

## 5장. 정규화



# 이상 현상(anomaly)

## 이상현상이란?

잘못 설계된 테이블로 삽입, 삭제, 수정 같은 데이터 조작을 하면 이상현상이 일어난다. 이상현상(anomaly)이란 테이블에 튜플을 삽입할 때 부득이하게 NULL값이 입력(**삽입 이상**)되거나, 삭제시 연쇄 삭제 현상(**삭제 이상**)이 발생하거나, 수정 시 데이터의 일관성이 훼손(**수정 이상**)되는 현상을 말한다.

데이터베이스 설계가 잘못되면 이상현상, 즉 SQL문의 결과가 틀리거나 원하는 결과가 나오지 않는 등의 문제가 발생한다.

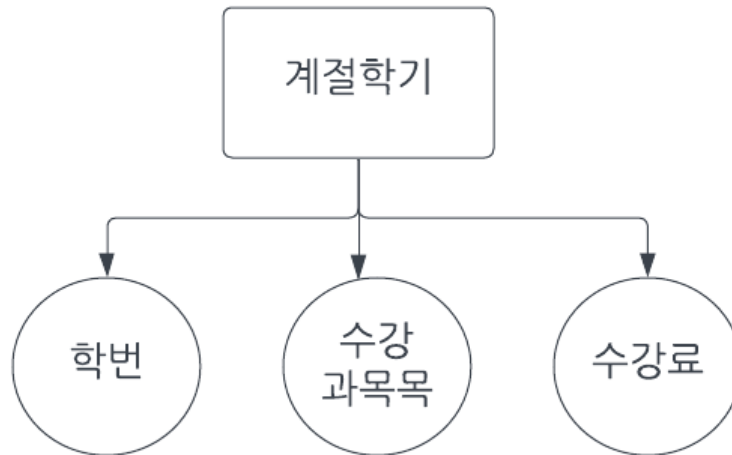


# 이상 현상(anomaly)

- 계절학기 수강 테이블

계절학기 수강 정보를 저장하는 summer\_class 테이블이다. 계절학기는 한 학생이 한 과목만 신청할 수 있다.

summer\_calss 테이블은 sid(학번), subject(수강과목), price(수강료)로 구성되어 있다.



# 이상 현상(anomaly)

- 계절학기 수강 테이블

summer\_class

sid	subject	price
100	C	30000
101	Java	45000
200	Python	40000
201	Java	45000



# 이상 현상(anomaly)

- 계절학기 수강 테이블

```
-- 여름(계절) 학기 테이블 생성
CREATE TABLE summer_class(
    sid    NUMBER,          -- 학번
    subject VARCHAR2(20),   -- 과목
    price  NUMBER           -- 수강료
);

INSERT INTO summer_class VALUES (100, 'C', 30000);
INSERT INTO summer_class VALUES (101, 'Java', 45000);
INSERT INTO summer_class VALUES (200, 'Python', 40000);
INSERT INTO summer_class VALUES (201, 'Java', 45000);
```

	SID	SUBJECT	PRICE
1	100	C	30000
2	101	Java	45000
3	200	Python	40000
4	201	Java	45000



# 이상 현상(anomaly)

- 질의 및 SQL 문

```
-- 계절학기를 듣는 학생의 학번과 수강하는 과목은?  
SELECT sid, subject  
FROM summer_class;  
  
-- python 강좌의 수강료는?  
SELECT price  
FROM summer_class  
WHERE subject = 'Python';  
  
-- 여름 학기를 듣는 학생 수와 수강료 총액은?  
SELECT COUNT(*) 학생수,  
       SUM(price) 수강료총액  
FROM summer_class;
```



# 이상 현상(anomaly)

- 삭제 이상

```
-- 삭제 이상 --  
-- 200번 학생이 수강신청을 취소하여 삭제  
DELETE FROM summer_class WHERE sid = 200;  
  
-- -- python 강좌의 수강료는 조회 안됨  
SELECT price  
FROM summer_class  
WHERE subject = 'Python';
```



