

오라클 11g XE



훈련교사 : 전 은 석

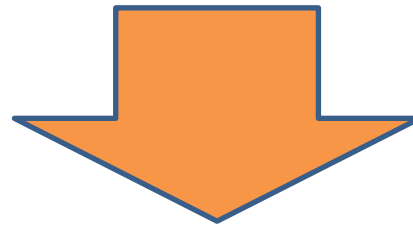
- 데이터베이스 특징
- 데이터베이스 시스템
- 관계 데이터베이스 관리 시스템(RDBMS)
- 데이터 모델과 데이터베이스의 발전사

❖ 데이터베이스 란

“ 어느 특정 조직의 응용 업무에 공동 사용하기 위하여 운영상

필요한 데이터를 **완벽화**, **비중복화**, **구조화**하여 컴퓨터 기억 장치에

저장한 데이터의 집합체”



“발생한 데이터를 통괄적인 관점에서 서로 연관된 정보의 **중복을**

최소화하여 한곳에 모아 저장함으로써 다수의 사용자로 하여금

필요한 정보를 공유하도록 한 정보의 집합체”

❖ 데이터베이스의 특징

- 물리적·논리적 데이터 독립을 지원한다.
- 중복을 최소화 자료의 불일치성을 피할 수 있다.
- 데이터를 공유(sharing)할 수 있다.
- 정보를 표준화(standardization)하여 저장한다.
- 보안성(security)을 제공한다.
- 무결성(integrity)이 유지된다.
- 상충되는 요구를 조절한다.

❖ 데이터베이스의 시스템

● 데이터베이스

- 중앙집중식 데이터베이스, 분산식 데이터베이스

● 하드웨어

- 범용컴퓨터, 전용컴퓨터

● 소프트웨어

- 데이터베이스 관리 시스템(DBMS)

● 사용자

- 데이터베이스 관리자(database administrator)
- 응용프로그래머(application programmer)
- 단말 사용자(end user)

❖ 관계 데이터베이스 관리 시스템(RDBMS)

● 기본개념

- 관계데이터 모델은 데이터베이스를 테이블들의 집합으로 나타내며 테이블은 행과 열로 이루어져 있다. 테이블의 각 행(row)은 특정 목적에 따라 연관된 데이터 값들로 구성된다. 테이블간의 관계는 공통 열(column)을 통해서만 이루어지며 어떠한 링크도 존재하지 않는다.

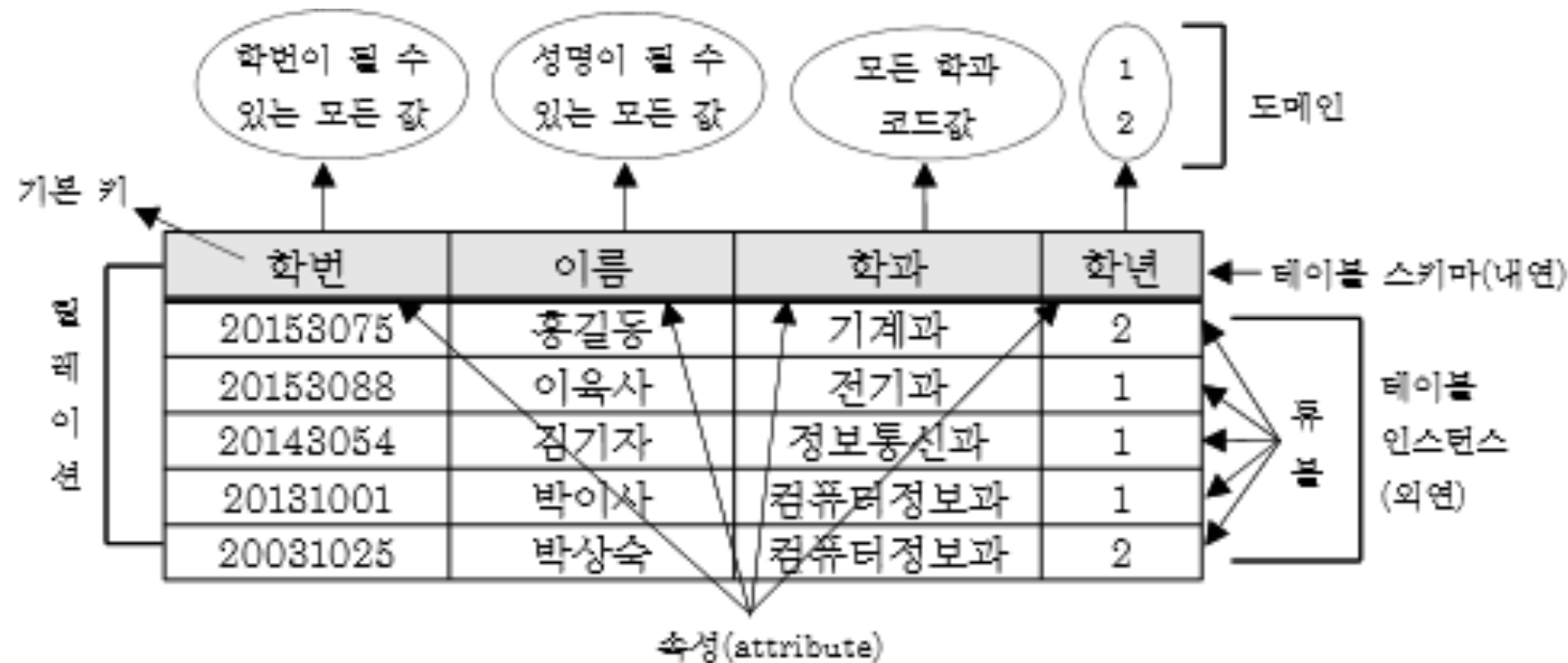


그림 1.2 관계데이터 모델의 예제

❖ 관계 데이터베이스 관리 시스템(RDBMS)

- 릴레이션(relation)

- 중앙집중식 데이터베이스, 분산식 데이터베이스

- 튜플(tuple)

- 테이블의 한 행(row)을 구성하는 <속성 이름, 값>쌍들의 집합으로 물리적인 용어로는 레코드(record)와 같다.

- 속성(attribute)

- 테이블의 각 열(column)을 의미하며 레코드 구조의 필드에 대응된다. 또한 속성(attribute)은 릴레이션이 갖는 성질(property)을 의미하며 관계데이터 모델에서 데이터의 가장 작은 논리적인 단위이다.

❖ 관계 데이터베이스 관리 시스템(RDBMS)

- 도메인(domain)

- 각 속성이 취할 수 있는 값의 집합으로 같은 타입이어야 한다.

- 단순 도메인(simple domain)

- 위에 정의된 도메인을 단순 도메인이라 하고, 이 단순 도메인으로 정의된 속성을 단순 속성이라 한다.

- 복합 도메인(composite domain)

- 단순 도메인들을 결합하여 구성된 도메인을 복합 도메인이라 하고, 이 복합 도메인으로 정의된 속성을 복합 속성이라 한다.

- 다치 속성(multivalued attribute)

- 단순 도메인들을 결합하여 구성된 도메인을 복합 도메인이라 하고, 이 복합 도메인으로 정의된 속성을 복합 속성이라 한다.

❖ 관계 데이터베이스 관리 시스템(RDBMS)

● 릴레이션 스키마(schema)

- 데이터베이스에 저장될 자료들의 논리적 구조 및 관계를 의미하고 릴레이션 이름과 속성들 그리고 스키마의 무결성 제약 조건으로 구성되며 시간에 무관한 정적 성질을 갖는 릴레이션의 영구 부분. 다른 이름으로 엔티티 유형(type), 릴레이션 유형, 릴레이션 내포(intension)라고도 부른다.

● 릴레이션 인스턴스(instance)

- 어느 한 시점에서 릴레이션이 포함하고 있는 전체 튜플을 의미하며 시간에 따라 변화하는 동적 성질을 갖고 있다. 엔티티 집합, 릴레이션 상태(state), 릴레이션 어커런스(occurrence), 릴레이션 외연(extension), 단순히 릴레이션이라고도 부른다.

● 릴레이션의 차수(degree)

- 한 릴레이션을 구성하는 속성 수를 의미한다.

● 카디널리티(cardinality)

- 특정 테이블의 튜플 개수를 의미한다.

❖ 관계 데이터베이스 관리 시스템(RDBMS)

릴레이션의 특징

- 튜플의 유일성

- 릴레이션의 인스턴스는 튜플들의 집합이고 집합은 중복된 원소를 포함하지 않으므로 릴레이션에는 중복된 튜플이 존재하지 않는다.

- 튜플의 무순서(위에서 아래로)

- 릴레이션에 있는 튜플들의 순서는 의미가 없다.

- 속성의 무순서(왼쪽에서 오른쪽으로)

- 릴레이션의 속성 사이의 순서는 의미가 없다.

- 속성의 원자 값

- 모든 속성의 값은 원자 값이다.

❖ 관계 데이터베이스 관리 시스템(RDBMS)

키(key)의 종류

● 슈퍼 키(super key)

- 테이블의 각 튜플을 유일하게 식별 할 수 있는 속성들의 조합으로 이루어진 키를 슈퍼 키라 한다. 두 개 이상의 속성으로 구성된 슈퍼 키 중에는 어떤 속성을 제거하더라도 각 튜플을 유일하게 식별 할 수 있다.

● 후보 키(candidate key)

- 속성 집합으로 구성된 테이블의 각 튜플을 유일하게 식별할 수 있는 속성이나 속성의 조합 들을 테이블의 후보 키라 하며. 속성의 조합으로 구성될 경우 어느 한 속성을 제거하면 튜플의 유일성을 잃게 되는 점이 슈퍼 키와 다르다.

● 기본 키(primary key)

- 한 테이블 내에서 각 튜플을 유일하게 식별 할 수 있는 속성 또는 속성의 조합으로 구성된 키로 후보 키 중 하나가 선택된다.
일반적으로 적은 개수의 속성으로 된 후보 키를 기본 키로 선택하는 것이 좋다.

❖ 관계 데이터베이스 관리 시스템(RDBMS)

● 대체 키(alternate key)

➢ 후보 키가 하나 이상일 때 그 중 하나를 기본 키로 지정하면 나머지 후보 키들은 대체 키가 된다.

● 외래 키(foreign key)

➢ 릴레이션 스키마 R의 어떤 속성이나 속성집합(외래 키(FK)라 가정)이 다른 릴레이션 스키마 S의 기본 키가 될 때, R의 속성 또는 속성집합을 외래 키라 한다, 이때 외래 키는 테이블 S를 참조(reference)한다고 하고, R에 있는 튜플들의 외래 키값은 S의 어떤 기본 키 값과 일치하거나 널을 가져야 한다.

❖ 관계 데이터베이스 관리 시스템(RDBMS)

제약조건

●도메인 제약 조건

- 각 속성 값은 반드시 해당 도메인에 속하는 원자 값이어야 한다는 조건

●키 제약 조건

- 릴레이션에는 릴레이션의 각 튜플을 유일하게 식별할 수 있는 수단 즉, 최소한 하나의 기본 키를 가지고 있어야 한다는 제약조건.

●무결성 제약조건

- 엔티티 무결성(entity integrity)

- 기본 키에 속해 있는 속성들의 값은 어떠한 경우에도 널 값을 가질 수 없다는 의미로서 만약 널 값을 갖게 되면 튜플을 유일하게 식별할 수 없게 된다.

- 참조 무결성(referential integrity)

- 한 테이블에 있는 튜플이 다른 테이블에 있는 튜플을 참조하려면 반드시 참조되는 튜플이 그 테이블 내에 존재해야 한다는 의미로 외래 키와 관련되어 있다.

❖ 데이터 모델과 데이터베이스의 발전사

● 데이터모델

- 현실세계의 정보들을 컴퓨터에 표현하기 위해 단순화, 추상화 형태로 체계적으로 표현할 수 있는 개념적 모형, 또한 데이터 모델을 이용하여 현실세계의 정보구조를 표현하려는 작업을 데이터 모델링(data modeling)이라고 한다.
- 계층형 데이터 모델, 네트워크(망)형 데이터 모델, 관계형 데이터 모델, 객체지향형 데이터 모델 등이 있다.
- 최근에는 관계형 데이터베이스 관리시스템과 객체 지향 데이터 모델의 장점을 기반으로 한 객체관계형 데이터베이스 관리 시스템(object-relational database management system)이 시장을 형성해 가고 있다.

❖ 데이터 모델과 데이터베이스의 발전사

● 파일시스템

➢ 응용 프로그램에서 여러 파일을 다룬다면 프로그램에서 각각의 파일과 연결 하여 처리하여야 함.

● 데이터베이스

➢ 기존의 파일 시스템에 비해 훨씬 효과적으로 데이터를 관리, 운용

➢ 파일 시스템과는 다르게 데이터베이스 관리시스템과의 연결 하나만으로 데이터베이스 내의 모든 데이터에 작업을 할 수 있다.

