Consider it very seriously

Database

(Database Language: Structured Query Language (SQL))

Spring, 2023

Jaewook Byun

Ph.D., Assistant Professor, Department of Software, Sejong University

jwbyun@sejong.ac.kr

https://sites.google.com/view/jack-dfpl/home

https://www.youtube.com/channel/UC988e-Y8nto0LXVae0aqaOQ

Schedule

Week	Title		
1	Course Overview and Environment Setting	Environment Setting and Revisiting Java	-
2	Introduction to Database		CH1
3	Relational Model	Relational Algebra	CH2
4	Database Language: SQL (DDL, DML, DQL, DCL)		CH3-5
5	Database Language: SQL (DDL, DML, DQL, DCL)		CH3-5
6	Database Language: SQL (DDL, DML, DQL, DCL)		CH3-5
7	Database Language: SQL (DDL, DML, DQL, DCL)		CH3-5
8	Midterm Examination		
9	Physical Database Design: Indexing		CH12
10	Physical Database Design: Indexing		CH12
11	Conceptual Database Design – E-R Data Model		CH6
12	Logical Database Design 1 – Schema Mapping		CH6
13	Logical Database Design 2 – Normalization		CH7
14	Query Processing and Optimization, or View (TBD)		CH13-14, CH3-5
15	Final Examination		

Subject to change

Calendar

- 수업일수는 요일별 15주 이상이며 수업 결손이 발생하지 않도록 진행

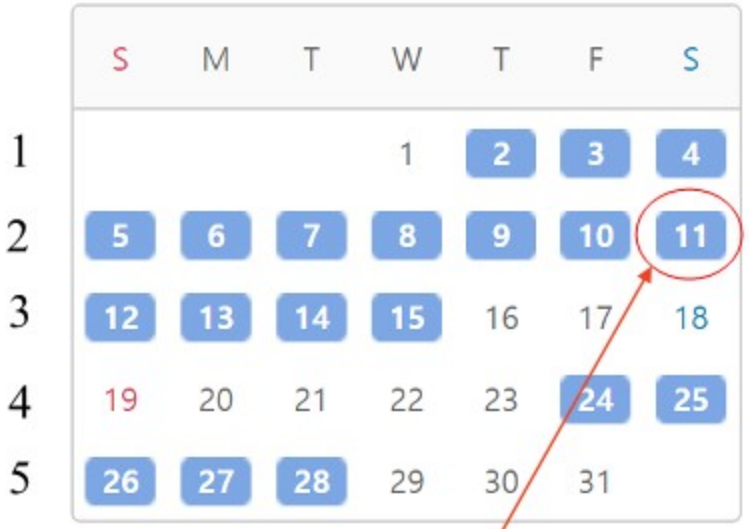
요 일 별 수업일수	월	화	수	목	금
	16일	15일	16일	16일	15일

나. 수업주차 : 한 주차는 목요일부터 수요일까지 임

수업주차	기간	수업주차	기간
1주차	03.02.(목) ~ 03.08.(수)	9주차	04.27.(목) ~ 05.03.(수)
2주차	03.09.(목) ~ 03.15.(수)	10주차	05.04.(목) ~ 05.10.(수)
3주차	03.16.(목) ~ 03.22.(수)	11주차	05.11.(목) ~ 05.17.(수)
4주차	03.23.(목) ~ 03.29.(수)	12주차	05.18.(목) ~ 05.24.(수)
5주차	03.30.(목) ~ 04.05.(수)	13주차	05.25.(목) ~ 05.31.(수)
6주차	04.06.(목) ~ 04.12.(수)	14주차	06.01.(목) ~ 06.07.(수)
7주차	04.13.(목) ~ 04.19.(수)	15주차	06.08.(목) ~ 06.14.(수)
8주차 (중간고사)	04.20.(목) ~ 04.26.(수)	16주차 (기말고사)	06.15.(목) ~ 06.21.(수)

Calendar

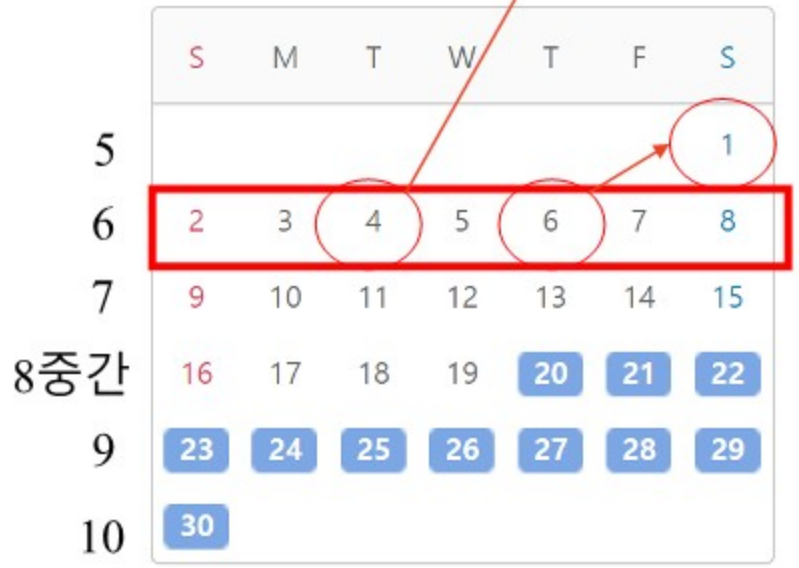
2023년 3월. March



	S	M	T	W	T	F	S
1				1	2	3	4
2	5	6	7	8	9	10	11
3	12	13	14	15	16	17	18
4	19	20	21	22	23	24	25
5	26	27	28	29	30	31	

- 2(목) 1학기 개강
- 3(금) - 8(수) 수강신청 과목 확인 및 변경
- 6(월) - 15(수) 교직신청
- 24(금) - 28(화) 수강신청과목 철회

2023년 4월. April



	S	M	T	W	T	F	S
5							1
6	2	3	4	5	6	7	8
7	9	10	11	12	13	14	15
8중간	16	17	18	19	20	21	22
9	23	24	25	26	27	28	29
10	30						

- 20(목) - 26(수) 1학기 중간고사
- 27(목) - 5.1(월) 1학기 중간고사 성적 입력

IEEE ICDE 2023, Anaheim, California, US

Calendar

2023년 5월. May

	S	M	T	W	T	F	S
10		1	2	3	4	5	6
11	7	8	9	10	11	12	13
12	14	15	16	17	18	19	20
13	21	22	23	24	25	26	27
14	28	29	30	31			

- 2(화) - 7(일) 1학기 중간고사 성적 열람 및 정정
- 4(목) - 30(화) 복수·부전공, 연계융합전공 신청
- 5(금) 창립 83주년 기념휴일 (창립일 : 1940. 5. 20)
- 29(월) - 31(수) 하계 계절학기 수강신청

Calendar

2023년 6월. June

	S	M	T	W	T	F	S
14					1	2	3
	4	5	6	7	8	9	10
15기말	11	12	13	14	15	16	17
	18	19	20	21	22	23	24
	25	26	27	28	29	30	

- 9(금) - 26(월) 1학기 강의평가
- 15(목) - 21(수) 1학기 기말고사 및 수업결손 보충
- 22(목) - 26(월) 1학기 기말고사 성적 입력
- 22(목) 하계방학 시작 및 계절학기 개강
- 27(화) - 7.3(월) 1학기 기말고사 성적 열람 및 정정

Calendar

2023년 7월. July

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

- 4(화) - 5(수)
- 24(월) - 30(일)

1학기 기말고사 성적마감

2학기 복학, 휴학 신청

Table of Contents

Introduction

- SQL
  - Structured Query Language
  - The standard query language for relational database management system (RDBMs)

  - History
    - 1970 – E. F. Codd develops relational database concept
    - 1986 – ANSI SQL standard released
    - 1989,1992,1999,2003,2006,2008 – Major ANSI standard updates

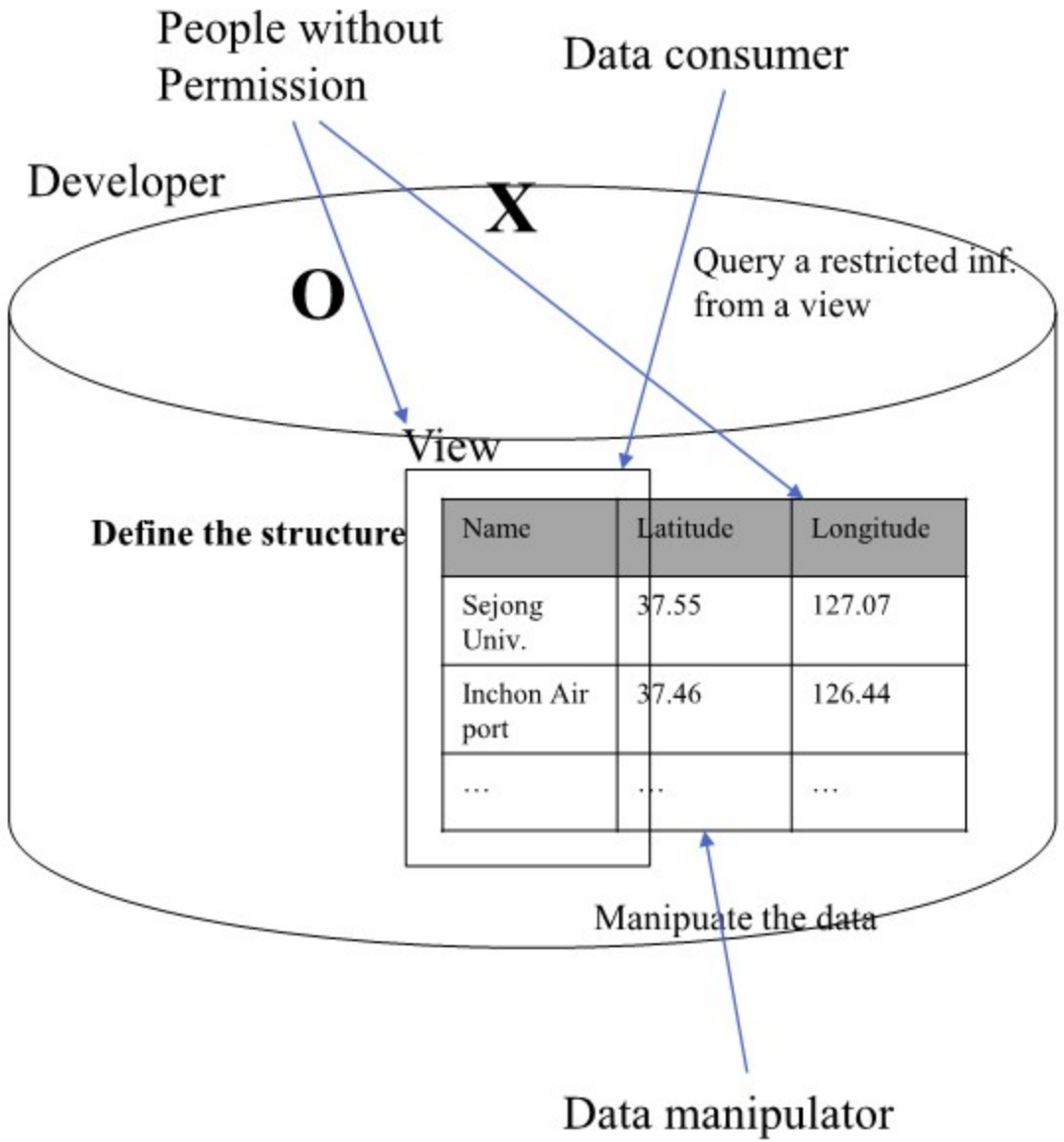
- Students will learn SQL with MariaDB, an open-source relational database
  - Learn and practice SQL based on MariaDB documentation
    - https://mariadb.com/kb/en/documentation/
  - Bridge the gap between relational model/algebra and SQL

Data Definition Language: CREATE DATABASE

- SQL
  - **Data definition**
  - Data manipulation
  - Data query
  - Data control

Example: geo data

Sejong Univ., 37.55, 127.07
Incheon Airport, 37.46, 126.44
…



Data Definition Language: CREATE DATABASE

- CREATE DATABASE
  - https://mariadb.com/kb/en/create-database/

```
CREATE [OR REPLACE] {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification] ...

create_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | COMMENT [=] 'comment'
```

- Example
  - **CREATE DATABASE db;**

```
MariaDB [(none)]> CREATE DATABASE db;
Query OK, 1 row affected (0.001 sec)
```

Data Definition Language: CREATE DATABASE

- Example
  - CREATE OR REPLACE DATABASE db;
    - Equals to

```
DROP DATABASE IF EXISTS db_name;
CREATE DATABASE db_name ...;
```

```
MariaDB [(none)]> CREATE OR REPLACE DATABASE db;
Query OK, 1 row affected (0.001 sec)
```

Data Definition Language: CREATE DATABASE

- Example
  - CREATE DATABASE db; (When exists)

```
MariaDB [(none)]> CREATE DATABASE db;
ERROR 1007 (HY000): Can't create database 'db'; database exists
```

  - CREATE DATABASE IF NOT EXISTS db;

```
MariaDB [(none)]> CREATE DATABASE IF NOT EXISTS db;
Query OK, 0 rows affected, 1 warning (0.000 sec)
```

  - SHOW WARNINGS;

```
MariaDB [(none)]> SHOW WARNINGS;
+-------+------+-----------------------------------------+
| Level | Code | Message                                 |
+-------+------+-----------------------------------------+
| Note  | 1007 | Can't create database 'db'; database exists |
+-------+------+-----------------------------------------+
1 row in set (0.000 sec)
```

Data Definition Language: CREATE DATABASE

```
CREATE [OR REPLACE] {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification] ...

create_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | COMMENT [=] 'comment'
```

- Example
  - CREATE OR REPLACE DATABASE db
    - CHARACTER SET = latin1
    - COLLATE = latin1_german2_ci ; ????

```
MariaDB [(none)]> CREATE OR REPLACE DATABASE db
    -> CHARACTER SET = latin1
    -> COLLATE = latin1_german2_ci;
Query OK, 2 rows affected (0.002 sec)
```

Data Definition Language: CREATE DATABASE

- Character set and Collation
  - https://mariadb.com/kb/en/supported-character-sets-and-collations/
  - Reason
    - A size of bytes to represent a character
      - euckr requires 2 bytes
      - ujis requires 3 bytes
      - Some character sets requires 4 bytes
    - How to sort a text
      - Next page

Charset	Description	Default collation	Maxlen
big5	Big5 Traditional Chinese	big5_chinese_ci	2
dec8	DEC West European	dec8_swedish_ci	1
cp850	DOS West European	cp850_general_ci	1
hp8	HP West European	hp8_english_ci	1
koi8r	KOI8-R Relcom Russian	koi8r_general_ci	1
latin1	cp1252 West European	latin1_swedish_ci	1
latin2	ISO 8859-2 Central European	latin2_general_ci	1
swe7	7bit Swedish	swe7_swedish_ci	1
ascii	US ASCII	ascii_general_ci	1
ujis	EUC-JP Japanese	ujis_japanese_ci	3
sjis	Shift-JIS Japanese	sjis_japanese_ci	2
hebrew	ISO 8859-8 Hebrew	hebrew_general_ci	1
tis620	TIS620 Thai	tis620_thai_ci	1
euckr	EUC-KR Korean	euckr_korean_ci	2
koi8u	KOI8-U Ukrainian	koi8u_general_ci	1
gb2312	GB2312 Simplified Chinese	gb2312_chinese_ci	2
greek	ISO 8859-7 Greek	greek_general_ci	1
cp1250	Windows Central European	cp1250_general_ci	1
gbk	GBK Simplified Chinese	gbk_chinese_ci	2

. . . . . .

Data Definition Language: CREATE DATABASE

- Character set and Collation
  - Order
    - A → Z

```
 SELECT "A" < "Z";
+-----------+
| "A" < "Z" |
+-----------+
|         1 |
+-----------+
```

    - A = a, case-insensitive

```
SELECT "A" < "a", "A" = "a";
+-----------+-----------+
| "A" < "a" | "A" = "a" |
+-----------+-----------+
|         0 |         1 |
+-----------+-----------+
```

- Example:
  - German by default

```
SELECT 'Mueller' = 'Müller';
+-----------------------+
| 'Müller' = 'Mueller' |
+-----------------------+
|                     0 |
+-----------------------+
```

  - latin1_german2_ci

```
SET collation_connection = latin1_german2_ci;

SELECT 'Mueller' = 'Müller';
+-----------------------+
| 'Mueller' = 'Müller'  |
+-----------------------+
|                     1 |
+-----------------------+
```

Data Definition Language: CREATE DATABASE

- Example
  - CREATE OR REPLACE DATABASE db
    - CHARACTER SET = ‘euckr’
    - COLLATE = ‘euckr_Korean_ci’ ;

```
MariaDB [(none)]> CREATE OR REPLACE DATABASE db
    -> CHARACTER SET = 'euckr'
    -> COLLATE = 'euckr_Korean_ci';
Query OK, 2 rows affected (0.002 sec)
```

  - CREATE OR REPLACE DATABASE db
    - COMMENT = ‘Welcome to database’;

```
MariaDB [(none)]> CREATE OR REPLACE DATABASE db
    -> COMMENT = "Welcome to database";
Query OK, 2 rows affected (0.002 sec)
```

Data Definition Language: CREATE DATABASE

- Example
  - SHOW DATABASES;

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| db                 |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.001 sec)
```

Data Definition Language: DROP DATABASE

- DROP DATABASE
  - https://mariadb.com/kb/en/drop-database/

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

- Example
  -

```
MariaDB [(none)]>
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.001 sec)
```

Data Definition Language: DROP DATABASE

- Example
  - DROP DATABASE db; (One more time)

```
MariaDB [(none)]> DROP DATABASE db;
ERROR 1008 (HY000): Can't drop database 'db'; database doesn't exist
```

  - DROP DATABASE IF EXISTS db;

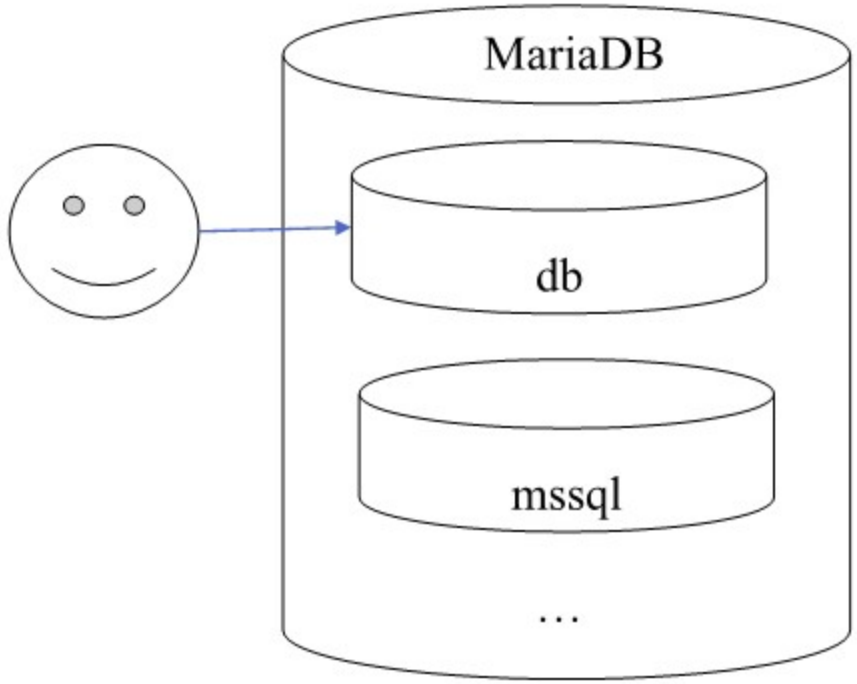
```
MariaDB [(none)]> DROP DATABASE IF EXISTS db;
Query OK, 0 rows affected, 1 warning (0.000 sec)

MariaDB [(none)]> SHOW WARNINGS;
+-------+------+----------------------------------------------+
| Level | Code | Message                                      |
+-------+------+----------------------------------------------+
| Note  | 1008 | Can't drop database 'db'; database doesn't exist |
+-------+------+----------------------------------------------+
1 row in set (0.000 sec)
```

Data Definition Language: USE DATABASE

- USE
  - https://mariadb.com/kb/en/use/

```
USE db_name
```

- Example
  - USE db ;




```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| db                 |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.001 sec)

MariaDB [(none)]> USE db;
Database changed
MariaDB [db]>
```

Data Definition Language: CREATE TABLE

- CREATE TABLE
  - https://mariadb.com/kb/en/create-table/

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...) [table_options    ]... [partition_options]
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)] [table_options   ]... [partition_options]
    select_statement
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
   { LIKE old_table_name | (LIKE old_table_name) }

select_statement:
    [IGNORE | REPLACE] [AS] SELECT ...   (Some legal select statement)
```

Data Definition Language: CREATE TABLE

- CREATE TABLE
  - https://mariadb.com/kb/en/create-table/

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...) [table_options    ]... [partition_options]
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)] [table_options    ]... [partition_options]
    select_statement
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
   { LIKE old_table_name | (LIKE old_table_name) }

select_statement:
    [IGNORE | REPLACE] [AS] SELECT ...   (Some legal select statement)
```

Data Definition Language: CREATE TABLE

- CREATE TABLE with create_definition

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...) [table_options    ]... [partition_options]
```

```
create_definition:
  { col_name column_definition | index_definition | period_definition | CHECK (expr) }

column_definition:
  data_type
    [NOT NULL | NULL] [DEFAULT default_value | (expression)]
    [AUTO_INCREMENT] [ZEROFILL] [UNIQUE [KEY] | [PRIMARY] KEY]
    [INVISIBLE] [{WITH|WITHOUT} SYSTEM VERSIONING]
    [COMMENT 'string'] [REF_SYSTEM_ID = value]
    [COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}]
    [reference_definition]
  | data_type [GENERATED ALWAYS]
  AS { { ROW {START|END} } | { (expression) [VIRTUAL | PERSISTENT | STORED] } }
      [UNIQUE [KEY]] [COMMENT 'string']

constraint_definition:
   CONSTRAINT [constraint_name] CHECK (expression)
```