# 이상 현상(anomaly)

- 삽입 이상

```
-- 삽입 이상 --  
-- C++ 강좌 개설, 수강료는 40000 아직 신청한 학생 없음  
INSERT INTO summer_class VALUES (NULL, 'C++', 35000);  
  
-- NULL값이 있는 경우 튜플은 5개이지만 수강학생은 총 4명임  
SELECT COUNT(*) 수강인원  
FROM summer_class;  
  
SELECT COUNT(sid) 수강인원  
FROM summer_class;  
  
SELECT COUNT(*) 수강인원  
FROM summer_class  
WHERE sid IS NOT NULL;
```





# 이상 현상(anomaly)

## ■ 수정 이상

```
-- Java 강좌 수강료가 45,000원에서 40,000원으로 변경되어 수정함
UPDATE summer_class
SET price = 40000
WHERE subject = 'Java';

SELECT * FROM summer_class;
-- 복구
ROLLBACK;

-- 만약 UPDATE문을 다음과 같이 작성하면 데이터 불일치 문제 발생함
-- 1건만 수정
UPDATE summer_class
SET price = 40000
WHERE subject LIKE 'Java' AND sid = 101;
```

```
-- Java 수강료 조회
SELECT price Java수강료
FROM summer_class
WHERE subject LIKE 'Java';
```



# 이상 현상(anomaly) 제거

- 수정된 계절학기 수강 테이블

**summer\_price**

subject	price
C	30000
Java	45000
Python	40000

**summer\_register**

sid	subject
100	C
101	Java
200	Python
201	Java



# 이상 현상(anomaly) 제거

- 수정된 계절학기 수강 테이블

```
CREATE TABLE summer_price(  
    subject VARCHAR2(20),  
    price    NUMBER  
);  
  
INSERT INTO summer_price VALUES ('C', 30000);  
INSERT INTO summer_price VALUES ('Java', 45000);  
INSERT INTO summer_price VALUES ('Python', 40000);
```



# 이상 현상(anomaly) 제거

- 수정된 계절학기 수강 테이블

```
CREATE TABLE summer_register(  
    sid      NUMBER,  
    subject VARCHAR2(20)  
);  
  
INSERT INTO summer_register VALUES (100, 'C');  
INSERT INTO summer_register VALUES (101, 'Java');  
INSERT INTO summer_register VALUES (200, 'Python');  
INSERT INTO summer_register VALUES (201, 'Java');
```



# 이상 현상(anomaly) 제거

- 수정된 계절학기 수강 테이블

```
-- 계절 학기를 듣는 학생의 학번과 수강하는 과목은?
```

```
SELECT sid, subject  
FROM summer_register;
```

```
-- Python 강좌의 수강료는?
```

```
SELECT price  
FROM summer_price  
WHERE subject = 'Python';
```

```
-- 수강료가 가장 비싼 과목은?
```

```
SELECT subject  
FROM summer_price  
WHERE price = (SELECT MAX(price) FROM summer_price);
```

```
-- 계절학기를 듣는 총 학생수와 수강료 총액은?
```

```
SELECT COUNT(*) 총학생수, SUM(price) 수강료총액  
FROM summer_price;
```



# 이상 현상(anomaly) 제거

- 삽입, 삭제 이상 없음

```
-- 삭제 이상 없음
-- 200번 학생의 수강 신청을 취소하시오
DELETE FROM summer_register
WHERE sid = 200;

-- Python 과목의 수강료는?
SELECT price Python수강료
FROM summer_price
WHERE subject = 'Python';

-- 삽입이상 없음
-- C++ 강좌와 수강료 35,000원 삽입
INSERT INTO summer_price VALUES ('C++', 35000);

-- 수강 신청 정보 확인
SELECT * FROM summer price;
```



# 이상 현상(anomaly) 제거

- 수정 이상 없음

```
-- 수정이상 없음
-- Java 과목의 수강료를 45,000원에서 40,000원으로 변경
UPDATE summer_price
SET price = 40000
WHERE subject = 'Java';

SELECT price Java수강료
FROM summer_price
WHERE subject = 'Java';
```



# 정규화(normalization)

- 정규화

이상 현상이 발생하는 테이블을 수정하여 정상으로 만드는 과정을 정규화(normalization)라 한다.

이상 현상의 원인은 대부분 두 가지 이상의 정보가 한 릴레이션에 저장되어 있기 때문에 발생한다. 따라서 이상현상은 릴레이션(테이블)을 분해하여 제거한다.

분해된 릴레이션에 이상현상이 남아있다면 이상현상이 없어질때 까지 분해한다.





