Session 19

Transaction Management and Globalization

Objectives

- Define and describe stored procedures
- Explain transactions and how to handle them
- Explain the impact of transactions on performance
- Describe MySQL support for different languages and time zones

Stored Routines 1-2

A stored routine is defined as a collection of SQL commands that are stored and executed in the server.

- One of the most substantial features supported by MySQL 5.0 and later versions is stored routines.
- A stored routine can either be a procedure or a function.
- Stored routines are created with the `CREATE PROCEDURE` and `CREATE FUNCTION` statements respectively.
- Stored routines are formed using a group of SQL commands.

Stored Routines 2-2

The table lists the prominent differences between the two types of stored routines:

	Procedures	Functions
Calling	Only with CALL	Possible in all SQL commands, For example, SELECT and UPDATE commands
Return	Returns one or more SELECT results	Returns a single value (command RETURN); the return values data type must be mentioned in the declaration with RETURNS
Parameters	Value and reference parameters possible (IN, OUT, and INOUT)	Only value parameters are allowed, thus, no identification with IN and so on
Commands allowed in the code	All SQL commands including SELECT, INSERT, UPDATE, DELETE, and CREATE TABLE	No commands that access tables
Calling other functions and procedures	May call other procedures and functions	May call only functions but not procedures

Stored Procedures

- In MySQL, stored procedures are saved in the table `mysql.proc`.
- Stored procedures have both **advantages** and **disadvantages**:

Greater speed

Increased database security

Reduced code redundancy, which results in better maintenance

Cannot be imported from one database system to another

Syntax Rules for Stored Procedures

Semicolon	• Stored procedures can contain numerous SQL commands and they must be delimited by semicolon.
BEGIN–END	• Stored procedure code that includes more than one instruction must be placed between BEGIN and END.
New lines	• A new line is syntactically the same as a space character. It is thus acceptable to place an IF–THEN–ELSE–END-IF construction on a single line or over multiple lines.
Case distinction	• Stored procedures are case-insensitive. For example, length is the same whether it is typed LENGTH or length.
Variables	• Local variables (internal to a stored procedure) and parameters are used with no @ character prefixed to them.
Comments	• Comments are written using a double hyphen (—) and extend up to the end of the line.

Parameters with Stored Procedures

- **CREATE PROCEDURE** is the command to create a procedure. Providing a parameter list to this command is optional. Thus, the syntax is as follows:

```
CREATE PROCEDURE name ([parameterlist])
options] sqlcode
```

- If there are multiple parameters, then they are separated by a comma. The syntax is as follows:

```
[IN or OUT or INOUT] <parametername>
<datatype>
```

- The three keywords IN, OUT, and INOUT determine the following:
  - **IN**: If the parameter is used only for input
  - **OUT**: If the parameter is used only for output
  - **INOUT**: If the parameter is used for data transferring in both directions

  The default is **IN**.

Deleting Stored Procedures

- To delete an existing procedure, use:
  `DROP PROCEDURE [IF EXISTS] name`

- The keyword `IF EXISTS` is optional and allows to execute the command without returning an error in case the stored procedure does not exist.

- Only users who have the privilege or access rights to create a routine and alter a routine can create, modify, or delete stored procedures.

- The `Execute` privilege determines users who are allowed to execute stored procedures.

- MySQL identifies the following two SECURITY characteristics:
  - SQL SECURITY INVOKER
  - SQL SECURITY DEFINER

Transactions

- A transaction is a sequence of one or more SQL statements that together form a logical unit of work.
- Every statement that forms a transaction executes certain portions of an operation.
- All the statements must complete successfully to conclude the operation.
- A transaction must pass the ACID test.

ACID Properties

Atomic

- A transaction has to be atomic by nature, means either none of the tasks in a transaction is executed or all of them are.
- In case of a failure after a few statements are processed, the actions of the processed statements must be rolled back.

Consistent

- A transaction must maintain consistency of the database even if there is a failure.
- When each transaction ends, the database should be dependable.

