

پوهنتون کابل

پوهنځی کمپیوتر ساینس

دپارتمنت سیستم های معلوماتی

Structured Query Language (SQL) Fundamentals

Lecture 09-11

تهیه کننده : پوهنیار محمد شعیب "زرین خیل"
سال : 1389

Structured Query Language (SQL) 09

By: M Shuaib Zarinkhail

2010



Retrieving Information – Database Objects (Show)

- ▶ **SHOW DATABASES**
 - Already Explained
 - When this command is run, you can select one database with **USE DATABASE**
 - e.g. use dbOne
- ▶ **DATABASE():** Enables you to check which database is active, you can use the **DATABASE()** function as
 - i.e. `Select database();`

Retrieving Information – Database Objects (Show)

- ▶ **SHOW CREATE DATABASE dbName**
 - Shows the CREATE DATABASE statements that created the given DB
 - SHOW CREATE SCHEMA is a synonym for this command

→ e.g. `show create database dbOne;`

Retrieving Information – Database Objects (Show)

- ▶ **SHOW TABLES [FROM dbName]**
 - Lists the database tables
 - The output for this command was only table names (Before MySQL 5.0.1)
 - This statement also lists any views in the DB (Beginning with MySQL 5.0.1)
 - If the **FULL** keyword is added to this command, table types are also shown e.g. → show full tables from dbOne;

Retrieving Information – Database Objects (Show)

- ▶ **SHOW CREATE TABLE TableName**
 - Shows the CREATE TABLE statement that already created for the given table
 - This command also works for views (MySQL 5.0.1 and later)

→ e.g. `show create table tFour;`

→ e.g. `show create view vOne;`

Retrieving Information - Database Objects (Show)

```
SHOW INDEX FROM tbl_name [FROM  
db_name]
```

- ▶ Lists all indexes from a specified table in a DB
- ▶ e.g. show index from tOne from dbOne;

Retrieving Information – Database Objects (Show)

- ▶ **SHOW TABLE STATUS [FROM dbName]**
 - This command provides more information about each table including
 - Storage engine
 - Rows and average row lengths
 - Indexes
 - Create, update, and check times
 - And so on
- e.g. show table status from dbTwo;

Retrieving Information – Database Objects (Show)

▶ SHOW ENGINES

- Already explained
- This command shows the storage engines of your software
- You can optionally add the STORAGE keyword to this command too
- e.g. `show storage engines;`

Retrieving Information – Database Objects (Show)

- ▶ **SHOW COLUMNS FROM TableName [FROM dbName]**
 - Describes the structure of a table
 - Synonym commands are **DESCRIBE** and **DESC**
 - Additionally, you can add the **FULL** keyword to show more details
 - e.g. show columns from tThree;
 - Equals to
 - e.g. describe tThree;

Retrieving Information – Database Objects (Show)

▶ SHOW PRIVILEGES

- Shows the list of system privileges that the MySQL Server supports
- The exact list of privileges depends on the version of the server software

→ e.g. `show privileges;`

Retrieving Information – Database Objects (Show)

▶ SHOW WARNINGS

- Shows the error, warning, and note messages that resulted from the last statement that generated messages
- The SHOW COUNT(*) WARNINGS functions displays the total number of errors, warnings, and notes from the recently generated messages

→ e.g. show warnings;

Retrieving Information – Database Objects (Show)

▶ SHOW ERRORS

- Similar to the SHOW WARNINGS command, but only lists errors
- The list of messages is reset for each new statement that uses a table
- The SHOW COUNT(*) ERRORS function displays the total number of errors, warnings, and notes

→ e.g. show errors;

Structured Query Language (SQL) 10

By: M Shuaib Zarinkhail

2010



Some Points for SQL DML

- ▶ Data can be queried from one or more tables
- ▶ This action is implemented using SELECT statement with the following syntax
→SELECT ColumnNames
 FROM TableNames
 WHERE Conditions
e.g. →select department, maxhours
 from project where project.name = 'fields';

Some Points for SQL DML

- ▶ **SELECT:**
 - Is used to retrieve rows selected from one or more tables
 - Can include UNION statements
 - Can have sub-queries
- ▶ **SELECT can also be used to retrieve rows computed without reference to any table**
e.g. → `select 7*24 as Week_Hours, 7*24*60 as Week_Minutes, 7*24*60*60 as Week_Seconds;`

Some Points for SQL DML

- ▶ The most commonly used clauses of SELECT are:
 - Each SELECT expression indicates columns that you want to retrieve
 - There must be at least one column in a command
 - Table references indicates the table(s) from which to retrieve rows
 - The WHERE clause, if given, indicates condition(s)
 - In the WHERE clause, any of the functions and operators supported by MySQL can be used

SQL DDL (Views) Definition

- ▶ Views are used to:
 - display read-only data (view of data) from one or more tables in a DB
 - insert data to base-tables in a DB
- ▶ Views can be created in a DB
- ▶ TEMPORARY Views can not be created