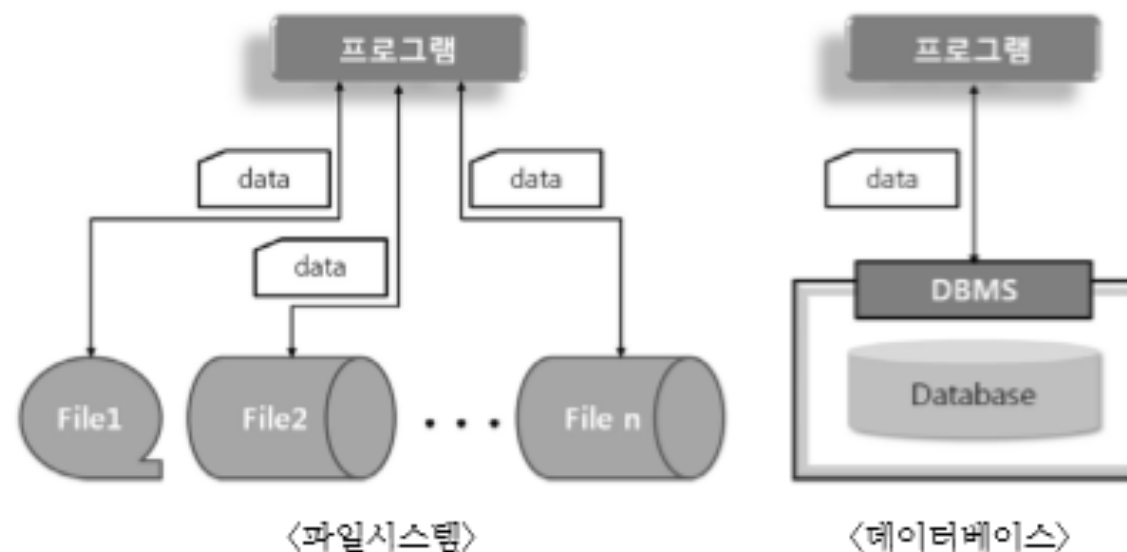


그림 1.3 데이터베이스와 파일시스템의 차이

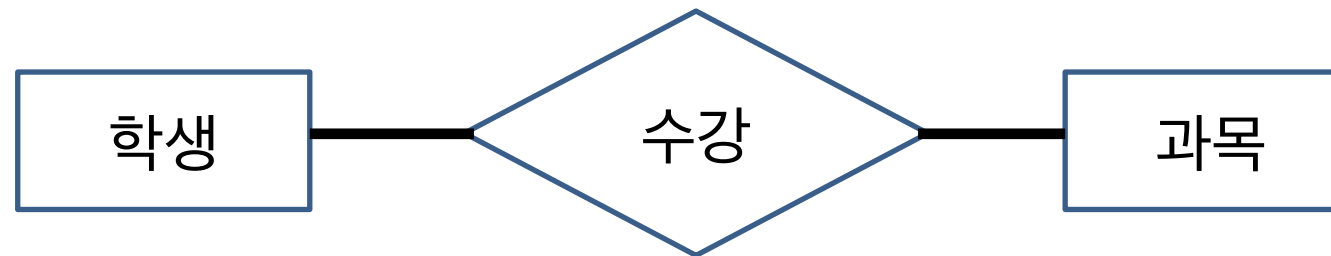
❖ 데이터 모델과 데이터베이스의 발전사

- 관계형 모델(Relation Model)

- 1970년 E.F.Codd에 의해 제안
- 데이터를 2차원 테이블 형태로 저장

❖ 데이터 모델과 데이터베이스의 발전사

● 관계형 모델(Relation Model)



학번	이름	학과	학년	과목번호	학번	점수
20153075	옥한빛	기계	1	101	20131001	80
20153088	이태연	기계	1	104	20131001	56
20143054	유가인	기계	2	106	20132003	72
20152088	조민우	전기전자	1	103	20152088	45
20142021	심수정	전기전자	2	101	20131025	65
20132003	박희철	전기전자	3	104	20131025	65
20151062	김인중	컴퓨터정보	1	108	20151062	81
20141007	진현무	컴퓨터정보	2	107	20143054	41
20131001	김종헌	컴퓨터정보	3	102	20153075	66
20131025	옥성우	컴퓨터정보	3	105	20153075	56
				102	20153088	61
				105	20153088	78

과목	과목이름	교수명	학년	학과
111	데이터베이스	이재영	2	컴퓨터정보
110	자동제어	정순정	2	전기전자
109	자동화설계	박민영	3	기계
101	컴퓨터개론	강종영	3	컴퓨터정보
102	기계공학법	김태영	1	기계
103	기초전자실험	김유석	1	전기전자
104	시스템분석설계	강석현	3	컴퓨터정보
105	기계요소설계	김명성	1	기계
106	전자회로실험	최영민	3	전기전자
107	CAD응용실습	구봉규	2	기계
108	소프트웨어공학	권민성	1	컴퓨터정보

❖ 데이터 모델과 데이터베이스의 발전사

● 오라클

- 가장 많이 쓰이는 데이터베이스 관리시스템이다.
- 1983년부터 트랜잭션 처리기능, 데이터의 일관성 처리 기능, 분산 처리 기능을 포함한 오라클 버전 3,4,5를 출시
- 1989년 PL/SQL 기능을 포함한 오라클 버전6 출시
- 1993년 오라클 버전7 참조무결성 기능, 저장 프로시저(stored procedure) 기능, 트리거 처리기능의 포함
- 1999년 객체지향 기능을 포함한 객체관계형 데이터베이스 관리시스템인 오라클 8버전이 출시, 추 후 다양한 기능이 보완된 오라클 8i 버전 출시
- 오라클 9i, 오라클 10g를 거쳐, 2007년에는 오라클 11g 버전을 출시
- 클라우드 컴퓨팅의 개념을 포함한 12c를 2013년 에 발표

❖ 목차

SELECT

조건 검색

검색 결과의 순서화 (정렬)

함수 (Function)

❖ SELECT

- 데이터를 검색하기 위한 명령어가 SELECT이며, 형식은 다음과 같다.

```
SELECT [DISTINCT] column-commalist  
FROM table-names  
[WHERE predicate]  
[GROUP BY column-commalist    [HAVING predicate]]  
[ORDER BY column-commalist]
```

❖ SELECT

- 테이블의 구조를 알기 위해 DESCRIBE 명령어를 사용한다.

SQL> describe student;

NAME	NULLABLE	TYPE	DEFAULT	COMMENT
STU_NO	NOT NULL	NUMBER(9)		
STU_NAME		VARCHAR2(12)		
STU_DEPT		VARCHAR2(20)		
STU_GRADE		NUMBER(1)		
STU_CLASS		CHAR(1)		
		CHAR(1)		
STU_GENDER		NUMBER(5,2)		
STU_HEIGHT				
STU_WEIGHT		NUMBER(5,2)		

❖ SELECT

- 테이블의 구조를 알기 위해 DESCRIBE 명령어를 사용한다.

SQL> describe enrol;

NAME	NULLABLE	TYPE	DEFAULT	COMMENT
SUB_NO	NOT NULL	CHAR(3)		
STU_NO	NOT NULL	NUMBER(9)		
ENR_GRADE		NUMBER(3)		

SQL> describe subject;

NAME	NULLABLE	TYPE	DEFAULT	COMMENT
SUB_NO	NOT NULL	CHAR(3)		
SUB_NAME		VARCHAR2(30)		
SUB_PROF		VARCHAR2(12)		
SUB_GRADE		NUMBER(1)		
SUB_DEPT		VARCHAR2(20)		

❖ SELECT

● 테이블에 있는 모든 데이터 검색

SQL> select *
2 from student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종현	컴퓨터정보	3	C	M		72
20131025	옥성우	컴퓨터정보	3	A	F	172	63

❖ SELECT

● 특정 열의 내용 검색

```
SQL> select stu_no, stu_name  
2 from student;
```

STU_NO	STU_NAME
20153075	옥한빛
20153088	이태연
20143054	유가인
20152088	조민우
20142021	심수정
20132003	박희철
20151062	김인중
20141007	진현무
20131001	김종헌
20131025	옥성우

❖ SELECT

● 중복 행 제거

```
SQL> select stu_dept  
2 from student;
```

STU_DEPT
기계
기계
기계
전기전자
전기전자
전기전자
컴퓨터정보
컴퓨터정보
컴퓨터정보
컴퓨터정보

```
SQL> select distinct stu_dept  
2 from student;
```

STU_DEPT
전기전자
기계
컴퓨터정보

❖ SELECT

● 중복 행 제거

SQL> select **distinct** stu_grade, stu_class
2 from student;

STU_GRADE	STU_CLASS
1	B
1	C
3	A
2	C
2	A
3	B
3	C

❖ SELECT

● 수식을 포함한 검색

**SQL> select stu_no, sub_no, enr_grade, enr_grade+10
2 from enrol;**

STU_NO	SUB_NO	ENR_GRADE	ENR_GRADE+10
20131001	101	80	90
20131001	104	56	66
20132003	106	72	82
20152088	103	45	55
20131025	101	65	75
20131025	104	65	75
20151062	108	81	91
20143054	107	41	51
20153075	102	66	76
20153075	105	56	66
20153088	102	61	71
20153088	105	78	88

❖ SELECT

● 결과열에 별칭(Alias) 부여하기

```
SQL> select stu_no as ID, stu_name as name  
2 from student;
```

ID	NAME
20153075	옥한빛
20153088	이태연
20143054	유가인
20152088	조민우
20142021	심수정
20132003	박희철
20151062	김인중
20141007	진현무
20131001	김종헌
20131025	옥성우

❖ SELECT

● 연결 연산자

**SQL> select stu_dept || stu_name as 학과성명
2 from student;**

학과성명
기계육한빛
기계이태연
기계유가인
전기전자조민우
전기전자심수정
전기전자박희철
컴퓨터정보김인중
컴퓨터정보진현무
컴퓨터정보김종헌
컴퓨터정보옥성우

❖ SELECT

● 연결 연산자

**SQL> select stu_dept || ',' || stu_name || '입니다' as 학과성명
2 from student;**

학과성명
기계,옥한빛입니다
기계,이태연입니다
기계,유가인입니다
전기전자,조민우입니다
전기전자,심수정입니다
전기전자,박희철입니다
컴퓨터정보,김인중입니다
컴퓨터정보,진현무입니다
컴퓨터정보,김종헌입니다
컴퓨터정보,옥성우입니다

❖ 조건 검색

● WHERE절 사용하기

- 일반적으로 비교연산자를 사용한다.
- 이때 사용되는 비교연산자는 =, <, >, <=, >=, <>의 6가지이다.
- 특히 '<>'는 '!=', '^='를 사용할 수도 있다.

SQL> select stu_name, stu_dept, stu_grade, stu_class

2 from student

3 where stu_dept = '컴퓨터정보';

STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS
김인중	컴퓨터정보	1	B
진현무	컴퓨터정보	2	A
김종헌	컴퓨터정보	3	C
옥성우	컴퓨터정보	3	A

❖ 조건 검색

● 논리 연산자

➤ 논리 연산자 **NOT, AND, OR**를 사용하여 여러 개의 조건을 결합하여 표현할 수 있다.

```
SQL> select stu_name, stu_dept, stu_grade, stu_class  
2  from student  
3  where stu_dept = '컴퓨터정보' and stu_grade = 2;
```

STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS
진현무	컴퓨터정보	2	A

❖ 조건 검색

● 범위조건

➤ WHERE절에 BETWEEN ~ AND을 사용하여 검색할 수 있다.

SQL> select *

2 from student

3 where stu_weight between 60 and 70;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131025	옥성우	컴퓨터정보	3	A	F	172	63

❖ 조건 검색

SQL> select *

2 from student

3 where stu_no **between** '20140001' **and** '20149999';

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20141007	진현무	컴퓨터정보	2	A	M	174	64
20142021	심수정	전기전자	2	A	F	168	45
20143054	유가인	기계	2	C	F	154	47

❖ 조건 검색

● LIKE를 이용한 검색

➢ 데이터의 일부를 알고 있을 경우, 와일드카드 문자와 함께 사용한다.

SQL> select stu_no, stu_name, stu_dept

2 from student

3 where stu_name like '김%';

STU_NO	STU_NAME	STU_DEPT
20151062	김인중	컴퓨터정보
20131001	김종헌	컴퓨터정보

● 오라클에서 LIKE와 같이 사용하는 와일드카드 문자는 다음과 같다.

