MySQL

database

Offered by NEILIT, Imphal in collaboration with Lamzing Technologies Pvt. Ltd.

About this course

Duration	~4-5 weeks
Timing	Mon to Fri 9:30am to 12:00pm
Instructor	Bijen Hemam, Lamzing Technologies Pvt Ltd

Brief Introduction

What is a Database

- Anything that stores data?
- Is Excel a database?
- Some Properties of a DB
  - Data is stored in some structure (excel has structure)
  - Handle large amounts of data limited by disk space (excel has max limit to rows and columns)
  - Provides constraints, some built in validation
  - Indexing for quick retrieval of data
  - Multiple authorized users can access it simultaneously (excel has support for shared access)

Types of Database

- Relational database
  - MySQL
  - Oracle
- NoSQL database
  - Key-Value Stores. e.g. Redis
  - Document Stores. e.g. MongoDB

- Others object oriented, hierarchical, network, time-series, graph, spatial etc

Resources

- MySQL database installation
  - https://dev.mysql.com/downloads/

- Clients to connect to the MySQL database
  - https://dev.mysql.com/downloads/workbench/
  - https://dbeaver.io/download/

- Tutorial
  - https://www.w3schools.com/MySQL/default.asp

- Online Playground
  - https://www.programiz.com/sql/online-compiler/

MySQL SETUP

Tools

- MySQL database
  - https://dev.mysql.com/downloads/
- MySQL Workbench
  - https://dev.mysql.com/downloads/workbench/

- DATA Setup

  - https://drive.google.com/file/d/1WI91dfN0x22xAK6Uodal1SMk6kRjJgsm/view?usp=drive_link

SQL Topics

- **Data Manipulation Language (DML)**
- Data types
- Data Definition Language (DDL)
- SQL Joins
- Subqueries
- Aggregation Functions
- Constraints
- Indexes
- Transactions
- Views
- Triggers
- Stored Procedures

SELECT statement

```
SELECT column1, column2, ...
FROM table_name;

SELECT DISTINCT column1, column2, ...
FROM table_name;
```

Example

```
SELECT first_name, last_name, email
FROM customers;

SELECT DISTINCT customer_id
FROM orders;
```

WHERE clause

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Operators in The WHERE Clause	
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple values

Example

```
SELECT first_name, last_name, email
FROM customers
WHERE last_name = ‘Rice’;

SELECT first_name, last_name, email
FROM customers
WHERE id > 4 and id < 8;
```

Example

```
SELECT date_ordered, customer_id, total_amount
FROM orders
WHERE date_ordered between ‘2000-03-20’ and
‘2010-03-20’;


SELECT first_name, last_name, email
FROM customers
WHERE id in (4,5,6,7,8);
```

Example

```
SELECT first_name, last_name, email
FROM customers
WHERE first_name LIKE ‘A%’;

SELECT first_name, last_name, email
FROM customers
WHERE email LIKE ‘%arden%’;
```

ORDER BY keyword

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Example

```
SELECT first_name, last_name, email
FROM customers
ORDER BY first_name, last_name DESC;

SELECT first_name, last_name, email
FROM customers
ORDER BY last_name ASC, first_name DESC;
```

NULL Values

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;

SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

Example

```sql
SELECT name, IndepYear
FROM country
WHERE IndepYear = NULL;
```

```sql
SELECT name, IndepYear
FROM country
WHERE IndepYear IS NULL;
```

```sql
SELECT name, IndepYear
FROM country
WHERE IndepYear IS NOT NULL;
```

INSERT INTO statement

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Example

```
INSERT INTO city(id,
name,countrycode,district,population)
VALUES (1, 'Ukhrul', 'IND', 'Manipur', 56000);

INSERT INTO city(id,
name,countrycode,district,population)
VALUES (4080, 'Ukhrul', 'IND', 'Manipur', 56000);
```

Example

```
INSERT INTO country
(Code, Name, Continent, Region, SurfaceArea, IndepYear,
Population, LifeExpectancy, GNP, GNPOld, LocalName,
GovernmentForm, HeadOfState, Capital, Code2)
VALUES('DC1', 'Demo Country1', 'Asia', '', 100, 1865,
200000, 56.3, 3216, 0, 'DLocal', 'DGov', '', 0, 'DC');

INSERT INTO country
(Code, Name, Continent, SurfaceArea, Population)
VALUES('DC2', 'Demo Country2', 'Asia', 100, 200000);
```

UPDATE statement

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Example

```sql
UPDATE country
Set Population= 3000, LifeExpectancy = 66
WHERE Code = 'DC2';

UPDATE Country
SET Continent= 'Asia2'
WHERE Code = 'DC2';
```

DELETE statement

```
DELETE FROM table_name WHERE condition;
```

Example

```
DELETE FROM country
WHERE Code = 'DC2';

DELETE FROM country
WHERE Code = 'IND';
```

LIMIT OFFSET clause

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number OFFSET number;
```

Example

