Mysql Intermediate

Import sakila

1. sakila database download

- https://dev.mysql.com/doc/index-other.html

2. Import sakila database

$ mysql -u root -p

sql> create database sakila;

sql> quit

$ mysql -u root -p sakila < sakila-schema.sql

$ mysql -u root -p sakila < sakila-data.sql

1. CEIL, ROUND, TRUNCATE

CEIL, ROUND, TRUNCATE는 소수점 올림, 반올림, 버림 함수입니다.

1.1 CEIL

CEIL는 실수 데이터를 올림 할 때 사용합니다.

```
# 12.345를 올림하여 정수로 나타냄
SELECT CEIL(12.345)

# 국가별 언어 사용 비율을 소수 첫번째자리에서 올림하여 정수로 나타냄
SELECT CountryCode, Language, Percentage, CEIL(Percentage)
FROM countrylanguage
```

2. ROUND

ROUND는 실수데이터를 반올림 할 때 사용합니다.

```
# 12.345를 소수 둘째자리까지 나타내고 소수 셋째자리에서 반올림
SELECT ROUND(12.345, 2)

# 국가별 언어 사용 비율을 소수 첫번째자리에서 반올림하여 정수로 나타냄
SELECT CountryCode, Language, Percentage, ROUND(Percentage, 0)
FROM countrylanguage
```

3. TRUNCATE

TRUNCATE는 실수 데이터를 버림 할 때 사용합니다.

```
# 12.345를 소수 둘째자리까지 나타내고 소수 셋째자리에서 버림
SELECT TRUNCATE(12.345, 2)

# 국가별 언어 사용 비율을 소수 첫번째자리에서 버림하여 정수로 나타냄
SELECT CountryCode, Language, Percentage, TRUNCATE(Percentage, 0)
FROM countrylanguage

SELECT CountryCode, Language, Percentage, ROUND(Percentage, 0), TRUNCATE(Percentage, 0)
FROM countrylanguage
```

2. Conditional

SQL에서도 다른 언어에서 처럼 조건문 사용이 가능합니다. IF, CASE 에 대해서 설명합니다.

2.1 IF

2.1.1 syntax

IF(조건, 참, 거짓)

2.1.2 example

```
# 도시의 인구가 100만이 넘으면 "big city" 그렇지 않으면 "small city"를 출력하는 city_scale 컬럼을 추가
SELECT name, population, IF(population > 1000000, "big city", "small city") AS city_scale
FROM city
```

2.2 IFNULL

2.2.1 syntax

IFNULL(참, 거짓)

2.2.2 example

```
# 독립년도가 없는 데이터는 0으로 출력
SELECT IndepYear, IFNULL(IndepYear, 0) as IndepYear
FROM country
```

2.3 CASE

2.3.1syntax

```
CASE
        WHEN (조건1) THEN (출력1)
        WHEN (조건2) THEN (출력2)
```

END AS (컬럼명)

2.3.2 example

```
# 나라별로 인구가 10억 이상, 1억 이상, 1억 이하인 컬럼을 추가하여 출력
SELECT name, population,
        CASE
                WHEN population > 1000000000 THEN "upper 1 bilion"
                WHEN population > 100000000 THEN "upper 100 milion"
                ELSE "below 100 milion"
        END AS result
FROM country
```

3. DATE_FORMAT

DATE_FORMAT은 날짜 데이터에 대한 포멧을 바꿔줍니다.

https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html

```
# sakila 데이터 베이스에서 월별 총 수입
SELECT DATE_FORMAT(payment_date, "%Y-%m") AS monthly, SUM(amount) AS amount
FROM payment
GROUP BY monthly
```

4. JOIN

JOIN은 여러개의 테이블에서 데이터를 모아서 보여줄 때 사용됩니다. JOIN에는 INNER JOIN, LEFT JOIN, RIGHT JOIN이 있습니다.