# 정규화(normalization)

## ■ 제1정규형 위배

제1 정규형은 릴레이션의 속성 값이 원자값이어야 한다는 조건이다.

원자값이란 더 이상 분해되지 않는 값을 의미한다.

모든 속성은 반드시 하나의 값만 가져야 한다.

연예인에서 양동근은 직업이 배우이자 가수로 원자값이 아니다.

연예인

이름	생년월일	직업
양동근	19790601	배우, 가수
박은빈	19920904	배우
장기하	19820220	가수, 작가



# 정규화(normalization)

- 제1정규형으로 변환

연예인

이름	생년월일
양동근	19790601
박은빈	19920904
장기하	19820220

직업

이름	직업
양동근	배우
양동근	가수
박은빈	배우
장기하	가수
장기하	작가



# 정규화(normalization)

- entertainer 테이블

```
CREATE TABLE entertainer(  
    name          VARCHAR2(20),  
    birthday      VARCHAR2(20),  
    job           VARCHAR2(50)  
);  
  
INSERT INTO entertainer VALUES ('양동근', '19790601', '배우,가수');  
INSERT INTO entertainer VALUES ('박은빈', '19920904', '배우');  
INSERT INTO entertainer VALUES ('장기하', '19820220', '가수,작가');
```



# 정규화(normalization)

```
SELECT * FROM entertainer
WHERE job = '가수';
```

실행 결과 x

SQL | 인출된 모든 행: 0(0.003초)

NAME	BIRTHDAY	JOB
------	----------	-----

Job이 가수인 데이터를 추출하는 일이 매우 번거로운 일이 될 수 있다.



# 정규화(normalization)

```
CREATE TABLE entertainer(  
    name          VARCHAR2(20),  
    birthday      VARCHAR2(20)  
);  
  
INSERT INTO entertainer VALUES ('양동근', '19790601');  
INSERT INTO entertainer VALUES ('박은빈', '19920904');  
INSERT INTO entertainer VALUES ('장기하', '19820220');
```

	NAME	BIRTHDAY
1	양동근	19790601
2	박은빈	19920904
3	장기하	19820220



# 정규화(normalization)

```
CREATE TABLE job(  
    name      VARCHAR2(20),  
    job       VARCHAR2(50)  
);  
  
INSERT INTO job VALUES ('양동근', '배우');  
INSERT INTO job VALUES ('양동근', '가수');  
INSERT INTO job VALUES ('박은빈', '배우');  
INSERT INTO job VALUES ('장기하', '가수');  
INSERT INTO job VALUES ('장기하', '작가');
```

	NAME	JOB
1	양동근	배우
2	양동근	가수
3	박은빈	배우
4	장기하	가수
5	장기하	작가

```
SELECT * FROM job WHERE job = '가수';
```

리포트 출력 x 질의 결과 x  
SQL | 인출된 모든 행: 2(0,005초)

NAME	JOB
양동근	가수
장기하	가수

# 정규화(normalization)

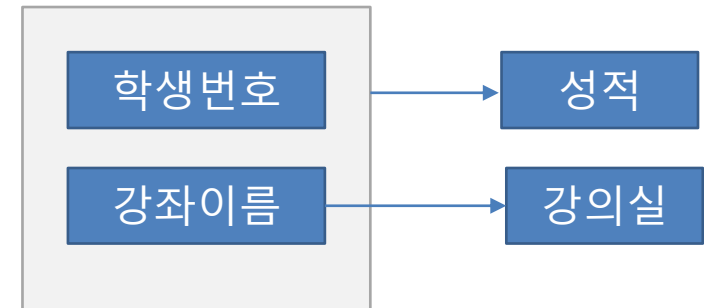
## ▪ 제2정규형

엔티티의 모든 일반 속성은 반드시 모든 주식별자에 종속되어야 한다.

주식별자가 단일 식별자가 아닌 복합식별자인 경우 일반 속성이 주식별자의 일부에만 종속될 수 있어 이상 현상이 발생함

수강강좌

학생번호	강좌이름	강의실	성적
501	데이터베이스	공학관 110	3.5
401	데이터베이스	공학관 110	4.0
402	경영학	상경관 103	3.5
502	웹 프로그래밍	공학관 111	4.0
501	웹 프로그래밍	공학관 111	3.5



# 정규화(normalization)

## ■ 이상 현상

수강강좌 릴레이션의 기본키는 (학생번호, 강좌이름)이며, 기본키의 일부인 강좌이름 속성이 강의실을 결정하는 '강좌이름 → 강의실' 종속관계를 가지고 있다.

