

기본적인 데이터 조회



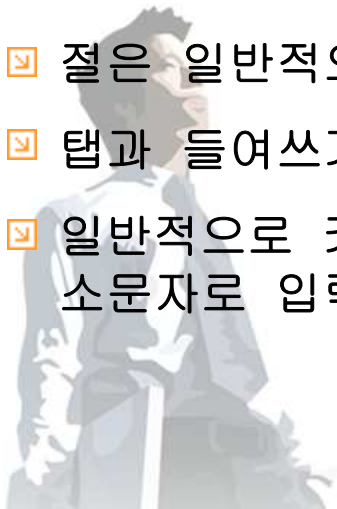
1. SQL 문법

1.1 필수 항목

- SQL문은 대소문자를 구분하지 않음
- SQL문은 하나 이상의 행에 입력 가능
- 키워드 (**keyword**)는 약어로 쓰거나 다음 행에 나눠쓰기 불가능
- 마지막 절의 끝에 세미콜론 (;) 입력

1.2 권장 사항

- 절은 일반적으로 서로 다른 행에 씀
- 탭과 들여쓰기를 사용하면 좀더 읽기 쉬운 코드 작성 가능
- 일반적으로 키워드는 대문자로 입력하고 테이블 이름, 열 등 다른 단어는 모두 소문자로 입력 권장



기본적인 데이터 조회

2. 테이블 구조

2.1 emp 테이블 구성

열 이름	empno	ename	job	mgr	hiredate	sal	comm	deptno
키 유형	PK							
널/고유	NN/UK	NN						
FK 테이블								dept
FK 컬럼								deptno
데이터 유형	NUMBER	VARCHAR2	VARCHAR2	NUMBER	DATE	NUMBER	NUMBER	NUMBER
길이	4	10	9	4	7,2	7,2	7,2	2

2.2 emp 테이블 구조 확인

```
SQL> DESC emp
```

이름	널?	유형
EMPNO	NOT NULL	NUMBER (4)
ENAME	NOT NULL	VARCHAR2 (10)
JOB		VARCHAR2 (9)
MGR		NUMBER (4)
HIREDATE		DATE
SAL		NUMBER (7, 2)
COMM		NUMBER (7, 2)
DEPTNO		NUMBER (2)

기본적인 데이터 조회

2.3 emp 테이블 조회

```
SQL> SELECT *
      2 FROM emp ;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81/11/17	5000		10
7698	BLAKE	MANAGER	7839	81/05/01	2850		30
7782	CLARK	MANAGER	7839	81/06/09	2450		10
7566	JONES	MANAGER	7839	81/04/02	2975		20
7902	FORD	ANALYST	7566	81/10/03	3000		20
7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30
7521	WARD	SALESMAN	7698	81/02/22	1250	500	30
7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30
7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30
7900	JAMES	CLERK	7698	81/10/03	950		30
7934	MILLER	CLERK	7782	82/01/23	1300		10
7788	SCOTT	ANALYST	7566	82/10/09	3000		20
7369	SMITH	CLERK	7902	80/10/17	800		20
7876	ADAMS	CLERK	7788	83/01/12	1100		20

3. 기본 SELECT 문

3.1 구문

```
SELECT    [DISTINCT] {*, column [alias],...}  
FROM      table;
```

- ☐ SELECT 하나 이상의 열로 구성되는 목록
- ☐ DISTINCT 중복되는 열 생략
- ☐ * 모든 열 선택
- ☐ *column* 명명된 열 선택
- ☐ *alias* 선택된 열에 다른 머리글 부여
- ☐ FROM *table* 해당 열을 포함하는 테이블 지정



기본적인 데이터 조회

3.2 모든 열 선택

```
SELECT    *  
FROM      emp ;
```

3.3 특정 열 선택

```
SELECT    empno  
FROM      emp ;
```

```
SELECT    empno ,  ename  
FROM      emp ;
```

```
SELECT    empno ,  empno  
FROM      emp ;
```



3.4 열 머리글 기본값

☐ iSQL*Plus

- 기본 머리글 정렬 : 가운데
- 기본 머리글 표시 : 대문자

☐ SQL*Plus

- 문자 및 날짜 열 머리글 : 왼쪽 정렬
- 숫자 열 머리글 : 오른쪽 정렬
- 기본 머리글 표시 : 대문자



3.5 산술 연산자

- ☐ SQL에서 사용되는 산술 연산자는 C와 같은 다른 프로그래밍 언어에서 사용되는 연산자와 동일

연산자	설명
+	더하기
-	빼기
*	곱하기
/	나누기