SQL DDL (Views) Definition

- ▶ The syntax for creating a view is:
→ CREATE VIEW ViewName AS SELECT ...

e.g. → Create view vOne AS Select ID,
Name From tOne;

SQL DDL (Views) Definition

You can replace an existing view to a new one with different definitions

→ CREATE OR REPLACE VIEW ViewName
AS SELECT ...

e.g. → Create or Replace view vOne AS
Select Name From tOne;

SQL DDL (Views) Definition

- ▶ Base-tables and views share the same name_space within a specific DB
 - Therefore their names should be unique from each other
- ▶ Views must have unique column names with no duplicates
 - Similar to database tables (Relations)

SQL DDL (Views) Definition

- ▶ A view can refer to tables or views in other databases too
 - This is done by qualifying the table or view name with the proper database name

Example (Next Slide)

SQL DDL (Views) Definition

- ▶ Creating View Example

→ CREATE VIEW VIEW_NAME AS SELECT
COLUMN_LIST FROM
DB_NAME.TABLE_NAME ...

e.g. → Create View vTwo AS Select Name,
Address From dayOne.tTwo;

SQL DDL (Views) Definition

- ▶ A view can be created from many kinds of SELECT statements
- ▶ It can refer to base-tables or other views
- ▶ It can use joins, UNIONS, and subqueries
Example (Next Slide)

SQL DDL (Views) Definition

▶ Practical Example for View

- CREATE TABLE tThree (qty INT, price INT);
- INSERT INTO tThree VALUES(3, 50);
- CREATE VIEW vThree AS SELECT qty, price, qty*price AS Value FROM tThree;
- SELECT * FROM vThree;

SQL DDL (Views) Definition

Restrictions of a view definition

- ▶ The SELECT statement in a view cannot:
 - contain a subquery in the FROM clause
 - refer to system variables
 - refer to user variables
 - refer to prepared statement parameters
- ▶ Any table or view referred to in the definition must exist
- ▶ The definition cannot refer to a TEMPORARY table

SQL DDL (Views) Definition

- ▶ ORDER BY is allowed in a VIEW definition
 - But only applicable if the SELECT statements do not have their own ORDER BY command

SQL DDL (Views) Definition

▶ ALTER VIEW

- Changes the definition of a VIEW
 - The VIEW must exist!
- The syntax is similar to that for CREATE VIEW
- Requires the CREATE VIEW and DROP privileges for the view and for the SELECT statement
- Example Next Slide

SQL DDL (Views) Definition

- ▶ ALTER VIEW Example

→ ALTER VIEW ViewName AS SELECT ...

e.g. → alter view vOne as select colFive,
colFour from tTwo where colThree IS
NULL;

SQL DDL (Views) Update

- ▶ Some views can update data in base tables
 - you can use them in statements such as
 - UPDATE, DELETE, or INSERT
- ▶ It updates the contents of the underlying table

SQL DDL (Views) Update

- ▶ For a view to be updatable, there must be a one-to-one relationship to its underlying table
 - By default it has the relationship
- ▶ There are certain other constructs that make a view non-updatable (Next Slide)

SQL DDL (Views) Update

- ▶ A view is not updatable if it contains any of the following:
 - Aggregate functions
 - SUM(), MIN(), MAX(), COUNT(), and so forth
 - DISTINCT – GROUP BY
 - HAVING – UNION or UNION ALL

Continues to the Next Slide

SQL DDL (Views) Update

- ▶ A view is not updatable if it contains any of the following:
 - Subquery in the select list
 - Certain joins
 - Non-updatable view in the FROM clause
 - A subquery in the WHERE clause that refers to a table in the FROM clause

SQL DDL (Views) Insert

- ▶ With respect to insertability
 - Views being updatable with INSERT statements
- ▶ It is sometimes possible for a multiple-table view to be updatable
- ▶ For a multiple-table updatable view
 - INSERT can work if it inserts into a single table

SQL DDL (Views) Insert

- ▶ A view is insertable if it satisfies these requirements for the view columns:
 - There must be no duplicate view column names
 - The view must contain all indexed columns in the base-table
 - The view columns must be simple column references and not derived columns
 - A derived column is one that is not a simple column reference but is derived from an expression

Structured Query Language (SQL) 1 1

By: M Shuaib Zarinkhail

2010



Backup MySQL Database – mysqldump

- ▶ You can use this command to backup your DBs from MySQL DBMS
- ▶ This command creates a MySQL text file including all codes for creating databases, their objects, relationships, constraints, rules, ...
- ▶ The mysqldump command runs from command line

Backup MySQL Database – mysqldump

- ▶ This command runs from “bin” folder as:

```
C:\Program Files\MySQL\MySQL Server  
5.0\bin\mysqldump --all-databases -u  
root > pathfile.sql
```

- ▶ In case of password protection:

```
...\bin\mysqldump --all-databases -p -u  
root > D:\backup_file_name.sql
```

Backup MySQL Database – mysqldump – Example

```
... \bin\mysqldump --all-databases -u  
root > e:\dbpractice\backup.sql
```

