SQL (Structured Query Language) 개요

스키마와 카탈로그의 개념

➢ SQL schema

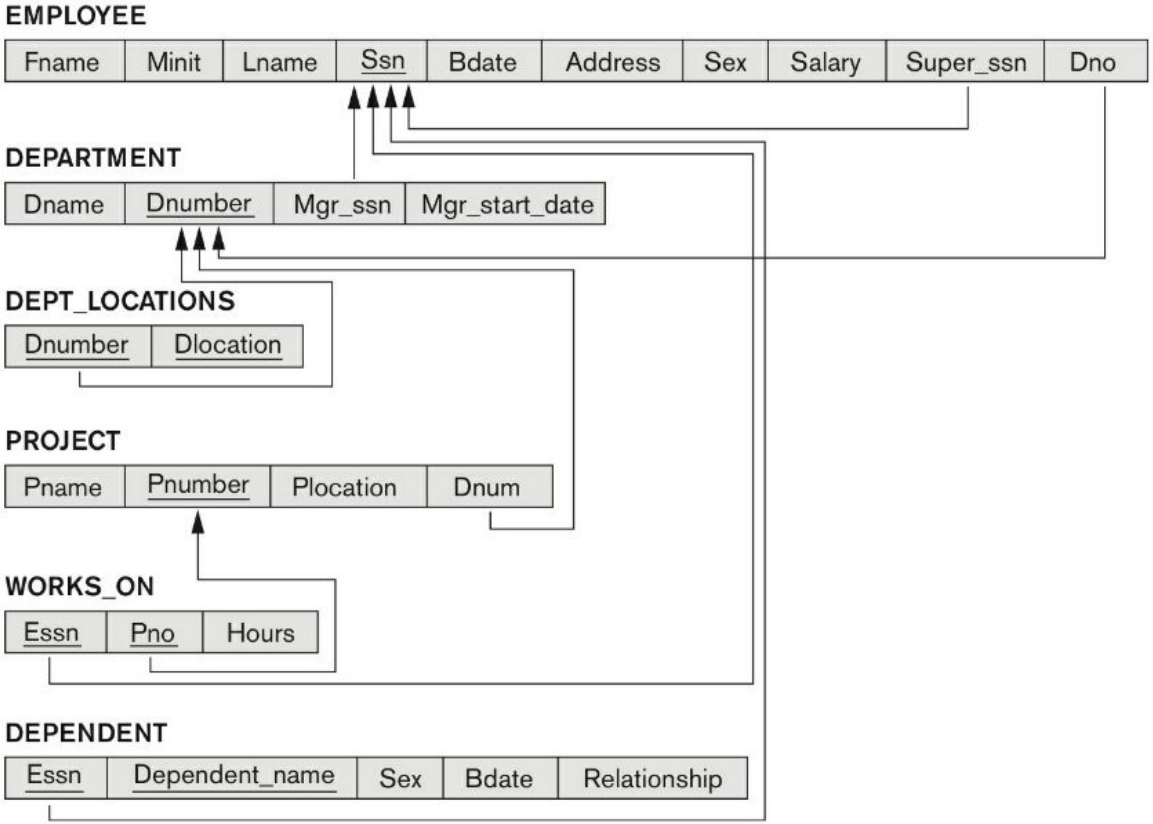
- 스키마 이름으로 식별됨
  - 권한 부여 식별자: 스키마를 소유하는 사용자나 계정
  - 각 원소에 대한 기술자 (descriptor): 테이블, 제약조건, 뷰 도메인 등
- **CREATE SCHEMA**
  - 스키마를 생성: 생성을 할 수 있는 권한은 시스템관리자나 DBA에 의해 허락된 사용자 계정에게만 명시적으로 허가해야 함
  - 추가적으로 이름, 권한부여 식별자를 할당받고 나중에 원소들 정의 가능

```
CREATE SCHEMA COMPANY.AUTHORIZATION ‘Jsmith’;
```

➢ 카탈로그 (catalog)