기호	의 미
%	0개 이상의 문자
_	1개의 문자

❖ 조건 검색

```
SQL> select stu_no, stu_name, stu_dept  
2   from student  
3  where stu_name like '__수%';
```

STU_NO	STU_NAME	STU_DEPT
20142021	심수정	전기전자

```
SQL> select *  
2   from student  
3  where stu_no like '2014%';
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20143054	유가인	기계	2	C	F	154	47
20142021	심수정	전기전자	2	A	F	168	45
20141007	진현무	컴퓨터정보	2	A	M	174	64

❖ 조건 검색

● 널(NULL) 값 처리

➤ 널 값이 존재할 수 있으며, 이는 때때로 자료처리에 있어서 문제를 발생시킨다.

```
SQL> select stu_no, stu_name, stu_height  
2   from student;
```

STU_NO	STU_NAME	STU_HEIGHT
20153075	옥한빛	177
20153088	이태연	162
20143054	유가인	154
20152088	조민우	188
20142021	심수정	168
20132003	박희철	
20151062	김인중	166
20141007	진현무	174
20131001	김종헌	
20131025	옥성우	172

➤ 신장(stu_height)열에 널 값이 나타남.

❖ 조건 검색

● NULL값을 가지는 연산

```
SQL> select stu_name, stu_height/30.46  
2   from student;
```

STU_NAME	STU_HEIGHT/30.48
옥한빛	5.807086614
이태연	5.31496063
유가인	5.052493438
조민우	6.167979003
심수정	5.511811024
박희철	
김인중	5.446194226
진현무	5.708661417
김종헌	
옥성우	5.643044619

❖ 조건 검색

● 데이터에 널 값의 존재 여부 질의문

```
SQL> select stu_no, stu_name, stu_height  
2   from student  
3  where stu_height is null;
```

STU_NO	STU_NAME	STU_HEIGHT
20132003	박희철	
20131001	김종헌	

❖ 조건 검색

● 널 값이 아닌 행의 결과

```
SQL> select stu_no, stu_name, stu_height  
2   from student  
3  where stu_height is not null;
```

STU_NO	STU_NAME	STU_HEIGHT
20153075	옥한빛	177
20153088	이태연	162
20143054	유가인	154
20152088	조민우	188
20142021	심수정	168
20151062	김인중	166
20141007	진현무	174
20131025	옥성우	172

❖ 조건 검색

● IN 연산자

➢ 여러 개 조건 값 중 하나만 만족하는 행을 처리할 경우 사용한다.

```
SQL> select stu_no, stu_name  
2   from student  
3   where stu_dept in ('컴퓨터정보', '기계');
```

STU_NO	STU_NAME
20153075	옥한빛
20153088	이태연
20143054	유가인
20151062	김인중
20141007	진현무
20131001	김종헌
20131025	옥성우

stu_dept = '컴퓨터정보' or
stu_dept = '기계'

❖ 검색 결과의 순서화 (정렬)

● 정렬(SORT) : 데이터를 어떤 기준에 의해 나열하는 것

➤ 기준(key)

- 기본키(primary key)
- 보조키(secondary keys)

➤ 나열방법

- 오름차순(ascending) : 생략가능
- 내림차순(descending)

===> **ORDER BY**
컬럼이름 나열방법

==> 나열방법 : ASC(오름차순정렬, 생략가능)
DESC(내림차순정렬)

❖ 검색 결과의 순서화 (정렬)

```
SQL> select stu_no, stu_name  
2  from student  
3  order by stu_no;
```

STU_NO	STU_NAME
20131001	김종헌
20131025	옥성우
20132003	박희철
20141007	진현무
20142021	심수정
20143054	유가인
20151062	김인중
20152088	조민우
20153075	옥한빛
20153088	이태연

❖ 검색 결과의 순서화 (정렬)

- 학생들의 신상을 학번의 내림차순으로 검색하는 질의문

```
SQL> select stu_no, stu_name  
2   from student  
3   order by stu_no desc;
```

STU_NO	STU_NAME
20153088	이태연
20153075	옥한빛
20152088	조민우
20151062	김인중
20143054	유가인
20142021	심수정
20141007	진현무
20132003	박희철
20131025	옥성우
20131001	김종헌

❖ 검색 결과의 순서화 (정렬)

- 별칭이 붙어 있는 열을 기준으로 정렬하는 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target  
2  from student  
3  order by target;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20142021	심수정	전기전자	40
20143054	유가인	기계	42
20153088	이태연	기계	45
20132003	박희철	전기전자	58
20131025	옥성우	컴퓨터정보	58
20141007	진현무	컴퓨터정보	59
20151062	김인중	컴퓨터정보	62
20131001	김종헌	컴퓨터정보	67
20153075	옥한빛	기계	75
20152088	조민우	전기전자	85

❖ 검색 결과의 순서화 (정렬)

- 열의 순서번호를 이용하여 정렬하는 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target  
2  from student  
3  order by 4;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20142021	심수정	전기전자	40
20143054	유가인	기계	42
20153088	이태연	기계	45
20132003	박희철	전기전자	58
20131025	옥성우	컴퓨터정보	58
20141007	진현무	컴퓨터정보	59
20151062	김인중	컴퓨터정보	62
20131001	김종헌	컴퓨터정보	67
20153075	옥한빛	기계	75
20152088	조민우	전기전자	85

❖ 검색 결과의 순서화 (정렬)

- 산술식의 열의 이름을 이용하여 정렬하는 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target  
2  from student  
3  order by stu_weight-5;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20142021	심수정	전기전자	40
20143054	유가인	기계	42
20153088	이태연	기계	45
20132003	박희철	전기전자	58
20131025	옥성우	컴퓨터정보	58
20141007	진현무	컴퓨터정보	59
20151062	김인중	컴퓨터정보	62
20131001	김종헌	컴퓨터정보	67
20153075	옥한빛	기계	75
20152088	조민우	전기전자	85

❖ 검색 결과의 순서화 (정렬)

● 여러 개의 열을 기준으로 정렬하는 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target  
2  from student  
3  order by stu_dept, target;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20143054	유가인	기계	42
20153088	이태연	기계	45
20153075	옥한빛	기계	75
20142021	심수정	전기전자	40
20132003	박희철	전기전자	58
20152088	조민우	전기전자	85
20131025	옥성우	컴퓨터정보	58
20141007	진현무	컴퓨터정보	59
20151062	김인중	컴퓨터정보	62
20131001	김종현	컴퓨터정보	67

❖ 검색 결과의 순서화 (정렬)

● 오름차순과 내림차순이 혼용된 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target  
2  from student  
3  order by stu_dept, stu_weight-5 desc;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20153075	옥한빛	기계	75
20153088	이태연	기계	45
20143054	유가인	기계	42
20152088	조민우	전기전자	85
20132003	박희철	전기전자	58
20142021	심수정	전기전자	40
20131001	김종헌	컴퓨터정보	67
20151062	김인중	컴퓨터정보	62
20141007	진현무	컴퓨터정보	59
20131025	옥성우	컴퓨터정보	58

❖ 검색 결과의 순서화 (정렬)

- SELECT절에 포함되지 않는 열을 이용하여 정렬하는 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target  
2  from student  
3  order by stu_height;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20143054	유가인	기계	42
20153088	이태연	기계	45
20151062	김인중	컴퓨터정보	62
20142021	심수정	전기전자	40
20131025	옥성우	컴퓨터정보	58
20141007	진현무	컴퓨터정보	59
20153075	옥한빛	기계	75
20152088	조민우	전기전자	85
20132003	박희철	전기전자	58
20131001	김종헌	컴퓨터정보	67

❖ 함수(Function)

● 함수란 ?

- 하나 이상의 인수를 전달받아 처리한 결과 값을 함수의 이름을 변수처럼 활용하여 반환해 주는 프로그램 모듈
- 단일 행 함수는 함수가 정의된 SQL문장이 실행 될 때 각각의 행에 대해 수행되며 각 행별로 하나의 결과 값을 반환함
- 그룹함수는 데이터를 그룹화하고 그룹 각각에 대한 결과를 반환하며, GROUP BY 절을 사용함

❖ 함수(Function)

● 단일행 함수

➤ 인수의 데이터 타입에 따라

- ✓ 숫자함수
- ✓ 문자함수
- ✓ 날짜함수
- ✓ 형변환함수
- ✓ 일반함수

➤ 함수의 활용에서 함수는 **인수의 수** 및 각 **인수의 역할**이 중요함으로
인수를 바꾸어가며 충분한 실습이 필요함

❖ 함수(Function)

● 숫자 함수

➤ 숫자 인수를 사용하는 함수

함 수	설 명
ROUND(인수1,인수2)	인수1의 값을 인수2의 자리로 반올림하여 반환
TRUNC(인수1,인수2)	인수1의 값을 인수2 자리까지 유지하고, 나머지는 절삭하여 반환
MOD(인수1, 인수2)	인수1 값을 인수2 값으로 나눈 나머지를 반환
ABS(인수)	인수의 절대값은 반환
FLOOR(인수)	소숫점 이하 자리를 절삭하여 반환

❖ 함수(Function)

● ROUND 함수

```
SQL> select round(345.678), round(345.678, 0),  
2 round(345.678, 1), round(345.678, -1)  
3 from dual;
```

ROUND(345.678)	ROUND(345.678,0)	ROUND(345.678,1)	ROUND(345.678,-1)
346	346	345.7	350

❖ 함수(Function)

● 문자 함수

➤ 인수를 문자로 하는 함수

함 수	설 명
LOWER(인수)	인수1을 모두 소문자로 변환하여 반환
UPPER(인수)	인수1을 모두 대문자로 변환하여 반환
INITCAP(인수)	인수1 단어의 첫 번째 문자를 대문자로 변환하여 반환
CONCAT(인수1,인수2)	두개의 문자 인수를 연결하여 반환
SUBSTR(인수1,인수2,인수3,인수4)	문자열인수1의 일부분을 추출하여 반환
LENGTH(인수)	문자인수의 길이를 반환
INSTR(인수1,인수2,인수3,인수4)	문자인수 중 특정 문자의 절대위치를 반환
LPAD(인수1,인수2,인수3)	자릿수를 지정하고 빈 공간을 특정 문자로 왼쪽부터 채워서 문자열을 반환
RPAD(인수1,인수2,인수3)	자릿수를 지정하고 빈 공간을 특정 문자로 오른쪽부터 채워서 문자열을 반환

❖ 함수(Function)

● UPPER함수

```
SQL> select upper('korea')  
2 from dual;
```

UPPER('KOREA')

korea

❖ 함수(Function)

● 날짜 함수

➢ 기본 날짜 형식은 'DD-MON-RR'형식(RR형식 : 21세기와 20세기 데이터의 처리가 가능)

함 수	설 명
SYSDATE	시스템의 오늘 날짜를 반환
날짜 연산	날짜에 +, - 연산을 함
MONTHS_BETWEEN(인수1, 인수2)	인수1, 2의 날수 차이를 반환
NEXT_DAY(인수1,인수2)	인수1에서 가장 가까운 인수2의 요일을 반환
ADD_MONTH(인수1, 인수2)	인수1에 인수2의 달을 더하여 반환
LAST_DAY(인수1)	인수1이 속한 달의 마지막날을 반환
ROUND(인수1,인수2)	인수1의 값을 인수2를 기준으로 반올림하여 반환
TRUNC(인수1)	인수1의 값을 인수2를 기준으로 절사하여 반환

❖ 함수(Function)

**SQL> select sysdate
2 from dual;**

SYSDATE
2016-07-26 15:49

➤ SYSDATE 함수는 시스템의 현재 날짜를 반환한다.

**SQL> select next_day(sysdate, 'WED')
2 from dual;**

NEXT_DAY(SYSDATE, 'WED')
2016-07-27 16:09

- 결과에서처럼 현재의 날짜를 기준으로 다음에 오늘 수요일을 구하게 된다.
- 이때 요일에 해당하는 숫자를 이용할 수 있다.
- 일-1, 월-2, 화-3, 수-4, 목-5, 금-6, 토-7

❖ 함수(Function)

**SQL> select round(sysdate, 'MON')
2 from dual;**

ROUND(SYSDATE,'MON')

2016-08-01

- 날짜도 숫자와 같이 반올림을 할 수 있다.
- 예에서 보듯이 두 번째 인수를 MON으로 하였을 경우 달을 기준으로 반올림하여 결과를 보여준다.
- 두 번째 인수를 조절하면 년, 월, 일 등을 기준으로 반올림할 수 있다.

❖ 함수(Function)

● 변환 함수

➤ 데이터의 형을 변환함

함 수	내 용
TO_NUMBER	문자 데이터를 숫자 데이터로 변환
TO_DATE	문자 데이터를 날짜 데이터로 변환
TO_CHAR	숫자, 날짜 데이터를 문자 데이터로 변환

● TO_CHAR 함수

➤ TO_CHAR 함수는 주로 출력에 형식을 지정하기 위해 사용되며, 날짜형, 숫자형 모든 데이터에 사용한다.

❖ 함수(Function)

● TO_CHAR 함수

➤ TO_CHAR 함수는 주로 출력에 형식을 지정하기 위해 사용되며, 날짜형, 숫자형 모든 데이터에 사용한다.

