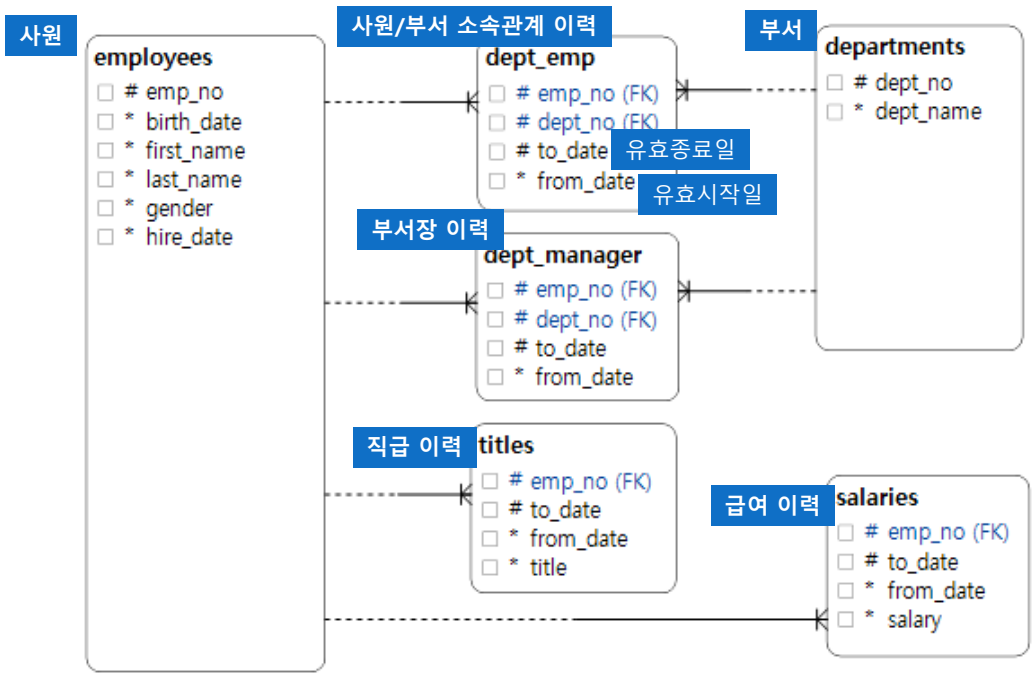
Data Analytics 과정 특강

고급 SQL 이해와 활용

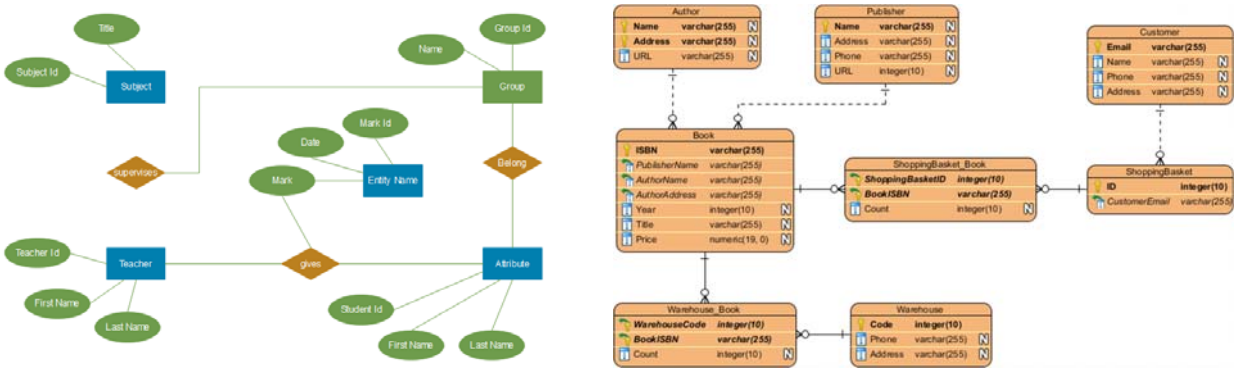
실습 1. 예제 데이터 모델 (ERD)

1-1. 예제 데이터 ERD



개념 이해 1. ERD(Entity Relationship Diagram)

- 엔터티(entity)와 관계(relationship)을 표현한 다이어그램
  - 구성요소 : 엔터티(Entity), 속성(Attribute), 제약조건(constraints), 관계(Relationship)
  - 관계 표현의 관점 : 대응수(mapping cardinality), 필수/선택(mandatory/optional), 절대종속/상대종속(strong/weak)
- 다양한 ERD 표기법 존재



....

실습 2. 예제 테이블 생성

2-1. Create table 예제

```
CREATE TABLE dept_manager (
   emp_no       NUMBER(6),
   dept_no      VARCHAR2(4)         NOT NULL,
   to_date      VARCHAR2(8)         NOT NULL,
   from_date    VARCHAR2(8)         NOT NULL,
                                                  기본키
        외래키

        외래키
);
```

개념 이해 2. 제약조건(Integrity constraints)

- 제약조건 : DB 무결성(integrity) 보장을 위한 장치
- 제약조건의 종류
  - Not Null
    - ✓ NULL을 허용하지 않음
    - ✓ Eg) name varchar2(10) NOT NULL
  - UNIQUE
    - ✓ 컬럼값의 유일성(uniqueness) 보장
    - ✓ Eg) bno number UNIQUE
  - Primary Key (기본키)
    - ✓ 해당 테이블의 대표 속성으로서 not null과 uniqueness 보장
    - ✓ Eg) bno number primary key
  - Foreign Key (외래키)
    - ✓ 참조 무결성(referential constraint) 보장을 위한 것으로 참조하는 테이블의 기본키 값만 가질 수 있음
    - ✓ Eg) constraints fk_cateId foreign key references cateTable(cateId)
  - Check
    - ✓ 입력될 수 있는 데이터의 범위 제한
    - ✓ Eg) bno number primary key CHECK (bno between 1 and 1000)

개념 이해 2. 제약조건(Integrity constraints)

- 참조무결성 제약조건을 보장하기 위한 4가지 옵션
  - 참조관계(부모-자식 관계)의 테이블에서 부모 테이블의 투플 삭제 시, 참조무결성 위배 가능성
  - 부모 테이블의 투플 삭제 시, 이를 참조하는 자식테이블 투플의 처리방법 4가지

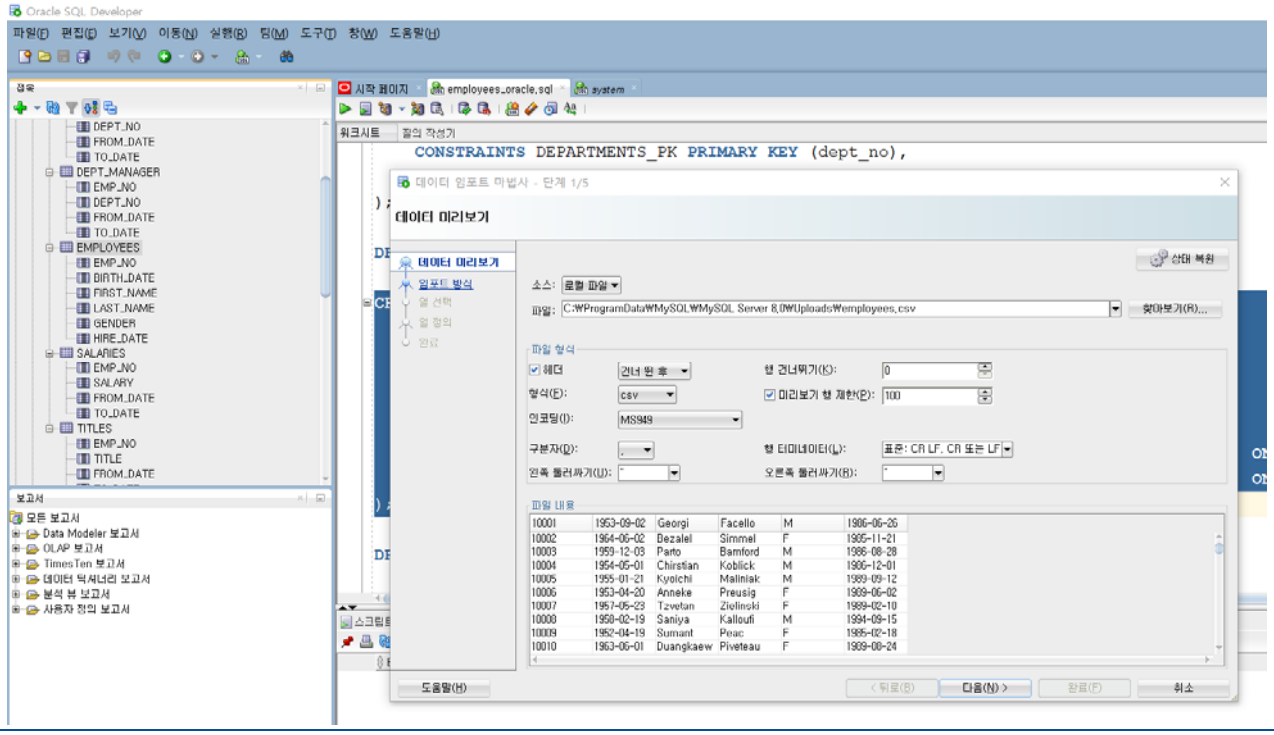
옵션	설명
RESTRICTED	자식 테이블에서 참조하고 있는 부모 테이블 투플 삭제(수정) 거부
CASCADE	부모 테이블의 투플 삭제(수정) 시, 이를 참조하는 자식 테이블 투플도 같이 삭제(수정)
DEFAULT	부모 테이블의 투플 삭제(수정) 시, 이를 참조하는 자식 테이블 투플의 속성값을 미리 지정한 기본값(default value)로 변경
NULL	부모 테이블의 투플 삭제 시(수정), 이를 참조하는 자식 테이블 투플의 속성값을 NULL로 변경(단, NULL값이 허용될 경우)