```
SQL> SELECT empno, sal, sal+100  
2 FROM emp;
```

```
SQL> SELECT empno, sal, comm, sal+comm  
2 FROM emp;
```

3.5 산술 연산자 (계속)

☞ 산술 연산자 우선순위

- 곱셈과 나눗셈은 덧셈과 뺄셈보다 우선순위가 높음
- 우선순위가 동일한 연산자는 왼쪽에서 오른쪽으로 계산
- 괄호는 우선순위를 적용하여 계산하고 명령문을 명확히 나타내는데 사용

```
SQL> SELECT ename, sal, 12*sal+100  
2 FROM emp;
```

```
SQL> SELECT ename, sal, 12*(sal+100)  
2 FROM emp;
```

3.6 널(NULL) 값 정의

- 널 값은 알 수 없는 값 (unknown), 사용할 수 없는 값 (unavailable), 할당할 수 없는 값 (unassigned), 적용할 수 없는 값 (inapplicable)을 의미
- 널은 0 또는 공백과 다름

```
SQL> SELECT ename, job, sal, comm
2 FROM emp;
```

ENAME	JOB	SAL	COMM
KING	PRESIDENT	5000	
BLAKE	MANAGER	2850	
CLARK	MANAGER	2450	
JONES	MANAGER	2975	
FORD	ANALYST	3000	
ALLEN	SALESMAN	1600	300
...			

3.7 산술식의 널 값

☐ 널 값을 포함하는 산술식은 널로 평가

```
SQL> SELECT ename, sal* 12 + comm  
2 FROM emp;
```

ENAME	SAL*12+COMM
-----	-----
KING	
BLAKE	
CLARK	
JONES	
FORD	
ALLEN	19500
WARD	15500
MARTIN	16400
TURNER	18000
JAMES	
...	

3.8 열 별칭 (Column Alias)

- ☐ 열 (Column) 머리글의 이름 변경
- ☐ 계산할 때 유용
- ☐ 열 이름 바로 다음에 입력
- ☐ 열 이름과 별칭 (Alias) 사이에 선택사항인 AS 키워드 사용
- ☐ 공백 또는 특수문자가 있거나 대소문자를 구분할 경우 큰 따옴표 (" ") 사용

```
SQL> SELECT ename AS name, sal salary  
2 FROM emp;
```

```
SQL> SELECT ename "Name", sal*12 "Annual Salary"  
2 FROM emp;
```

```
SQL> SELECT empno, sal 연봉#, sal "#연봉"  
2 FROM emp;
```

3.9 연결 연산자 (Concatenation Operator)

- ☐ 열 또는 문자열을 다른 열에 연결
- ☐ 두 개의 세로선 (||)으로 표시
- ☐ 문자식인 결과 열을 생성

```
SQL> SELECT  ename || job  
2 FROM      emp ;
```

```
SQL> SELECT  ename || job AS "사원 직책"  
2 FROM      emp ;
```



3.10 리터럴(Literal) 문자열

- ❑ 리터럴은 **SELECT** 목록에 포함된 문자, 숫자 또는 날짜
- ❑ 날짜 및 문자 리터럴 값은 작은 따옴표('')로 묶음
- ❑ 각 문자열은 각 행이 반환될 때마다 한 번 출력

```
SQL> SELECT ename||'씨는 '|| job ||'입니다.'  
2          AS "사원 직책"  
3 FROM      emp;
```



3.11 중복 행(row) 제거

- ☐ 특별히 표시하지 않는 한 **SQL*Plus**는 중복 행을 제거하지 않은 상태로 질의 결과를 표시

```
SQL> SELECT deptno  
2 FROM emp;
```

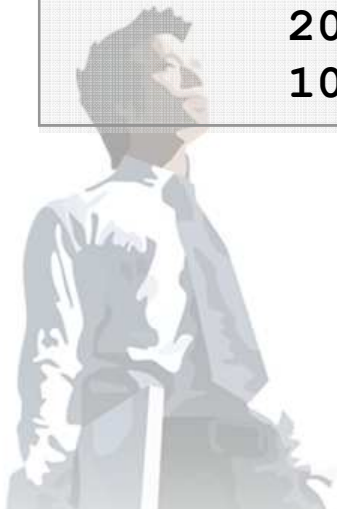
```
DEPTNO  
-----  
10  
30  
10  
20  
20  
30  
30  
30  
30  
...
```


3.11 중복 행(row) 제거 (계속)

☐ SELECT 절에서 SELECT 다음에 DISTINCT 키워드를 사용하여 중복 행을 제거