- 스키마들의 모임으로서 이름을 가짐
- 데이터 베이스 설치시에 디폴트 환경과 스키마를 가짐
  - **INFORMATION_SCHEMA**: 카탈로그 내에 있는 모든 스키마들과 이들 내의 원소 기술문제들에 대한 정보를 제공
- 참조 무결성과 같은 무결성 제약조건은 동일한 카탈로그 내의 스카마들 안에 존재하는 릴레이션 사이에서만 정의 가능

COMPANY relational database schema



COMPANY database state 예제

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

COMPANY database state 예제

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

SQL의 CREATE TABLE 명령

➢ CREATE TABLE

- 새로운 릴레이션을 구체화
  - table 의 이름 제시
  - 애트리뷰트, 애트리뷰트 타입, 초기 제약 조건 명시
- 스키마: COMPANY, 릴레이션: EMPLOYEE 의 경우
  - `CREATE TABLE COMPANY.EMPLOYEE …`
  - `CREATE TABLE EMPLOYEE …`
- 기본 테이블 혹은 릴레이션 (base tables or base relations)
  - `CREATE TABLE` 명령어를 통해서 정의된 릴레이션
  - 기본 테이블의 SQL 애트리뷰트는 `CREATE TABLE` 명령에 명시되는 순서대로 순서를 갖지만, 행 (투플)은 릴레이션 내에서 순서를 갖지 않음
- 가상 테이블 혹은 릴레이션 (virtual tables or virtual relations)
  - `CREATE VIEW` 명령을 통해 **생성되는** 릴레이션
- 제약조건들은 `CREATE TABLE` 문에서는 포함시키지 않고 릴레이션들을 다 생성한 후에 `ALTER TABLE` 문을 사용해서 나중에 추가하는 것이 바람직함

SQL 에서 애트리뷰트 데이터 타입과 도메인

➢ 기본 데이터 타입

- 수치형 데이터 타입 (numeric)
  - 정수: INTEGER, INT, SMALLINT
  - 부동 소수 점수 (실수): FLOAT, REAL, DOUBLE PRECISION
- 문자형 데이터 타입 (character string)
  - 고정 길이 (문자 수 n): CHAR(n), CHARACTER(n)
  - 가변 길이 (최대 문자 수 n): VARCHAR(n), CHAR VARYING (n), CHARACTER VARYING (n)
- 비트열 (bit-string)
  - 고정 길이: BIT(n)
  - 가변 길이 (최대 비트 수): BITVARYING(n)
- 불리언 (boolean)
  - 참 또는 거짓 (TRUE or FALSE), 널 값 (UNKNOWN)
- 날짜 (DATE)
  - 10개의 자리 수: YEAR, MONTH, DAY 을 구성 요소로 가짐 (형식 예: YYYY-MM-DD)
  - TIME 은 최소한 8개의 자릿수를 가짐: HOUR, MINUTE, SECOND (형식 예: HH:MM:SS)
  - **RDBMS** 의 날짜 형식 변경을 위한 여러 함수들이 존재
- 타임스탬프 (TIMESTAMP)

SQL 에서 애트리뷰트 데이터 타입과 도메인

➢ 기본 데이터 타입

- 수치형 데이터 타입 (numeric)
- 문자형 데이터 타입 (character string)
- 비트열 (bit-string)
- 불리언 (boolean)
- 날짜 (DATE)
- 타임스탬프(TIMESTAMP)
  - DATE 와 TIME 필드를 포함 (6자리의 초의 소수점 이하 부분 표시)
  - 선택적으로 WITH TIME ZONE 를 추가로 포함 가능
  - TIMESTAMP '2014-09-27 09:12:47.648302'

➢ 도메인

- 도메인 선언 가능: 도메인을 선언한 후에 해당 도메인 이름을 애트리뷰트 선언에 이용할 수 있음
  - CREATE DOMAIN SSN_TYPE AS CHAR(9);
- 사용자 정의 타입 생성 가능
  - 생성된 사용자 정의 타입들은 애트리뷰트를 위한 데이터 타입 또는 테이블을 생성하기 위한 기초로 사용 가능
  - CREATE TYPE

SQL에서 기본 제약 조건의 명시

➢ 기본 제약조건

- 관계형 모델은 3가지 기본 제약조건 타입을 가지고 있음 (SQL 에서 서포트)
  - 키 제약조건: 기본 키 값은 중복될 수 없다.
  - 엔티티 무결성 제약조건: 기본키 값은 NULL 일 수 없다.
  - 참조 무결성 제약조건: 외래키는 기본키에 이미 존재하는 값이거나 NULL rkqtdmf rkwudi gksek.

