SQLite Document

SQL(Structured Query Language) is specific purposed programming language to manipulate tables from relational database management system(RDBMS). SQL was constructed for creating and manipulating schemas, querying data and handling accessment of database object.

SQLite is a database engine written in the C language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases. It is the most widely deployed database engine, as it used by several of the top web browsers, operating systems, mobile phones, and other embedded systems. SQLite has bindings to many programming languages. It generally follows PostgreSQL syntax but does not enforce type checking.

1. DML

DML(Data Manipulation Language) is a family of computer languages including commands permitting users to manipulate data in database. This manipulation involves inserting data into database tables, retreiving existing data, deleting data from existing tables and modifying existing data.

1.1 Querying in DB

Query is a precise request for information retreval with database and information systems.

SELECT

SELECT choose columns from database.

```
SELECT [col_name1],
       [col_name2 from subquery](SELECT ... FROM ... ;),
       [col_name3] AS alias_name,
       ...
```

FROM

FROM specifies table from database.

```
SELECT ...
  FROM [table_name] or [table from subquery](SELECT ... FROM ...;)
```

LIMIT

LIMIT specifies rows to show.

```
SELECT ...
  FROM ...
   ...
 LIMIT 3;
```

WEHRE

WHERE is used to retreive rows from table which make specific conditions. We can use all boolean operator used in Python, such as <, <=, >, >=, =, !=, <>

```
SELECT ...
  FROM ...
 WHERE [condition_1] OR [condition_2] AND [condition_3]
   ...;
```

1. IN

We can use IN to specify a list of values we want to match against.

```
SELECT ...
  FROM [table_name]
 WHERE [col_name1] IN ('value1', 'value2', ...)
   ...;
```

2. LIKE

We can also matches a part of a string in specific column. We should use % % to specify the patterns.

```
SELECT ...
    FROM ...
WHERE [col_name] LIKE "%pattern%";
```

GROUP BY / HAVING

GROUP BY is used to group rows in unique values. When we need data in specific conditions after apply grouping we use HAVING.

```
SELECT [col_name1], FUNC([col_name2]) AS [new_col1]
  FROM ...
 GROUP BY [col_name1], [col_name2], ...
 HAVING [condition_1]
```

ORDER BY

ORDER BY is used when we want to sort tables in specific columns. we can replace [col_name] by integer on positional argument in SELECT caluse, but this replacement hurt readability of codes.

```
SELECT ...
  FROM ...
 WHERE ...
   ...
 ORDER BY [col_name1], [col_name2] DESC;
```

AS

When we apply aggregation functions to columns, the final name of calculated columns is like 'AVG(col)'. To prevent this, we can make new name with AS clause.

```
SELECT COUNT([col_name1]) AS [new_col1],
       (SELECT [col_name2] FROM [table_name]) AS [new_col2],
   ...
  FROM ...
   ...;
```

DISTINCT

DISTINCT clause takes unique variable in specific column.

```
SELECT COUNT(DISTINCT [col_name1]) AS 'count of unique variable'
  FROM [table_name]
...
```

CASE WHEN

We can use if/then logic in SQL using CASE ~ WHEN. With this cluase, we can do binning rows by conditions.

```
SELECT [col_name1],
        CASE
        WHEN [condition_1] THEN [value_1]
        WHEN [condition_2] THEN [value_2]
        ELSE [value_3]
        END AS [new_col],
        ...
  FROM [table_name]
   ...;
```

Order of Clause

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY
7. LIMIT

1.2 Aggregate functions

Aggregate function is a function where the values of multiple rows are grouped together to form a single summary value. The common used aggregate functions is AVG(), COUNT(), MAX(), MIN(), MED(), SUM().