Oracle examples :

```
CONSTRAINTS dept_manager_fk1 FOREIGN KEY (emp_no) REFERENCES employees(emp_no) ON DELETE NO ACTION
CONSTRAINTS dept_manager_fk1 FOREIGN KEY (emp_no) REFERENCES employees(emp_no) ON DELETE CASCADE
CONSTRAINTS dept_manager_fk1 FOREIGN KEY (emp_no) REFERENCES employees(emp_no) ON DELETE SET DEFAULT '10000'
CONSTRAINTS dept_manager_fk1 FOREIGN KEY (emp_no) REFERENCES employees(emp_no) ON DELETE SET NULL
```

실습 3. 데이터 적재(Data Importing)

3-1. SQL Developer 상에서의 table import 예제 화면



실습 4. 데이터 정제(Data Cleansing)

4-1. 동시에 2개 부서에 소속된 사원을 찾아서 to_date(유효종료일)이 가장 큰 것만 취하고 나머지는 삭제

4-1-① 동시에 두개 부서에 소속된 사원 현황

4-1-② 중복 이력 중 하나만 선택하고 나머지 삭제
(cleansing rule : to_date가 가장 큰 것 선택)

subquery

subquery : 삭제 할 투플의 (emp_no, from_date)를 구하는 SQL

실습 4. 데이터 정제(Data Cleansing)

4-1. 동시에 2개 부서에 소속된 사원을 찾아서 to_date(유효종료일)이 가장 큰 것만 취하고 나머지는 삭제

subquery : 삭제 할 투플의 (emp_no, from_date)를 구하는 SQL

4-1-③ data cleansing 후 PK 제약조건 add

실습 4. 데이터 정제(Data Cleansing)

4-2. 이력날짜 8자리로 맞추기 (예, '1992-05-10' → '19920510')

4-2-① Rollback segment 확장

rollback segments에 관련된 tablespace와 data file 및 사이즈 조회 (system 사용자로 로그인 후 조회)

```
select file_name, tablespace_name, bytes from dba_data_files ;
```

rollback segments에 관련된 data file size-up

```
ALTER DATABASE DATAFILE 'C:₩ORACLEXE₩APP₩ORACLE₩ORADATA₩XE₩UNDOTBS1.DBF' RESIZE 100M ;
```

4-2-② Column re-size

실습 4. 데이터 정제(Data Cleansing)

4-3. 현재 상태의 종료일자(to_date) : '99990101 -> 99991231'로 변경

4-4. 이력 관리 : 양편 넣기 -> 한편 넣기

① cleansing rule : to_date 컬럼값이 '99991231'이 아닌 투플을 대상으로 to_date를 하루 앞 당기게 수정

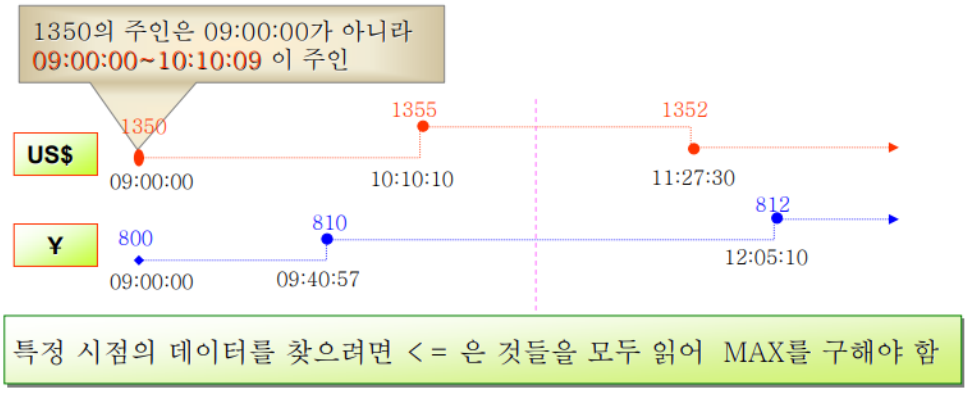
② cleansing rule : 1)번 cleansing으로 인해 'to_date < from_date'인 튜플에 대해서 to_date를 from_date로 update

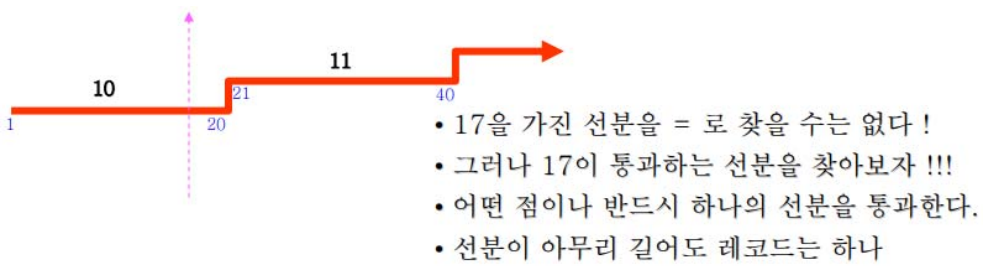
③ data cleansing 후 PK/FK 제약조건 추가

```
ALTER TABLE titles ADD CONSTRAINTS TITLES_PK PRIMARY KEY (emp_no, to_date) ;
ALTER TABLE titles ADD CONSTRAINTS TITLES_FK1 FOREIGN KEY (emp_no) REFERENCES employees (emp_no)
ON DELETE CASCADE ;
ALTER TABLE salaries ADD CONSTRAINTS SALARIES_PK PRIMARY KEY (emp_no, to_date) ;
ALTER TABLE salaries ADD CONSTRAINTS SALARIES_FK1 FOREIGN KEY (emp_no) REFERENCES employees
(emp_no) ON DELETE CASCADE ;
```


개념 이해 3. 이력 데이터 관리 기법 (1)

- 이력 관리 : 점(Event) vs. 선분





* 출처 : www.en-core.com

개념 이해 3. 이력 데이터 관리 기법 (2)

- 점(Event) 이력 모델

부서변경이력

사원번호	부서코드	변경일자	...
7788	10	1999/ 02/ 04	...
7788	30	2000/ 07/ 21	NULL
7788	20	2002/ 05/ 15	...
8123	40	2000/ 03/ 23	
8123	30	2001/ 11/ 05	
8123	10	2002/ 09/ 17	
...	...	...	