➢ 애트리뷰트 제약 조건과 디폴트 값 명시

- 널 제약 조건
  - SQL은 애트리뷰트 값으로 널을 허용하기 때문에 어떤 애트리뷰트에 널 값이 허용되지 않는다면 제약조건 NOT NULL 을 명시
  - 기본키 애트리뷰트에는 반드시 NOT NULL 을 명시
- 디폴드 값 명시
  - 애트리뷰트의 디폴트 값을 명시할 수 있음
  - 애트리뷰트 정의에 ‘ DEFAULT <값> ’
- 애트리뷰트나 도메인 값 제한
  - 애트리뷰트나 도메인 정의 뒤에 따라오는 CHECK 절을 이용
  - `Dnumber INT NOT NULL CHECK (Dnumber >0 AND Dnumber <21);`
  - `CREATE DOMAIN D_NUM AS INTEGER CHECK (D_NUM >0 AND D_NUM <21);`

SQL에서 기본 제약 조건의 명시

➢ 키와 참조 무결성 제약 조건의 명시

- 키나 참조 무결성 제약조건을 매우 중요하므로 **CREATE TABLE** 구문에 키와 참조 무결성을 위한 특별한 절을 가짐
- 기본키 명시 (**PRIMARY KEY**)
  - Dnumber INT **PRIMARY KEY;**
- 다른 후보키들 (대체키 혹은 유일키) 명시 (**UNIQUE**)
  - Dname VARCHAR(15) **UNIQUE;**
- 외래 키에 대한 참조 무결성을 명시 (FOREIGN KEY)
  - 위반 시에 디폴트 동작: 위반을 초래한 갱신 연산을 거절 (reject)
  - 참조 트리거된 동작절(referential traiggered action)을 추가 가능: 위반된 경우 취할 다른 동작 명시
    - SET NULL, CASCADE, SET DEFAULT
    - 위반된 경우를 가리키는 ON DELETE 나 ON UPDATE와 함께 사용

➢ 제약 조건에 이름 부여

- 키워드 CONSTRAINT 다음에 제약 조건의 이름 부여 가능 (이름 부여는 선택적)
- 특정 스키마 내의 모든 제약 조건의 이름은 **유일해야 함**

디폴트 애트리뷰트 값과 참조 트리거된 동작 절

```
CREATE TABLE EMPLOYEE
    ( ... ,
      Dno          INT            NOT NULL       DEFAULT 1,
    CONSTRAINT EMPPK
      PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
      FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
                    ON DELETE SET NULL          ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
      FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                    ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPARTMENT
    ( ... ,
      Mgr_ssn CHAR(9)          NOT NULL       DEFAULT '888665555',
      ... ,
    CONSTRAINT DEPTPK
      PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
      UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
      FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                    ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPT_LOCATIONS
    ( ... ,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
                    ON DELETE CASCADE           ON UPDATE CASCADE);
```

SQL에서 기본 제약 조건의 명시

➢ CHECK 를 사용하여 튜플에 제약 조건 명시

- **CREATE TABLE** 뒤 쪽에 **CHECK** 절을 사용하여 제약조건을 명시하면, 이러한 제약 조건은 행이 삽입이나 삭제될 때마다 검사됨
  - 각 행에 개별적으로 적용되므로 행 제약 조건이라고도 함
- **CREATE TABLE** DEPARTMENT … **CHECK** (Dep_create_date <= mgr_start_date);
- **CHECK** 절은 SQL 의 **CREATE ASSERTION** 구문을 사용하여 보다 일반적인 제약 조건을 명시하는데 사용될 수 있음

