Introduction to Databases

PHP & MySQL Database Access

Key Concepts

- Creating databases
- Creating users
- Privileges
- Tables
- Database queries
- Data manipulation

MySQL Databases

- Create a database
  `create database` *`dbname`*`;`
- Set the current database
  `use` *`dbname`*`;`
- List the databases
  `show databases;`
- List the tables
  `show tables;`
- Show information about a table
  `describe` *`tablename`*`;`

Privileges for Users

Privilege	Applies To	Description
SELECT	tables, columns	Allows users to select rows (records) from tables.
INSERT	tables, columns	Allows users to insert new rows into tables.
UPDATE	tables, columns	Allows users to modify values in existing table rows.
DELETE	tables	Allows users to delete existing table rows.
INDEX	tables	Allows users to create and drop indexes on particular tables.
ALTER	tables	Allows users to alter the structure of existing tables by, for example, adding columns, renaming columns or tables, and changing data types of columns.
CREATE	databases, tables	Allows users to create new databases or tables. If a particular database or table is specified in GRANT, they can only create that database or table, which means they will have to drop it first.
DROP	databases, tables	Allows users to drop (delete) databases or tables.

Privileges for Administrators

Privilege	Description
CREATE TEMPORARY TABLES	Allows an administrator to use the keyword TEMPORARY in a CREATE TABLE statement.
FILE	Allows data to be read into tables from files and vice versa.
LOCK TABLES	Allows the explicit use of a LOCK TABLES statement.
PROCESS	Allows an administrator to view server processes belonging to all users.
RELOAD	Allows an administrator to reload grant tables and flush privileges, hosts, logs, and tables.
REPLICATION CLIENT	Allows use of SHOW STATUS on replication masters and slaves. Replication is explained in Chapter 12.
REPLICATION SLAVE	Allows replication slave servers to connect to the master server. Replication is explained in Chapter 12.
SHOW DATABASES	Allows a list of all databases to be seen with a SHOW DATABASES statement. Without this privilege, users see only databases on which they have other privileges.
SHUTDOWN	Allows an administrator to shut down the MySQL server.
SUPER	Allows an administrator to kill threads belonging to any user.

Special Privileges

Privilege	Description
`ALL`	Grants all the privileges listed in Tables 11.1 and 11.2. You can also write `ALL PRIVILEGES` instead of `ALL`.
`USAGE`	Grants no privileges. This privilege creates a user and allows her to log on, but it doesn't allow her to do anything. Usually, you will add more privileges later.

GRANT and REVOKE Syntax

- GRANT

```
GRANT privileges [columns]
ON item
TO user_name
[IDENTIFIED BY 'password']
[REQUIRE ssl_options]
[WITH [GRANT OPTION ! limit_options]]
```

- REVOKE

```
REVOKE privileges [(columns)]
ON item
FROM username
```

Examples of Using GRANT

- Set up an administrator

```
grant all on *
to fred identified by 'mnb123';
```

- Set up a user with no privileges

```
grant usage on books.*
to sally identified by 'magic123';
```

- Grant Sally privileges on the books database

```
grant select, insert, update, delete
on books.*
to sally;
```

Creating Tables

- Use MySQL Administrator
- Use the create table command

```
create table customers
(customerid int unsigned not null
  auto_increment primary key,
  name char(50) not null,
  address char(100) not null,
  city char(30) not null);
```

Integral Data Types

Type	Range	Storage (Bytes)	Description
`TINYINT[(M)]`	–127..128 or 0..255	1	Very small integers
`BIT`			Synonym for `TINYINT`
`BOOL`			Synonym for `TINYINT`
`SMALLINT[(M)]`	–32768..32767 or 0..65535	2	Small integers
`MEDIUMINT[(M)]`	–8388608.. 8388607 or 0..16777215	3	Medium-sized integers
`INT[(M)]`	$-2^{31}..2^{31}-1$ or $0..2^{32}-1$	4	Regular integers
`INTEGER[(M)]`			Synonym for `INT`
`BIGINT[(M)]`	$-2^{63}..2^{63}-1$ or $0..2^{64}-1$	8	Big integers

Floating Point Data Types

Type	Range	Storage (bytes)	Description
FLOAT(*precision*)	Depends on precision	Varies	Can be used to specify single or double precision floating-point numbers.
FLOAT[(M,D)]	±1.175494351E-38 ±3.402823466E+38	4	Single precision floating-point number. These numbers are equivalent to FLOAT(4) but with a specified display width and number of decimal places.
DOUBLE[(M,D)]	±1. 7976931348623157E +308 ±2.2250738585072014E -308	8	Double precision floating-point number. These numbers are equivalent to FLOAT(8) but with a specified display width and number of decimal places.
DOUBLE PRECISION[(M,D)]	As above		Synonym for DOUBLE[(M, D)].
REAL[(M,D)]	As above		Synonym for DOUBLE[(M, D)].
DECIMAL[(M[,D])]	Varies	M+2	Floating-point number stored as char. The range depends on M, the display width.
NUMERIC[(M,D)]	As above		Synonym for DECIMAL.
DEC[(M,D)]	As above		Synonym for DECIMAL.
FIXED[(M,D)]	As above		Synonym for DECIMAL.