```
SELECT code, name, population
FROM country
order by name
limit 11

SELECT code, name, population
FROM country
order by name
limit 11 offset 10;
```

MIN() MAX() functions

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;

SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

Example

```
SELECT min(population)
FROM country;

SELECT max(population)
FROM country;
```

- Find the min and max population of countries in Europe?

Solution

```
SELECT min(population), max(population)
FROM country
WHERE Continent  = 'Europe'
```

COUNT() AVG() SUM() functions

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

Example

```sql
SELECT count(name)
FROM country; -- number of countries

SELECT AVG(Population)
FROM country; -- average population of countries

SELECT SUM(SurfaceArea)
FROM country; -- Total area of all the countries
```

Example

```
SELECT count(name),SUM(SurfaceArea),AVG(Population)
FROM country; -- multiple aggregate functions
```

- Find the total population of each Continent?

Solution

```
SELECT Continent,SUM(Population)
FROM country
group by Continent;
```

lamzing

CASE() function

```
CASE
     WHEN condition1 THEN result1
     WHEN condition2 THEN result2
     WHEN conditionN THEN resultN
     ELSE result
END;
```

Similar to IF condition in other programming languages

Example

```
SELECT code, name, population,
CASE
    WHEN population < 1000000 THEN 'SMALL POPULATION'
    WHEN population < 5000000 THEN 'MEDIUM POPULATION'
    ELSE 'LARGE POPULATION'
END AS population_size
FROM country;
```

SQL Topics

- Data Manipulation Language (DML)
- **Data types**
- Data Definition Language (DDL)
- SQL Joins
- Subqueries
- Aggregation Functions
- Constraints
- Indexes
- Transactions
- Views
- Triggers
- Stored Procedures

DataTypes

String Type	Number Type	Date Type
CHAR (0-255)	INT (4)	DATE
VARCHAR (0 to 65,535)	BIGINT (8)	DATETIME
ENUM	BOOL	TIMESTAMP
TEXT (large text)	DOUBLE	TIME
BLOB (images)	DECIMAL	YEAR

CHAR vs VARCHAR

Value	CHAR(4)	Storage Required	VARCHAR(4)	Storage Required
''	' '	4 bytes	''	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

SQL Topics

- Data Manipulation Language (DML)
- Data types
- **Data Definition Language (DDL)**
- SQL Joins
- Subqueries
- Aggregation Functions
- Constraints
- Indexes
- Transactions
- Views
- Triggers
- Stored Procedures

CREATE TABLE statement

```
CREATE TABLE example_person (
id int NOT NULL PRIMARY KEY AUTO_INCREMENT,
first_name CHAR(4) NOT NULL,
last_name VARCHAR(4) NOT NULL,
gender ENUM('M','F') NOT NULL,
bio TEXT NOT NULL
);
```

Example Inserts

```
INSERT INTO world.example_person
(first_name, last_name, gender, bio)
VALUES('AB', 'yuri', 'M', 'Yuri has been preparing
for the UPSC exam last 1 years.');
```

TABLE with BLOB

```
CREATE TABLE example_person_data (
  id int AUTO_INCREMENT NOT NULL PRIMARY KEY,
  person_id int NOT NULL,
  small_val int NOT NULL,
  large_val bigint NOT NULL,
  married bool NOT NULL,
  height double(4,2) NOT NULL,
  weight decimal(4,2) NOT NULL,
  CONSTRAINT example_person_data_FK FOREIGN KEY (person_id)
REFERENCES example_person(id)
);
```

Example Inserts

check folder accessible to mysql server and store the file in the folder

```
select @@GLOBAL.secure_file_priv;

INSERT INTO example_person_photo (person_id, photo)
VALUES
 (1, LOAD_FILE('/var/lib/mysql-files/flower.jpg'));
```

TABLE with Numeric datatype

```
CREATE TABLE `example_person_data` (
  `id` int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `person_id` int NOT NULL,
  `small_val` int NOT NULL,
  `large_val` bigint NOT NULL,
  `married` tinyint(1) NOT NULL,
  `height` double(4,2) NOT NULL,
  `weight` decimal(4,2) NOT NULL,
  CONSTRAINT `example_person_data_FK` FOREIGN KEY
(`person_id`) REFERENCES `example_person` (`id`)
);
```

Double, Decimal differs during calculation

```
INSERT INTO example_person_data(person_id, small_val, large_val, married,
height, weight)
VALUES(1, 2147483647, 9999999999999999, false, 0.0, 0.0);

INSERT INTO world.example_person_data(person_id, small_val, large_val, married,
height, weight)
VALUES(1, 2147483647, 9999999999999999, false, -13.21, 0.0);

INSERT INTO world.example_person_data(person_id, small_val, large_val, married,
height, weight)
VALUES(1, 2147483647, 9999999999999999, false, 59.60, 46.40);

INSERT INTO world.example_person_data(person_id, small_val, large_val, married,
height, weight)
VALUES(1, 2147483647, 9999999999999999, false, 30.40, 30.40);
```

SUM double vs decimal