SQL 의 기본 검색 질의

➢ SELECT statement

- 데이터베이스에서 정보를 검색하는 기본문
- SQL 모델과 관계 모델의 중요한 차이점
  - SQL의 테이블(릴레이션)은 모든 애트리뷰트 값이 동일한 투플을 하나 이상 가지는 것을 허용함
  - SQL 테이블은 투플들의 집합이 아니고 투플들의 다중집합(multiset, **bag**)임
- 따라서, 키 제약 조건을 선언하거나 SELECT 문에 DISTINCT 옵션을 함께 사용하면 SQL이 집합의 성질을 갖도록 할 수 있음

➢ 기본 SQL 질의의 SELECT-FROM-WHERE 구조

- SELECT <애트리뷰트 리스트>
- FROM <테이블 리스트>
- WHERE <조건>
  - <애트리뷰트 리스트>: 질의에서 검색되는 값들에 대한 속성 이름들의 리스트
  - <테이블 리스트>: 질의 처리를 위해 필요한(질의의 대상이 되는) 릴레이션들의 리스트
  - <조건>: 질의를 통해 검색되는 투플들을 명시하는 조건(부울)식
- 비교 연산자
  - SQL: =, <, <=, >, >=, <> | C나 C++: =, <, <=, >, >=, !=

SQL 의 기본 검색 질의

➢ 기본 SQL 질의의 SELECT-FROM-WHERE 구조

- SELECT <애트리뷰트 리스트>
- FROM <테이블 리스트>
- WHERE <조건>
- 비교 연산자: [SQL] =, <, <=, >, >=, <> | [C나 C++] =, <, <=, >, >=, !=
- 프로젝션 애트리뷰트 (projection attribute)
  - SELECT 절에서 검색할 애트리뷰트를 명시
- 선택 조건 (selection condition)
  - WHERE 절은 검색될 투플들이 만족해야 하는 불리언 조건을 명시
- 조인 조건 (join condition)
  - 서로 다른 릴레이션의 투플들의 애트리뷰트들이 같은 두 투플을 결합하는 것을 표시

SQL 질의 예제

[질의 0] 이름이 **'john B. Smitth'**인 사원(들)의 생년월일(**Bdate**)과 주소(**Adress**)를 검색하라.
질의는 **FROM** 절에 표시된 **EMPLOYEE** 릴레이션만 검색한다

[질의 1] **'Research'** 부서에서 근무하는 모든 사원의 이름(**Fname, Lname**)과 주소(**Address**)를 검색하라.

[질의 2] **'Stafford'**에 위치한 모든 프로젝트 들에 대해서 프로젝트 번호(**Pnumber**), 담당부서 번호(**Dnum**), 부서 관리자의 성(**Lname**), 주소(**Address**), 생년월일(**Bdate**)을 검색하라

SQL 질의 예제

[질의 0] 이름이 **'john B. Smitth'**인 사원(들)의 생년월일(**Bdate**)과 주소(**Adress**)를 검색하라.

질의는 **FROM** 절에 표시된 **EMPLOYEE** 릴레이션만 검색한다

```
SELECT Bdate, Adress
FROM EMPLOYEE
WHERE Fname= 'John' AND Mint='B' AND Lname='Smith'
```

[질의 1] **'Research'** 부서에서 근무하는 모든 사원의 이름(**Fname, Lname**)과 주소(**Address**)를 검색하라.

```
SELECT Fname, Lname, Address
FROM EMPLOYEE, DEPARTMENT
WHERE Dname='Research' AND Dnumber = Dno;
```

조인 조건 (Dnumber = Dno)

SQL 질의 예제

[질의 2] 'Stafford'에 위치한 모든 프로젝트 들에 대해서 프로젝트 번호(Pnumber), 담당부서 번호(Dnum), 부서 관리자의 성(Lname), 주소(Address), 생년월일(Bdate)을 검색하라

SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber **AND** Mgr_ssn=Ssn **AND** Plocation='Stafford'