- 삭제 이상 : 402번 학생이 수강을 취소하면 경영학 과목의 강의실에 대한 정보가 사라진다.
- 삽입 이상 : 인공지능 과목이 개설되어 공학관 112호를 사용하게 되었는데 아직 신청한 학생이 없다. 이 경우 수강강좌 릴레이션에 학생번호와 성적을 NULL 값으로 삽입해야 하는 문제가 발생한다.
- 수정 이상 : 데이터베이스 강의실을 공학관 113호로 변경할 경우 데이터 불일치가 발생할 가능성이 있다.





# 정규화(normalization)

- 이상 현상의 원인

이상현상의 원인은 수강강좌 릴레이션의 함수 종속성 다이어그램을 보면 알 수 있다. 수강강좌 릴레이션의 기본키는 (학생번호, 강좌이름)이고, 기본키가 아닌 속성은 성적, 강의실이다.

성적과 강의실은 모두 기본키에 함수적으로 종속되어 있지만 강의실은 기본키의 일부분인 강좌 이름에 한 번 더 종속되어 있다. 즉 기본키의 부분집합인 강좌이름에 종속된다. 이와 같이 기본키가 아닌 속성이 기본키에 완전 함수 종속이 아닌 불완전 함수 종속되어 있으면 이상현상이 발생한다

(학생번호, 강좌이름) 강의실 종속성의 경우 학생번호를 제거해도 '강좌이름  $\rightarrow$  강의실' 종속성은 여전히 성립한다.



# 정규화(normalization)

- 제2정규형으로 변환

수강강좌(학생번호, 강좌이름, 강의실, 성적)

→ 수강(학생번호, 강좌이름, 성적), 강의실(강좌이름, 강의실)

수강

학생번호	강좌이름	성적
501	데이터베이스	3.5
401	데이터베이스	4.0
402	경영학	3.5
502	웹 프로그래밍	4.0
501	웹 프로그래밍	3.5



# 정규화(normalization)

- 제2정규형으로 변환

수강강좌(학생번호, 강좌이름, 강의실, 성적)

→ 수강(학생번호, 강좌이름, 성적), 강의실(강좌이름, 강의실)

