

데이터베이스와 SQL

목차

1. 데이터베이스 개요
2. 오라클 설치 및 수행
3. SQL

1. 데이터베이스 개요

- 데이터베이스 정의
- 데이터베이스 특징
- 데이터베이스 추상화
- 데이터 독립성
- 관계형 데이터 모델
- 데이터베이스 언어(SQL)
- 데이터베이스 관리시스템(DBMS)

※ 본 자료의 데이터베이스 개요는 '모바일 시대의 컴퓨터개론, 강환수와 3인, 인피니티북스' 에서 인용하였음을 밝힙니다.

데이터베이스 정의

- 데이터와 정보
 - 데이터가 사람에게 유용한 의미로 쓰여질 수 있도록 처리되면 정보 (Information)
- 데이터베이스 정의
 - 데이터베이스는 간단히 '관련 있는 데이터의 저장소'
 - 데이터베이스는 여러 사람이나 응용시스템에 의해 참조 가능하도록 서로 논리적으로 연관되어 통합 관리되는 데이터의 모임
- 특징
 - 데이터베이스는 통합된(Integrated) 관련(Related) 있는 데이터
 - 중복(Redundancy)을 최소화하여 보조기억장치에 저장



[그림 1.1] 데이터에서 정보로 활용하기 위한 체계적 저장 관리

데이터베이스 특징

<표 1.1> 데이터베이스 특징

데이터베이스 특징	내용
통합된 데이터	데이터의 특성, 실체 상호 간의 의미 관계와 형식 관계를 기술한 개념적인 구조에 따라서 편성된 데이터의 집합
관련 있는 데이터	동시에 복수의 적용 업무나 응용 시스템에 대한 데이터의 공급 기지로서 공유할 필요가 있는 데이터를 보관, 관리
중복의 최소화	동일한 내용의 데이터가 중복되지 않아야 하고, 다양한 접근 방식이 마련되어 있어야 하며, 검색이나 갱신이 효율적으로 이루어질 수 있도록 중복을 최소화
보조기억장치에 저장	자기 디스크나 자기 테이프 등 컴퓨터에서 사용할 수 있는 보조 기억 장치에 저장
무결성	데이터가 정확성을 항상 유지
동시 접근	여러 사람이 동시에 자료에 접근하더라도 문제없이 작업을 수행
보안 유지	데이터베이스의 관리 및 접근을 효율적으로 관리하여 보안 유지
장애 회복	문제가 발생하더라도 이전 상태로 복구 가능

5

데이터베이스 추상화

• 데이터베이스 추상화의 3단계

- 물리적 단계(physical level) **DB관리자(DBA)**
- 논리적 단계(logical level) **DB설계자(DA)**
- 뷰 단계(view level) **End User**

• 스키마

- 데이터베이스의 전체적인 설계
- 즉 스키마란 데이터베이스를 구성하는 정보의 종류와 관계의 구체적인 기술(description)

- ✓ 개념스키마
- ✓ 논리스키마
- ✓ 물리스키마



그림 7.6 데이터베이스 추상화 3단계와 데이터베이스 스키마

[그림 1.2] 데이터베이스 추상화 3단계와 데이터베이스 스키마

6

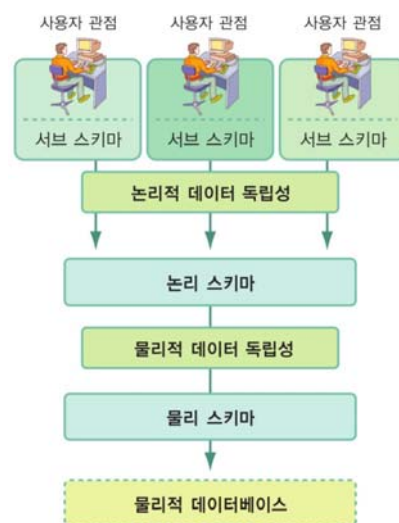
데이터베이스 추상화

- 물리적 단계: 내부 단계(internal level)
 - 저장 장치의 내부에 실질적으로 데이터가 저장될 구조와 위치를 결정
 - 하위 수준의 접근 방식을 다루고 바이트들이 어떻게 저장 장치로부터 변환이 되는지 다룸
 - 하드웨어와 직접적인 상호 작용을 다룸
 - 데이터베이스의 물리적 구조를 기술한 것으로 하위 데이터 모델을 통해 표현
- 논리적 단계: 개념 단계(conceptual level)
 - 데이터베이스에 저장될 데이터의 종류와 데이터 간의 관계를 기술
 - 논리 스키마는 복잡한 데이터베이스의 내부 구조를 알 필요 없이 비교적 간단한 데이터 구조로 전체 데이터베이스를 기술
- 뷰 단계: 외부 단계(external level)
 - 추상화의 최상위 단계, 사용자와 직접적인 상호작용
 - 논리적 단계에서 나온 데이터를 사용자에게 친숙한 형태의 뷰(view)로 변환하여 사용자에게 제공
 - 서브 스키마 또는 외부 스키마(external schema)
 - 사용자마다 다른 뷰에서 본인의 관심인 데이터베이스의 일부분을 정의한 것

7

데이터 독립성

- 논리적 데이터 독립성
 - 논리적 데이터 독립성은 사용자의 응용 프로그램 자체에 영향을 주지 않고 논리적 단계에서의 논리 스키마를 수정할 수 있는 능력
- 물리적 데이터 독립성
 - 사용자의 응용프로그램 자체나 데이터 베이스의 논리 스키마에 영향을 주지 않고 데이터의 물리적 스키마를 수정할 수 있는 능력



[그림 1.3] 데이터 독립성

8

관계형 데이터 모델

- 관계의 구조
 - 관련성을 표현한 이차원 테이블을 관계(relation)
 - 관계 스키마
 - 관계의 구조를 정의
 - 관계 이름인 학생
 - 학생의 속성 구성인 [학생(학번, 이름, 학과, 주소, 지도교수)]
 - 정적인 특성
 - 관계 이름과 속성 이름이 처음에 한번 결정되면 시간의 흐름과 관계없이 동일한 내용이 계속 유지
 - 관계 사례
 - 관계 스키마에 삽입되는 실제 데이터 값
 - 실제로 관계 내부에 삽입된 하나의 자료
 - (2000003, 김근태, 001, 인천, 0002)
 - 동적인 특성
 - 시간이 변함에 따라 실제 사례 값이 변함

학생				
학번	이름	학과	주소	지도교수
2000001	오진호	001	서울	0001
2000002	권다애	002	경기도	0015
2000003	김근태	001	인천	0002
2000004	양보원	003	대전	0022
2000005	김태수	001	서울	0003

[그림 1.4] 관계 스키마와 관계 사례

관계형 데이터 모델

- 속성
 - 관계에서 각 열을 속성(attribute)
 - 속성은 실제 데이터베이스에서는 필드
 - DBMS에서는 열(column)이라고 표현
- 튜플
 - 하나의 관계에서 각 행을 튜플(tuple)
 - 즉 튜플은 관계에서 정의된 모든 속성 값들의 집합
 - 튜플은 실제 데이터베이스에서는 레코드
 - DBMS에서는 행(row)이라고 표현
 - 도메인
 - 하나의 속성이 취할 수 있는 모든 값의 범위를 의미
- 관계의 특징
 - 관계에서 중복된 튜플은 삽입될 수 없음
 - 원자 값(atomic value)
 - 튜플 내의 모든 값은 더 이상 나눌 수 없는 값이어야 함

		속성(열)				
		A1	A2	An
튜플(행)	t1					
	t2					
	...					
	tn					
		도메인 D1	도메인 D2	도메인 Dn

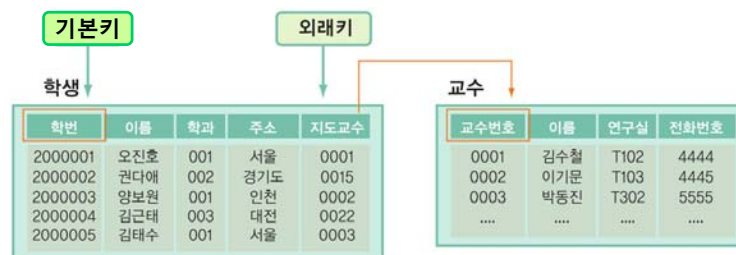
[그림 1.5] 관계의 구성 요소

<표 1.2> 관계의 특징

특징	내용
속성 이름의 유일성	한 관계에서 속성 이름은 유일해야 한다.
원자 값	튜플 내의 모든 값은 더 이상 나눌 수 없는 원자 값(atomic value)이어야 한다.
튜플 간의 순서	관계에서 튜플 간의 순서는 무의미하다.
속성 간의 순서	한 관계에서 속성 간의 순서는 무의미하다.
중복 불허	한 관계에서 두 튜플의 속성 값이 모두 같은 것은 불허한다.