두 개의 조인 조건 (Dnum = Dnumber, Mgr_ssn=Ssn)

SQL 질의 예제: 결과

Bdate	Address
1965-01-09	731Fondren, Houston, TX

Fname	Lname	Address
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

(c)

Pnumber	Dnum	Lname	Address	Bdate
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

모호한 애트리뷰트

➢ 모호한 애트리뷰트 이름

- SQL 에서는 동일한 이름을 갖는 애트리뷰트가 여러 테이블에 사용될 수 있음
  - 이 경우에, 다중 테이블 질의를 통해 동일한 이름을 갖는 두 개 이상의 애트리뷰트를 참조하려면 모호함을 피하기 위해 릴레이션 이름과 함께 애트리뷰트 이름을 사용함 (**[릴레이션이름].[애트리뷰트이름]**)
  - 예: 만약 EMPLOYEE 의 Dno → Dnumber, Lname → Name 으로 가정, DEPARTMENT의 Dname

[질의 1A]

```
SELECT Fname, EMPLOYEE.Name, Address
FROM EMPLOYEE, DEPARTMENT
WHERE DEPARTMENT.Name='Research' AND DEPARTMENT.Dnumber = EMPLOYEE.Dnumber;
```

[질의 1']

```
SELECT EMPLOYEE.Fname, EMPLOYEE.Lname, EMPLOYEE.Address
FROM EMPLOYEE, DEPARTMENT
WHERE DEPARTMENT.Dname='Research' AND DEPARTMENT.Dnumber = EMPLOYEE.Dno;
```

SQL 재명명

➢ SQL 재명명 (AS)

- 릴레이션의 별명
  - 같은 릴레이션을 두 번 참조하는 질의에서도 애트리뷰트 이름의 모호함이 나타남
  - 한 릴레이션(EMPLOYEE)에 대해서 **별명** 또는 **투플 변수**라고 부르는 또 다른 릴레이션의 이름들(E 와 S)을 선언하여 사용 가능

[질의 8] 각 사원에 대해 사원의 이름(Fname)과 성(Lname), 직속 상사의 이름(Fname)과 성(Lname)을 검색하라

```
SELECT E.Fname, E.Lname, S.Fname, S.Lname
FROM EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.Super_ssn = S.Ssn;
```

- 위의 질의 예제에서는 조인 조건 (E.Super_ssn = S.ssn)을 만족하는 투플들을 찾아냄으로써 한 릴레이션을 **자기 자신과 조인**
- 애트리뷰트 재명명
  - SQL 별명을 사용하여 질의 내의 릴레이션의 애트리뷰트 이름들을 재명명 가능
  - EMPLOYEE AS E(Fn, Mi, Ssn, Bd, Addr, Sex, Sal, Sssn, Dno)
- SQL 질의에서 **WHERE** 내에 있는 릴레이션들에 대해 투플 변수를 명시하기 위해 별명 또는 재명명 방식 사용 가능 → **질의 이해에 도움**

WHERE 절의 생략과 '*'의 사용

➢ WHERE 절의 생략

- 투플 선택에 대한 조건이 없다는 것을 의미
- FROM 절에서 명시한 릴레이션의 모든 투플이 질의 결과로 검색됨
- 릴레이션들의 크로스 프로덕트(cross product)
  - FROM 절에 두 개 이상의 릴레이션이 명시되고, WHERE 절이 없다면 이 릴레이션들의 모든 가능한 투플의 조합이 선택됨

[질의 9와 10] EMPLOYEE의 모든 Ssn을 선택하고 (Q9), EMPLOYEE 의 Ssn 과 DEPARTMENT의 Dname의 모든 조합을 선택하라 (Q10)

(Q9)
```
SELECT Ssn
FROM EMPLOYEE
```

(Q10)
```
SELECT Ssn, Dname
FROM EMPLOYEE, DEPARTMENT
```

➢ '*'의 사용

- 선택된 투플들의 모든 애트리뷰트 값을 검색하려면 SELECT 절에 모든 애트리뷰트의 이름을 명시적으로 열거하는 대신에 별표(*)를 명시
- 별표(*) 앞에 릴레이션 이름 또는 별명이 올 수 있음
  - SELECT * / FROM EMPLOYEE / WHERE Dno=5;