사원별, 부서개편일 전의 부서코드를 찾아라. (부서개편일 : 2002 년 3 월 1 일)

1 Subquery를 이용한 방법 (1)

```
SELECT 사원번호, 부서코드
FROM   부서변경이력
WHERE  변경일자 = (select MAX(변경일자) from 부서변경이력 where 변경일자 < '2002/03/01');
```

2 Subquery를 이용한 방법 (2)

```
SELECT 사원번호, 부서코드
FROM   부서변경이력
WHERE  ROWID = ( select /*+ index_desc(부서변경이력 idx_변경일자) */ ROWID  RID
                 from 부서변경이력 where 변경일자 < '2002/03/01 and rownum <= 1);
```

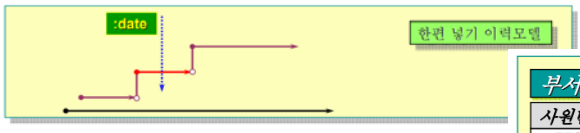
* 출처 : www.en-core.com

개념 이해 3. 이력 데이터 관리 기법 (3)

- 선분 이력 모델 (한편넣기)

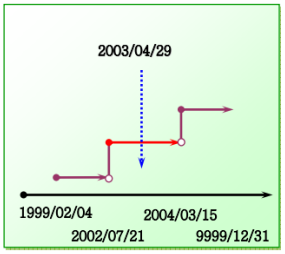
■ 정의
- '일자(Day)'로 이력관리가 되고 있는 엔터티에서, 동일 일자에 대해 이벤트가 최대 한번만 발생하는 경우에 사용되는 기법
- 시간(time)까지 관리하는 이력관리에서 적용

■ 조회 방법
- where :date >= 시작일 and :date <= 종료일
- where :date between 시작일 and 종료일



부서변경이력

사원번호	부서코드	유효시작일	유효종료일	~~~
7788	10	1999/ 02/ 04	2002/ 07/ 20	~~~
7788	30	2002/ 07/ 21	2004/ 05/ 14	NULL
7788	20	2004/ 05/ 15	9999/ 12/ 31	~~~
8123	40	2000/ 03/ 23	2001/ 11/ 04	
8123	30	2001/ 11/ 05	2002/ 09/ 16	
8123	10	2002/ 09/ 17	9999/ 12/ 31	
~~	~~~	~~~	~~~	



사번 '7788' 인 사람이, 2001 년 4 월 29 일에 근무했던 부서는 어디인가?

```
SELECT 사원번호, 부서코드
FROM   부서변경이력
WHERE  유효시작일 <= '2003/04/29'
AND    유효종료일 >= '2003/04/29'
AND    사원번호 = '7788'
```

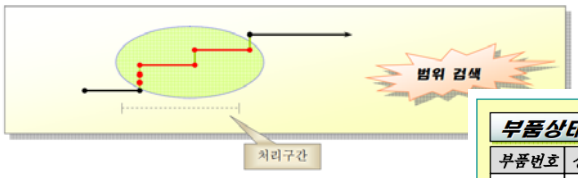
```
SELECT 사원번호, 부서코드
FROM 부서변경이력
WHERE '2003/04/29' BETWEEN 유효시작일
                       AND 유효종료일
AND  사원번호 = '7788'
```

* 출처 : www.en-core.com

개념 이해 3. 이력 데이터 관리 기법 (4)

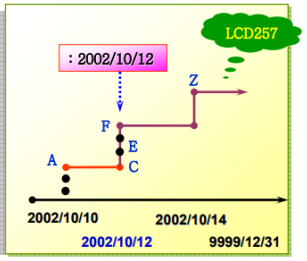
선분 이력 모델 (양편넣기)

- 용도
  - 이력에 이벤트까지 관리하고자 하는 경우 사용
  - 보험업무처럼 부문에 따라 이력을 보는 기준이 다른 경우
- 조회방법 : 원하는 정보의 형태에 따라 Query 문이 달라짐
  - ① where 시작일 between :구간1 and :구간2



부품상태변경이력

부품번호	상태코드	유효시작일	유효종료일	발생순번
CDR788	E	2002/10/12	9999/12/31	5
LCD257	A	2002/10/10	2002/10/12	1
LCD257	C	2002/10/12	2002/10/12	1
LCD257	E	2002/10/12	2002/10/12	2
LCD257	F	2002/10/12	2002/10/14	3
LCD257	Z	2002/10/14	9999/12/31	1
~~	~~~	~~~	~~~	~~~



1.코드번호 'LCD257' 인 부품의 당일 초기상태는 ? (2002/10/12 기준)

```
SELECT 부품번호, 상태코드, 발생순번 FROM 부품상태변경이력
WHERE 유효시작일  < '2002/10/12'
   AND 유효종료일 => '2002/10/12'
   AND 부품번호 = 'LCD257'
```

* 출처 : www.en-core.com

실습 5. 이력 조회

5-1. 부서별 현재 부서장의 사번(emp_no), 이름(first_name+last_name), 성별(gender), 입사일(hire_date), 직급(title), 급여(salary) 조회

5-2. 'Lein Bale' 사원의 2000년 1월 1일 당시 소속부서, 직급, 급여 출력

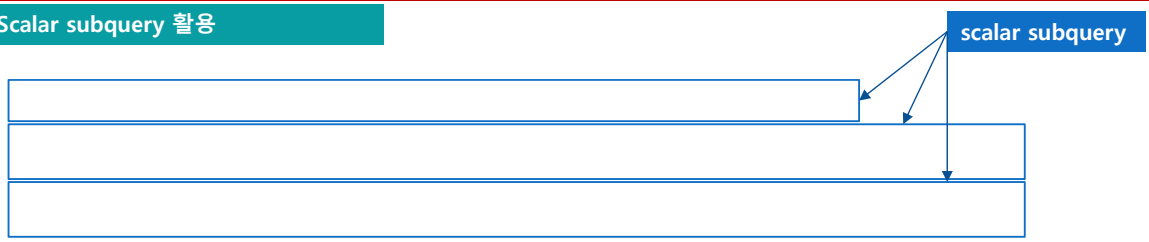
5-2-① 조인 이용

과거 시점 조회는 BETWEEN 연산으로

실습 5. 이력 조회

5-2. 'Lein Bale' 사원의 2000년 1월 1일 당시 소속부서, 직급, 급여 출력

5-2-② Scalar subquery 활용



5-2-③ 5-2-② SQL의 개선 : 어떤 면에서의 개선일까?

개념 이해 4. 서브쿼리(Sub-query)

- **서브쿼리(sub-query)**
  - SQL 문장의 하부 절에 사용하는 select 쿼리문
  - 서브쿼리 위치에 따른 명칭
    - ✓ SELECT문에 있는 서브쿼리 : **스칼라 서브쿼리(scalar subquery)**
    - ✓ FROM절에 있는 서브쿼리 : **인라인 뷰(inline view)**
    - ✓ WHERE절에 있는 서브쿼리 : **서브쿼리(subquery)**
  - 서브쿼리의 반환값에 따른 서브쿼리 종류
    - ✓ 단일 행 서브쿼리(Single-Row Subquery) : 서브쿼리의 결과가 1행
    - ✓ 다중 행 서브쿼리(Multiple-Row Subquery) : 서브쿼리의 결과가 여러 행
    - ✓ 다중 컬럼 서브쿼리(Multi-Column Subquery) : 서브쿼리의 결과가 여러 컬럼
  - Scalar subquery의 경우는 단일 행 서브쿼리만 허용 : 반드시 오직 하나의 행만 반환함을 보장해야 오류가 없음
  - 상호연관 서브쿼리(Correlated Subquery) : 메인쿼리의 값을 서브쿼리가 사용하고, 서브쿼리의 값을 받아서 메인쿼리가 계산하는 구조의 쿼리

실습 6. 인덱스(index) / 조인(join) / 실행계획(explain plan) / 힌트(hint)

6-1. 'Lein Bale' 사원의 사번(emp_no), 생년월일(birth_date), 성별(gender), 고용일(hire_date), 소속부서명(dept_name), 직급(title), 급여(salary) 조회

6-1-① 무엇이 잘못되었을까?

```
select *
from employees
where first_name||' '||last_name = 'Lein Bale' ;
```

6-1-② 인덱스 생성(last_name)

6-1-③ ①번 SQL 튜닝 (실행계획과 수행시간 비교)

실습 6. 인덱스(index) / 조인(join) / 실행계획(explain plan) / 힌트(hint)

6-2. 1999년도에 입사한 사원의 사번(emp_no), 생년월일(birth_date), 성별(gender), 고용일(hire_date), 소속부서명(dept_name), 직급(title), 급여(salary) 조회

6-2-① 답은 맞는거 같은데... 튜닝포인트는 없을까?

```
select a.emp_no
      , a.birth_date, a.gender, a.hire_date
      , c.dept_name, d.title, e.salary
from employees a, dept_emp b, departments c, titles d, salaries e
where to_char(a.hire_date,'yyyymmdd') between '19990101' and '19991231'
  and a.emp_no = b.emp_no and b.to_date = '99991231'
  and b.dept_no = c.dept_no
  and a.emp_no = d.emp_no and d.to_date = '99991231'
  and a.emp_no = e.emp_no and e.to_date = '99991231' ;
```

6-2-② 인덱스 생성(hire_date)