- ▶ Or in case of password protection type:

```
... \bin\mysqldump --all-databases -p  
-u root > e:\dbpractice\backup.sql
```

Note: The extension of the backup file should be sql and “--all-databases” do not have space between characters

Backup MySQL Database – mysqldump

- ▶ The mentioned command backups all existing databases from the DBMS
- ▶ If you want to backup one or more (not all) database(s) from the DBMS type the following command:

```
... \bin\mysqldump --databases  
db1Name, db2Name -p -u root >  
pathfilename.sql
```


Restore MySQL Database

- ▶ To restore MySQL database using sql backup file, type:

```
...bin\mysql -u root < pathfile.sql
```

- In case of password protection add -p

- ▶ Example

```
...bin\mysql -p -u root < e:\dbprac\  
backup.sql
```

Note: This code adds all databases and their objects from backup file to the DBMS

SQL-DML (Insert Data)

- ▶ **INSERT**
 - Inserts new rows to a table in a DB
 - Updates data in a table
- ▶ Three methods to use the INSERT command in SQL
- 3. **INSERT ... VALUES**
- 4. **INSERT ... SET**
- 5. **INSERT ... SELECT**

INSERT ... VALUES

- ▶ Inserts new and pure data to a table
- ▶ The following options can be used with this command
 - HIGH PRIORITY – LOW PRIORITY
 - DELAYED – IGNORE
- ▶ → INSERT INTO TableName (Columns) VALUES (Values)
- ▶ e.g. → insert into tOne (id, name) values (5, 'Riaz');

INSERT ... SET

- ▶ Inserts new data and sets existing data in a table
- ▶ The following options can be used with this command
 - HIGH PRIORITY – LOW PRIORITY
 - DELAYED – IGNORE
- ▶ → INSERT INTO TableName SET ColOne = Value, ColTwo = Value ...
- ▶ e.g. → insert into tOne set id = 18, name = 'Ahmad';

INSERT ... SELECT

- ▶ Inserts many rows to one table with a single command
- ▶ The following options can be used with this command
 - HIGH PRIORITY – LOW PRIORITY
 - DELAYED – IGNORE
- ▶ →INSERT INTO TableName (Columns)
 SELECT Columns
 FROM AnotherTableName;
- ▶ e.g. →insert tOne (id, name) select id, name from tTwo;

SQL-DML (Insert Data)

The columns for which the statement provides values can be specified as follows:

- ▶ You can provide a comma-separated list of column names following the table name
 - In this case, a value for each named column must be provided by the VALUES list or the SELECT statement

SQL-DML (Insert Data)

- ▶ If you do not specify a list of column names for `INSERT ... VALUES` or `INSERT ... SELECT`, values for every column in the table must be provided by the `VALUES` list or the `SELECT` statement
- If you do not know the order of the columns in the table, use `DESCRIBE` to find out

Column Values (Several Ways)

- ▶ The SET clause indicates the column names explicitly
- ▶ Normally, any column not explicitly given a value is set to its default (explicit or implicit) value
 - For example, if you specify a column list that does not name all the columns in the table, unnamed columns are set to their default values

Column Values – DEFAULT

- ▶ You can use the keyword `DEFAULT` to explicitly set a column to its default value (New in MySQL 4.0.3.)
 - This makes it easier to write `INSERT` statements that assign values to all but a few columns
 - It enables you to avoid writing an incomplete `VALUES` list that does not include a value for each column in the table

Column Values – DEFAULT(col_name)

- ▶ You can use DEFAULT(col_name) as a more general form that can be used in expressions to produce a given column's default value
 - e.g. `Select default(colOne) from tOne;`
- ▶ You can also update a column's default values (MySQL 5.0.2)
 - e.g. `Update tOne set colOne = default(colOne) * 1.05 where colTwo > 100;`

Column Values (Several Ways)

- ▶ If both the column list and the VALUES list are empty, INSERT creates a row with each column set to its default value:
 - e.g. `INSERT INTO tbl_name () VALUES();`
- ▶ You can specify an expression `expr` to provide a column value
 - An expression `expr` can refer to any column that was set earlier in a value list
 - Example and conditions NEXT SLIDE

Column Values (Several Ways)

- ▶ You can assign the value of col2 to col1
- ▶ You can do this because the value for col2 refers to col1, which has previously been assigned:
 - `INSERT INTO tbl_name (col1,col2) VALUES(15,col1*2);`
- ▶ But the following is not legal, because the value for col1 refers to col2, which is assigned after col1:
 - `INSERT INTO tbl_name (col1,col2) VALUES(col2*2,15);`

Column Values (Several Ways)

- ▶ INSERT statements that use VALUES syntax can insert multiple rows at a time
- ▶ To do this, include multiple lists of column values, each enclosed within parentheses and separated by commas
 - e.g. `INSERT INTO tbl_name (a,b,c) VALUES(1,2,3), (4,5,6), (7,8,9);`