강의실

강좌이름	강의실
데이터베이스	공학관 110
경영학	상경관 103
웹 프로그래밍	공학관 111



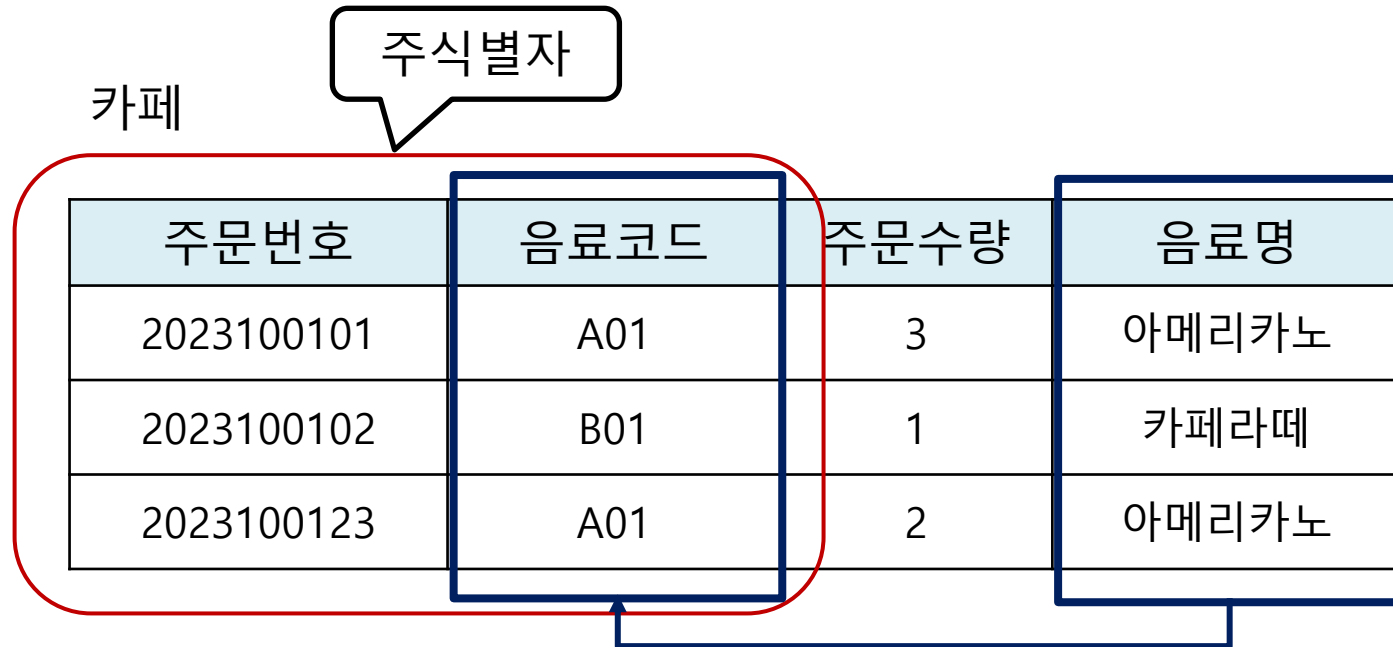
# 제2정규형 실습 예제

- 제2정규형 위배

카페

주식별자

주문번호	음료코드	주문수량	음료명
2023100101	A01	3	아메리카노
2023100102	B01	1	카페라떼
2023100123	A01	2	아메리카노



일반속성인 음료명이 주식별자 중  
음료코드 속성에 종속됨



# 제2정규형 실습 예제

- 제2정규형으로 변환

주문

주문번호	음료코드	주문수량
2023100101	A01	3
2023100102	B01	1
2023100123	A01	2

음료

음료코드	음료명
A01	아메리카노
B01	카페라떼



# 제2정규형 실습 예제

1. Lucid Chart를 사용하여 ERD를 작성하세요  
(ERD와 IE 표기법으로 2가지로 작성)
2. 테이블 표를 참조하여 mydb에서 café\_order와 drink 테이블을 작성하고 데이터를 저장하시오.
3. 아래의 출력형태로 SQL문을 작성하시오.

	ORDER_NO	DRINK_CODE	DRINK_NAME	ORDER_CNT
1	2023100101	A01	아메리카노	3
2	2023100102	B01	카페라떼	1
3	2023100103	A01	아메리카노	2



# 제3정규형

## ■ 제3정규형

릴레이션의 어떤 속성이 일반 속성에 종속되어 있는 경우를 말한다.

주식별자가 아닌 모든 속성간에는 서로 종속될 수 없다.

엔터테인먼트

주식별자

일련 번호	이름	생년월일	소속사 코드	소속사명
1	임시완	19881201	A101	플럼에이전시
2	박은빈	19920904	B201	나무엑터스
3	문근영	19870506	B201	나무엑터스

일반속성인 소속사명이 다른 일반  
속성인 소속사코드에 종속됨



# 제3정규형

- 제3정규형으로 변환

엔터테이너(일련번호, 이름, 생년월일, 소속사코드, 소속사명)

→ 엔터테이너(일련번호, 이름, 생년월일, 소속사코드),  
소속사(소속사코드, 소속사명)

엔터테이너

일련 번호	이름	생년월일	소속사 코드
1	임시완	19881201	A101
2	박은빈	19920904	B201
3	문근영	19870506	B201

소속사

소속사 코드	소속사명
A101	플럼에이전시
B201	나무엑터스





# ROWNUM

- ROWNUM

- ROWNUM은 SELECT문 결과에 대해서 논리적인 일련번호를 부여한다.  
(순번을 정해놓은 **SUDO COLUMN** 이다. – Oracle DBMS에서만 제공됨)
- ROWNUM은 조회되는 행 수를 제한할 때 많이 사용된다.
- ROWNUM은 화면에 데이터를 출력할 때 부여되는 논리적 순번이다.  
만약 ROWNUM을 사용해서 페이지 단위 출력을 하기 위해서는 인라인 뷰(Inline View)를 사용해야 한다.