```
SQL> select empno, ename,  
           to_char(hiredate, 'yyyy-mm') as 입사년월  
2  from emp;
```

EMPNO	ENAME	입사년월
7369	SMITH	1980-12
7499	ALLEN	1981-02
7521	WARD	1981-02
7566	JONES	1981-04
7654	MARTIN	1981-09
7698	BLAKE	1981-05
7782	CLARK	1981-06
7788	SCOTT	1987-04
7839	KING	1981-11
.....

❖ 함수(Function)

● TO_NUMBER 함수

➢ TO_NUMBER 함수는 숫자 형태의 문자를 숫자로 변환할 때 사용한다.

```
SQL> select to_char(to_number(1234.5678), '9999.999')  
2 from dual;
```

```
TO_CHAR(TO_NUMBER(1234.5678), '9999.999')
```

```
1234.568
```

```
SQL> select to_char(to_number(1234.5678), '999.999')  
2 from dual;
```

```
TO_CHAR(TO_NUMBER(1234.5678), '999.999')
```

```
#####
```

❖ 함수(Function)

● TO_DATE 함수

➤ TO_DATE 함수는 날짜 형태의 문자를 날짜로 변환할 때 사용한다.

SQL> select empno, ename

2 from emp

3 where hiredate = to_date('1980-12-17','yy-mm-dd');

EMPNO	ENAME
7369	SMITH

❖ 함수(Function)

- 일반 함수

- NVL 함수

➤ 인수 1이 널이면, 인수2를 아니면 인수를 반환함

```
SQL> select nvl(stu_height, 0)  
2 from student;
```

NVL(STU_HEIGHT,0)
177
162
154
188
168
0
166
174
0
172

❖ 함수(Function)

● NVL2 함수

➢ 인수 1이 널이 아니면 인수 2를 널이면 인수 3을 반환함

SQL> select ename, sal, comm, nvl2(comm, sal+comm, sal)
2 from emp;

ENAME	SAL	COMM	NVL2(COMM,SAL+COMM,SAL)
SMITH	800		800
ALLEN	1600	300	1900
WARD	1250	500	1750
JONES	2975		2975
MARTIN	1250	1400	2650
BLAKE	2850		2850
CLARK	2450		2450
SCOTT	3000		3000
KING	5000		5000
TURNER	1500	0	1500
ADAMS	1100		1100
JAMES	950		950
FORD	3000		3000
MILLER	1300		1300

❖ 함수(Function)

● NULLIF 함수

➤ 인수 1과 인수 2의 값을 비교하여 그 값이 같으면 NULL을 아니면 인수1의 값을 반환함

```
SQL> select nvl(nullif('A', 'A'), '널 값')  
2 from dual;
```

NVL(NULLIF('A','A'),'널 값')

널 값

❖ 함수(Function)

- CASE 함수 조건에 따른 처리

```
SELECT column-names  
  CASE WHEN condition-1 THEN statement-1,  
    WHEN condition-2 THEN statement-2,  
      .....  
    WHEN condition-n THEN statement-n,  
  ELSE statement  
END
```

❖ 함수(Function)

● CASE 함수

```

SQL> select empno, ename, sal,
2      case job when 'SALESMAN' then sal * 1.1
3            when 'CLERK' then sal * 1.15
4            when 'MANAGER' then sal * 1.2
5            else sal
6      end as 급여인상
7 from emp;
  
```

EMPNO	ENAME	SAL	급여인상
7369	SMITH	800	920
7499	ALLEN	1600	1760
7521	WARD	1250	1375
7566	JONES	2975	3570
7654	MARTIN	1250	1375
7698	BLAKE	2850	3420
7782	CLARK	2450	2940
7788	SCOTT	3000	3000
.....

❖ 함수(Function)

● DECODE 함수

```
DECODE ( column-name, condition-1, statement-1,  
                                condition-2, statement-2,  
                                .....  
                                condition-n, statement-n,  
                                statement)
```

❖ 함수(Function)

● DECODE 함수

```
SQL> select empno, ename, job, sal,
2  decode(job, 'SALESMAN', sal * 1.1,
3  'CLERK', sal * 1.15,
4  'MANAGER', sal * 1.2,
5  sal) as 인상된급여
6  from emp;
```

EMPNO	ENAME	JOB	SAL	인상된급여
7369	SMITH	CLERK	800	960
7499	ALLEN	SALESMAN	1600	1760
7521	WARD	SALESMAN	1250	1375
7566	JONES	MANAGER	2975	3867.5
7654	MARTIN	SALESMAN	1250	1375
7698	BLAKE	MANAGER	2850	3705
7782	CLARK	MANAGER	2450	3185
7788	SCOTT	ANALYST	3000	3000
7839	KING	PRESIDENT	5000	5000
.....

❖ 함수(Function)

● 그룹함수

- 여러 행에 대한 연산 즉 평균, 개수 등의 결과값을 반환하는 함수
- SELECT문에서 GROUP BY절을 사용함

● 그룹함수의 종류

함 수	기 능
COUNT()	조건을 만족하는 열의 데이터 값들의 개수를 반환
COUNT(*)	모든 행의 개수를 반환
SUM()	조건을 만족하는 열의 데이터 값들의 합을 반환
AVG()	조건을 만족하는 열의 데이터 값들의 평균을 반환
MAX()	조건을 만족하는 열의 데이터 값들 중 최댓값을 반환
MIN()	조건을 만족하는 열의 데이터 값들 중 최솟값을 반환
STDDEV()	조건을 만족하는 열의 데이터 값들의 표준편차를 반환
VARIANCE()	조건을 만족하는 열의 데이터 값들의 분산 값을 반환

❖ 함수(Function)

● MAX와 MIN함수

```
SQL> select max(enr_grade), min(enr_grade)
2  from enrol;
```

MAX(ENR_GRADE)	MIN(ENR_GRADE)
81	41

```
SQL> select min(stu_weight), max(stu_weight)
2  from student
3  where stu_dept = '기계';
```

MIN(STU_WEIGHT)	MAX(STU_WEIGHT)
47	80

❖ 함수(Function)

● COUNT 함수

```
SQL> select count(*), count(stu_height)
2 from student;
```

COUNT(*)	COUNT(STU_HEIGHT)
10	8

```
SQL> select count(stu_dept), count(distinct stu_dept)
2 from student;
```

COUNT(STU_DEPT)	COUNT(DISTINCTSTU_DEPT)
10	3

count() 사용할 때 입력데이터를 컬럼을 입력할 경우 컬럼의 데이터가 입력이 되지 않은 경우 (NULL인 경우)는 그 행은 카운트 하지 않는다.

❖ 함수(Function)

● SUM과 AVG 함수

```
SQL> select sum(stu_weight), to_char(avg(stu_weight), '9999.99')  
2   from student  
3   where stu_dept = '컴퓨터정보';
```

SUM(STU_WEIGHT)	TO_CHAR(AVG(STU_WEIGHT), '9999.99')
266	66.50

```
SQL> select count(*) as 학생, sum(stu_height) as 신장합,  
2   count(stu_height) "해당학생수", avg(stu_height) "평균신장"  
3   from student;
```

학생	신장합	해당학생수	평균신장
10	1361	8	170.125

❖ 함수(Function)

● 단일행을 이용한 GROUP BY절

```
SQL> select stu_dept, avg(stu_weight)
2  from student
3  group by stu_dept;
```

STU_DEPT	AVG(STU_WEIGHT)
전기전자	66
기계	59
컴퓨터정보	66.5

❖ 함수(Function)

● 단일행을 이용한 GROUP BY절

```
SQL> select stu_dept, count(*)  
2  from student  
3  where stu_weight >= 50  
4  group by stu_dept;
```

STU_DEPT	COUNT(*)
전기전자	2
기계	2
컴퓨터정보	4

❖ 함수(Function)

● 다중열 GROUP BY절

```
SQL> select stu_dept, stu_grade, count(*)  
2 from student  
3 group by stu_dept, stu_grade;
```

STU_DEPT	STU_GRADE	COUNT(*)
기계	2	1
기계	1	2
전기전자	1	1
컴퓨터정보	1	1
컴퓨터정보	3	2
컴퓨터정보	2	1
전기전자	2	1
전기전자	3	1

❖ 함수(Function)

● HAVING절 사용

➤ 그룹함수를 적용한 결과에 다시 조건을 부여할 때는 HAVING 절을 사용한다.

```
SQL> select stu_grade, avg(stu_height)
2  from student
3  where stu_dept = '기계'
4  group by stu_grade having avg(stu_height) >= 160;
```

STU_GRADE	AVG(STU_HEIGHT)
1	169.5

❖ 함수(Function)

● HAVING절 사용

```
SQL> select stu_dept, max(stu_height)
2  from student
3  group by stu_dept having max(stu_height) >= 175;
```

STU_DEPT	MAX(STU_HEIGHT)
전기전자	188
기계	177

```
SQL> select to_char(max(avg(stu_height)), '999.99')
2  from student
3  group by stu_dept;
```

TO_CHAR(MAX(AVG(STU_HEIGHT)), '999.99')
178.00

❖ 목차

데이터 갱신(Insert, Update, Delete)

TCL(Transaction Control Language)

❖ 데이터 갱신(Insert, Update, Delete)

● SQL의 데이터 조작성문(DML)

- 데이터 갱신 → UPDATE
- 데이터 삽입 → INSERT
- 데이터 삭제 → DELETE

● 트랜잭션 제어를 위한 TCL(Transaction Control Language)

- 트랜잭션 종료 → COMMIT
- 트랜잭션 작업 철회 → ROLLBACK

❖ 데이터 갱신(Insert, Update, Delete)

● INSERT

➤ 새로운 데이터를 테이블에 삽입하는 연산

```
INSERT  
    INTO table-name [(col-name, [, col-name] ... )]  
    VALUES (constant [, constant] ... );
```

또는

```
INSERT  
    INTO table-name [(col-name, [, col-name] ... )]  
    SELECT ... FROM ... WHERE ...;
```

❖ 데이터 갱신(Insert, Update, Delete)

- 단일 튜플 삽입

- 실습을 위해 a_enroll 테이블 생성

```
SQL> create table a_enrol  
2   as select *  
3   from enrol  
4   where stu_no < 20150000;
```

❖ 데이터 갱신(Insert, Update, Delete)

- a_enrol 테이블의 구조(desc) 및 데이터 확인(select)

SQL> desc a_enrol;

이름	널	유형
SUB_NO	NOT NULL	CHAR(3)
STU_NO		NUMBER(9)
ENR_GRADE		NUMBER(3)

SQL> select *

2 from a_enrol;

SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41

❖ 데이터 갱신(Insert, Update, Delete)

➤ a_enroll 테이블에 데이터 삽입

SQL> insert into a_enrol(sub_no, stu_no, enr_grade)

2 values (108, 20151062, 92);



1개의 행이 만들어 졌습니다.

SQL> insert into a_enrol **생략가능**
2 values (109, 20152088, 85);

1개의 행이 만들어 졌습니다.

❖ 데이터 갱신(Insert, Update, Delete)

➤ 일부 컬럼만 값이 존재하는 경우 생략 불가능

```
SQL> insert into a_enrol(sub_no, stu_no)
2 values ( 110, 20152088 );
```

1개의 행이 만들어 졌습니다.

➤ 테이블의 데이터 확인

```
SQL> select *
2 from a_enrol;
```

SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41
108	20151062	92
109	20152088	85
110	20152088	

❖ 데이터 갱신(Insert, Update, Delete)

➤ NULL값의 명시적 표현

```
SQL> insert into a_enrol  
2 values(111, 20153075, null);
```

1개의 행이 만들어 졌습니다.

➤ 테이블의 데이터 확인

```
SQL> select *  
2 from a_enrol;
```

SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41
108	20151062	92
109	20152088	85
110	20152088	
111	20153075	

❖ 데이터 갱신(Insert, Update, Delete)

➤ 복수 행 삽입

- 부질의의 결과를 테이블에 입력
- 부질의 결과의 열의 위치 및 수가 테이블과 일치하여야 함

SQL> insert into a_enrol

2 select * from enrol

3 where stu_no like '2015%';

6개의 행이 만들어 졌습니다.

SQL> select * from a_enrol

SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41
108	20151062	92
109	20152088	85
110	20152088	
111	20153075	
103	20152088	45
108	20151062	81
102	20153075	66
105	20153075	56
102	20153088	61
105	20153088	78

❖ 데이터 갱신(Insert, Update, Delete)

● UPDATE

➤ 데이터 값을 변경함

- 조건절을 만족하는 테이블내의 모든 데이터 변경
- 조건절이 없는 경우 테이블의 모든 데이터 변경
- WHERE절에 부질의 가능

UPDATE table-name

SET col-name = expression, [col-name = expression]

[WHERE predicate]

❖ 데이터 갱신(Insert, Update, Delete)

➤ 전체 데이터에 대한 변경

```
SQL> select * from a_enrol;
```

```
SQL> update a_enrol  
2   set enr_grade = enr_grade + 5;
```

16행이 갱신되었습니다.

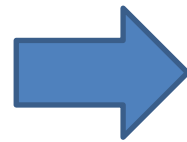
```
SQL> select * from a_enrol;
```

❖ 데이터 갱신(Insert, Update, Delete)

➤ 갱신 전후에 a_enrol 테이블의 데이터 확인

SQL> select * from a_enrol;

SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41
108	20151062	92
109	20152088	85
110	20152088	
111	20153075	
103	20152088	45
108	20151062	81
102	20153075	66
105	20153075	56
102	20153088	61
105	20153088	78



SUB_NO	STU_NO	ENR_GRADE
101	20131001	85
104	20131001	61
106	20132003	77
101	20131025	70
104	20131025	70
107	20143054	46
108	20151062	97
109	20152088	90
110	20152088	
111	20153075	
103	20152088	50
108	20151062	86
102	20153075	71
105	20153075	61
102	20153088	66
105	20153088	83

❖ 데이터 갱신(Insert, Update, Delete)

➤ 조건에 맞는 데이터 변경

```
SQL> select * from a_enrol;
```

```
SQL> update a_enrol  
2   set enr_grade = enr_grade + 10  
3   where sub_no = 104 ;
```

2행이 갱신되었습니다.