4.1 MAKE TEST TABLE & DATA

```
# create table & data
```

```
CREATE TABLE user (
        user_id int(11) unsigned NOT NULL AUTO_INCREMENT,
        name varchar(30) DEFAULT NULL,
        PRIMARY KEY (user_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE addr (
        id int(11) unsigned NOT NULL AUTO_INCREMENT,
        addr varchar(30) DEFAULT NULL,
        user_id int(11) DEFAULT NULL,
        PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO user(name)
VALUES ("jin"),
("po"),
("alice"),
("petter");

INSERT INTO addr(addr, user_id)
VALUES ("seoul", 1),
("pusan", 2),
("deajeon", 3),
("deagu", 5),
("seoul", 6);
```

4.2 INNER JOIN

두 테이블 사이에 공통된 값이 없는 row는 출력하지 않는다.

```
# 두 테이블을 합쳐 id, name, addr 출력
SELECT id, user.name, addr.addr
```

```
FROM user
JOIN addr
ON user.user_id = addr.user_id

# world 데이터베이스에서 도시이름과 국가이름을 출력
SELECT country.name AS city_name, city.name AS country_name
FROM city
JOIN country
ON city.CountryCode = country.code
```

4.3 LEFT JOIN

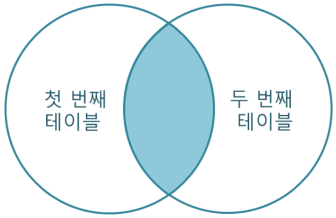
왼쪽 테이블을 기준으로 왼쪽 테이블의 모든 데이터가 출력되고 매핑되는 키값이 없으면 NULL로 출력된다.

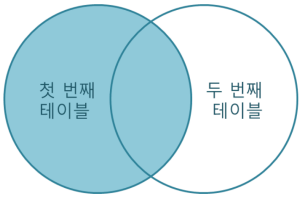
```
# 두 테이블을 합쳐 id, name, addr 출력
SELECT id, user.name, addr.addr
FROM user
LEFT JOIN addr
ON user.user_id = addr.user_id
```

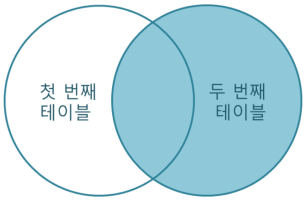
4.4 RIGHT JOIN

오른쪽 테이블을 기준으로 왼쪽 테이블의 모든 데이터가 출력되고 매핑되는 키값이 없으면 NULL로 출력된다.

```
# 두 테이블을 합쳐 id, name, addr 출력
SELECT id, user.name, addr.addr
FROM user
RIGHT JOIN addr
ON user.user_id = addr.user_id
```







inner join left join right join

5. UNION

UNION은 SELECT 문의 결과 데이터를 하나로 합쳐서 출력합니다. 컬럼의 갯수와 타입, 순서가 같아야 합니다.

UNION은 자동으로 distinct를 하여 중복을 제거해 줍니다. 중복제거를 안하고 컬럼 데이터를 합치고 싶으면 UNION ALL을 사용합니다.

또한 UNION을 이용하면 Full Outer Join을 구현할수 있습니다.

5.1 UNION

user 테이블의 name 컬럼과 addr 테이블의 addr 컬럼의 데이터를 하나로 함쳐서 출력

SELECT name

FROM user

UNION

SELECT addr

FROM addr

5.2 UNION ALL

중복데이터를 제거하지 않고 결과 데이터 합쳐서 출력

SELECT name

FROM user

UNION ALL

SELECT addr

FROM addr

5.3 FULL OUTER JOIN

```
# union을 이용하여 full outer join 구현
SELECT id, user.name, addr.addr
FROM user
LEFT JOIN addr
ON user.user_id = addr.user_id
UNION
SELECT id, user.name, addr.addr
FROM user
RIGHT JOIN addr
ON user.user_id = addr.user_id
```

