پوهنتون کابل

پوهنځی کمپیوترساینس

دیپارتمنت سیستم های معلوماتی

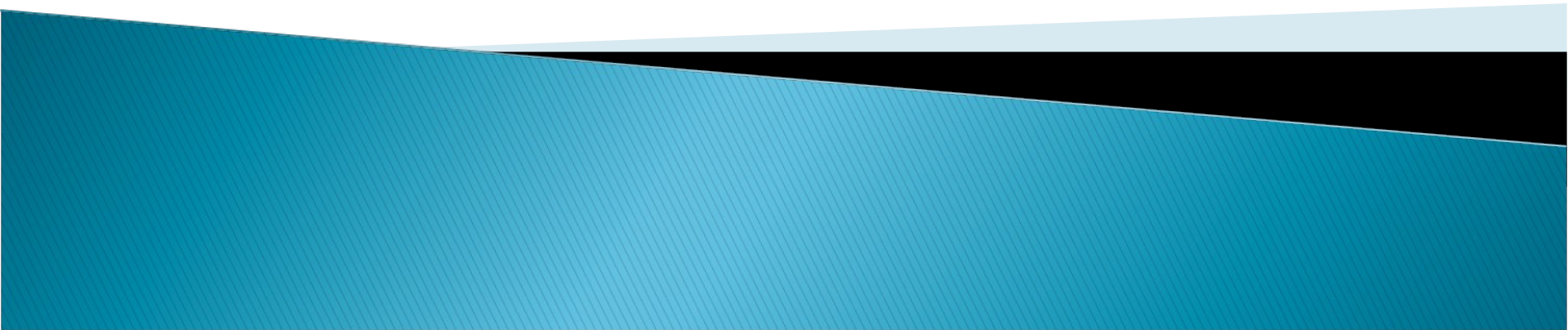
Structured Query Language (SQL) Fundamentals

Lectures 12-13

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

Structured Query Language (SQL) 12

By: M Shuaib Zarinkhail 2010



Warnings – Insert Data

- Warnings indicates the number of attempts to insert column values that were problematic in some way
- Warnings can occur under the following condition and similar cases:
  - Inserting NULL into a column that has been declared NOT NULL

Warning Conditions

- For multiple-row INSERT statements or INSERT INTO ... SELECT statements, the column is set to the implicit default value for the column data type
  - This is 0 for numeric types, the empty string ('') for string types, and the “zero” value for date and time types
- INSERT INTO ... SELECT statements are handled the same way as multiple-row inserts

Warning Conditions

- For a single-row INSERT, no warning occurs when NULL is inserted into a NOT NULL column. Instead, the statement fails with an error
- Setting a numeric column to a value that lies outside the column's range
  - The value is clipped to the closest endpoint of the range

Warning Conditions

- Assigning a value such as '10.34 a' to a numeric column
  - The trailing nonnumeric text is stripped off and the remaining numeric part is inserted
  - If the string value has no leading numeric part, the column is set to 0

Warning Conditions

- Inserting a string into a string column (CHAR, VARCHAR, TEXT, or BLOB) that exceeds the column's maximum length
  - The value is truncated to the column's maximum length
- Inserting a value into a date or time column that is illegal for the data type
  - The column is set to the appropriate zero value for the type

SQL-DML (Insert Data)

- If INSERT inserts a row into a table that has an AUTO_INCREMENT column, you can find the value used for that column by using the SQL LAST_INSERT_ID() function
  - e.g. select last_insert_id();
- If you use the IGNORE keyword, errors that occur while executing the INSERT statement are treated as warnings

SQL-DML (Replace Data)

- You can use REPLACE instead of INSERT to overwrite old rows
- REPLACE is the counterpart to INSERT IGNORE in the treatment of new rows that contain unique key values
- The new rows are used to replace the old rows rather than being discarded

SQL-DML (Replace Data)

