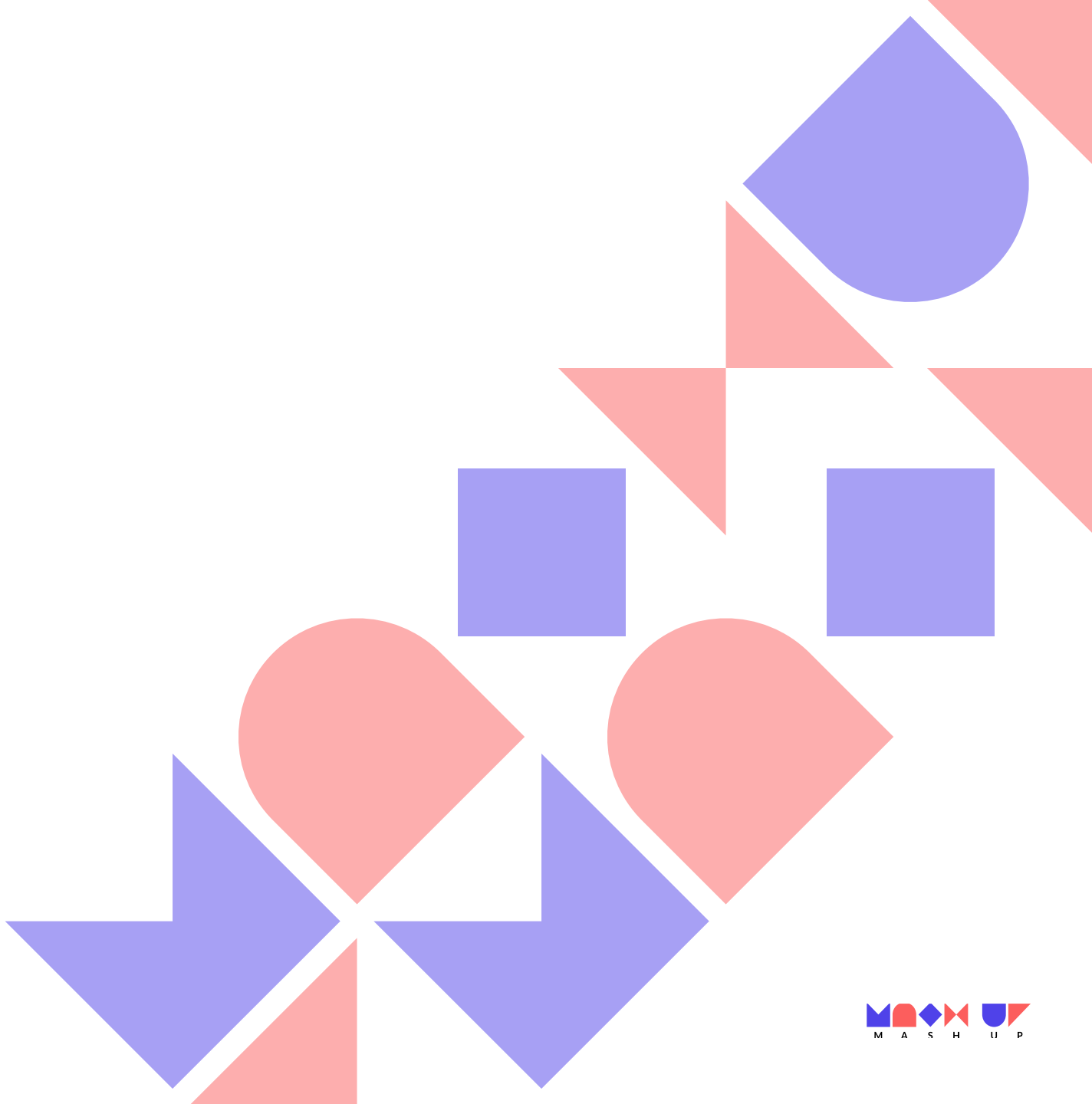
Back-End Study

SQL

2018. 11. 3

백엔드팀 이동준



CONTENTS

Oracle

MongoDB

SQL

NOSQL

MySQL