```
select person_id,
       sum(height) as height,
       sum(weight) as weight
from example_person_data
group by person_id
```

TABLE with Date datatype

```
CREATE TABLE `example_person_date` (
   `id` int NOT NULL AUTO_INCREMENT PRIMARY KEY,
   `person_id` int NOT NULL,
   `birth_year` year NOT NULL,
   `birth_date` date NOT NULL,
   `birth_time` time NOT NULL,
   `birth_timezone` timestamp NOT NULL,
   `birth_datetime` datetime NOT NULL,
  CONSTRAINT `example_person_date_FK` FOREIGN KEY
(`person_id`) REFERENCES `example_person` (`id`));
```

Date Type

- DATE - format YYYY-MM-DD
- DATETIME - format: YYYY-MM-DD hh:mm:ss
- TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
- YEAR - format YYYY or YY

- Datetime is used to store just date and time and so has more range (before mysql version 5.6.4)
- datetime range: 1000-01-01 00:00:00 and 9999-12-31 23:59:59
- Timestamp range: '1970-01-01 00:00:01' to '2038-01-19 08:44:07'

Example

```
INSERT INTO world.example_person_date(person_id, birth_year,
birth_date, birth_time, birth_timezone, birth_datetime)
VALUES(1, 1980, '1980-01-23', '17:15', '2023-07-28
08:55:22.322', '2023-07-28 08:55:22.322');
```

SQL Topics

- Data Manipulation Language (DML)
- Data types
- Data Definition Language (DDL)
- **Subqueries**
- SQL Joins
- Aggregation Functions
- Constraints
- Indexes
- Transactions
- Views
- Triggers
- Stored Procedures

Example

- Countries with a population greater than say 500000000

```
SELECT name, population
FROM country
WHERE code IN ( SELECT code FROM country
WHERE population > 500000000 );

SELECT name, population
FROM country
WHERE population > 500000000;
```

Example

- **Cities** in the world with a population greater than the average population of all the cities in the world

```
SELECT name, population
FROM city
WHERE population >
( SELECT AVG(population) FROM city );
```

Example

- **Countries** in the world with a population greater than the average population of **its cities**

```
select c.Code, c.Name, c.Population
from country c
where Population >
( SELECT AVG(population)
 FROM city
 where countrycode = c.Code
);
```

Example

- Find countries with no cities

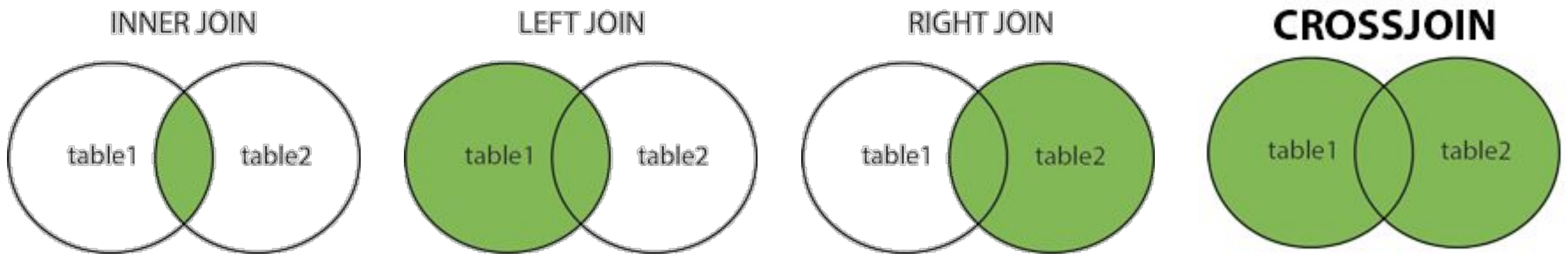
```
select count(*)
from country c1
where NOT EXISTS (
select name from city where countryCode = c1.Code
)
```

SQL Topics

- Data Manipulation Language (DML)
- Data types
- Data Definition Language (DDL)
- Subqueries
- **SQL Joins**
- Aggregation Functions
- Constraints
- Indexes
- Transactions
- Views
- Triggers
- Stored Procedures

Types of Table Join

- INNER JOIN: Returns records that have matching values in both tables
- LEFT JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT JOIN: Returns all records from the right table, and the matched records from the left table
- CROSS JOIN: Returns all records from both tables



https://www.w3schools.com/MySQL/mysql_join.asp

Example – inner join

```
SELECT code, name
from country
inner join city on Code = CountryCode;
```

```
SELECT c1.code, c1.name
from country c1
inner join city c2 on c1.Code = c2.CountryCode;
```

Exercise

1. Include City Name?
2. Add aliases to differentiate between the country name and city name?
3. Order the cities in India based on the population?
4. Find population of Imphal?
5. Find the District in India with most cities?
6. Find average population of indian District/states and sort it by highest value?
7. Find all the languages spoken in India?
8. List names of all the countries where ‘Dutch’ is spoken?
9. Which country has maximum language spoken in it?
10. Which language is spoken in maximum countries?

Example – Question 9