- REPLACE works exactly like INSERT, except that if an old row in the table has the same value as a new row for a PRIMARY KEY or a UNIQUE index
- Replace either inserts, or deletes and inserts (updates) data
- Note that unless the table has a PRIMARY KEY or UNIQUE index, using a REPLACE statement makes no sense (equivalent to INSERT)

SQL-DML (Replace Data)

- Just as happens for INSERT
  - Values for all columns are taken from the values specified in the REPLACE statement
  - Any missing columns are set to their default values
- To use REPLACE, you must have both the INSERT and DELETE privileges for the table

SQL-DML (Replace Data)

- The REPLACE statement returns a count to indicate the number of rows affected
  - This is the sum of the rows deleted and inserted
- If the count is 1 for a single-row REPLACE, a row was inserted and no rows were deleted
- If the count is greater than 1, one or more old rows were deleted before the new row was inserted

INSERT … ON DUPLICATE KEY UPDATE

- INSERT … ON DUPLICATE KEY UPDATE is added in MySQL 4.1.0
- If you specify ON DUPLICATE KEY UPDATE, and a row is inserted that would cause a duplicate value in a UNIQUE index or PRIMARY KEY an UPDATE of the old row is performed
- *Example NEXT SLIDE*

INSERT … ON DUPLICATE KEY UPDATE

- For example, if column ‘a’ is declared as UNIQUE and contains the value 1, the following two statements have identical effect:
  - INSERT INTO tOne (a,b,c) VALUES (1,2,3) ON DUPLICATE KEY UPDATE c=c+1;
  - UPDATE tOne SET c=c+1 WHERE a=1;

INSERT … ON DUPLICATE KEY UPDATE

- If column ‘b’ is also unique, the INSERT is equivalent to this UPDATE statement:

  UPDATE tOne SET c=c+1 WHERE a=1 OR b=2 LIMIT 1;

  - The LIMIT keyword limits the number of affected rows by a command
  - The LIMIT keyword is combined with the SELECT, UPDATE, DELETE and some other SQL commands

INSERT … ON DUPLICATE KEY UPDATE

- In general, try to avoid using an ON DUPLICATE KEY UPDATE clause on tables with multiple unique indexes
- The ON DUPLICATE KEY UPDATE clause can contain multiple column assignments, separated by commas
- INSERT INTO tOne (a,b,c) VALUES (1,2,3) ON DUPLICATE KEY UPDATE b=b*1.05, c=c+1;

INSERT … ON DUPLICATE KEY UPDATE

- As of MySQL 4.1.1, you can use the VALUES(col_name) function in the UPDATE clause
- By using this function you can refer to column values from the INSERT portion of the ON DUPLICATE KEY UPDATE statement
- VALUES(col_name) in the ON DUPLICATE KEY UPDATE clause refers to the value of col_name that would be inserted
- This function is especially useful in multiple-row inserts

INSERT … ON DUPLICATE KEY UPDATE

- The VALUES() function is meaningful only in INSERT … UPDATE statements and returns NULL otherwise
  - e.g. → INSERT INTO table1 (a,b,c) VALUES (1,2,3), (4,5,6) ON DUPLICATE KEY UPDATE c=VALUES(a) +VALUES(b);
- That statement is identical to the following two statements:
  - INSERT INTO table1 (a,b,c) VALUES (1,2,3) ON DUPLICATE KEY UPDATE c=3;
  - INSERT INTO table1 (a,b,c) VALUES (4,5,6) ON DUPLICATE KEY UPDATE c=9;

SQL-DML (Insert Data)

- Data can not be entered to multiple tables by one command
- Data can be entered from multiple tables by one command
  - As we can retrieve data from multiple tables by SELECT statement
  - e.g. →INSERT INTO tOne SELECT * FROM tTwo JOIN tThree;

SQL-DML (Insert Data)

To reduce insertion problems in a DB:

- Avoid Surrogate Keys
- Avoid Null Values
- Use Default Values
- Use logical sequences for field names
- Avoid Constraints
  - If used, explain them completely
- Run DESCRIBE command before insertion

Practical Points for Data Insertion

- Use field names in every INSERT command
- Type values for every field (including default fields)
- Use TRANSACTIONs for all INSERT commands (explains later)

Sample data for PROJECT table

Project ID	Name	Department	MaxHours
1000	03 Portfolio Analysis	Finance	75.0
1200	03 Tax Prep	Accounting	145.0
1400	04 Product Plan	Marketing	138.0
1500	04 Portfolio Analysis	Finance	110.0

Sample data for EMPLOYEE table

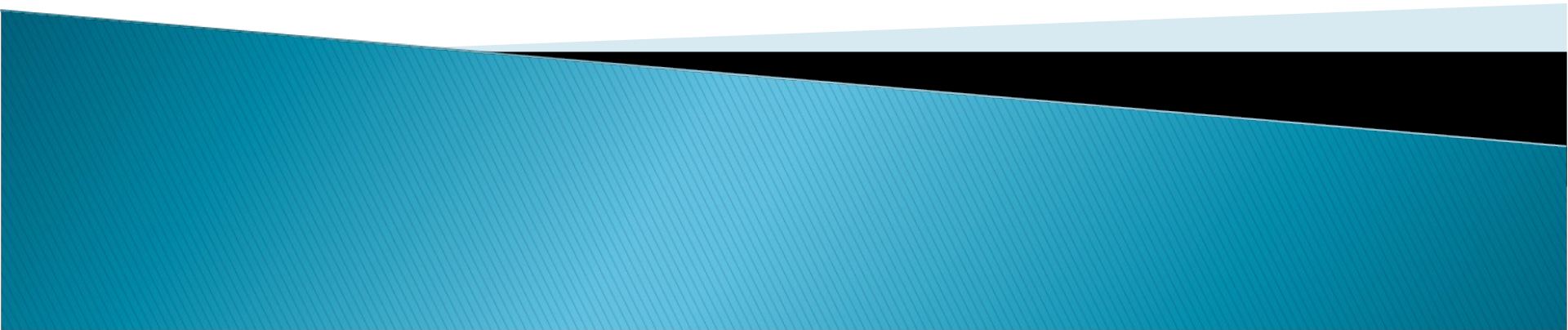
Employee Number	Name	Phone	Department
100	Mary Jacobs	285-8879	Accounting
200	Keni Numoto	287-0098	Marketing
300	Heather Jones	287-9981	Finance
400	Rosalie Jackson	285-1273	Accounting
500	James Nestor	287-0123	Info Systems
600	Richard Wu	287-3222	Info Systems
700	Kim Sung		Marketing

Sample data for ASSIGNMENT table

ProjectID	EmployeeNum	HoursWorked
1000	100	17.50
1000	300	12.50
1000	400	8.00
1000	500	20.25
1200	100	45.75
1200	400	70.50
1200	600	40.50
1400	200	75.00
1400	700	20.25
1400	500	25.25

Structured Query Language (SQL) 13

By: M Shuaib Zarinkhail 2010



SQL DML (Update Data)

- One of the important features in database field is updating data and keeping databases updated
- This is possible by UPDATE command
- To update data MySQL user needs both the DELETE and INSERT privileges

SQL DML (Update Data)

- To update data in a DB:

Type → UPDATE TABLE

SET Columns=data_values

WHERE criteria

e.g. →update project set MaxHours = 76.0

where Department = Finance;

- Update can be implemented on a single table and/or on multiple tables at a time

UPDATE Syntax – One Table

Single-table syntax:

- UPDATE table_reference
  SET col1 = expr1, col2 = expr2
  WHERE where_conditions
  ORDER BY ... LIMIT row_count

e.g. update tone set sal=sal*1.05 where department='Accounting' order by ...;