SQL 집합으로의 테이블

➢ 중복된 투플 제거

- SQL은 질의 결과에서 자동적으로 중복된 튜플을 제거하지 않음
  - 중복을 없애는 연산은 비용이 큼 (정렬 + 제거)
  - 사용자가 질의 결과에 중복된 투플들이 나타나는 것을 원하는 경우가 있음
  - 집단 함수(평균, 합 등)가 투플들에 적용될 때 대부분의 경우에 중복 삭제가 필요하지 않음
- DISTINCT 사용
  - 키를 가진 SQL 테이블은 키 값이 각 투플마다 구별되야 하므로 집합으로 표현되도록 제한
  - SQL 질의 결과에서 중복된 투플을 삭제하려면 SELECT 절에 키워드 DISTINCT 사용
- SELECT ALL 사용
  - 그냥 SELECT 와 동일
  - 중복 투플을 제거하지 않음

[질의 11] 모든 사원의 급여(Salary)를 검색하고(Q11), 구별되는 급여를 모두 검색하라(Q11A).

(Q11)
```
SELECT All Salary
FROM EMPLOYEE
```

(Q11A)
```
SELECT DISTINCT Salary
FROM EMPLOYEE
```

SQL 에서의 관계 대수 연산

➢ 집합 관계 대수 연산

- 합집합(UNION), 차집합(EXCEPT), 교집합(INTERSECT) 연산들을 포함
- 집합 관계 대수 연산들도 중복 투플 제거 옵션을 추가할 수 있음
- 집합 연산들은 타입 호환성을 갖는 릴레이션들에만 적용 가능함
  - 동일한 애트리뷰트를 가지며 이 애트리뷰트들이 양쪽 릴레이션에서 같은 순서로 나타남

Query 4. Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

```
Q4A:  (SELECT    DISTINCT Pnumber
       FROM      PROJECT, DEPARTMENT, EMPLOYEE
       WHERE     Dnum=Dnumber AND Mgr_ssn=Ssn
                 AND Lname='Smith' )
       UNION
      ( SELECT   DISTINCT Pnumber
       FROM      PROJECT, WORKS_ON, EMPLOYEE
       WHERE     Pnumber=Pno AND Essn=Ssn
                 AND Lname='Smith' );
```

부분 문자열 패턴 비교

➢ LIKE 비교 연산자

- 문자열의 패턴 비교에 사용
- 부분 문자열(substring)의 표현
  - '%' : 0 보다 큰 임의의 개수의 문자
  - '_' : 임의의 한 개의 문자를 의미
- 탈출문자(escape character)
  - 문자열 내의 상수로 밑줄이나 %가 필요한 경우 그 문자열 뒤에 ESCAPE를 써서 표시한 **탈출문자**(예: \)를 앞에 붙임
  - **작은따옴표**('')는 문자열의 시작과 끝을 나타내므로 이를 문자열 내에 포함시키려면 **두 개의 연속된** 작은따옴표로 표현

[질의 12] 주소가 'Houston, Texas'인 모든 사원을 검색하라

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Address LIKE '%Houston,Tx%';
```

[질의 13] 1970년대에 태어난 모든 사원을 검색하라

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Bdate LIKE '_ _ 7 _ _ _ _ _';
```

산술 연산자 및 기타 연산자

- **질의 내에서 산술식 허용**
  - 표준 산술 연산자: 더하기(+), 빼기(-), 곱하기(*), 나누기(/)
    - 수치 값 혹은 숫치 애트리뷰트에 적용할 수 있음
- **기타 연산자**
  - 집합 연산자 (concatenate operator)
    - 문자열 데이터 타입에 대해서 집합연산자 ‘ ‘ 를 사용하여 문자열을 합칠 수 있음
  - 기간 증가 연산자
    - 날짜, 시간, 타임스탬프 또는 기간 데이터 타입에 대해 날짜, 시간 또는 타임스탬프를 주어진 기간만큼 증가(+)시키거나 감소(-)시킬 수 있음
  - 비교 연산자(BETWEEN)