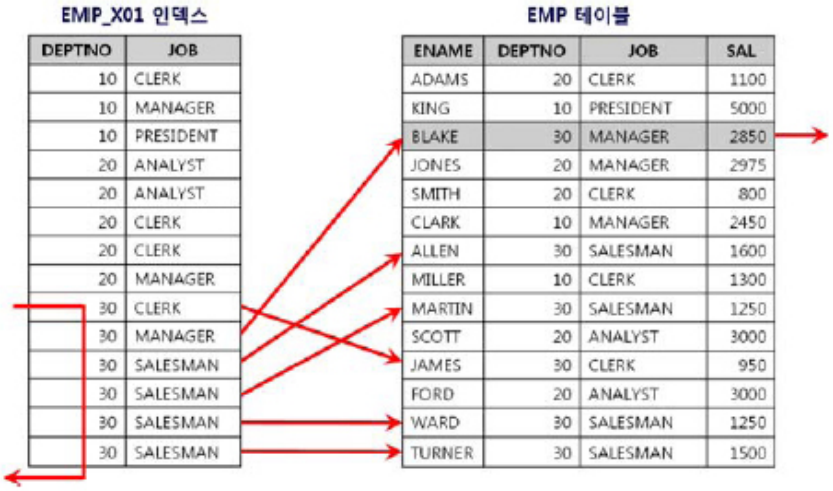
실습 6. 인덱스(index) / 조인(join) / 실행계획(explain plan) / 힌트(hint)

6-2. 1999년도에 입사한 사원의 사번(emp_no), 생년월일(birth_date), 성별(gender), 고용일(hire_date), 소속부서명(dept_name), 직급(title), 급여(salary) 조회

6-2-③ ①번 SQL 튜닝

개념 이해 5. 인덱스(Index)

- 목적 : 검색 속도 향상
- 구성 : 인덱스 컬럼(1개 이상) + ROWID (row의 위치정보)
- 특징 : 인덱스 컬럼 기준으로 정렬되어 있고, 물리적인 데이터를 가지는 데이터베이스 객체
- 종류 : B-tree 인덱스, Bitmap 인덱스, FBI(Function Based Index), Clustered Index 등
- 단점 : Insert, Update, Delete 시에 Index도 갱신해야 하므로 부하 가중

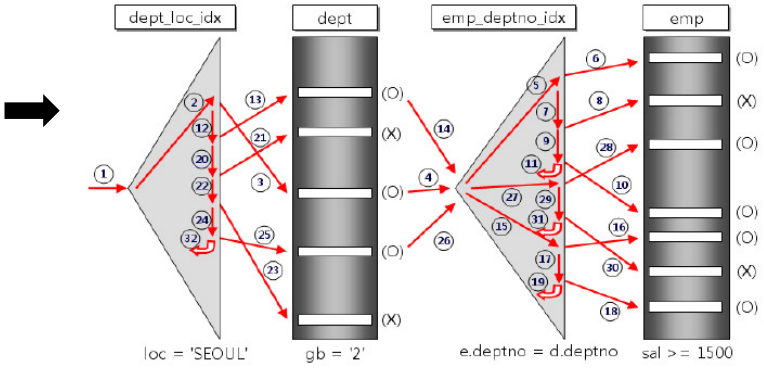


개념 이해 6. 조인(Join) : How-to-do 관점

Nested loop join

- 2개 이상의 테이블에서 하나의 집합을 기준으로 순차적으로 상대방 Row를 결합하여 원하는 결과를 조합하는 방식
- 먼저 선행 테이블의 처리 범위를 하나씩 액세스하면서 추출된 값으로 연결할 테이블을 조인한다
- 특징
  - ✓ 좁은 범위에 유리한 성능을 보여줌
  - ✓ 순차적으로 처리하며, Random Access 위주
  - ✓ 후행 테이블(Driven)에는 조인을 위한 인덱스 생성 필요

```
select *
from dept a, emp b
where a.loc = 'Seoul'
  and a.gb = '2'
  and a.dept_no = b.dept_no
  and b.sal >= 1500 ;
```

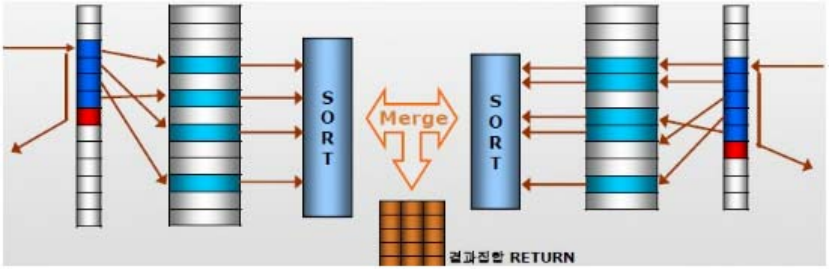


* 출처: https://needjarvis.tistory.com/162 [자비스가 필요해]

개념 이해 6. 조인(Join) : How-to-do 관점

Sort-Merge join

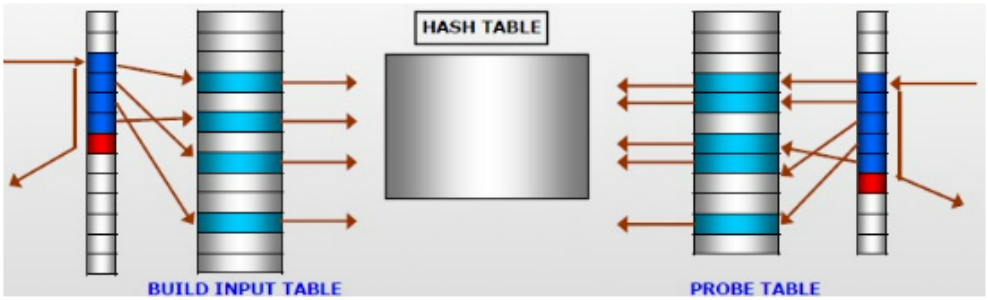
- 조인의 대상범위가 넓을 경우 발생하는 Random Access를 줄이기 위한 경우나 연결고리에 마땅한 인덱스가 존재하지 않을 경우 해결하기 위한 조인 방안
- 양쪽 테이블의 처리범위를 각자 Access하여 정렬한 결과를 차례로 Scan하면서 연결고리의 조건으로 Merge하는 방식
- 특징
  - ✓ 연결을 위해 랜덤 액세스를 하지 않고 스캔을 하면서 수행
  - ✓ Nested Loop Join처럼 선행집합 개념이 없음
  - ✓ 정렬을 위한 영역(Sort Area Size)에 따라 효율에 큰 차이 발생
  - ✓ 조인 연산자가 '='이 아닌 경우 nested loop 조인보다 유리한 경우가 많음



* 출처: https://needjarvis.tistory.com/162 [자비스가 필요해]

개념 이해 6. 조인(Join) : How-to-do 관점

- **Hash join**
  - 해싱 함수(Hashing Function) 기법을 활용하여 조인을 수행하는 방식(해싱 함수는 직접적인 연결을 담당하는 것이 아니라 연결될 대상을 특정 지역(partition)에 모아두는 역할만을 담당
  - 해시값을 이용하여 테이블을 조인하는 방식
  - Sort-Merge 조인은 소트의 부하가 많이 발생하여, Sort 대신 해쉬값을 이용하는 조인
  - 특징
    - ✓ 대용량 처리의 선결조건인 랜덤 액세스와 정렬에 대한 부담을 해결할 수 있는 대안
    - ✓ parallel processing을 이용한 hash 조인은 대용량 데이터를 처리하기 위한 최적의 솔루션
    - ✓ 2개의 조인 테이블 중 small rowset을 가지고 hash table 생성



* 출처: https://needjarvis.tistory.com/162 [자비스가 필요해]

개념 이해 7. 실행계획(Explain plan)

- **Explain plan (또는 Execution plan)**
  - 옵티마이져가 SQL문을 실행하기 위해 수행한 일련의 동작(OPERATIONS)을 트리형식으로 표현한 것
  - 사용자로 하여금 어떻게 SQL을 수행할 것인지를 보여주고 SQL 성능 판단을 위한 정보 제공
- **실행계획이 포함하는 정보**
  - 쿼리문이 참조한 테이블들의 순서
  - 쿼리문이 참조한 테이블들의 접근방법(ACESS PATH)
  - 조인에 의해 영향받는 테이블들의 조인방법(JOIN MEHTOD)
  - 데이터 조작방법(filter, sort, or aggregation, etc)
- **실행계획 상에 나타나는 성능에 악영향을 주는 요소**
  - Full scans : 의도하지 않은 Full scans
  - Unselective range scans : 100건을 조회하기 위해 백만건을 스캔하는 경우
  - Late predicate filters : 미리 처리범위를 좁히지 못하는 경우
  - Wrong join order : 잘못된 조인순서는 처리범위를 증가시킨다.
  - Late filter operations : 필터로 버릴 것이 있다면 조인 전에 필터하는 것이 좋다.