# ROWNUM

- ROWNUM

```
CREATE TABLE EX_SCORE (  
    NAME      VARCHAR2 (10) ,  
    SCORE     NUMBER  
);  
  
INSERT INTO EX_SCORE VALUES ('김하나', 94);  
INSERT INTO EX_SCORE VALUES ('이하나', 100);  
INSERT INTO EX_SCORE VALUES ('박하나', 97);  
INSERT INTO EX_SCORE VALUES ('정하나', 87);  
INSERT INTO EX_SCORE VALUES ('최하나', 87);  
INSERT INTO EX_SCORE VALUES ('윤하나', 91);  
INSERT INTO EX_SCORE VALUES ('임하나', 66);  
INSERT INTO EX_SCORE VALUES ('장하나', 80);  
INSERT INTO EX_SCORE VALUES ('고하나', 80);  
INSERT INTO EX SCORE VALUES ('함하나', 95);
```



# ROWNUM

- ROWNUM

```
SELECT ROWNUM, NAME, SCORE
FROM EX_SCORE
WHERE ROWNUM <= 5;
-- WHERE ROWNUM BETWEEN 1 AND 5;
-- WHERE ROWNUM BETWEEN 2 AND 5; (10이 반드시 포함되어야함)
```

ROWNUM	NAME	SCORE
1	이하나	100
2	박하나	97
3	함하나	95
4	김하나	94
5	윤하나	91



# ROWNUM

- ROWNUM

```
-- 성적 순위  
SELECT NAME,  
       SCORE,  
       RANK() OVER(ORDER BY SCORE DESC) RANK,  
       DENSE_RANK() OVER(ORDER BY SCORE DESC) DENSE_RANK  
FROM EX_SCORE;
```

	NAME	SCORE	RANK	DENSE_RANK
1	이하나	100	1	1
2	박하나	97	2	2
3	함하나	95	3	3
4	김하나	94	4	4
5	윤하나	91	5	5
6	정하나	87	6	6
7	최하나	87	6	6
8	장하나	80	8	7
9	고하나	80	8	7
10	임하나	66	10	8



# ROWNUM

- 페이지 처리

```
CREATE TABLE t_board(  
    bno NUMBER(5) PRIMARY KEY,           --글번호  
    title VARCHAR2(200) NOT NULL,        --글제목  
    writer VARCHAR2(20) NOT NULL,        --작성자  
    content VARCHAR2(2000) NOT NULL,      --글내용  
    regdate DATE DEFAULT SYSDATE         --등록일  
);  
  
CREATE SEQUENCE seq NOCACHE; -- 일련번호  
INSERT INTO t_board(bno, title, writer, content)  
VALUES(seq.NEXTVAL, '가임인사', '강남역', '안녕하세요, 가임인사 드려요');  
INSERT INTO t_board(bno, title, writer, content)  
VALUES(seq.NEXTVAL, '공지사항입니다.', '관리자', '가임인사를 남겨주세요');  
INSERT INTO t_board(bno, title, writer, content)  
VALUES(seq.NEXTVAL, '가임인사입니다.', '이강', '안녕하세요~');  
INSERT INTO t_board(bno, title, writer, content)  
VALUES(seq.NEXTVAL, '좋은 하루', '긴하루', '좋은 하루 되세요');
```



# ROWNUM

- 페이지 처리 - 재귀복사

```
-- 재귀 복사 : INSERT INTO 테이블이름 (SELECT 절)
INSERT INTO t_board(bno, title, writer, content)
(SELECT seq.nextval, title, writer, content FROM t_board);

-- ROWNUM 사용
SELECT ROWNUM, bno, title, content
FROM t_board
WHERE ROWNUM > 0 AND ROWNUM <= 10;
--WHERE ROWNUM > 11 AND ROWNUM <= 10; -- ROWNUM은 1을 포함해야함
```

ROWNUM	BNO	TITLE	CONTENT
1	1	가입인사	안녕하세요, 가입인사 드려요
2	2	공지사항입니다.	가입인사를 남겨주세요
3	3	가입인사입니다.	안녕하세요~
4	4	좋은 하루	좋은 하루 되세요
5	5	가입인사	안녕하세요, 가입인사 드려요
6	6	공지사항입니다.	가입인사를 남겨주세요
7	7	가입인사입니다.	안녕하세요~
8	8	좋은 하루	좋은 하루 되세요
9	9	가입인사	안녕하세요, 가입인사 드려요
10	10	공지사항입니다.	가입인사를 남겨주세요

