Chapter 15

MySQL Disaster Recovery

MySQL Logging

- MySQL uses two types of log files
- **Redo Log** is the primary MySQL transaction log
- **Binary Log** is an auxiliary transaction log
  - provides incremental backup and point-in-time recovery capability

MySQL Redo Log

- Implements traditional transaction logging
  - Records each update made to table data with an increasing Log Sequence Number
  - Includes checkpoint, commit, rollback records
- Used to recover database to a consistent state after a database crash
  - Recovery procedure is automatic

MySQL Binary Log

- Not a traditional transaction log
  - Records only committed transactions
  - No checkpoint / rollback records
- Stores info in a series of files
  - Base filename specified in MySQL config
  - Adds sequence number extension .000001, .000002, etc.
- Used to implement incremental backups
- Must be enabled in MySQL config

Log File Management

- MySQL config specifies maximum size for binary and redo logs
- **Binary log:** when max size reached, MySQL starts a new log file
  - Configure maximum number of binary log files to store
- **Redo log:** when max size reached, MySQL rotates between a fixed number of log files

MySQL Backup

MySQL Backup Options

- Offline ("cold") backup
  - Database File Copy
- Online ("hot") backup
  - MySQL Enterprise Backup
  - mysqldump
- Incremental backup with binary logs

Cold Backup Procedure

1. Shut down MySQL Server
2. Copy data files using Windows file copy or batch script
3. Start MySQL Server

Hot Backup Options

Enterprise Backup

- Creates both full and incremental backups
- Backup files written in an efficient binary format
- Restores more quickly than mysqldump scripts

mysqldump

- Full backups only
- Generates sql script
- Restores can take a long time

mysqldump

- By default, does not guarantee a consistent backup of the database (!)
  - Backup script may contain partial transaction updates
- Use the --single-transaction option to obtain a consistent backup

Incremental Backup

- Enable binary logs
- To take an incremental backup:
  - Use FLUSH LOGS to start a new binary log file
  - Consider old binary log file as the incremental backup

Incremental Backup

- To restore an incremental backup:
  - Must first use the **mysqlbinlog** utility to convert binary log file to a format that can be processed by mysql client
  - Use mysql command line tool or Workbench Query Editor to replay the transactions in the converted log
- Example:
  - mysqlbinlog binlog.000001 binlog.000002 | mysql -u root -p

MySQL Backup Procedure

- Using mysqldump and binary logs:
  1. Use mysqldump to take a full backup
  2. Periodically use FLUSH LOGS to take incremental backup

MySQL Restore Procedure

- Using mysqldump and binary logs:
  1. Use mysql to restore full database backup created by mysqldump
  2. Restore incremental backups contained in binary logs using mysqlbinlog and mysql utilities

MySQL Point in Time Recovery

- mysqlbinlog provides --stop-datetime option to process all records up to a specific point in time
- Use this option to process all log records up to a given moment
- Example:
  - mysqlbinlog --stop-datetime="2005-04-20 9:59:59" translog.000005 | mysql -u root -p

Further Reading

- MySQL Redo Log
  - https://dev.mysql.com/doc/refman/8.0/en/innodb-redo-log.html
- MySQL Binary Log
  - https://dev.mysql.com/doc/refman/8.0/en/binary-log.html