Systems of applying aggregate functions in tables works by rows. When we use WHERE clause SQL make bundles of rows in table making specific conditions and also GROUP BY clause follows this. Aggregation is applied on this bundles of rows.

```
SELECT SUM([col_name1]) AS [new_col1],
       ...
  FROM [table_name]
 WHERE [condition_1] ...
 GROUP BY [col1_name2]
   ...;
```

CAST

And SQL supports that standard arithmetic opertors(+, -, *, /), Arithmetic between two integers returns an integer so we need to change using CAST(AS Float)

```
SELECT CAST([col_name1] AS Float) / CAST([col_name2] AS Float) AS [new_col1],
       ...
  FROM ...
   ...;
```

ROUND

We can make value in specific number decimal places using ROUND().

```
SELECT ROUND([col_name1], 3) AS [rounded_col]
  FROM ...
   ...;
```

||

We can concatenate string type data into a single column using '||' opertor.

```
SELECT "string1 : " || [col_name1] AS [new_col],
       ...
  FROM ...
   ...;
```

1.3 Merge

We can join data from more than two tables by stacking JOIN clauses.

INNER JOIN

```
SELECT [alias_table_name1].[col_name1], [alias_table_name2].[col_name2], ...
  FROM [table_name1] AS [alias_table_name1]
  INNER JOIN [table_name2] AS [alais_table_name2] ON [alias_table_name1].[key1] = [alias_table_name2].[key2]
```

LEFT JOIN

```
SELECT [alias_table_name1].[col_name1], [alias_table_name2].[col_name2], ...
  FROM [table_name1] AS [alias_table_name1]
  LEFT JOIN [table_name2] AS [alais_table_name2] ON [alias_table_name1].[key1] = [alias_table_name2].[key2]
```

RIGHT JOIN

```
SELECT [alias_table_name1].[col_name1], [alias_table_name2].[col_name2], ...
  FROM [table_name1] AS [alias_table_name1]
  RIGHT JOIN [table_name2] AS [alais_table_name2] ON [alias_table_name1].[key1] = [alias_table_name2].[key2]
```

FULL OUTER JOIN

```
SELECT [alias_table_name1].[col_name1], [alias_table_name2].[col_name2], ...
    FROM [table_name1] AS [alias_table_name1]
    FULL OUTER JOIN [table_name2] AS [alais_table_name2] ON [alias_table_name1].[key1] = [alias_table_name2].[
key2]
```

UNION

UNION clause selects rows that occurs in one or more SELECT statements. This clause can stack duplicated rows among tables.

```
[select statement]
UNION
[select statement]
```

INTERSECT

INTERSECT clause selects rows that occurs both SELECT statement. This clause stacks rows without duplicated rows.

```
[select statement]
INTERSECT
[select statement]
```

EXCEPT

EXCEPT clause selects rows that occur in the first select statement but not the second select statement.

```
[select statement]
EXCEPT
[select statement]
```

1.4 Views

In a database, a view is the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object. This pre-established query command is kept in the database dictionary. Unlike ordinary base tables in a relational database, a view does not form part of physical schema: as a result set, it is a virtual table computed or collated dynamically from data in the database when access to that view is requested.

CREATE

```
CREATE VIEW [db_name].[view_name] AS
    SELECT ...
      FROM ...
       ...;
```

DROP

```
DROP VIEW [db_name].[view_name];
```

1.5 INSERT

An SQL INSERT statement adds one or more records to any single table in a relational database.

```
INSERT INTO [table_name] ([column_name1], [column_name2], ...)
     VALUES (([value11], [value12], ...),
             ([value21], [value22], ...),
             ...);
```

1.6 UPDATE

An SQL UPDATE statement changes the data of one or more records in a table. Either all the rows can be updated, or a subset may be chosen using a cnodition.

```
UPDATE [table_name]
SET
[column_name1] = [expression],
[column_name2] = [expression],
...
WHERE [condition_1];
```

1.7 DELETE

The DELETE statement removes one or more records from a table.

```
DELETE FROM [table_name]
 WHERE [condition_1];
```

2. DDL

2.1 CREATE

CREAT clause make tables in database.

```
CREATE TABLE [table_name] (
    [column_name1] [column_type1] PRIMARY KEY,
    [column_name2] [column_type2],
    FOREIGN KEY ([column_name2]) REFERENCE parent_table([column_name2])
    ...
);
```

There are some type we can define such as TEXT, INTEGER, REAL, NUMERIC, BLOB. 테이블 내부에는 데이터 타입과는 별개로 기본키(Primary Key)와 왜래키(Foreign Key)가 존재한다. 기본키는 테이블 내부에 고유한 값을 가지고 있는 열이면서 테이블을 식별하는 기준이 된다. 왜래키는 다른 테이블과 연결할 수 있는 열을 의미하며 다른 테이블의 기본키가 해당 테이블의 왜래키로 작동하게 된다.

