COMP283-Lecture 6

Applied Database Management

Introduction	
Database Administration	More Optimisation
	Maintaining Data Integrity
	Improving Performance

COMP283-Lecture 6
DB Administration: Full-text index

- Full Text Index
  - Index large text based attributes (e.g. documents)
  - Should (ideally) be kept up to date
    - automatically - potentially a higher processor load
    - manually - index not always up to date (not MySQL)
  - MySQL - choice of storage engine.
    - Only some support Full Text Indexes.
    - Only CHAR, VARCHAR and TEXT type columns

COMP283-Lecture 6
DB Administration: Full-text index

- Modern Operating Systems use full text indexes (& more) for searching. e.g. Mac OS
- Filesystem tracks changes to files using low-level filesystem hooks.
- The search index can be maintained in real time, automatically.
- A list is maintained by the OS, of files whose contents have changed.
  - For what purpose?
- In the event of filesystem errors, or modifications to the filesystem through some other means, the index has to be rebuilt.

Quick Demo!

COMP283-Lecture 6
DB Administration: Constraints

- Constraints on a table are Rules.
  - Applied to the data being inserted or modified.
- Constraints ensure that appropriate data is entered.
- Constraints can set default values.
  - If a record is inserted and it missing some attributes, it can have default values applied.

COMP283-Lecture 6
DB Administration: Views

- Views are stored queries. Treated like tables – they can be indexed and queried.
- Views are useful to restricting access to a limited subset of data attributes (can be across multiple tables)
- Views are useful to tunnel through security.
- 3 Types: Standard, Indexed, and Partitioned views.
- Example – create a view:

```
CREATE VIEW vwRockMusic AS
SELECT strArtist, strAlbum, strSong FROM
tblAlbums
WHERE tblAlbums.Genre = “Rock”;
```

COMP283-Lecture 6
DB Administration: Stored Procedures

- Stored procedures are a sequence of executable SQL statements, compiled and saved within the database.
- A stored procedure is often a saved query – the difference between a Stored Procedure and a view is that the stored procedure can have parameters passed to it and is more dynamic.   i.e. You can pass query criteria (WHERE clause parameters) to the stored procedure.
- Can simplify client applications and avoid the need for changes to it if DB structure changes needed. How?
- The principal* must have, directly or inherited, Execute permission on the Stored Procedure.

*user or application program

COMP283-Lecture 6
DB Administration: Stored Procedures

- MySQL Stored Procedure introduced in Version 5.0.
- In most DBMS stored procedures are compiled and stored in the database.
- MySQL stored procedures are compiled on demand.
- After compiling a stored procedure, it is cached.
- MySQL maintains its own stored procedure cache for every single connection. If an application uses a stored procedure multiple times in a single connection, the compiled version is used, otherwise, the stored procedure works like a query.

COMP283-Lecture 6
DB Administration: Stored Procedures: +VE

- Help reduce the traffic between application and database server
  - Instead of sending multiple lengthy SQL statements, the application has to send only name and parameters of the stored procedure.
- Are reusable and transparent to any applications.
  - They don't expose the database interface to all applications
  - developers don't have to develop functions that are already supported in stored procedures.
- They are secure.
  - The DBA can grant appropriate permissions to applications that access stored procedures in the database without giving any permissions on the underlying database tables.

COMP283-Lecture 6
DB Administration: Stored Procedures: -VE

- If using lots of stored procedures, memory usage of every connection will increase substantially.
- If you overuse a large number of logical operations inside stored procedures, CPU usage will also increase - the database server is not well-designed for logical operations.
- Constructs of stored procedures make it more difficult to develop stored procedures that have complicated business logic.
- It is difficult to debug stored procedures.
  - Few DBMS allow you to debug stored procedures. MySQL is not one of them.
- Not easy to develop and maintain stored procedures.
  - Often required a specialized skill set that not all application developers possess.

COMP283-Lecture 6
DB Administration: Stored Procedures

- Examples of Stored Procedures in MySQL

```
DELIMITER //
CREATE PROCEDURE GetAllProducts()
  BEGIN
  SELECT *  FROM products;
  END //
DELIMITER ;
```

changes the standard delimiter (;) to //
(not part of stored proc definition)

COMP283-Lecture 6
DB Administration: Stored Procedures

- Examples of Stored Procedures in MySQL

```
DELIMITER //
CREATE PROCEDURE GetAllProducts()
  BEGIN
  SELECT *  FROM products;
  END //
DELIMITER ;
```

Use this to create the new stored procedure. (note () after the name)

COMP283-Lecture 6
DB Administration: Stored Procedures

- Examples of Stored Procedures in MySQL

```
DELIMITER //
CREATE PROCEDURE GetAllProducts()
  BEGIN
  SELECT *  FROM products;
  END //
DELIMITER ;
```

Stored procedure body is between BEGIN and END statements

COMP283-Lecture 6
DB Administration: Stored Procedures

- Examples of Stored Procedures in MySQL

```
DELIMITER //
CREATE PROCEDURE GetAllProducts()
  BEGIN
  SELECT *  FROM products;
  END //
DELIMITER ;
```

changes the standard delimiter back to ;

COMP283-Lecture 6
DB Administration: Stored Procedures

```
DELIMITER //
CREATE PROCEDURE CountOrderByStatus(
 IN orderStatus VARCHAR(25),
 OUT total INT)
BEGIN
 SELECT count(orderNumber)
 INTO total
 FROM orders
 WHERE status = orderStatus;
END//
DELIMITER ;
```

```
CALL CountOrderByStatus('in process',@total);
SELECT @total AS  total_in_process;
```

```
+------------------+
| total_in_process |
+------------------+
| 301              |
+------------------+
```

COMP283-Lecture 6
DB Administration: Triggers

- A trigger is a named database object associated with a table, that activates when a particular event occurs for the table.
  - e.g. perform checks of values to be inserted into a table or to perform calculations on values involved in an update.
- In MySQL, a trigger is defined to activate when a statement inserts, updates, or deletes rows in the associated table.
- These row operations are trigger events.
  - e.g. rows can be inserted by INSERT or LOAD DATA statements, and an insert trigger activates for each inserted row.
- A trigger can be set to activate either before or after the trigger event.

COMP283-Lecture 6
DB Administration: Triggers

- Examples of Triggers in MySQL

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
    -> FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> SET @sum = 0;
mysql> INSERT INTO account VALUES(137,14.98),(141,1937.50),
(97,-100.00);
mysql> SELECT @sum AS 'Total amount inserted';
+-----------------------+
| Total amount inserted |
+-----------------------+
| 1852.48               |
+-----------------------+
```

COMP283-Lecture 6
DB Administration: Triggers

- Can use a trigger in MySQL to setup an audit trail e.g.

```
CREATE TABLE employees_audit (
    id INT AUTO_INCREMENT PRIMARY KEY,
    employeeNumber INT NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    changedat DATETIME DEFAULT NULL,
    action VARCHAR(50) DEFAULT NULL
);
```

```
DELIMITER $$
CREATE TRIGGER before_employee_update
    BEFORE UPDATE ON employees
    FOR EACH ROW
BEGIN
    INSERT INTO employees_audit
    SET action = 'update',
     employeeNumber = OLD.employeeNumber,
        lastname = OLD.lastname,
        changedat = NOW();
END$$
DELIMITER ;
```

COMP283-Lecture 6

Conclusion

- Full Text Indexing
- Maintaining Data Integrity
- Improving Performance