UPDATE Syntax – Many Tables

Multiple-table syntax:

- UPDATE table-references
  SET tbl1.col1 = expr1, tbl2.col1 = expr1 WHERE where_conditions
  e.g. update tone, tthree set tone.hours=default and tthree.sal=4000 where tthree.address='Kabul';

UPDATE Syntax – Single Table

- For the single-table syntax, the UPDATE statement updates columns of existing rows in the named table with new values
- The SET clause indicates which columns to modify and the values they should be given
- Each value can be given as an expression or the keyword DEFAULT to set a column explicitly to its default value

UPDATE Syntax – Single Table

- The WHERE clause specifies the conditions that identify which rows to update
- If the ORDER BY clause is specified, the rows are updated in the order that is specified
- The LIMIT clause places a limitation on the number of rows that can be updated

UPDATE Syntax – Multiple Table

- In this form, UPDATE, updates rows in each table named in table_references that satisfy the conditions
- In Multiple-Table update, ORDER BY and LIMIT cannot be used
  - where_condition is an expression that evaluates to true for each row to be updated

The UPDATE statement supports the following modifiers

1. If you use the LOW_PRIORITY keyword
    - Execution of the UPDATE is delayed until no other clients are reading from the table
    - This affects only the storage engines that use table-level locking (MyISAM, MEMORY, MERGE)

The UPDATE statement supports the following modifiers

1. If you use the IGNORE keyword
   - The update statement does not abort even if errors occur during the update
   - Rows for which duplicate-key conflicts occur are not updated
   - Rows for which columns are updated to values that would cause data conversion errors are updated to the closest valid values

UPDATE Syntax – One Table

- If you access a column from the table to be updated in an expression, UPDATE uses the current value of the column
  - For example, the following statement sets the age column to one more than its current value:
  - UPDATE persondata SET age=age+1;

UPDATE Syntax – One Table

- Single-table UPDATE assignments are generally evaluated from left to right
- For multiple-table updates, there is no guarantee that assignments are carried out in any particular order
- If you set a column to the value it currently has, MySQL notices this and does not update it

UPDATE Syntax – One Table

- If an UPDATE statement includes an ORDER BY clause, the rows are updated in the order specified by the clause
  - This can be useful in certain situations that might otherwise result in an error
  - Suppose that a table ‘t’ contains a column ‘id’ that has a unique index
  - *Continues to NEXT SLIDE*

UPDATE Syntax – One Table

- The following statement could fail with a duplicate-key error, depending on the order in which rows are updated:

```
UPDATE tOne SET id = id + 1;
```

- For example, if the table contains 1 and 2 in the id column and 1 is updated to 2 before 2 is updated to 3, an error occurs
  - *Continues to NEXT SLIDE*

UPDATE Syntax – One Table

- To avoid this problem, add an ORDER BY clause to cause the rows with larger id values to be updated before those with smaller values:

  e.g. UPDATE t SET id = id + 1 ORDER BY id DESC;

UPDATE Syntax – Many Tables

- Starting with MySQL 4.0.4, you can perform UPDATE operations covering multiple tables
- However, you cannot use ORDER BY or LIMIT with a multiple-table UPDATE statement

UPDATE Syntax – Many Tables

- The table_references clause lists the tables involved in the join
  - JOIN syntax will describe later

e.g.

```
UPDATE items,month
SET items.price=month.price
WHERE items.id=month.id;
```

UPDATE Syntax – Many Tables

- The preceding example shows an inner join that uses the comma operator
  - Multiple-table UPDATE statements can use any type of join allowed in SELECT statements, such as RIGHT JOIN, LEFT JOIN, and FULL JOIN

UPDATE Syntax – Many Tables

- Before MySQL 4.0.18, you need the UPDATE privilege for all tables used in a multiple-table UPDATE, even if they were not updated
- As of MySQL 4.0.18, you need only the SELECT privilege for any columns that are read but not modified