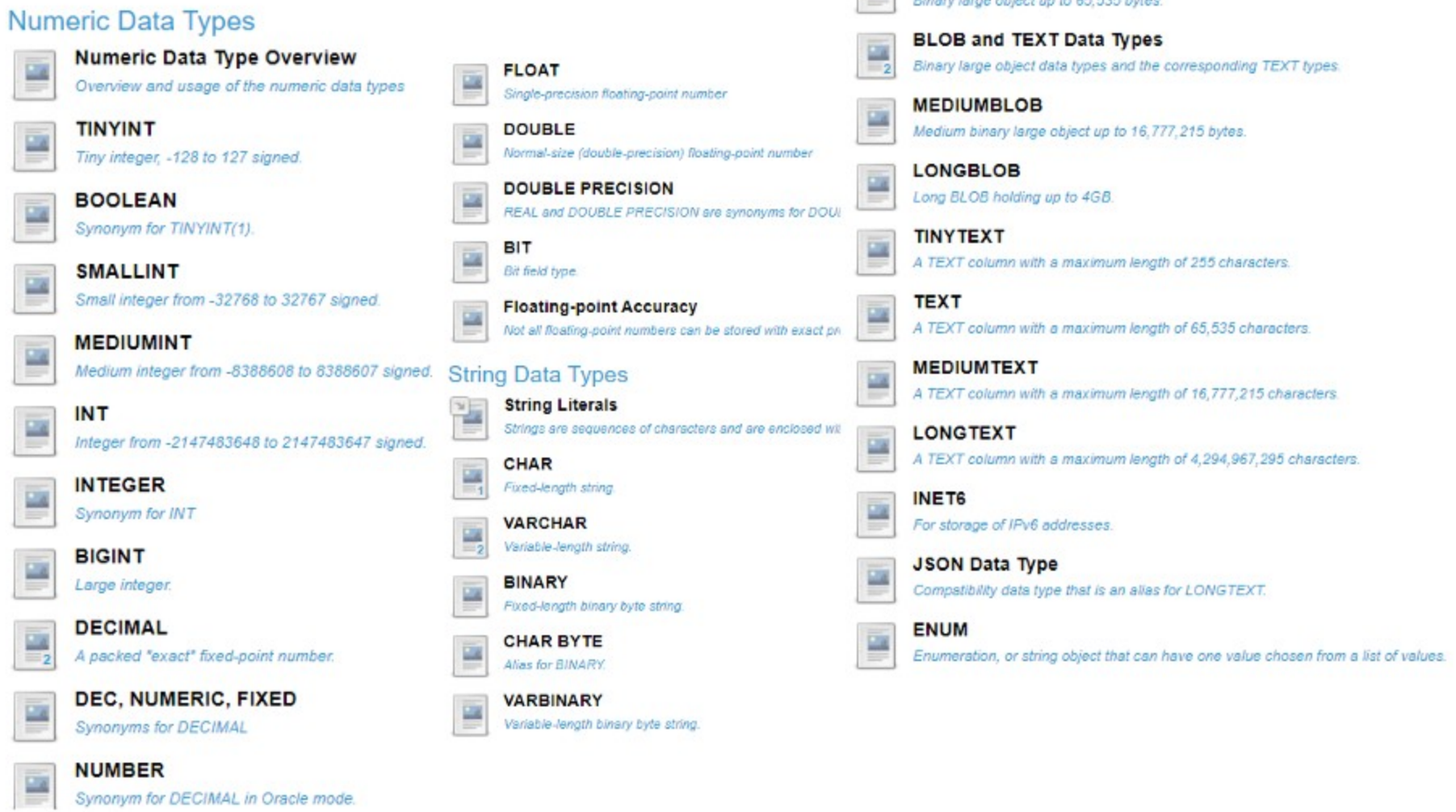
Data Definition Language: CREATE TABLE

- CREATE TABLE with create_definition

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...) [table_options ...] [partition_options]

create_definition:
  { col_name column_definition | index_definition | special_definition | CHECK (expr) }

column_definition:
  data_type
```

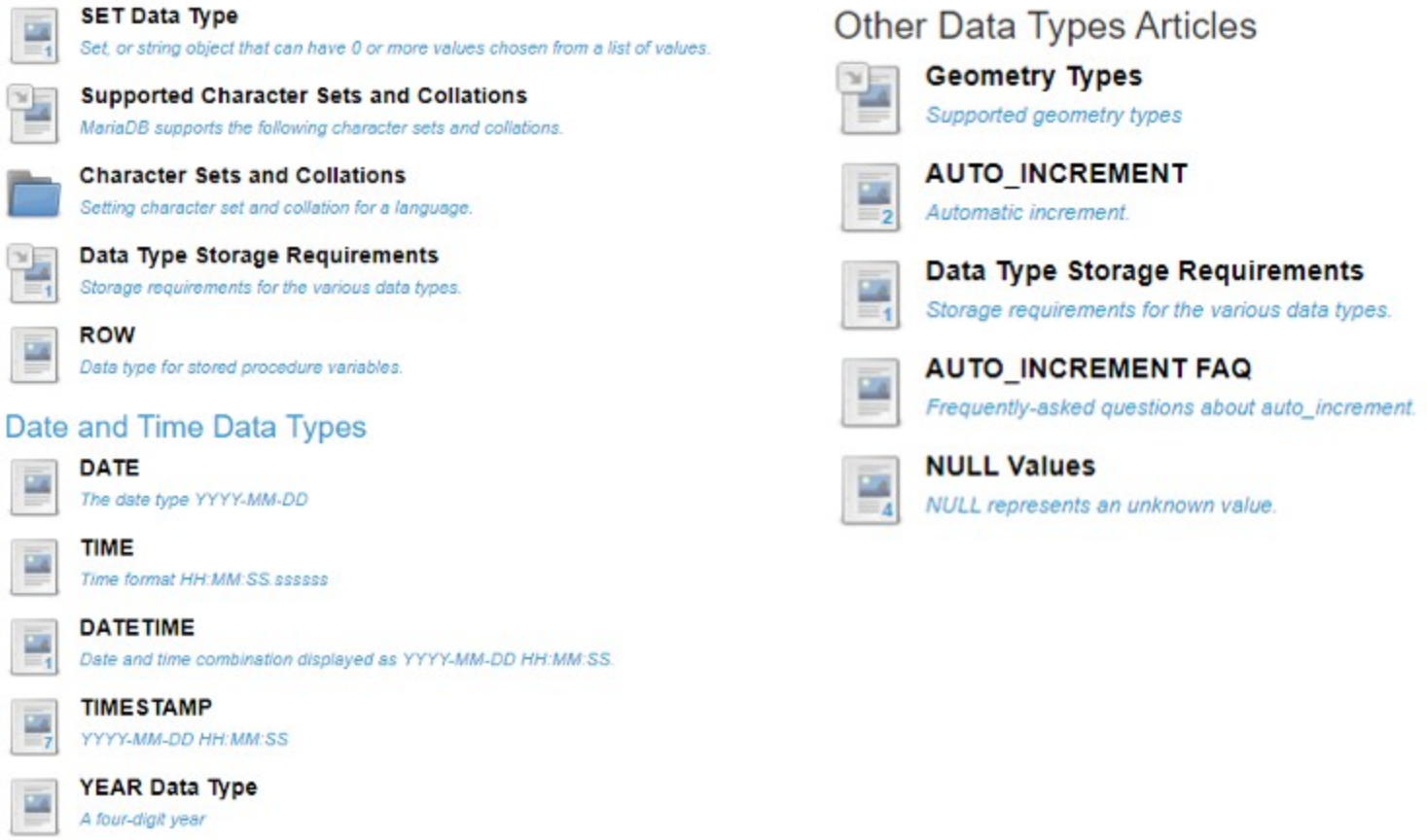
Data Definition Language: CREATE TABLE

- CREATE TABLE with create_definition
  - data type



Data Definition Language: CREATE TABLE

- CREATE TABLE with create_definition
  - data type



Data Definition Language: CREATE TABLE

- CREATE TABLE with create_definition with VARCHAR

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...) [table_options    ]... [partition_options]
```

- Create a table for customer2 (customer, customer_street, customer_city, longitude, latitude, last_update)
  - with VARCHAR(50), a variable-length string

```
[NATIONAL] VARCHAR(M) [CHARACTER SET charset_name] [COLLATE collation_name]
```

```
MariaDB [db]> CREATE TABLE customer2 (
    -> customer VARCHAR(50),
    -> customer_street VARCHAR(50),
    -> customer_city VARCHAR(50),
    -> latitude VARCHAR(50),
    -> longitude VARCHAR(50),
    -> last_update VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.015 sec)
```

Data Definition Language: CREATE TABLE

- CREATE TABLE with create_definition with VARCHAR
  - https://mariadb.com/kb/en/describe/
  - DESCRIBE table_name;
    - DESCRIBE customer2;

```
MariaDB [db]> DESCRIBE customer2;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| customer        | varchar(50) | YES  |     | NULL    |       |
| customer_street | varchar(50) | YES  |     | NULL    |       |
| customer_city   | varchar(50) | YES  |     | NULL    |       |
| latitude        | varchar(50) | YES  |     | NULL    |       |
| longitude       | varchar(50) | YES  |     | NULL    |       |
| last_update     | varchar(50) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
6 rows in set (0.022 sec)
```

Data Definition Language: CREATE TABLE

- Select appropriate data type
  - Regarding time complexity
    - e.g., Show me good stores where its valid date is in 1st day of month
      - Get all records
      - Split the validDate with '-'
      - Evaluate the first element of splitted string with '1'
  - Regarding storage requirements
    - e.g.,
      - '2021-02-01' with utf8 (3bytes) → 30 bytes
      - 1612105200 BIGINT → 8 bytes
      - for 100 million records?

last_update
"2021-02-**01**"
"2021-02-01"
2021-02-01
2021-02-02
2021-02-02
2021-02-03
2021-02-03
2021-02-04
2021-02-04
2021-02-04
2021-02-04

Data Definition Language: CREATE TABLE

- Data Types
  - https://mariadb.com/kb/en/data-types/
  - Numeric Data Types
    - BOOLEAN, BIGINT, DOUBLE, BIT, etc.
  - String Data Types
    - VARCHAR, BINARY, BLOB, JSON Data Type, SET, ENUM, etc.
  - Date and Time data Types
    - DATE, TIMESTAMP
  - Other data types
    - Geometry Types

Data Definition Language: CREATE TABLE

- CREATE TABLE with create_definition with VARCHAR
  - Enhanced Customer2 with various data types
  - CREATE OR REPLACE TABLE customer2 (create_definition,...)

- Practice
  - Change data types of the following field:
    - latitude DOUBLE: https://mariadb.com/kb/en/double/
    - longitude DOUBLE: https://mariadb.com/kb/en/double/
    - last_update DATE: https://mariadb.com/kb/en/date/

How about Geometry POINT?
- POINT
  - https://mariadb.com/kb/en/geometry-types/

customer	customer_street	customer_city	latitude	longitude	last_update
Adams	Spring	Pittsfield	97.1	34.59	2021-02-01
Brooks	Senator	Brooklyn	81.4	30.19	2021-02-01

Data Definition Language: CREATE TABLE

- Practice
  - Change data types of the following field:
    - latitude DOUBLE: https://mariadb.com/kb/en/double/
    - longitude DOUBLE: https://mariadb.com/kb/en/double/
    - last_update DATE: https://mariadb.com/kb/en/date/

```
MariaDB [db]> CREATE OR REPLACE TABLE customer2 (
    -> customer VARCHAR(50),
    -> customer_street VARCHAR(50),
    -> customer_city VARCHAR(50),
    -> latitude DOUBLE,
    -> longitude DOUBLE,
    -> last_update DATE
    -> );
Query OK, 0 rows affected (0.023 sec)

MariaDB [db]> DESCRIBE customer2;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| customer        | varchar(50) | YES  |     | NULL    |       |
| customer_street | varchar(50) | YES  |     | NULL    |       |
| customer_city   | varchar(50) | YES  |     | NULL    |       |
| latitude        | double      | YES  |     | NULL    |       |
| longitude       | double      | YES  |     | NULL    |       |
| last_update     | date        | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
6 rows in set (0.022 sec)
```

Data Definition Language: CREATE TABLE

- CREATE TABLE IF NOT EXISTS table_name
  - CREATE TABLE IF NOT EXISTS customer2

```
MariaDB [db]> CREATE TABLE IF NOT EXISTS customer2 (customer VARCHAR(50)
, customer_street VARCHAR(50), customer_city VARCHAR(50), latitude DOUBL
E, longitude DOUBLE, last_update DATE);
Query OK, 0 rows affected, 1 warning (0.000 sec)

MariaDB [db]> SHOW WARNINGS;
+-------+------+-------------------------------------+
| Level | Code | Message                             |
+-------+------+-------------------------------------+
| Note  | 1050 | Table 'customer2' already exists |
+-------+------+-------------------------------------+
1 row in set (0.000 sec)
```

Data Definition Language: DROP TABLE

- DROP TABLE
  - https://mariadb.com/kb/en/drop-table/

```
DROP [TEMPORARY] TABLE [IF EXISTS] [/*COMMENT TO SAVE*/]
    tbl_name [, tbl_name] ...
    [WAIT n|NOWAIT]
    [RESTRICT | CASCADE]
```

-

```
MariaDB [db]>
Query OK, 0 rows affected (0.009 sec)

MariaDB [db]> DESCRIBE customer2;
ERROR 1146 (42S02): Table 'db.customer2' doesn't exist
```

Data Definition Language: DROP TABLE

- DROP TABLE
  - DROP TABLE customer2 (One more time)

```
MariaDB [db]> DROP TABLE customer2;
ERROR 1051 (42S02): Unknown table 'db.customer2'
```

  - DROP TABLE IF EXISTS customer2;

```
MariaDB [db]> DROP TABLE IF EXISTS customer2;
Query OK, 0 rows affected, 1 warning (0.002 sec)

MariaDB [db]> SHOW WARNINGS;
+-------+------+-----------------------------+
| Level | Code | Message                     |
+-------+------+-----------------------------+
| Note  | 1051 | Unknown table 'db.customer2' |
+-------+------+-----------------------------+
1 row in set (0.000 sec)
```

Data Definition Language: ALTER TABLE

- ALTER TABLE
  - https://mariadb.com/kb/en/alter-table/
  - …

```
ALTER [ONLINE] [IGNORE] TABLE [IF EXISTS] tbl_name
    [WAIT n | NOWAIT]
    alter_specification [, alter_specification] ...

alter_specification:
    table_option ...
  | ADD [COLUMN] [IF NOT EXISTS] col_name column_definition
        [FIRST | AFTER col_name ]
  |
 ADD [COLUMN] [IF NOT EXISTS] (col_name column_definition,...
  | ADD {INDEX|KEY} [IF NOT EXISTS] [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]] PRIMARY KEY
        [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        UNIQUE [INDEX|KEY] [index_name]
        [index_type] (index_col_name,...) [index_option] ...
  | ADD FULLTEXT [INDEX|KEY] [index_name]
        (index_col_name,...) [index_option] ...
  | ADD SPATIAL [INDEX|KEY] [index_name]
        (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
        FOREIGN KEY [IF NOT EXISTS] [index_name] (index_col_n
        reference_definition
  |
 ADD PERIOD FOR SYSTEM_TIME (start_column_name, end_column_na
  |
 ALTER [COLUMN] col_name SET DEFAULT literal | (expression)
  | ALTER [COLUMN] col_name DROP DEFAULT
  |
 CHANGE [COLUMN] [IF EXISTS] old_col_name new_col_name column_
        [FIRST|AFTER col_name]
  | MODIFY [COLUMN] [IF EXISTS] col_name column_definition
        [FIRST | AFTER col_name]
  | DROP [COLUMN] [IF EXISTS] col_name [RESTRICT|CASCADE]
  | DROP PRIMARY KEY
  | DROP {INDEX|KEY} [IF EXISTS] index_name
  | DROP FOREIGN KEY [IF EXISTS] fk_symbol
  | DROP CONSTRAINT [IF EXISTS] constraint_name
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO] new_tbl_name
  | ORDER BY col_name [, col_name] ...
  | RENAME COLUMN old_col_name TO new_col_name
  | RENAME {INDEX|KEY} old_index_name TO new_index_name
  |
CONVERT TO CHARACTER SET charset_name [COLLATE collation_nam
  | [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | ALGORITHM [=] {DEFAULT|INPLACE|COPY|NOCOPY|INSTANT}
  | LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
  | FORCE
  | partition_options
  | ADD PARTITION (partition_definition)
  | DROP PARTITION partition_names
  | COALESCE PARTITION number
  |
REORGANIZE PARTITION [partition_names INTO (partition_defini
  | ANALYZE PARTITION partition_names
  | CHECK PARTITION partition_names
  | OPTIMIZE PARTITION partition_names
  | REBUILD PARTITION partition_names
  | REPAIR PARTITION partition_names
  | EXCHANGE PARTITION partition_name WITH TABLE tbl_name
  | REMOVE PARTITIONING
  | ADD SYSTEM VERSIONING
  | DROP SYSTEM VERSIONING
```

```
index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH | RTREE}

index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'
  | CLUSTERING={YES| NO}

table_options:
    table_option [[,] table_option] ...
```

Data Definition Language: ALTER TABLE

- ALTER TABLE
  - Column
    - Add, Drop, Modify, Change, Rename
  - Primary Key
    - Add, Drop
  - Foreign Key
    - Add, Drop
  - (Unique|Fulltext|Spatial) Index
    - Rename, Add, Drop
  - ...

Data Definition Language: ALTER TABLE

- Question
  - Other way to change the schema instead of CREATE OR REPLACE?
  - Change data types of the following field:
    - geopoint POINT: https://mariadb.com/kb/en/geometry-types/
    - last_update DATE: https://mariadb.com/kb/en/date/

```
MariaDB [db]> CREATE OR REPLACE TABLE customer2 (customer VARCHAR(50), customer_street
VARCHAR(50), customer_city VARCHAR(50), geopoint POINT, last_update DATE);
Query OK, 0 rows affected (0.014 sec)

MariaDB [db]> DESCRIBE customer2;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| customer        | varchar(50) | YES  |     | NULL    |       |
| customer_street | varchar(50) | YES  |     | NULL    |       |
| customer_city   | varchar(50) | YES  |     | NULL    |       |
| geopoint        | point       | YES  |     | NULL    |       |
| last_update     | date        | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
5 rows in set (0.017 sec)
```

Data Definition Language: ALTER TABLE

- ALTER TABLE

CREATE OR REPLACE TABLE customer2 (customer VARCHAR(50), customer_street VARCHAR(50), customer_city VARCHAR(50), latitude DOUBLE, longitude DOUBLE, last_update DATE);

customer	customer_street	customer_city	latitude	longitude	last_update
Adams	Spring	Pittsfield	97.1	34.59	2021-02-01
Brooks	Senator	Brooklyn	81.4	30.19	2021-02-01

- Drop latitude, longitude to Insert geopoint

customer	customer_street	customer_city	last_update
Adams	Spring	Pittsfield	2021-02-01
Brooks	Senator	Brooklyn	2021-02-01

- Rename customer to customer_name for avoiding ambiguous

customer_name	customer_street	customer_city	last_update
Adams	Spring	Pittsfield	2021-02-01
Brooks	Senator	Brooklyn	2021-02-01

- Add geopoint instead of latitude, longitude

customer	customer_street	customer_city	geopoint	last_update
Adams	Spring	Pittsfield	97.1,34.59	2021-02-01
Brooks	Senator	Brooklyn	81.4,30.19	2021-02-01

Data Definition Language: ALTER TABLE

- ALTER TABLE
  - Column
    - Drop: latitude and longitude

```
ALTER [ONLINE] [IGNORE] TABLE [IF EXISTS] tbl_name
    [WAIT n | NOWAIT]
    alter_specification [, alter_specification] ...
```

```
alter_specification:
    table_option ...
  | DROP [COLUMN] [IF EXISTS] col_name [RESTRICT|CASCADE]
```

-

```
MariaDB [db]>
Query OK, 0 rows affected (0.009 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [db]> DESCRIBE customer2;
+-----------------+------------+------+-----+---------+-------+
| Field           | Type       | Null | Key | Default | Extra |
+-----------------+------------+------+-----+---------+-------+
| customer        | varchar(50) | YES |     | NULL    |       |
| customer_street | varchar(50) | YES |     | NULL    |       |
| customer_city   | varchar(50) | YES |     | NULL    |       |
| last_update     | date       | YES |     | NULL    |       |
+-----------------+------------+------+-----+---------+-------+
4 rows in set (0.022 sec)
```

Data Definition Language: ALTER TABLE

- ALTER TABLE
  - Column
    - Rename customer to customer_name for avoiding ambiguity

MariaDB starting with 10.5.2

From MariaDB 10.5.2, it is possible to rename a column using the RENAME COLUMN syntax, for example:

```
ALTER TABLE t1 RENAME COLUMN c_old TO c_new;
```

-

```
MariaDB [db]>
Query OK, 0 rows affected (0.004 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [db]> DESCRIBE customer2;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| customer_name   | varchar(50) | YES  |     | NULL    |       |
| customer_street | varchar(50) | YES  |     | NULL    |       |
| customer_city   | varchar(50) | YES  |     | NULL    |       |
| last_update     | date        | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.021 sec)
```

Data Definition Language: ALTER TABLE

- ALTER TABLE
  - Column
    - Add geopoint POINT instead of latitude, longitude

```
... ADD COLUMN [IF NOT EXISTS]  (col_name column_definition,...)
```

-

```
MariaDB [db]>
Query OK, 0 rows affected (0.003 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [db]>
MariaDB [db]> DESCRIBE customer2;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| customer_name   | varchar(50) | YES  |     | NULL    |       |
| customer_street | varchar(50) | YES  |     | NULL    |       |
| customer_city   | varchar(50) | YES  |     | NULL    |       |
| last_update     | date        | YES  |     | NULL    |       |
| geopoint        | point       | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
5 rows in set (0.022 sec)
```

Data Definition Language: ALTER TABLE

- ALTER TABLE
  - Column
    - Change datatype of type from VARCHAR(50) to VARCHAR(100);

```
| MODIFY [COLUMN] [IF EXISTS] col_name column_definition
      [FIRST | AFTER col_name]
```

Try VARCHAR(50), VARCHAR(10), VARCHAR(5), INTEGER, …
It works if any data exists

-

```
MariaDB [db]>
Query OK, 0 rows affected (0.033 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [db]> DESCRIBE customer2;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| customer_name   | varchar(50)  | YES  |     | NULL    |       |
| customer_street | varchar(100) | YES  |     | NULL    |       |
| customer_city   | varchar(50)  | YES  |     | NULL    |       |
| last_update     | date         | YES  |     | NULL    |       |
| geopoint        | point        | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
5 rows in set (0.023 sec)
```

Data Definition Language: Integrity Constraints

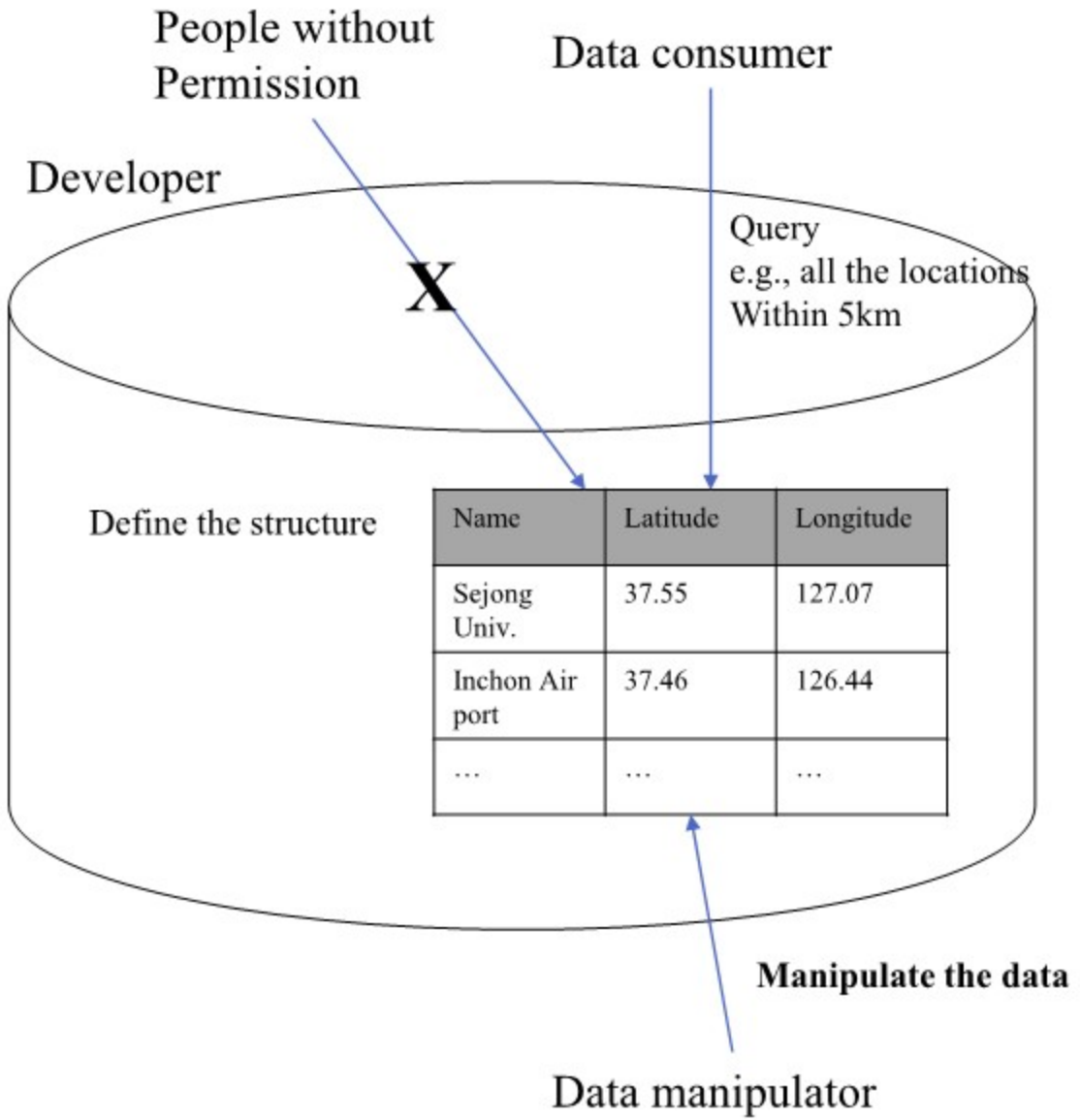
- Integrity constraints
  - guarantee to keep data consistency
  - e.g.,
    - account balance cannot be null
    - different accounts cannot have a same account number
    - account number in depositor relation should be matched with that in account relation
    - A salary per hour should exceed $6.00

Data Manipulation Language: INSERT INTO TABLE

- SQL
  - Data definition
  - Data manipulation
  - Data query
  - Data control

Example: geo data

Sejong Univ., 37.55, 127.07
Incheon Airport, 37.46, 126.44
…



Data Manipulation Language: INSERT INTO TABLE

- INSERT INTO table_name
  - https://mariadb.com/kb/en/insert/

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
 [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
 {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
 [ ON DUPLICATE KEY UPDATE
   col=expr
     [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

Or:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)]
    SET col={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

Or:

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

Data Manipulation Language: INSERT INTO TABLE

- Go over: Data Definition Language
  - Practice: Create ‘grade' table

id INTEGER	name VARCHAR(10)	attendance DOUBLE	midterm DOUBLE	assignment DOUBLE	final DOUBLE
615453	J. B.	10	30	30	30

Data Manipulation Language: INSERT INTO TABLE

- Go over: Data Definition Language

id INTEGER	name VARCHAR(10)	attendance DOUBLE	midterm DOUBLE	assignment DOUBLE	final DOUBLE
615453	J. B.	10	30	30	30

```
MariaDB [db]> CREATE TABLE grade ( id INTEGER, name VARCHAR(10), attendance DOUBLE, midterm DOUBLE, assignment DOUBLE,
 final DOUBLE);
Query OK, 0 rows affected (0.017 sec)

MariaDB [db]> DESCRIBE grade;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | YES  |     | NULL    |       |
| name       | varchar(10) | YES  |     | NULL    |       |
| attendance | double      | YES  |     | NULL    |       |
| midterm    | double      | YES  |     | NULL    |       |
| assignment | double      | YES  |     | NULL    |       |
| final      | double      | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.023 sec)
```