Redis

ORM

JPA

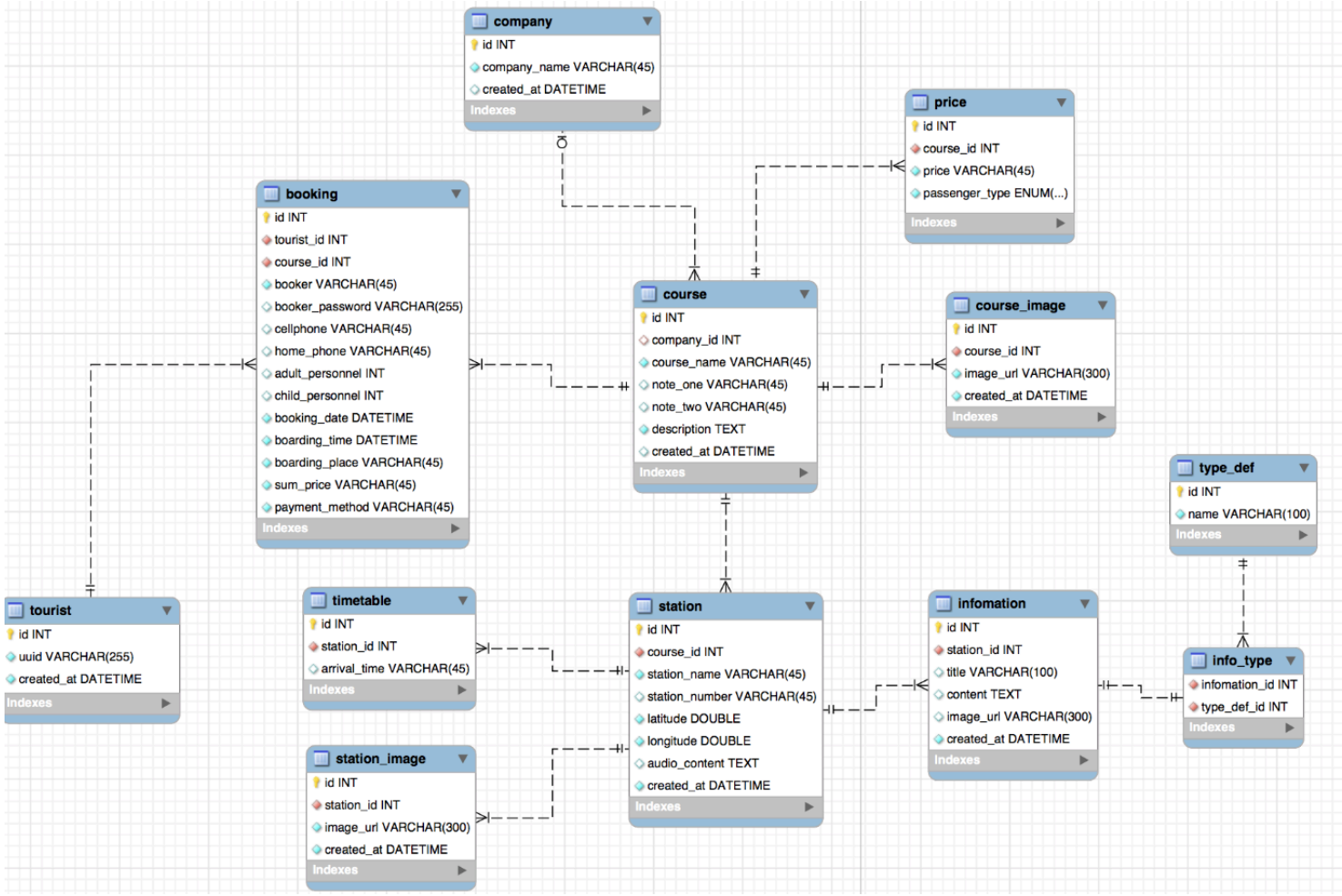
Django ORM

데이터베이스 설계

모델링

ER-다이어그램

정규화(1차, 2차, 3차....) 관계(일대일,일대다,다대다..)