```
SQL> SELECT DISTINCT deptno  
      2 FROM emp;
```

DEPTNO
30
20
10



3.12 ALL 키워드

☐ ALL 키워드는 모든 값을 출력 (default)

```
SQL> SELECT ALL deptno  
2 FROM emp;
```

DEPTNO

10
30
10
20
20
30
30
30
30
...

데이터 제한 및 정렬

데이터 제한 및 정렬



1. WHERE 절

1.1 구문

☐ WHERE 절을 사용하여 반환되는 행을 제한

☐ WHERE 절은 FROM 절 다음에 위치

```
SELECT          [DISTINCT] { * | column [alias], ... }  
FROM            table  
[WHERE          condition(s) ] ;
```

☐ WHERE 조건을 만족하는 행만 질의하도록 제한

☐ *condition(s)* 열 이름, 표현식, 상수 및 비교 연산자로 구성



데이터 제한 및 정렬

1.2 where절에서 문자열 및 날짜

- ☐ 문자열 또는 날짜 값은 작은 따옴표(' ')로 묶음
- ☐ 문자 값은 대소문자를 구분하며 날짜 값은 형식을 구분
- ☐ 기본 날짜 형식은 DD-MON-YY

```
SQL> SELECT empno, ename, deptno  
2 FROM emp  
3 WHERE deptno = 10;
```

```
SQL> SELECT empno, ename, deptno  
2 FROM emp  
3 WHERE ename = 'MILLER';
```

```
SQL> SELECT empno, ename, hiredate, deptno  
2 FROM emp  
3 WHERE hiredate = '83/01/12';
```

데이터 제한 및 정렬

2. 비교 연산자

연산자	의미
=	같음
>	보다 큼
>=	크거나 같음
<	보다 작음
<=	작거나 같음
<> (!=)	같지 않음

```
SQL> SELECT *  
  2  FROM emp  
  3  WHERE deptno = 10;
```

```
SQL> SELECT *  
  2  FROM emp  
  3  WHERE deptno > 10;
```

2. 비교 연산자 (계속)

```
SQL> SELECT empno, ename, job  
2 FROM emp  
3 WHERE empno = 7934;
```

```
SQL> SELECT empno, ename  
2 FROM emp  
3 WHERE ename = 'BLAKE';
```

```
SQL> SELECT *  
2 FROM emp  
3 WHERE hiredate = '83/01/12';
```



2. 비교 연산자 (계속)

```
SQL> SELECT value  
      2 FROM NLS_SESSION_PARAMETERS  
      3 WHERE parameter = 'NLS_DATE_FORMAT' ;
```

VALUE

RR/MM/DD

```
SQL> ALTER SESSION  
      2 SET NLS_DATE_FORMAT = 'YYYY-MM-DD' ;
```

```
SQL> SELECT empno, hiredate  
      2 FROM emp ;
```



3.1 BETWEEN 연산자

- `value [NOT] BETWEEN low AND high`
- 검사할 값 ← ← 최소범위한계 ← 최대범위한계
- `<= and >=`

```
SQL> SELECT      ename, sal
2 FROM          emp
3 WHERE         sal BETWEEN 1000 AND 1500;
```

EMPNO	ENAME	SAL
7521	WARD	1250
7654	MARTIN	1250
7844	TURNER	1500
7934	MILLER	1300
7876	ADAMS	1100

3.2 IN 연산자

- IN 연산자를 사용하여 목록(list) 값을 조회
- 목록에 문자형이나 날짜형이면 반드시 작은 따옴표(' ')를 붙임

value [NOT] IN (*value1*, *value2*, *value3*)

→ 검사할 값

→ , 로 분리된 값 목록

```
SQL> SELECT empno, ename
2 FROM emp
3 WHERE empno IN (7934, 7502, 7500);
```

EMPNO ENAME

7934 MILLER

3.3 LIKE 연산자

- ☐ **LIKE** 연산자를 사용하면 유효한 검색 문자열 값인 대체 문자를 사용하여 검색할 수 있음
- ☐ 검색 조건은 리터럴 문자 또는 숫자를 포함할 수 있음
 - % 에는 문자가 오지 않거나 여러 개 올 수 있음
 - _ 에는 문자가 하나만 올 수 있음

```
SQL> SELECT empno, ename  
2 FROM emp  
3 WHERE ename LIKE 'A%';
```

```
SQL> SELECT empno, ename  
2 FROM emp  
3 WHERE ename LIKE '%T';
```

```
SQL> SELECT empno, ename  
2 FROM emp  
3 WHERE ename LIKE 'MILLE_';
```

3.3 LIKE 연산자 (계속)

☐ **ESCAPE** 식별자를 사용하여 “%” 또는 “_”를 검색할 수 있음