```
SQL> select * from a_enrol;
```

❖ 데이터 갱신(Insert, Update, Delete)

➤ 부질의를 갖는 UPDATE문

```
SQL> update a_enrol  
2   set enrol_grade = enr_grade + 10  
3   where sub_no = (select sub_no  
4                   from subject  
5                   where sub_name = '시스템 분석설계');
```

2행이 갱신되었습니다.

❖ 데이터 갱신(Insert, Update, Delete)

● DELETE

➤ 데이터를 삭제하는 연산

- 조건절을 만족하는 테이블내의 모든 데이터 삭제
- 조건절이 없는 경우 테이블의 모든 데이터 삭제
- WHERE절에 부질의 가능

```
DELETE
```

```
FROM table-name
```

```
[WHERE predicate]
```

❖ 데이터 갱신(Insert, Update, Delete)

➤ 특정 튜플 삭제

```
SQL> delete from a_enrol  
2   where stu_no = 20131001;
```

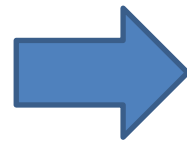
2행이 삭제되었습니다.

❖ 데이터 갱신(Insert, Update, Delete)

➤ 삭제 전후에 a_enrol 테이블의 데이터 확인

SQL> select * from a_enrol;

SUB_NO	STU_NO	ENR_GRADE
101	20131001	80
104	20131001	56
106	20132003	72
101	20131025	65
104	20131025	65
107	20143054	41
108	20151062	92
109	20152088	85
110	20152088	
111	20153075	
103	20152088	45
108	20151062	81
102	20153075	66
105	20153075	56
102	20153088	61
105	20153088	78



SUB_NO	STU_NO	ENR_GRADE
106	20132003	77
101	20131025	70
104	20131025	70
107	20143054	46
108	20151062	97
109	20152088	90
110	20152088	
111	20153075	
103	20152088	50
108	20151062	86
102	20153075	71
105	20153075	61
102	20153088	66
105	20153088	83

❖ 데이터 갱신(Insert, Update, Delete)

➤ 테이블내의 모든 데이터가 삭제

```
SQL> delete from a_enrol
```

14행이 삭제되었습니다.

❖ TCL(Transaction Control Language)

● 트랜잭션

- 트랜잭션이란 사용자에게 의해 실행된 SQL문의 집합
- 변경된 데이터는 TCL에 의해 데이터베이스에 반영됨
- 트랜잭션 처리는 데이터 무결성(Integrity) 유지
- DML문의 한 번 이상 실행이 하나의 트랜잭션이 되며,
- DDL문은 하나의 명령이 하나의 트랜잭션이 된다.

❖ TCL(Transaction Control Language)

➤ 실습에 앞서 a_student, b_student 테이블 생성하고, 데이터 확인(P. 141)

```
SQL> create table a_student
2  select *
3  from student
4  where stu_dept in ('기계', '전기전자');
```

```
SQL> select *
2  from a_student;
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20142021	심수정	전기전자	2	A	F	168	45
20152088	조민우	전기전자	1	C	M	188	90
20143054	유가인	기계	2	C	F	154	47
20153088	이태연	기계	1	C	F	162	50
20153075	옥한빛	기계	1	C	M	177	80

❖ TCL(Transaction Control Language)

➤ 실습에 앞서 b_student 테이블 생성하고, 데이터 확인

```
SQL> create table b_student
2   as select *
3   from student
4   where stu_detp in ('전기전자', '컴퓨터정보');
```

```
SQL> select *
2   from b_student;
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종현	컴퓨터정보	3	C	M		72
20131025	옥성우	컴퓨터정보	3	A	F	172	63

❖ TCL(Transaction Control Language)

➤ b_student의 내용을 삭제

```
SQL> delete from b_student;
```

7행이 삭제되었습니다.

```
SQL> select * frm b_student;
```

선택된 레코드가 없습니다.

```
SQL> rollback;
```

롤백이 완료되었습니다.

❖ TCL(Transaction Control Language)

● b_student 테이블의 내용을 검색

SQL> select * from b_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종헌	컴퓨터정보	3	C	M		72
20131025	옥성우	컴퓨터정보	3	A	F	172	63

❖ TCL(Transaction Control Language)

➤ DDL명령을 통한 자동 COMMIT

```
SQL> delete from b_student;
```

7행이 삭제되었습니다.

➤ 테이블을 생성하는 DDL명령어를 실행

```
SQL> create table c_student  
2 (stu_no number,  
3 stu_ame char(10));
```

테이블이 생성되었습니다.

```
SQL> select * from b_student;
```

❖ TCL(Transaction Control Language)

- 비정상적인 작업의 종료
- a_student 테이블의 내용을 검색

SQL> select * from a_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20142021	심수정	전기전자	2	A	F	168	45
20152088	조민우	전기전자	1	C	M	188	90
20143054	유가인	기계	2	C	F	154	47
20153088	이태연	기계	1	C	F	162	50
20153075	옥한빛	기계	1	C	M	177	80

❖ TCL(Transaction Control Language)

➤ a_student 테이블의 모든 데이터를 삭제

SQL> delete from a_student;

10행이 삭제되었습니다.

➤ a_student 테이블의 데이터가 삭제된 것을 검색

SQL> select * from a_student;

선택된 레코드가 없습니다.

❖ TCL(Transaction Control Language)

➤작업관리자를 이용하여 SQL Plus 프로세스를 비정상적으로 종료

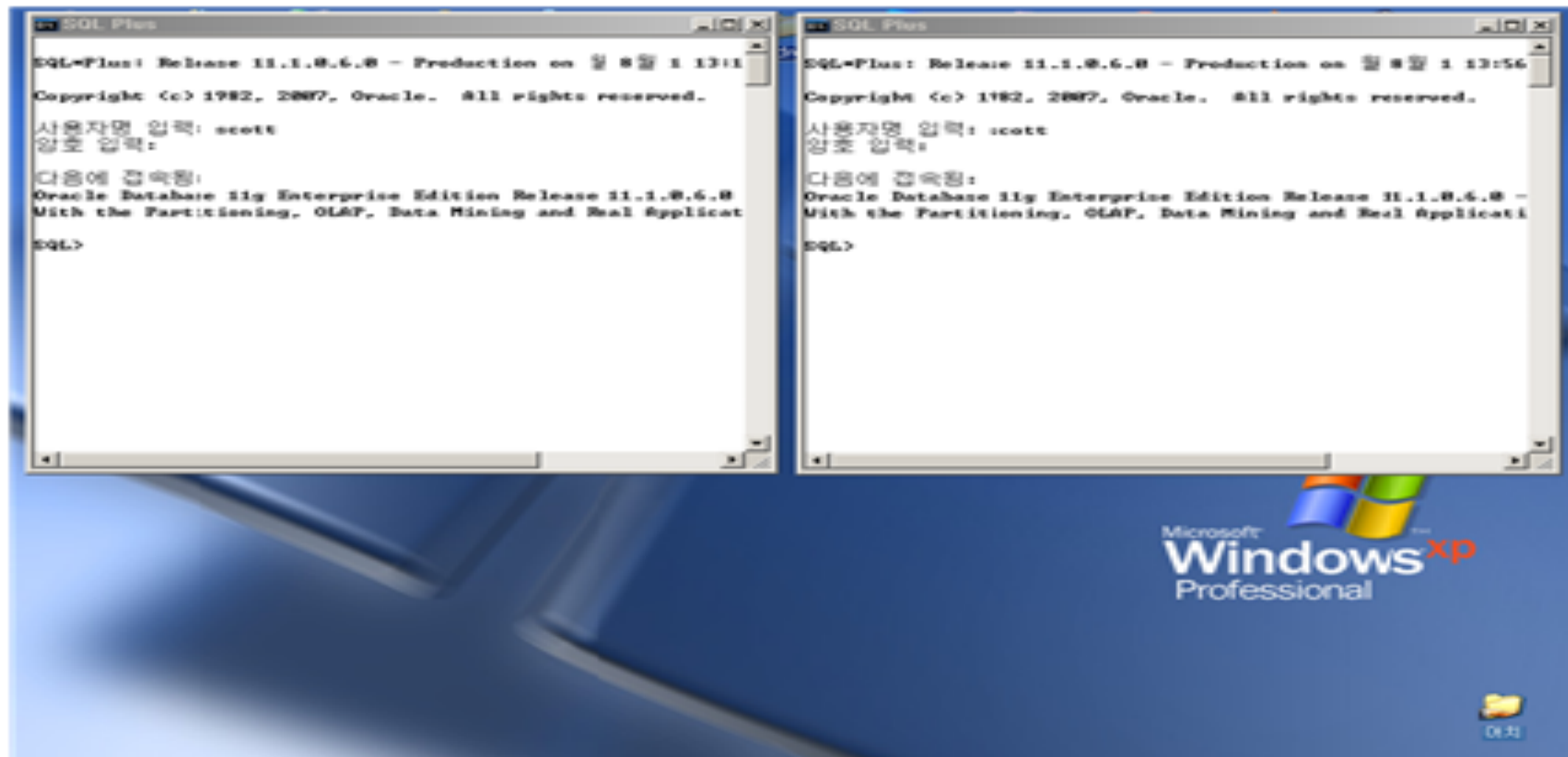
SQL> select * from a_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20142021	심수정	전기전자	2	A	F	168	45
20152088	조민우	전기전자	1	C	M	188	90
20143054	유가인	기계	2	C	F	154	47
20153088	이태연	기계	1	C	F	162	50
20153075	옥한빛	기계	1	C	M	177	80

❖ TCL(Transaction Control Language)

● 병행처리

➢ 실습을 위해 SQLPlus를 두번 실행함.



❖ TCL(Transaction Control Language)

- 각각의 화면에 다음과 같이 시간순서를 고려하여 입력하고 실행

시간	프로그램1	프로그램2
①	SQL> select * 2 from a_sudent;	
②	SQL> insert 2 into a_student(stu_no, stu_name) 3 values(10,'홍');	
③	SQL> select * 2 from a_student;	
④		SQL> select * 2 from a_student;
⑤	SQL> commit;	
⑥		SQL> select * 2 from a_student;

조인(Join)

부질의(SubQuery)

집합(SET)연산자

❖ 조인(Join)

● 조인(Join)

- 상호 특성 관련성을 갖는 두 개 이상의 테이블로부터 새로운 테이블을 생성하는데 사용되는 연산
- 두 개 이상의 테이블을 “조인(join)” 할 수 있는 기능은 관계시스템이 갖는 가장 강력한 특징.
- 정규화를 거친 데이터베이스의 각 테이블은 분리되어 관련된 다른 테이블과의 결합이 필요하며 이런 경우 관련 열의 관계성을 유도하여 결합됨

● 오라클에서 조인의 종류

- Cross 조인
- Equi 조인
- Non-Equi 조인
- Self 조인
- Outer 조인

- 오라클에서만 사용하는 JOIN과 모든 데이터베이스에서 사용할 수 있는 ANSI JOIN이 있다.

❖ 조인(Join)

● CROSS 조인

- 크로스 조인(cross join)은 관계시스템의 관계 대수 8가지 연산 중 카티션 프로덕트(**Cartesian Product**)를 구현함
- 2개 이상의 테이블을 조건 없이 실행되는 조인 연산