```
select c.name, count(c2.`Language`)
from country c
inner join countrylanguage c2 on c.Code = c2.CountryCode
group by c.Code
having count(c2.`Language`) = (
   select max(num_languages) from (
   SELECT COUNT(language) AS num_languages
   FROM countrylanguage
   GROUP BY CountryCode
   ORDER BY 1 desc ) as lang_count
);
```

Example – left join, right join

```
select ep.first_name, ep.last_name, ep.gender,
epd.person_id, epd.birth_date
from example_person ep
left join example_person_date epd on ep.id = epd.person_id;

select ep.first_name, ep.last_name, ep.gender,
epd.person_id, epd.birth_date
from example_person ep
right join example_person_date epd on ep.id = epd.person_id;
```

Example – cross join

```
CREATE TABLE example_shift (
shift_name varchar(45) NULL,
shift_start TIME NULL,
shift_end TIME NULL);

INSERT INTO example_shift (shift_name, shift_start,
shift_end)
VALUES('Morning', '05:00', '11:00'),
  ('Afternoon', '11:00', '18:00'),
  ('Evening', '19:00', '23:00'),
  ('Night', '23:00', '05:00');
```

Example – cross join

```
select ep.first_name, ep.last_name, es.*
from example_person ep
cross join example_shift es
```

SQL Topics

- Data Manipulation Language (DML)
- Data types
- Data Definition Language (DDL)
- SQL Joins
- Subqueries
- **Aggregation Functions**
- Constraints
- Indexes
- Transactions
- Views
- Triggers
- Stored Procedures

Aggregate Function and Clauses

- COUNT
- MIN
- MAX
- AVG
- SUM
- GROUP BY
- HAVING
- OVER, PARTITION BY (also call window function)

Example

- Find number of cities in each country

```
SELECT  CountryCode, count(name) AS no_of_cities
FROM city
GROUP BY CountryCode
```

- Find countries with number of cities between 15 and 20

```
SELECT  CountryCode, count(name) AS no_of_cities
FROM city
GROUP BY CountryCode
HAVING count(name) BETWEEN 15 AND 20
```

Example

- Find the country and the city with the maximum population

```
SELECT c.name, MAX(ci.population) AS max_city_population
FROM country c
LEFT JOIN city ci ON c.Code  = ci.CountryCode
GROUP BY c.code;
```

Example

- FInd average population of each state in the country India

```
SELECT District, avg(Population) AS avg_state_population
FROM city
WHERE CountryCode = 'IND'
GROUP BY District
```

Example

- In the previous example include the population of the city for comparison

```
SELECT District, name, Population as city_population,
avg(Population) OVER(PARTITION by District) AS avg_state_pop
FROM city
WHERE CountryCode = 'IND'
order by District
```

Exercise

- Find population density of India, Nepal, Bhutan

Density = Population / SurfaceArea

SQL Topics

- Data Manipulation Language (DML)
- Data types
- Data Definition Language (DDL)
- SQL Joins
- Subqueries
- Aggregation Functions
- **Constraints**
- Indexes
- Transactions
- Views
- Triggers
- Stored Procedures

Types of constraints

PRIMARY KEY	**Uniquely identifies each row in a table**
FOREIGN KEY	Ensure inserted data is present in parent table
NOT NULL	Ensures that a column cannot have a NULL value
UNIQUE	Ensures that all values in a column are different
CHECK	Ensures that the values in a column satisfies a specific condition
DEFAULT	Sets a default value for a column if no value is specified

Example

```
CREATE TABLE world.example_constraint (
id int auto_increment NOT NULL PRIMARY KEY,
not_null varchar(100) NOT NULL,
unique_key varchar(100) NULL,
status varchar(100) DEFAULT 'STARTED',
CONSTRAINT example_constraint_UN UNIQUE KEY (unique_key),
CONSTRAINT example_constraint_CHECK CHECK (status in
('STARTED', 'DONE', 'HOLD')) );

ALTER TABLE example_constraint
ADD CONSTRAINT example_constraint_CHECK CHECK (status in
('STARTED', 'DONE', 'HOLD'));
```

Example

```sql
INSERT INTO example_constraint(not_null, unique_key)
VALUES(NULL, 'unique1');

INSERT INTO world.example_constraint(not_null, unique_key)
VALUES('some value', 'unique1');

INSERT INTO world.example_constraint(not_null, unique_key)
VALUES('another value', 'unique1');

UPDATE example_constraint set status = 'PENDING'
WHERE id = 1;
```

SQL Topics

- Data Manipulation Language (DML)
- Data types
- Data Definition Language (DDL)
- SQL Joins
- Subqueries
- Aggregation Functions
- Constraints
- **Indexes**
- Transactions
- Views
- Triggers
- Stored Procedures

Why apply Index

- Indexes speed up READ from database
- But WRITE to database becomes slower
- Indexes are added to tables with large data
- Primary key, Foreign Key, Unique Key columns are indexed by default

Example