```
SQL> INSERT INTO dept
      2 VALUES (99, 'SALES_TEST', 'SEOUL');
```

```
SQL> SELECT *
      2 FROM dept
      3 WHERE dname LIKE '%_%';
```

```
SQL> SELECT *
      2 FROM dept
      3 WHERE dname LIKE '%\_%' ESCAPE '\';
```

3.4 IS NULL 연산자

- IS NULL 연산자를 사용하여 널 값을 조회
- 값이 없음을 확인하는 데 사용

value IS [NOT] NULL

└─ 검사할 변수

```
SQL> SELECT empno, comm  
2 FROM emp  
3 WHERE comm = NULL;
```

```
SQL> SELECT empno, comm  
2 FROM emp  
3 WHERE comm IS NULL;
```

4. 논리 연산자

4.1 AND 연산자

☐ AND는 조건이 모두 TRUE일 경우 TRUE

☐ AND 진리표

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

```
SQL> SELECT empno, ename, job, sal, deptno  
2 FROM emp  
3 WHERE sal < 3000  
4 AND deptno = 10;
```

4.2 OR 연산자

☐ OR는 조건 중 하나가 TRUE이면 TRUE

☐ OR 진리표

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

```
SQL> SELECT empno, ename, job, sal, deptno  
2 FROM emp  
3 WHERE sal < 3000  
4 OR deptno = 10;
```

4.3 NOT 연산자

☐ NOT 연산자는 BETWEEN, LIKE, NULL 등 다른 SQL 연산자와 함께 사용 가능

- WHERE duty **NOT** IN ('사장', '대리')
- WHERE sal **NOT** BETWEEN 1000 AND 1500
- WHERE empname **NOT** LIKE '%종%'
- WHERE bonus **IS NOT** NULL

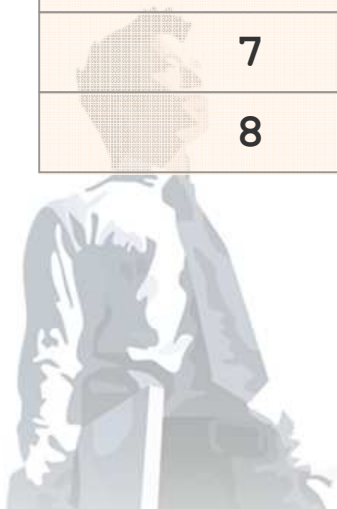
☐ NOT 진리표

NOT	TRUE	FALSE	NULL
TRUE	FALSE	TRUE	TRUE

```
SQL> SELECT empno, ename, job
2 FROM emp
3 WHERE job NOT IN ('SALESMAN', 'CLERK');
```

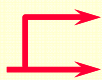

5. 연산자 우선순위 규칙

우선 순위	연산자
1	산술 연산자
2	연결 연산자
3	비교 연산자
4	IS NULL, LIKE, IN
5	BETWEEN
6	NOT 연산자
7	AND 연산자
8	OR 연산자

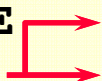


5. 연산자 우선순위 규칙 (계속)

```
SQL> SELECT  empno, ename, job, sal
  2  FROM      emp
  3  WHERE     job = 'SALESMAN'
  4  OR        job = 'PRESIDENT'
  5  AND       sal >= 2000;
```



```
SQL> SELECT  empno, ename, job, sal
  2  FROM      emp
  3  WHERE     (job = 'SALESMAN'
  4  OR        job = 'PRESIDENT' )
  5  AND       sal >= 2000;
```



6. ORDER BY 절

6.1 개요

- ☐ ORDER BY 절을 사용하여 행을 정렬
 - ASC : 오름차순, 기본값
 - DESC : 내림차순
- ☐ ORDER BY 절은 SELECT 절의 가장 끝에 위치
- ☐ ORDER BY 절을 사용하지 않으면 정렬 순서를 정의할 수 없으며, 동일한 질의를 두 번 할 경우 동일한 순서로 조회되지 않음
- ☐ ORDER BY 목록의 순서가 정렬 순서
- ☐ SELECT 목록에 없는 열을 기준으로 정렬할 수도 있음



데이터 제한 및 정렬

6.2 오름차순 정렬