```
SQL> select student.*, enrol.*
2    from student, enrol;
```

Student X enroll
10 12 \rightarrow 120

[illegible]

❖ 조인(Join)

● EQUI 조인

- 2개 이상의 테이블에 **관련 있는 공통 열**의 값을 이용하여 논리적으로 결합하는 연산이 수행되는 조인

```
SQL> select student.stu_no, stu_name, stu_dept, enr_grade
2   from student, enrol
3  where student.stu_no = enrol.stu_no;
```

STU_NO	STU_NAME	STU_DEPT	ENR_GRADE
20131001	김종헌	컴퓨터정보	80
20131001	김종헌	컴퓨터정보	56
20132003	박희철	전기전자	72
20152088	조민우	전기전자	45
20131025	옥성우	컴퓨터정보	65
20131025	옥성우	컴퓨터정보	65
20151062	김인중	컴퓨터정보	81
20143054	유가인	기계	41
.....

❖ 조인(Join)

- 101번 과목을 수강하는 학생들의 학번과 이름 검색

SQL> select student.stu_no, stu_name

2 from student, enrol

3 where student.stu_no = enrol.stu_no and

4 sub_no = 101; ←

sub_no = 101 **or** sub_no = 102

STU_NO	STU_NAME
20131001	김종현
20131025	옥성우

❖ 조인(Join)

- 과목번호 101 또는 102를 수강하는 학생의 학번과 이름 검색

```
SQL> select student.stu_no, stu_name  
2   from student, enrol  
3   where student.stu_no = enrol.stu_no  
4   and sub_no = 101 or sub_no = 102;
```

STU_NO	STU_NAME
20153075	옥한빛
20153088	이태연
20143054	유가인
20152088	조민우
20142021	심수정
20132003	박희철
.....
20131025	옥성우
20131001	김종현
20131025	옥성우

❖ 조인(Join)

● 논리 연산자의 우선순위

```
SQL> select student.stu_no, stu_name  
2   from student, enrol  
3   where student.stu_no = enrol.stu_no  
4   and (sub_no = 101 or sub_no = 102);
```

STU_NO	STU_NAME
20153075	옥한빛
20153088	이태연
20131001	김종헌
20131025	옥성우

❖ 조인(Join)

● 3 테이블 조인

➤ '컴퓨터개론' 과목을 수강하는 학생들의 학번, 이름, 과목이름 검색

```
SQL> select student.stu_no, stu_name, sub_name  
2   from student, enrol, subject  
3   where student.stu_no = enrol.stu_no  
4   and enrol.sub_no = subject.sub_no  
5   and enrol.sub_no = 101;
```

STU_NO	STU_NAME	SUB_NAME
20131001	김종헌	컴퓨터개론
20131025	옥성우	컴퓨터개론

❖ 조인(Join)

● Non-Equi 조인

➢ WHERE 절에서 사용하는 '='이 아닌 연산자를 사용

```
SQL> select empno, ename, sal, grade  
2   from emp, salgrade  
3   where sal between losal and hisal;
```

sal >= losal and sal <= hisal

EMPNO	ENAME	SAL	GRADE
7900	JAMES	950	1
7369	SMITH	800	1
7876	ADAMS	1100	1
7521	WARD	1250	2
7654	MARTIN	1250	2
7934	MILLER	1300	2
7499	ALLEN	1600	3
7844	TURNER	1500	3
.....

❖ 조인(Join)

● SELF 조인

- 같은 테이블 간의 조인, 테이블의 별칭을 사용함
- 다음은 자신의 상급자를 구하는 질의문

SQL> select a.empno as 사원번호, a.ename as 사원이름,
 2 b.empno as 상급자사원번호, b.ename as 상급자이름
 3 from emp a, emp b
 4 where a.mgr = b.empno → JOIN ~ ON

사원번호	사원이름	상급자사원번호	상급자이름
7369	SMITH	7902	FORD
7499	ALLEN	7698	BLAKE
7521	WARD	7698	BLAKE
7566	JONES	7839	KING
.....

❖ 조인(Join)

● Outer 조인

➢ OUTER JOIN은 조인 조건을 만족하지 않는 행들도 질의 결과에 포함함

➢ 수강(enrol) 테이블을 기준으로 과목이름 검색

```
SQL> select a.*, sub_name
2   from enrol a, subject b
3   where a.sub_no = b.sub_no
4   order by 1;
```

*과목번호 109 ~ 111은 수강생이 없어
조인에서 제외됨

SUB_NO	SUB_NAME
109	자동화설계
110	자동제어
111	데이터베이스

SUB_NO	STU_NO	ENR_GRADE	SUB_NAME
101	20131001	80	컴퓨터개론
101	20131025	65	컴퓨터개론
102	20153075	66	기계공작법
102	20153088	61	기계공작법
103	20152088	45	기초전자실험
104	20131001	56	시스템분석설계
104	20131025	65	시스템분석설계
105	20153088	78	기계요소설계
105	20153075	56	기계요소설계
106	20132003	72	전자회로실험
107	20143054	41	CAD응용실습
108	20151062	81	소프트웨어공학

❖ ANSI 조인(ANSI Join)

● CROSS 조인

- 크로스 조인(cross join)은 관계시스템의 관계 대수 8가지 연산 중 카티션 프로덕트(**Cartesian Product**)를 구현함
- 2개 이상의 테이블을 조건 없이 실행되는 조인 연산

```
SQL> select student.*, enrol.*
2    from student cross join enrol;
```

Student X enroll
10 12 \rightarrow 120

[illegible]

❖ ANSI 조인(ANSI Join)

● NATURAL JOIN

```
SQL> select stu_no, stu_name, stu_dept, enr_grade  
2   from student natural join enrol;
```

➢ 두 테이블에 같은 열의 이름이 2쌍 이상 존재하면 사용 못함.

● JOIN ~ USING

```
SQL> select stu_no, stu_name, stu_dept, enr_grade  
2   from student join enrol using(stu_no);
```

● JOIN ~ ON

```
SQL> select stu_no, stu_name, stu_dept, enr_grade  
2   from student join enrol on student.stu_no = enrol.stu_no ;
```

❖ ANSI 조인(ANSI Join)

```
SQL> select stu_no, stu_name  
2   from student natural join enrol  
3   where sub_no = 101 or sub_no = 102;
```

```
SQL> select stu_no, stu_name  
2   from student join enrol using(stu_no)  
3   where sub_no = 101 or sub_no = 102;
```

```
SQL> select student.stu_no, stu_name  
2   from student join enrol on student.stu_no = enrol.stu_no  
3   where sub_no = 101 or sub_no = 102;
```

❖ ANSI 조인(ANSI Join)

SQL> select a.*, sub_name

2 from enrol a **right outer join** subject b

3 on a.sub_no = b.sub_no

4 order by 1;

✓ RIGTH OUTER JOIN

✓ LEFT OUTER JOIN

✓ FULL OUTER JOIN

SUB_NO	STU_NO	ENR_GRADE	SUB_NAME
101	20131001	80	컴퓨터개론
101	20131025	65	컴퓨터개론
102	20153088	61	기계공작법
102	20153075	66	기계공작법
103	20152088	45	기초전자실험
104	20131025	65	시스템분석설계
104	20131001	56	시스템분석설계
105	20153075	56	기계요소설계
105	20153088	78	기계요소설계
106	20132003	72	전자회로실험
107	20143054	41	CAD응용실습
108	20151062	81	소프트웨어공학
			데이터베이스
			자동화설계
			자동제어

❖ ANSI 조인(ANSI Join)

● SELF 조인과 OUTER 조인의 결합

SQL>select a.empno as 사원번호, a.ename as 사원이름,
 2 b.empno as 상급자사원번호, b.ename as 상급자이름
 3 from emp a left outer join emp b on a.mgr = b.empno

사원번호	사원이름	상급자사원번호	상급자이름
7782	CLARK	7839	KING
7698	BLAKE	7839	KING
7566	JONES	7839	KING
.....
7499	ALLEN	7698	BLAKE
7934	MILLER	7782	CLARK
7876	ADAMS	7788	SCOTT
7369	SMITH	7902	FORD
7839	KING		

❖ 부질의(SubQuery)

● 부질의(SubQuery)

- SELECT문내에 또 다른 SELECT문을 포함함(부질의)
- '옥성우' 학생보다 신장이 큰 학생들의 학번, 이름 신장 검색

```
SQL>select stu_height  
2  from student  
3  where stu_name = '옥성우';
```

STU_HEIGHT
172

```
SQL>select stu_height  
2  from student  
3  where stu_height > 172;
```

STU_HEIGHT
177
188
174

❖ 부질의(SubQuery)

- 하나의 질의문으로 처리함

```
SQL> select stu_no, stu_name, stu_height  
2   from student  
3   where stu_height >  
4   (select stu_height  
5   from student  
6   where stu_name = '옥성우');
```

STU_NO	STU_NAME	STU_HEIGHT
20153075	옥한빛	177
20152088	조민우	188
20141007	진현무	174

- WHERE절, HAVING절, FROM절

❖ 부질의(SubQuery)

- SELF 조인 이용하여 재 작성함

```
SQL> select a.stu_no, a.stu_name, a.stu_height  
2   from student a, student b  
3   where a.stu_height > b.stu_height  
4   and b.stu_name = '옥성우';
```

STU_NO	STU_NAME	STU_HEIGHT
20153075	옥한빛	177
20152088	조민우	188
20141007	진현무	174

❖ 부질의(SubQuery)

● 단일열 부질의

- 열의 값 한 개를 반환하는 부질의
- 박희철 학생과 같은 체중을 가지고 있는 학생의 정보 검색

SQL> select *

2 from student

3 where stu_weight =

4 (select stu_weight

5 from student

6 where stu_name = '박희철');

STU_WEIGHT

63

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20131025	옥성우	컴퓨터정보	3	A	F	172	63

❖ 부질의(SubQuery)

```
SQL> select *  
2   from student  
3   where stu_weight =  
4     (select stu_weight  
5        from student  
6        where stu_name = '박희철')  
7   and stu_name <> '박희철';
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20131025	옥성우	컴퓨터정보	3	A	F	172	63

❖ 부질의(SubQuery)

- 부질의 결과가 다중일 경우(IN 연산자 사용)
- '컴퓨터정보'과 학생과 같은 반을 가진 학생의 정보 검색

SQL> select *

2 from student


3 where stu_class in

4 (select distinct stu_class

5 from student

6 where stu_dept = '컴퓨터정보')

7 and stu_dept <> '컴퓨터정보';



STU_CLASS
B
A
C

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20142021	심수정	전기전자	2	A	F	168	45
20152088	조민우	전기전자	1	C	M	188	90
20143054	유가인	기계	2	C	F	154	47
20153088	이태연	기계	1	C	F	162	50
20153075	옥한빛	기계	1	C	M	177	80

❖ 부질의(SubQuery)

➤ 전체 학생들의 평균신장 보다 큰 학생의 정보 검색

SQL> select *

2 from student

3 where stu_height >

4 (select avg(stu_height)

5 from student);

STU_HEIGHT

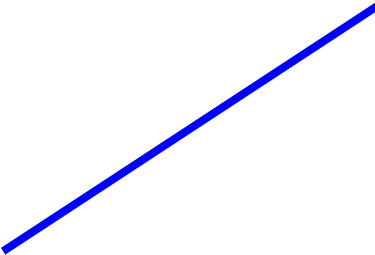
170.125

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20152088	조민우	전기전자	1	C	M	188	90
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131025	옥성우	컴퓨터정보	3	A	F	172	63

❖ 부질의(SubQuery)

➤ 신장이 모든 학과들의 평균 신장보다 큰 학생의 정보 검색

```
SQL> select *  
2   from student  
3   where stu_height > all  
4     (select avg(stu_height)  
5      from student  
6      group by stu_dept);
```



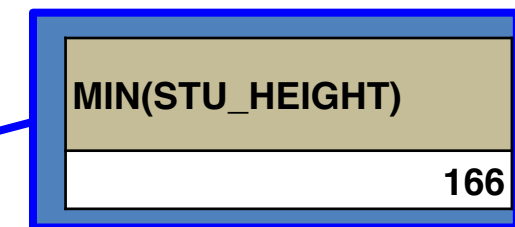
STU_HEIGHT
178
164.3333333
170.6666667

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20152088	조민우	전기전자	1	C	M	188	90

❖ 부질의(SubQuery)

➤ 컴퓨터정보과의 최소 신장보다 최소 신장이 더 큰 학과의 학과명과 최소 신장 검색

```
SQL> select stu_dept, min(stu_height)
2   from student
3  group by stu_dept having min(stu_height) >
4   (select min(stu_height)
5    from student
6   where stu_dept = '컴퓨터정보');
```



MIN(STU_HEIGHT)
166

STU_DEPT	MIN(STU_HEIGHT)
전기전자	168

❖ 부질의(SubQuery)