company
id INT
company_name VARCHAR(45)
created_at DATETIME
Indexes
price
id INT
course_id INT
price VARCHAR(45)
passenger_type ENUM(...)
Indexes
booking
id INT
tourist_id INT
course_id INT
booker VARCHAR(45)
booker_password VARCHAR(255)
cellphone VARCHAR(45)
home_phone VARCHAR(45)
adult_personnel INT
child_personnel INT
booking_date DATETIME
boarding_time DATETIME
boarding_place VARCHAR(45)
sum_price VARCHAR(45)
payment_method VARCHAR(45)
Indexes
course
id INT
company_id INT
course_name VARCHAR(45)
note_one VARCHAR(45)
note_two VARCHAR(45)
description TEXT
created_at DATETIME
Indexes
course_image
id INT
course_id INT
image_url VARCHAR(300)
created_at DATETIME
Indexes
type_def
id INT
name VARCHAR(100)
Indexes
tourist
id INT
uuid VARCHAR(255)
created_at DATETIME
Indexes
timetable
id INT
station_id INT
arrival_time VARCHAR(45)
Indexes
station
id INT
course_id INT
station_name VARCHAR(45)
station_number VARCHAR(45)
latitude DOUBLE
longitude DOUBLE
audio_content TEXT
created_at DATETIME
Indexes
infomation
id INT
station_id INT
title VARCHAR(100)
content TEXT
image_url VARCHAR(300)
created_at DATETIME
Indexes
info_type
infomation_id INT
type_def_id INT
Indexes
station_image
id INT
station_id INT
image_url VARCHAR(300)
created_at DATETIME
Indexes