```
CREATE TABLE `example_large_data` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `day` varchar(45) DEFAULT NULL,
  `date_time` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
);
```

- Load data from example_large_data.csv

Example

- create index on date_time column

```
CREATE INDEX date_time_index ON example_large_data
(date_time);
```

-
  ```
  CREATE INDEX date_time_index2 ON example_large_data (day,
  date_time);
  ```

- To delete the index

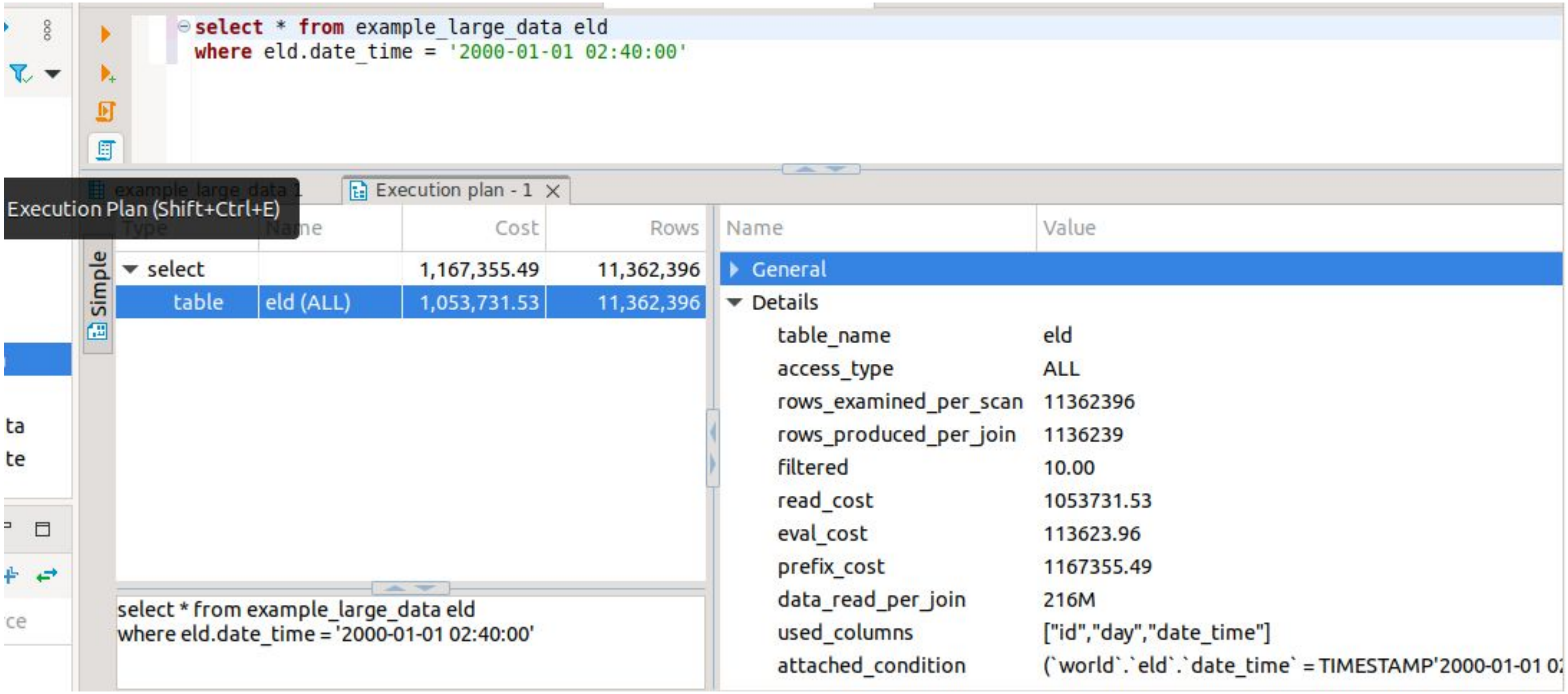
```
ALTER TABLE world.example_large_data DROP INDEX
date_time_index;
```

Explain Plan or Execution Plan

- Only applicable to SELECT queries
- Shows cost of query or how many records it compared to arrive at result
- List’s indexes available and/or used to arrive at the result

```
select *
from example_large_data
where date_time = '2000-01-01 02:40:00'
```

BEFORE Index