Data Manipulation Language: INSERT INTO TABLE

- INSERT INTO table_name
  - https://mariadb.com/kb/en/insert/

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
 [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
 {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
 [ ON DUPLICATE KEY UPDATE
   col=expr
     [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

- $\text{grade} \leftarrow \text{grade} \cup \{(615453, "J.B.", 10, 30, 30, 30)\}$

id INTEGER	name VARCHAR(10)	attendance DOUBLE	midterm DOUBLE	assignment DOUBLE	final DOUBLE
615453	J. B.	10	30	30	30

- INSERT INTO grade VALUES (615453, 'J.B.', 10, 30, 30, 30);

```
MariaDB [db]> INSERT INTO grade VALUES (615453, 'J.B.', 10, 30, 30, 30);
Query OK, 1 row affected (0.001 sec)
```

Data Manipulation Language: INSERT INTO TABLE

- INSERT INTO table_name
  - retrieve full records of the table
    - SELECT * FROM *table_name*
    - SELECT * FROM grade;

```
MariaDB [db]> SELECT * FROM grade;
+--------+------+------------+---------+------------+-------+
| id     | name | attendance | midterm | assignment | final |
+--------+------+------------+---------+------------+-------+
| 615453 | J.B. |         10 |      30 |         30 |    30 |
+--------+------+------------+---------+------------+-------+
1 row in set (0.000 sec)
```

Data Manipulation Language: INSERT INTO TABLE

- INSERT INTO table_name
  - You can insert two or more records
    - INSERT INTO grade VALUES (615453, 'J.B.', 10,30,30,30), (123, 'J.C.', 9,27,25,23);
    - INSERT grade VALUE (615453, 'J.B.', 10,30,30,30), (123, 'J.C.', 9,27,25,23);

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
 [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
 {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
 [ ON DUPLICATE KEY UPDATE
   col=expr
     [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

```
MariaDB [db]> INSERT INTO grade VALUES (615453, 'J.B.', 10,30,30,30), (123, 'J.C.', 9,27,25,23);
Query OK, 2 rows affected (0.001 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [db]> SELECT * FROM grade;
+--------+------+------------+---------+------------+-------+
| id     | name | attendance | midterm | assignment | final |
+--------+------+------------+---------+------------+-------+
| 615453 | J.B. |         10 |      30 |         30 |    30 |
| 615453 | J.B. |         10 |      30 |         30 |    30 |
|    123 | J.C. |          9 |      27 |         25 |    23 |
+--------+------+------------+---------+------------+-------+
3 rows in set (0.000 sec)
```

Data Manipulation Language: DELETE FROM TABLE

- DELETE FROM table_name
  - https://mariadb.com/kb/en/delete/

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
  FROM tbl_name [PARTITION (partition_list)]
  [WHERE where_condition]
  [ORDER BY ...]
  [LIMIT row_count]
  [RETURNING select_expr
    [, select_expr ...]]
```

- DELETE ALL ROWS
  - DELETE FROM grade;

```
MariaDB [db]> DELETE FROM grade;
Query OK, 3 rows affected (0.001 sec)

MariaDB [db]> SELECT * FROM grade;
Empty set (0.000 sec)
```

Data Manipulation Language: UPDATE TABLE

- UPDATE TABLE
  - https://mariadb.com/kb/en/update/
  - updates columns of existing rows in the named table with new values.

Single-table syntax:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
  [PARTITION (partition_list)]
  SET col1={expr1|DEFAULT} [,col2={expr2|DEFAULT}] ...
  [WHERE where_condition]
  [ORDER BY ...]
  [LIMIT row_count]
```

Multiple-table syntax:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
    SET col1={expr1|DEFAULT} [, col2={expr2|DEFAULT}] ...
    [WHERE where_condition]
```

Data Manipulation Language: UPDATE TABLE

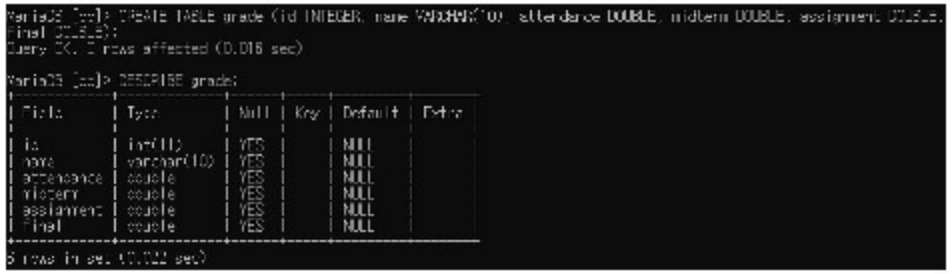
- UPDATE TABLE
  - Motivating example

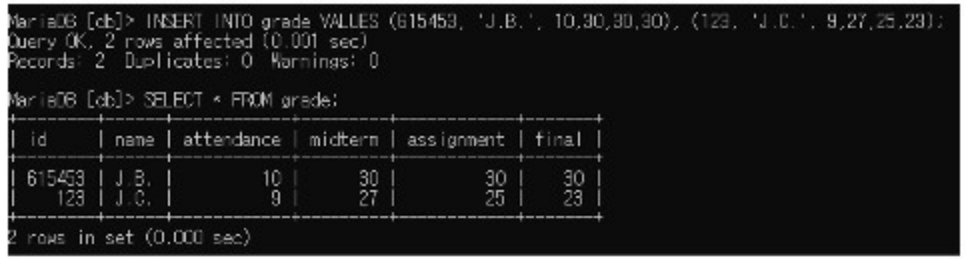
id INTEGER	name VARCHAR(10)	attendance DOUBLE	midterm DOUBLE	assignment DOUBLE	final DOUBLE	total DOUBLE
615453	J. B.	10	30	30	30	~~100~~

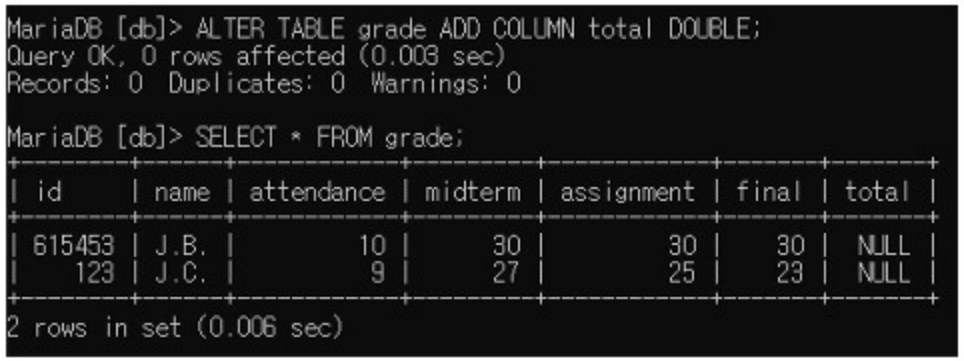
  - Add ‘total’ column to store the total point
    - Revisit Alter Table
  - Update the total point with the sum of ‘attendance’ ‘midterm’ ‘assignment’ ‘final’

Data Manipulation Language: UPDATE TABLE

- UPDATE TABLE
  - Motivating example
  - Add ‘total’ column to store the total point
    - Revisit Alter Table







Data Manipulation Language: UPDATE TABLE

- UPDATE TABLE
  - Motivating example
    - Update the total point with the sum of ‘attendance’ ‘midterm’ ‘assignment’ ‘final’

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
  [PARTITION (partition_list)]
  SET col1={expr1|DEFAULT} [,col2={expr2|DEFAULT}] ...
  [WHERE where_condition]
  [ORDER BY ...]
  [LIMIT row_count]
```

Data Manipulation Language: UPDATE TABLE

- UPDATE TABLE
  - One more motivating example

id INTEGER	name VAR CHAR(10)	attendance DOUBLE	midterm DOUBLE	assignment DOUBLE	final DOUBLE	total DOUBLE	last_update DATE
615453	J. B.	10	30	30	30	100	

  - Add ‘last_update’ column to store the time of the last update of this row

```
MariaDB [db]> ALTER TABLE grade ADD COLUMN last_update DATE;
Query OK, 0 rows affected (0.003 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [db]> SELECT * FROM grade;
+--------+------+------------+---------+------------+-------+-------+-------------+
| id     | name | attendance | midterm | assignment | final | total | last_update |
+--------+------+------------+---------+------------+-------+-------+-------------+
| 615453 | J.B. |         10 |      30 |         30 |    30 |   100 | NULL        |
|    123 | J.C. |          9 |      27 |         25 |    23 |    84 | NULL        |
+--------+------+------------+---------+------------+-------+-------+-------------+
2 rows in set (0.006 sec)
```

  - Update the last update of currently ‘null’ to current time

Data Manipulation Language: UPDATE TABLE

- UPDATE TABLE
  - Update the last update of currently ‘null’ to current time
    - using CURDATE() **built-in function**

```
MariaDB [db]> UPDATE grade SET last_update=CURDATE();
Query OK, 2 rows affected (0.001 sec)
Rows matched: 2  Changed: 2  Warnings: 0

MariaDB [db]> SELECT * FROM grade;
+--------+------+------------+---------+------------+-------+-------+-------------+
| id     | name | attendance | midterm | assignment | final | total | last_update |
+--------+------+------------+---------+------------+-------+-------+-------------+
| 615453 | J.B. |         10 |      30 |         30 |    30 |   100 | 2021-02-28  |
|    123 | J.C. |          9 |      27 |         25 |    23 |    84 | 2021-02-28  |
+--------+------+------------+---------+------------+-------+-------+-------------+
2 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  - We would try some of selected functions by the lecturer
  - https://mariadb.com/kb/en/built-in-functions/

1. String Functions
2. Date & Time Functions
3. Aggregate Functions
4. Numeric Functions
5. Control Flow Functions
6. Bit Functions and Operators
7. Encryption, Hashing and Compression Functions
8. Information Functions
9. Miscellaneous Functions
10. Dynamic Columns Functions
11. Geographic Functions
12. JSON Functions
13. Spider Functions
14. Window Functions

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - CHARACTER_LENGTH(str) or CHAR_LENGTH()

CHARACTER_LENGTH

Synonym for CHAR_LENGTH().

CHAR_LENGTH

Length of the string in characters.

```
+-------------+
| stringValue |
+-------------+
| Jack        |
| 123         |
| MariaDB     |
+-------------+
```

```
MariaDB [DatabaseProgramming]> SELECT stringValue from bif;
+-------------+
| stringValue |
+-------------+
| Jack        |
| 123         |
| MariaDB     |
+-------------+
3 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT CHAR_LENGTH(stringValue) from bif;
+--------------------------+
| CHAR_LENGTH(stringValue) |
+--------------------------+
|                        4 |
|                        3 |
|                        7 |
+--------------------------+
3 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - LENGTH(str)

LENGTH

Length of the string in bytes.

LENGTHB

Synonym for LENGTH().

```
MariaDB [DatabaseProgramming]> SELECT * from bif;
+-------------+------------+--------------+----------+-----------------------------+
| stringValue | dateValue  | integerValue | bitValue | geoValue                    |
+-------------+------------+--------------+----------+-----------------------------+
| Jack        | 2020-08-30 |           35 | Z        |     rr
龍p=?@qᄒᆨ??@ |
| 123         | NULL       |         NULL | NULL     | NULL                        |
| MariaDB     | NULL       |         NULL | NULL     | NULL                        |
| M           | NULL       |         NULL | NULL     | NULL                        |
| 432         | NULL       |          123 | NULL     | NULL                        |
+-------------+------------+--------------+----------+-----------------------------+
5 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT LENGTH(stringValue), LENGTH(dateValue), LENGTH(integerValue), LENGTH(bitValue), LE
NGTH(geoValue) from bif;
+--------------------+------------------+----------------------+-----------------+------------------+
| LENGTH(stringValue) | LENGTH(dateValue) | LENGTH(integerValue) | LENGTH(bitValue) | LENGTH(geoValue) |
+--------------------+------------------+----------------------+-----------------+------------------+
|                  4 |               10 |                    2 |               1 |               25 |
|                  3 |             NULL |                 NULL |            NULL |             NULL |
|                  7 |             NULL |                 NULL |            NULL |             NULL |
|                  1 |             NULL |                 NULL |            NULL |             NULL |
|                  3 |             NULL |                    3 |            NULL |             NULL |
+--------------------+------------------+----------------------+-----------------+------------------+
5 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - CHAR(N,... [USING charset_name])

     - INSERT INTO *table_name* SET *column_name* = value;

CHAR Function

Returns string based on the integer values for the individual characters.

```
MariaDB [DatabaseProgramming]> INSERT INTO bif SET stringValue = CHAR(77,97,114,'105', 97, '68', 66);
Query OK, 1 row affected (0.017 sec)

MariaDB [DatabaseProgramming]> SELECT stringValue from bif;
+-------------+
| stringValue |
+-------------+
| Jack        |
| 123         |
| MariaDB     |
+-------------+
3 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - CHR(N)

CHR

Returns string based on integer values of the individual characters.

```
MariaDB [DatabaseProgramming]> INSERT INTO bif SET stringValue = CHR(77);
Query OK, 1 row affected (0.002 sec)

MariaDB [DatabaseProgramming]> SELECT stringValue from bif;
+-------------+
| stringValue |
+-------------+
| Jack        |
| 123         |
| MariaDB     |
| M           |
+-------------+
4 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - CONCAT(str1, str2, ...)

```
SELECT CONCAT(id,
',',name,',',attendance,',',midterm,',',assignment,',', final,',', total,',',
last_update)  FROM grade;
```

CONCAT

Returns concatenated string.

```
MariaDB [DatabaseProgramming]> SELECT stringValue from bif;
+-------------+
| stringValue |
+-------------+
| Jack        |
| 123         |
| MariaDB     |
| M           |
+-------------+
4 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT CONCAT(stringValue, 'hello') FROM bif;
+-----------------------------+
| CONCAT(stringValue, 'hello') |
+-----------------------------+
| Jackhello                   |
| 123hello                    |
| MariaDBhello                |
| Mhello                      |
+-----------------------------+
4 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - CONCAT_WS(separator, str1, str2, ...)

```
SELECT
CONCAT_WS(',',id,name,attendance,midterm,assignment,final,total,last_update) FROM grade;
```

CONCAT_WS

Concatenate with separator.

```
MariaDB [DatabaseProgramming]> SELECT CONCAT_WS(',',stringValue, 'hello') FROM bif;
+-------------------------------------+
| CONCAT_WS(',',stringValue, 'hello') |
+-------------------------------------+
| Jack,hello                          |
| 123,hello                           |
| MariaDB,hello                       |
| M,hello                             |
+-------------------------------------+
4 rows in set (0.001 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - LCASE(str) or LOWER(str)

LCASE

Synonym for LOWER().

LOWER

Returns a string with all characters changed to lowercase.

```
MariaDB [DatabaseProgramming]> SELECT stringValue from bif;
+-------------+
| stringValue |
+-------------+
| Jack        |
| 123         |
| MariaDB     |
| M           |
| 432         |
+-------------+
5 rows in set (0.001 sec)

MariaDB [DatabaseProgramming]> SELECT LCASE(stringValue) from bif;
+---------------------+
| LCASE(stringValue) |
+---------------------+
| jack                |
| 123                 |
| mariadb             |
| m                   |
| 432                 |
+---------------------+
5 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - UCASE(str) or UPPER(str)

UCASE

Synonym for UPPER().

```
MariaDB [DatabaseProgramming]> SELECT stringValue FROM bif;
+-------------+
| stringValue |
+-------------+
| abc         |
| bcd         |
| cde         |
|    abc      |
| bcd         |
|    cde      |
+-------------+
6 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT UCASE(TRIM(stringValue)) FROM bif;
+--------------------------+
| UCASE(TRIM(stringValue)) |
+--------------------------+
| ABC                      |
| BCD                      |
| CDE                      |
| ABC                      |
| BCD                      |
| CDE                      |
+--------------------------+
6 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions

```
TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str), TRIM([remstr FROM] str)
```

TRIM

Returns a string with all given prefixes or suffixes removed.

```
MariaDB [DatabaseProgramming]> INSERT INTO bif SET stringValue = '   abc';
Query OK, 1 row affected (0.003 sec)

MariaDB [DatabaseProgramming]> INSERT INTO bif SET stringValue = 'bcd   ';
Query OK, 1 row affected (0.002 sec)

MariaDB [DatabaseProgramming]> INSERT INTO bif SET stringValue = '   cde   ';
Query OK, 1 row affected (0.004 sec)

MariaDB [DatabaseProgramming]> SELECT stringValue FROM bif;
+--------------+
| stringValue  |
+--------------+
| abc          |
| bcd          |
| cde          |
|    abc       |
| bcd          |
|    cde       |
+--------------+
6 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT TRIM(stringValue) FROM bif;
+-------------------+
| TRIM(stringValue) |
+-------------------+
| abc               |
| bcd               |
| cde               |
| abc               |
| bcd               |
| cde               |
+-------------------+
6 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions

```
SUBSTRING(str,pos),
SUBSTRING(str FROM pos),
SUBSTRING(str,pos,len),
SUBSTRING(str FROM pos FOR len)
```

SUBSTRING

Returns a substring from string starting at a given position.

```
MariaDB [DatabaseProgramming]> SELECT stringValue FROM bif;
+-------------+
| stringValue |
+-------------+
| abc         |
| bcd         |
| cde         |
+-------------+
3 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT SUBSTRING(stringValue, 2, 1) FROM bif;
+-----------------------------+
| SUBSTRING(stringValue, 2, 1) |
+-----------------------------+
| b                           |
| c                           |
| d                           |
+-----------------------------+
3 rows in set (0.000 sec)
```

Q: Print out ab, bc, cd?

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - REPLACE(str, from_str, to_str)

REPLACE Function

Replace occurrences of a string.

```
MariaDB [DatabaseProgramming]> SELECT stringValue FROM bif;
+-------------+
| stringValue |
+-------------+
| 20-21-23    |
| 18-12-21    |
| 1812-21     |
+-------------+
3 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT REPLACE(stringValue, '-', ',') FROM bif;
+---------------------------------+
| REPLACE(stringValue, '-', ',') |
+---------------------------------+
| 20,21,23                        |
| 18,12,21                        |
| 1812,21                         |
+---------------------------------+
3 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - INSTR(str, substr)

INSTR

Returns the position of a string within a string.

```
MariaDB [DatabaseProgramming]> SELECT stringValue from bif;
+-------------+
| stringValue |
+-------------+
| Jack        |
| 123         |
| MariaDB     |
| M           |
| 432         |
+-------------+
5 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT INSTR(stringValue, 'a') from bif;
+--------------------------+
| INSTR(stringValue, 'a') |
+--------------------------+
|                        2 |
|                        0 |
|                        2 |
|                        0 |
|                        0 |
+--------------------------+
5 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - POSITION(substr IN str)

POSITION

Returns the position of a substring in a string.

```
MariaDB [DatabaseProgramming]> SELECT stringValue FROM bif;
+-------------+
| stringValue |
+-------------+
| 20-21-23    |
| 18-12-21    |
| 1812-21     |
+-------------+
3 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT POSITION('21' in stringValue) FROM bif;
+-----------------------------------+
| POSITION('21' in stringValue) |
+-----------------------------------+
|                                 4 |
|                                 7 |
|                                 6 |
+-----------------------------------+
3 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - STRCMP(expr1, expr2)

STRCMP

Compares two strings in sort order.

```
MariaDB [DatabaseProgramming]> DELETE FROM bif;
Query OK, 3 rows affected (0.002 sec)

MariaDB [DatabaseProgramming]> INSERT INTO bif SET stringValue = 'abc';
Query OK, 1 row affected (0.016 sec)

MariaDB [DatabaseProgramming]> INSERT INTO bif SET stringValue = 'bcd';
Query OK, 1 row affected (0.016 sec)

MariaDB [DatabaseProgramming]> INSERT INTO bif SET stringValue = 'cde';
Query OK, 1 row affected (0.002 sec)

MariaDB [DatabaseProgramming]> SELECT stringValue FROM bif;
+-------------+
| stringValue |
+-------------+
| abc         |
| bcd         |
| cde         |
+-------------+
3 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT STRCMP('bcd',stringValue) FROM bif;
+----------------------------+
| STRCMP('bcd',stringValue) |
+----------------------------+
|                          1 |
|                          0 |
|                         -1 |
+----------------------------+
3 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - CAST(expr AS type)

CAST

Casts a value of one type to another type.

M123231
M321225
M522124

Q: if larger than 300000 -> print 1
else -> print 0

```
MariaDB [DatabaseProgramming]> SELECT stringValue from bif;
+-------------+
| stringValue |
+-------------+
| Jack        |
| 123         |
+-------------+
2 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT CAST(stringValue as INTEGER) from bif;
+--------------------------------+
| CAST(stringValue as INTEGER) |
+--------------------------------+
|                              0 |
|                            123 |
+--------------------------------+
2 rows in set, 1 warning (0.000 sec)

MariaDB [DatabaseProgramming]> SHOW WARNINGS;
+---------+------+-----------------------------------------+
| Level   | Code | Message                                 |
+---------+------+-----------------------------------------+
| Warning | 1292 | Truncated incorrect INTEGER value: 'Jack' |
+---------+------+-----------------------------------------+
1 row in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  1. String Functions
     - HEX(N_or_S)

HEX

Returns hexadecimal value.

```
MariaDB [DatabaseProgramming]> SELECT stringValue, integerValue from bif;
+-------------+--------------+
| stringValue | integerValue |
+-------------+--------------+
| Jack        |           35 |
| 123         |         NULL |
| MariaDB     |         NULL |
| M           |         NULL |
| 432         |          123 |
+-------------+--------------+
5 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT HEX(stringValue), HEX(integerValue) from bif;
+------------------+--------------------+
| HEX(stringValue) | HEX(integerValue)  |
+------------------+--------------------+
| 4A61636B         | 23                 |
| 313233           | NULL               |
| 4D617269614442   | NULL               |
| 4D               | NULL               |
| 343332           | 7B                 |
+------------------+--------------------+
5 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  2. Date & Time Functions
     - Revisit ALTER TABLE to add timeVALUE TIME column

ALTER TABLE bif ADD COLUMN timeValue TIME;

```
MariaDB [DatabaseProgramming]> DESCRIBE bif;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| stringValue  | varchar(10) | YES  |     | NULL    |       |
| dateValue    | date        | YES  |     | NULL    |       |
| integerValue | int(11)     | YES  |     | NULL    |       |
| bitValue     | bit(8)      | YES  |     | NULL    |       |
| geoValue     | point       | YES  |     | NULL    |       |
| timeValue    | time        | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
6 rows in set (0.006 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  2. Date & Time Functions
     - CURDATE & CURTIME

CURDATE

Returns the current date.

CURTIME

Returns the current time.

```
MariaDB [DatabaseProgramming]> DELETE FROM bif;
Query OK, 6 rows affected (0.002 sec)

MariaDB [DatabaseProgramming]> INSERT INTO bif SET dateValue = CURDATE();
Query OK, 1 row affected (0.002 sec)

MariaDB [DatabaseProgramming]> SELECT dateValue FROM bif;
+------------+
| dateValue  |
+------------+
| 2020-08-30 |
+------------+
1 row in set (0.000 sec)
```

```
MariaDB [DatabaseProgramming]> INSERT INTO bif SET timeValue = CURTIME();
Query OK, 1 row affected (0.015 sec)

MariaDB [DatabaseProgramming]> SELECT dateValue, timeValue FROM bif;
+------------+-----------+
| dateValue  | timeValue |
+------------+-----------+
| 2020-08-30 | NULL      |
| NULL       | 12:52:47  |
+------------+-----------+
2 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  2. Date & Time Functions
     - INTERVAL and arithmetic expression
       - INTERVAL time_quantity time_unit
  - Practice
    - INSERT ONE ROW where its dateValue is yesterday
  - Q: 1 year ago?

```
MariaDB [DatabaseProgramming]> SELECT dateValue FROM bif;
+------------+
| dateValue  |
+------------+
| 2020-08-30 |
| NULL       |
+------------+
2 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT dateValue + INTERVAL 1 SECOND FROM bif;
+-----------------------------------+
| dateValue + INTERVAL 1 SECOND     |
+-----------------------------------+
| 2020-08-30 00:00:01               |
| NULL                              |
+-----------------------------------+
2 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT dateValue - INTERVAL 1 SECOND FROM bif;
+-----------------------------------+
| dateValue - INTERVAL 1 SECOND     |
+-----------------------------------+
| 2020-08-29 23:59:59               |
| NULL                              |
+-----------------------------------+
2 rows in set (0.000 sec)
```

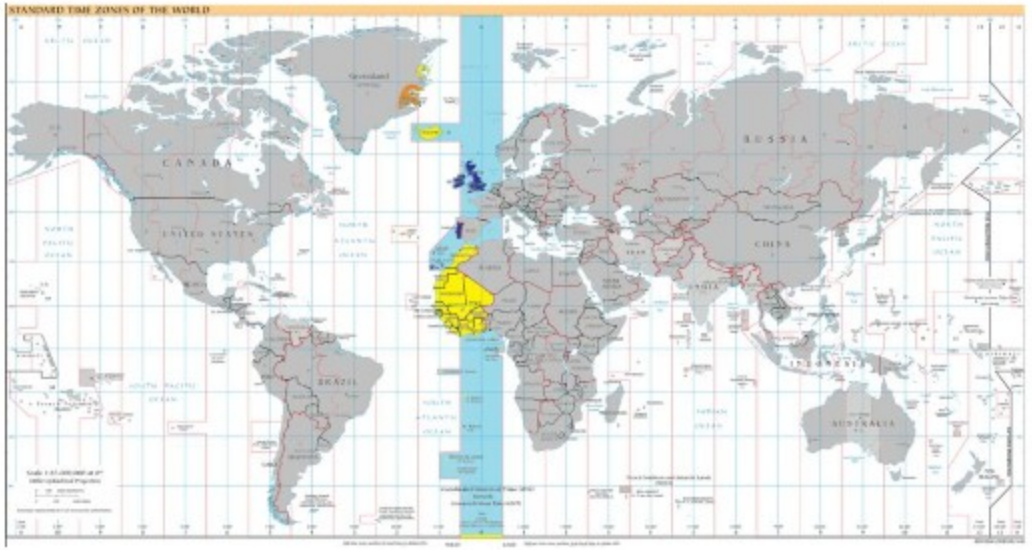
Unit	Description
MICROSECOND	Microseconds
SECOND	Seconds
MINUTE	Minutes
HOUR	Hours
DAY	Days
WEEK	Weeks
MONTH	Months
QUARTER	Quarters
YEAR	Years
SECOND_MICROSECOND	Seconds.Microseconds
MINUTE_MICROSECOND	Minutes.Seconds.Microseconds
MINUTE_SECOND	Minutes.Seconds
HOUR_MICROSECOND	Hours.Minutes.Seconds.Microseconds
HOUR_SECOND	Hours.Minutes.Seconds
HOUR_MINUTE	Hours.Minutes
DAY_MICROSECOND	Days Hours.Minutes.Seconds.Microseconds
DAY_SECOND	Days Hours.Minutes.Seconds
DAY_MINUTE	Days Hours.Minutes
DAY_HOUR	Days Hours
YEAR_MONTH	Years-Months

Built-in Functions

- MariaDB Built-in Functions
  2. Date & Time Functions
     - CONVERT_TZ(dt, from_tz, to _tz)

CONVERT_TZ

Converts a datetime from one time zone to another.