관계형 데이터 모델

- 다양한 키의 종류
 - 후보키(candidate key)
 - 하나의 관계에서 유일성과 최소성(minimality)을 만족하는 키
 - 기본키(primary key)
 - 주키는 관계에서 여러 튜플 중에서 하나의 튜플을 식별하는 역할을 수행
 - 외래키(foreign key)
 - 외래키는 어느 관계의 속성들 중에서 일부가 다른 관계의 주키가 될 때, 이 키를 외래키라 함
 - 이 외래키를 이용하여 관계와 관계를 서로 연결할 수 있음



[그림 1.6] 기본키와 외래키

11

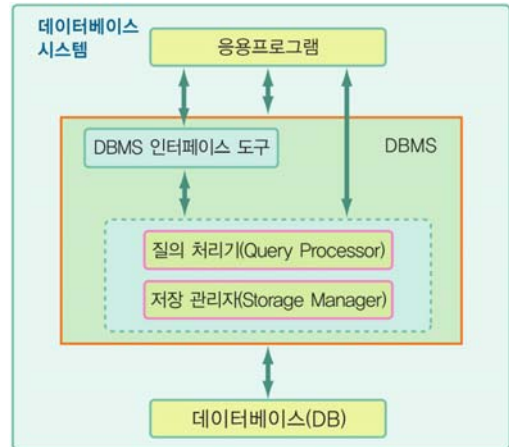
데이터베이스 언어(SQL)

- 데이터베이스 언어
 - DBMS (Database Management System) 을 통해서 데이터베이스의 구축 및 사용자와 데이터베이스 간의 소통 수단으로 데이터 정의, 조작, 제어하는데 사용되는 언어
 - 데이터베이스 언어의 종류로는 그 역할에 따라 데이터베이스 정의어(DDL), 조작어(DML), 제어어(DCL)로 구분
- DDL (Data Definition Language)
 - 데이터베이스를 구축하거나 수정하는데 사용되며 데이터베이스의 구조와 데이터의 형식, 접근방식을 정의하는 언어
 - DDL은 번역된 결과가 Data Dictionary 라는 데이터 사전 파일에 저장됨
 - 종류 : CREATE, ALTER, DROP, RENAME, TRUNCATE
- DML (Data Manipulation Language)
 - 사용자 DBMS와 응용프로그램 간의 interface를 제공하며 사용자가 데이터를 직접 처리할 수 있게 하는 명령어로 데이터베이스의 Sub Language 역할 수행
 - 종류 : SELECT, INSERT, UPDATE, DELETE
- DCL (Data Control Language)
 - 데이터베이스에 접근하고 객체들을 사용하도록 권한을 주고 회수하는 언어
 - 종류 : GRANT, REVOKE

12

데이터베이스 관리시스템(DBMS)

- 데이터베이스 관리시스템
 - DBMS: Database Management System
 - 데이터베이스를 정의하고
 - 데이터베이스를 구축하고
 - 데이터베이스를 조작하고
 - 데이터베이스를 제어
 - 데이터베이스에서 정보를 쉽게 활용할 수 있도록 만든 프로그램이자 소프트웨어
- DBMS 구성
 - 저장 관리자(Storage Manager)
 - 질의 처리기(Query Processor)
 - DBMS 인터페이스 도구(DBMS Interface Tool)



[그림 1.7] 데이터베이스의 구성요소와 DBMS

데이터베이스 관리시스템(DBMS)

- 상용 DBMS
 - MySql
 - MySQL은 대표적인 오픈 소스 DBMS 제품
 - MySQL은 원래 mSQL이라는 DBMS에서 기반이 되어 새로 개발된 DBMS
 - 오라클 ← 여기서는 오라클로 실습
 - 1977년 설립된 오라클(Oracle)사가 개발한 오라클은 세계적으로 가장 성공한 DBMS의 한 제품
 - SQL 서버
 - 원래 사이베이스(Sybase)사의 DBMS 엔진을 윈도우 NT에 탑재하면서 시작
 - 버전 4.2까지 사이베이스와 공동 개발을 함
 - 마이크로소프트사는 사이베이스의 제품을 완전히 사들여 독자적으로 DBMS의 내부 커널을 재설계하여 SQL 서버 6.0을 발표함

2. 오라클 설치 및 수행

- 오라클 개요
- 오라클 설치
- SQL*PLUS
- 예제 데이터베이스 생성 (오라클 사용자 생성)
- 예제 테이블 생성과 데이터 입력
- 테이블 생성 확인
- 테이블 구조 확인
- 테이블 데이터 확인

※ 본 자료의 예제 데이터베이스, 테이블, 데이터는 'oracle을 기반으로 하는 데이터베이스배움터, 홍의경, 생능출판사'에서 인용하였음을 밝힙니다.

오라클 개요

□ 오라클

- ✓ 오라클사가 개발한 객체 관계 DBMS
- ✓ 가장 높은 시장 점유율과 신뢰성을 지닌 데이터베이스 관리 시스템
- ✓ 유닉스, 리눅스, 윈도우 등 대부분의 운영체제를 지원
- ✓ PC에서 대형 기종에 이르기까지 다양한 하드웨어와 운영체제 플랫폼을 지원
- ✓ 엔터프라이즈 에디션, 표준 에디션, 개인용 에디션 등 3가지 에디션으로 공급됨

오라클 개요

□ 오라클 개요

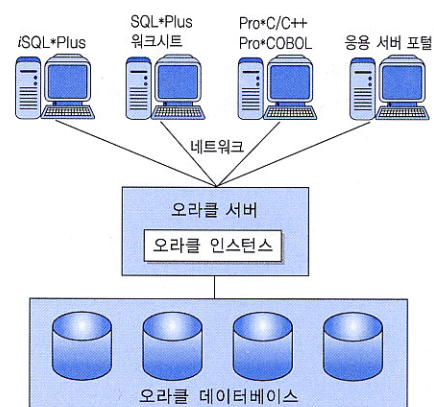
- ✓ 전자상거래와 데이터 웨어하우징을 위한 고성능의 인터넷 플랫폼을 제공
- ✓ 사용자가 객체 관계 데이터베이스는 물론이고, 워드 프로세서 문서, 스프레드 시트 문서, 파워포인트로 작성한 발표 자료, XML, 그래픽스, 비디오 등과 같은 멀티미디어 데이터 타입 등을 관리할 수 있음
- ✓ 온라인 트랜잭션 처리(OLTP:Online Transaction Processing), 데이터 웨어하우스, OLAP(Online Analytical Processing), 전자 상거래 등 최근에 등장하고 있는 데이터베이스의 중요한 응용 분야에도 활용할 수 있음

17

오라클 개요

□ 오라클 서버와 인스턴스

- ✓ 오라클 서버 : 오라클 인스턴스와 오라클 데이터베이스로 구성
- ✓ 오라클 인스턴스 : 백그라운드 프로세스들과 메모리 구조의 조합
- ✓ 사용자가 오라클 서버에 SQL 문을 입력하기 전에 반드시 오라클 인스턴스에 연결되어 있어야 함
- ✓ 접속(connection) : 사용자 프로세스와 서버 프로세스 간의 통신 경로
- ✓ 세션(session) : 사용자가 오라클 서버로부터 인증될 때부터 시작하여 사용자가 로그아웃을 하거나 비정상적으로 종료될 때까지 지속

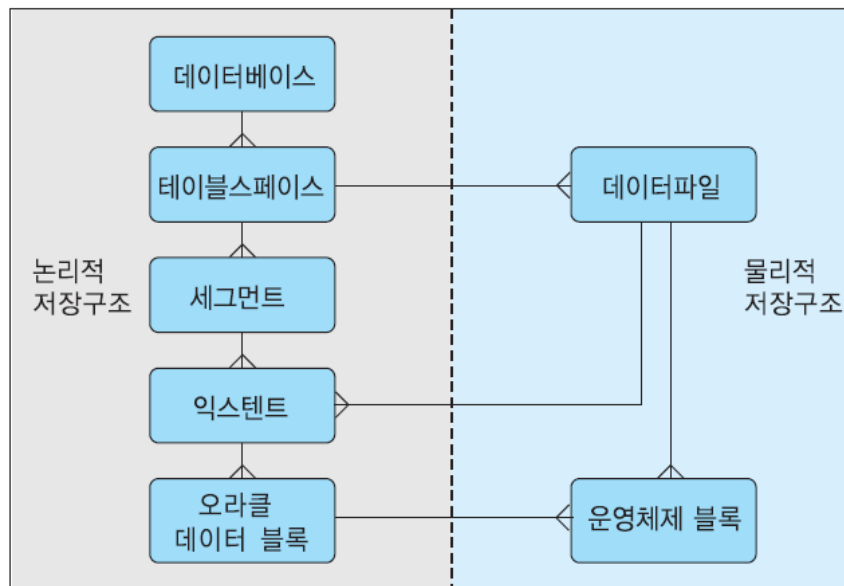


[그림 2.1] 오라클 아키텍처

18

오라클 개요

□ 오라클 저장구조



[그림 2.2] 오라클 저장구조

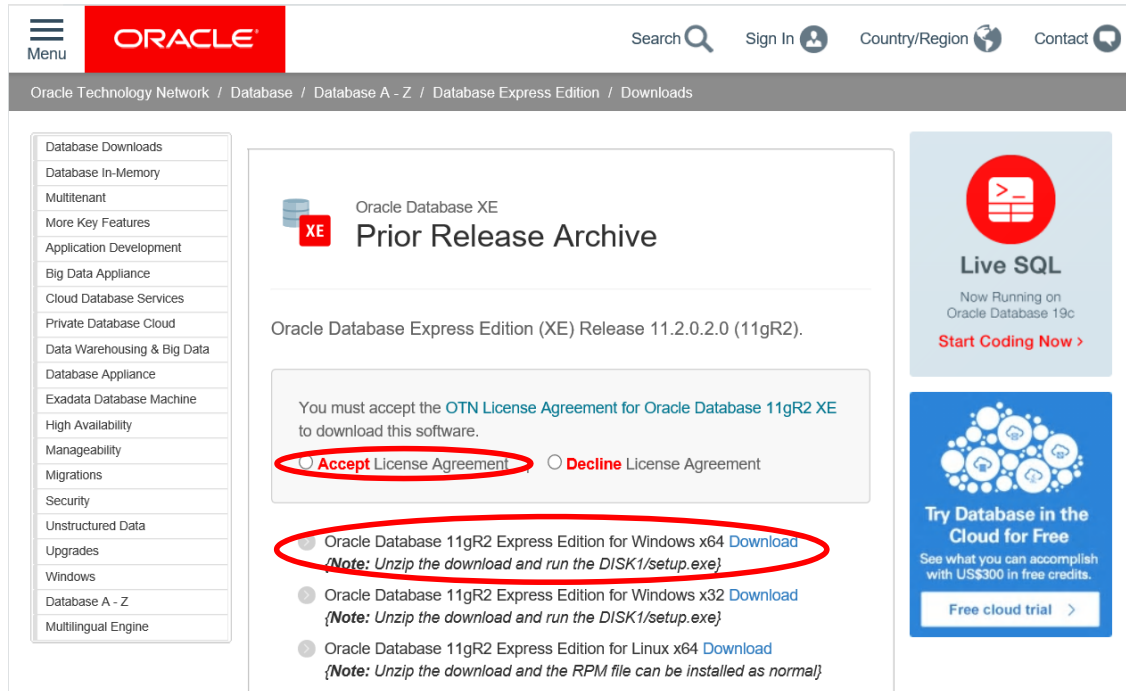
오라클 설치

□ 오라클 11g Express 에디션을 컴퓨터 시스템에 설치

- ✓ 오라클 11g express 에디션은 윈도우 환경에 설치 가능
- ✓ 오라클 홈페이지(<http://otn.oracle.com>)에 접속하여 오라클을 다운로드하는 웹페이지를 찾음
- ✓ 오라클 사이트에 회원으로 가입한 사람만 다운로드할 수 있음

