Database Programming *Cheat Sheet by* **pizzaroot**

DO NOT FORGET SEMICOLON;

CREATE DATABASE

```
CREATE DATABASE dbp;
CREATE SCHEMA dbp;
```

USE

```
USE dbp;
```

CREATE TABLE

```
CREATE TABLE grade (id INT PRIMARY KEY, score DOUBLE);
```

```
CREATE TABLE IF NOT EXISTS location (name VARCHAR(50), coord POINT);
```

INSERT

```
INSERT INTO grade VALUES (1, 74.9), (2, 63.8);
INSERT grade VALUES (1, 74.9), (2, 63.8);
```

SELECT

```
SELECT * FROM loan;
```

Cartesian Product ×

```
SELECT * FROM loan JOIN borrower;
SELECT * FROM loan INNER JOIN borrower;
SELECT * FROM loan CROSS JOIN borrower;
SELECT * FROM loan , borrower;
```

NATURAL JOIN ⋈

```
SELECT * FROM loan NATURAL JOIN borrower;
```

NATURAL LEFT OUTER JOIN ⟕

```
SELECT * FROM loan NATURAL LEFT OUTER JOIN borrower;
SELECT * FROM loan NATURAL LEFT JOIN borrower;
```

NATURAL RIGHT OUTER JOIN ⟖

```
SELECT * FROM loan NATURAL RIGHT OUTER JOIN borrower;
SELECT * FROM loan NATURAL RIGHT JOIN borrower;
```

EXCEPT −

```
SELECT * FROM grade EXCEPT SELECT * FROM grade WHERE score < 30;
SELECT * FROM grade EXCEPT DISTINCT SELECT * FROM grade WHERE score < 30;
SELECT DISTINCT * FROM grade EXCEPT ALL SELECT * FROM grade WHERE score < 30;
```

DISTINCT

```
SELECT DISTINCT * FROM depositor;
SELECT DISTINCTROW * FROM depositor;
```

DELETE

```
DELETE FROM location WHERE name;
```

DROP TABLE

```
DROP TABLE location;
```

DROP DATABASE

```
DROP DATABASE dbp;
DROP SCHEMA dbp;
```

SHOW TABLES

```
SHOW TABLES;
```

RENAME ρ

```
SELECT * FROM score EXCEPT SELECT a.x FROM score AS a, score AS b WHERE a.x < b.x;
```

Integrity constraints

CHECK(expr**)**

```
CREATE OR REPLACE TABLE j (j JSON CHECK(JSON_VALID(j)));
```

NOT NULL

```
ALTER TABLE j MODIFY j JSON NOT NULL CHECK(JSON_VALID(j));
```

UNIQUE

```
CREATE TABLE relation (i INT, j VARCHAR(50), UNIQUE(i, j));
```

```
CREATE TABLE number (n INT UNIQUE);
```

PRIMARY KEY at most one primary key per table and cannot be NULL

```
CREATE TABLE grade (id INT PRIMARY KEY, score DOUBLE);
CREATE TABLE grade (id INT, score DOUBLE, PRIMARY KEY (id));
```

Built-in Functions

IF(expr1**,** expr2**,** expr3**)**

SELECT **IF(3 < 4, 'YES', 'NO')**;

```
3 < 4 ? "YES" : "NO"
```

RAND(), FLOOR()

generate random odd positive integer less than 100
SELECT **FLOOR(RAND() * 50) * 2 + 1**;

DATE_FORMAT(date**,** format**), FROM_UNIXTIME(**unix_timestamp**)**

2023-09-09
SELECT **DATE_FORMAT(FROM_UNIXTIME(1694185200), "%Y-%m-%d")**;

2023/9/9
SELECT **DATE_FORMAT(FROM_UNIXTIME(1694185200), "%Y/%c/%e")**;

September 9th, 2023
SELECT **DATE_FORMAT(FROM_UNIXTIME(1694185200), "%M %D, %Y")**;

DAYOFWEEK(date**)** 1-based indexing

3 (= Tuesday)
SELECT **DAYOFWEEK(**FROM_UNIXTIME(1698073200)**)**;