Isolated

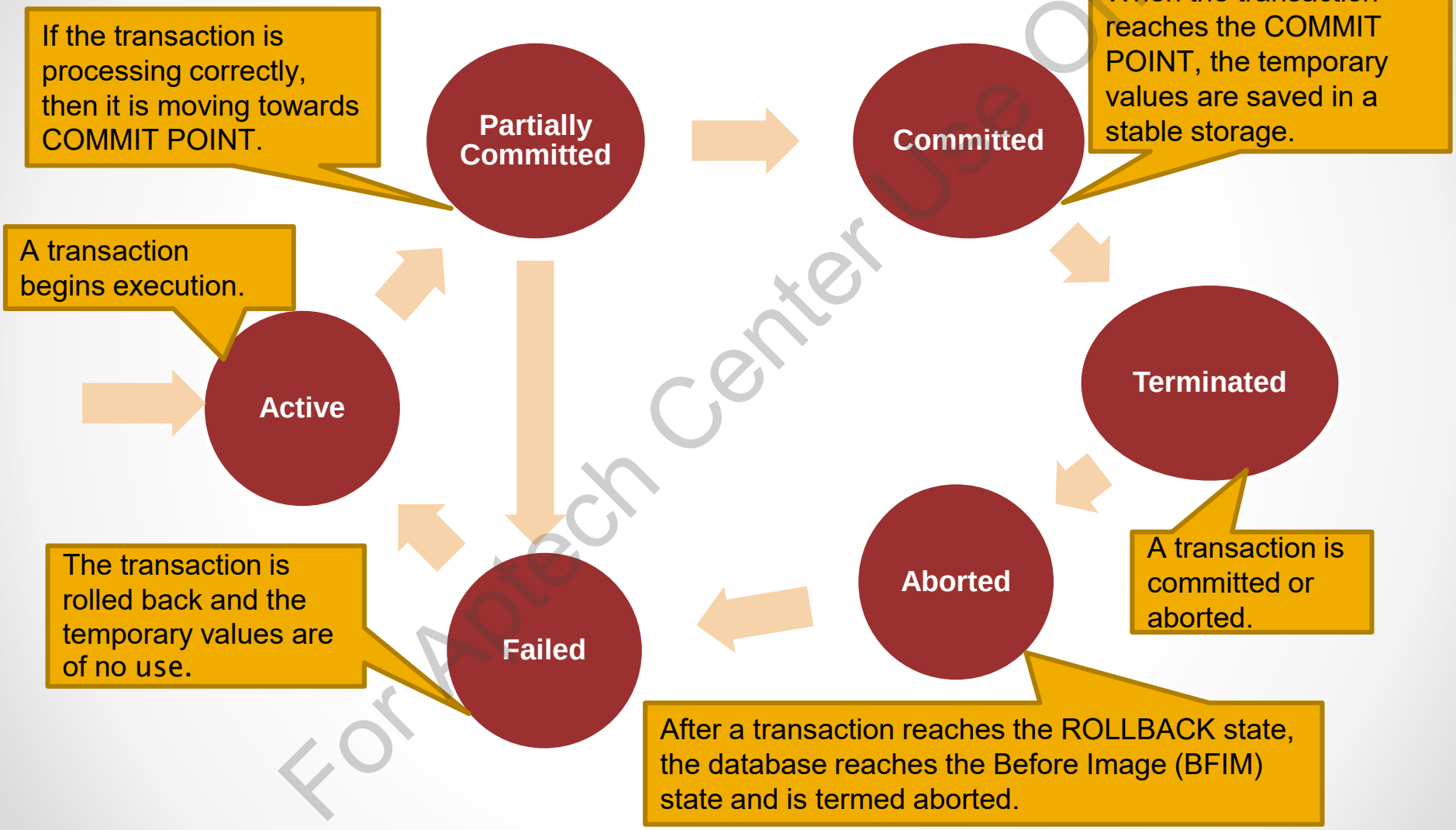
- A transaction must work independently without affecting other transactions.
- No transaction can modify data from other transactions until those transactions have executed completely.

Durable

- After a transaction is completed, all the modifications done by it must be saved.
- There must be consistency in data even if there is any hardware or application failure after execution of the transaction.