모델링

01

SQL

- SQL
- SQL 키워드
- SQL 실행순서

관계형 데이터 베이스

RDBMS

SQL

절차지향 X

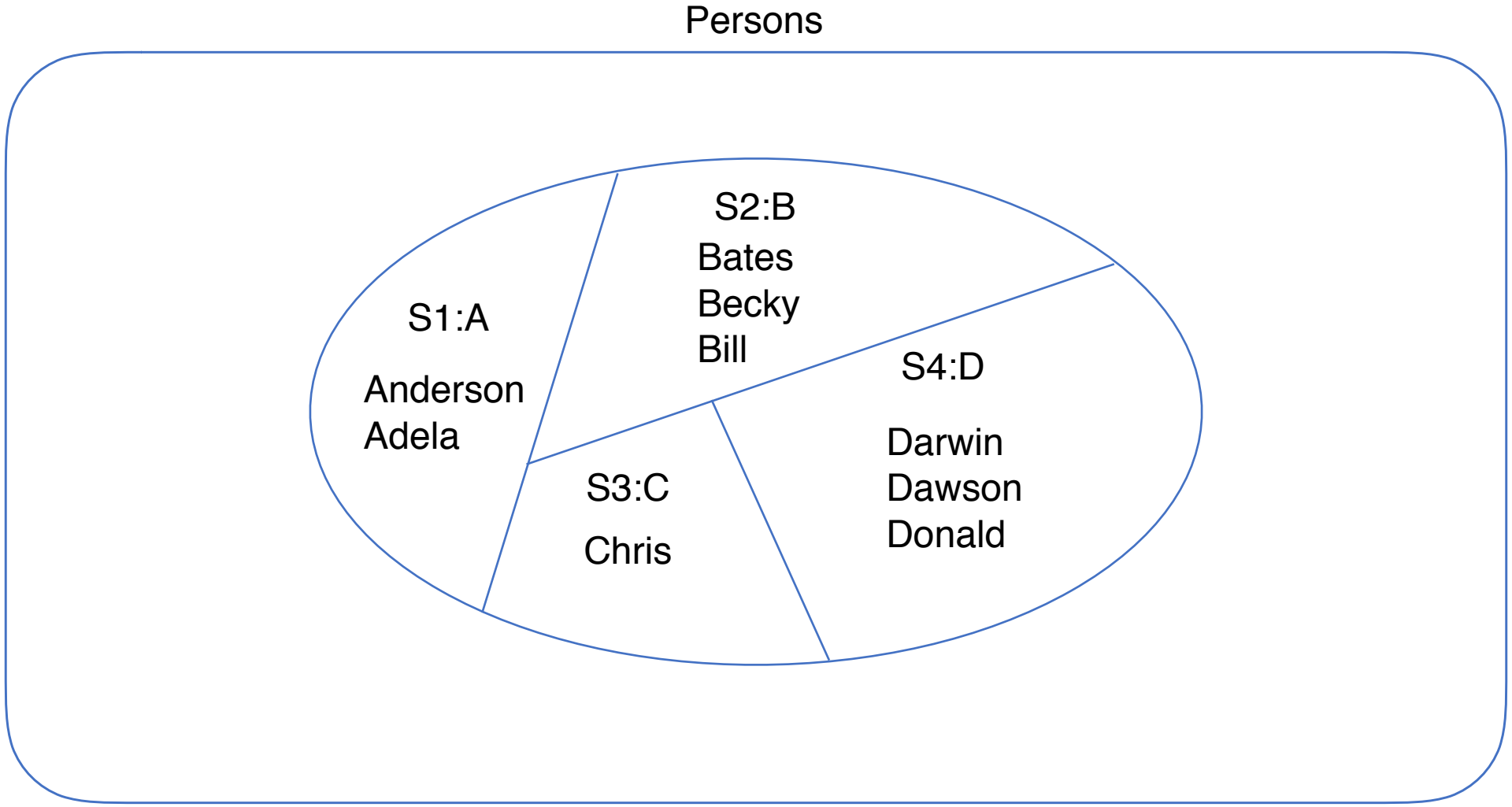
집합지향 O

Persons

name(이름)	age(나이)	height(키)	weight(몸무게)
Anderson	30	188	90
Adela	21	167	55
Bates	87	158	48
Becky	54	187	70
Bill	39	177	120
Chris	90	175	48
Darwin	12	160	55
Dawson	25	182	90
Donald	30	176	53


Persons
Anderson
Adela
Bill
Becky
Bates
Chris
Dawson
Darwin
Donald

```
SELECT SUBSTRING(name, 1, 1) AS label,
COUNT(*)
FROM Persons
GROUP BY SUBSTRING(name, 1,1)
```


Persons
S1:A
Anderson
Adela
S2:B
Bates
Becky
Bill
S3:C
Chris
S4:D
Darwin
Dawson
Donald

SQL 키워드

TABLE

VIEW

결합

집약함수

INDEX

윈도우함수

서브쿼리

등....

집합

CRUD

CREATE READ UPDATE DELETE

SQL 구문

INSERT

DELETE

UPDATE

*SELECT

SQL 구문

INSERT INTO 테이블명 (value1, value2) values(…..)

DELETE FROM 테이블명 WHERE 조건

UPDATE SET 속성명=데이터… WHERE 조건

SQL 구문

SELECT 속성명 FROM 테이블명

WHERE 조건

GROUP BY 속성명

HAVING 조건

ORDER BY 속성명

SELECT 구문

서브쿼리

IN()

VIEW

상관쿼리

OVER(PARTITION BY)

UINON

CASE

SUM(), MAX(), MIN()….

등….

SELECT 구문 실행 순서

SELECT 속성명 ← 5

FROM 테이블명 ← 1

WHERE 조건 ← 2

GROUP BY 속성명 ← 3

HAVING 조건 ← 4

ORDER BY 속성명 ← 6

SELECT 실행 순서

FROM → WHERE → GROUP BY →

HAVING → SELECT → ORDER BY

실행 순서를 왜 알아야 할까?

성능!!!

올바른 쿼리문 작성!!!