- Korea: +09:00
- Greenwich Mean Time: +00:00

- Q: Print out current Greenwich Mean Time

```
MariaDB [DatabaseProgramming]> SELECT dateValue FROM bif;
+------------+
| dateValue  |
+------------+
| 2020-08-30 |
| NULL       |
+------------+
2 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT CONVERT_TZ(dateValue, '+09:00', '+00:00') FROM bif;
+-----------------------------------------+
| CONVERT_TZ(dateValue, '+09:00', '+00:00') |
+-----------------------------------------+
| 2020-08-29 15:00:00                     |
| NULL                                    |
+-----------------------------------------+
2 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  2. Date & Time Functions
     - UNIX_TIMESTAMP FROM_UNIXTIME(unix_timestamp)

UNIX_TIMESTAMP

Returns a Unix timestamp.

FROM_UNIXTIME

Returns a datetime from a Unix timestamp.

- https://www.epochconverter.com/

```
MariaDB [DatabaseProgramming]> SELECT dateValue FROM bif;
+------------+
| dateValue  |
+------------+
| 2020-08-30 |
| NULL       |
+------------+
2 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT UNIX_TIMESTAMP(dateValue) FROM bif;
+---------------------------+
| UNIX_TIMESTAMP(dateValue) |
+---------------------------+
|                1598713200 |
|                      NULL |
+---------------------------+
2 rows in set (0.000 sec)
```

```
MariaDB [DatabaseProgramming]> INSERT INTO bif SET dateValue=FROM_UNIXTIME(0);
Query OK, 1 row affected, 1 warning (0.003 sec)

MariaDB [DatabaseProgramming]> SELECT dateValue FROM bif;
+------------+
| dateValue  |
+------------+
| 2020-08-30 |
| NULL       |
| 1970-01-01 |
+------------+
3 rows in set (0.000 sec)
```

System.currentTimeMillis();

long java.lang.System.currentTimeMillis()

@IntrinsicCandidate

Returns the current time in milliseconds. Note that while the unit of time of the return value is a millisecond, the granularity of the value depends on the underlying operating system and may be larger. For example, many operating systems measure time in units of tens of milliseconds.

See the description of the class `Date` for a discussion of slight discrepancies that may arise between "computer time" and coordinated universal time (UTC).

Returns:

the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.

See Also:

java.util.Date

FROM_UNIXTIME gets **a second value** instead of milliseconds only positive values are available

Built-in Functions

- MariaDB Built-in Functions
  2. Date & Time Functions
     - YEAR
     - MONTH
     - DAYOFMONTH
     - DAYOFWEEK (1=Sunday)
     - HOUR
     - MINUTE
     - SECOND

```
MariaDB [DatabaseProgramming]> SELECT dateValue FROM bif;
+------------+
| dateValue  |
+------------+
| 2020-08-30 |
| NULL       |
| 1970-01-01 |
+------------+
3 rows in set (0.000 sec)

MariaDB [DatabaseProgramming]> SELECT YEAR(dateValue),MONTH(dateValue),DAYOFMONTH(dateValue) FROM bif;
+--------------------+---------------------+-------------------------+
| YEAR(dateValue)    | MONTH(dateValue)    | DAYOFMONTH(dateValue)   |
+--------------------+---------------------+-------------------------+
|               2020 |                   8 |                      30 |
|               NULL |                NULL |                    NULL |
|               1970 |                   1 |                       1 |
+--------------------+---------------------+-------------------------+
3 rows in set (0.002 sec)
```

```
MariaDB [DatabaseProgramming]> SELECT DAYOFWEEK(dateValue) FROM bif;
+-------------------------+
| DAYOFWEEK(dateValue)    |
+-------------------------+
|                       1 |
|                    NULL |
|                       5 |
+-------------------------+
3 rows in set (0.001 sec)
```

```
MariaDB [DatabaseProgramming]> SELECT timeValue FROM bif;
+-------------+
| timeValue   |
+-------------+
| NULL        |
| 12:52:47    |
| NULL        |
+-------------+
3 rows in set (0.001 sec)

MariaDB [DatabaseProgramming]> SELECT HOUR(timeValue), MINUTE(timeValue), SECOND(timeValue) FROM bif;
+--------------------+----------------------+----------------------+
| HOUR(timeValue)    | MINUTE(timeValue)    | SECOND(timeValue)    |
+--------------------+----------------------+----------------------+
|               NULL |                 NULL |                 NULL |
|                 12 |                   52 |                   47 |
|               NULL |                 NULL |                 NULL |
+--------------------+----------------------+----------------------+
3 rows in set (0.000 sec)
```

Q: Today's day of week?
Q: Tomorrow's day of week?

Built-in Functions

- MariaDB Built-in Functions
  2. Date & Time Functions
     - https://mariadb.com/kb/en/str_to_date/
     - STR_TO_DATE(str,format)

Option	Description
%a	Short weekday name in current locale (Variable lc_time_names).
%b	Short form month name in current locale. For locale en_US this is one of: Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov or Dec.
%c	Month with 1 or 2 digits.
%D	Day with English suffix 'th', 'nd', 'st' or 'rd'. (1st, 2nd, 3rd...).
%d	Day with 2 digits.
%e	Day with 1 or 2 digits.
%f	Sub seconds 6 digits.
%H	Hour with 2 digits between 00-23.
%h	Hour with 2 digits between 01-12.
%I	Hour with 2 digits between 01-12.
%i	Minute with 2 digits.
%j	Day of the year (001-366)
%k	Hour with 1 digits between 0-23

- STR_TO_DATE('June 2, 2014', '%M %e, %Y');

```
MariaDB [databaseprogramming]> INSERT INTO bif SET dateValue=STR_TO_DATE('June 2, 2014', '%M %e, %Y');
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> SELECT dateValue FROM bif;
+------------+
| dateValue  |
+------------+
| 2020-08-30 |
| NULL       |
| 1970-01-01 |
| 2014-06-02 |
+------------+
4 rows in set (0.000 sec)
```

Q: The day of week of 'April 07 2022'?

Built-in Functions

- MariaDB Built-in Functions
  2. Date & Time Functions
     - DATE_FORMAT(date, format[, locale])
     - https://mariadb.com/kb/en/date_format/

       - DATE_FORMAT(dateValue, '%W %M %Y');
  - Practice: show like Thursday 7th April 2022

```
MariaDB [databaseprogramming]> SELECT dateValue FROM bif;
+------------+
| dateValue  |
+------------+
| 2020-08-30 |
| NULL       |
| 1970-01-01 |
| 2014-06-02 |
+------------+
4 rows in set (0.000 sec)

MariaDB [databaseprogramming]> SELECT DATE_FORMAT(dateValue, '%W %M %Y') FROM bif;
+-----------------------------------+
| DATE_FORMAT(dateValue, '%W %M %Y') |
+-----------------------------------+
| Sunday August 2020                |
| NULL                              |
| Thursday January 1970             |
| Monday June 2014                  |
+-----------------------------------+
4 rows in set (0.001 sec)
```

Option	Description
%a	Short weekday name in current locale (Variable lc_time_names).
%b	Short form month name in current locale. For locale en_US this is one of: Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov or Dec.
%c	Month with 1 or 2 digits.
%D	Day with English suffix 'th', 'nd', 'st' or 'rd". (1st, 2nd, 3rd...).
%d	Day with 2 digits.
%e	Day with 1 or 2 digits.
%f	Sub seconds 6 digits.
%H	Hour with 2 digits between 00-23.
%h	Hour with 2 digits between 01-12.
%I	Hour with 2 digits between 01-12.
%i	Minute with 2 digits.
%j	Day of the year (001-366)
%k	Hour with 1 digits between 0-23

Built-in Functions

- MariaDB Built-in Functions
  3. Aggregate Functions
     - AVG MAX MIN
     - STD SUM VARIANCE

```
MariaDB [databaseprogramming]> DELETE FROM bif;
Query OK, 4 rows affected (0.002 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET integerValue = 30;
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET integerValue = 40;
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET integerValue = 50;
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET integerValue = 60;
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET integerValue = 25;
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> SELECT integerValue FROM bif;
+--------------+
| integerValue |
+--------------+
|           30 |
|           40 |
|           50 |
|           60 |
|           25 |
+--------------+
```

```
MariaDB [databaseprogramming]> SELECT AVG(integerValue), MAX(integerValue), MIN(integerValue), STD(integerVAlue), SUM(integerValu
e), VARIANCE(integerValue) FROM bif;
+--------------------+--------------------+--------------------+--------------------+--------------------+------------------------+
| AVG(integerValue) | MAX(integerValue) | MIN(integerValue) | STD(integerVAlue) | SUM(integerValue) | VARIANCE(integerValue) |
+--------------------+--------------------+--------------------+--------------------+--------------------+------------------------+
|            41.0000 |                 60 |                 25 |            12.8062 |                205 |                164.0000 |
+--------------------+--------------------+--------------------+--------------------+--------------------+------------------------+
1 row in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  3. Aggregate Functions
     - COUNT
     - COUNT_DISTINCT

COUNT

Returns count of non-null values.

COUNT DISTINCT

Returns count of number of different non-NULL values.

```
MariaDB [databaseprogramming]> SELECT stringValue, integerValue FROM bif;
+-------------+--------------+
| stringValue | integerValue |
+-------------+--------------+
| A           |         NULL |
| B           |         NULL |
| A           |         NULL |
| NULL        |           35 |
+-------------+--------------+
4 rows in set (0.000 sec)
```

```
MariaDB [databaseprogramming]> SELECT COUNT(stringValue) FROM bif;
+---------------------+
| COUNT(stringValue) |
+---------------------+
|                   3 |
+---------------------+
1 row in set (0.000 sec)
```

```
MariaDB [databaseprogramming]> SELECT COUNT(DISTINCT stringValue) FROM bif;
+-----------------------------+
| COUNT(DISTINCT stringValue) |
+-----------------------------+
|                           2 |
+-----------------------------+
1 row in set (0.001 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  3. Aggregate Functions
     - BIT_AND BIT_OR BIT_XOR

```
MariaDB [databaseprogramming]> INSERT INTO bif SET bitValue = b'11110000';
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET bitValue = b'00001111';
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET bitValue = b'10101010';
Query OK, 1 row affected (0.002 sec)
```

```
MariaDB [databaseprogramming]> SELECT BIT_AND(bitValue),BIT_OR(bitValue),BIT_XOR(bitValue) FROM bif;
+-----------------+----------------+-----------------+
| BIT_AND(bitValue) | BIT_OR(bitValue) | BIT_XOR(bitValue) |
+-----------------+----------------+-----------------+
|               0 |            255 |              85 |
+-----------------+----------------+-----------------+
1 row in set (0.000 sec)
```

BIN – to Binary
OCT, HEX

Built-in Functions

- MariaDB Built-in Functions
  4. Numeric Functions
     - +, -, /, *, %
     - ( )
     - POW
     - SQRT

```
MariaDB [databaseprogramming]> ALTER TABLE bif ADD COLUMN (doubleValue DOUBLE);
Query OK, 0 rows affected (0.007 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [databaseprogramming]> DESCRIBE bif;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| stringValue | varchar(10) | YES  |     | NULL    |       |
| dateValue   | date        | YES  |     | NULL    |       |
| integerValue| int(11)     | YES  |     | NULL    |       |
| bitValue    | bit(8)      | YES  |     | NULL    |       |
| geoValue    | point       | YES  |     | NULL    |       |
| timeValue   | time        | YES  |     | NULL    |       |
| doubleValue | double      | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
7 rows in set (0.021 sec)
```

```
MariaDB [databaseprogramming]> SELECT doubleValue FROM bif;
+-------------+
| doubleValue |
+-------------+
|        35.3 |
|        25.3 |
|        43.3 |
|        27.3 |
+-------------+
4 rows in set (0.000 sec)
```

```
MariaDB [databaseprogramming]> SELECT (doubleValue+(5*7)-2)*2 FROM bif;
+--------------------------+
| (doubleValue+(5*7)-2)*2  |
+--------------------------+
|                    136.6 |
|                    116.6 |
|                    152.6 |
|                    120.6 |
+--------------------------+
4 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  4. Numeric Functions
     - FLOOR (35.5)
     - CEILING (35.5)

FLOOR

Largest integer value not greater than the argument.

CEILING

Returns the smallest integer not less than X.

```
MariaDB [databaseprogramming]> SELECT doubleValue FROM bif;
+-------------+
| doubleValue |
+-------------+
|        35.3 |
|        25.3 |
|        43.3 |
|        27.3 |
+-------------+
4 rows in set (0.000 sec)

MariaDB [databaseprogramming]> SELECT FLOOR(doubleValue), CEILING(doubleValue) FROM bif;
+--------------------+----------------------+
| FLOOR(doubleValue) | CEILING(doubleValue) |
+--------------------+----------------------+
|                 35 |                   36 |
|                 25 |                   26 |
|                 43 |                   44 |
|                 27 |                   28 |
+--------------------+----------------------+
4 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  4. Numeric Functions
     - RAND
       - generate 0 <= x < 1 floating value
     - Practice: generate 5 to 15 integer random value

INSERT INTO bif SET doubleValue =

SELECT _____ ;

Built-in Functions

- MariaDB Built-in Functions
    4. Numeric Functions
        - COS, TAN, SIN, ...
        - LOG, LOG10, LOG2, ...

Built-in Functions

- MariaDB Built-in Functions
  5. Control Flow Functions
     - CASE OPERATOR

```
CASE value WHEN [compare_value] THEN result [WHEN [compare_value] THEN
result ...] [ELSE result] END

CASE WHEN [condition] THEN result [WHEN [condition] THEN result ...]
[ELSE result] END
```

     - CASE integerValue WHEN 60 THEN ‘60'

```
MariaDB [databaseprogramming]> SELECT integerValue FROM bif;
+--------------+
| integerValue |
+--------------+
|           95 |
|           85 |
|           75 |
|           65 |
|           55 |
+--------------+
5 rows in set (0.001 sec)
```

```
MariaDB [databaseprogramming]> SELECT CASE WHEN integerValue >= 90 THEN 'A' WHEN integerValue >=80 THEN 'B' ELSE 'C' END
 FROM bif;
+-----------------------------------------------------------------------------+
| CASE WHEN integerValue >= 90 THEN 'A' WHEN integerValue >=80 THEN 'B' ELSE 'C' END |
+-----------------------------------------------------------------------------+
| A                                                                           |
| B                                                                           |
| C                                                                           |
| C                                                                           |
| C                                                                           |
+-----------------------------------------------------------------------------+
5 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  5. Control Flow Functions
     - IF Function

```
IF(expr1,expr2,expr3)
```

```
MariaDB [databaseprogramming]> SELECT integerValue FROM bif;
+--------------+
| integerValue |
+--------------+
|           95 |
|           85 |
|           75 |
|           65 |
|           55 |
+--------------+
5 rows in set (0.001 sec)

MariaDB [databaseprogramming]> SELECT IF(integerValue>=60, 'PASS', 'FAIL') FROM bif;
+--------------------------------------+
| IF(integerValue>=60, 'PASS', 'FAIL') |
+--------------------------------------+
| PASS                                 |
| PASS                                 |
| PASS                                 |
| PASS                                 |
| FAIL                                 |
+--------------------------------------+
5 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  5. Control Flow Functions
     - IF Function
       - INTERVAL
       - BINARY , COLLATE
       - !
       - - (unary minus), [[bitwise-not|]] (unary bit inversion)
       - || (string concatenation)
       - ^
       - * , / , DIV , % , MOD
       - - , +
       - << , >>
       - &
       - |
       - = (comparison), <=> , >= , > , <= , < , <> , != , IS , LIKE , REGEXP , IN
       - BETWEEN , CASE , WHEN , THEN , ELSE , END
       - NOT
       - && , AND
       - XOR
       - || (logical or), OR
       - = (assignment), :=

Built-in Functions

- MariaDB Built-in Functions
  6. Bit Functions and Operators
     - Bit Functions

&

Bitwise AND

<<

Left shift

>>

Shift right

BIT_COUNT

Returns the number of set bits

^

Bitwise XOR

|

Bitwise OR

~

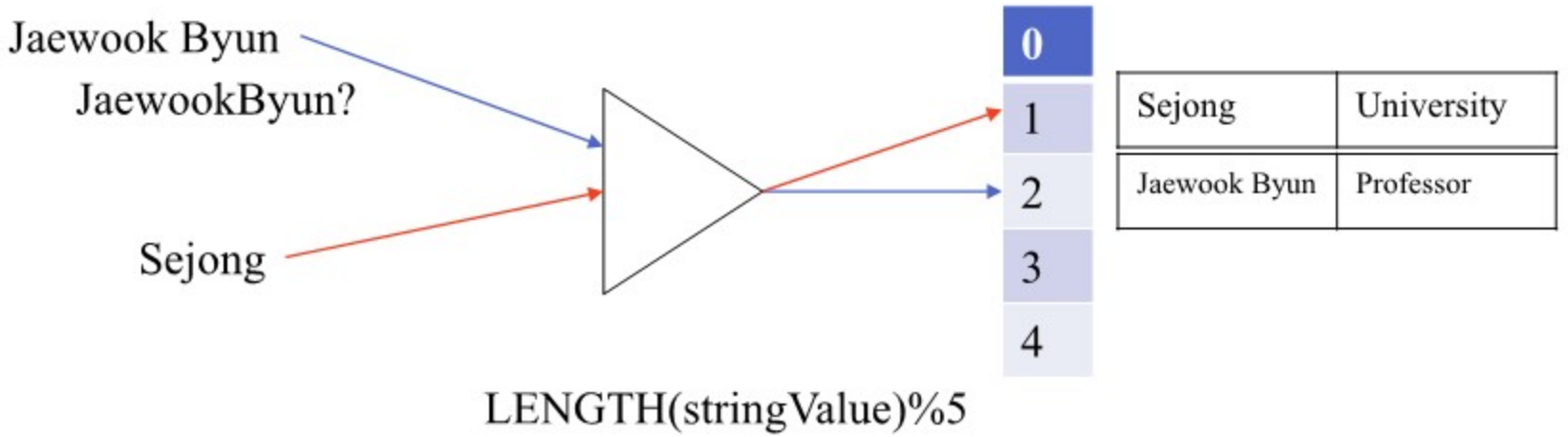
Bitwise NOT

Parentheses

Parentheses modify the precedence of other operators in an expression

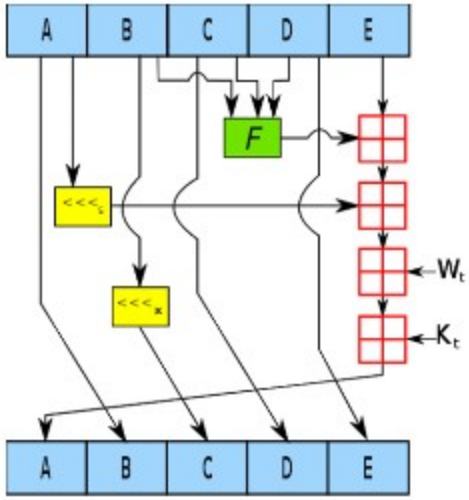
Built-in Functions

- MariaDB Built-in Functions
  7. Encryption, Hashing and Compression Functions
     - Application for Hashing
       - Data Structure, Hash Table or Map
       - Check redundancy
       - Check forgery



Built-in Functions

- MariaDB Built-in Functions
  7. Encryption, Hashing and Compression Functions
     - SHA2(str, hash_len)
       - hash_len: 224, 256, 384, 512




```
MariaDB [databaseprogramming]> SELECT stringValue FROM bif;
+-------------+
| stringValue |
+-------------+
| Sejong      |
| Jaewook     |
+-------------+
2 rows in set (0.000 sec)

MariaDB [databaseprogramming]> SELECT SHA2(stringValue,256) FROM bif;
+------------------------------------------------------------------+
| SHA2(stringValue,256)                                            |
+------------------------------------------------------------------+
| 4394eae0a16eb125f8546e8b94f2e89cb74bdfb0e144cd45f85bc29cbb9be7aa |
| 235184838a66683751859e17ea7d2299c8cefeae16556cfbfe0f4e725982f353 |
+------------------------------------------------------------------+
2 rows in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  7. Encryption, Hashing and Compression Functions
     - PASSWORD(str)

```
MariaDB [databaseprogramming]> INSERT INTO bif SET stringValue = PASSWORD('12345');
ERROR 1406 (22001): Data too long for column 'stringValue' at row 1
MariaDB [databaseprogramming]> ALTER TABLE bif ADD COLUMN (passwordValue VARCHAR(41));
Query OK, 0 rows affected (0.006 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [databaseprogramming]> DESCRIBE bif;
+---------------+------------+------+-----+---------+-------+
| Field         | Type       | Null | Key | Default | Extra |
+---------------+------------+------+-----+---------+-------+
| stringValue   | varchar(10) | YES  |     | NULL    |       |
| dateValue     | date       | YES  |     | NULL    |       |
| integerValue  | int(11)    | YES  |     | NULL    |       |
| bitValue      | bit(8)     | YES  |     | NULL    |       |
| geoValue      | point      | YES  |     | NULL    |       |
| timeValue     | time       | YES  |     | NULL    |       |
| doubleValue   | double     | YES  |     | NULL    |       |
| passwordValue | varchar(41) | YES  |     | NULL    |       |
+---------------+------------+------+-----+---------+-------+
8 rows in set (0.008 sec)
```

```
MariaDB [databaseprogramming]> INSERT INTO bif SET passwordValue = PASSWORD('12345');
Query OK, 1 row affected (0.003 sec)

MariaDB [databaseprogramming]> SELECT passwordValue FROM bif;
+-------------------------------------------+
| passwordValue                             |
+-------------------------------------------+
| *00A51F3F48415C7D4E8908980D443C29C69B60C9 |
+-------------------------------------------+
1 row in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  7. Encryption, Hashing and Compression Functions
     - PASSWORD(str)
       - Calculates and returns a hashed password string from the plaintext password *str*.

```
MariaDB [databaseprogramming]> INSERT INTO bif SET passwordValue = PASSWORD('12345');
Query OK, 1 row affected (0.003 sec)

MariaDB [databaseprogramming]> SELECT passwordValue FROM bif;
+-------------------------------------------+
| passwordValue                             |
+-------------------------------------------+
| *00A51F3F48415C7D4E8908980D443C29C69B60C9 |
+-------------------------------------------+
1 row in set (0.000 sec)
```

```
MariaDB [databaseprogramming]> SELECT IF(PASSWORD('12345')=passwordValue,1,0) FROM bif;
+-----------------------------------------+
| IF(PASSWORD('12345')=passwordValue,1,0) |
+-----------------------------------------+
|                                       1 |
+-----------------------------------------+
1 row in set (0.001 sec)

MariaDB [databaseprogramming]> SELECT IF(PASSWORD('123456')=passwordValue,1,0) FROM bif;
+-----------------------------------------+
| IF(PASSWORD('123456')=passwordValue,1,0) |
+-----------------------------------------+
|                                       0 |
+-----------------------------------------+
1 row in set (0.000 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  7. Encryption, Hashing and Compression Functions
     - https://mariadb.com/kb/en/encode/
     - ENCODE(str, pw_str)
     - DECODE(encoded_str, pw_str)

ENCODE

Encrypts a string.