Lifecycle of a Transaction

The six states of a transaction are:



ANSI/ISO SQL Transaction Model

ANSI/ISO SQL standard describes following seven statements that support processing of a transaction:

START TRANSACTION	• Sets the attributes of a new transaction.
SET TRANSACTION	• Sets the attributes of the next transaction to be executed.
SET CONSTRAINTS	• Sets the constraint mode within a current transaction.
SAVEPOINT	• Sets a savepoint in a transaction. A savepoint is a position in a transaction that serves as an intermediate retrieval point.
RELEASE SAVEPOINT	• Allows you to free a savepoint and also release all the resources it may be storing.
COMMIT	• Ends a successful transaction and saves all the modifications done to the database.
ROLLBACK	• Without a savepoint, it ends a failed transaction and returns it to the beginning of a transaction. With a savepoint, it takes back the transaction to the specified savepoint.

Controlling Transactions 1-5

MySQL uses the following statements to control transactions:

START TRANSACTION

SET TRANSACTION

SAVEPOINT

ROLLBACK TO SAVEPOINT

RELEASE SAVEPOINT

COMMIT

ROLLBACK

Controlling Transactions 2-5

START TRANSACTION	SET TRANSACTION
Begins a new transaction and allows a user to set a few attributes of the transaction.	Sets attributes to be used by the next transaction. It cannot affect any existing transaction as these attributes cannot be used within the current transaction.
Does not have any LOCAL option	Has a LOCAL option, which allows to set options for local server execution of a transaction that ranges over multiple servers.

Following are the three attributes that can be set by these statements:

- **Isolation level**: It describes how independent a transaction will be from other transactions.
- **Access level**: It describes if the transaction can include statements to make any changes to the database (READ WRITE) or just read (READ ONLY) it.
- **Diagnostics size:** It sets the diagnostics area that is used for diagnostic information raised by SQL statements.

Controlling Transactions 3-5

SAVEPOINT	• Sets a savepoint with a unique name. • If a savepoint with the same name already exists, the old one is removed and the new one is set.
ROLLBACK TO SAVEPOINT	• Rolls back a transaction to the specified savepoint without aborting it. • Changes done to the transaction after setting the savepoint are cancelled.
RELEASE SAVEPOINT	• Takes out the specified savepoint from the list of savepoints that are set. • COMMIT or ROLLBACK does not take place because it is an error for a nonexistent savepoint.

Controlling Transactions 4-5

The commit and rollback statements help to explicitly end transactions. Following is the syntax:

- `COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]`
- `ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE]`

Where:

- `WORK` is added to make it compatible with some SQL implementations that support it.
- `AND [NO] CHAIN` specifies whether a new transaction is to be automatically started with the same attributes as the transaction that just terminated.
- `TO SAVEPOINT` calls a ROLLBACK to a savepoint that was set previously in a transaction rather than starting from the beginning.

Controlling Transactions 5-5

The following statements cause an implicit commit:

- DDL statements that define or alter database objects such as ALTER DATABASE and ALTER EVENT
- Statements that implicitly use or alter tables in the MySQL database such as ALTER USER and CREATE USER
- Transaction-control and locking statements such as BEGIN and LOCK TABLES
- Data loading statements such as LOAD DATA INFILE
- Replication control statements such as ANALYZE TABLE and CACHE INDEX
- Administrative statements such as START SLAVE and STOP SLAVE

Transaction and Performance

The performance of a database should not decrease with the number of users supported.

A database has to provide service to multiple users with many actions such as update, delete, and insert.

Each user must be able to use the database without being interrupted by the actions of other users.

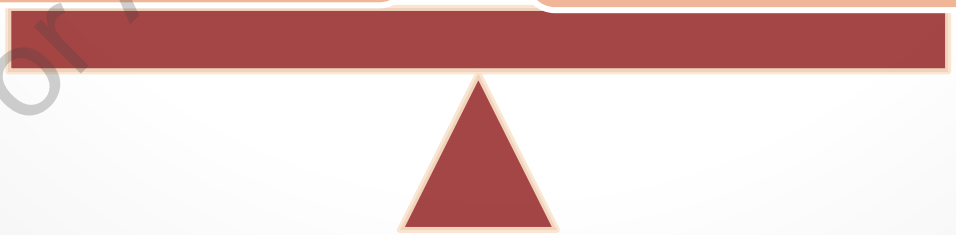
Anytime during a transaction, the database view remains consistent to a user.