02

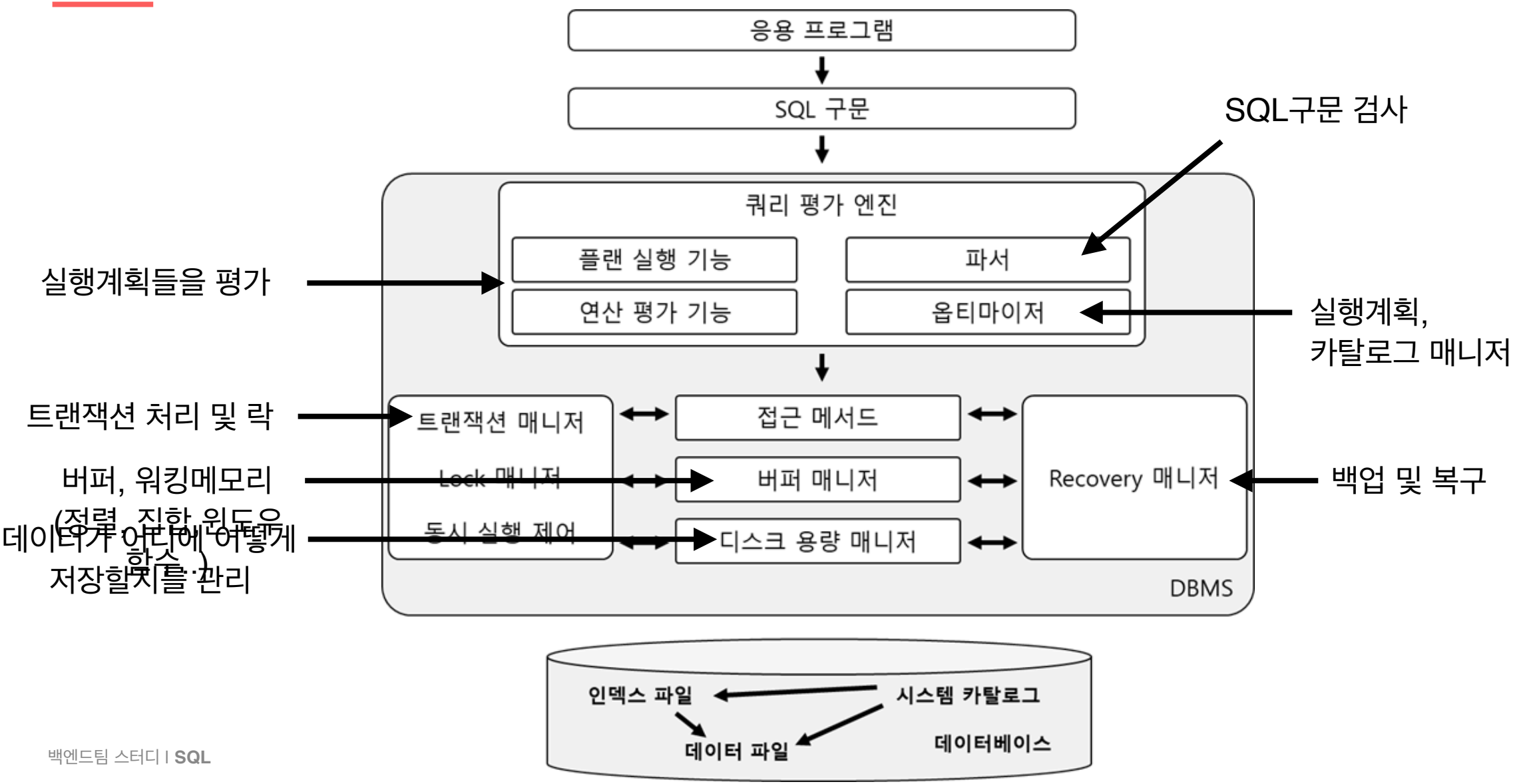
DBMS

- DBMS
- DBMS Architecture
- 실행계획

DBMS란?

데이터베이스 관리 시스템

DBMS Architecture



실행계획

옵티 마이저

카탈로그 매니저

성능(I/O 접근)

DBMS 실행계획 확인법

Oracle => Explain plan for SQL 구문

Mysql => Explain SQL 구문

PostgreSQL => Explain SQL 구문

....

실행계획 예제

```
EXPAINT PLAN FOR

SELECT e.ename, d.dname

FROM emp e, dept d

WHER e.deptno = d.deptno
```

실행계획 예제

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	308	6 (17)	00:00:01
1	MERGE JOIN		14	308	6 (17)	00:00:01
2	TABLE ACCESS BY INDEX ROWID	DEPT	4	52	2 (0)	00:00:01
3	INDEX FULL SCAN	PK_DEPT	4		1 (0)	00:00:01
* 4	SORT JOIN		14	126	4 (25)	00:00:01
5	TABLE ACCESS FULL	EMP	14	126	3 (0)	00:00:01