DECODE

Decrypts a string encoded with ENCODE().

```
MariaDB [databaseprogramming]> INSERT INTO bif SET stringValue="Jaewook";
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> SELECT ENCODE(stringValue, 'jack') FROM bif;
+-----------------------------+
| ENCODE(stringValue, 'jack') |
+-----------------------------+
| 724-꿈?                     |
+-----------------------------+
1 row in set (0.001 sec)

MariaDB [databaseprogramming]> SELECT DECODE(ENCODE(stringValue, 'jack'),'jack') FROM bif;
+-----------------------------------------+
| DECODE(ENCODE(stringValue, 'jack'),'jack') |
+-----------------------------------------+
| Jaewook                                 |
+-----------------------------------------+
1 row in set (0.001 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  7. Encryption, Hashing and Compression Functions
     - COMPRESS
       - returns a binary, compressed string
     - UNCOMPRESS
       - runcompresses string compressed with COMPRESS()

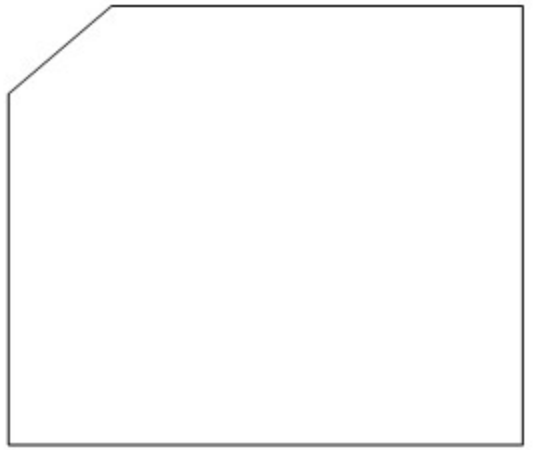
```
MariaDB [databaseprogramming]> SELECT UNCOMPRESS(COMPRESS(stringValue)) FROM bif;
+-----------------------------------+
| UNCOMPRESS(COMPRESS(stringValue)) |
+-----------------------------------+
| Jaewook                           |
+-----------------------------------+
1 row in set (0.001 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  11. Geographic Functions

Point

Line



Polygon

Built-in Functions

- MariaDB Built-in Functions
  - 11. Geographic Functions
    - POINT
      - PointFromText('POINT(n n)')
      - AsText(geoValue)

```
MariaDB [databaseprogramming]> INSERT INTO bif SET geoValue = PointFromText('POINT(10 10)');
Query OK, 1 row affected (0.004 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET geoValue = PointFromText('POINT(20 10)');
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET geoValue = PointFromText('POINT(10 20)');
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET geoValue = PointFromText('POINT(20 20)');
Query OK, 1 row affected (0.002 sec)
```

```
MariaDB [databaseprogramming]> SELECT AsText(geoValue) FROM bif;
+------------------+
| AsText(geoValue) |
+------------------+
| POINT(10 10)     |
| POINT(20 10)     |
| POINT(10 20)     |
| POINT(20 20)     |
+------------------+
4 rows in set (0.001 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  - 11. Geographic Functions
    - POINT
      - get its property
      - X
      - Y
      - distance from Point(0 0)?

```
MariaDB [databaseprogramming]> SELECT AsText(geoValue), X(geoValue), Y(geoValue) FROM bif;
+-----------------+-------------+-------------+
| AsText(geoValue) | X(geoValue) | Y(geoValue) |
+-----------------+-------------+-------------+
| POINT(10 10)     |          10 |          10 |
| POINT(20 10)     |          20 |          10 |
| POINT(10 20)     |          10 |          20 |
| POINT(20 20)     |          20 |          20 |
+-----------------+-------------+-------------+
4 rows in set (0.001 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  11. Geographic Functions
      - Various useful functions

CONTAINS
Whether one geometry contains another.

CROSSES
Whether two geometries spatially cross

DISJOINT
Whether the two elements do not intersect.

EQUALS
Indicates whether two geometries are spatially equal.

INTERSECTS
Indicates whether two geometries spatially intersect.

OVERLAPS
Indicates whether two elements spatially overlap.

ST_CONTAINS
Whether one geometry is contained by another.

ST_CROSSES
Whether two geometries spatially cross.

ST_DIFFERENCE
Point set difference.

ST_DISJOINT
Whether one geometry is spatially disjoint from another.

ST_DISTANCE
The distance between two geometries.

ST_EQUALS
Whether two geometries are spatoially equal.

ST_INTERSECTS
Whether two geometries spatially intersect.

ST_LENGTH
Length of a LineString value.

ST_OVERLAPS
Whether two geometries overlap.

ST_TOUCHES
Whether one geometry g1 spatially touches another.

ST_WITHIN
Whether one geometry is within another.

TOUCHES
Whether two geometries spatially touch.

WITHIN
Indicate whether a geographic element is spacially within another.

Built-in Functions

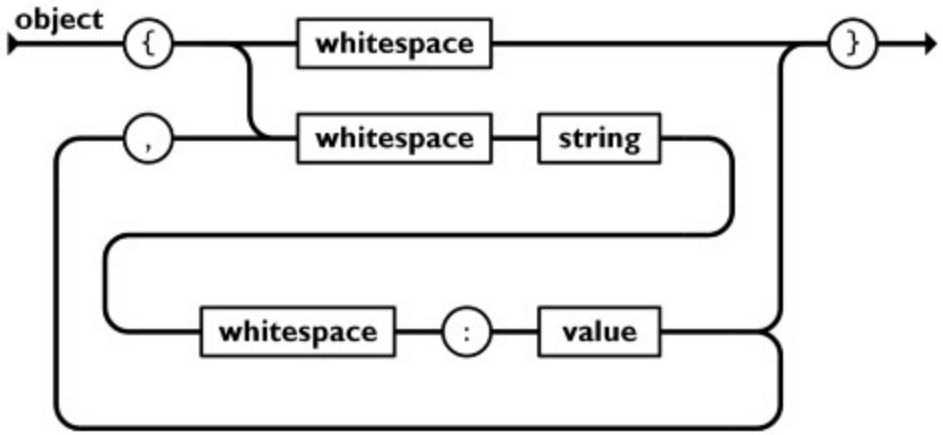
- MariaDB Built-in Functions
  11. Geographic Functions
      - Various useful functions

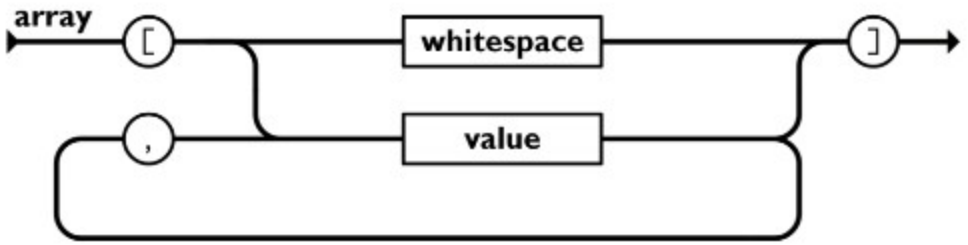
      - Example: https://mariadb.com/kb/en/st_distance/
      - **ST_DISTANCE**
        - ST_DISTANCE(g1,g2)

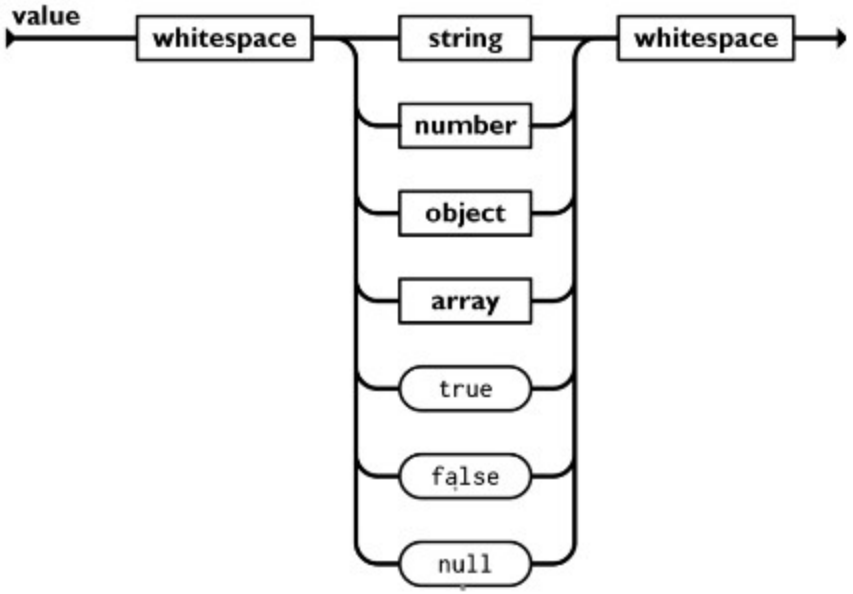
  - **Practice**
    - **DISTANCE FROM Point('100 100');**

Built-in Functions

- MariaDB Built-in Functions
  12. JSON Functions







Built-in Functions

- MariaDB Built-in Functions
  - 12. JSON Functions
    - ADD JSON column
    - with or without
    - validation

```
MariaDB [databaseprogramming]> CREATE TABLE j1 (j JSON);
Query OK, 0 rows affected (0.015 sec)

MariaDB [databaseprogramming]> CREATE TABLE j2 (j JSON CHECK (JSON_VALID(j)));
Query OK, 0 rows affected (0.042 sec)

MariaDB [databaseprogramming]> DESCRIBE j1;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| j     | longtext | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
1 row in set (0.021 sec)

MariaDB [databaseprogramming]> DESCRIBE j2;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| j     | longtext | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
1 row in set (0.007 sec)
```

Built-in Functions

- MariaDB Built-in Functions
  12. JSON Functions
      - Various useful functions

JSON_ARRAY
Returns a JSON array containing the listed values.

JSON_ARRAYAGG
Returns a JSON array containing an element for each value in a given set of JSON or SQL values.

JSON_ARRAY_APPEND
Appends values to the end of the given arrays within a JSON document.

JSON_ARRAY_INSERT
Inserts a value into a JSON document.

JSON_COMPACT
Removes all unnecessary spaces so the json document is as short as possible.

JSON_CONTAINS
Whether a value is found in a given JSON document or at a specified path within the document.

JSON_CONTAINS_PATH
Indicates whether the given JSON document contains data at the specified path or paths.

JSON_DEPTH
Maximum depth of a JSON document.

JSON_TYPE
Returns the type of a JSON value.

JSON_UNQUOTE
Unquotes a JSON value, returning a string.

JSON_VALID
Whether a value is a valid JSON document or not.

JSON_VALUE
Given a JSON document, returns the specified scalar.

JSON_DETAILED
Represents JSON in the most understandable way emphasizing nested structures.

JSON_EXISTS
Determines whether a specified JSON value exists in the given data.

JSON_EXTRACT
Extracts data from a JSON document.

JSON_INSERT
Inserts data into a JSON document.

JSON_KEYS
Returns keys from top-level value of a JSON object or top-level keys from the path.

JSON_LENGTH
Returns the length of a JSON document, or the length of a value within the document.

JSON_LOOSE
Adds spaces to a JSON document to make it look more readable.

JSON_MERGE
Merges the given JSON documents.

JSON_MERGE_PATCH
RFC 7396-compliant merge of the given JSON documents.

JSON_MERGE_PRESERVE
Synonym for JSON_MERGE.

JSON_OBJECT
Returns a JSON object containing the given key/value pairs.

JSON_OBJECTAGG
Returns a JSON object containing key-value pairs.

JSON_QUERY
Given a JSON document, returns an object or array specified by the path.

JSON_QUOTE
Quotes a string as a JSON value.

JSON_REMOVE
Removes data from a JSON document.

JSON_REPLACE
Replaces existing values in a JSON document.

JSON_SEARCH
Returns the path to the given string within a JSON document.

JSON_SET
Updates or inserts data into a JSON document.

Built-in Functions

- MariaDB Built-in Functions
  12. JSON Functions
    - Various useful functions
      - Insert the following inf. As json object

Name	ID	Temperature
“jack”	615458	36.5
“james”	123456	36.7
“kildong”	432156	37.0

- INSERT INTO j2 VALUES ('{ "name" : "jack" , "ID" : 615458, "temperature" : 36.5}');
- INSERT INTO j2 VALUES ('{ "name" : "james" , "ID" : 123456, "temperature" : 36.7}');
- INSERT INTO j2 VALUES ('{ "name" : "james" , "ID" : 432156, "temperature" : 37.0}');

Built-in Functions

- MariaDB Built-in Functions
  12. JSON Functions
    - JSON_VALUE(col_name, path)
      - https://mariadb.com/kb/en/json_value/
    - e.g., JSON_VALUE(j, '$.name');

Name	ID	Temperature
"jack"	615458	36.5
"james"	123456	36.7
"kildong"	432156	37.0

    - Practice show all the temperature values of three rows
      -

Built-in Functions

- MariaDB Built-in Functions
  12. JSON Functions
    - JSON_TYPE(json_value)
    - https://mariadb.com/kb/en/json_type/

Name	ID	Temperature
“jack”	615458	36.5
“james”	123456	36.7
“kildong”	432156	37.0

- Practice show types of all the temperature values of three rows
  -

Built-in Functions

- MariaDB Built-in Functions
  12. JSON Functions
    - JSON_VALID(json_doc or value)
    - https://mariadb.com/kb/en/json_valid/

Name	ID	Temperature
“jack”	615458	36.5
“james”	123456	36.7
“kildong”	432156	37.0

    - Practice show types of all the temperature values of three rows
    -

Built-in Functions

- MariaDB Built-in Functions
  12. JSON Functions
    - JSON_EXISTS(*json_doc*, *path*)
    - https://mariadb.com/kb/en/json_exists/

Name	ID	Temperature
“jack”	615458	36.5
“james”	123456	36.7
“kildong”	432156	37.0

    - Practice: test if the json has name field and name2 field
      -
      -

Built-in Functions

- MariaDB Built-in Functions
  12. JSON Functions
    - JSON_DETAILED(*json_doc*)
    - https://mariadb.com/kb/en/json_detailed/

Name	ID	Temperature
“jack”	615458	36.5
“james”	123456	36.7
“kildong”	432156	37.0

- Practice: beautify the json doc
  -

Built-in Functions

- MariaDB Built-in Functions
  12. JSON Functions
    - JSON_KEYS(*json_doc*)
    - https://mariadb.com/kb/en/json_keys/

Name	ID	Temperature
“jack”	615458	36.5
“james”	123456	36.7
“kildong”	432156	37.0

- Practice: show all keys in a json document for each row
  -

Built-in Functions

- MariaDB Built-in Functions
  12. JSON Functions
    - JSON_LENGTH(*json_doc[, path]*)
    - https://mariadb.com/kb/en/json_length/

Name	ID	Temperature
“jack”	615458	36.5
“james”	123456	36.7
“kildong”	432156	37.0

    - Practice: show all keys in a json document for each row
      -

Built-in Functions

- MariaDB Built-in Functions
  3. Aggregate Functions
     - JSON_OBJECTAGG(key, value)

JSON_OBJECTAGG

Returns a JSON object containing key-value pairs.

```
MariaDB [databaseprogramming]> DELETE FROM bif;
Query OK, 4 rows affected (0.002 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET stringValue='Jack', integerValue=53;
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET stringValue='James', integerValue=33;
Query OK, 1 row affected (0.003 sec)

MariaDB [databaseprogramming]> INSERT INTO bif SET stringValue='Jake', integerValue=43;
Query OK, 1 row affected (0.002 sec)

MariaDB [databaseprogramming]> SELECT stringValue, integerValue FROM bif;
+-------------+--------------+
| stringValue | integerValue |
+-------------+--------------+
| Jack        |           53 |
| James       |           33 |
| Jake        |           43 |
+-------------+--------------+
3 rows in set (0.000 sec)
```

```
MariaDB [databaseprogramming]> SELECT JSON_OBJECTAGG(stringValue, integerValue) FROM bif;
+-----------------------------------------+
| JSON_OBJECTAGG(stringValue, integerValue) |
+-----------------------------------------+
| {"Jack":53, "James":33, "Jake":43}      |
+-----------------------------------------+
1 row in set (0.001 sec)
```

Data Definition Language: Integrity Constraints

- Integrity constraints
  - guarantee to keep data consistency
  - e.g.,
    - account balance cannot be null
    - different accounts cannot have a same account number
    - account number in depositor relation should be matched with that in account relation
    - A salary per hour should exceed $6.00

Data Definition Language: Integrity Constraints

- Integrity constraints
  - Integrity constraints in a single relation
    - not null

account relation

account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
null	Mianus	700

    - unique
    - check (<predicate>)

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...) [table_options    ]... [partition_options]
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)] [table_options    ]... [partition_options]
    select_statement
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
   { LIKE old_table_name | (LIKE old_table_name) }

select_statement:
    [IGNORE | REPLACE] [AS] SELECT ...   (Some legal select statement)
```

```
create_definition:
  { col_name column_definition | index_definition | period_definition | CHECK (expr) }

column_definition:
  data_type
    [NOT NULL | NULL] [DEFAULT default_value | (expression)]
    [ON UPDATE [NOW | CURRENT_TIMESTAMP] [(precision)]]
    [AUTO_INCREMENT] [ZEROFILL] [UNIQUE [KEY] | [PRIMARY] KEY]
    [INVISIBLE] [{WITH|WITHOUT} SYSTEM VERSIONING]
    [COMMENT 'string'] [REF_SYSTEM_ID = value]
    [reference_definition]
  | data_type [GENERATED ALWAYS]
  AS { { ROW {START|END} } | { (expression) [VIRTUAL | PERSISTENT | STORED] } }
      [UNIQUE [KEY]] [COMMENT 'string']

constraint_definition:
   CONSTRAINT [constraint_name] CHECK (expression)
```

Data Definition Language: Integrity Constraints

- Integrity constraints
  - Integrity constraints in a single relation
    - not null
    - unique
      - should be unique but can be null

account relation

account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
null OK	Brighton	900
null	Mianus	700
A-102 X	Perryridge	300

```
CREATE TABLE account (
        account_number char(10) UNIQUE,
        branch_name VARCHAR(50),
        balance INTEGER not null
);
```

- check (<predicate>)

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...) [table_options    ]... [partition_options]
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)] [table_options    ]... [partition_options]
    select_statement
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
   { LIKE old_table_name | (LIKE old_table_name) }

select_statement:
    [IGNORE | REPLACE] [AS] SELECT ...   (Some legal select statement)
```

```
create_definition:
  { col_name column_definition | index_definition | period_definition | CHECK (expr) }

column_definition:
  data_type
    [NOT NULL | NULL] [DEFAULT default_value | (expression)]
    [ON UPDATE [NOW | CURRENT_TIMESTAMP] [(precision)]]
    [AUTO_INCREMENT] [ZEROFILL] [UNIQUE [KEY] | [PRIMARY] KEY]
    [INVISIBLE] [{WITH|WITHOUT} SYSTEM VERSIONING]
    [COMMENT 'string'] [REF_SYSTEM_ID = value]
    [reference_definition]
  | data_type [GENERATED ALWAYS]
  AS { { ROW {START|END} } | { (expression) [VIRTUAL | PERSISTENT | STORED] } }
      [UNIQUE [KEY]] [COMMENT 'string']

constraint_definition:
   CONSTRAINT [constraint_name] CHECK (expression)
```

Data Definition Language: Integrity Constraints

- Integrity constraints
  - Integrity constraints in a single relation
    - not null
    - unique
    - check (<predicate>)

```
CREATE TABLE account (
        account_number char(10) null,
        branch_name VARCHAR(50),
        balance INTEGER not null,
        CHECK ( balance >= 0 )
);
```

```
CREATE TABLE student (
        name      VARHCAR(50) not null,
        student_id          VARCHAR(50),
        degree_level        VARCHAR(50),
        PRIMARY KEY (student_id),
        CHECK ( degree_level in ('Bachelors','Masters','Doctorate'))
);
```

Data Definition Language: Integrity Constraints

- Integrity constraints
  - Integrity constraints in a single relation
    - not null
    - unique
    - check (<predicate>)

a	b
1	2
3	4
5	6

c	d
1	5
3	8
4	9

```
CREATE TABLE x (
        a       INTEGER,
        b       INTEGER
);
```

```
CREATE TABLE y (
        c       INTEGER,
        d       INTEGER,
        CHECK ( c in ( SELECT a FROM x))
);
```

```
INSERT INTO x VALUES (1,2), (3,4), (5,6);

INSERT INTO y VALUES (1,5), (3,8), (4,9);
```

Data Definition Language: Integrity Constraints

- Integrity constraints
  - Integrity constraints in a single relation
    - not null
    - unique
    - check (<predicate>)
    - primary key
      - A candidate key selected by database manager
      - A table must have at most one primary key and implicitly NOT NULL

Student_schema = (<u>student_identifier</u>, citizen_identifier, student_name, department)

```
CREATE OR REPLACE TABLE student (
        student_identifier VARCHAR(50),
        citizen_identifier VARCHAR(50),
        student_name VARCHAR(50),
        department VARCHAR(50),
        PRIMARY KEY (student_identifier)
);
```

```
INSERT INTO student VALUES ('1','a','Jack','CS');
                                    ?
INSERT INTO student VALUES ('1','b','James','CS');
                                    ?
INSERT INTO student VALUES (null,'b','James','CS');
                                    ?
```

Data Definition Language: Integrity Constraints

- Integrity constraints
  - Integrity constraints in a single relation
    - not null
    - unique
    - check (<predicate>)
    - primary key
    - foreign key
      - A key to link two relations
      - A key of a relation R is a foreign key if the key is a primary key of other relation S
      - R: referencing relation
      - S: referenced relation
      - R and S can be same
      - should exist in referenced relation or null

Student_schema = (<u>student_identifier</u>, citizen_identifier, student_name, department)

Referencing table

Person_schema = (<u>citizen_identifier</u>, age, address)

Referenced table

```
CREATE OR REPLACE TABLE person (citizen_identifier
VARCHAR(50), age INTEGER, address VARCHAR(50), PRIMARY
KEY (citizen_identifier));
insert into person VALUES ('a', 50, 'abc');
```

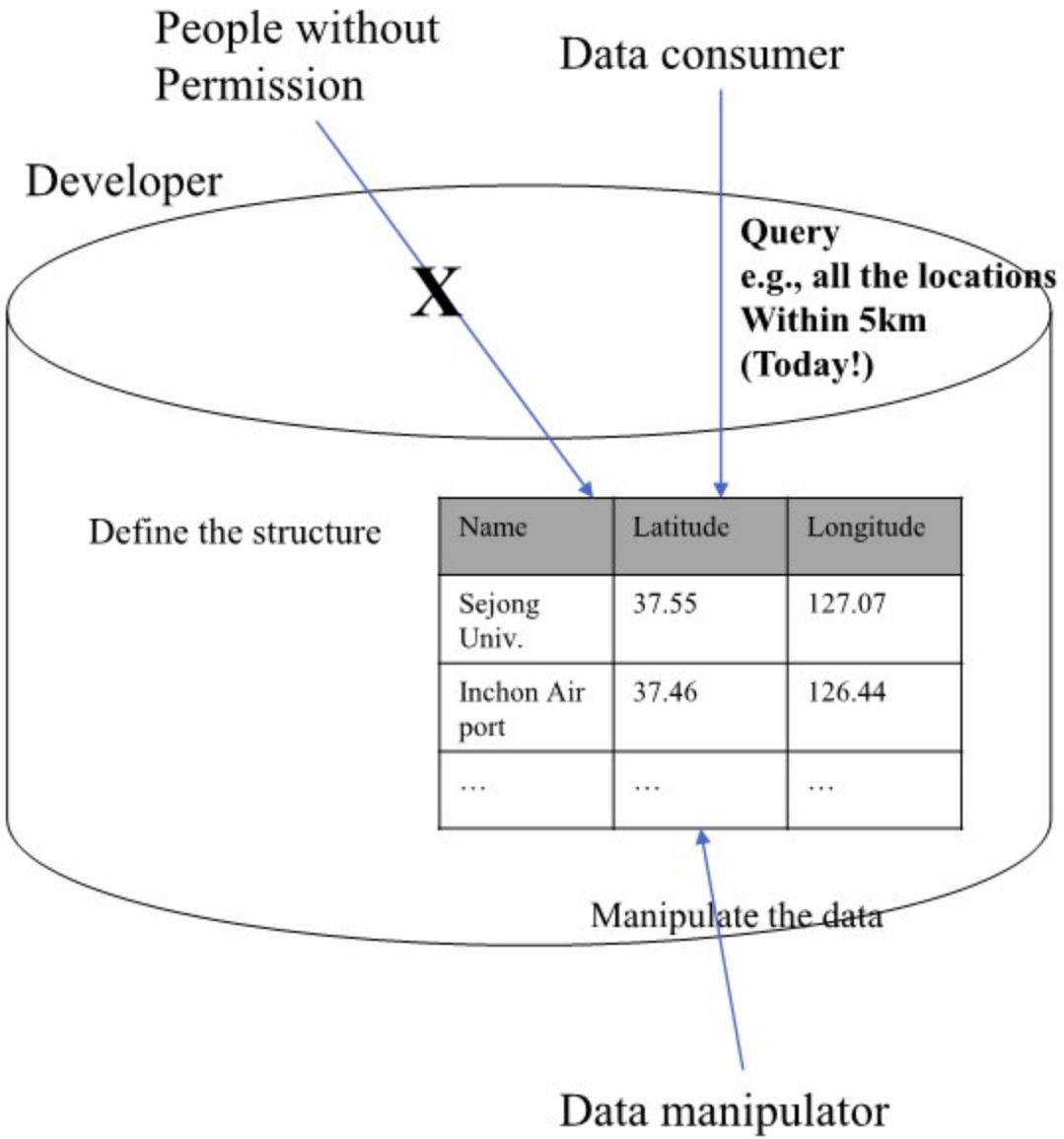
```
CREATE OR REPLACE TABLE student (
        student_identifier VARCHAR(50),
        citizen_identifier VARCHAR(50),
        student_name VARCHAR(50),
        department VARCHAR(50),
        PRIMARY KEY (student_identifier),
        FOREIGN KEY (citizen_identifier) REFERENCES person (citizen_identifier)
);
INSERT INTO student VALUES ('1', 'a', 'Jack', 'CS');
                    ?
INSERT INTO student VALUES ('2', 'c', 'Jack', 'CS');
                    ?
INSERT INTO student VALUES ('3', null, "James", "SW");
```

Data Query Language

- SQL
  - Data definition
  - Data manipulation
  - **Data query**
  - Data control

Example: geo data

Sejong Univ., 37.55, 127.07
Incheon Airport, 37.46, 126.44
…



Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE (https://mariadb.com/kb/en/select/#select-expressions)
  - Projection
    - The name of a column
    - Any expression using functions
    - * to select all columns
    - Can explicitly set a table
      - loan.*
      - loan.loan_number
    - etc.

