES has the following columns:

• GRANTEE -- shows the user to whom a privilege has been granted. This column provides the same information as the HOST and USER columns of the mysql.user table.

• TABLE_CATALOG -- always shows NULL, since MySQL does not support the concept of a database catalog. The mysql.user table has no equivalent column.

• PRIVILEGE_TYPE -- shows the privilege granted. This column provides the same information as the *_PRIV columns of the mysql.user table.

• IS_GRANTABLE -- shows whether the privilege was granted WITH GRANT OPTION; either 'YES' or 'NO'. The mysql.user table has no equivalent column.

Here are two, roughly equivalent, commands:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.USER_PRIVILEGES\G
*************************** 1. row ***************************
       GRANTEE: 'root'@'localhost'
 TABLE_CATALOG: NULL
PRIVILEGE_TYPE: SELECT
  IS_GRANTABLE: YES
*************************** 2. row ***************************
       GRANTEE: 'root'@'localhost'
 TABLE_CATALOG: NULL
PRIVILEGE_TYPE: INSERT
  IS_GRANTABLE: YES
*************************** 3. row ***************************
       GRANTEE: 'root'@'localhost'
 TABLE_CATALOG: NULL
PRIVILEGE_TYPE: UPDATE
  IS_GRANTABLE: YES
*************************** 4. row ***************************
       GRANTEE: 'root'@'localhost'
 TABLE_CATALOG: NULL
PRIVILEGE_TYPE: DELETE
  IS_GRANTABLE: YES
*************************** 5. row ***************************
       GRANTEE: 'root'@'localhost'
 TABLE_CATALOG: NULL
PRIVILEGE_TYPE: CREATE
  IS_GRANTABLE: YES
*************************** 6. row ***************************
...
```

```
mysql> SELECT * FROM mysql.user\G
*************************** 1. row ***************************
                 Host: localhost
                 User: root
             Password:
          Select_priv: Y
          Insert_priv: Y
          Update_priv: Y
          Delete_priv: Y
          Create_priv: Y
            Drop_priv: Y
          Reload_priv: Y
        Shutdown_priv: Y
         Process_priv: Y
            File_priv: Y
           Grant_priv: Y
      References_priv: Y
           Index_priv: Y
           Alter_priv: Y
         Show_db_priv: Y
           Super_priv: Y
Create_tmp_table_priv: Y
     Lock_tables_priv: Y
         Execute_priv: Y
      Repl_slave_priv: Y
     Repl_client_priv: Y
     Create_view_priv: Y
       Show_view_priv: Y
            ssl_type:
          ssl_cipher:
         x509_issuer:
        x509_subject:
        max_questions: 0
          max_updates: 0
      max_connections: 0
*************************** 2. row ***************************
...
```

VIEWS Table

The standard SQL INFORMATION_SCHEMA.VIEWS table shows the views that are available to the current user. This table has no direct MySQL SHOW equivalent. VIEWS has the following columns:

• TABLE_CATALOG -- always shows NULL, since MySQL does not support the concept of a database catalog.

• TABLE_SCHEMA -- shows the name of the schema (i.e., the database) in which an available view resides.

• TABLE_NAME -- shows the name of a view which the current user may access (i.e., on which the current user has been granted a privilege).

• VIEW_DEFINITION -- shows the SELECT statement that makes up this view's definition. This information is equivalent to that provided by the CREATE VIEW column of SHOW CREATE VIEW.

• CHECK_OPTION -- shows the value of the WITH CHECK OPTION clause used to define the view; either 'NONE', 'LOCAL', or 'CASCADED'.

• IS_UPDATABLE -- shows whether the view is an updatable view; either 'YES' or 'NO'.

Here is an example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.VIEWS
    -> WHERE TABLE_NAME='v1'\G
*************************** 1. row ***************************
  TABLE_CATALOG: NULL
   TABLE_SCHEMA: tp
     TABLE_NAME: v1
VIEW_DEFINITION: select `tp`.`t1`.`col1` AS `col1`,`tp`.`t1`.`col2`
                 AS `col2` from `tp`.`t1`
   CHECK_OPTION: NONE
   IS_UPDATABLE: YES
```

Bugs and Feature Requests

MySQL 5.0 was not a production release version at the time this book was written, so some aspects of the INFORMATION_SCHEMA implementation were incomplete. This section provides some information on bugs and features outstanding when we went to press.

Bugs

To get current details on outstanding INFORMATION_SCHEMA bugs, go to the following web page and search for "information_schema" or "information":
http://bugs.mysql.com

Feature Requests

There were no feature requests outstanding at the time of writing.

Resources

For my last act I'll show you where you can go for further information.

Source Code

```
sql/sql_db.cc
sql/sql_parse.cc
sql/sql_show.cc
```

```
sql/mysql_priv.h
sql/table.h
mysql-test/t/information_schema.test
mysql-test/t/information_schema_inno.test
```

As always with MySQL, you can download our source code and see how Sergey did it. The main programs for INFORMATION_SCHEMA are sql_db.cc and sql_show.cc. Other interesting files are information_schema_inno.test and information_schema.test, which will give you an idea of how thorough our test suite is.

Conclusion

You've come to the end of the book. I don't bother with a review or an index, since I'm sure you've had no trouble memorizing it all.

If you enjoyed this book, you should look for others in the "MySQL 5.0 New Features" series. The previous books talk about "Stored Procedures", "Triggers", and "Views".

Thank you very much for your attention. If you have any comments about this book, please send them to one of the MySQL forums at:
http://forums.mysql.com

About MySQL

MySQL AB develops and supports a family of high performance, affordable database servers and tools. The company's flagship product is MySQL, the world's most popular open source database, with more than six million active installations. Many of the world's largest organizations, including Yahoo!, Sabre Holdings, The Associated Press, Suzuki and NASA are realizing significant cost savings by using MySQL to power high-volume Web sites, business-critical enterprise applications and packaged software.

With headquarters in Sweden and the United States – and operations around the world – MySQL AB supports both open source values and corporate customers' needs in a profitable, sustainable business. For more information about MySQL, please visit www.mysql.com.