DAYOFMONTH(date**)** 1-based indexing

24
SELECT **DAYOFMONTH(**FROM_UNIXTIME(1698073200)**)**;

MONTH(date**)** 1-based indexing

10
SELECT **MONTH(**FROM_UNIXTIME(1698073200)**)**;

Examples

Create user-defined aggregate function that returns maximum distance from the origin

```
DELIMITER //
CREATE OR REPLACE AGGREGATE FUNCTION MAX_VECTOR(x POINT) RETURNS DOUBLE
BEGIN
  DECLARE ans DOUBLE DEFAULT 0;
  DECLARE d DOUBLE DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR NOT FOUND
  RETURN ans;
  LOOP
    FETCH GROUP NEXT ROW;
    SET d = ST_DISTANCE(x, POINT(0, 0));
    IF d > ans THEN
      SET ans = d;
    END IF;
  END LOOP;
END //
DELIMITER ;
```

DQL that returns maximum distance between two points

```
CREATE OR REPLACE TABLE location (location POINT);

INSERT INTO location VALUES (POINT(1,2)), (POINT(3,4)), (POINT(2,0)), (POINT(0,0));

SELECT MAX(ST_DISTANCE(a.location, b.location)) FROM location AS a, location AS b;
```

DQL that returns second maximum value in a relation with degree of 1 without using aggregate function nor ORDER BY

```
CREATE OR REPLACE TABLE score (x INT);

INSERT INTO score VALUES (43), (44), (1), (2), (8), (3), (13), (20);

(SELECT * FROM score EXCEPT SELECT a.x FROM score AS a, score AS b, score AS c WHERE a.x < b.x AND b.x < c.x) EXCEPT (SELECT * FROM score EXCEPT SELECT a.x FROM score AS a, score AS b WHERE a.x < b.x);
```

Data Loading - XML

```
public class XMLLoader {
    public static void main(String[] args) throws Exception {
        Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(new InputSource(new FileReader("C:\\data.xml")));

        Node root = doc.getChildNodes().item(0);

        String tblName = root.getNodeName();

        NodeList records = root.getChildNodes();

        NodeList attributes = records.item(1).getChildNodes();

        List<String> columnNames = new ArrayList<>();
        List<String> columnTypes = new ArrayList<>();

        Map<String, String> type = new HashMap<>();

        type.put("xsd:string", "TEXT");
        type.put("xsd:double", "DOUBLE");
        type.put("xsd:integer", "INT");

        type.put("INT", "");
        type.put("TEXT", "'");
        type.put("DOUBLE", "");

        for (int i = 0; i < attributes.getLength(); i++) {
            String attribute = attributes.item(i).getNodeName();
            if (attribute.equals("#text")) continue;
            columnNames.add(attribute);
            columnTypes.add(type.get(attributes.item(i).getAttributes().getNamedItem("xsi:type").getTextContent()));
        }

        String query = "CREATE OR REPLACE TABLE " + tblName + " (";

        for (int i = 0; i < columnNames.size(); i++) {
            if (i > 0) query += ", ";
            query += columnNames.get(i);
            query += " " + columnTypes.get(i);
        }

        query += ")";

        String id = "root";
        String password = "1234";

        Connection connection = DriverManager.getConnection("jdbc:mariadb://localhost:3306", id, password);

        Statement stmt = connection.createStatement();

        stmt.executeUpdate("CREATE DATABASE IF NOT EXISTS dbp");
        stmt.executeUpdate("USE dbp");
        stmt.executeUpdate(query);

        for (int i = 0; i < records.getLength(); i++) {
            Node record = records.item(i);
            if (record.getNodeName().equals("#text")) continue;
            String dml = "INSERT INTO " + tblName + " VALUES (";
            NodeList attrs = record.getChildNodes();
            int idx = 0;
            for (int j = 0; j < attrs.getLength(); j++) {
                String attribute = attrs.item(j).getNodeName();
                if (attribute.equals("#text")) continue;
                if (idx > 0) dml += ", ";
                dml += type.get(columnTypes.get(idx));
                dml += attrs.item(j).getTextContent().trim();
                dml += type.get(columnTypes.get(idx));
                idx++;
            }
            dml += ")";

            stmt.executeUpdate(dml);
        }
    }
}
```