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
      [WHERE where_condition]
      [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_condition]
      [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      procedure|[PROCEDURE procedure_name(argument_list)]
      [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] [export_options]

 INTO DUMPFILE 'file_name'  INTO var_name [, var_name] ]


      [[FOR UPDATE | LOCK IN SHARE MODE] [WAIT n | NOWAIT] ] ]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE (https://mariadb.com/kb/en/select/#select-expressions)
  - Projection
    - The name of a column
    - Any expression using functions
    - * to select all columns
    - Can explicitly set a table
      - loan.*
      - loan.loan_number
    - DISTINCT
    - etc.

- PROJECT, π (\pi)
  - Syntax
    - $\pi_{attr1,attr2\ldots}(relation)$
  - Projects the list of attributes of interest to be appeared in the result
  - **Note:** a relation is a set of tuples (no redundancy)

loan relation

loan_number	branch_name	Amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

$\pi_{branch\_name}(loan)$

$\pi_{branch\_name}(loan)$

branch_name
Round Hill
Downtown
Perryridge
Redwood
Mianus

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE (https://mariadb.com/kb/en/select/#select-expressions)
  - ALL vs. DISTINCT or DISTINCTROW

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
      [WHERE where_condition]
      [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_condition]
      [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      procedure|[PROCEDURE procedure_name(argument_list)]
      [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] [export_options]

 INTO DUMPFILE 'file_name'  INTO var_name [, var_name] ]


      [[FOR UPDATE | LOCK IN SHARE MODE] [WAIT n | NOWAIT] ] ]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE (https://mariadb.com/kb/en/select/#select-expressions)
  - ALL vs. DISTINCT or DISTINCTROW
    - CREATE TABLE student (id INT, semester INT, name VARCHAR(50));
    - INSERT INTO student VALUES (1,1,'J'), (2,2,'K'), (3,1,'M'), (4,2, 'P'), (5,3, 'K'), (6,1,'J');

```
MariaDB [db]> SELECT * FROM student;
+------+----------+------+
| id   | semester | name |
+------+----------+------+
|    1 |        1 | J    |
|    2 |        2 | K    |
|    3 |        1 | M    |
|    4 |        2 | P    |
|    5 |        3 | K    |
|    6 |        1 | J    |
+------+----------+------+
6 rows in set (0.000 sec)
```

```
MariaDB [db]> SELECT semester FROM student;
+----------+
| semester |
+----------+
|        1 |
|        2 |
|        1 |
|        2 |
|        3 |
|        1 |
+----------+
6 rows in set (0.000 sec)
```

```
MariaDB [db]> SELECT DISTINCT semester FROM student;
+----------+
| semester |
+----------+
|        1 |
|        2 |
|        3 |
+----------+
3 rows in set (0.000 sec)
```

```
MariaDB [db]> SELECT DISTINCTROW semester FROM student;
+----------+
| semester |
+----------+
|        1 |
|        2 |
|        3 |
+----------+
3 rows in set (0.000 sec)
```

```
MariaDB [db]> SELECT semester, name FROM student;
+----------+------+
| semester | name |
+----------+------+
|        1 | J    |
|        2 | K    |
|        1 | M    |
|        2 | P    |
|        3 | K    |
|        1 | J    |
+----------+------+
6 rows in set (0.000 sec)
```

```
MariaDB [db]> SELECT DISTINCT semester, name FROM student;
+----------+------+
| semester | name |
+----------+------+
|        1 | J    |
|        2 | K    |
|        1 | M    |
|        2 | P    |
|        3 | K    |
+----------+------+
5 rows in set (0.000 sec)
```

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE (https://mariadb.com/kb/en/select/#select-expressions)
  - ALIAS
    - field AS alias
    - field AS ‘alias’ if alias includes empty space, special characters
    - CREATE TABLE student (id INT, semester INT, name VARCHAR(50));
    - INSERT INTO student VALUES (1,1,'J'), (2,2,'K'), (3,1,'M'), (4,2, 'P'), (5,3, 'K'), (6,1,'J');

```
MariaDB [db]> SELECT id AS identifier FROM student;
+------------+
| identifier |
+------------+
|          1 |
|          2 |
|          3 |
|          4 |
|          5 |
|          6 |
+------------+
6 rows in set (0.000 sec)
```

```
MariaDB [db]> SELECT id AS 'student identifier' FROM student;
+--------------------+
| student identifier |
+--------------------+
|                  1 |
|                  2 |
|                  3 |
|                  4 |
|                  5 |
|                  6 |
+--------------------+
6 rows in set (0.000 sec)
```

```
MariaDB [db]> SELECT id AS '!i' FROM student;
+------+
| !i   |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
+------+
6 rows in set (0.000 sec)
```

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - Warming up

Retrieve all the records from loan;	loan
Retrieve all the loan_number from loan; if loan_number is a candidate key	$\pi_{loan\_number}(loan)$
Retrieve all the loan_number, branch_name FROM loan;	$\pi_{loan\_number,branch\_name}(loan)$
Retrieve all the non-redundant branch_name FROM loan;	$\pi_{branch\_name}(loan)$

Relation algebra's output is a **set** of relations

DISTINCT

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - where_condition (Filter)
    - =
    - >
    - >=
    - <
    - <=
    - !=
    - AND (&&)
    - OR (||)
    - NOT (!)
    - BETWEEN AND
    - IN
    - +,-,*,/,%
    - Functions
    - IS NULL
    - IS NOT NULL

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
      [WHERE where_condition]
      [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_condition]
      [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      procedure|[PROCEDURE procedure_name(argument_list)]
      [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] [export_options]

 INTO DUMPFILE 'file_name'  INTO var_name [, var_name] ]


      [[FOR UPDATE | LOCK IN SHARE MODE] [WAIT n | NOWAIT] ] ]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```

Data Query Language: more operators

- Data Query Language
  - Operators
    - https://mariadb.com/kb/en/operators/

Arithmetic Operators

Addition Operator (+)
Addition.

DIV
Integer division.

Division Operator (/)
Division.

MOD
Modulo operation. Remainder of N d

Modulo Operator (%)
Modulo operator. Returns the remain

Multiplication Operator (*)
Multiplication.

Subtraction Operator (-)
Subtraction and unary minus.

Assignment Operators

Assignment Operator (:=)
Assignment operator for assigning a value.

Assignment Operator (=)
The equal sign as an assignment operator.

Logical Operators

!
Logical NOT.

&&
Logical AND.

XOR
Logical XOR.

||
Logical OR.

Comparison Operators

!=
Not equal operator.

<
Less than operator.

<=
Less than or equal operator.

<=>
NULL-safe equal operator.

=
Equal operator.

>
Greater than operator.

>=
Greater than or equal operator.

BETWEEN AND
True if expression between two values.

COALESCE
Returns the first non-NULL parameter

GREATEST
Returns the largest argument.

IN
True if expression equals any of the val

INTERVAL
Index of the argument that is less than t

IS
Tests whether a boolean is TRUE, FALS

IS NOT
Tests whether a boolean value is not TR

IS NOT NULL
Tests whether a value is not NULL.

IS NULL
Tests whether a value is NULL.

ISNULL
Checks if an expression is NULL.

LEAST
Returns the smallest argument.

NOT BETWEEN
Same as NOT (expr BETWEEN min AND max)

NOT IN
Same as NOT (expr IN (value,...))

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - where_condition (Filter)
    - =, >, >=, <, <=, !=
    - AND(&&), OR(||), NOT(!)
    - +,-,*,/,%
    - Functions
    - IS (NOT) NULL

- SELECT, σ (\sigma)
  - Syntax
    - $\sigma_{predicate}(relation)$
  - Select tuples that satisfy a given predicate using
    - $=, \neq, <, \leq, >, \geq$ and
    - $AND\ (\wedge), OR\ (\vee), NOT(\neg)$

loan relation

loan_number	branch_name	Amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

$\sigma_{branch\_name="Perryridge"}(loan)$

$\sigma_{branch\_name="Perryridge"}(loan)$

loan_number	branch_name	Amount
L-15	Perryridge	1500
L-16	Perryridge	1300

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - where_condition (Filter)
    - =, >, >=, <, <=, !=
    - AND(&&), OR, NOT
    - +,-,*,/,%
    - Functions
    - IS (NOT) NULL

- SELECT, σ (\sigma)
  - Syntax
    - $\sigma_{predicate}(relation)$
  - Select tuples that satisfy a given predicate using
    - $=, \neq, <, \leq, >, \geq$ and
    - $AND\ (\wedge), OR\ (\vee), NOT(\neg)$

loan2 relation

loan_number	branch_name	amount	date
L-11	Round Hill	900	2019-03-29
L-14	Downtown	1500	2019-03-30
L-15	Perryridge	1500	2019-08-02
L-16	Perryridge	1300	2019-09-05
L-17	Downtown	1000	2020-01-13
L-23	Redwood	2000	2020-02-01
L-93	Mianus	500	2021-03-02

INSERT INTO loan2 VALUES ('L-11', 'Round Hill', 900, '2019-03-29')
it works!

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - Filtering

$\sigma_{loan\_number="L-11"}(loan2)$
$\sigma_{amount>1000}(loan2)$
$\sigma_{amount>(800*2)}(loan2)$
$\sigma_{YEAR(date)=2020}(loan2)$
$\sigma_{MONTH(date)=8}(loan2)$
$\sigma_{branch\_name="Perryridge" \wedge amount<1400}(loan2)$
$\pi_{loan\_number}(\sigma_{DAYOFMONTH(date)=29}(loan2))$

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
    - Filtering
        - LIKE keyword
            - % matches any number of characters, including zero.
            - _ matches any single character.

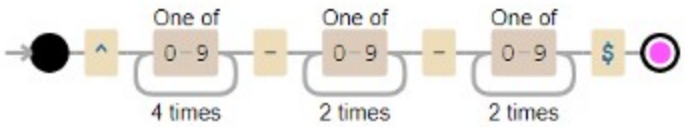
$\sigma_{loan\_number\ LIKE\ "L-1\_"}\ (loan2)$
$\sigma_{branch\_name\ LIKE\ "\%Hill"}\ (loan2)$
$\sigma_{date\ LIKE\ "\%-03-\%"}\ (loan2)$
$\sigma_{date\ LIKE\ "\%-03-\%"\ \vee date\ LIKE"\%-08-\%"}\ (loan2)$

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - Filtering
    - REGEXP string built-in function
      - ^ indicates the start point
      - $ indicates the end point
      - [] indicates one of comma separated values or consecutive values by –
        - e.g., [0-9] <- one of number from 0 to 9
        - e.g., [0,1] <- one of 0, 1
        - e.g., [0-9a-zA-Z] <- one of number or alphabet
      - []+ one or more
      - []* zero or more
      - {n} n times
    - ^[0-9]{4}-[0-9]{2}-[0-9]{2}$

‘^[0,9,2]*$’

- SELECT * FROM loan2 WHERE date REGEXP '^[0-9]{4}-[0-9]{2}-[0-9]{2}$';
  - https://www.debuggex.com/



Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - UNION
    - Syntax

```
SELECT ... UNION [ALL | DISTINCT] SELECT ... [UNION [ALL | DISTINCT] SELECT ...]
[ORDER BY [column [, column ...]]] [LIMIT {[offset,] row_count | row_count OFFSET offset}] ,
```

- UNION, ∪ (\cup)
  - Syntax
    - $relation_1 \cup relation_2$
  - Returns all the tuples that are either in two relations
  - Two relations to be merged must have
    - the identical number of attributes (degree)
    - domain of each attribute on two relations is identical
  - **Note:** the result is also a relation of a set of tuples (no redundancy)

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

$\pi_{customer\_name}(borrower) \cup \pi_{customer\_name}(depositor)$

depositor relation

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - UNION

```
SELECT ... UNION [ALL | DISTINCT] SELECT ... [UNION [ALL |
DISTINCT] SELECT ...] [ORDER BY [column [, column ...]]] [LIMIT
{[offset,] row_count | row_count OFFSET offset}],
```

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

depositor relation

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

$\pi_{customer\_name}(borrower) \cup \pi_{customer\_name}(depositor)$
$\pi_{customer\_name}(borrower) \cup \pi_{customer\_name}(depositor)$, with redundancy
$\pi_{customer\_name}(borrower) \cup \pi_{customer\_name}(depositor)$

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - INTERSECT
    - Syntax

```
SELECT ... (INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] |
UNION [ALL | DISTINCT]) SELECT ...
[(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] | UNION [ALL
| DISTINCT]) SELECT ...]
[ORDER BY [column [, column ...]]] [LIMIT {[offset,] row_count | row_count
OFFSET offset}]
```

- INTERSECTION, ∩ (\cap)
  - Syntax
    - $relation_1 \cap relation_2$
  - Returns all the tuples that are in both two relations
  - Two relations to be merged must have
    - the identical number of attributes (degree)
    - domain of each attribute on two relations is identical
  - **Note:** the result is also a relation of a set of tuples (no redundancy)

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

$\pi_{customer\_name}(borrower) \cap \pi_{customer\_name}(depositor)$

?

depositor relation

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - INTERSECT

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

depositor relation

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

$$\pi_{customer\_name}(borrower) \cap \pi_{customer\_name}(depositor)$$

INTERSECT
INTERSECT ALL
INTERSECT DISTINCT
are all same for above query
Which examples make it different?

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - EXCEPT
    - Syntax

```
SELECT ...
EXCEPT [ALL | DISTINCT] SELECT ...
[(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] | UNION
[ALL | DISTINCT]) SELECT ...]
 [ORDER BY [column [, column ...]]] [LIMIT {[offset,] row_count |
row_count OFFSET offset}]
```

- DIFFERENCE, −
  - Syntax
    - $relation_1 - relation_2$
  - Returns all the tuples that are in a relation 1 but not in a relation 2
  - Two relations to be merged must have
    - the identical number of attributes (degree)
    - domain of each attribute on two relations is identical
  - **Note:** the result is also a relation of a set of tuples (no redundancy)

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

$\pi_{customer\_name}(borrower) - \pi_{customer\_name}(depositor)$

?

depositor relation

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - EXCEPT

```
SELECT ...
EXCEPT [ALL | DISTINCT] SELECT ...
[(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] | UNION
[ALL | DISTINCT]) SELECT ...]
 [ORDER BY [column [, column ...]]] [LIMIT {[offset,] row_count |
row_count OFFSET offset}]
```

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

depositor relation

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

$\pi_{customer\_name}(borrower) - \pi_{customer\_name}(depositor)$
$\pi_{customer\_name}(depositor) - \pi_{customer\_name}(borrower)$

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - ORDER BY (SORTING)
    - col_name
    - ASC | DESC
    - not supported in Relational algebra
      - SET: no order
        no redundancy

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
      [WHERE where_condition]
      [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_condition]
      [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      procedure|[PROCEDURE procedure_name(argument_list)]
      [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] [export_options]

 INTO DUMPFILE 'file_name'  INTO var_name [, var_name] ]


      [[FOR UPDATE | LOCK IN SHARE MODE] [WAIT n | NOWAIT] ] ]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - Sorting
    - ORDER BY col_name ASC|DESC, col_name ASC|DESC, ...

Sort date in an ascending order from $\pi_{loan\_number,date}(loan2)$
Sort date in a descending order from $\pi_{loan\_number,date}(loan2)$
Sort date in an ascending order based on its month from $\pi_{loan\_number,date}(loan2)$
Sort date based on its month (DESC) and its year (DESC) for the same month from $\pi_{loan\_number,date}(loan2)$

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - LIMIT
    - OFFSET row_count

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
      [WHERE where_condition]
      [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_condition]
      [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      procedure|[PROCEDURE procedure_name(argument_list)]
      [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] [export_options]

 INTO DUMPFILE 'file_name'  INTO var_name [, var_name] ]


      [[FOR UPDATE | LOCK IN SHARE MODE] [WAIT n | NOWAIT] ] ]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - Sorting
    - ORDER BY col_name ASC|DESC, col_name ASC|DESC, … LIMIT (offset,) row_count

CHECK the followings
SELECT loan_number, date FROM loan2 ORDER BY MONTH(date) DESC, YEAR(date) DESC LIMIT 1;
SELECT loan_number, date FROM loan2 ORDER BY MONTH(date) DESC, YEAR(date) DESC LIMIT 2;
SELECT loan_number, date FROM loan2 ORDER BY MONTH(date) DESC, YEAR(date) DESC LIMIT 3;
SELECT loan_number, date FROM loan2 ORDER BY MONTH(date) DESC, YEAR(date) DESC LIMIT 0,3;
SELECT loan_number, date FROM loan2 ORDER BY MONTH(date) DESC, YEAR(date) DESC LIMIT 1,3;
SELECT loan_number, date FROM loan2 ORDER BY MONTH(date) DESC, YEAR(date) DESC LIMIT 2,3;
SELECT loan_number, date FROM loan2 ORDER BY MONTH(date) DESC, YEAR(date) DESC LIMIT 3,3;

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - LIMIT
    - Pagination
      - Assume that you have billion records, it is too burden to give whole records to million clients
      - IDEA
        - Return a limit of records and continue

```
MariaDB [db]> insert into n VALUES (1),(2),(3),(4),(5),(6),(7),(8),(9),(10),(11),(12),(13);
Query OK, 13 rows affected (0.021 sec)
Records: 13  Duplicates: 0  Warnings: 0

MariaDB [db]> select * from n;
+------+
| n    |
+------+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
|    7 |
|    8 |
|    9 |
|   10 |
|   11 |
|   12 |
|   13 |
+------+
```

Data Query Language: SELECT FROM TABLE

- Aggregation function, $\mathcal{G}$, \scriptG
  - Syntax
    - $\mathcal{G}_{F_1(A_1),F_2(A_2),\ldots,F_n(A_n)}(r)$
    - F: aggregation function (e.g., avg, count, count-distinct, min, max)
    - A: attributes
  - Takes a set of attribute values and return a single value as a result

- Example

pt_works relation

employee_name	branch_name	salary
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Rao	Austin	1500
Sato	Austin	1600

$\mathcal{G}_{count-distance(branch\_name)}$(pt_works)

$count-distinct(branch\_name)$
3

$\mathcal{G}_{sum(salary)}$(pt_works)

$sum(salary)$
16500

Sum of salary for each branch?

Data Query Language: SELECT FROM TABLE

- Aggregate Functions
  - AVG MAX MIN STD SUM VARIANCE

pt_works relation

employee_name	branch_name	salary
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Rao	Austin	1500
Sato	Austin	1600

$\mathcal{G}_{sum(salary)}$(pt_works)

$sum(salary)$
16500

```
MariaDB [db]>
+-------------+
| SUM(salary) |
+-------------+
|       16500 |
+-------------+
1 row in set (0.000 sec)
```

Data Query Language: SELECT FROM TABLE

- Aggregate Functions
  - JSON_ARRAYAGG

```
CREATE TABLE a (a int);

INSERT INTO a VALUES (1),(2),(3);

SELECT JSON_ARRAYAGG(a) FROM a;
```

a relation

a
1
2
3

```
+-----------------+
| JSON_ARRAYAGG(a) |
+-----------------+
| [1,2,3]         |
+-----------------+
```

```
SELECT JSON_CONTAINS(JSON_ARRAYAGG(a),1,'$')
FROM a;
```

```
+-----------------------------------+
| JSON_CONTAINS(JSON_ARRAYAGG(a),1,'$') |
+-----------------------------------+
|                                 1 |
+-----------------------------------+
```

Data Query Language: SELECT FROM TABLE

- Aggregate Functions
  - COUNT

COUNT_DISTINCT

COUNT

Returns count of non-null values.

COUNT DISTINCT

Returns count of number of different non-NULL values.

Sum of salary for each branch?

pt_works relation

employee_name	branch_name	salary
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Rao	Austin	1500
Sato	Austin	1600

$\mathcal{G}_{count-distance(branch\_name)}$(pt_works)

$count-distinct(branch\_name)$
3

```
MariaDB [db]> SELECT COUNT(branch_name) FROM pt_works;
+--------------------+
| COUNT(branch_name) |
+--------------------+
|                  8 |
+--------------------+
1 row in set (0.000 sec)

MariaDB [db]> SELECT COUNT(DISTINCT branch_name) FROM pt_works;
+-----------------------------+
| COUNT(DISTINCT branch_name) |
+-----------------------------+
|                           3 |
+-----------------------------+
1 row in set (0.000 sec)
```

Data Query Language: SELECT FROM TABLE

- Aggregation function, $\mathcal{G}$, \scriptG
  - Syntax
    - $_{G_1,G_2,\ldots,G_m}\mathcal{G}_{F_1(A_1),F_2(A_2),\ldots,F_n(A_n)}(r)$
    - G: a list of attributes to be grouped
    - F: aggregation function (e.g., avg, count, count-distinct, min, max)
    - A: attribute name
  - Degree: m+n
  - Takes a set of attribute values and return a single value as a result
- Example

pt_works relation grouped by branch_name

employee_name	branch_name	salary
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Rao	Austin	1500
Sato	Austin	1600

$_{branch\_name}\mathcal{G}_{count-distinct(branch\_name)}(pt\_works)$

branch_name	$count-distinct(branch\_name)$
Perryridge	3
Downtown	3
Austin	2

$_{branch\_name}\mathcal{G}_{\text{count}(branch\_name)}(pt\_works)$

branch_name	$\text{count}(branch\_name)$
Perryridge	3
Downtown	3
Austin	2

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - GROUP BY (Aggregation)
    - col_name
    - ASC|DESC
    - ,
    - use group operation in select_expr

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
      [WHERE where_condition]
      [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_condition]
      [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      procedure|[PROCEDURE procedure_name(argument_list)]
      [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] [export_options]

 INTO DUMPFILE 'file_name'  INTO var_name [, var_name] ]


      [[FOR UPDATE | LOCK IN SHARE MODE] [WAIT n | NOWAIT] ] ]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - GROUP BY (Aggregation)
    - col_name ASC|DESC,
    - use group operation in select_expr

```
MariaDB [db]> SELECT branch_name, AVG(salary) FROM pt_works GROUP BY branch_name;
+-------------+--------------------+
| branch_name | AVG(salary)        |
+-------------+--------------------+
| Austin      |               1550 |
| Downtown    | 1766.6666666666667 |
| Perryridge  |               2700 |
+-------------+--------------------+
3 rows in set (0.000 sec)
```

Is it possible in relational algebra? two or more groups?