오라클 설치

- <https://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/xe-prior-releases-5172097.html>



[그림 2.3] 오라클 다운로드 사이트

21

3장. 오라클

오라클 설치

□ Oracle Universal Installer(OUI)

- ✓ OUI는 오라클 제품의 설치 및 구성 과정을 안내
- ✓ 오라클의 구성요소들을 설치하고, 업그레이드하고, 제거하고, 데이터베이스를 생성하는데 사용됨
- ✓ 오라클 8i부터 모든 플랫폼에서 사용 가능한 자바 기반의 GUI 형식의 프로그램으로 만들어짐

22

오라클 설치

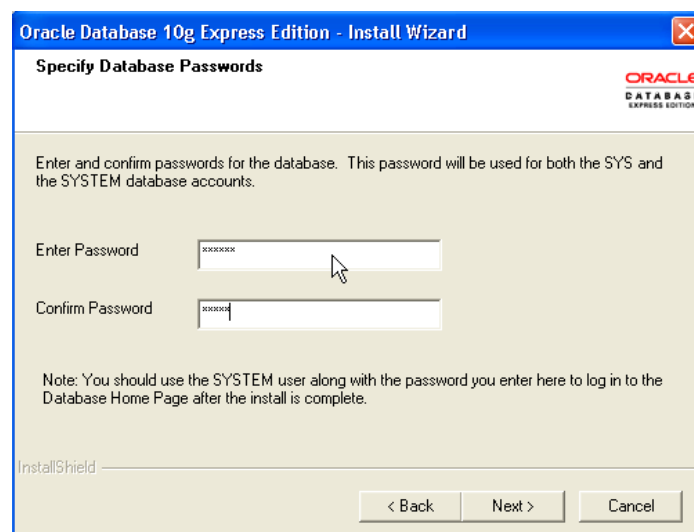
❑ DBA 계정 : SYS, SYSTEM

- ✓ 데이터베이스 생성 과정에 **SYS**와 **SYSTEM** 계정이 자동적으로 만들어짐
- ✓ 이 두 계정은 데이터베이스 관리자 역할
- ✓ SYS의 데이터 사전의 소유자
- ✓ SYSTEM의 오라클 도구들이 사용하는 내부 테이블과 뷰들의 소유자

23

오라클 설치

- system 계정 비밀번호 설정



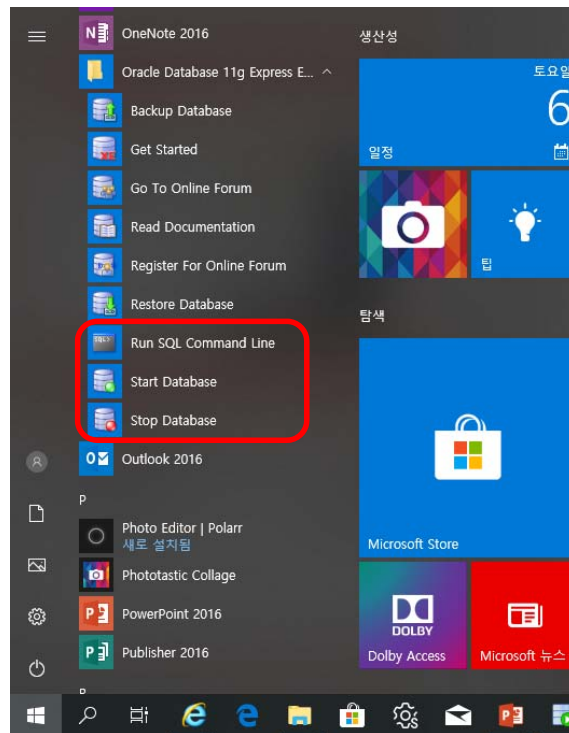
[그림 2.4] 오라클 설치 과정에서 관리자(system) 계정 비밀번호 설정 화면

24

오라클 설치

❑ 오라클 관련 프로그램

- SQL 명령줄 실행
- 데이터베이스 시작
- 데이터베이스 중지



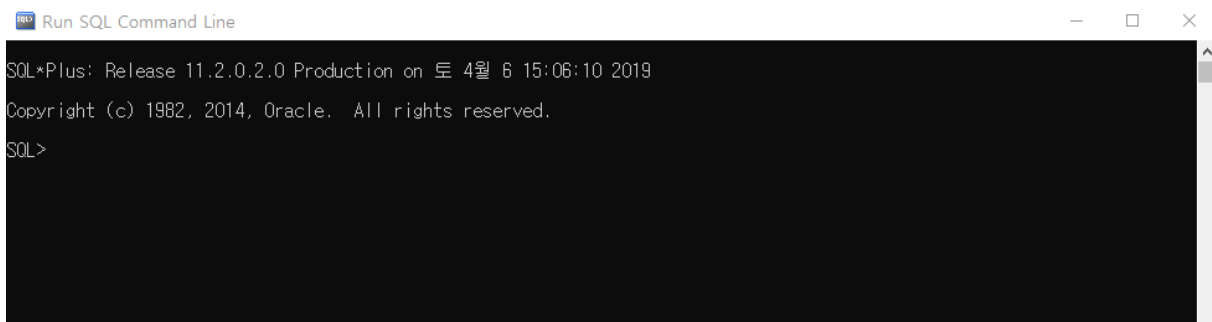
[그림 2.5] 오라클 설치 후 프로그램 메뉴

25

SQL*Plus

❑ SQL*Plus 워크시트 창

- ✓ SQL*Plus 워크시트에서 SQL문이나 PL/SQL문을 입력하고, 편집하고, 실행할 수 있음
- ✓ SQL*Plus 워크시트에서 클라이언트의 스크립트를 실행할 수도 있음
- ✓ SQL*Plus 워크시트는 실행한 명령에 대한 기록을 유지하므로 이전에 실행한 명령을 쉽게 읽어 들여 다시 실행할 수 있음



[그림 2.6] SQL*Plus 워크시트 창

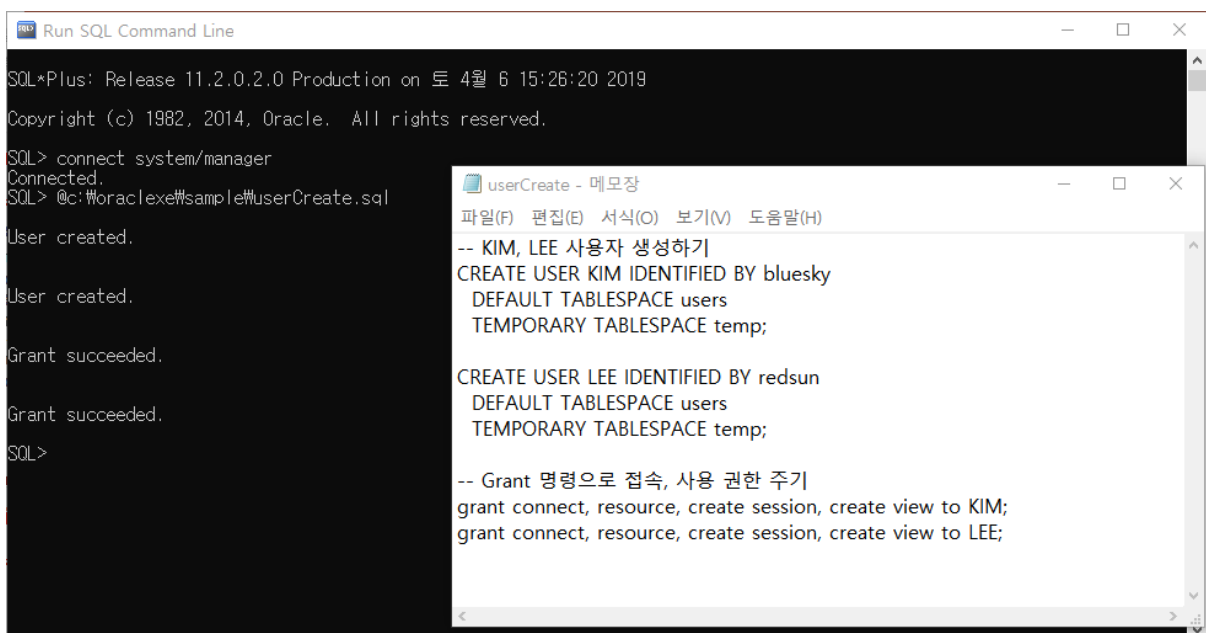
26

예제 데이터베이스 생성 (오라클 사용자 생성)

- ❑ 오라클의 경우, 데이터베이스 스키마 = 오라클 사용자
- ❑ CREATE USER : 사용자 생성
 - ✓ 데이터베이스 관리자(ID : system)는 CREATE USER문을 사용하여 사용자 생성
 - ✓ 디폴트 사용자 테이블스페이스는 users이고, 임시 테이블스페이스는 temp
- ❑ GRANT : 권한 부여
 - ✓ 데이터베이스 관리자는 생성된 사용자에게 다음 권한을 부여
 - connect와 resource 역할(role)
 - create session과 create view 권한(privilege)

27

예제 데이터베이스 생성 (오라클 사용자 생성)



```
Run SQL Command Line
SQL*Plus: Release 11.2.0.2.0 Production on 토 4월 6 15:26:20 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect system/manager
Connected.
SQL> @c:\#oracle\exe\sample\userCreate.sql

User created.

User created.

Grant succeeded.

Grant succeeded.

SQL>
```

```
userCreate - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

-- KIM, LEE 사용자 생성하기
CREATE USER KIM IDENTIFIED BY bluesky
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp;

CREATE USER LEE IDENTIFIED BY redsun
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp;

-- Grant 명령으로 접속, 사용 권한 주기
grant connect, resource, create session, create view to KIM;
grant connect, resource, create session, create view to LEE;
```

[그림 2.7] 사용자 생성과 권한 부여

28

예제 테이블 생성과 데이터 입력

❑ 실습 예제(sampledata.sql)