# ROWNUM

- 페이지 처리 – 인라인 뷰 사용

```
-- 인라인 뷰(서브 쿼리) : 10개씩 출력(페이지 처리)
SELECT *
FROM
  (SELECT ROWNUM rn, bno, title, content
   FROM t_board)
--WHERE ROWNUM > 11 AND ROWNUM <= 10;
WHERE rn >= 11 AND rn <= 20; -- 별칭을 사용해야 검색됨
```

	RN	BNO	TITLE	CONTENT
1	11	11	가입인사입니다.	안녕하세요~
2	12	12	좋은 하루	좋은 하루 되세요
3	13	13	가입인사	안녕하세요, 가입인사 드려요
4	14	14	공지사항입니다.	가입인사들 남겨주세요
5	15	15	가입인사입니다.	안녕하세요~
6	16	16	좋은 하루	좋은 하루 되세요
7	17	17	가입인사	안녕하세요, 가입인사 드려요
8	18	18	공지사항입니다.	가입인사들 남겨주세요
9	19	19	가입인사입니다.	안녕하세요~
10	20	20	좋은 하루	좋은 하루 되세요

# ROWNUM

- ROWID

```
-- ROWID - 데이터를 구분할 수 있는 유일한 값,  
-- 데이터 파일의 저장 블록을 확인할 수 있음  
SELECT ROWID, bno, title  
FROM t_board;  
  
SELECT bno, title, content  
FROM t_board  
WHERE ROWID = 'AAAVQiAAHAAA AVEAAB';
```

ROWID	BNO	TITLE
1 AAAVQiAAHAAA AVEAAA	1	가입인사
2 AAAVQiAAHAAA AVEAAB	2	공지사항입니다.
3 AAAVQiAAHAAA AVEAAC	3	가입인사입니다.
4 AAAVQiAAHAAA AVEAAD	4	좋은 하루
5 AAAVQiAAHAAA AVEAAE	5	가입인사
6 AAAVQiAAHAAA AVEAAF	6	공지사항입니다.
7 AAAVQiAAHAAA AVEAAG	7	가입인사입니다.





## 옵티마이저(Optimizer)와 실행 계획

- 옵티마이저는 SQL의 실행 계획을 수립하고, SQL을 실행하는 데이터베이스 관리 시스템의 소프트웨어이다.
- 동일한 결과가 나오는 SQL도 어떻게 실행하느냐에 따라 성능이 달라진다.
- 데이터 디렉터리에는 오브젝트 통계, 시스템 통계등의 정보를 사용해서 예상되는 비용을 산정한다.
- 옵티마이저는 여러 개의 실행 계획 중에서 최저비용을 가지고 있는 계획을 선택해서 SQL을 실행한다.



# SQL 최적화

## 옵티마이저(Optimizer)와 실행 계획

Oracle 명령어

DESC PLAN\_TABLE;

이름	널? 유형
-----	
STATEMENT_ID	VARCHAR2 (30)
PLAN_ID	NUMBER
TIMESTAMP	DATE
REMARKS	VARCHAR2 (4000)
OPERATION	VARCHAR2 (30)
OPTIONS	VARCHAR2 (255)
OBJECT_NODE	VARCHAR2 (128)
OBJECT_OWNER	VARCHAR2 (128)
OBJECT_NAME	VARCHAR2 (128)
OBJECT_ALIAS	VARCHAR2 (261)
OBJECT_INSTANCE	NUMBER (38)
OBJECT_TYPE	VARCHAR2 (30)
OPTIMIZER	VARCHAR2 (255)
SEARCH_COLUMNS	NUMBER



# SQL 최적화

```
SELECT * FROM employee;
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	3
TABLE ACCESS	EMPLOYEE	FULL	5	3

Other XML



# SQL 최적화

```
-- 급여가 최고인 사원과 최저인 사원을 검색하시오
SELECT MAX(salary) 최고급여, MIN(salary) 최저급여
FROM employee;