In mid-transaction, a user cannot see uncommitted changes done by others.

In mid-transaction, users cannot be affected by committed changes made by others.

Locking and Deadlock

Locking	Deadlock
Locking prevents other transactions from using the locked data simultaneously.	None of the transactions can proceed because each one holds a lock on the data that the other one needs.
A transaction can obtain exclusive access temporarily to the required sections of a database.	The lock will not be released by either of the transactions as they are waiting for the data to become available.
The two types of lock are shared and exclusive locks.	Deadlocks are automatically identified by InnoDB and the respective transactions are rolled back to break deadlock.



Character Sets and Collations

A **character set** comprises a collection of symbols and encodings.

A **collation** comprises a set of rules used to compare characters in a character set.

To use following features, you must be familiar with different character sets and collations:

- Save strings with the help of different character sets.
- Compare strings with the help of different collations.
- Combine strings with different collations on the same server, character sets, database, or table.
- Validate the requirements of a character set and collation at all stages.

Time Zones Supported by MySQL 1-2

Different Time Zones

System Time Zone	Server's Current Time Zone	Per-connection Time Zones
On startup, the server sets the **system_time_zone** variable after deciding the host machine's time zone. To set the system time zone, set the option **--timezone=timezone_name** to **mysqld_safe**.	The variable global **time_zone** shows the time zone that the server is following. Use the option **--default-time-zone=timezone** in the command line to initially set the global time zone value.	Every client connecting to the server has its own time zone setting specified by the variable session **time_zone**. Client can modify the value using **SET time_zone = timezone**.

Time Zones Supported by MySQL 2-2

- To view the existing values of the client-specific time zone, use `SELECT @@global.time_zone`.
- To view the existing values of global time zone, use `SELECT @@session.time_zone`.
- The values for time_zone can be specified in multiple formats and they are case-insensitive:

The value 'SYSTEM' implies that the time zone matches the system time zone.

The value can be specified as a string representing an offset from UTC, for example, '+11:00' or '-7:00'.

The value can be specified as names, for example, 'Europe/Helsinki', 'US/Eastern', or 'MET'.

Populating the Time Zone Tables

- The MySQL database has numerous tables to maintain time zone information.
- The time zone tables are created during the installation, but must be manually populated.
- To populate time zone tables using the `mysql_tzinfo_to_sql` program, on the command line, provide the `zoneinfo` directory path name to `mysql_tzinfo_to_sql` and pass the output into the mysql program.
- When time zone rules changes, applications following the old rules become outdated.
- To stay updated, it is essential for the system to use the current time zone information.

MySQL Server Locale Support

The features of the locale used by MySQL server are:

The **lc_time_names** system variable specifies the locale that governs the language used for displaying the day, month, and abbreviations.

This variable affects the output from the functions: **DATE_FORMAT()**, **DAYNAME()**, and **MONTHNAME()**.

This variable has no impact on the **FORMAT()** function, but it enables a locale, which is used for the decimal point, grouping between separators, and thousands separator of the result number.

'en_US' is the default irrespective of the system's locale setting. The value can also be specified during the server startup or by setting the **GLOBAL** value if granted with **SUPER** privilege.

Any client can view the value of **lc_time_names** or set its SESSION value to influence the locale for its own connection.

Summary

- A stored routine is defined as a collection of SQL commands that are stored and executed in the server.
- A stored routine can be either a procedure or a function.
- CREATE PROCEDURE is the command to create a procedure. Providing a parameter list to this command is optional.
- A transaction is a sequence of one or more SQL statements that together form a logical unit of work.
- The six states of a transaction are active, partially committed, failed, aborted, committed and terminated.
- Locking prevents other transactions from using the locked data simultaneously.
- Deadlock is a condition in which none of the transactions can proceed because each one holds a lock on the data that the other one needs.
- A character set comprises a collection of symbols and encodings.
- A collation comprises a set of rules used to compare characters in a character set.
- Different time zone settings used by MySQL are System time zone, Server's current time zone, and Per-connection time zones.