SQL 명령줄 실행

→ connect kim/bluesky

→ @c:\Woraclexe\sample\sampledata.sql

```
Run SQL Command Line

SQL*Plus: Release 11.2.0.2.0 Production on 토 4월 6 15:45:59 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect kim/bluesky
Connected.
SQL> @c:\Woraclexe\sample\sampledata.sql
```

```
sampledata - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
-- DEPARTMENT 테이블 생성
CREATE TABLE DEPARTMENT (
  DEPTNO NUMBER NOT NULL,
  DEPTNAME CHAR(10),
  FLOOR NUMBER,
  PRIMARY KEY(DEPTNO)
);

INSERT INTO DEPARTMENT VALUES(1, '영업', 8);
INSERT INTO DEPARTMENT VALUES(2, '기획', 10);
INSERT INTO DEPARTMENT VALUES(3, '개발', 9);
INSERT INTO DEPARTMENT VALUES(4, '총무', 7);

-- EMPLOYEE 테이블 생성
CREATE TABLE EMPLOYEE (
  EMPNO NUMBER NOT NULL,
  EMPNAME CHAR(10) UNIQUE,
  TITLE CHAR(10) DEFAULT '사원',
  MANAGER NUMBER,
  SALARY NUMBER CHECK (SALARY < 6000000),
  DNO NUMBER DEFAULT 1 CHECK (DNO IN (1,2,3,4)),
  PRIMARY KEY(EMPNO),
  FOREIGN KEY(MANAGER) REFERENCES EMPLOYEE(EMPNO),
  FOREIGN KEY(DNO) REFERENCES DEPARTMENT(DEPTNO) ON DELETE CASCADE
);

INSERT INTO EMPLOYEE VALUES(4377, '이성래', '사장', NULL, 5000000, 2);
INSERT INTO EMPLOYEE VALUES(3426, '박영권', '과장', 4377, 3000000, 1);
INSERT INTO EMPLOYEE VALUES(3011, '이수민', '부장', 4377, 4000000, 3);
INSERT INTO EMPLOYEE VALUES(3427, '최종철', '사원', 3011, 1500000, 3);
INSERT INTO EMPLOYEE VALUES(1003, '조민희', '과장', 4377, 3000000, 2);
INSERT INTO EMPLOYEE VALUES(2106, '김창섭', '대리', 1003, 2500000, 2);
INSERT INTO EMPLOYEE VALUES(1365, '김상원', '사원', 3426, 1500000, 1);

-- EMP_PLANNING 뷰 생성
CREATE VIEW EMP_PLANNING
AS
SELECT E.EMPNAME, E.TITLE, E.SALARY
FROM EMPLOYEE E, DEPARTMENT D
WHERE E.DNO=D.DEPTNO AND D.DEPTNAME='기획';
```

[그림 2.8] 테이블 생성, 데이터 입력, 뷰 생성

29

테이블 생성 확인

❑ 정의한 테이블과 뷰가 데이터베이스에 생성되었는가를 확인

✓ 입력 창에 다음 명령을 입력하여 실행

SELECT * FROM tab;

✓ **tab**은 데이터베이스 내의 테이블에 관한 정보를 나타내는 데이터 사전 뷰

✓ **TNAME**은 테이블의 이름을 나타내고, **TABTYPE**은 테이블의 유형, 즉 테이블 또는 뷰를 나타냄

```
Run SQL Command Line

SQL*Plus: Release 11.2.0.2.0 Production on 토 4월 6 15:59:24 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect kim/bluesky
Connected.
SQL> set linesize 200
SQL> select * from tab ;

TNAME                                TABTYPE    CLUSTERID
-----                                -
DEPARTMENT                           TABLE
EMPLOYEE                             TABLE
EMP_PLANNING                         VIEW
```

[그림 2.9] 테이블 생성 확인 (select * from tab)

30

테이블 구조 확인

□ DEPARTMENT 테이블의 구조 확인

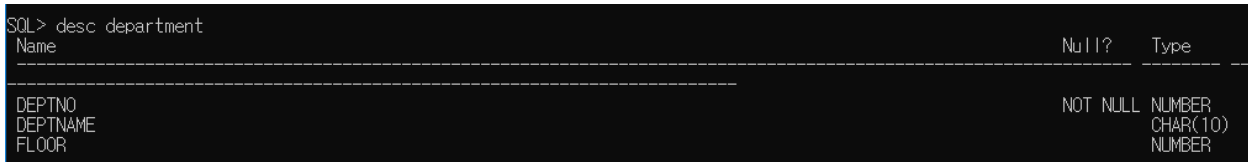
- ✓ 입력 창에 다음 명령을 입력하여 실행

DESCRIBE DEPARTMENT;

또는

DESC DEPARTMENT;

- ✓ 이름은 DEPARTMENT 테이블에 속한 애트리뷰트들을 나타내고, 널?은 애트리뷰트가 널값을 허용하는가를 나타내며, 유형은 애트리뷰트의 데이터 타입과 길이를 의미



```
SQL> desc department
```

Name	Null?	Type
DEPTNO	NOT NULL	NUMBER
DEPTNAME		CHAR(10)
FLOOR		NUMBER

[그림 2.9] 테이블 구조 확인 (desc)

테이블 데이터 확인

□ DEPARTMENT 테이블의 내용 확인

- ✓ 입력 창에 다음 명령을 입력하여 실행

SELECT * FROM DEPARTMENT;



```
SQL> select * from department ;
```

DEPTNO	DEPTNAME	FLOOR
1	영업	8
2	기획	10
3	개발	9
4	총무	7

[그림 2.10] 테이블 데이터 확인

3. SQL

- SQL 개요
- 데이터 정의어와 무결성 제약조건
- SELECT문
- INSERT, DELETE, UPDATE문

※ 본 자료의 SQL 예제는 'oracle을 기반으로 하는 데이터베이스배움터, 홍의경, 생능출판사 ' 에서 인용하였음을 밝힙니다.

SQL 개요

□ SQL 개요

- ✓ SQL은 현재 DBMS 시장에서 관계 DBMS가 압도적인 우위를 차지하는데 중요한 요인의 하나
- ✓ SQL은 IBM 연구소에서 1974년에 **System R**이라는 관계 DBMS 시제품을 연구할 때 관계 대수와 관계 해석을 기반으로, 집단 함수, 그룹화, 갱신 연산 등을 추가하여 개발된 언어
- ✓ 1986년에 ANSI(미국 표준 기구)에서 SQL 표준을 채택함으로써 SQL이 널리 사용되는데 기여
- ✓ 다양한 상용 관계 DBMS마다 지원하는 SQL 기능에 다소 차이가 있음
- ✓ 본 책에서는 SQL2를 따름

SQL 개요

<표 3.1> SQL의 발전 역사

버전	특징
SEQUEL	Structured English Query Language의 약어. Sysetm R 프로젝트에서 처음으로 제안됨
SQL	Structured Query Language의 약어. 1983년에 IBM의 DB2, 1991년에 IBM SQL/DS에 사용됨
SQL-86	1986년에 미국 ANSI에서 표준으로 채택됨. 1987년에 ISO에서 표준으로 채택됨
SQL-89	무결성 제약조건 기능이 강화됨
SQL2(SQL-92)	새로운 데이터 정의어와 데이터 조작어 기능이 추가됨. 약 500페이지 분량
SQL3(SQL-99)	객체 지향과 순환, 멀티미디어 기능 등이 추가됨. 약 2000페이지 분량

SQL 개요

□ SQL 개요(계속)

- ✓ SQL은 비절차적 언어(선언적 언어)이므로 사용자는 **자신이 원하는 바(what)**만 명시하며, 원하는 것을 **처리하는 방법(how)**은 명시할 수 없음
- ✓ 관계 DBMS는 사용자가 입력한 SQL문을 번역하여 사용자가 요구한 데이터를 찾는데 필요한 모든 과정을 담당
- ✓ SQL의 장점은 자연어에 가까운 구문을 사용하여 질의를 표현할 수 있다는 것
- ✓ 두 가지 인터페이스
 - 대화식 SQL(interactive SQL)
 - 내포된 SQL(embedded SQL)

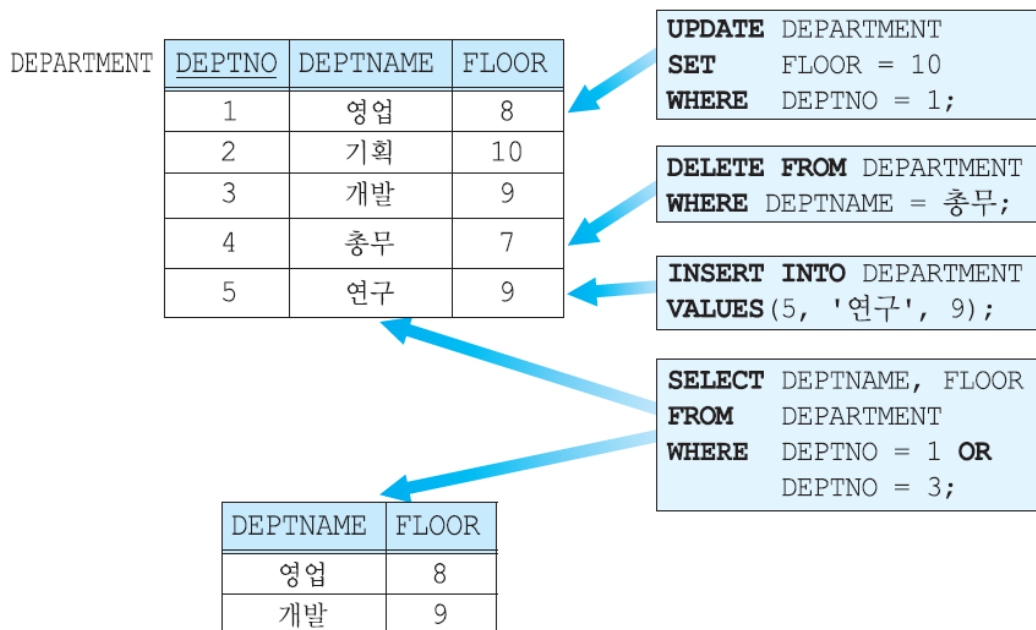
SQL 개요

□ 오라클 SQL의 구성요소

- ✓ 데이터 검색 (SELECT)
- ✓ 데이터 조작어 (DML) : INSERT, UPDATE, DELETE
- ✓ 데이터 정의어 (DDL) : CREATE, ALTER, DROP
- ✓ 트랜잭션 제어 (TCL) : COMMIT, ROLLBACK
- ✓ 데이터 제어어 (DCL) : GRANT, REVOKE