SELECT empname, salary
FROM employee
WHERE salary = (SELECT MAX(salary) FROM employee)
OR salary = (SELECT MIN(salary) FROM employee);
```

	EMPNAME	SALARY
1	한성년	4500000
2	신세계	1000000

SELECT STATEMENT				2	3
TABLE ACCESS	EMPLOYEE	FULL		2	3
Filter Predicates					
OR					
SALARY= (SELECT MAX(SALARY) FROM EMPLOYEE EMPLOYEE)					
SALARY= (SELECT MIN(SALARY) FROM EMPLOYEE EMPLOYEE)					
SORT		AGGREGATE		1	
TABLE ACCESS	EMPLOYEE	FULL		5	3
SORT		AGGREGATE		1	
TABLE ACCESS	EMPLOYEE	FULL		5	3
Other XML					

# SQL 최적화

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(null, null, 'ALLSTATS LAST'));
```

Id	Operation	Name	E-Rows	OMem	lMem	Used-Mem
0	SELECT STATEMENT					
1	SORT ORDER BY		1	1024	1024	
* 2	COUNT STOPKEY					
* 3	FILTER					
4	NESTED LOOPS		1			
5	NESTED LOOPS		1			
* 6	HASH JOIN		1	970K	970K	193K (0)
* 7	TABLE ACCESS FULL	ARGUMENT\$	4574			
* 8	TABLE ACCESS FULL	OBJ\$	4560			
* 9	INDEX RANGE SCAN	I_USER2	1			
10	TABLE ACCESS CLUSTER	USER\$	1			
* 11	INDEX UNIQUE SCAN	I_USER#	1			
* 12	FIXED TABLE FULL	X\$KZSPR	2			
13	NESTED LOOPS SEMI		1			



# SQL 최적화

```
SELECT b.empname,  
       b.salary  
FROM (  
    SELECT a.empname,  
           a.salary,  
           ROW_NUMBER() OVER(ORDER BY salary) minsal,  
           ROW_NUMBER() OVER(ORDER BY salary DESC) maxsal  
    FROM employee a  
    ) b  
WHERE b.minsal = 1 OR b.maxsal = 1;
```

EMPNAME	SALARY
1 한성년	4500000
2 신세계	1000000

# SQL 최적화

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				5
VIEW				5
Filter Predicates				
OR				
B.MINSAL=1				
B.MAXSAL=1				
WINDOW		SORT		5
WINDOW		SORT		5
TABLE ACCESS	EMPLOYEE	FULL		3
Other XML				
{info}				

Plan hash value: 3713220770

Id	Operation	Name	E-Rows
0	SELECT STATEMENT		
1	COLLECTION ITERATOR PICKLER FETCH	DISPLAY_CURSOR	8168

## 인덱스(INDEX)

- 인덱스는 데이터를 빠르게 검색할 수 있는 방법을 제공함
- 인덱스는 **인덱스 키(예 bookid)로 정렬**되어 있기 때문에 원하는 데이터를 빠르게 조회한다.
- 인덱스는 오름차순 및 내림차순으로 탐색이 가능하다
- 하나의 테이블에 여러 개의 인덱스를 생성할 수 있고 하나의 인덱스는 여러 개의 칼럼으로 구성될 수 있다.
- **오라클은 힌트**를 사용하면 확실하게 인덱스를 실행 시킬수 있다.





# SQL 최적화

- INDEX 생성 : **CREATE INDEX** 인덱스이름 **ON** 테이블명(칼럼명)

- ✓ 힌트 사용법

`/*+ INDEX(테이블명 인덱스명) */`

```
-- 작성자가 관리자인 게시물 실행
SELECT * FROM board WHERE writer = '관리자';

-- INDEX 생성 : CREATE INDEX 인덱스명 ON 테이블명 (칼럼명)
--CREATE INDEX idx_admin ON board(writer);

-- 힌트 사용해서 인덱스 실행 --
SELECT /*+ INDEX(board idx_admin) */ * FROM board WHERE writer = '관리자';
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				6 3
TABLE ACCESS	BOARD	BY INDEX ROWID BATCHED		6 3
INDEX	IDX_ADMIN	RANGE SCAN		6 1
Access Predicates				
WRITER='관리자'				



# SQL 최적화

## INDEX 삭제 : DROP INDEX 인덱스이름

```
-- INDEX 삭제  
DROP INDEX idx_admin;
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				6 3
TABLE ACCESS	<a href="#">BOARD</a>	FULL		6 3
Filter Predicates				
WRITER='관리자'				