```
SQL> SELECT    empno, ename, hiredate  
2  FROM      emp  
3  ORDER BY   hiredate ;
```

EMPNO	ENAME	HIREDATE
7369	SMITH	80/10/17
7499	ALLEN	81/02/20
7521	WARD	81/02/22
7566	JONES	81/04/02
7698	BLAKE	81/05/01
7782	CLARK	81/06/09
7844	TURNER	81/09/08
7654	MARTIN	81/09/28
7902	FORD	81/10/03
7900	JAMES	81/10/03
7839	KING	81/11/17
7934	MILLER	82/01/23
7788	SCOTT	82/10/09
7876	ADAMS	83/01/12

데이터 제한 및 정렬

6.3 내림차순 정렬

```
SQL> SELECT      empno, ename, hiredate  
2  FROM          emp  
3  ORDER BY      hiredate DESC;
```

EMPNO	ENAME	HIREDATE
7876	ADAMS	83/01/12
7788	SCOTT	82/10/09
7934	MILLER	82/01/23
7839	KING	81/11/17
7902	FORD	81/10/03
7900	JAMES	81/10/03
7654	MARTIN	81/09/28
7844	TURNER	81/09/08
7782	CLARK	81/06/09
7698	BLAKE	81/05/01
7566	JONES	81/04/02
7521	WARD	81/02/22
7499	ALLEN	81/02/20
7369	SMITH	80/10/17

데이터 제한 및 정렬

6.4 열 별칭 정렬

```
SQL> SELECT      empno, ename, sal*12 annsal  
2  FROM          emp  
3  ORDER BY      annsal;
```

EMPNO	ENAME	ANNSAL
7369	SMITH	9600
7900	JAMES	11400
7876	ADAMS	13200
7521	WARD	15000
7654	MARTIN	15000
7934	MILLER	15600
7844	TURNER	18000
7499	ALLEN	19200
7782	CLARK	29400
7698	BLAKE	34200
7566	JONES	35700
7788	SCOTT	36000
7902	FORD	36000
7839	KING	60000

데이터 제한 및 정렬

6.5 여러 열 정렬

```
SQL> SELECT      ename, deptno, sal  
2 FROM          emp  
3 ORDER BY deptno, sal DESC;
```

ENAME	DEPTNO	SAL
KING	10	5000
CLARK	10	2450
MILLER	10	1300
SCOTT	20	3000
FORD	20	3000
JONES	20	2975
ADAMS	20	1100
SMITH	20	800
BLAKE	30	2850
ALLEN	30	1600
TURNER	30	1500
MARTIN	30	1250
WARD	30	1250
JAMES	30	950

집합 (set) 연산자

집합(set) 연산자



집합 (set) 연산자

1. 집합(SET) 연산자 개요

1.1 집합 연산자 규칙

- ❑ **SELECT** 목록에 있는 표현식의 개수와 데이터형은 일치
- ❑ **UNION ALL**을 제외하고는 모두 중복되는 행을 자동적으로 제거
- ❑ 첫번째 질의의 열 이름이 결과에 출력
- ❑ **UNION ALL**을 제외하고는 모든 결과는 **default**로 오름차순 정렬
- ❑ 실행순서를 변경하기 위해서 괄호를 사용
- ❑ **ORDER BY**절은 문장의 가장 마지막에 위치
- ❑ **ORDER BY**절에서 사용된 열 이름이나 별칭은 첫번째 **SELECT** 목록에 위치
- ❑ 집합 연산자는 서브쿼리에 사용 가능



집합 (set) 연산자

1.2 집합 연산자 종류

연산자	의 미
UNION	검색된 결과에서 모든 레코드 검색 (중복값 제외)
UNION ALL	검색된 결과에서 모든 레코드 검색 (중복값 포함)
MINUS	첫 번째 문장에 두 번째 문장을 뺀 나머지 결과
INTERSECT	검색된 결과에서 공통된 레코드 검색

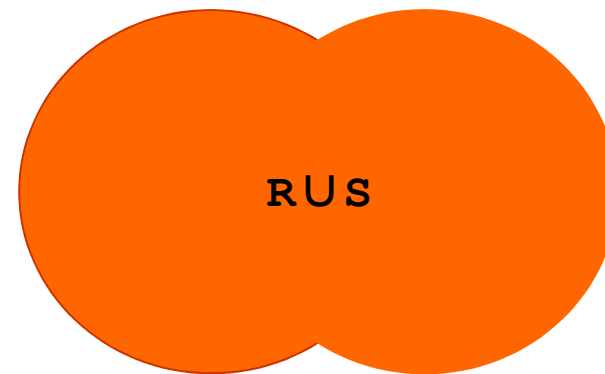
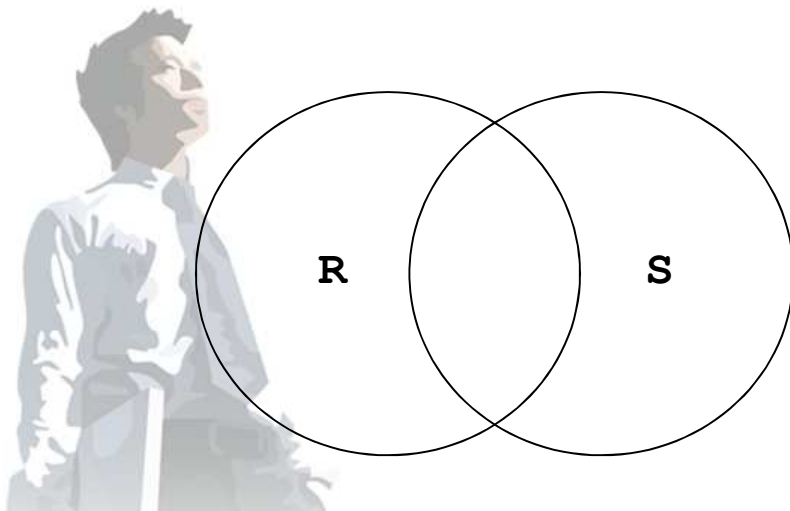


집합 (set) 연산자

2. 집합 연산자 활용

2.1 UNION

- ☐ 두 테이블의 조합
- ☐ 열의 수와 데이터 유형은 두개의 **SELECT** 문장에서 일치
- ☐ 열의 이름은 똑같은 필요 없음
- ☐ **IN** 연산자는 **UNION** 연산자보다 높은 우선순위
- ☐ 결과는 디폴트로 오름차순



집합 (set) 연산자

2.1 UNION (계속)