03

성능

- 조건 분기
- 반복문
- 등등

조건분기

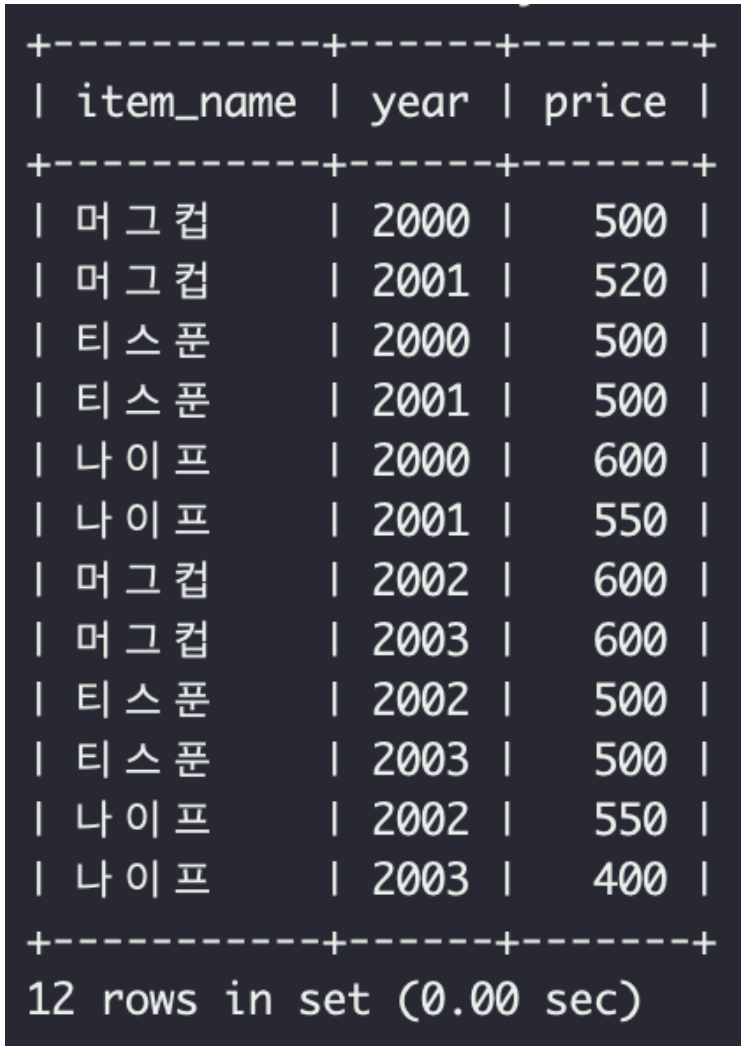
실행계획을 상상하면서 짜라

조건 분기를 WHERE 구로 하는 사람들은 초보자다

잘 하는 사람은 SELECT 구만으로 조건 분기를 한다.

UNION vs CASE

```
mysql> select * from items;
+---------+------+-----------+--------------+--------------+
| item_id | year | item_name | price_tax_ex | price_tax_in |
+---------+------+-----------+--------------+--------------+
|     100 | 2000 | 머그컵    |          500 |          525 |
|     100 | 2001 | 머그컵    |          520 |          546 |
|     100 | 2002 | 머그컵    |          600 |          630 |
|     100 | 2003 | 머그컵    |          600 |          630 |
|     101 | 2000 | 티스푼    |          500 |          525 |
|     101 | 2001 | 티스푼    |          500 |          525 |
|     101 | 2002 | 티스푼    |          500 |          525 |
|     101 | 2003 | 티스푼    |          500 |          525 |
|     102 | 2000 | 나이프    |          600 |          630 |
|     102 | 2001 | 나이프    |          550 |          577 |
|     102 | 2002 | 나이프    |          550 |          577 |
|     102 | 2003 | 나이프    |          400 |          420 |
+---------+------+-----------+--------------+--------------+
```



item_name	year	price
머그컵	2000	500
머그컵	2001	520
티스푼	2000	500
티스푼	2001	500
나이프	2000	600
나이프	2001	550
머그컵	2002	600
머그컵	2003	600
티스푼	2002	500
티스푼	2003	500
나이프	2002	550
나이프	2003	400

12 rows in set (0.00 sec)

2001년 까지는 세금이 포함 되지 않은 가격을,
2002년 부터는 세금이 포함된 가격을
‘price’필드로 표시하게 해라

UNION

```
SELECT item_name, year, price_tax_ex AS price
    FROM Items
    WHERE year <= 2001

UNION ALL

SELECT item_name, year, price_tax_ex AS price
    FROM Items
    WHERE year >= 2002;
```

CASE

```
SELECT item_name, year,
    CASE WHEN year <= 2001 THEN price_tax_ex
         WHEN year >= 2002 THEN price_tax_in
         END AS price
  FROM Items
  WHERE year <= 2001;
```