$\text{branch\_}name\mathcal{G}_{AVG(salary)}(pt\_works)$
Show branch_name and the average salary of ft_works grouped by branch_name
Find the branch where its average salary is larger than 2000

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - HAVING (Filter)
    - =
    - >
    - >=
    - <
    - <=
    - !=
    - AND
    - OR
    - +,-,*,/,%
    - Functions

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
      [WHERE where_condition]
      [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_condition]
      [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      procedure|[PROCEDURE procedure_name(argument_list)]
      [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] [export_options]

 INTO DUMPFILE 'file_name'  INTO var_name [, var_name] ]


      [[FOR UPDATE | LOCK IN SHARE MODE] [WAIT n | NOWAIT] ] ]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```

Data Query Language: SELECT FROM TABLE

- SELECT FROM TABLE
  - HAVING (Filter)
    - = > >= < <= != AND OR
    - + - * / % Functions

Find the branch grouped by branch_name where its average amount is larger than or equal to 1000
Find the average amount of each branch if the number of branch is larger than 1

Data Query Language: JOIN

- Data Query Language
  - JOIN
    - Syntax

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
```

```
table_references:
    table_reference [, table_reference] ...

table_reference:
    table_factor
  | join_table
```

```
join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor
```

```
join_condition:
    ON conditional_expr
  | USING (column_list)
```

Data Query Language: JOIN

- Data Query Language: JOIN
  - CARTESIAN PRODUCT, × (\times)
    - Syntax
      - $relation_1 \times relation_2$
      - $(relation_1.attr_1, relation_1.attr_2, \dots) \times (relation_2.attr_1, relation_2.attr_2, \dots)$
    - Returns a cross product of two relations
      - A cross product: combination of each tuple in a relation 1 with each tuple in a relation 2
      - $(relation_1.attr_1, relation_1.attr_2, \dots, relation_2.attr_1, relation_2.attr_2, \dots)$
        - Degree = degree of a relation 1 + degree of a relation 2
        - Cardinality = cardinality of a relation 1 * cardinality of a relation 2

Note: you can explicitly state an attribute in a specific relation

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

$borrower \times loan$

?

Degree?
Cardinality?

loan relation

loan_number	branch_name	Amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

Data Query Language: JOIN

- Operators of relational algebra
  - CARTESIAN PRODUCT, × (\times)
    - Practice

```
join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor
```

borrower × *loan* relation

customer_name	borrower.loan_number	loan.loan_number	branch_name	Amount
Adams	L-16	L-11	Round Hill	900
Adams	L-16	L-14	Downtown	1500
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Adams	L-16	L-17	Downtown	1000
Adams	L-16	L-23	Redwood	2000
Adams	L-16	L-93	Mianus	500
Curry	L-93	L-11	Round Hill	900
Curry	L-93	L-14	Downtown	1500
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Curry	L-93	L-17	Downtown	1000
Curry	L-93	L-23	Redwood	2000
Curry	L-93	L-93	Mianus	500
…	…	…	…	….

Data Query Language: JOIN

- Operators of relational algebra
  - CARTESIAN PRODUCT, × (\times)
    - Practice
      - Goal: Finds all the customer names who has a loan in 'Perryridge' branch

1. Finds all the cross product information in both relations where the branch name is Perryridge

$$\sigma_{branch\_name='Perryridge'}(borrower \times loan)$$

customer_name	borrower.loan_number	loan.loan_number	branch_name	Amount
Adams	L-16	L-15	Perryridge	1500
Adams	L-16	L-16	Perryridge	1300
Curry	L-93	L-15	Perryridge	1500
Curry	L-93	L-16	Perryridge	1300
Hayes	L-15	L-15	Perryridge	1500
Hayes	L-15	L-16	Perryridge	1300
Jackson	L-14	L-15	Perryridge	1500
Jackson	L-14	L-16	Perryridge	1300
Jones	L-17	L-15	Perryridge	1500
Jones	L-17	L-16	Perryridge	1300
Smith	L-11	L-15	Perryridge	1500
Smith	L-11	L-16	Perryridge	1300
Smith	L-23	L-15	Perryridge	1500
Smith	L-23	L-16	Perryridge	1300
Williams	L-17	L-15	Perryridge	1500
Williams	L-17	L-16	Perryridge	1300

Data Query Language: JOIN

- Operators of relational algebra
  - CARTESIAN PRODUCT, × (\times)
    - Practice
      - Goal: Finds all the customer names who has a loan in 'Perryridge' branch

2. Then, select tuples if two loan numbers are matched (valid)

$$\sigma_{borrower.loan\_number=loan.loan\_number}(\sigma_{branch\_name="Perryridge"}(borrower \times loan))$$

customer_name	borrower.loan_number	loan.loan_number	branch_name	Amount
Adams	L-16	L-16	Perryridge	1300
Hayes	L-15	L-15	Perryridge	1500

3. Project customer name(s) = goal

$$\pi_{customer\_name}(\sigma_{borrower.loan\_number=loan.loan\_number}(\sigma_{branch\_name="Perryridge"}(borrower \times loan)))$$

customer_name
Adams
Hayes

Data Query Language: JOIN

- Operators of relational algebra
  - RENAME, ρ (\rho)
    - Syntax
      - $\rho_{renamed\_relation}(original\_relation)$
    - Rename the name of original relation to renamed_relation
  - Practice 1
    - Goal: Find the maximum balance in account relation

account relation

account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Data Query Language: JOIN

- Operators of relational algebra
  - RENAME, ρ (\rho)
    - Syntax
      - $\rho_{renamed\_relation}(original\_relation)$
    - Rename the name of original relation to renamed_relation
  - Practice 1
    - Goal: Find **the maximum balance** in account relation
    - Step1 : List all the cross product of account relation

```
table_references:
    table_reference [, table_reference] ...

table_reference:
    table_factor
  | join_table
```

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
```

```
join_condition:
    ON conditional_expr
  | USING (column_list)
```

```
join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor
```

table_reference:
    table_name AS alias
    …

$\rho_{a_1}(account)$

account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

$\rho_{a_1}(account) \times \rho_{a_2}(account)$

?

$\rho_{a_2}(account)$

account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Data Query Language: JOIN

- Operators of relational algebra
  - RENAME, ρ (\rho)
  - Practice 1
    - Step1: List all the cross product of account relation

SELECT * FROM a **AS** a1 CROSS JOIN a AS a2;

$$\rho_{a_1}(account) \times \rho_{a_2}(account)$$

a1.account_number	a1.branch_name	a1.balance	a2.account_number	a2.branch_name	a2.balance
A-101	Downtown	500	A-101	Downtown	500
A-101	Downtown	500	A-102	Perryridge	400
A-101	Downtown	500	A-201	Brighton	900
A-101	Downtown	500	A-215	Mianus	700
A-101	Downtown	500	A-217	Brighton	750
A-101	Downtown	500	A-222	Redwood	700
A-101	Downtown	500	A-305	Round Hill	350
A-102	Perryridge	400	A-101	Downtown	500
A-102	Perryridge	400	A-102	Perryridge	400
A-102	Perryridge	400	A-201	Brighton	900
A-102	Perryridge	400	A-215	Mianus	700
A-102	Perryridge	400	A-217	Brighton	750
A-102	Perryridge	400	A-222	Redwood	700
A-102	Perryridge	400	A-305	Round Hill	350
…	…	…	…	…	…

Data Query Language: JOIN

- Operators of relational algebra
  - RENAME, ρ (\rho)
  - Practice 1
    - Step 2: select a1's balances if a1's balance is less than a2's balance

$$\pi_{a1.balance}(\sigma_{a1.balance<a2.balance}(\rho_{a_1}(account) \times \rho_{a_2}(account)))$$

a1.balance
500
400
700
750
350

Note: it contains all the balances in account relation except the maximum value

Data Query Language: JOIN

- Operators of relational algebra
  - RENAME, ρ (\rho)
  - Practice 1
    - Step 3: remove the non-maximum balance in account relation = goal

$\pi_{balance}(account)$

balance
500
400
900
700
750
700
350

$-$

$\pi_{a1.balance}(\sigma_{a1.balance<a2.balance}(\rho_{a_1}(account) \times \rho_{a_2}(account)))$

a1.balance
500
400
700
750
350

$=$

balance
900

Data Query Language: JOIN

- Operators of relational algebra
  - RENAME, ρ (\rho)
  - Practice 2
    - Goal: find the names of customers who lives in a street and a city that Smith lives

customer relation

customer	customer_street	customer_city
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Hayes	Main	Harrison
Johbnson	Alma	Palo Alto
Jones	MAIN	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Puthnam	Stamford
Williams	Nassau	Princeton

Data Query Language: JOIN

- Operators of relational algebra
  - RENAME, ρ (\rho)
  - Practice 2
    - Goal: find the names of customers who lives in a street and a city that Smith lives
    - Step 1: Retrieves a street and a city that Smith lives as smith_address

customer_street	customer_city
North	Rye

- Step 2: List up all the cross product between customer and smith_address

$$customer \times \rho_{smith\_address}(\pi_{customer\_street,customer\_city}(\sigma_{customer="Smith"}(customer)))$$

customer	customer_street	customer_city	smith_address.customer_street	smith_address.customer_city
Adams	Spring	Pittsfield	North	Rye
Brooks	Senator	Brooklyn	North	Rye
Curry	North	Rye	North	Rye
Glenn	Sand Hill	Woodside	North	Rye
Hayes	Main	Harrison	North	Rye
Johbnson	Alma	Palo Alto	North	Rye
Jones	MAIN	Harrison	North	Rye
Lindsay	Park	Pittsfield	North	Rye
Smith	North	Rye	North	Rye
Turner	Puthnam	Stamford	North	Rye
Williams	Nassau	Princeton	North	Rye

Data Query Language: JOIN

- Operators of relational algebra
  - RENAME, ρ (\rho)
  - Practice 2
    - Goal: find the names of customers who lives in a street and a city that Smith lives
    - Step 3: select tuples where a customer lives in a street and a city of Smith

$$\sigma_{customer.customer\_street=smith\_address.customer\_street \wedge customer.customer\_city=smith\_address.customer\_city}\Big(customer \times \rho_{smith\_address}\Big(\pi_{customer\_street,customer\_city}\Big(\sigma_{customer\_name="Smith"}(customer)\Big)\Big)\Big)$$

customer	customer.customer_street	customer.customer_city	smith_address.customer_street	smith_address.customer_city
Curry	North	Rye	North	Rye
Smith	North	Rye	North	Rye

    - Step 4: project the name of the customer

$$\pi_{customer\_name}\Big(\sigma_{customer.customer\_street=smith\_address.customer\_street \wedge customer.customer\_city=smith\_address.customer\_city}\Big(customer \times \rho_{smith\_address}\Big(\pi_{customer\_street,customer\_city}\Big(\sigma_{customer\_name="Smith"}(customer)\Big)\Big)\Big)\Big)$$

customer
Curry
Smith

Data Query Language: JOIN

$$\pi_{customer\_name,loan.loan\_number,branch\_name,amount}(\sigma_{borrower.loan\_number=loan.loan\_number}(borrower \times loan))$$

- Operators of relational algebra
  - Motivation
    - Merge borrower and loan relations where its loan_number is matched

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

$$\pi_{customer\_name,loan.loan\_number,branch\_name,amount}(\sigma_{borrower.loan\_number=loan.loan\_number}(borrower \times loan))$$

?

loan relation

loan_number	branch_name	Amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

customer_name	loan_number	branch_name	amount
Adams	L-16	Perryridge	1300
Curry	L-93	Mianus	500
Hayes	L-15	Perryridge	1500
Jackson	L-14	Downtown	1500
Jones	L-17	Downtown	1000
Smith	L-11	Round Hill	900
Smith	L-23	Redwood	2000
Williams	L-17	Downtown	1000

Data Query Language: JOIN

- Data Query Language
  - JOIN
    - NATURAL JOIN
      - JOIN TABLES based on a common field

```
table_references:
    table_reference [, table_reference] ...

table_reference:
    table_factor
  | join_table
```

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
```

```
join_condition:
    ON conditional_expr
  | USING (column_list)
```

```
join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor
```

SELECT customer_name, loan.loan_number, branch_name, amount FROM borrower CROSS JOIN loan WHERE borrower.loan_number = loan.loan_number;

=

Data Query Language: JOIN

```
join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor
```

- Operators of relational algebra
  - Theta Join, $\bowtie_\theta$, \bowtie_\theta
    - Syntax
      - $relation_1 \bowtie_\theta relation_2 = \sigma_\theta(relation_1 \times relation_2)$
    - Merge two relations based on the predicate (theta) for their common attributes
    - Degree = degree of relation 1 and 2
    - Example
      - $\pi_{\text{Customer} \cap \text{Depositor}}(customer \bowtie_{customer.customer=depositor.customer\_name} depositor)$
      - Now, we can join relations with semantically identical but differently named attributes

```
join_condition:
    ON conditional_expr
  | USING (column_list)
```

customer relation

customer	customer_street	customer_city
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Hayes	Main	Harrison
Johbnson	Alma	Palo Alto
Jones	MAIN	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Puthnam	Stamford
Williams	Nassau	Princeton

depositor relation

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Data Query Language: JOIN

- Operators of relational algebra
  - Motivation
    - Revisit a natural join $r \bowtie s$
    - We want all the information on employee in employee and ft-work relations
      - The yellow information is omitted in the joined relation
    - Introduces OUTER JOIN

employee relation

employee_name	Street	City
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

ft_works relation

employee_name	branch_name	salary
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

SELECT * FROM employee NATURAL JOIN ft_works;

$$employee \bowtie ft\_works$$

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500

Data Query Language: JOIN

- Data Query Language
  - JOIN
    - OUTER JOIN
      - LEFT JOIN
      - RIGHT JOIN

```
table_references:
    table_reference [, table_reference] ...

table_reference:
    table_factor
  | join_table
```

```
SELECT
    [ALL | DISTINCT | DISTINCTROW]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [ FROM table_references
```

```
join_condition:
    ON conditional_expr
  | USING (column_list)
```

```
join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor
```

Data Query Language: JOIN

```
join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor
```

- Operators of relational algebra
  - LEFT OUTER JOIN, ⟕, (ALT+10197)
    - $r \leftouterjoin s = (r \bowtie s) \cup \left(r - \pi_R(r \bowtie s)\right) \times \{null, \ldots, null\}$
    - An extension of the natural join that avoids loss of information
    - Include tuples of r not matched in any values of common fields in tuples of s

employee relation

employee_name	Street	City
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

ft_works relation

employee_name	branch_name	salary
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

$employee \bowtie ft\_works$

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500

$employee \leftouterjoin ft\_works$

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Smith	Resolver	Death Valley	**null**	**null**

Data Query Language: JOIN

- Operators of relational algebra
  - RIGHT OUTER JOIN, ⟖, (ALT+10198)
    - $r ⟖ s$
    - An extension of the natural join that avoids loss of information
    - Include tuples of r not matched in any values of common fields in tuples of s

employee relation

employee_name	Street	City
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

ft_works relation

employee_name	branch_name	salary
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

$employee ⋈ ft\_works$

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500

$employee ⟖ ft\_works$

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Gates	null	null	Redmond	5300

Data Query Language: JOIN

- Operators of relational algebra
  - FULL OUTER JOIN, ⟗, (ALT+10199)
    - $r ⟗ s$
    - An extension of the natural join that avoids loss of information
    - Include tuples of r not matched in any values of common fields in tuples of s

Not supported

employee relation

employee_name	Street	City
Coyote	Toon	Hollywood
Rabbit	Tunnel	Carrotville
Smith	Revolver	Death Valley
Williams	Seaview	Seattle

ft_works relation

employee_name	branch_name	salary
Coyote	Mesa	1500
Rabbit	Mesa	1300
Gates	Redmond	5300
Williams	Redmond	1500

$employee ⟗ ft\_works$

employee_name	Street	City	branch_name	Salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Williams	Seaview	Seattle	Redmond	1500
Smith	Resolver	Death Valley	Null	null
Gates	null	null	Redmond	5300

Data Query Language: DML + DQL

- INSERT INTO table_name
  - https://mariadb.com/kb/en/insert/

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
 [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
 {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
 [ ON DUPLICATE KEY UPDATE
   col=expr
     [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

Or:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)]
    SET col={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

Or:

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE
      col=expr
        [, col=expr] ... ] [RETURNING select_expr
      [, select_expr ...]]
```

Data Query Language: DML + DQL

- DML + DQL
  - INSERT INTO table (col_name,…) SELECT …

Copy and paste rows in loan
Copy and paste rows in loan if its amount > 1000
Copy and paste rows only with loan_number and branch_name in loan

Data Query Language: DML, more details

- DELETE FROM table_name

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
  FROM tbl_name [PARTITION (partition_list)]
  [WHERE where_condition]
  [ORDER BY ...]
  [LIMIT row_count]
  [RETURNING select_expr
    [, select_expr ...]]
```

- DELETE ALL
  - DELETE FROM loan ;

DELETE rows in loan if amount is empty;

Data Query Language: Subquery

+ 2023.1 DB

- Data Query Language (dql.P15)
  - Subquery
    - From the demand that wants to use a result of SELECT (subquery) to SELECT statement
    - e.g., Print out branch_name and amount of loan that 'Adams' has

SELECT branch_name, amount FROM loan WHERE loan_number = ?

SELECT loan_number FROM borrower WHERE customer_name='Adams';

loan relation

loan_number	branch_name	Amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Data Query Language: Subquery

+ 2023.1 DB

- Data Query Language (dql.P15)
  - Subquery
    - https://mariadb.com/kb/en/subqueries/
    - Scalar Subqueries
      - Using a single value from a subquery
    - Multi Column Subqueries
      - Using a tuple from a subquery
    - Multi Row and IN/ALL/ANY/EXISTS
      - Comparing all the returned rows from a subquery
    - Subqueries in a FROM Clause

WHERE Clause

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Scalar

Multi Column

Multi Row

Data Query Language: Subquery

+ 2023.1 DB

- Data Query Language
  - Subquery
    - Scalar Subqueries
      - Basic idea: want to use a return value of subquery
        - a single value (One attribute)

Practice: sum of balance of **account** where its branch_name is equal to L-14's branch_name

account relation

account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

loan relation

loan_number	branch_name	Amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

Data Query Language: Subquery

+ 2023.1 DB

- Data Query Language (dql.P16)
  - Subquery
    - Multi Column Subqueries
      - Basic idea: want to use return values of subquery
        - a row or a tuple of multiple columns

```
CREATE OR REPLACE TABLE staff (name VARCHAR(10), age TINYINT); CREATE OR REPLACE TABLE customer (name VARCHAR(10), age TINYINT); INSERT INTO staff VALUES ('Bilhah',37), ('Valerius',61), ('Maia',25); INSERT INTO customer VALUES ('Thanasis',48), ('Valerius',61), ('Brion',51);
```

staff

Name VARCHAR(10)	Age INTEGER
'Bilhah'	37
'Valerius'	61
'Maia'	25

customer

Name VARCHAR(10)	Age INTEGER
'Thanasis'	48
'Valerius'	61
'Brion'	51

  - Practice
    - Find staff(s) if its name and age are identical to 'Valerius' 61 years old customer

Data Query Language: Subquery

+ 2023.1 DB

- Data Query Language
  - Subquery
    - Multi Row Subqueries (IN , ANY, SOME, ALL)
      - IN: https://mariadb.com/kb/en/in/
      - SYNTAX: expr IN (value,...)
        - Returns 1 if expr is equal to any of the values in the IN list, else returns 0.
  - Practice: Find borrower names who have (a) loan in 'Perryridge'

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

loan relation

loan_number	branch_name	Amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

Data Query Language: Subquery

+ 2023.1 DB

- Data Query Language
  - Subquery
    - Multi Column/Row Subqueries
      - Basic idea: want to use a return value of subquery
        - a row or a tuple of multiple columns

customer3 relation

customer	customer_street	customer_city
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Hayes	Main	Harrison
Hayes	**Alma**	**Palo Alto**
Johbnson	Alma	Palo Alto
Jones	MAIN	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Puthnam	Stamford
Williams	Nassau	Princeton

  - Practice
    - Search customers living in street and city that Hayes lives

Data Query Language: Subquery

+ 2023.1 DB

- Data Query Language
  - Subquery
    - Multi Row Subqueries (IN , ANY, SOME, ALL)
      - ANY keyword will return true if the comparison returns true for at least one row returned by the subquery.
      - scalar_expression comparison_operator ANY <Table subquery>

a table

Point INTEGER
89
50
60
65
85

b table

Point INTEGER
99
30
50
60
75

- Find a row of table b where its point is larger than any points in table a

Data Query Language: Subquery

+ 2023.1 DB

- Data Query Language (dql.P18)
  - Subquery
    - EXISTS
      - EXISTS keyword will return true if the subquery returns any rows. Conversely, subqueries using NOT EXISTS will return true only if the subquery returns no rows from the table.

NOT EXISTS, UNIQUE, NOT UNIQUE

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

depositor relation

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

- All the depositor having loan

Data Query Language: Subquery

+ 2023.1 DB

- Data Query Language
  - Subquery
    - FROM
      - as also known as Inline View
      - can also form part of the FROM clause. Such subqueries are commonly called derived tables.
      - If a subquery is used in this way, you must also use an AS clause to name the result of the subquery.

```
CREATE TABLE student2 (name CHAR(10), test CHAR(10), score
TINYINT); INSERT INTO student2 VALUES ('Chun', 'SQL', 75), ('Chun',
'Tuning', 73), ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31), ('Kaolin', 'SQL',
56), ('Kaolin', 'Tuning', 88), ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);
```

name VARCHAR(10)	test VARCHAR(10)	score INTEGER	
'Chun'	'SQL'	75	
'Chun'	'Tuning'	73	148
'Esben'	'SQL'	43	
'Esben'	'Tuning'	31	74
'Kaolin'	'SQL'	56	
'Kaolin'	'Tuning'	88	144
'Tatiana'	'SQL'	87	
'Tatiana'	'Tuning'	83	170

- Question
  - Show the average of total points that each person gets

$$\mathcal{G}_{AVG(total)}((\rho_a(\pi_{SUM(score)AS\ total}(\ \ _{name}\mathcal{G}_{SUM(score)}(student)))))$$

Data Query Language: Subquery

- $\mathcal{G}_{AVG(total)}((\rho_a(\pi_{SUM(score)AS\ total}(\ \ _{name}\mathcal{G}_{SUM(score)}(student)))))$

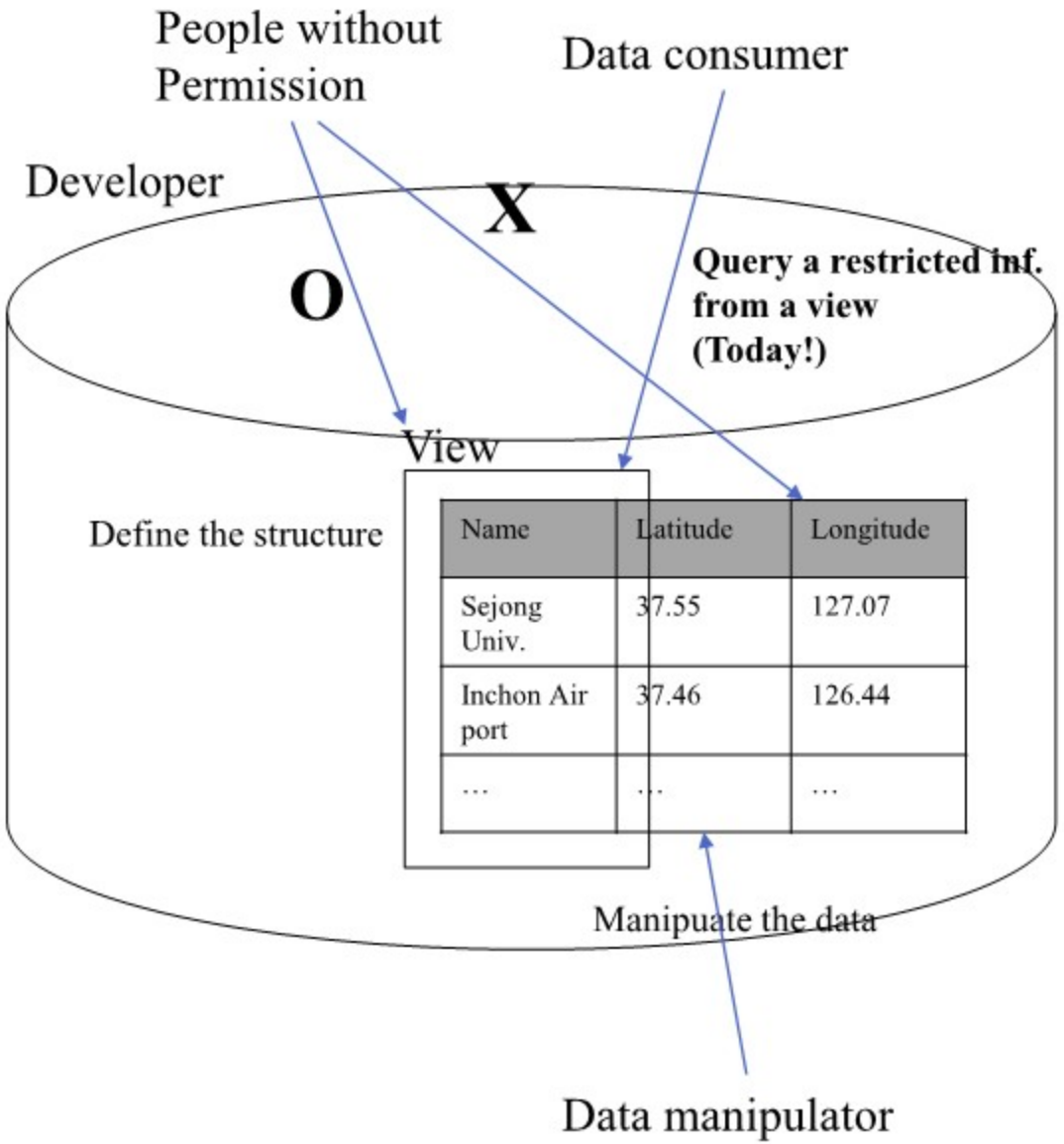
```
MariaDB [db1]> SELECT name, SUM(score) FROM student GROUP BY name;
+---------+------------+
| name    | SUM(score) |
+---------+------------+
| Chun    |        148 |
| Esben   |         74 |
| Kaolin  |        144 |
| Tatiana |        170 |
+---------+------------+
4 rows in set (0.000 sec)