```
SQL> CONN hr/hr
```

```
SQL> SELECT  employee_id, first_name
2  FROM      employees
3  WHERE     hire_date LIKE '94%'
4  UNION
5  SELECT  employee_id, first_name
6  FROM      employees
7  WHERE     department_id = 100;
```

EMPLOYEE_ID	FIRST_NAME
108	Nancy
109	Daniel
114	Den
203	Susan
204	Hermann
205	Shelley
206	William

EMPLOYEE_ID	FIRST_NAME
108	Nancy
109	Daniel
110	John
111	Ismael
112	Jose Manuel
113	Luis

집합 (set) 연산자

2.1 UNION (계속)

```
SQL> SELECT  employee_id
2  FROM      employees
3  UNION
4  SELECT    department_id, department_name
5  FROM      departments;
```

```
select employee_id
```

```
*
```

1행에 오류:

ORA-01789: 질의의 결과 열의 수가 틀립니다



집합 (set) 연산자

2.1 UNION (계속)

```
SQL> SELECT  first_name, department_id
2  FROM      employees
3  UNION
4  SELECT  department_id, department_name
5  FROM      departments;
```

```
SELECT  first_name, department_id
        *
```

1행에 오류:

ORA-01790: 대응하는 식과 같은 데이터 유형이어야 합니다



집합 (set) 연산자

2.1 UNION (계속)

```
SQL> SELECT    employee_id, first_name
  2  FROM      employees
  3  ORDER BY   first_name
  4  UNION
  5  SELECT    employee_id, first_name
  6  FROM      employees;
```

UNION

*

4행에 오류:

ORA-00933: SQL 명령어가 올바르게 종료되지 않았습니다



집합 (set) 연산자

2.1 UNION (계속)