37

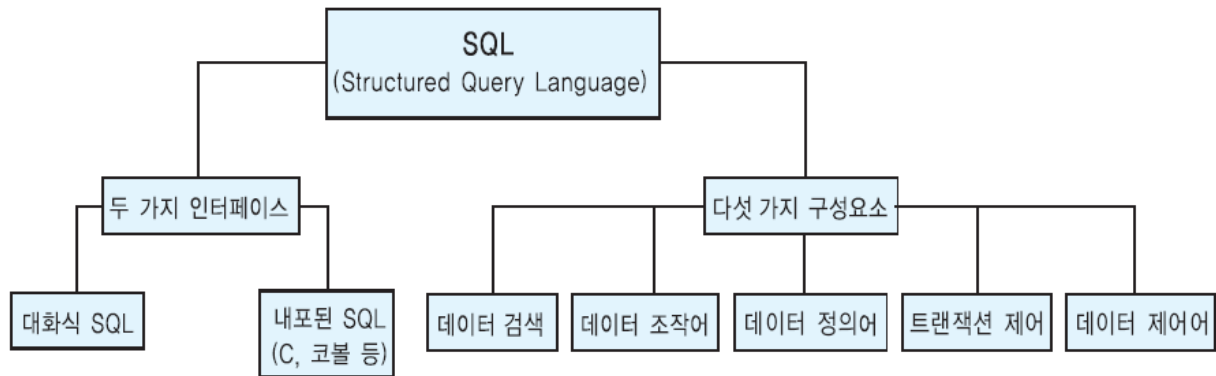
SQL 개요



[그림 3.1] 데이터 검색과 데이터 조작어의 기능

38

SQL 개요



[그림 3.2] SQL의 인터페이스와 구성요소

데이터 정의어와 무결성 제약조건

<표 3.2> 데이터 정의어의 종류

CREATE	DOMAIN	도메인을 생성
	TABLE	테이블을 생성
	VIEW	뷰를 생성
	INDEX	인덱스를 생성. SQL2의 표준이 아님
ALTER	TABLE	테이블의 구조를 변경
DROP	DOMAIN	도메인을 제거
	TABLE	테이블을 제거
	VIEW	뷰를 제거
	INDEX	인덱스를 제거. SQL2의 표준이 아님

데이터 정의어와 무결성 제약조건

□ 릴레이션 정의 (테이블 생성)

```
CREATE TABLE EMPLOYEE
    (EMPNO      NUMBER    NOT NULL,
     EMPNAME    CHAR(10) ,
     TITLE      CHAR(10) ,
     MANAGER    NUMBER,
     SALARY      NUMBER,
     DNO        NUMBER,
     PRIMARY KEY (EMPNO) ,
     FOREIGN KEY (MANAGER) REFERENCES EMPLOYEE (EMPNO) ,
     FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DEPTNO) );
```

41

데이터 정의어와 무결성 제약조건

<표 3.3> 릴레이션의 정의에 사용되는 데이터 타입

데이터 타입	의미
INTEGER 또는 INT	정수형
SMALLINT	작은 정수형
NUMBER(n, s) 또는 DECIMAL(n, s)	n개의 숫자에서 소수 아래 숫자가 s개인 십진수
REAL	실수형
FLOAT (n)	적어도 n개의 숫자가 표현되는 실수형
CHAR (n) 또는 CHARACTER (n)	n바이트 문자열. n을 생략하면 1
VARCHAR (n) 또는 CHARACTER VARYING (n)	최대 n바이트까지의 가변 길이 문자열
BIT (n) 또는 BIT VARYING (n)	n개의 비트열 또는 최대 n개까지의 가변 비트열
DATE	날짜형
BLOB	Binary Large Object. 멀티미디어 데이터 등을 저장

42

데이터 정의어와 무결성 제약조건

❑ 릴레이션 제거

```
DROP TABLE DEPARTMENT;
```

❑ ALTER TABLE

```
ALTER TABLE EMPLOYEE ADD PHONE CHAR(13);
```

❑ 인덱스 생성

```
CREATE UNIQUE INDEX EMPINDEX ON EMPLOYEE(EMPNO);
```

❑ 도메인 생성

```
CREATE DOMAIN DEPTNAME CHAR(10) DEFAULT '개발';
```

43

데이터 정의어와 무결성 제약조건

❑ 제약조건

- ✓ NOT NULL 제약조건 : (1)
- ✓ 기본키(PRIMARY KEY) 제약조건 : (6)
- ✓ 유일성(UNIQUE) 제약조건 : (2)
- ✓ 체크(CHECK) 제약조건 : (4), (5)
- ✓ 참조무결성(FOREIGN KEY) 제약조건 : (7), (8), (9)

```
CREATE TABLE EMPLOYEE
(EMPNO    NUMBER    NOT NULL,                (1)
 EMPNAME  CHAR(10)  UNIQUE,                  (2)
 TITLE    CHAR(10)  DEFAULT '사원',          (3)
 MANAGER   NUMBER,
 SALARY    NUMBER    CHECK (SALARY < 6000000), (4)
 DNO       NUMBER    CHECK (DNO IN (1,2,3,4)) DEFAULT 1, (5)
 PRIMARY KEY (EMPNO),                        (6)
 FOREIGN KEY (MANAGER) REFERENCES EMPLOYEE (EMPNO) (7)
 FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DEPTNO) (8)
 ON DELETE SET DEFAULT ON UPDATE CASCADE); (9)
```

44

데이터 정의어와 무결성 제약조건

❑ 참조 무결성 제약조건 유지

ON DELETE NO ACTION

ON DELETE CASCADE

ON DELETE SET NULL

ON DELETE SET DEFAULT

ON UPDATE NO ACTION

ON UPDATE CASCADE

ON UPDATE SET NULL

ON UPDATE SET DEFAULT

45

데이터 정의어와 무결성 제약조건

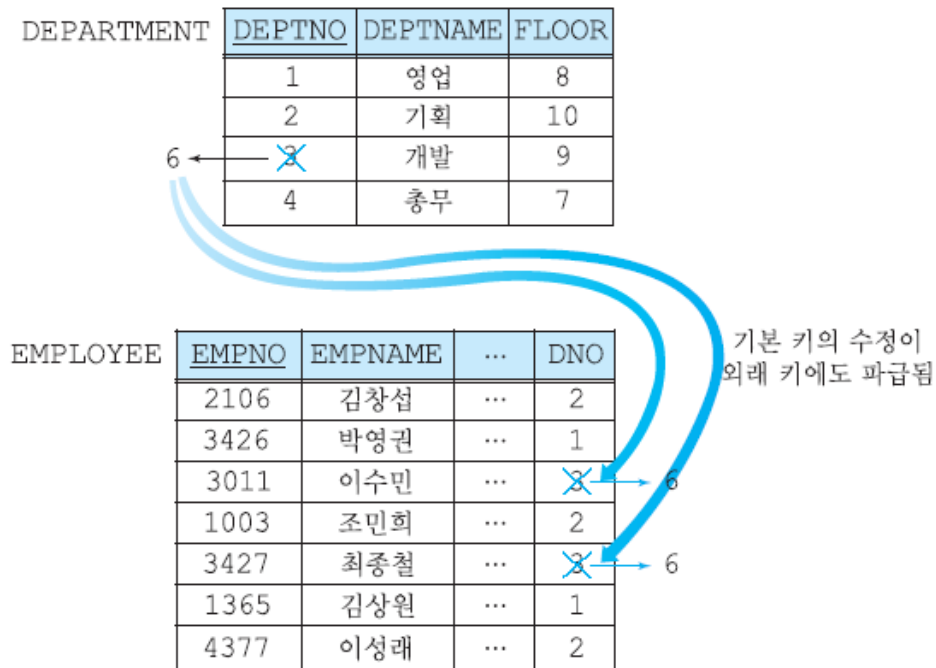
예 : ON UPDATE CASCADE

4.5절에서 설명할 UPDATE문을 사용하여 다음과 같이 DEPARTMENT 릴레이션의 3번 부서의 부서번호를 6번으로 수정하면 EMPLOYEE 릴레이션에서 3번 부서에 근무하는 모든 직원들의 소속 부서번호가 자동적으로 6으로 수정된다.

```
UPDATE DEPARTMENT
SET     DEPTNO = 6
WHERE   DEPTNO = 3;
```

46

데이터 정의어와 무결성 제약조건



[그림 3.3] 참조무결성 제약조건 유지 (UPDATE CASCADE)

47

데이터 정의어와 무결성 제약조건

□ 무결성 제약조건 của 추가 및 삭제

```
ALTER TABLE STUDENT ADD CONSTRAINT STUDENT_PK
PRIMARY KEY (STNO);
```

```
ALTER TABLE STUDENT DROP CONSTRAINT STUDENT_PK;
```

48

SELECT문

□ SELECT문

- ✓ 관계 데이터베이스에서 정보를 검색하는 SQL문
- ✓ 관계 대수의 실렉션과 의미가 완전히 다름
- ✓ 관계 대수의 실렉션, 프로젝션, 조인, 카티션 곱 등을 결합한 것
- ✓ 관계 데이터베이스에서 가장 자주 사용됨
- ✓ 여러 가지 질의들의 결과를 보이기 위해서 그림 4.8의 관계 데이터베이스 상태를 사용함

49

SELECT문

EMPLOYEE	<u>EMPNO</u>	EMPNAME	TITLE	MANAGER	SALARY	DNO
	2106	김창섭	대리	1003	2500000	2
	3426	박영권	과장	4377	3000000	1
	3011	이수민	부장	4377	4000000	3
	1003	조민희	과장	4377	3000000	2
	3427	최종철	사원	3011	1500000	3
	1365	김상원	사원	3426	1500000	1
	4377	이성래	사장	∧	5000000	2

DEPARTMENT	<u>DEPTNO</u>	DEPTNAME	FLOOR
	1	영업	8
	2	기획	10
	3	개발	9
	4	총무	7

[그림 3.4] 테이블 상태

50

SELECT문

□ 기본적인 SQL 질의

✓ **SELECT**절과 **FROM**절만 필수적인 절이고, 나머지는 선택 사항

SELECT	[DISTINCT] 애트리뷰트(들)	(1)
FROM	릴레이션(들)	(2)
[WHERE	조건	(3)
	[중첩 질의]]	(4)
[GROUP BY	애트리뷰트(들)]	(5)
[HAVING	조건]	(6)
[ORDER BY	애트리뷰트(들) [ASC DESC]] ;	(7)