2.2 ALTER

The ALTER statement modifies an existing database object. An ALTER statement in SQL changes the properties of an object inside of RDBMS. The types of objects that can be altered depends on which RDBMS is being used.

```
ALTER TABLE [table_name]
  ADD COLUMN [column_name1] [column_type1];

ALTER TABLE [table_name]
 DROP COLUMN [column_name];
```

2.3 DELETE

DELETE cluase drop tables in database

```
DELETE TABLE [table_name];
```

3. Subquery vs CTEs

3.1 Subqueries

Queries can be nested so that the results of one query can be used in another query via a relational operator or aggregation function. A nested query is also known as subquery. While joins and other table operations provide computationally superior alternatives in many cases, the use of subqueries introduces a hierachy in execution that can be useful or necessary.

```
SELECT [col_name1],
       (SELECT [col_name2] FROM [table_name2]) AS [new_col],
       ...
  FROM [table_name1]
 WHERE [col_name] > (SELECT FUNC([col_name3] FROM [table_name3])
   ...;
```

3.2 Common Table Expression

A common table exrpession, or CTE is a temporary named result set, derived from a simple query and defined within the execution scope of a SELECT, INSERT, UPDATE, or DELETE statement. CTEs cna be thought of as alternatives to subquery, views, and inline user-defined functions.

```
WITH [CTE_name1] AS
     (
        SELECT ...
        FROM [table_1]
        ...
     ),
     [CTE_name2] AS
     (
         ...
     )
SELECT ...
  FROM [CTE_name1]
   ...;
```

4. Skill Queries

4.1 Check if there are same values in two tables

두 테이블 내부에서 주키와 왜래키가 아닌 열의 원소들이 동일한지 확인하는 방법은 각 테이블의 차집합이 NULL값인지 연산 후 AND 연산자를 통해 결과가 0 False가 나오면 두 테이블의 행의 원소가 동일함을 알 수 있다.

```
(
SELECT t.track_id
  FROM track AS t
 WHERE t.album_id = (SELECT t2.album_id
                     FROM track AS t2
                     WHERE t2.track_id = ifs.first_track)

EXCEPT

SELECT il.track_id
  FROM invoice_line AS il
 WHERE il.invoice_id = (SELECT il2.invoice_id
                        FROM invoice_line AS il2
                        WHERE il2.invoice_id = ifs.invoice_id)
) IS NULL

AND

(
SELECT il.track_id
  FROM invoice_line AS il
 WHERE il.invoice_id = (SELECT il2.invoice_id
                        FROM invoice_line AS il2
                        WHERE il2.invoice_id= ifs.invoice_id)

EXCEPT

SELECT t.track_id
  FROM track AS t
 WHERE t.album_id = (SELECT t2.album_id
                     FROM track AS t2
                     WHERE t2.track_id = ifs.first_track)
) IS NULL
```

5. SQLite Shell

We can execute SQLite on shell using sqlite3 [db_name].db command. We can us dot commands for view options and using shell commands. Below is kinds of dot commands :

- .headers on : Show columns name
- .mode column : Return tidy table
- .help : Print all dot commands
- tables : Print all tables and views in database
- .shell [command] : We can execute shell commands
- .quit : quit SQLite Shell
- .schema [table_name] : Check schema of table

6. SQLite Python Script

We can use queries in Python using 'sqlite3' module. After we connect to db with sqlite3.connect(), we should close connection as we did on opening files.

```
import sqlite3
sqlite3.connect('[db_name].db')
cursor = conn.cursor()

queries = """
SELECT ...
  FROM ...
   ...;
"""

cursor.execute(query)
cursor.fetchall()
# cursor.fetchone() : return only row one
# cursor.fetchmany(n) : return n rows

conn.close()
```