MariaDB [db1]> SELECT SUM(score) AS total FROM student GROUP BY name;
+-------+
| total |
+-------+
|   148 |
|    74 |
|   144 |
|   170 |
+-------+
4 rows in set (0.000 sec)

MariaDB [db1]> SELECT AVG(total) FROM (SELECT SUM(score) AS total FROM student GROUP BY name) AS a;
+------------+
| AVG(total) |
+------------+
|   134.0000 |
+------------+
1 row in set (0.000 sec)

MariaDB [db1]>
```

Introduction to SQL and MariaDB

- SQL
  - Data definition
  - Data manipulation
  - Data query
  - Data control

Example: geo data

Sejong Univ., 37.55, 127.07
Incheon Airport, 37.46, 126.44
…



View

- View
  - A logical or virtual table derived by table(s)
  - Not store any data but store its definition (The definition is stored in System Catalog)
  - can be defined based on another view
  - The drop of a base table yields the drop of the view
  - Advantages
    - Logical data independence
    - Access control
      - hide unnecessary information for some users
      - provide only necessary information for some users
    - Simplicity
    - Meeting various users' demand

SELECT customer, customer_city FROM customer;
→ staff VIEW
SELECT customer FROM staff_customer;
→ probation_staff VIEW

chief manager → see all the inf.

staff → see inf. except street

customer relation

customer	customer_street	customer_city
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Hayes	Main	Harrison
Johbnson	Alma	Palo Alto
Jones	MAIN	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Puthnam	Stamford
Williams	Nassau	Princeton

View

- Create View
  - Syntax

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW [IF NOT EXISTS] view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

```
MariaDB [db]> SELECT * FROM all_customer;
+---------------+---------------+
| branch_name   | customer_name |
+---------------+---------------+
| Downtown      | Johnson       |
| Perryridge    | Hayes         |
| Brighton      | Johnson       |
| Mianus        | Smith         |
| Brighton      | Jones         |
| Redwood       | Lindsay       |
| Round Hill    | Turner        |
| Round Hill    | Smith         |
| Downtown      | Jackson       |
| Perryridge    | Adams         |
| Downtown      | Jones         |
| Downtown      | Williams      |
| Redwood       | Smith         |
| Mianus        | Curry         |
+---------------+---------------+
14 rows in set (0.006 sec)
```

    - CREATE VIEW view_name [(column_list)] AS
      SELECT statement

- Try
  - CREATE VIEW all_customer as (SELECT branch_name, customer_name FROM depositor, account WHERE depositor.account_number = account.account_number) UNION (SELECT branch_name, customer_name FROM borrower, loan WHERE borrower.loan_number = loan.loan_number);
  - CREATE VIEW branch_total_loan (branch_name, total_loan) AS
    SELECT branch_name, SUM(amount) FROM loan
    GROUP BY branch_name;
    - total_loan = SUM(amount) : naming

View

- Create View
  - Syntax

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW [IF NOT EXISTS] view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

```
MariaDB [db]> SELECT * FROM all_customer;
+-------------+---------------+
| branch_name | customer_name |
+-------------+---------------+
| Downtown    | Johnson       |
| Perryridge  | Hayes         |
| Brighton    | Johnson       |
| Mianus      | Smith         |
| Brighton    | Jones         |
| Redwood     | Lindsay       |
| Round Hill  | Turner        |
| Round Hill  | Smith         |
| Downtown    | Jackson       |
| Perryridge  | Adams         |
| Downtown    | Jones         |
| Downtown    | Williams      |
| Redwood     | Smith         |
| Mianus      | Curry         |
+-------------+---------------+
14 rows in set (0.005 sec)
```

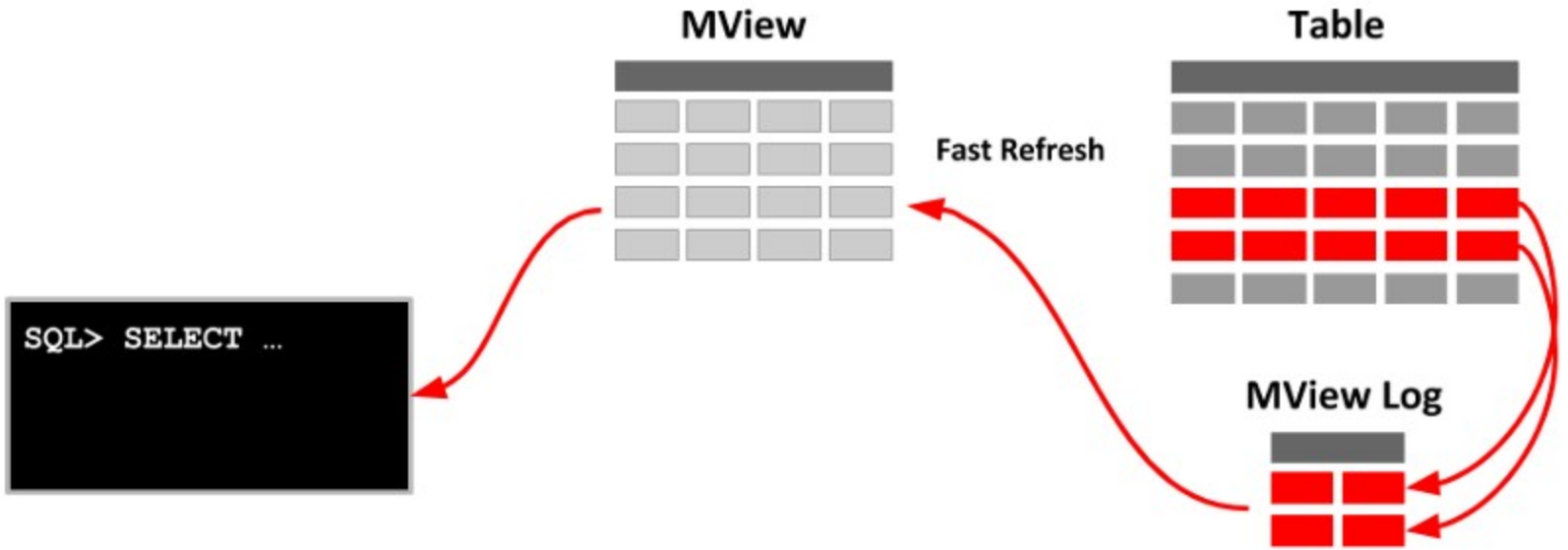
  - You can use a view as like normal table
    - SELECT customer_name FROM all_customer WHERE branch_name = 'Perryridge';

```
MariaDB [db]> SELECT customer_name FROM all_customer WHERE branch_name = 'Perryridge';
+---------------+
| customer_name |
+---------------+
| Hayes         |
| Adams         |
+---------------+
2 rows in set (0.000 sec)
```

View

- How it works
  - A view does not store any data but store its definition
  - A view is computed whenever a query related to the view is evaluated

- ((Normal) View) vs. Materialized View
  - guarantees its latest state



https://oracle-base.com/articles/misc/materialized-views

View

- View Expansion
  - CREATE VIEW perryridge_customer AS SELECT customer_name FROM all_customer WHERE branch_name = "Perryridge";
    - perryridge_customer is defined based on all_customer view

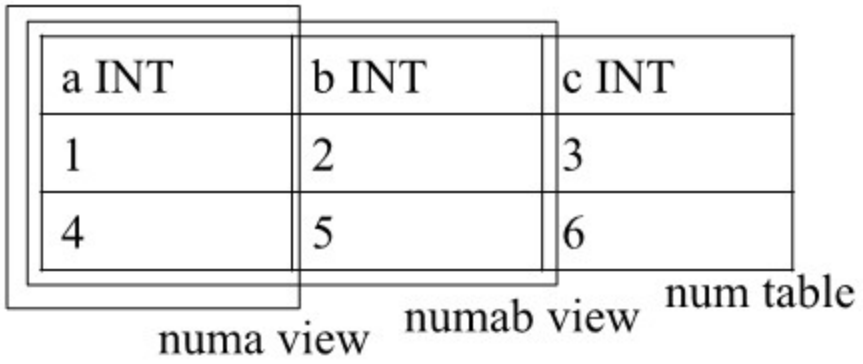
```
REPEAT
    Find any view relation vi in exp
    Replace the view relation vi by the expression defining vi
UNTIL no more view relations are present in exp
```

- SELECT * FROM perryridge_customer WHERE customer_name = ‘John’;
- → expanding
- SELECT * FROM
  (SELECT customer_name FROM all_customer WHERE branch_name = ‘Perryridge’)
  WHERE customer_name = ‘John’;
- → expanding
- SELECT * FROM
  (SELECT customer_name FROM
  (SELECT branch_name, customer_name FROM depositor, account WHERE depositor.account_number = account.account_number) UNION (SELECT branch_name, customer_name FROM borrower, loan WHERE borrower.loan_number = loan.loan_number)
  WHERE branch_name = ‘Perryridge’)
  WHERE customer_name = ‘John’;

View

- Drop View
  - Syntax

```
DROP VIEW [IF EXISTS]
    view_name [, view_name] ...
    [RESTRICT | CASCADE]
```



  - DROP VIEW all_customer;

  - RESTRICT
    - delete the view if the view is not derived by another view
  - CASCADE
    - delete the view and views deriving the view

  - Try

View

- Restriction – View
  - Example 1: Consider the following view
    - CREATE VIEW loan_branch as
      SELECT loan_number, branch_name
      FROM loan;
  - What if the following SQL is executed, then?
    - INSERT INTO loan_branch VALUES ('L-37', 'Perryridge');

  - Two possible options
    - $loan \leftarrow loan \cup ("L-37", "Perryridge", null)$
    - or yield error

loan_branch view

loan relation

loan_number	branch_name	Amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

MariaDB default

```
MariaDB [db]> SELECT * FROM loan_branch;
+-------------+-------------+
| loan_number | branch_name |
+-------------+-------------+
| L-11        | Round Hill  |
| L-14        | Downtown    |
| L-15        | Perryridge  |
| L-16        | Perryridge  |
| L-17        | Downtown    |
| L-23        | Redwood     |
| L-93        | Mianus      |
+-------------+-------------+
7 rows in set (0.005 sec)
```

```
MariaDB [db]> SELECT * FROM loan;
+-------------+-------------+--------+
| loan_number | branch_name | amount |
+-------------+-------------+--------+
| L-11        | Round Hill  |    900 |
| L-14        | Downtown    |   1500 |
| L-15        | Perryridge  |   1500 |
| L-16        | Perryridge  |   1300 |
| L-17        | Downtown    |   1000 |
| L-23        | Redwood     |   2000 |
| L-93        | Mianus      |    500 |
| L-37        | Perryridge  |   NULL |
+-------------+-------------+--------+
8 rows in set (0.000 sec)
```

View

- Restriction – View
  - Example 2: Consider the following view
    - CREATE VIEW loan_info AS
      SELECT customer_name, amount
      FROM borrower, loan
      WHERE borrower.loan_number = loan.loan_number;
  - What if the following SQL is executed, then?
    - INSERT INTO loan_info VALUES ('Johnson', 1900);

loan relation

loan_number	branch_name	Amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

borrower relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

```
MariaDB [db]> SELECT * FROM loan_info;
+---------------+--------+
| customer_name | amount |
+---------------+--------+
| Adams         |   1300 |
| Curry         |    500 |
| Hayes         |   1500 |
| Jackson       |   1500 |
| Jones         |   1000 |
| Smith         |    900 |
| Smith         |   2000 |
| Williams      |   1000 |
+---------------+--------+
8 rows in set (0.005 sec)
```

(null, null, 1900)

('Johnson', null)

MariaDB default

```
MariaDB [db]> INSERT INTO loan_info VALUES ('Johnson', 1900);
ERROR 1394 (HY000): Can not insert into join view 'db.loan_info' without fields list
```

View

- Restriction – View
  - There are so many problems related to View updates
  - Each relational database handles the updates in their own ways

  - The (general) updatable conditions
    - FROM statement indicates only one relation
    - SELECT statement should include attributes names
      - e.g., expression, aggregation, distinct, group by, having not allowable
    - The view does not extend a view that is not updatable
    - Attributes out of Select statement Nullable

View

- Type of View
  - Column subset view
  - Row subset view
  - Join view
  - Statistical summary view

Privilege

- Motivation
  - We can create a view for specific users to access to a subset of information which is allowable
- A type of privileges
  - READ
  - INSERT
  - UPDATE
  - DELETE

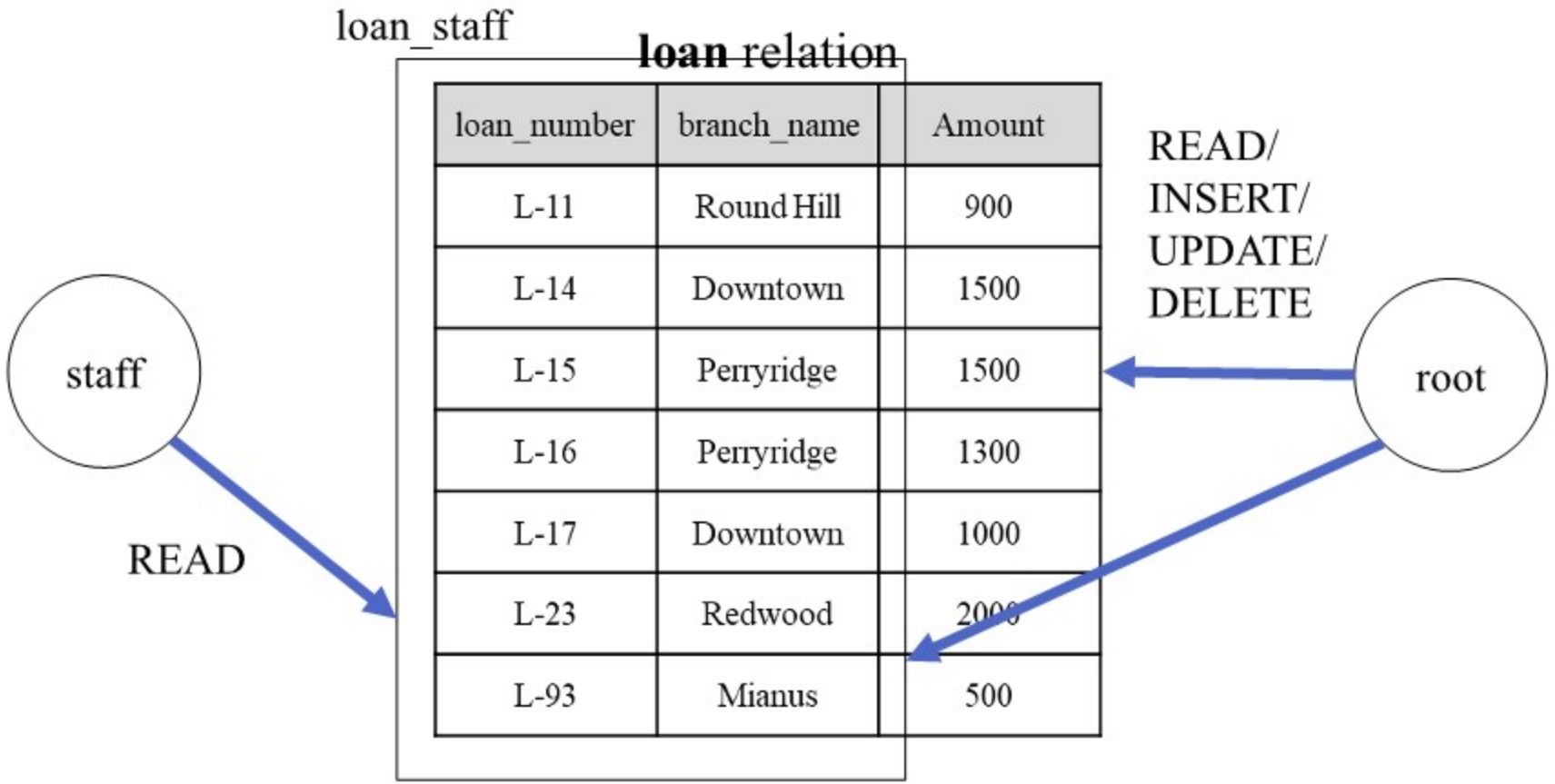
loan relation

loan_number	branch_name	Amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

staff can see it

Privilege

- Learn a brief concept based on a scenario



Privilege

- Step 0. CREATE VIEW
  - CREATE VIEW loan_staff AS SELECT loan_number, branch_name FROM loan;

```
MariaDB [db]> SELECT * FROM loan;
+-------------+-------------+--------+
| loan_number | branch_name | amount |
+-------------+-------------+--------+
| L-11        | Round Hill  |    900 |
| L-14        | Downtown    |   1500 |
| L-15        | Perryridge  |   1500 |
| L-16        | Perryridge  |   1300 |
| L-17        | Downtown    |   1000 |
| L-23        | Redwood     |   2000 |
| L-93        | Mianus      |    500 |
+-------------+-------------+--------+
7 rows in set (0.000 sec)
```

```
MariaDB [db]> SELECT * FROM loan_staff;
+-------------+-------------+
| loan_number | branch_name |
+-------------+-------------+
| L-11        | Round Hill  |
| L-14        | Downtown    |
| L-15        | Perryridge  |
| L-16        | Perryridge  |
| L-17        | Downtown    |
| L-23        | Redwood     |
| L-93        | Mianus      |
+-------------+-------------+
7 rows in set (0.005 sec)
```

Privilege

- Step 1. CREATE USER
  - Syntax

```
CREATE [OR REPLACE] USER [IF NOT EXISTS]
 user_specification [,user_specification ...]
  [REQUIRE {NONE | tls_option [[AND] tls_option ...] }]
  [WITH resource_option [resource_option ...] ]
  [lock_option] [password_option]

user_specification:
  username [authentication_option]

authentication_option:
  IDENTIFIED BY 'password'
  | IDENTIFIED BY PASSWORD 'password_hash'
  | IDENTIFIED {VIA|WITH} authentication_rule [OR authentication_rule  ...]

authentication_rule:
    authentication_plugin
  | authentication_plugin {USING|AS} 'authentication_string'
  | on_plugin {USING|AS} PASSWORD('password')

  SSL
  | X509
  | CIPHER 'cipher'
  | ISSUER 'issuer'
  | SUBJECT 'subject'

resource_option:
  MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
```

- CREATE USER ‘staff’@localhost IDENTIFIED BY ‘staffpw’;

```
MariaDB [(none)]> CREATE USER 'staff'@localhost IDENTIFIED BY 'staffpw';
Query OK, 0 rows affected (0.002 sec)
```

  - staff from localhost
    pw: staffpw

- CHECK
  - SELECT user FROM mysql.user;

```
MariaDB [(none)]> SELECT user FROM mysql.user;
+-------------+
| User        |
+-------------+
| root        |
| root        |
| root        |
| mariadb.sys |
| root        |
| staff       |
+-------------+
6 rows in set (0.006 sec)
```

Privilege

- Step 2. GRANT Privilege
  - Syntax

- GRANT SELECT ON db2.loan_staff TO staff@localhost;

- CHECK
  - SHOW GRANTS FOR staff@localhost;

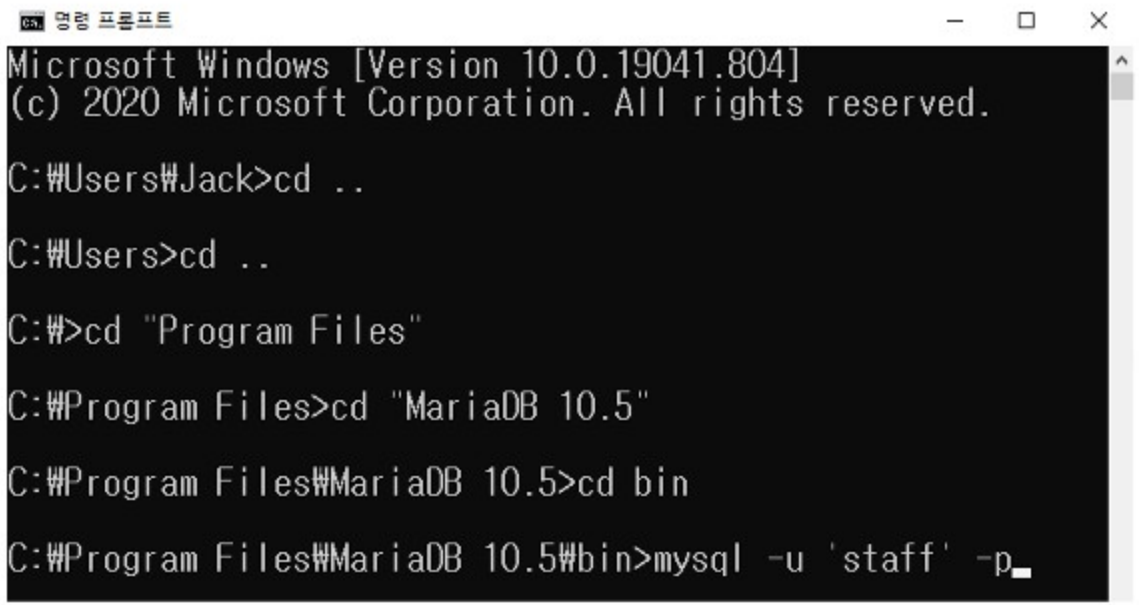
```
GRANT
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    TO user_specification [ user_options ...]

user_specification:
  username [authentication_option]

authentication_option:
  IDENTIFIED BY 'password'
  | IDENTIFIED BY PASSWORD 'password_hash'
  | IDENTIFIED {VIA|WITH} authentication_rule [OR authentication_rule  ...]

authentication_rule:
    authentication_plugin
  | authentication_plugin {USING|AS} 'authentication_string'
  | authentication_plugin {USING|AS} PASSWORD('password')

GRANT PROXY ON username
    TO username [, username] ...
    [WITH GRANT OPTION]

user_options:
    [REQUIRE {NONE | tls
    [WITH with_option [w

object_type:
    TABLE
  | FUNCTION
  | PROCEDURE

priv_level:
  | resource_option
```

```
MariaDB [db]> SELECT CURRENT_USER();
+----------------+
| CURRENT_USER() |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.000 sec)
```

```
MariaDB [db]> SHOW GRANTS FOR staff@localhost;
+-----------------------------------------------------------------------------------------------------------------------------------------+
| Grants for staff@localhost                                                                                                              |
+-----------------------------------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `staff`@`localhost` IDENTIFIED BY PASSWORD '*BE152FDA5EDAF33247F1DA62F36D5354B1B0C675' |
| GRANT SELECT ON `db`.`loan_staff` TO `staff`@`localhost`                                                                                |
+-----------------------------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.000 sec)
```

Privilege

- Step 3. Access MariaDB by staff@localhost



Privilege

- Step 4. Check accessibility

```
C:₩Program Files₩MariaDB 10.5₩bin>mysql -u 'staff' -p
Enter password: *******
Welcome to the MariaDB monitor.  Commands end with ; or ₩g.
Your MariaDB connection id is 13
Server version: 10.5.5-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '₩h' for help. Type '₩c' to clear the current input statement.


MariaDB [(none)]> use db
Database changed
MariaDB [db]> SELECT * FROM loan;
ERROR 1142 (42000): SELECT command denied to user 'staff'@'localhost' for tabl
e 'loan'
MariaDB [db]> SELECT * FROM loan_staff;
+-------------+-------------+
| loan_number | branch_name |
+-------------+-------------+
| L-11        | Round Hill  |
| L-14        | Downtown    |
| L-15        | Perryridge  |
| L-16        | Perryridge  |
| L-17        | Downtown    |
| L-23        | Redwood     |
| L-93        | Mianus      |
+-------------+-------------+
7 rows in set (0.000 sec)

MariaDB [db]> INSERT INTO loan_staff VALUES ('L-13','Perryridge');
ERROR 1142 (42000): INSERT command denied to user 'staff'@'localhost' for tabl
e 'loan_staff'
MariaDB [db]>
```

Privilege

- Step 5. Revoke Privilege
  - Syntax

```
REVOKE
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM user [, user] ...

REVOKE ALL PRIVILEGES, GRANT OPTION
    FROM user [, user] ...
```

- REVOKE SELECT ON loan_staff FROM staff@localhost;

```
MariaDB [db]> REVOKE SELECT ON loan_staff FROM staff@localhost;
Query OK, 0 rows affected (0.001 sec)
```

- Drop User
  - Syntax

```
DROP USER [IF EXISTS] user_name [, user_name] ...
```

- CHECK
  - SHOW GRANTS FOR staff@localhost;

```
MariaDB [db]> DROP USER staff@localhost;
Query OK, 0 rows affected (0.001 sec)

MariaDB [db]> SELECT user FROM mysql.user;
+-------------+
| User        |
+-------------+
| root        |
| root        |
| root        |
| mariadb.sys |
| root        |
+-------------+
5 rows in set (0.001 sec)
```

Privilege

- See details here
  - https://mariadb.com/kb/en/grant/

  - Privilege Levels
    - Global
      - *.*
    - Database
      - db_name.*
    - Table
      - db_name.tbl_name
    - Column
      - e.g., GRANT SELECT (col1,col2,…) on db_name.tbl_name
    - Function
    - Procedure

Summary

- SQL
  - DDL
  - DML
  - DQL
  - DCL

- Next Class