SELECT문

□ 별칭(alias)

✓ 서로 다른 릴레이션에 동일한 이름을 가진 애트리뷰트가 속해 있을 때 애트리뷰트의 이름을 구분하는 방법

EMPLOYEE.DNO

FROM EMPLOYEE AS E, DEPARTMENT AS D

SELECT문

- 릴레이션의 모든 애트리뷰트나 일부 애트리뷰트들을 검색

예 : *를 사용하여 모든 애트리뷰트들을 검색

질의: 전체 부서의 모든 애트리뷰트들을 검색하라.

```
SELECT      *  
FROM        DEPARTMENT;
```

DEPTNO	DEPTNAME	FLOOR
1	영업	8
2	기획	10
3	개발	9
4	총무	7

SELECT문

예 : 원하는 애트리뷰트들의 이름을 열거

질의: 모든 부서의 부서번호와 부서이름을 검색하라.

```
SELECT      DEPTNO, DEPTNAME  
FROM        DEPARTMENT;
```

DEPTNO	DEPTNAME
1	영업
2	기획
3	개발
4	총무

SELECT문

□ 상이한 값들을 검색

예 : DISTINCT절을 사용하지 않을 때

질의: 모든 사원들의 직급을 검색하라.

```
SELECT    TITLE
FROM      EMPLOYEE;
```

TITLE
대리
과장
부장
과장
사원
사원
사장

55

SELECT문

예 : DISTINCT절을 사용할 때

질의: 모든 사원들의 상이한 직급을 검색하라.

```
SELECT    DISTINCT TITLE
FROM      EMPLOYEE;
```

TITLE
대리
과장
부장
사원
사장

56

SELECT문

□ 특정한 튜플들의 검색

예 : WHERE절을 사용하여 검색 조건을 명시

질의: 2번 부서에 근무하는 직원들에 관한 모든 정보를 검색하라.

```
SELECT      *
FROM        EMPLOYEE
WHERE       DNO = 2;
```

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
1003	조민희	과장	4377	3000000	2
2016	김창섭	대리	1003	2500000	2
4377	이성래	사장	^	5000000	2

57

SELECT문

□ 문자열 비교

예 : %를 사용하여 문자열 비교

질의: 이씨 성을 가진 직원들의 이름, 직급, 소속 부서번호를 검색하라.

```
SELECT      EMPNAME, TITLE, DNO
FROM        EMPLOYEE
WHERE       EMPNAME LIKE '이%';
```

EMPNAME	TITLE	DNO
이수민	부장	3
이성래	사장	2

58

SELECT문

❑ 다수의 검색 조건

✓ 아래와 같은 질의는 잘못되었음

```
SELECT      FLOOR
FROM        DEPARTMENT
WHERE       DEPTNAME= '영업' AND DEPTNAME= '개발' ;
```

<표 3.4> 연산자들의 우선 순위

연산자	우선순위
비교 연산자	1
NOT	2
AND	3
OR	4

SELECT문

예 : 부울 연산자를 사용한 프레디키트

질의: 직급이 과장이면서 1번 부서에서 근무하는 직원들의 이름과 급여를 검색하라.

```
SELECT      EMPNAME, SALARY
FROM        EMPLOYEE
WHERE       TITLE = '과장' AND DNO = 1;
```

EMPNAME	SALARY
박영권	3000000

SELECT문

□ 부정 검색 조건

예 : 부정 연산자

질의: 직급이 과장이면서 1번 부서에 속하지 않은 사원들의 이름과 급여를 검색하라.

```
SELECT    EMPNAME, SALARY
FROM      EMPLOYEE
WHERE     TITLE = '과장' AND DNO <> 1;
```

EMPNAME	SALARY
조민희	3000000

61

SELECT문

□ 범위를 사용한 검색

예 : 범위 연산자

질의: 급여가 3000000원 이상이고, 4500000원 이하인 사원들의 이름, 직급, 급여를 검색하라.

```
SELECT    EMPNAME, TITLE, SALARY
FROM      EMPLOYEE
WHERE     SALARY BETWEEN 3000000 AND 4500000;
```

BETWEEN은 양쪽의 경계값을 포함하므로 이 질의는 아래의 질의와 동등하다.

```
SELECT    EMPNAME, TITLE, SALARY
FROM      EMPLOYEE
WHERE     SALARY >= 3000000 AND SALARY <= 4500000;
```

EMPNAME	TITLE	SALARY
박영권	과장	3000000
이수민	부장	4000000
조민희	과장	3000000

62

SELECT문

□ 리스트를 사용한 검색

예 : IN

질의: 1번 부서나 3번 부서에 소속된 직원들에 관한 모든 정보를 검색하라.

```
SELECT      *
FROM        EMPLOYEE
WHERE       DNO IN (1, 3);
```

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
1365	김상원	사원	3426	1500000	1
3011	이수민	부장	4377	4000000	3
3426	박영권	과장	4377	3000000	1
3427	최종철	사원	3011	1500000	3

63

SELECT문

□ SELECT절에서 산술 연산자(+, -, *, /) 사용

예 : 산술 연산자

질의: 직급이 과장인 직원들에 대하여 이름과, 현재의 급여, 급여가 10% 인상됐을 때의 값을 검색하라.

```
SELECT      EMPNAME, SALARY, SALARY * 1.1 AS NEWSALARY
FROM        EMPLOYEE
WHERE       TITLE = '과장';
```

EMPNAME	SALARY	NEWSALARY
박영권	3000000	3300000
조민희	3000000	3300000

64

SELECT문

□ 널값

- ✓ 널값을 포함한 다른 값과 널값을 +, - 등을 사용하여 연산하면 결과는 널
- ✓ COUNT(*)를 제외한 집단 함수들은 널값을 무시함
- ✓ 어떤 애트리뷰트에 들어 있는 값이 널인가 비교하기 위해서 'DNO=NULL' 처럼 나타내면 안됨

```
SELECT  EMPNO, EMPNAME
FROM    EMPLOYEE
WHERE   DNO = NULL;
```

SELECT문

□ 널값(계속)

- ✓ 다음과 같은 비교 결과는 모두 거짓

NULL > 300

NULL = 300

NULL <> 300

NULL = NULL

NULL <> NULL

- ✓ 올바른 표현

```
SELECT  EMPNO, EMPNAME
FROM    EMPLOYEE
WHERE   DNO IS NULL;
```

SELECT문

<표 3.5> unknown에 대한 OR 연산

	true	false	unknown
true	true	true	true
false	true	false	unknown
unknown	true	unknown	unknown

<표 3.6> unknown에 대한 AND 연산

	true	false	unknown
true	true	false	unknown
false	false	false	false
unknown	unknown	false	unknown

<표 3.7> unknown에 대한 NOT 연산

true	false
false	true
unknown	unknown

67

SELECT문

□ ORDER BY절

- ✓ 사용자가 SELECT문에서 질의 결과의 순서를 명시하지 않으면 기본 키의 값이 증가하는 순서대로 사용자에게 제시됨
- ✓ ORDER BY절에서 하나 이상의 애트리뷰트를 사용하여 검색 결과를 정렬할 수 있음
- ✓ ORDER BY절은 SELECT문에서 가장 마지막에 사용되는 절
- ✓ 디폴트 정렬 순서는 오름차순(ASC)
- ✓ DESC를 지정하여 정렬 순서를 내림차순으로 지정할 수 있음
- ✓ 널값은 오름차순에서는 가장 마지막에 나타나고, 내림차순에서는 가장 앞에 나타남
- ✓ SELECT절에 명시한 애트리뷰트들을 사용해서 정렬해야 함

68

SELECT문

예 : ORDER BY

질의: 2번 부서에 근무하는 사원들의 급여, 직급, 이름을 검색하여 급여의 오름차순으로 정렬하라.

```
SELECT    SALARY, TITLE, EMPNAME
FROM      EMPLOYEE
WHERE     DNO = 2
ORDER BY  SALARY;
```

SALARY	TITLE	EMPNAME
2500000	대리	김창섭
3000000	과장	조민희
5000000	사장	이성래

69

SELECT문

□ 집단 함수

- ✓ 데이터베이스에서 검색된 여러 튜플들의 집단에 적용되는 함수
- ✓ 각 집단 함수는 한 릴레이션의 한 개의 애트리뷰트에 적용되어 단일 값을 반환함
- ✓ SELECT절과 HAVING절에만 나타날 수 있음
- ✓ COUNT(*)를 제외하고는 모든 집단 함수들이 널값을 제거한 후 남아 있는 값들에 대해서 집단 함수의 값을 구함
- ✓ COUNT(*)는 결과 릴레이션의 모든 행들의 총 개수를 구하는 반면에 COUNT(애트리뷰트)는 해당 애트리뷰트에서 널값이 아닌 값들의 개수를 구함
- ✓ 키워드 DISTINCT가 집단 함수 앞에 사용되면 집단 함수가 적용되기 전에 먼저 중복을 제거함

70

SELECT문

<표 3.8> 집단 함수 종류

집단 함수	기능
COUNT	튜플이나 값들의 개수
SUM	값들의 합
AVG	값들의 평균값
MAX	값들의 최대값
MIN	값들의 최소값

SELECT문

예 : 집단 함수

질의: 모든 사원들의 평균 급여와 최대 급여를 검색하라.

```
SELECT    AVG (SALARY) AS AVGSAL, MAX (SALARY) AS MAXSAL
FROM      EMPLOYEE;
```

AVGSAL	MAXSAL
2928571	5000000

SELECT문

□ 그룹화

- ✓ GROUP BY절에 사용된 애트리뷰트에 동일한 값을 갖는 튜플들이 각각 하나의 그룹으로 묶임
- ✓ 이때 사용된 애트리뷰트를 **그룹화 애트리뷰트**(grouping attribute)라고 함
- ✓ 각 그룹에 대하여 결과 릴레이션에 하나의 튜플이 생성됨
- ✓ SELECT절에는 각 그룹마다 하나의 값을 갖는 애트리뷰트, 집단 함수, 그룹화에 사용된 애트리뷰트들만 나타날 수 있음
- ✓ 다음 질의는 그룹화를 하지 않은 채 EMPLOYEE 릴레이션의 모든 튜플에 대해서 사원번호와 모든 사원들의 평균 급여를 검색하므로 잘못됨

```
SELECT  EMPNO, AVG(SALARY)
FROM    EMPLOYEE;
```