- 일반적으로 **부질의**의 질의문을 **조인** 질의로 표현 가능

➢ 101번 과목을 수강한 학생들의 정보 검색

```
SQL> select *  
2   from student  
3   where stu_no in  
4       (select stu_no  
5          from enrol  
6          where sub_no = 101);
```

||

```
SQL> select *  
2   from student a, enrol b  
3   where a.stu_no = b.stu_no and b.sub_no = 101;
```

❖ 부질의(SubQuery)

- 101번 과목을 수강한 학생들의 학번, 이름, 점수를 검색

```
SQL> select a.stu_no, a.stu_name, b.enr_grade  
2   from student a, enrol b  
3   where a.stu_no = b.stu_no and b.sub_no = 101;
```

STU_NO	STU_NAME	ENR_GRADE
20131001	김종헌	80
20131025	옥성우	65

- 부질의로 표현 불가

❖ 부질의(SubQuery)

- 복수열 부질의

- 복수열 값을 반환하는 부질의

- 실습을 위한 사원(emp) 테이블과 구조가 같은 test테이블 생성

```
SQL> create table test(empno, ename, sal, comm, deptno)
2  as
3      select empno, ename, sal, comm, deptno
4      from emp
5      where deptno = 1;
```

```
SQL> insert into test values(11, 'apple', 1000, null, 30);
```

```
SQL> insert into test values(12, 'banana', 2000, 100, 30);
```

```
SQL> insert into test values(13, 'cheese', 1000, 0, 10);
```

```
SQL> insert into test values(14, 'dog', 2000, null, 20);
```

```
SQL> insert into test values(15, 'egg', 1000, 100, 20);
```


❖ 부질의(SubQuery)

● test 테이블 데이터 검색

```
SQL> select *  
2 from test;
```

EMPNO	ENAME	SAL	COMM	DEPTNO
11	apple	1000		30
12	banana	2000	100	30
13	cheese	1000	0	10
14	dog	2000		20
15	egg	1000	100	20

❖ 부질의(SubQuery)

➤ 부질의 결과값이 복수열

SQL> select *

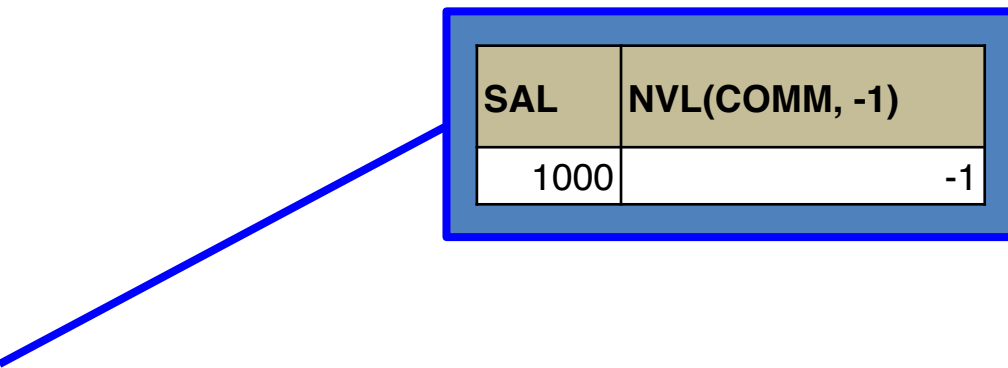
2 from test

3 where (sal, nvl(comm, -1)) =

4 (select sal, nvl(comm,-1)

5 from test

6 where empno = 11);



SAL	NVL(COMM, -1)
1000	-1

EMPNO	ENAME	SAL	COMM	DEPTNO
11	apple	1000		30

❖ 부질의(SubQuery)

➤ 부질의 결과값이 복수열, 복수행

```
SQL> select *  
2  from test  
3  where (sal, nvl(comm, -1)) in (select sal, nvl(comm, -1)  
4  
5      from test  
      where deptno = 30);
```

EMPNO	ENAME	SAL	COMM	DEPTNO
11	apple	1000		30
12	banana	2000	100	30


SAL	NVL(COMM,-1)
1000	-1
2000	100


❖ 부질의(SubQuery)

● FROM 절의 부질의(In-Line 뷰)

➢ 학생들의 학과별 평균 신장보다 큰 신장의 학생들 정보 검색

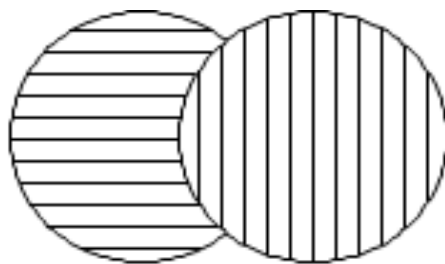
```
SQL> select stu_dept, round(avg(stu_height),2) as avg_height  
2   from student  
3   group by stu_dept;
```

```
SQL> select stu_no, stu_name, a.stu_dept, stu_height, avg_height  
2   from student a,  
3    ) b  
4  
5 where a.stu_dept = b.stu_dept and stu_height > avg_height;
```

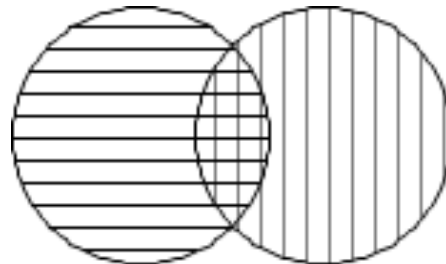


❖ 집합(SET)

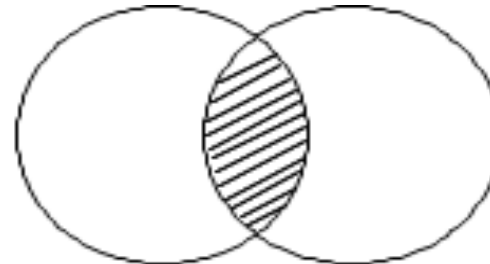
연산자	설 명
UNION	두 질의 결과값의 합으로 중복을 제거됨
UNION ALL	두 질의 결과값의 합으로 중복을 포함됨
INTERSECT	두 질의 결과값의 공통되는 값
MINUS	첫 번째 질의 결과에서 두 번째 질의 결과에 있는 행을 제거한 값



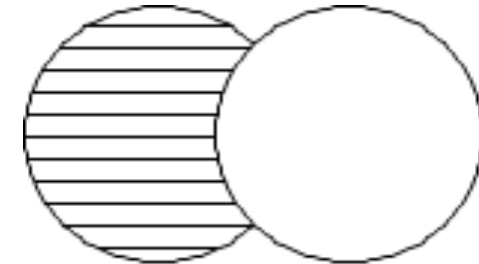
UNION



UNION ALL



INTERSECT



MINUS

❖ 집합(SET)

- 집합 연산자 사용 규칙은 다음과 같다.
 - 두 개 이상의 SELECT문장의 열의 수와 데이터 타입 일치
 - 열의 이름 일치하지 않을 경우 첫 번째 SELECT문의 열의 이름
 - 정렬을 위해서는 마지막 SELECT문에 ORDER BY절 표현
 - 집합 연산자 부질의에 사용가능

❖ 집합(SET)

- 실습을 위해 a_student, b_student 테이블 생성

```
SQL> create table a_student
```

```
2 as select *
```

```
3 from student
```

```
4 where stu_dept in ('기계', '전기전자');
```

```
SQL> create table b_student
```

```
2 as select *
```

```
3 from student
```

```
4 where stu_dept in ('전기전자', '컴퓨터정보');
```

❖ 집합(SET)

➤ 생성된 a_student, b_student의 내용 확인

SQL> select * from a_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63

SQL> select * from b_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종헌	컴퓨터정보	3	C	M		72
20131025	옥성우	컴퓨터정보	3	A	F	172	63

❖ 집합(SET)

● UNION

SQL> select * from a_student

2 union

3 select * from b_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종헌	컴퓨터정보	3	C	M		72
20131025	옥성우	컴퓨터정보	3	A	F	172	63

❖ 집합(SET)

● UNION

SQL> select * from a_student

2 union all

3 select * from b_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종헌	컴퓨터정보	3	C	M		72
20131025	옥성우	컴퓨터정보	3	A	F	172	63

❖ 집합(SET)

● INTERSECT

SQL> select * from a_student

2 intersect

3 select * from b_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63

❖ 집합(SET)

● MINUS

```
SQL> select * from a_student  
2 minus  
3 select * from b_student;
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47

❖ 데이터 정의어(DDL)

개체(Object)

테이블(Table)

제약조건(Constraint)

뷰(View)

❖ 개체(Object)

● 데이터 정의어 (DDL : Data Definition Language)

- 데이터베이스의 3층 스키마를 정의
- 데이터베이스 여러 개체 기술

● 개체(Object)

Object	설 명
Table	행과 열로 구성된 2차원 테이블로 데이터 저장하는 개체
View	하나 이상의 테이블로부터 유도된 데이터의 부분집합 개체
Sequence	순차적인 숫자 값을 생성하는 개체
Index	빠른 검색 위해 사용하는 개체

❖ 테이블(Table)

● 테이블(Table)

- 테이블은 데이터를 저장할 수 있는 개체
- 테이블의 생성을 위해서는 **CREATE TABLE**
- 변경을 위해서는 **ALTER TABLE**
- 삭제를 위해서는 **DROP TABLE**

● 테이블 생성 SQL

```
SQL> create table test1  
2   (u_id varchar2(10),  
3   u_date date);
```

❖ 데이터 속성(type)

● 데이터 속성(type)

타 입	설 명
NUMBER(n,m)	숫자 데이터에 대한 정의에 사용
CHAR(n)	문자 데이터에 대한 정의에 사용
VARCHAR2(n)	가변길이 문자데이터에 대한 정의에 사용
DATE	날짜 데이터에 대한 정의에 사용
LONG	2GB의 가변길이 문자 데이터에 대한 정의에 사용
TIMESTAMP	년,월,일,시,분,초, 6자리 소수부 초 형태로 시간정보를 정의에 사용
LOB	4GB의 텍스트, 동영상, 이미지, 사운드 등에 대한 정의에 사용
ROWID	각행에 대한 논리적인 위치(주소), 의사열

❖ 테이블 생성

● 테이블 생성

```
CREATE TABLE table-name ( column-name1 data-  
type default-value,  
column-name2 data-type default-value,  
.....  
column-name data-type default-value );
```

또는

```
CREATE TABLE table-name  
AS sub-query;
```

❖ 테이블 생성

- STUDENT 테이블 생성

```
SQL> create table student(  
2   stu_no char(9),  
3   stu_name varchar2(12),  
4   stu_dept varchar2(20),  
5   stu_grade number(1),  
6   stu_class char(1),  
7   stu_gender char(1),  
8   stu_height number(5,2),  
9   stu_weight number(5,2));
```

테이블이 생성되었습니다.

❖ 테이블 생성

- 기존의 테이블을 이용하여 새로운 테이블 생성

SQL> create table t_student

2 as select * from student where stu_dept = '기계';

테이블이 생성되었습니다.

❖ 테이블 생성

- 생성된 새로운 테이블 확인

SQL> desc t_student;

이름	널?	유형
STU_NO	NOT NULL	CHAR(9)
STU_NAME		VARCHAR2(12)
STU_DEPT		VARCHAR2(20)
STU_GRADE		NUMBER(1)
STU_CLASS		CHAR(1)
STU_GENDER		CHAR(1)
STU_HEIGHT		NUMBER(5,2)
STU_WEIGHT		NUMBER(5,2)

❖ 테이블 생성

- 새로 생성한 t_student의 데이터를 확인

SQL> select * from t_student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47

❖ 테이블 변경

● 테이블 변경

- ALTER TABLE 명령어 사용
- 새로운 열을 삽입, 기존의 열을 삭제, 기존 열을 변경한다.

(1) 새로운 열 추가

```
ALTER TABLE table-name      ADD ( column-name1  data-type,  
                                   column-name2  data-type,  
                                   ..... );
```

❖ 테이블 변경

- t_student 테이블에 army 열을 삽입

```
SQL> alter table t_student  
2 add ( army char(1));
```

테이블이 변경되었습니다.

❖ 테이블 변경

- 새로운 열 삽입 확인.

SQL> desc t_student;

이름	널?	유형
STU_NO	NOT NULL	CHAR(9)
STU_NAME		VARCHAR2(12)
STU_DEPT		VARCHAR2(20)
STU_GRADE		NUMBER(1)
STU_CLASS		CHAR(1)
STU_GENDER		CHAR(1)
STU_HEIGHT		NUMBER(5,2)
STU_WEIGHT		NUMBER(5,2)
ARMY		CHAR(1)

❖ 테이블 변경

(2) 열 구조 변경

```
ALTER TABLE table-name      MODIFY ( column-name1  data-  
type,  
                                column-name2  data-type,  
                                ..... );
```

❖ 테이블 변경

- t_student 테이블에 army 열의 구조를 변경