개념 이해 7. 실행계획(Explain plan)

실행계획 예

```
select *
from employees
where first_name = 'Lein'
  and last_name = 'Bale' ;
```

➡

인덱스 EMPLOYEES_IDX1을 통하여 LAST_NAME='Bale'조건으로 필터링된 row 각각에 대하여 EMPLOYEES 테이블에서 FIRST_NAME='Lein'조건을 체크하여 만족하는 row 출력

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	178
TABLE ACCESS	EMPLOYEES	BY INDEX ROWID	1	178
Filter Predicates				
FIRST_NAME='Lein'				
INDEX	EMPLOYEES_IDX1	RANGE SCAN	183	3
Access Predicates				
LAST_NAME='Bale'				

```
Other XML
  {info}
    info type="db_version"
      11.2.0.2
    info type="parse_schema"
    "DA"
    info type="plan_hash"
      1147874131
    info type="plan_hash_2"
      3431891562
    {hint}
      INDEX_RS_ASC(@"SEL$1" "A"@"SEL$1" ("EMPLOYEES","LAST_NAME"))
      OUTLINE_LEAF(@"SEL$1")
      ALL_ROWS
      DB_VERSION('11.2.0.2')
      OPTIMIZER_FEATURES_ENABLE('11.2.0.2')
      IGNORE_OPTIM_EMBEDDED_HINTS
```


개념 이해 8. 힌트(Hint)

Oracle hint

- SQL에 포함되어 쓰여져 Optimizer의 실행 계획을 원하는 대로 유도하는 일종의 지시문
- 오라클 Optiomizer라고 해서 항상 최선의 실행 계획을 수립할 수는 없으므로 테이블이나 인덱스의 잘못된 실행 계획을 개발자가 직접 바꿀 수 있도록 도와주는 줌
- 힌트를 사용하게 되면 액세스 경로, 조인의 순서, Optiomizer의 목표(Goal)를 변경 가능하며 SQL 문장 내에 "/*+ 힌트내용 */" 형태로 사용되어짐 (*와 +는 띄어쓰기 없이 표현되어야 함에 주의)

Hint의 종류

Optimization Goals and Approaches	Acess Method Hints	Join Order Hints	Join Operation Hints	Parallel Execution Hints	Query Transformation Hints	Other Hints
• ALL_ROWS 혹은 FIRST_ROWS • CHOOSE • RULE	• AND_EQUAL • CLUSTER • FULL • HASH • INDEX 혹은 NO_INDEX • NDEX_ASC 혹은 INDEX_DESC • INDEX_COMBINE • INDEX_FFS • ROWID	• ORDERED • STAR	• DRIVING_SITE • HASH_AJ, MERGE_AJ 혹은 NL_AJ • LEADING • USE_HASH 혹은 USE_MERGE • USE_NL	• PARALLEL 혹은 NOPARALLEL • PARALLEL_INDEX • PQ_DISTRIBUTE • NOPARALLEL_INDEX	• EXPAND_GSET_TO_UNION • FACT 혹은 NOFACT • MERGE • NO_EXPAND • NO_MERGE • REWIRTE 혹은 NOREWRITE • STAR_TRANSFORMATION • USE_CONCAT	• APPEND 혹은 NOAPPEND • CACHE 혹은 NOCACHE • CURSOR_SHARED_EXACT • DYNAMIC_SAMPLING • NESTED_TABLE_GET_REFS • UNNEST 혹은 NO_UNNEST • ORDERED_PREDICATES

* 출처: https://luckys.tistory.com/342 [Lucky's...]

실습 7. 재귀적 관계(Recursive Relationship)

7-1. sample table 생성

```
CREATE TABLE EMP
       (EMPNO NUMBER(4) CONSTRAINT EMP_PK PRIMARY KEY,
        ENAME VARCHAR2(10),
       JOB VARCHAR2(9),
       MGR NUMBER(4),
       HIREDATE DATE,
       SAL NUMBER(7,2),
       COMM NUMBER(7,2),
       DEPTNO NUMBER(2));
```

7-2. sample data 생성

```
INSERT INTO EMP VALUES
(7369,'SMITH','CLERK',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);
INSERT INTO EMP VALUES
(7499,'ALLEN','SALESMAN',7698,to_date('20-2-1981','dd-mm-yyyy'),1600,300,30);
…
```


실습 7. 재귀적 관계(Recursive Relationship)

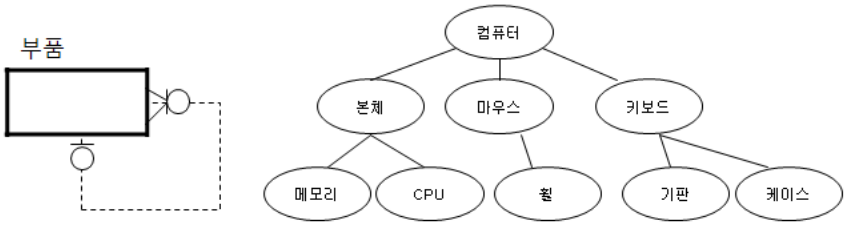
7-3. Emp 테이블에서 사원 간 상하관계 출력

출력 결과

```
조직도
KING(PRESIDENT)
  JONES(MANAGER)
    FORD(ANALYST)
      SMITH(CLERK)
  BLAKE(MANAGER)
    ALLEN(SALESMAN)
    WARD(SALESMAN)
    MARTIN(SALESMAN)
    TURNER(SALESMAN)
    JAMES(CLERK)
  CLARK(MANAGER)
    MILLER(CLERK)
```

개념 이해 9. 재귀적 관계(Recursive relationship)

- **재귀적 관계 (또는 순환관계)**
  - 하나의 엔티티 내에서 엔티티와 엔티티가 관계를 맺고 있는 형태
  - 부서, 부품, 메뉴 등과 같이 계층 구조 형태를 표현할 때 유용



부품은 다른 부품으로 조립된다.

- **START WITH ... CONNECT BY PRIOR ...**
  - START WITH 절 : 계층 구조의 데이터를 읽어나가는데 있어 시작점을 지정
  - CONNECT BY 절은 다음에 읽을 자식 데이터를 지정
    - ✓ 'PRIOR 자식컬럼 = 부모컬럼' 형태 : 부모 -> 자식 방향으로 내려가는 순방향 조회 시 사용
    - ✓ 'PRIOR 부모컬럼 = 자식컬럼' 형태 : 자식 -> 부모 방향으로 올라가는 역방향 조회 시 사용
  - NOCYCLE : 사이클이 발생한 이후의 데이터는 읽지 않습니다.
  - ORDER SIBLINGS BY 절 : 동일한 LEVEL의 데이터 사이에서 정렬을 합니다.

실습 8. (집계함수+문자열함수) 활용 / 그룹핑컬럼 가공

8-1. 부서별 급여를 가장 많이 받는 사원의 이름(first_name+last_name)과 급여(salary)

출력 결과

	DEPT_NAME	EMP_NAME	SALARY
1	Research	Ramachenga Soicher	130211
2	Development	Khosrow Sgarro	144434
3	Quality Management	Shin Luck	132103
4	Human Resources	Yinlin Flowers	141953
5	Customer Service	Vidya Hanabata	144866
6	Production	Youjian Cronau	138273
7	Finance	Lunjin Swick	142395
8	Sales	Tokuyasu Pesch	158220
9	Marketing	Akemi Warwick	145128

실습 8. (집계함수+문자열함수) 활용 / 그룹핑컬럼 가공

8-2. 현직 사원에 대한 입사연도별 급여 평균

입사연도 순으로 정렬된 결과를 얻기 위해서
Sort group by를 활용하기 위한 hint

출력 결과

	입사연도	급여평균
1	1985	78870
2	1986	77411
3	1987	75928
4	1988	74202
5	1989	73053
6	1990	71484
7	1991	69813
8	1992	68286
9	1993	67091
10	1994	65333
11	1995	63705
12	1996	62425
13	1997	60795
14	1998	59673
15	1999	58199
16	2000	58192

실습 9. UNION ALL을 활용한 실행계획 분리

9-1. 사원 이름으로 사원정보(성명, 연령, 입사일자, 소속부서명, 직급명, 급여) 검색 (first_name으로 찾기 / last_name으로 찾기)

9-1-① 인덱스 생성 (first_name)

```
create index employees_idx3 on employees (first_name) ;
```