73

SELECT문

예 : 그룹화

질의: 모든 사원들에 대해서 사원들이 속한 부서번호별로 그룹화하고, 각 부서마다 부서번호, 평균 급여, 최대 급여를 검색하라.

```
SELECT  DNO, AVG(SALARY) AS AVGSAL, MAX(SALARY) AS MAXSAL
FROM    EMPLOYEE
GROUP BY DNO;
```

74

SELECT문

EMPLOYEE

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
3426	박영권	과장	4377	3000000	1
1365	김상원	사원	3426	1500000	1
2106	김창섭	대리	1003	2500000	2
1003	조민희	과장	4377	3000000	2
4377	이성래	사장	^	5000000	2
3011	이수민	부장	4377	4000000	3
3427	최종철	사원	3011	1500000	3

→ 그룹

DNO	AVGSAL	MAXSAL
1	2250000	3000000
2	3500000	5000000
3	2750000	4000000

[그림 3.4] 그룹화

SELECT문

□ HAVING

- ✓ 어떤 조건을 만족하는 그룹들에 대해서만 집단 함수를 적용할 수 있음
- ✓ 각 그룹마다 하나의 값을 갖는 애트리뷰트를 사용하여 각 그룹이 만족해야 하는 조건을 명시함
- ✓ HAVING절은 그룹화 애트리뷰트에 같은 값을 갖는 튜플들의 그룹에 대한 조건을 나타내고, 이 조건을 만족하는 그룹들만 질의 결과에 나타남
- ✓ HAVING절에 나타나는 애트리뷰트는 반드시 GROUP BY절에 나타나거나 집단 함수에 포함되어야 함

SELECT문

예 : 그룹화

질의: 모든 사원들에 대해서 사원들이 속한 부서번호별로 그룹화하고, 평균 급여가 2500000원 이상인 부서에 대해서 부서번호, 평균 급여, 최대 급여를 검색하라.

```
SELECT DNO, AVG (SALARY) AS AVGSAL, MAX (SALARY) AS MAXSAL
FROM EMPLOYEE
GROUP BY DNO
HAVING AVG (SALARY) >= 2500000;
```

77

SELECT문

EMPLOYEE	EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
	3426	박영권	과장	4377	3000000	1
	1365	김상원	사원	3426	1500000	1
	2106	김창섭	대리	1003	2500000	2
	1003	조민희	과장	4377	3000000	2
	4377	이성래	사장	^	5000000	2
	3011	이수민	부장	4377	4000000	3
	3427	최종철	사원	3011	1500000	3

GROUP BY



DNO	AVGSAL	MAXSAL
1	2250000	3000000
2	3500000	5000000
3	2750000	4000000

HAVING



DNO	AVGSAL	MAXSAL
2	3500000	5000000
3	2750000	4000000

[그림 3.5] HAVING에 의한 그룹 필터링

78

SELECT문

□ 집합 연산

- ✓ 집합 연산을 적용하려면 두 릴레이션이 합집합 호환성을 가져야 함
- ✓ UNION(합집합), EXCEPT(차집합), INTERSECT(교집합), UNION ALL(합집합), EXCEPT ALL(차집합), INTERSECT ALL(교집합)

79

SELECT문

예 : 합집합

질의: 김창섭이 속한 부서이거나 개발 부서의 부서번호를 검색하라.

```
(SELECT      DNO
FROM      EMPLOYEE
WHERE      EMPNAME = '김창섭' )
UNION
(SELECT      DEPTNO
FROM      DEPARTMENT
WHERE      DEPTNAME = '개발' );
```

DNO
2
3

80

SELECT문

□ 조인

- ✓ 조인은 두 개의 릴레이션으로부터 연관된 튜플들을 결합
- ✓ 조인의 일반적인 형식은 아래의 SELECT문과 같이 FROM절에 두 개 이상의 릴레이션들이 열거되고, 두 릴레이션에 속하는 애트리뷰트들을 비교하는 조인 조건이 WHERE절에 포함됨
- ✓ 조인 조건은 두 릴레이션 사이에 속하는 애트리뷰트 값들을 비교 연산자로 연결한 것
- ✓ 가장 흔히 사용되는 비교 연산자는 =

```
SELECT    ...  
FROM      R, S  
WHERE     R.A <비교 연산자> S.B ;
```

조인 조건

81

SELECT문

□ 조인(계속)

- ✓ 조인 조건을 생략했을 때와 조인 조건을 틀리게 표현했을 때는 카티션 곱이 생성됨
- ✓ 조인 질의가 수행되는 과정을 개념적으로 살펴보면 먼저 조인 조건을 만족하는 튜플들을 찾고, 이 튜플들로부터 SELECT절에 명시된 애트리뷰트들만 프로젝트하고, 필요하다면 중복을 배제하는 순서로 진행됨
- ✓ 조인 조건이 명확해지도록 애트리뷰트 이름 앞에 릴레이션 이름이나 튜플 변수를 사용하는 것이 바람직
- ✓ 두 릴레이션의 조인 애트리뷰트 이름이 동일하다면 반드시 애트리뷰트 이름 앞에 릴레이션 이름이나 튜플 변수를 사용해야 함

82

SELECT문

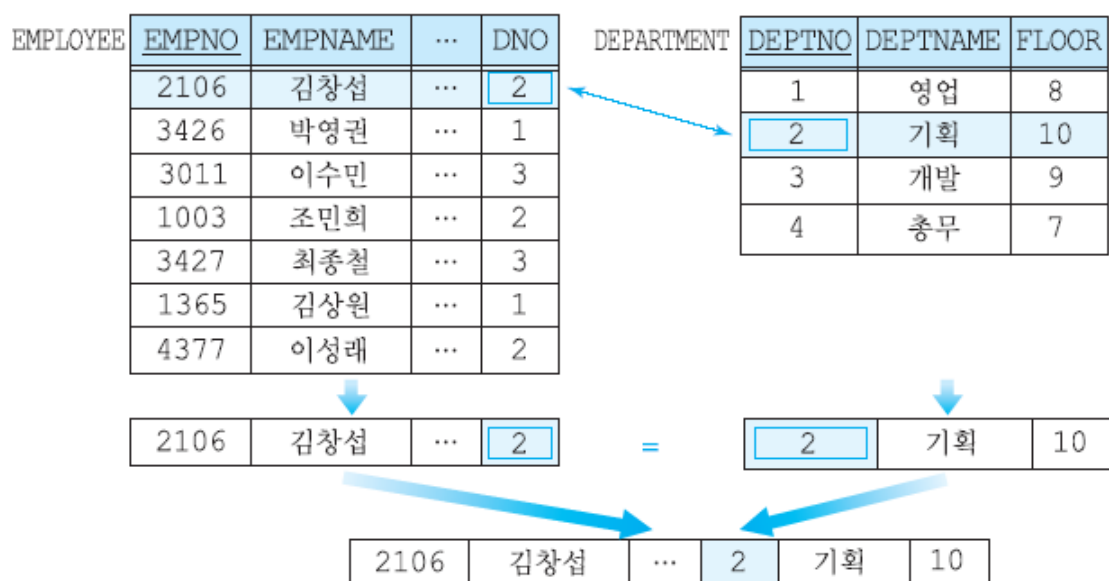
예 : 조인 질의

질의: 모든 사원의 이름과 이 사원이 속한 부서 이름을 검색하라.

```
SELECT      EMPNAME, DEPTNAME
FROM        EMPLOYEE AS E, DEPARTMENT AS D
WHERE       E.DNO = D.DEPTNO;
```

83

SELECT문



[그림 3.6] 조인(join)

84

SELECT문

최종 결과 릴레이션은 아래와 같다.

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO	DEPTNAME	FLOOR
1003	조민희	과장	4377	3000000	2	기획	10
1365	김상원	사원	3426	1500000	1	영업	8
2106	김창섭	대리	1003	2500000	2	기획	10
3011	이수민	부장	4377	4000000	3	개발	9
3426	박영권	과장	4377	3000000	1	영업	8
3427	최종철	사원	3011	1500000	3	개발	9
4377	이성래	사장	∧	5000000	2	기획	10

85

SELECT문

□ 자체 조인(self join)

- ✓ 한 릴레이션에 속하는 튜플을 동일한 릴레이션에 속하는 튜플들과 조인하는 것
- ✓ 실제로는 한 릴레이션이 접근되지만 FROM절에 두 릴레이션이 참조되는 것처럼 나타내기 위해서 그 릴레이션에 대한 별칭을 두 개 지정해야 함

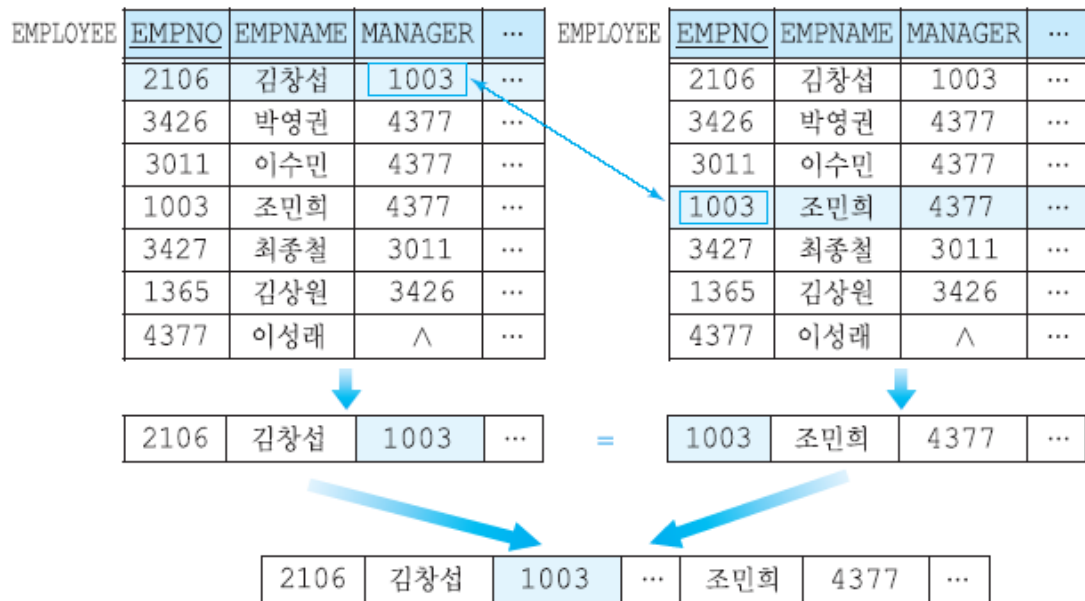
예 : 자체 조인

질의: 모든 사원에 대해서 사원의 이름과 직속 상사의 이름을 검색하라.

```
SELECT      E.EMPNAME, M.EMPNAME
FROM        EMPLOYEE E, EMPLOYEE M
WHERE       E.MANAGER = M.EMPNO;
```

86

SELECT문



[그림 3.7] 자체 조인(self join)

87

SELECT문

최종 결과 릴레이션은 아래와 같다.