실행 계획 비교

```
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | PRIMARY     | Items | NULL       | ALL  | NULL          | NULL | NULL    | NULL |   12 |    33.33 | Using where |
|  2 | UNION       | Items | NULL       | ALL  | NULL          | NULL | NULL    | NULL |   12 |    33.33 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
2 rows in set, 1 warning (0.01 sec)
```

UNION

```
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | Items | NULL       | ALL  | NULL          | NULL | NULL    | NULL |   12 |    33.33 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

CASE

조건분기 결론

- 테이블 full(all) scan이 두번(UNION), 한번(CASE)

- index가 구성되어 있다면? 두번의 index scan, 한번의 full(all) scan

- sql문을 작성시 실행계획을 생각하면서 작성하여야 한다.

반복문

company	year	sale
A	30	188
A	21	167
A	87	158
A	54	187
B	39	177
B	90	175
B	12	160
B	25	182

company	year	sale	var

- 이전 데이터가 없을 경우 : NULL
- 이전 데이터보다 매출이 올랐을 경우 : +
- 이전 데이터보다 매출이 내렸을 경우 : -
- 이전 데이터와 매출이 동일한 경우 : =

반복문

- 이전 데이터가 없을 경우 : NULL
- 이전 데이터보다 매출이 올랐을 경우 : +
- 이전 데이터보다 매출이 내렸을 경우 : -
- 이전 데이터와 매출이 동일한 경우 : =

company	year	sale	var
A	2002	50	
A	2003	52	+
A	2004	55	+
A	2007	55	+
B	2001	27	
B	2005	28	+
B	2006	28	=
B	2009	30	+

함수 생성(반복문/LOOP)

```
CREATE OR REPLACE PROCEDURE PROC_INSERT_VAR
IS

/*커서 선언*/
CURSOR C_slaes IS
    SELECT company, year, sale
        FROM Sales
    ORDER BY company, year;

/*레코드 타입 선언*/
res_sales c_sales%ROWTYPE;

/*카운터*/
i_pre_sale INTEGER := 0;
c_company CHAR(1) := '*';
c_var CHAR(1) := '*';
```

```
BEGIN

OPER c_sales;
    LOOP
    /*레코드를 패치해서 변수에 대입*/
    fetch c_sales into rec_sales;
    /*레코드 없다면 반복을 종료*/
    exit when c_sales%notfound;

    IF (C_company = rec_sales.company) THEN
        IF(i_pre_sale < rec_slaes.slae>) THEN
            c_var := '+';
        ELSIF(i_pre_sale > rec_sales.sale) THEN
            c_var := '-';
        ELSE
            c_var := '=';
....
```

집합 지향(포장계)

```
INSERT INTO Sales2
SELECT company, year, sale,
        CASE SIGN(sale - MAX(sale)
                OVER ( PARTITION BY company
                        ORDER BY year
                        ROWS BETWEEN 1 PRECEDING
                                AND 1 PRECEDING) )
        WHEN 0 THEN '='
        WHEN 1 THEN '+'
        WHEN -1 THEN '-'
        ELSE NULL END AS var
FROM Sales;
```

윈도우 함수 옵션
현재 레코드에서 1개 이전 부터 1개 이전까지의 레코드 범위

SQL - 반복문

https://github.com/korea8378/TIL/blob/master/SQL/%EC%8B%A4%ED%96%89%EA%B3%84%ED%9A%8D.md

결합

JOIN

크로스 결합 내부 결합 외부 결합 자기 결합

결합 알고리즘

서브쿼리

데이터 I/O 비용 발생

최적화 X(옵티마이저)

연산 비용 추가

윈도우 함수나 다른 방법으로 해결 가능

서브쿼리 사용이 더 나은 경우도 있다

인덱스

순번

갱신

데이터 모델

등등

책 추천


하루 30분 36강으로 배우는
완전 초보의 SQL 따라잡기
SQL
30
36
첫걸음
아사이 아츠시 지음
박준용 옮김
한빛미디어


DB 성능 최적화를 위한
SQL 실전 가이드
SQL
레벨업
미크 지음
윤인성 옮김
한빛미디어

THANKS FOR WATCHING

감사합니다

2018.11.3

백엔드팀 이동준