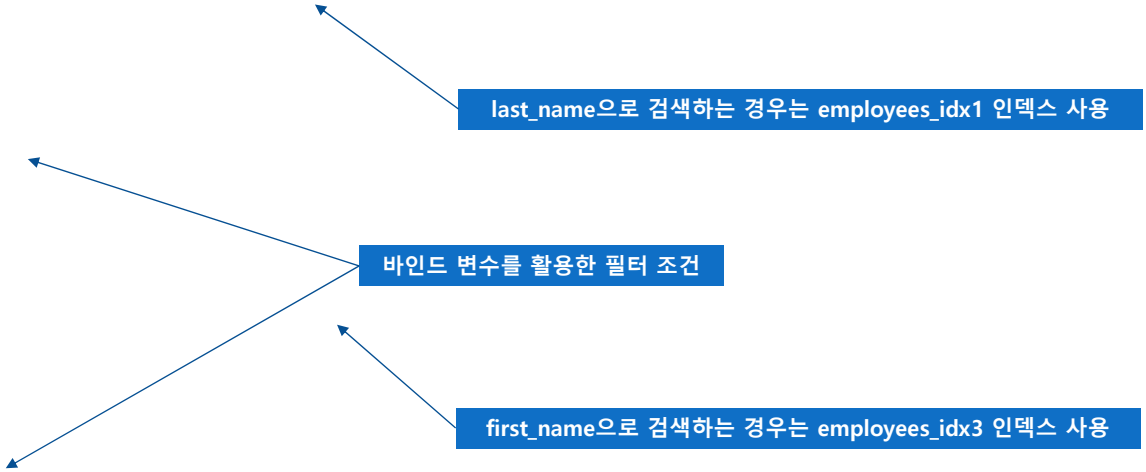
9-1-② 비효율은 없을까? (eg. first_name : Shigeu, last_name : Matzen)

바인드 변수 (bind variable)

실습 9. UNION ALL을 활용한 실행계획 분리

9-1. 사원 이름으로 사원정보(성명, 연령, 입사일자, 소속부서명, 직급명, 급여) 검색 (first_name으로 찾기 / last_name으로 찾기)

9-1-③ 실행계획 분리 (first_name으로 검색하는 경우와 last_name으로 검색하는 경우를 분리하여 SQL 작성)



실습 10. 부분합 / 데이터복제 / ROLLUP과 CUBE

10-1. 현재 부서별 사원들의 급여 합계와 급여 총합계 산출 (데이터복제를 활용한 부분합)

10-1-① 복제 테이블 생성

```
create table copy_t (
  no number(2) not null,
  no2 varchar2(2) not null ) ;
```

10-1-② 복제 테이블 데이터 생성

ROWNUM의 이해

- **정의**
  오라클에서 지원하는 가상컬럼으로 쿼리의 결과에 1부터 하나씩 증가하여 붙는 가상(pseudo) 컬럼
- **주요용도**
  주로 여러개의 결과를 출력하는 쿼리문을 실행 후 결과의 개수를 제한하여 가져오는데 사용 (stop key라고 불림)
- **주의할 점**
  rownum이 결과에서 1부터 순서대로 증가하여 붙기 때문에 rownum=2 나 rownum>1과 같은 방식으로는 원하는 결과를 얻을 수 있음

실습 10. 부분합 / 데이터복제 / ROLLUP과 CUBE

10-1. 현재 부서별 사원들의 급여 합계와 급여 총합계 산출 (데이터복제를 활용한 부분합)

10-1-③ 데이터복제를 통한 급여 부분합(엑셀 부분합과 유사) 산출

- copy_t 테이블과의 곱집합(cartesian product)을 통한 데이터복제
  - copy_t.no = 1 → 부서별 급여합 산출에 사용
  - copy_t.no = 2 → 전체 급여합 산출에 사용

출력 결과

	부서명	급여합
1	Customer Service	1182134209
2	Development	4153249050
3	Finance	977049936
4	Human Resources	824464664
5	Marketing	1188233434
6	Production	3616319369
7	Quality Management	951919236
8	Research	1048650423
9	Sales	3349845802
10	합계	17291866123

실습 10. 부분합 / 데이터복제 / ROLLUP과 CUBE

10-2. 현재 부서별 사원들의 급여 합계와 급여 총합계 산출 (rollup() 함수를 활용한 부분합)

rollup() 함수 사용

앞의 데이터복제 활용결과와 동일

출력 결과

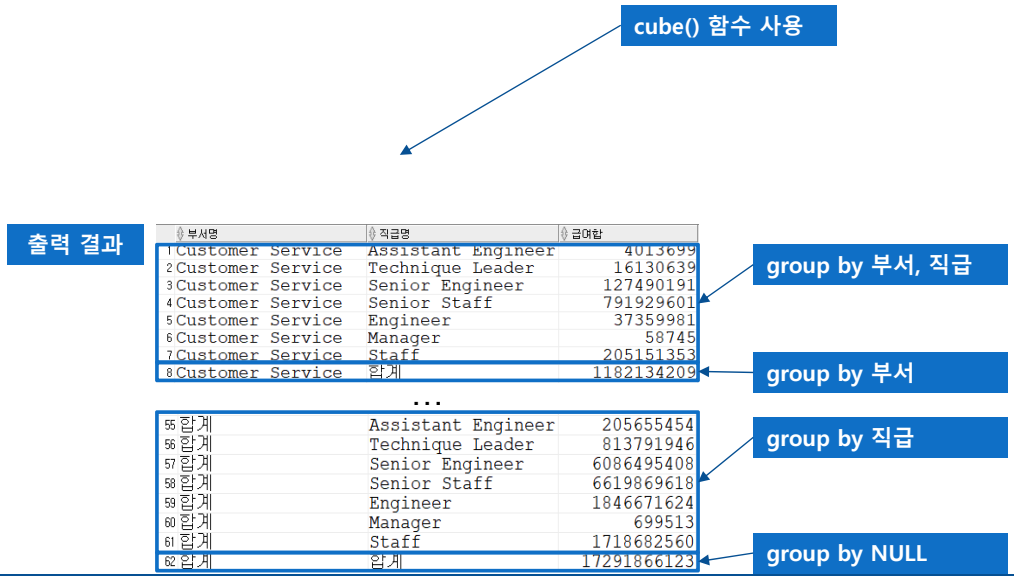
	부서명	급여합
1	Customer Service	1182134209
2	Development	4153249050
3	Finance	977049936
4	Human Resources	824464664
5	Marketing	1188233434
6	Production	3616319369
7	Quality Management	951919236
8	Research	1048650423
9	Sales	3349845802
10	합계	17291866123

개념 이해 10. Rollup() / Cube() / Grouping Sets() 함수

- **Rollup()**
  - ROLLUP에 지정된 Grouping Columns의 List는 Subtotal을 생성하기 위해 사용됨
  - Grouping Columns의 수를 N이라고 했을 때 N+1 Level의 Subtotal이 생성됨
  - Rollup(A, B) = (group by A, B) ∪ (group by A) ∪ (group by NULL)
- **Cube()**
  - 결합 가능한 모든 값에 대하여 다차원 집계를 생성
  - Grouping Columns의 수를 N이라고 했을 때 2^N Level의 Subtotal이 생성됨
  - Cube(A, B) = (group by A, B) ∪ (group by A) ∪ (group by B) ∪ (group by NULL)
- **Grouping Sets()**
  - GROUPING SETS에 표시된 인수들에 대한 개별 집계를 구하기 위해 사용됨
  - Grouping Columns의 수를 N이라고 했을 때 N Level의 Subtotal이 생성됨
  - Grouping Sets(A, B) = (group by A) ∪ (group by B)
- **Grouping()**
  - ROLLUP, CUBE, GROUPING SETS 등 새로운 그룹 함수를 지원하기 위해 추가된 함수
  - If *expr* = (ROLLUP이나 CUBE에 의한 소계가 계산된 결과), grouping(*expr*) = 1. if not, grouping(*expr*) = 0
  - CASE/DECODE를 이용해, 소계를 나타내는 필드에 원하는 문자열을 지정할 수 있음

실습 10. 부분합 / 데이터복제 / ROLLUP과 CUBE

10-3. 현재 부서별/직급별 급여합 및 전체 급여합 산출 (cube() 함수를 활용한 부분합)



	부서명	직급명	급여합
1	Customer Service	Assistant Engineer	4013699
2	Customer Service	Technique Leader	16130639
3	Customer Service	Senior Engineer	127490191
4	Customer Service	Senior Staff	791929601
5	Customer Service	Engineer	37359981
6	Customer Service	Manager	58745
7	Customer Service	Staff	205151353
8	Customer Service	합계	1182134209
...			
55	합계	Assistant Engineer	205655454
56	합계	Technique Leader	813791946
57	합계	Senior Engineer	6086495408
58	합계	Senior Staff	6619869618
59	합계	Engineer	1846671624
60	합계	Manager	699513
61	합계	Staff	1718682560
62	합계	합계	17291866123

실습 11. 분석함수(analytic function)

11-1. 각 사원의 급여와 소속부서 평균급여의 차이

분석함수 avg() over ()

SQL formatted form by SQL Developer

출력 결과

	성명	급여차이
1	Shaowei Kitai	-5290
2	Fox Kugler	-16957
3	Zengping Peir	28542
4	Manton Selvestrel	17111
5	Mark Picht	-15161
6	Ashish Angelopoulos	-13649
7	Saniya Pepe	-574
8	Vincent Nergos	-21248
9	Bouchung Merlo	-19422
10	Sudharsan Langford	-2545
11	Masasuke Koprowski	28218
12	Morris DiGiano	-18672
13	Kasidit Cools	10932
14	Florina Koshiba	6390
15	Pasqua Kilgour	-11008
16	Fun Mawatari	26618
17	Mari Budinsky	-10906
18	Lucian Baak	-417
19	Lijia Litzkow	4620
20	Heon Thiran	14256
21	Goh Kaelbling	-1916
22	Udi Famili	-30872
23	Rildo Pepe	-2577

...