E.EMPNAME	M.EMPNAME
김창섭	조민희
박영권	이성래
이수민	이성래
조민희	이성래
최종철	이수민
김상원	박영권

88

SELECT문

예 : 조인과 ORDER BY의 결합

질의: 모든 사원에 대해서 소속 부서이름, 사원의 이름, 직급, 급여를 검색하라. 부서 이름에 대해서 오름차순, 부서이름이 같은 경우에는 SALARY에 대해서 내림차순으로 정렬하라.

```
SELECT DEPTNAME, EMPNAME, TITLE, SALARY
FROM EMPLOYEE E, DEPARTMENT D
WHERE E.DNO = D.DEPTNO
ORDER BY DEPTNAME, SALARY DESC;
```

DEPTNAME	EMPNAME	TITLE	SALARY	
개발	이수민	부장	4000000	
개발	최종철	사원	1500000	내림차순
기획	이성래	사장	5000000	
기획	조민희	과장	3000000	내림차순
기획	김창섭	대리	2500000	
영업	박영권	과장	3000000	
영업	김상원	사장	1500000	내림차순

89

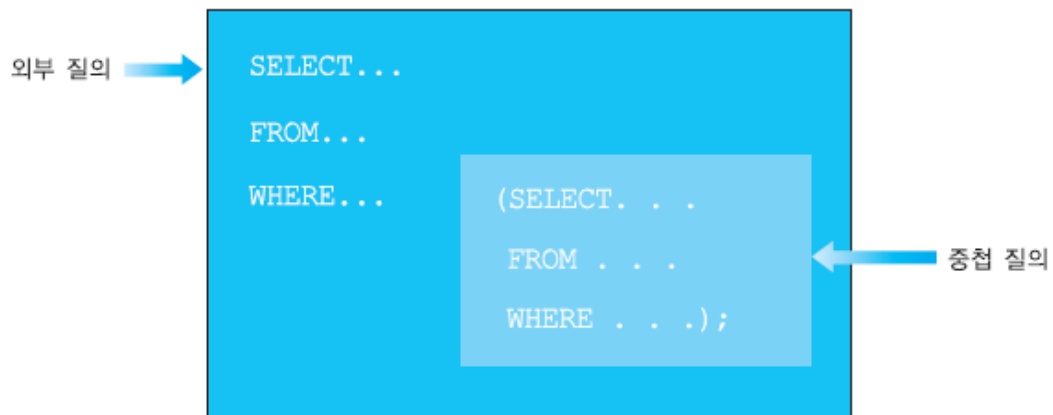
SELECT문

❑ 부질의(sub-query)

- ✓ 외부 질의의 WHERE절에 다시 SELECT ... FROM ... WHERE 형태로 포함된 SELECT문
- ✓ 중첩질의(nested query)라고 함
- ✓ INSERT, DELETE, UPDATE문에도 사용될 수 있음
- ✓ 중첩 질의의 결과로 한 개의 스칼라값(단일 값), 한 개의 애트리뷰트로 이루어진 릴레이션, 여러 애트리뷰트로 이루어진 릴레이션이 반환될 수 있음

90

SELECT문



[그림 3.8] 부질의(sub-query)

SELECT문

□ 한 개의 스칼라값이 반환되는 경우

예 : 한개의 스칼라 값이 반환되는 경우

질의: 박영권과 같은 직급을 갖는 모든 직원들의 이름과 직급을 검색하라.

```
SELECT EMPNAME, TITLE
FROM EMPLOYEE
WHERE TITLE = (SELECT TITLE
                FROM EMPLOYEE
                WHERE EMPNAME = '박영권') ;
```

중첩 질의

과장

EMPNAME	TITLE
박영권	과장
조민희	과장

SELECT문

예 : IN을 사용한 질의

질의: 영업부나 개발부에 근무하는 직원들의 이름을 검색하라.

```

SELECT  EMPNAME
FROM    EMPLOYEE
WHERE   DNO IN (1, 3)

```

(SELECT DEPTNO
 FROM DEPARTMENT
 WHERE DEPTNAME = '영업' OR DEPTNAME = '개발') ;

93

SELECT문

이 질의를 중첩 질의를 사용하지 않은 다음과 같은 조인 질의로 나타낼 수 있다. 실제로, 중첩 질의를 사용하여 표현된 대부분의 질의를 중첩 질의가 없는 조인 질의로 표현할 수 있다.

```

SELECT  EMPNAME
FROM    EMPLOYEE E, DEPARTMENT D
WHERE   E.DNO = D.DEPTNO
        AND (D.DEPTNAME = '영업' OR D.DEPTNAME = '개발') ;

```

EMPNAME
박영권
이수민
최종철
김상원

94

SELECT문

❑ 여러 애트리뷰트들로 이루어진 릴레이션이 반환되는 경우

- ✓ 중첩 질의의 결과로 여러 애트리뷰트들로 이루어진 릴레이션이 반환되는 경우에는 EXISTS 연산자를 사용하여 중첩 질의의 결과가 빈 릴레이션인지 여부를 검사함
- ✓ 중첩 질의의 결과가 빈 릴레이션이 아니면 참이 되고, 그렇지 않으면 거짓

95

SELECT문

예 : EXISTS를 사용한 질의

질의: 영업부나 개발부에 근무하는 직원들의 이름을 검색하라.

```
SELECT EMPNAME
FROM EMPLOYEE E
WHERE EXISTS
  (SELECT *
   FROM DEPARTMENT D
   WHERE E.DNO = D.DEPTNO
   AND (DEPTNAME = '영업' OR DEPTNAME = '개발'));
```

EMPNAME
박영권
이수민
최종철
김상원

96

INSERT, DELETE, UPDATE문

□ INSERT문

- ✓ 기존의 릴레이션에 튜플을 삽입
- ✓ 참조되는 릴레이션에 튜플이 삽입되는 경우에는 참조 무결성 제약조건의 위배가 발생하지 않으나 참조하는 릴레이션에 튜플이 삽입되는 경우에는 참조 무결성 제약조건을 위배할 수 있음
- ✓ 릴레이션에 한 번에 한 튜플씩 삽입하는 것과 한 번에 여러 개의 튜플들을 삽입할 수 있는 것으로 구분
- ✓ 릴레이션에 한 번에 한 튜플씩 삽입하는 INSERT문

```
INSERT  
INTO    릴레이션(애틀리뷰트1, ..., 애틀리뷰트n)  
VALUES (값1, ..., 값n);
```

INSERT, DELETE, UPDATE문

예 : 한 개의 튜플을 삽입

질의: DEPARTMENT 릴레이션에 (5, 연구, 0) 튜플을 삽입하는 INSERT문은 아래와 같다.

```
INSERT INTO DEPARTMENT  
VALUES (5, '연구', 0);
```

DEPARTMENT	DEPTNO	DEPTNAME	FLOOR
	1	영업	8
	2	기획	10
	3	개발	9
	4	총무	7
	5	연구	0

INSERT, DELETE, UPDATE문

□ INSERT문(계속)

- ✓ 릴레이션에 한 번에 여러 개의 튜플들을 삽입하는 INSERT문

```
INSERT
  INTO    릴레이션 (애틀리뷰트1, ..., 애틀리뷰트n)
  SELECT  ... FROM    ... WHERE  ...;
```

예 : 여러 개의 튜플을 삽입

질의: EMPLOYEE 릴레이션에서 급여가 3000000 이상인 사원들의 이름, 직급, 급여를 검색하여 HIGH_SALARY라는 릴레이션에 삽입하라. HIGH_SALARY 릴레이션은 이미 생성되어 있다고 가정한다.

```
INSERT INTO HIGH_SALARY (ENAME, TITLE, SAL)
SELECT EMPNAME, TITLE, SALARY
FROM EMPLOYEE
WHERE SALARY >= 3000000;
```

99

INSERT, DELETE, UPDATE문

□ DELETE문

- ✓ 삭제 연산은 한 릴레이션으로부터 한 개 이상의 튜플들을 삭제함
- ✓ 참조되는 릴레이션의 삭제 연산의 결과로 참조 무결성 제약조건이 위배될 수 있으나, 참조하는 릴레이션에서 튜플을 삭제하면 참조 무결성 제약조건을 위배하지 않음
- ✓ DELETE문의 구문

```
DELETE
  FROM    릴레이션
  WHERE   조건;
```

100

INSERT, DELETE, UPDATE문

예 : DELETE문

질의: DEPARTMENT 릴레이션에서 4번 부서를 삭제하라.

```
DELETE FROM DEPARTMENT  
WHERE DEPTNO = 4;
```

101

INSERT, DELETE, UPDATE문

□ UPDATE문

- ✓ 한 릴레이션에 들어 있는 튜플들의 애트리뷰트 값들을 수정
- ✓ 기본 키나 외래 키에 속하는 애트리뷰트의 값이 수정되면 참조 무결성 제약조건을 위반할 수 있음
- ✓ UPDATE문의 구문

```
UPDATE   릴레이션  
SET      애트리뷰트 = 값 또는 식[, ...]  
WHERE    조건;
```

예 : UPDATE문

질의: 사원번호가 2106인 사원의 소속 부서를 3번 부서로 옮기고, 급여를 5% 올려라.

```
UPDATE EMPLOYEE  
SET     DNO = 3, SALARY = SALARY * 1.05  
WHERE   EMPNO = 2106;
```

102