```
SQL> alter table t_student  
2 modify(army number);
```

테이블이 변경되었습니다.

❖ 테이블 변경

- army 열 변경 확인

SQL> desc t_student;

이름	널?	유형
STU_NO	NOT NULL	CHAR(9)
STU_NAME		VARCHAR2(12)
STU_DEPT		VARCHAR2(20)
STU_GRADE		NUMBER(1)
STU_CLASS		CHAR(1)
STU_GENDER		CHAR(1)
STU_HEIGHT		NUMBER(5,2)
STU_WEIGHT		NUMBER(5,2)
ARMY		NUMBER

❖ 테이블 변경

(3) 열의 삭제

```
ALTER TABLE table-name  
    DROP ( column-name1 , column-name2 ..... );
```

❖ 테이블 변경

- t_student 테이블에 army 열 삭제

```
SQL> alter table t_student  
2 drop( army);
```

테이블이 변경되었습니다.

❖ 테이블 변경

- army 열 삭제 확인

SQL> desc t_student;

이름	널?	유형
STU_NO	NOT NULL	CHAR(9)
STU_NAME		VARCHAR2(12)
STU_DEPT		VARCHAR2(20)
STU_GRADE		NUMBER(1)
STU_CLASS		CHAR(1)
STU_GENDER		CHAR(1)
STU_HEIGHT		NUMBER(5,2)
STU_WEIGHT		NUMBER(5,2)

❖ 테이블 이름 변경

- 테이블 이름변경

```
RENAME old_table_name TO new_table_name;
```

- t_student 테이블 이름을 test_student로 변경

```
SQL> rename t_student to test_student;
```

테이블이 변경되었습니다.

❖ 테이블 이름 변경

- 이름변경 확인

```
SQL> desc t_student;
```

ERROR : ORA-04043: 객체 t_student가 존재하지 않습니다

```
SQL> desc test_student;
```

이름	널?	유형
STU_NO	NOT NULL	CHAR(9)
STU_NAME		VARCHAR2(12)
STU_DEPT		VARCHAR2(20)
STU_GRADE		NUMBER(1)
STU_CLASS		CHAR(1)
STU_GENDER		CHAR(1)
STU_HEIGHT		NUMBER(5,2)
STU_WEIGHT		NUMBER(5,2)

❖ 테이블내의 데이터 삭제

- 테이블 내의 데이터 삭제

```
TRUNCATE TABLE table-name
```

- test_student 내의 모든 데이터 삭제

```
SQL> truncate table test_student;
```

테이블이 잘렸습니다.

- test_student 테이블 검색

```
SQL> select * from test_student;
```

선택된 레코드가 없습니다.

❖ 테이블 삭제

- 테이블 삭제

```
DROP TABLE table-name;
```

- test_student 테이블 삭제

```
SQL> rename t_student to test_student;
```

테이블이 삭제되었습니다.

- test_student 테이블 검색

```
SQL> desc test_student;
```

ERROR : ORA-04043: 객체 test_student가 존재하지 않습니다.

❖ 제약조건

● Constraints(제약조건)

- 제약조건은 데이터베이스 상태가 항상 만족해야 할 기본 규칙
- 제약조건은 데이터베이스 스키마에 명시
- 데이터가 삽입, 삭제, 수정 등의 연산이 일어나도 지속적으로 만족

(1) 도메인 제약 조건

- 각 열의 값은 반드시 해당 도메인에 속하는 원자

(2) 키 제약 조건

- 테이블에는 테이블의 각 레코드를 유일하게 식별할 수 있는 수단 즉, 최소한 하나의 기본키를 가지고 있어야 한다.

❖ 제약조건

(3) 무결성 제약 조건(integrity constraint)

- 엔티티 무결성(entity integrity)
 - 기본키 속성들의 값은 어떠한 경우에도 널 값을 가질 수 없다.
- 참조 무결성(referential integrity)
 - 한 테이블에 있는 레코드가 다른 테이블에 있는 레코드를 참조하려면 반드시 참조되는 레코드가 그 테이블 내에 존재해야 한다.(외래키)

❖ 제약조건

● 오라클 제약조건 5가지 유형

제약 조건	설 명
NOT NULL	열에 NULL값을 허용하지 않음
UNIQUE KEY	열 또는 열의 조합이 유일성(유일한 값)을 가져야 함
PRIMARY KEY	열 또는 열의 조합이 NULL값이 아니며, 유일성을 가져야 함
FOREIGN KEY	다른 테이블의 열을 참조되는 테이블에 값이 존재하여야 함
CHECK	열에 들어갈 값에 대한 조건을 명시함

❖ NOT NULL

● NOT NULL

- 해당 열에 NULL 값이 저장되어서는 안 되는 경우 사용
- Primary Key 제약조건은 NOT NULL 조건 만족

```
SQL> create table t_student(  
2   stu_no char(9),  
3   stu_name varchar2(12),  
4   stu_dept varchar2(20)  
5   constraint n_stu_dept not null,  
6   stu_grade number(1),  
7   stu_class char(1),  
8   stu_gender char(1),  
9   stu_height number(5,2),  
10  stu_weight number(5,2));
```

❖ UNIQUE KEY

● UNIQUE KEY

- 기본 키가 아닌 열의 값이 유일한 값을 유지하여야 할 때 사용
- 자동으로 INDEX가 만들어진다.

```
SQL> create table t_student(  
2   stu_no char(9),  
3   stu_name varchar2(12)  
4   constraint u_stu_name unique,  
5   stu_dept varchar2(20)  
6   constraint n_stu_dept not null,  
7   stu_grade number(1),  
8   stu_class char(1),  
9   stu_gender char(1),  
10  stu_height number(5,2),  
11  stu_weight number(5,2));
```

❖ PRIMARY KEY

● PRIMARY KEY

- 기본 키는 한 테이블에 단 한 개만 존재
- 자동으로 INDEX가 만들어진다.

```
SQL> create table t_student(  
2   stu_no char(9),  
3   stu_name varchar2(12)  
4   constraint u_stu_name unique,  
5   stu_dept varchar2(20)  
6   constraint n_stu_dept not null,  
7   stu_grade number(1),  
8   stu_class char(1),  
9   stu_gender char(1),  
10  stu_height number(5,2),  
11  stu_weight number(5,2));  
12  constraint p_stu_no primary key(stu_no)
```


❖ PRIMARY KEY

```
SQL> create table t_enrol(  
2   sub_no char(3),  
3   stu_no char(9),  
4   enr_grade number(3),  
5   constraint p_enol primary key(sub_no,stu_no));
```

❖ FOREIGN KEY

● FOREIGN KEY

- 참조 무결성을 유지하기 위해 자식 테이블의 열을 외래 키로 선언
- Enrol 테이블의 sub_no, stu_no는 외래 키이며, 부모 테이블은 각각 subject, student 이다.

```
SQL> create table t_enrol(  
2   sub_no char(3),  
3   stu_no char(9),  
4   enr_grade number(3)  
5       constraint enr_sub_no fk1 foreign key(sub_no)  
6                               references subject(sub_no),  
7   constraint enr_stu_no fk2 foreign key(stu_no)  
8                               references subject(stu_no),  
9   constraint p_enol primary key(sub_no,stu_no));
```

❖ CHECK

● CHECK

➢ 어떤 열의 값에 조건을 제시

```
SQL> create table t_student(  
2   stu_no char(9),  
3   stu_name varchar2(12)  
4       constraint u_stu_name unique,  
5   stu_dept varchar2(20)  
6       constraint n_stu_dept not null,  
7   stu_grade number(1),  
8   stu_class char(1),  
9   stu_gender char(1),  
10      constraint c_stu_gender check (stu_gender in('M','F'))  
11   stu_height number(5,2),  
12   stu_weight number(5,2));  
13      constraint p_stu_no primary key(stu_no)
```

❖ 제약 조건의 확인

● 제약 조건의 확인

➢ 데이터 사전의 테이블은 USER_CONSTRAINTS

```
SQL> select *
2   from user_constraints
3  where table_name = 'T_STUDENT';
```

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	STU_NO_PK	P	T_STUDENT

```
SQL> select *
2   drop
3  where table_name = 'T_ENROL';
```

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SCOTT	ENR_PK1	P	T_ENROL
SCOTT	ENR_SUB_NO_FK1	R	T_ENROL
SCOTT	ENR_STU_NO_FK1	R	T_ENROL

❖ 제약 조건의 삭제

● 제약 조건의 삭제

➤ 제약조건을 삭제할 경우, ALTER TABLE 명령 사용

```
ALTER TABLE table-name  
    DROP CONSTRAINT constraint_name [CASCADE];
```

```
SQL> alter table t_enrol  
2 drop constraint f_enr_sub_no cascade;
```

❖ 제약조건의 활성화, 비활성화

● 제약 조건의 활성화, 비활성화

➤ 제약조건을 비활성화 할 경우, ALTER TABLE 명령을 사용

```
ALTER TABLE table-name  
DISABLE | ENABLE CONSTRAINT constraint_name [CASCADE];
```

```
SQL> alter table t_student  
2  disable constraint n_stu_dept;
```

```
SQL> alter table t_student  
2  enable constraint n_stu_dept;
```

❖ 뷰(View)

● 뷰(VIEW)

➤가상의 테이블 (virtual table)

➤뷰의 장점

- ① 뷰는 데이터베이스를 재구성하여 논리적 데이터 독립성 제공
- ② 원하는 데이터만을 조작함으로 데이터의 보완기능 강화

```
CREATE VIEW view-name  
    [ col-name [, col-name ..... ] ]  
AS  
    subquery;
```

```
DROP VIEW view-name;
```

❖ 뷰(View)

- 단순 뷰(VIEW)

- 단순 뷰는 단일 베이스 테이블로 부터 유도된 뷰이다.

```
SQL> create or replace view v_student1  
2 as select * from student where stu_dept = '컴퓨터정보';
```

뷰가 생성되었습니다.

❖ 뷰(View)

●뷰의 생성을 확인

```
SQL> select *
      2  from v_student1;
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종현	컴퓨터정보	3	C	M		72
20131025	옥성우	컴퓨터정보	3	A	F	172	63

❖ 뷰(View)

● 조인 뷰(VIEW)

➢ 조인뷰는 2개 이상의 베이스 테이블로부터 유도된 뷰

```
SQL> create or replace view v_enrol1  
2 as select sub_name, a.sub_no, stu_no, enr_grade  
3 from enrol a, subject b  
4 where a.sub_no = b.sub_no;
```

뷰가 생성되었습니다.

❖ 뷰(View)

●뷰의 생성을 확인

```
SQL> select *  
2 from v_enrol1;
```

SUB_NAME	SUB_NO	STU_NO	ENR_GRADE
컴퓨터개론	101	20131001	80
컴퓨터개론	101	20131025	65
기계공작법	102	20153075	66
기계공작법	102	20153088	61
기초전자실험	103	20152088	45
시스템분석설계	104	20131001	56
시스템분석설계	104	20131025	65
기계요소설계	105	20153088	78
기계요소설계	105	20153075	56
전자회로실험	106	20132003	72
CAD응용실습	107	20143054	41
소프트웨어공학	108	20151062	81

❖ 뷰(View)

● 인라인(INLINE)뷰

- 인라인 뷰는 FROM절에 SELECT문으로 정의된 뷰이다.
- 학과별 평균 신장보다 큰 학생들의 학번, 이름, 신장 검색

```
SQL> select stu_no, stu_name, a.stu_dept, stu_height
2   from student a, (select stu_dept, avg(stu_height)
3                      as avg_height
4                      from student
5                      group by stu_dept) b
6   where a.stu_dept = b.stu_dept
7   and a.stu_height > b.avg_height
```

STU_NO	STU_NAME	STU_DEPT	STU_HEIGHT
20153075	옥한빛	기계	177
20152088	조민우	전기전자	188
20141007	진현무	컴퓨터정보	174
20131025	옥성우	컴퓨터정보	172

❖ 뷰(View)

● TOP-N 질의

- “신장이 큰 상위 5명의 학생” 또는 “성적이 높은 상위 5명의 학생”과 같이 최댓값 또는 최솟값을 가진 열에 몇 개의 레코드만 추출할 때 사용되는 기능

```
SELECT item-commalist
      FROM (SELECT item-commalist
            FROM table name
            ORDER BY item)
WHERE ROWNUM <= n;
```

❖ 뷰(View)

- 학생 테이블에서 신장이 큰 상위 5명의 학번, 이름, 신장 검색

```
SQL> select stu_no, stu_name, stu_height
2  from (select stu_no, stu_name, stu_height
3         from student
4         where stu_height is not null
5         order by stu_height desc)
6  where rownum <= 5;
```

STU_NO	STU_NAME	STU_HEIGHT
20152088	조민우	188
20153075	옥한빛	177
20141007	진현무	174
20131025	옥성우	172
20142021	심수정	168