MySQL - RDBMS

Agenda

- Normalization
- Codd's rules
- MySQL Architecture
- RDBMS to NoSQL
- Full Text Searches
- Temporary Tables

Normalization

ER Diagram

- Lucid Chart (for ED diagram)
- Plant UML (for ED diagram)
    - https://www.planttext.com/
    - Syntax: https://plantuml.com/

```
@startuml

entity customers {
    - cid
    - cname
    - caddr
    - cstreet
    - ccity
    - cpin
    - cphone
}

entity orders {
    - oid
    - odate
    - odeldate
}

entity products {
    - pid
    - pname
    - prate
}

entity order_details {
    - oid
    - pid
    - pqty
}
```

```
entity city_pin {
    - pin
    - city
}

city_pin "1" -left- "*" customers
customers "1" -left- "*" orders
orders "1" -left- "*" order_details
products "1" -right- "*" order_details

@enduml
```

Physical Model

- SQL queries to create tables, relations and constraints.

Using project database

- CRUD operations on the tables
- Advanced reports -- Analysis

Analysis

- Which product is sold maximum in last year?
    - GROUP BY + JOIN (products & order_details)
- Find top 10 customers (max amount orders).
    - GROUP BY + JOIN (products, order_details, orders, customers)
    - If orders table also keep "final_bill" then JOIN (orders, customers)
- Find top 2 cities making maximum business.
    - GROUP BY + JOIN (products, order_details, orders, customers, city_pin)
    - If orders table also keep "final_bill" then JOIN (orders, customers, city_pin)

MySQL Architecture

```
CREATE TABLE items(
    id INT PRIMARY KEY,
    name VARCHAR(20),
    price DOUBLE
) ENGINE=MyISAM;

INSERT INTO items VALUES(1, 'Item1', 200.00);
INSERT INTO items VALUES(2, 'Item2', 100.00);

CREATE TABLE cust(
    id INT NOT NULL,
    name VARCHAR(20) NOT NULL
) ENGINE=CSV;

INSERT INTO cust VALUES (1, 'Nilesh'), (2, 'Nitin'), (3, 'Yogesh'), (4, 'Vijay'),
```

```
(5, 'Soniya');

SELECT * FROM cust;
```

Physical architecture

- Configuration files
  - /etc/mysql/my.cnf
- Installed Files
  - Executable files
    - mysqld
    - mysqladmin
    - mysql
    - mysqldump
    - ...
  - Log files
    - pid files
    - socket files
    - document files
    - libraries
- Data Files
  - Data directory
    - Server logs
    - Status files
    - Innodb tablespaces & log buffer
  - Database directory files
    - Data & Index files (.ibd)
    - Object structure files (.frm, .opt)

Logical architecture

- Refer diagram in slides.
- Client
- Server (mysqld)
  - Accept & process client requests
  - Multi-threaded process
  - Dynamic memory allocation
    - Global allocation
    - Session allocation
- Parser
  - SQL syntax checking.
  - Generate sql_id for query.
  - Check user authentication.
- Optimizer
  - Generate efficient query execution plan
  - Make use of appropriate indexes
  - Check user authorization.

- Query cache
    - Server level (global) cache
    - Speed up execution if identical query is executed previously
- Key cache
    - Cache indexes
    - Only for MyISAM engine
- Storage engine
    - Responsible for storing data into files.
    - Features like transaction, storage size, speed depends on engine.
    - Supports multiple storage engines
    - Can be changed on the fly (table creation)
    - Important engines are InnoDB, MyISAM, NDB, Memory, Archive, CSV.

InnoDB engine

- Fully transactional ACID.
- Table & Row-level locking.
- Offers REDO and UNDO for transactions.
- Shared file to store objects (Data and Index in the same file - .ibd)
- InnoDB Read physical data and build logical structure (Blocks and Rows)
- Logical storage called as TABLESPACE.
- Data storage in tablespace
    - Multiple data files
    - Logical object structure using InnoDB data and log buffer

MyISAM

- Non-transactional storage engine
- Table-level locking
- Speed for read
- Data storage in files and use key, metadata and query cache
    - .frm for table structure
    - .myi for table index
    - .myd for table data

NDB

- Fully Transactional and ACID Storage engine.
- Row-level locking.
- Offers REDO and UNDO for transactions.
- NDB use logical data with own buffer for each NDB engine.
- Clustering: Distribution execution of data and using multiple mysqld.

MySQL as NoSQL

- "Flexible Schema" -- JSON datatype
- Still ACID transactions -- Tight consistency
- Indexing on JSON fields

- Horizontal scaling limited -- GB to TB.

RDBMS to NoSQL migration

- An application is developed using RDBMS & in use.
- For getting advantages of NoSQL, we want to shift to NoSQL databases.
- This includes redevelopment of whole application, specially data layer of the application.
- Migrating from RDBMS to NoSQL will change from (NoSQL) database to database.

Migration from MySQL to MongoDb

1. Understand structure of data in MySQL i.e. understand tables, constraints, relations among tables & indexes.
2. Define data model for MongoDb. Make appropriate use of embedded model & reference model. Define collections, validations & indexes.
3. Create views in RDBMS which will make data available as per requirement of Mongo Db model.
4. Write a custom script that will read data from views & write into JSON files (in desired format).
5. In mongo db create collection, then import all json files using **mongoimport** command.
    - mongoimport -d dacdb -c emp emp.json
6. Create indexes & validators on mongo collections.