```
select * from example_large_data eld
where eld.date_time = '2000-01-01 02:40:00'
```

Type	Name	Cost	Rows
select		1,167,355.49	11,362,396
table	eld (ALL)	1,053,731.53	11,362,396

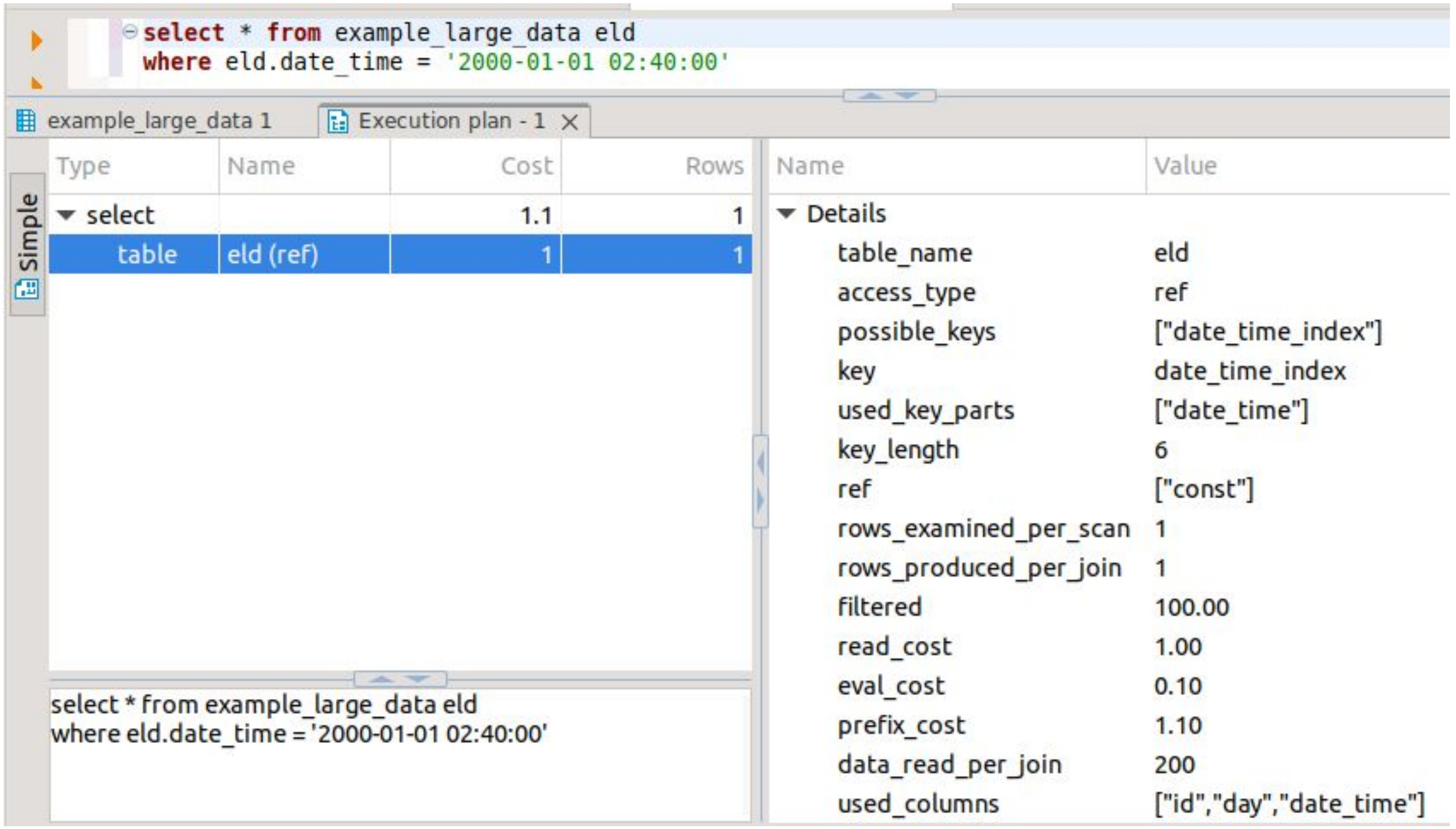
select * from example_large_data eld
where eld.date_time = '2000-01-01 02:40:00'

Name	Value
General	
Details	
table_name	eld
access_type	ALL
rows_examined_per_scan	11362396
rows_produced_per_join	1136239
filtered	10.00
read_cost	1053731.53
eval_cost	113623.96
prefix_cost	1167355.49
data_read_per_join	216M
used_columns	["id","day","date_time"]
attached_condition	(`world`.`eld`.`date_time` = TIMESTAMP'2000-01-01 0

lamzing

AFTER Index

- Record comparison is reduced drastically
- Mysql engine chooses an index own its own



```
select * from example_large_data eld
where eld.date_time = '2000-01-01 02:40:00'
```

Type	Name	Cost	Rows
select		1.1	1
table	eld (ref)	1	1

```
select * from example_large_data eld
where eld.date_time = '2000-01-01 02:40:00'
```

Name	Value
Details	
table_name	eld
access_type	ref
possible_keys	["date_time_index"]
key	date_time_index
used_key_parts	["date_time"]
key_length	6
ref	["const"]
rows_examined_per_scan	1
rows_produced_per_join	1
filtered	100.00
read_cost	1.00
eval_cost	0.10
prefix_cost	1.10
data_read_per_join	200
used_columns	["id","day","date_time"]

Multiple Index

```
CREATE INDEX date_time_index2 ON example_large_data
(day,date_time);
```

- Choose a specific index to use when multiple indexes are present