6. Sub-Query

sub query는 query 문 안에 있는 query를 의미합니다. SELECT절 FROM절, WHERE 등에 사용이 가능합니다.

```
# 전체 나라수, 전체 도시수, 전체 언어수를 출력 ( SELECT 절에 사용 )
SELECT
    (SELECT count(name) FROM city) AS total_city,
    (SELECT count(name) FROM country) AS total_country,
    (SELECT count(DISTINCT(Language)) FROM countrylanguage) AS total_language
FROM DUAL

# 800만 이상되는 도시의 국가코드, 이름, 도시인구수를 출력 ( FROM 절에 사용 )
SELECT *
FROM
    (SELECT countrycode, name, population
    FROM city
    WHERE population > 8000000) AS city
JOIN
```

```
        (SELECT code, name
        FROM country) AS country
ON city.countrycode = country.code

# 800만 이상 도시의 국가코드, 국가이름, 대통령이름을 출력( WHERE 절에 사용 )
SELECT code, name, HeadOfState
FROM country
WHERE code IN (
        SELECT DISTINCT(countrycode) FROM city WHERE population > 8000000
)
```

7. Index

테이블에서 데이터를 검색할때 빠르게 찾을수 있도록 해주는 기능입니다. where 절에 들어가는 컬럼을 index로 설정해 놓으면 설정한 컬럼을 조건으로 검색할때 빠르게 검색할수 있습니다. 자주 검색하는 조건은 index로 설정해 놓으면 좋지만 너무 많은 인덱스가 설정되게 되면 데이터가 입력 될때마다 index에 데이터를 넣어야 함으로 데이터 입력시 속도가 느려질수 있습니다. 그러므로 인덱스는 검색 조건으로 자주 사용하는 컬럼에 설정해 놓으면 좋습니다.

7.1 syntax

```
CREATE INDEX 인덱스이름
ON 테이블이름 (컬럼이름1, 컬럼이름2)
```

7.2 example

```
# city 테이블에 Population 컬럼을 인덱스로 추가
CREATE INDEX Population
ON city (Population)

# city 테이블에 Population 인덱스 제거
DROP INDEX Population
ON city
```

7.3 explain

query를 실행하기 전에 index로 검색을 하는지 확인 할수 있습니다.

100만이 넘는 도시의 데이터를 출력의 실행계획을 확인

EXPLAIN

SELECT *

FROM city

WHERE population > 1000000

Extra 컬럼에 Index가 없으면 Using where로 검색이 되지만 Index가 있으면 Using index로 검색됨을 확인할 수 있습니다.

Type 컬럼의 값이 ALL이면 Index를 사용하지 않고 있다는 의미 입니다.

검색 절차

스토리지(Index + Data) -> (rows)-> Mysql 엔진(필터링 전) -> (filtered) -> Mysql 엔진(필터링 후)

7.4 emplyees의 salaries로 테스트

employees.sql을 데이터 베이스에 저장

employees.sql 파일을 sftp를 이용하여 ubuntu 서버로 이동

ubuntu 서버에서 mysql 로그인

$ mysql -u root -p

mysql> create database employees;

mysql> use employees;

mysql> source employees.sql

employees 데이터 베이스로 이동

USE employees

salaries 테이블의 to_date 컬럼을 인덱스로 설정

```
CREATE INDEX tdate
ON salaries (to_date)

# salaries 테이블에 tdate 인덱스 제거
DROP INDEX tdate
ON salaries

# 실행계획 확인 및 쿼리 실행
EXPLAIN
SELECT count(*)
FROM salaries
WHERE to_date > "2000-01-01"
```

8. View

가상 테이블로 특정 데이터만 보고자 할때 사용합니다. 실제 데이터를 저장하고 있지는 않습니다. 한마디로 특정 컬럼의 데이터를 보여주는 역할만 합니다. 뷰를 사용 함으로 쿼리를 더 단순하게 만들수 있습니다. 한번 생성된 뷰는 수정이 불가능 하며 인덱스설정이 불가능 합니다.

8.1 syntax

```
CREATE VIEW <뷰이름> AS
(QUERY)
```

8.2 example

```
# 국가코드와 국가이름이 있는 뷰 생성
CREATE VIEW code_name AS
SELECT code, name
FROM country

# city 테이블에 국가 이름 추가
```

```
SELECT *
FROM city
JOIN code_name
ON city.countrycode = code_name.code
```