# The City Lit Institute

##### Department of Computing

## Keeley Street, Holborn, London WC2B 4BA

**MySQL**

**(with Apache server)**

**Tuning**

**LECTURER** : **ALEXANDER ADU-SARKODIE**

MSc. Telematics (IT & Telecom), MSc. Eng., Dip. Russ. Lang., Teach. Cert, AMIAEng (UK), MBCS (UK), MIfL(UK)

**Blog** : <http://www.blogger.com/profile/14800490193632788559>

**Email:** [aadusarkodie1@gmail.com](mailto:aadusarkodie1@gmail.com)

### MySQL Schema Optimization:

* Keep your database trim.
* Archive old data – to remove excessive row returns or searches on queries.
* Put indexes on your data.
* Do not overuse indexes, compare with your queries.
* Compress text and blob data types – to save space and reduce number of disk reads.
* UTF 8 and UTF16 is slower than latin1.
* Use Triggers sparingly.
* Keep redundant data to a minimum – do not duplicate data unnecessarily.
* Use linking tables rather than extending rows.
* Pay attention to your data types, use the smallest one possible for your real data.
* Separate blob/text data from other data if other data is often used for queries when blob/text are not.
* Check and optimize tables often.
* Rewrite InnoDB tables often to optimize.
* Sometimes, it is faster to drop indexes when adding columns and then add indexes back.
* Use different storage engines for different needs.
* Use ARCHIVE storage engine for Logging tables or Auditing tables – this is much more efficient for writes.
* Store session data in memcache rather than MySQL – memcache allows for auto-expiring values and prevents you from having to create costly reads and writes to MySQL for temporal data.
* Use VARCHAR instead CHAR when storing variable length strings – to save space since CHAR is fixed length and VARCHAR is not (utf8 is not affected by this).
* Make schema changes incrementally – a small change can have drastic effects.
* Test all schema changes in a development environment that mirrors production.
* Do NOT arbitrarily change values in your config file, it can have disastrous affects.
* Sometimes less is more in MySQL configs.
* When in doubt use a generic MySQL config file.

**Query Optimization:**

* Use the slow query log to find slow queries.
* Use EXPLAIN to determine queries are functioning appropriately.
* Test your queries often to see if they are performing optimally – performance will change over time.
* Avoid count(\*) on entire tables, it can lock the entire table.
* Make queries uniform so subsequent similar queries will use query cache.
* Use GROUP BY instead of DISTINCT when appropriate.
* Use indexed columns in WHERE, GROUP BY, and ORDER BY clauses.
* Keep indexes simple, do not reuse a column in multiple indexes.
* Sometimes MySQL chooses the wrong index, use USE INDEX for this case
* Check for issues using SQL\_MODE=STRICT.
* Use a LIMIT on UNION instead of OR for less than 5 indexed fields.
* Use INSERT ON DUPLICATE KEY or INSERT IGNORE instead of UPDATE to avoid the SELECT prior to update.
* Use a indexed field and ORDER BY instead of MAX.
* Avoid using ORDER BY RAND().
* LIMIT M,N can actually slow down queries in certain circumstances, use sparingly.
* Use UNION instead of sub-queries in WHERE clauses.
* For UPDATES, use SHARE MODE to prevent exclusive locks.
* On restarts of MySQL, remember to warm your database, to ensure that your data is in memory and queries are fast.
* Use DROP TABLE then CREATE TABLE instead of DELETE FROM to remove all data from a table.
* Minimize the data in your query to only the data you need, using \* is overkill most of the time.
* Consider persistent connections instead of multiple connections to reduce overhead.
* Benchmark queries, including using load on the server, sometimes a simple query can have affects on other queries.
* When load increases on your server, use SHOW PROCESSLIST to view slow/problematic queries.
* Test all suspect queries in a development environment where you have mirrored production data.

### MySQL Backup Procedures:

* Backup from secondary replicated server.
* Stop replication during backups to prevent inconsistencies on data dependencies and foreign constraints.
* Stop MySQL altogether and take a backup of the database files.
* Backup binary logs at same time as dumpfile if MySQL dump used – to make sure replication does not break.
* Make dumps per table for easier single table recovery – if data is isolated from other tables.
* Use –opt when using mysqldump.
* Check and Optimize tables before a backup.
* When importing temporarily disable foreign constraints for a faster import.
* When importing temporarily disable unique checks for a faster import.
* Calculate size of database/tables data and indexes after each backup to monitor growth.
* Monitor slave replication for errors and delay with a cron script.
* Perform Backups regularly.
* Test your backups regularly.