```
SELECT *
FROM example_large_data
FORCE INDEX (date_time_index2)
where date_time = '2000-01-01 02:40:00'
```

SQL Topics

- Data Manipulation Language (DML)
- Data types
- Data Definition Language (DDL)
- SQL Joins
- Subqueries
- Aggregation Functions
- Constraints
- Indexes
- **Transactions**
- Views
- Triggers
- Stored Procedures

ACID properties

- Atomicity
  - All operations completes or none
- Consistency
  - Ensure database is always in valid state. e.g. constraint violations are rolledback.
- Isolation
  - Multiple transactions / sessions do not overlap
- Durability
  - once data is committed, changes are permanent. next error or power failure will not affect committed data

Transaction

- A transaction is a logical unit of work that contains 1 or more SQL statements
- By default all the statements succeeds or fails
    - Programmatically managed transaction is possible
- Transaction Type
    - READ ONLY
    - READ-WRITE
- MySQL @@autocommit determines where the DB manages the transaction or it is programmatically managed
- Support ACID Properties

ROLLBACK

```
SET autocommit=0;
START TRANSACTION;
INSERT INTO city(Name, CountryCode, District, Population)
VALUES('TEST01', 'IND', 'Manipur', 10000);
INSERT INTO city(Name, CountryCode, District, Population)
VALUES('TEST02', 'IND', 'Manipur', 10000);
ROLLBACK;
INSERT INTO city(Name, CountryCode, District, Population)
VALUES('TEST03', 'IND', 'Manipur', 10000);
COMMIT;
```

SAVEPOINT

```
SET autocommit=0;
START TRANSACTION;

INSERT INTO city(Name, CountryCode, District, Population)
VALUES('TEST04', 'IND', 'Manipur', 10000);
SAVEPOINT savepoint1;
INSERT INTO city(Name, CountryCode, District, Population)
VALUES('TEST05', 'IND', 'Manipur', 10000);
ROLLBACK TO savepoint1;

COMMIT;
```

SQL Topics

- Data Manipulation Language (DML)
- Data types
- Data Definition Language (DDL)
- SQL Joins
- Subqueries
- Aggregation Functions
- Constraints
- Indexes
- Transactions
- **Views**
- Triggers
- Stored Procedures

Example

```sql
CREATE VIEW country_with_city_population AS
SELECT c.name as country, c2.name as city, c2.population as
city_population
FROM country c
INNER JOIN city c2 on c.code = c2.countryCode
```

Example

```
CREATE OR REPLACE VIEW country_with_city_population AS
SELECT c.name as country, c2.name as city, c2.population as
city_population
FROM country c
INNER JOIN city c2 on c.code = c2.countryCode
WHERE c.Code in
('IND', 'NPL', 'LKA', 'PAK', 'BGD', 'BTN', 'MDV')
```

SQL Topics

- Data Manipulation Language (DML)
- Data types
- Data Definition Language (DDL)
- SQL Joins
- Subqueries
- Aggregation Functions
- Constraints
- Indexes
- Transactions
- Views
- **Triggers**
- Stored Procedures

Triggers

- Action / Operation performed automatically in the database due to an event
- Events
  - INSERT, UPDATE, DELETE
  - CREATE table, UPATE table, DROP table

Example

- Create a table to store changes in city table

```
CREATE TABLE world.city_audit_table (
id bigint auto_increment NOT NULL PRIMARY KEY,
operation varchar(100) NOT NULL,
old_value varchar(200) NULL,
new_value varchar(200) NULL,
transaction_date timestamp NULL
);
```

Example

- Trigger when INSERT is performed to city table

```
delimiter $$
CREATE TRIGGER city_audit_trigger_insert
AFTER INSERT ON city
FOR EACH ROW
BEGIN
  INSERT INTO city_audit_table (operation, old_value, new_value,
transaction_date)
  VALUES ('INSERT', '', concat('name:',NEW.name,',
countrycode:',NEW.countrycode,', district:',NEW.district,',
population:',NEW.population), NOW()
  );
END;
$$
```

Example

```sql
INSERT INTO city (Name, CountryCode, District, Population)
VALUES('Moreh', 'IND', 'Manipur', 50000);
```

Example

- Trigger when UPDATE is performed to city table

```
delimiter $$
CREATE TRIGGER city_audit_trigger_update
AFTER UPDATE ON city
FOR EACH ROW
BEGIN
  INSERT INTO city_audit_table (operation, old_value, new_value,
transaction_date)
  VALUES ('UPDATE',
  concat('name:',OLD.name,', countrycode:',OLD.countrycode,',
district:',OLD.district,', population:',OLD.population),
  concat('name:',NEW.name,', countrycode:',NEW.countrycode,',
district:',NEW.district,', population:',NEW.population),
  NOW() );
END;
$$
```

Example

```sql
UPDATE city set Name = 'CCpur'
where District = 'Manipur'
```

Example

- Trigger when DELETE is performed to city table

```
delimiter $$
CREATE TRIGGER city_audit_trigger_delete
AFTER DELETE ON city
FOR EACH ROW
BEGIN
   INSERT INTO city_audit_table (operation, old_value, new_value, 
transaction_date)
   VALUES ('DELETE', concat('name:',OLD.name,', 
countrycode:',OLD.countrycode,', district:',OLD.district,', 
population:',OLD.population), '', NOW() );
END;
$$
```

Example