Date and Time Data Types

Type	Range	Description
DATE	1000-01-01 9999-12-31	A date. Will be displayed as YYYY-MM-DD.
TIME	-838:59:59 838:59:59	A time. Will be displayed as HH:MM:SS. Note that the range is much wider than you will probably ever want to use.
DATETIME	1000-01-01 00:00:00 9999-12-31 23:59:59	A date and time. Will be displayed as YYYY-MM-DD HH:MM:SS.
TIMESTAMP[(*M*)]	1970-01-01 00:00:00 Sometime in 2037 timestamps.	A timestamp, useful for transaction reporting. The display format depends on the value of *M* (see Table 11.8, which follows). The top of the range depends on the limit on Unix.
YEAR[(*2*|4)]	70–69 (1970–2069) 1901–2155	A year. You can specify two- or four-digit format. Each has a different range, as shown.

Timestamp Data Types

Type Specified	Display
TIMESTAMP	YYYYMMDDHHMMSS
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

Regular String Data Types

Type	Range	Description
[NATIONAL] CHAR(*M*) [BINARY | ASCII | UNICODE]	0 to 255 characters	Fixed-length string of length *M*, where *M* is between 0 and 255. The NATIONAL keyword specifies that the default character set should be used. This is the default in MySQL anyway, but is included because it is part of the ANSI SQL standard. The BINARY keyword specifies that the data should be treated as case sensitive. (The default is case sensitive.) The ASCII keyword specifies that the latin1 character set will be used for this column. The UNICODE keyword specifies that the ucs character set will be used.
CHAR		Synonym for CHAR(1).
[NATIONAL] VARCHAR(*M*) [BINARY]	1 to 255 characters	Same as above, except they are variable length.

TEXT and BLOB Data Types

Type	Maximum Length (Characters)	Description
TINYBLOB	2^8 –1 (that is, 255)	A tiny binary large object (BLOB) field
TINYTEXT	2^8 –1 (that is, 255)	A tiny TEXT field
BLOB	2^{16} –1 (that is, 65,535)	A normal-sized BLOB field
TEXT	2^{16} –1 (that is, 65,535)	A normal-sized TEXT field
MEDIUMBLOB	2^{24} –1 (that is, 16,777,215)	A medium-sized BLOB field
MEDIUMTEXT	2^{24} –1 (that is, 16,777,215)	A medium-sized TEXT field
LONGBLOB	2^{32} –1 (that is, 4,294,967,295)	A long BLOB field
LONGTEXT	2^{32} –1 (that is, 4,294,967,295)	A long TEXT field

Querying a Database from PHP

1. Check and filter data sent by the user
2. Set up a connection to the database
3. Query the database
4. Retrieve the results
5. Present the results to the user

Setting Up the Connection

1. Connect to the server

```
$db = mysql_connect('localhost',
   'username', 'password');
```

2. Check for an error

```
if (mysql_errno($db)) {
    echo mysql_error($db);
}
```

3. Select the database if necessary

```
mysql_select_db('dbname',$db);
```

Query the Database

1. Store the query in a variable

```
$query = "select * from books";
```

2. Issue the query

```
$result = mysql_query($query,$db);
```

Processing Results

- Get the number of rows

```
$num_results = mysql_num_rows($result);
```

- Return the row as a key-value array

```
$row = mysql_fetch_array($result);
```

- Get the field information for a row

```
$myFields = mysql_fetch_field($row);
```

Processing Results (cont'd)

- Return the row as an enumerated array

```
$row = mysql_fetch_row($result);
```

- Return the row as an object

```
$row = mysql_fetch_object($result);
```

Closing the Connection

- Free the result set

```
mysql_free_result($result);
```

- Close the connection

```
mysql_close($db);
```

Inserting, Updating, and Deleting Data

- Store the query in a variable

```
$query = "insert into books values
  ('".$isbn."', '".$author."',
  '".$title."')";
```

- Issue the query

```
mysql_query($query,$db);
```

- Check the number of rows affected

```
echo mysql_affected_rows($query);
```

Sql Injection

- <?php
  For example:
  $_POST['username'] = 'aidan';
  $_POST['password'] = "' OR '‘1=1'";
  // Query database to check if there are any matching users
  $query = "SELECT * FROM users WHERE user='{$_POST['usern
  ame']}' AND password='{$_POST['password']}'";
  mysql_query($query);
  echo $query;
  ?>
- SELECT * FROM users WHERE user='aidan' AND password='' OR '‘1=1''