Full Text Search

- FTS allows keyword based searches. Need not to match exact text.
- FTS is applicable only for CHAR, VARCHAR and TEXT types.
- FTS is different than LIKE or REGEXP.
- FTS uses Natural Language Processing to make search more user friendly.
- MySQL FTS features
    - High speed - based on fulltext index.
    - Moderate index size - index size is not too large.
    - Dynamic index - index is auto-updated on DML.

```
CREATE TABLE table_name(
    column_list,
    ...,
    FULLTEXT (column1,column2, ...)
);
-- OR
ALTER TABLE table_name
ADD FULLTEXT(column_name1, column_name2, ...);
-- OR
CREATE FULLTEXT INDEX index_name
ON table_name(idx_column_name, ...);
```

```
ALTER TABLE table_name
DROP INDEX index_name;
```

```
CREATE TABLE tutorial (
id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
title VARCHAR(200),
description TEXT,
FULLTEXT(title,description)
) ENGINE=InnoDB;

INSERT INTO tutorial (title,description) VALUES
('SQL Joins','An SQL JOIN clause combines rows from two or more tables. It creates
a set of rows in a temporary table.'),
('SQL Equi Join','SQL EQUI JOIN performs a JOIN against equality or matching
column(s) values of the associated tables. An equal sign (=) is used as comparison
operator in the where clause to refer equality.'),
('SQL Left Join','The SQL LEFT JOIN, joins two tables and fetches rows based on a
condition, which is matching in both the tables and the unmatched rows will also
be available from the table before the JOIN clause.'),
('SQL Cross Join','The SQL CROSS JOIN produces a result set which is the number of
rows in the first table multiplied by the number of rows in the second table, if
no WHERE clause is used along with CROSS JOIN.'),
('SQL Full Outer Join','In SQL the FULL OUTER JOIN combines the results of both
left and right outer joins and returns all (matched or unmatched) rows from the
tables on both sides of the join clause.'),
('SQL Self Join','A self join is a join in which a table is joined with itself
(which is also called Unary relationships), especially when the table has a
FOREIGN KEY which references its own PRIMARY KEY.');

SELECT id, title, LEFT(description, 40) FROM tutorial;

SELECT * FROM tutorial WHERE MATCH(title,description) AGAINST ('left and right
join');
-- columns must be full text indexed

SELECT * FROM tutorial WHERE MATCH(title,description) AGAINST ('left right');

SELECT * FROM tutorial WHERE MATCH(title,description) AGAINST ('left right' IN
NATURAL LANGUAGE MODE);

SELECT id, MATCH(title,description) AGAINST ('left right' IN NATURAL LANGUAGE
MODE) AS score FROM tutorial;
-- relevence depend on the number of words in the row, the number of unique words
in that row, the total number of words in the collection, the number of documents
(rows) that contain a particular word.

SELECT * FROM tutorial WHERE  MATCH(title,description) AGAINST ('+Joins -right' IN
BOOLEAN MODE);

SELECT * FROM tutorial WHERE  MATCH(title,description) AGAINST ('full');

SELECT * FROM tutorial WHERE  MATCH(title,description) AGAINST ('full' WITH QUERY
EXPANSION);
```

FTS Modes

- Natural Language Mode
  - Default mode of searching.
  - Assign relevence to each row based on search terms.
    - 0 - Not similar.
    - +ve - Similar.
- Boolean Mode
  - Expert search based on operators to change relevence.
  -
    - Include, the word must be present.
  - – Exclude, the word must not be present.
  -
    > Include, and increase ranking value.
  - < Include, and decrease the ranking value.
  - () Group words into subexpressions (allowing them to be included, excluded, ranked, and so forth as a group).
  - ~ Negate a word's ranking value.
  -
    - Wildcard at the end of the word.
  - "" Defines a phrase (as opposed to a list of individual words, the entire phrase is matched for inclusion or exclusion).
- With Query Expansion
  - Search related words as well.

FTS Restrictions

- No spelling mistakes
- Minimum length of the search term defined in MySQL full-text search engine is 4.
- Stop-words are ignored e.g. is, are, and, to, on, in, as, ...

Temporary Tables

- Like Materialized View in Oracle.
- It stores data temporarily in in-memory table -- for current user session.
- Temporarily tables are visible only for current user current session.
- When current session EXIT, temporarily tables are relased.

```
SELECT * FROM emp WHERE sal > 2500;

-- CREATE VIEW v_richemp AS SELECT * FROM emp WHERE sal > 2500;
CREATE TEMPORARY TABLE tmp_richemp AS SELECT * FROM emp WHERE sal > 2500;

SELECT * FROM tmp_richemp;

UPDATE emp SET sal = 5500.0 WHERE ename='KING';
-- change in base table

SELECT * FROM emp;
```

```
SELECT * FROM tmp_richemp;
-- changes are not visible in temp table

SHOW TABLES;

SHOW FULL TABLES;
-- temp tables are not visible

DROP TEMPORARY TABLE tmp_richemp;

SELECT * FROM tmp_richemp;
-- error

CREATE TEMPORARY TABLE tmp_richemp AS SELECT * FROM emp WHERE sal > 2500;
-- new temp table created

SELECT * FROM tmp_richemp;

EXIT;
-- temp tables are destroyed
```

```
SELECT * FROM tmp_richemp;
-- error

SELECT * FROM INNODB_TEMP_TABLE_INFO;
-- see hidden tables in current session (but need PROCESS privilege)
```