```
DELETE FROM city
where id = 4084
```

SQL Topics

- Data Manipulation Language (DML)
- Data types
- Data Definition Language (DDL)
- SQL Joins
- Subqueries
- Aggregation Functions
- Constraints
- Indexes
- Transactions
- Views
- Triggers
- **Stored Procedures**

Function vs Procedure

- Function
  - Returns a single value
  - Ideally Returns same value for same input (also known Deterministic)
  - Transaction do not support COMMIT, ROLLBACK

- Procedure
  - Does not have a return value directly
  - But can return via an OUT parameter or a select resultset
  - Transaction supported

Create Function

- Function will return a data

```
DELIMITER //
CREATE FUNCTION sum_two_numbers(a INT, b INT)
RETURNS INT NOT DETERMINISTIC NO SQL
BEGIN
  DECLARE result INT;
  SET result = a + b;
  RETURN result;
END;
//
SELECT sum_two_numbers(13,5) from dual
```

Create Procedure

```
DELIMITER //
CREATE PROCEDURE get_cities_by_country(country_code
VARCHAR(50))
BEGIN
  SELECT CountryCode, name, Population, District
  FROM city c
  WHERE CountryCode  = country_code;
END;
//
call get_cities_by_country('IND');
```

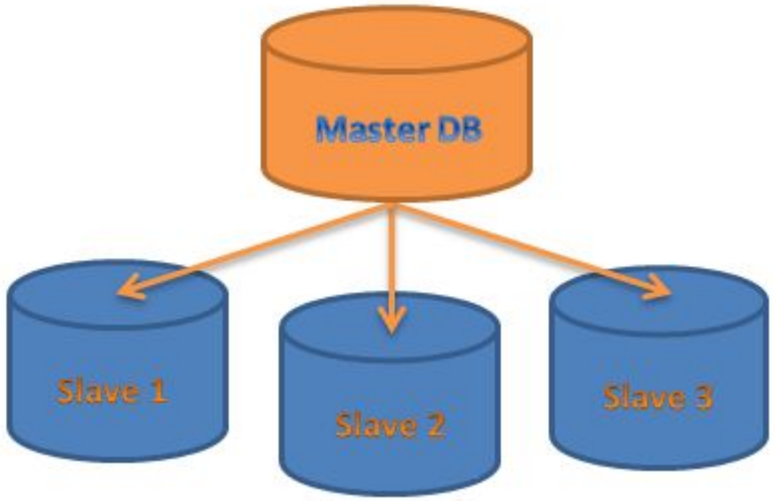
Create Procedure with OUT parameter

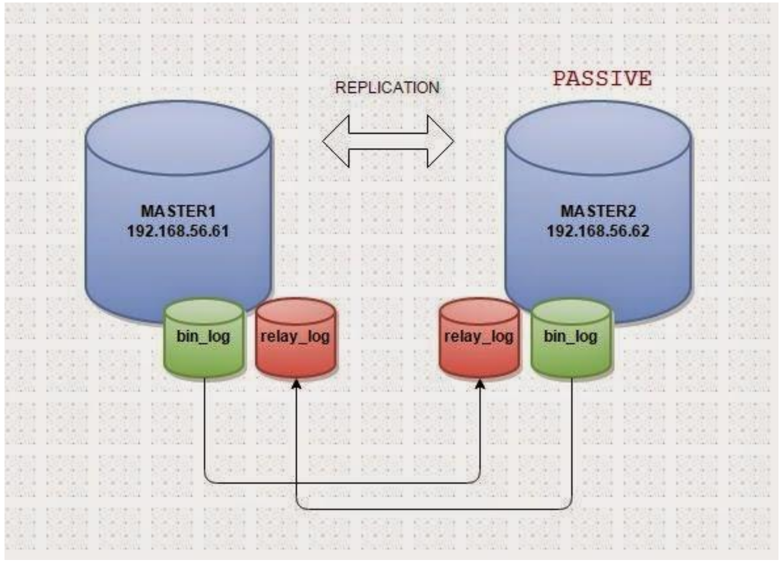
```
DELIMITER $$
CREATE PROCEDURE sum_city_population(
  IN country_code varchar(45),
    IN district varchar(45),
    OUT total_population int )
BEGIN
    SELECT SUM(c.population) INTO total_population
    FROM city c
    WHERE c.CountryCode = country_code
    and (c.District = district OR district = '');
END;
$$
```

MySQL Automatic Replication Method

- Master-Slave
  - Write only to master, read from all
  - Manual/Scripted Promotion of Slave

- Master-Master
  - Read-Write to both master
  - Synchronization challenges





https://mariadb.com/resources/blog/database-master-slave-replication-in-the-cloud/
http://msutic.blogspot.com/2015/02/mariadbmysql-master-master-replication.html

DEMO APPLICATION

Fix the SQL Query and make the application work!