```
SQL> SELECT      employee_id, first_name
  2  FROM          employees
  3  UNION
  4  SELECT      employee_id, first_name
  5  FROM          employees
  6  ORDER BY    first_name;
```

EMPLOYEE_ID	FIRST_NAME
-------------	------------

121	Adam
-----	------

196	Alana
-----	-------

147	Alberto
-----	---------

103	Alexander
-----	-----------

115	Alexander
-----	-----------

185	Alexis
-----	--------

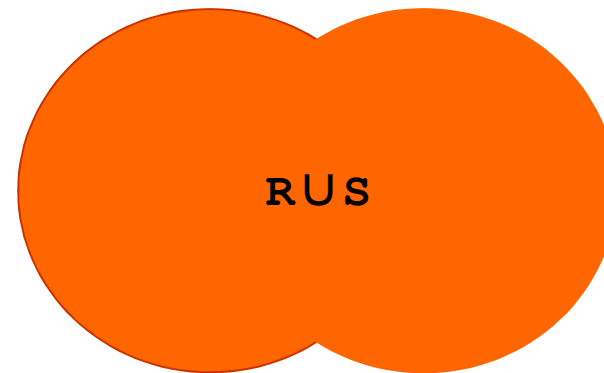
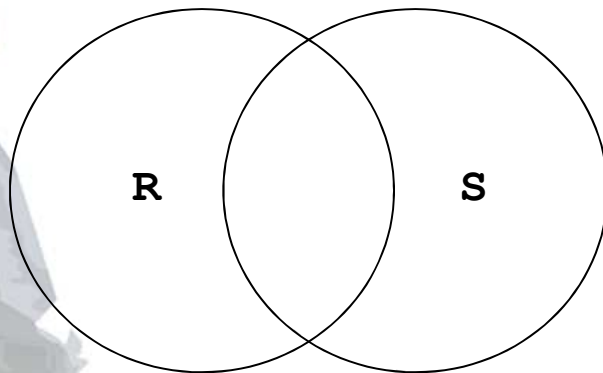
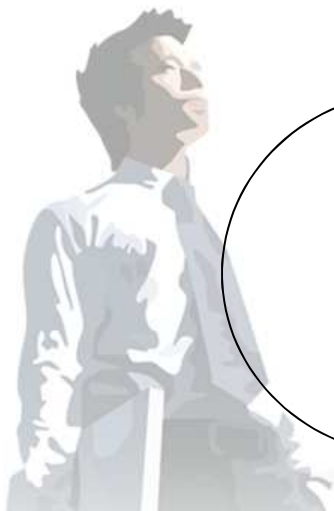
158	Allan
-----	-------

...	
-----	--

집합 (set) 연산자

2.2 UNION ALL

- 모든 행을 리턴
- UNION과는 달리 중복되는 행은 제거되지 않으며, 결과는 디폴트로 정렬되지 않음
- DISTINCT 키워드를 사용 못함
- 나머지는 UNION과 동일



집합 (set) 연산자

2.2 UNION ALL (계속)

```

SQL> SELECT  employee_id, first_name
      2 FROM    employees
      3 WHERE   hire_date LIKE '94%'
      4 UNION ALL
      5 SELECT  employee_id, first_name
      6 FROM    employees
      7 WHERE   department_id = 100;

```

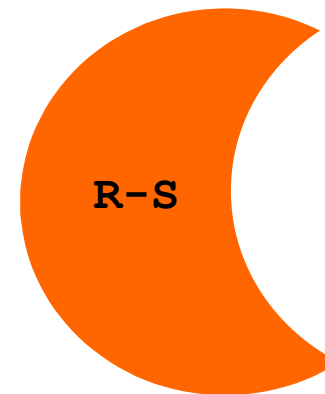
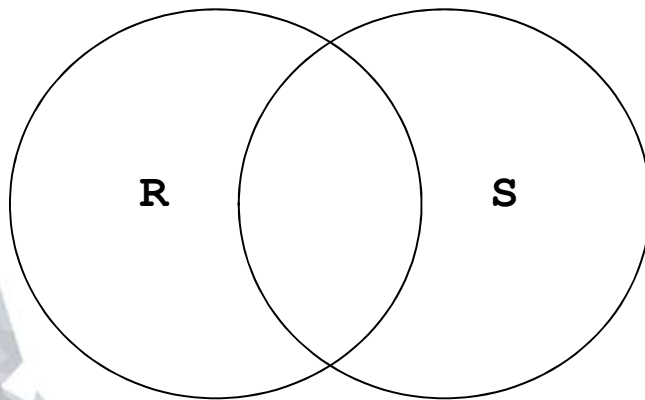
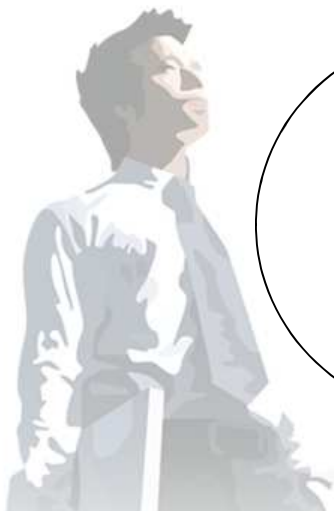
EMPLOYEE_ID	FIRST_NAME
108	Nancy
109	Daniel
114	Den
203	Susan
204	Hermann
205	Shelley
206	William

EMPLOYEE_ID	FIRST_NAME
108	Nancy
109	Daniel
110	John
111	Ismael
112	Jose Manuel
113	Luis

집합 (set) 연산자

2.3 MINUS

- ☐ 두 번째 질의에 의해 리턴되는 행을 제외하고 첫번째 질의에 의해서만 리턴되는 행을 리턴
- ☐ 열의 수와 데이터형은 두개의 **SELECT** 문장에서 일치
- ☐ 열의 이름은 똑같은 필요 없음



집합 (set) 연산자

2.3 MINUS (계속)