실습 11. 분석함수(analytic function)

11-2. 각 사원의 사번, 사원명, 입사일자, 급여변동일, 변동일당시급여, 급여인상액, 급여누적평균

분석함수 lag() over ()

분석함수 avg() over ()

출력 결과

	사번	성명	입사일자	급여변동일	변동일당시급여	급여인상액	급여누적평균
1	10001	Georgi Facello	86/06/26	86/06/26	60117	(null)	60117
2	10001	Georgi Facello	86/06/26	87/06/26	62102	1985	61110
3	10001	Georgi Facello	86/06/26	88/06/25	66074	3972	62764
4	10001	Georgi Facello	86/06/26	89/06/25	66596	522	63722
5	10001	Georgi Facello	86/06/26	90/06/25	66961	365	64370
6	10001	Georgi Facello	86/06/26	91/06/25	71046	4085	65483
7	10001	Georgi Facello	86/06/26	92/06/24	74333	3287	66747
8	10001	Georgi Facello	86/06/26	93/06/24	75286	953	67814
9	10001	Georgi Facello	86/06/26	94/06/24	75994	708	68723
10	10001	Georgi Facello	86/06/26	95/06/24	76884	890	69539
11	10001	Georgi Facello	86/06/26	96/06/23	80013	3129	70491
12	10001	Georgi Facello	86/06/26	97/06/23	81025	1012	71369
13	10001	Georgi Facello	86/06/26	98/06/23	81097	72	72118
14	10001	Georgi Facello	86/06/26	99/06/23	84917	3820	73032
15	10001	Georgi Facello	86/06/26	00/06/22	85112	195	73837
16	10001	Georgi Facello	86/06/26	01/06/22	85097	-15	74541
17	10001	Georgi Facello	86/06/26	02/06/22	88958	3861	75389
18	10002	Bezalel Simmel	85/11/21	96/08/03	65828	(null)	65828
19	10002	Bezalel Simmel	85/11/21	97/08/03	65909	81	65869

...

개념 이해 11. 분석함수 (Analytic function)

분석함수 (analytic function)

- 테이블에 있는 데이터를 특정 용도로 분석하여 결과를 반환하는 함수
- 쿼리 결과Set을 대상으로 계산을 수행하는 함수
- SELECT 절에서 수행됨
- FROM, WHERE, GROUP BY 절에서 사용 불가
- ORDER BY 구문에서는 사용 가능

집계함수 vs. 분석함수

집계함수

집계함수는 여러행 또는 테이블 전체 행으로부터 그룹별로 집계하여 결과를 반환한다.

```
1  SELECT deptno
2       , SUM(sal) s_sal
3    FROM emp
4   GROUP BY deptno;
```

[그림] 집계함수 실행결과

DEPTNO	S_SAL
10	300
20	200
30	100

분석함수

분석 함수는 집계 결과를 각 행마다 보여준다.

```
1  SELECT deptno
2       , empno
3       , sal
4       , SUM(sal) OVER(PARTITION BY deptno) s_sal
5    FROM emp;
```

[그림] 분석함수 실행결과

DEPTNO	EMPNO	SAL	S_SAL
10	01	50	300
10	02	100	300
10	03	150	300
20	04	100	200
20	05	100	200
30	06	100	100

* 출처: http://www.gurubee.net/lecture/2671 [꿈꾸는 개발자, DBA커뮤니티 구루비]

개념 이해 11. 분석함수 (Analytic function)

집계함수 vs. 분석함수

- 집계함수는 그룹별 최대, 최소, 합계, 평균, 건수 등을 구할 때 사용되며, **그룹별 1개의 행을 반환**
- 분석함수는 그룹마다가 아니라 결과Set의 각 행마다 그룹별 계산결과를 보여줌

Syntax

```
1  SELECT ANALYTIC_FUNCTION ( arguments )
2         OVER ( [ PARTITION BY 컬럼List ]
3                [ ORDER BY 컬럼List ]
4                [ WINDOWING 절 (Rows|Range Between)]
5              )
6    FROM 테이블 명;
```

- ANALYTIC_FUNCTION : 분석함수명(입력인자)
- OVER : 분석함수임을 나타내는 키워드.
- PARTITION BY : 계산 대상 그룹을 정한다.
- ORDER BY : 대상 그룹에 대한 정렬을 수행한다.
- WINDOWING 절 : 분석함수의 계산 대상 범위를 지정한다.
  ORDER BY 절에 종속적이다.
  기본 생략 구문 : 정렬된 결과의 처음부터 현재행까지 [RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW]

분석함수의 종류

* 출처: http://www.gurubee.net/lecture/2671 [꿈꾸는 개발자, DBA커뮤니티 구루비]

- 순위함수 : RANK, DENSE_RANK, ROW_NUMBER, NTILE
- 집계함수 : SUM, MIN, MAX, AVG, COUNT
- 기타함수 : LEAD, LAG, FIRST_VALUE, LAST_VALUE, RATIO_TO_REPORT, KEEP, LISTAGG

실습 12. 행/열 전환(Pivoting)

12-1. 직급별 사원수 산출 : (행→열) 전환

12-1-① 직급별 사원수 산출을 위한 view 생성

```
select * from v_title_emp_rows ;
```

출력 결과

	TITLE	CNT_EMP
1	Staff	25526
2	Manager	9
3	Engineer	30983
4	Technique Leader	12055
5	Assistant Engineer	3588
6	Senior Staff	82024
7	Senior Engineer	85939

12-1-② 직급에 대한 (행→열) 전환

출력 결과

	항목명	MANAGER	TECHNIQUE_LEADER	SENIOR_ENGINEER	ENGINEER	ASSISTANT_ENGINEER	SENIOR_STAFF	STAFF
1	사원수	9	12055	85939	30983	3588	82024	25526

개념 이해 12. 뷰 (View)

- **뷰(view)**
  - 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된 이름을 가지는 가상 테이블
  - 저장장치 내에 물리적으로 존재하지 않지만 사용자에게 있는 것처럼 간주되는 효과
  - 데이터 보정작업, 처리과정 시험 등 임시적인 작업을 위한 용도로 활용
  - 조인문의 사용 최소화로 사용상의 편의성을 최대화
- **특징**
  - 논리적 데이터 독립성을 제공한다.
  - 동일 데이터에 대해 동시에 여러사용자의 상이한 응용이나 요구를 지원해 준다.
  - 사용자의 데이터관리를 간단하게 해준다.
  - 접근 제어를 통한 자동 보안이 제공된다.