```

SQL> SELECT  employee_id, first_name
      2 FROM    employees
      3 WHERE   hire_date LIKE '94%'
      4 MINUS
      5 SELECT  employee_id, first_name
      6 FROM    employees
      7 WHERE   department_id = 100;

```

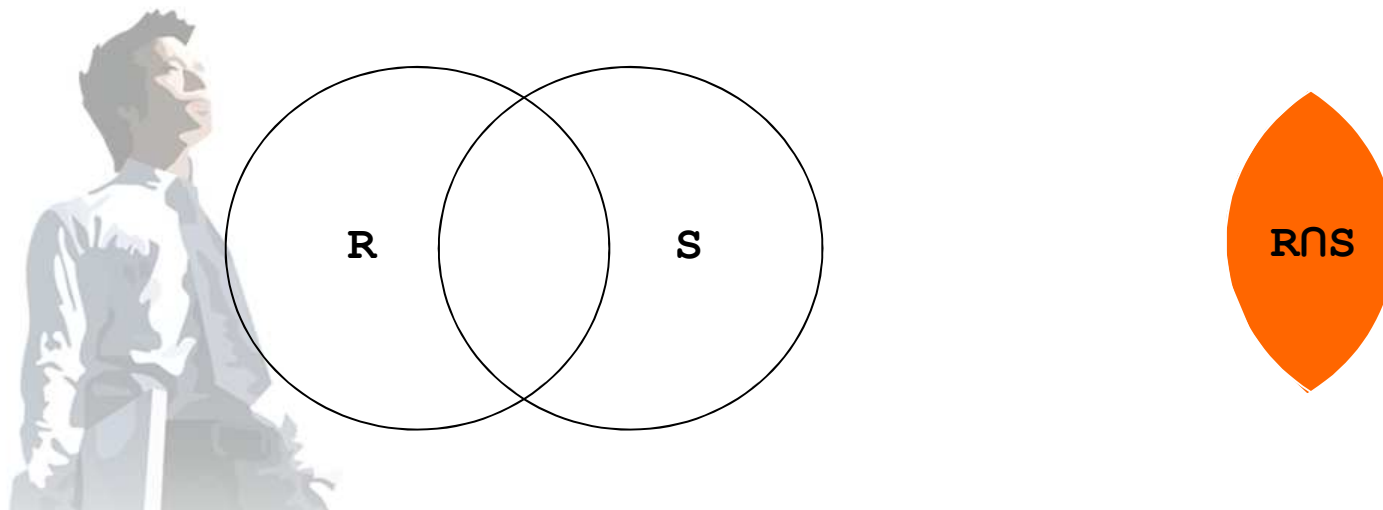
EMPLOYEE_ID	FIRST_NAME
108	Nancy
109	Daniel
114	Den
203	Susan
204	Hermann
205	Shelley
206	William

EMPLOYEE_ID	FIRST_NAME
108	Nancy
109	Daniel
110	John
111	Ismael
112	Jose Manuel
113	Luis

집합 (set) 연산자

2.4 INTERSECT

- ☐ 두 질의에서 모든 공통 행을 리턴
- ☐ 열의 수와 데이터형은 두개의 **SELECT** 문장에서 일치
- ☐ 열의 이름은 똑같은 필요 없음
- ☐ **INTERSECT**된 테이블의 순서를 바꾸어도 결과는 바뀌지 않음



집합 (set) 연산자

2.4 INTERSECT (계속)

```

SQL> SELECT  employee_id, first_name
  2  FROM      employees
  3  WHERE     hire_date LIKE '94%'
  4  INTERSECT
  5  SELECT  employee_id, first_name
  6  FROM      employees
  7  WHERE     department_id = 100;

```

EMPLOYEE_ID	FIRST_NAME
108	Nancy
109	Daniel
114	Den
203	Susan
204	Hermann
205	Shelley
206	William

EMPLOYEE_ID	FIRST_NAME
108	Nancy
109	Daniel
110	John
111	Ismael
112	Jose Manuel
113	Luis