실습 12. 행/열 전환(Pivoting)

12-2. 직급별 사원수 산출 : (열→행) 전환

12-2-① 직급별 사원수 산출을 위한 view 생성 (12-1-①번 SQL을 view로 생성)

```
select * from v_title_emp_columns ;
```

출력 결과

	항목명	MANAGER	TECHNIQUE_LEADER	SENIOR_ENGINEER	ENGINEER	ASSISTANT_ENGINEER	SENIOR_STAFF	STAFF
1	사원수	9	12055	85939	30983	3588	82024	25526

실습 12. 행/열 전환(Pivoting)

12-2. 직급별 사원수 산출 : (열→행) 전환

12-2-② 직급에 대한 (열→행) 전환

출력 결과

	TITLE	CNT_EMP
1	Manager	9
2	Technique Leader	12055
3	Senior Engineer	85939
4	Engineer	30983
5	Assistant Engineer	3588
6	Senior Staff	82024
7	Staff	25526

실습 12. 행/열 전환(Pivoting)

12-2. 직급별 사원수 산출 : (열→행) 전환

12-2-③ With문 이용하기 (12-2-①번과 12-2-②번을 합쳐서 표현)

실습 12. 행/열 전환(Pivoting)

12-3. 10-3번 예(부서별/직급별 급여합 산출)에 대한 (행->열) 전환

12-3-① 부서별/직급별 급여합에 대한 view 생성 (10-3번 SQL을 view로 생성)

실습 12. 행/열 전환(Pivoting)

12-3. 10-3번 예(부서별/직급별 급여합 산출)에 대한 (행->열) 전환

12-3-① 부서별/직급별 급여합에 대한 view 생성 (10-3번 SQL을 view로 생성)

```
select * from v_dept_title_salaries ;
```

출력 결과

	부서명	직급명	급여합
1	Customer Service	Assistant Engineer	4013699
2	Customer Service	Technique Leader	16130639
3	Customer Service	Senior Engineer	127490191
4	Customer Service	Senior Staff	791929601
5	Customer Service	Engineer	37359981
6	Customer Service	Manager	58745
7	Customer Service	Staff	205151353
8	Customer Service	합계	1182134209

group by 부서, 직급

group by 부서

...

	부서명	직급명	급여합
55	합계	Assistant Engineer	205655454
56	합계	Technique Leader	813791946
57	합계	Senior Engineer	6086495408
58	합계	Senior Staff	6619869618
59	합계	Engineer	1846671624
60	합계	Manager	699513
61	합계	Staff	1718682560
62	합계	합계	17291866123

group by 직급

group by NULL

실습 12. 행/열 전환(Pivoting)

12-3. 10-3번 예(부서별/직급별 급여합 산출)에 대한 (행->열) 전환

12-3-② 직급에 대한 (행->열) 전환

...

출력 결과

group by 부서, 직급

group by 부서

	부서명	Assitant Engineer	Engineer	Senior Engineer	Staff	Senior Staff	Technique Leader	Manager	합계
1	Customer Service	4013699	37359981	127490191	205151353	791929601	16130639	58745	1182134209
2	Development	95209078	838991757	2754762929	17601672	74353054	372256050	74510	4153249050
3	Finance	(null)	(null)	(null)	196491569	780474910	(null)	83457	977049936
4	Human Resources	(null)	(null)	(null)	165450013	658949251	(null)	65400	824464664
5	Marketing	(null)	(null)	(null)	247470796	940656147	(null)	106491	1188233434
6	Production	80311082	724966534	2394396077	19361437	77216162	320011423	56654	3616319369
7	Quality Management	21607713	195207616	650842874	(null)	(null)	84188157	72876	951919236
8	Research	4513882	50145736	159003337	166313637	647388761	21205677	79393	1048650423
9	Sales	(null)	(null)	(null)	700842083	2648901732	(null)	101987	3349845802
10	합계	205655454	1846671624	6086495408	1718682560	6619869618	813791946	699513	17291866123

group by 직급

group by NULL

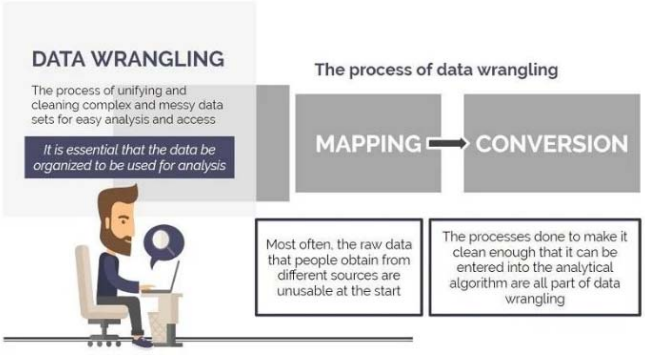
실습 12. 행/열 전환(Pivoting)

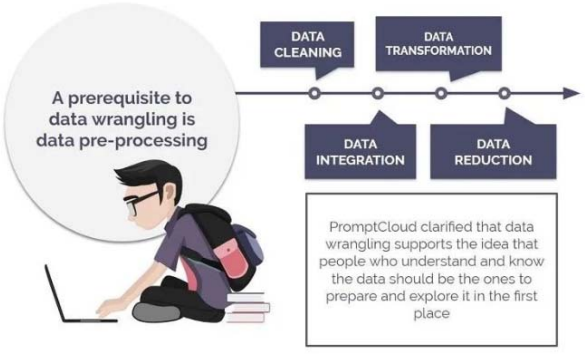
12-3. 10-3번 예(부서별/직급별 급여합 산출)에 대한 (행->열) 전환

12-3-③ With문 이용하기 (12-3-①번과 12-3-②번을 합쳐서 표현)

개념 이해 13. Data Wrangling

- **데이터 랭글링(Data Wrangling) = 데이터 먼징(Data Munging)**
  - 복잡하고 지저분한 상태의 데이터를 간단한 분석과 접근을 위해 통합하는 과정
  - 데이터를 분석할 수 있는 상태로 조직하는 필수 과정 (데이터 전처리에서의 필수 과정)
  - 데이터 전처리 : 데이터 정제, 데이터 통합, 데이터 변형, 데이터 축소 등
  - Data Wrangling vs. ETL(Extraction/Transformation/Loading)
    - ✓ Data Wrangling : 분석가의 관점. 데이터를 준비하는 과정 (수동)
    - ✓ ETL : 최종 사용자 관점. 대부분 루틴한 과정 (자동)





* 출처: https://m.post.naver.com/viewer/postView.nhn?volumeNo=18028254&memberNo=43011790 [데이터 랭글링이 ... 이유]

개념 이해 14. With 문

With 문

- WITH 구문내의 쿼리의 결과(SUB쿼리)가 여러번 사용될때(호출될때) 유용
- 서브쿼리 블럭에 이름을 지정할 수 있도록 해 줌.
- 오라클 옵티마이저는 쿼리를 인라인뷰(inline view 방식)나 임시 테이블(materialized 방식)로 여김.
- Oracle 9 이상 지원

[WITH 구문 사용방법]

```
WITH ALIAS명 AS ( SUB쿼리 )
SELECT 컬럼명 FROM ALIAS명;
```

WITH 구문 예제)

```
WITH AA AS
  (SELECT ROWNUM, 'TEST1', SYSDATE
     FROM DUAL
   UNION ALL
   SELECT ROWNUM, 'TEST2', SYSDATE
     FROM DUAL
   UNION ALL
   SELECT ROWNUM, 'TEST3', SYSDATE FROM DUAL)
SELECT * FROM AA;
```

	ROWNUM	'TEST1'	SYSDATE
1	1	test1	2014-04-10 오후 2:39:55
2	1	test2	2014-04-10 오후 2:39:55
3	1	test3	2014-04-10 오후 2:39:55

[WITH 구문(2개 SUB쿼리) 사용방법]

```
WITH ALIAS명_1 AS ( SUB쿼리 ),
     ALIAS명_2 AS ( SUB쿼리 )
SELECT 컬럼명 FROM ALIAS명 where 조인조건;
```

WITH 구문(2개 SUB쿼리) 예제)

```
WITH AA AS
 (SELECT ROWNUM AS SEQ, 'TEST1' AS NAME, SYSDATE
    FROM DUAL
  UNION ALL
  SELECT ROWNUM AS SEQ, 'TEST2' AS NAME, SYSDATE
    FROM DUAL
  UNION ALL
  SELECT ROWNUM AS SEQ, 'TEST3' AS NAME, SYSDATE FROM DUAL),
BB AS
 (SELECT ROWNUM AS SEQ, 'TEST1' AS NAME, SYSDATE
    FROM DUAL
  UNION ALL
  SELECT ROWNUM AS SEQ, 'TEST2' AS NAME, SYSDATE
    FROM DUAL
  UNION ALL
  SELECT ROWNUM AS SEQ, 'TEST3' AS NAME, SYSDATE FROM DUAL)

SELECT * FROM AA, BB WHERE AA.NAME=BB.NAME
```

	SEQ	NAME	SYSDATE	SEQ	NAME	SYSDATE
1	1	test1	2014-04-10 오후 3:00:57	1	test1	2014-04-10 오후 3:00:57
2	1	test2	2014-04-10 오후 3:00:57	1	test2	2014-04-10 오후 3:00:57
3	1	test3	2014-04-10 오후 3:00:57	1	test3	2014-04-10 오후 3:00:57

* 출처: https://powerofwriting.tistory.com/entry/Oracle-WITH-구문-예제