질의 결과의 정렬

➢ ORDER BY 절 사용

- 하나 이상의 애트리뷰트를 기준으로 질의 결과에 들어 있는 투플들을 정렬하는 것이 가능
- 정렬 순서
  - 디폴트 정렬은 오름차순 (ASC)
  - 내림 차순으로 보고자 한다면 키워드 (DESC)를 명시
  - `ORDER BY D.Dname DESC E.lname ASC E.Fname ASC`

[질의 15] 사원 및 각 사원이 근무하는 프로젝트들의 리스트를 검색하는데, 부서 이름 순서대로, 그리고 각 부서 내에서는 사원의 성과 이름의 알파벳 순서대로 구하라.

```
SELECT D.Dname, E.Lname, E.Fname, P.Pname
FROM DEPARTMENT AS D, EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE D.Dname=E.Dno AND E.Ssn=W.Essn, AND W.Pno=P.Pnumber
ORDER_BY D.Dname, E.Lname, E.Fname;
```

기본 SQL 검색 질의에 대한 논의와 요약

➢ SQL 에서의 검색 질의

- SQL 에서 간단한 검색 질의 하나는 4개의 절까지 가질 수 있으며 첫 두 항목(**SELECT, FROM** 절)은 필수적으로 가져야 함

```
SELECT      <애트리뷰트 리스트>
FROM        <테이블 리스트>
[WHERE      <조건>]
[ORDER_BY   <애트리뷰트 리스트>]
```

  - **SELECT** 절: 검색하려고 하는 애트리뷰트들의 리스트 나열
  - **FROM** 절: 질의에 필요한 모든 릴레이션(테이블)을 명시
  - **WHERE** 절: 필요하다면 조인 조건을 포함하여 **FROM** 절에 명시된 릴레이션들로부터 투플들을 선택하기 위한 조건을 명시
  - ORDER_BY절: 질의 결과를 출력하는 순서를 명시
- 추가로 사용되는 두 개의 절: **GROUP_BY, HAVING**
  - 집단 함수에 추가적인 기능을 부여

SQL 에서의 삽입

➢ INSERT 명령

- 릴레이션에 투플을 추가하는데 사용됨
- 릴레이션 이름, 애트리뷰트 값들의 리스트를 명시해야 함
  - 애트리뷰트 값들의 순서는 CREATE TABLE 에서 명시한 애트리뷰트 들의 순서와 같아야 함
  - **INSERT INTO** EMPLOYEE
  - **VALUES** (‘Richard’, ‘K’, ‘Marini’, ‘65329853’, ‘1962-12-30’, ‘98 Oak Forest, Katy, TX’, ‘M’, 37000, ’653298653’, 4);
- **INSERT** 명령의 두 번째 형식에서는 **INSERT** 명령에서 명시한 값에 대응하는 애트리뷰트 이름을 명시적으로 나타냄
  - 릴레이션이 많은 애트리뷰트를 가지고 있지만 새로운 투플에는 단지 일부 애트리뷰트 값만 명시하는 경우에 유용
  - 하지만 이 경우에는 명시하지 않은 애트리뷰트들은 디폴트 값을 갖거나 널이 허용된 애트리뷰트이어야 함
  - **INSERT INTO** EMPLOYEE(Fname, Lname, Dno, Ssn)
  - **VALUES** (‘Richard’, ‘Marini’, 4, ‘65329853’);
- INSERT 명령을 처리할 때에는 무결성 제약조건을 만족시키는 지를 확인해야 함
  - DDL 으로 명시

SQL 에서의 삽입

➢ INSERT 와 질의 결과 삽입

- 한 질의의 결과로 검색되는 다수의 투플을 생성된 릴레이션에 삽입할 수 있음
- 질의로부터 생성된 릴레이션은 유도된 릴레이션이다.
  - 더 이상 필요하지 않을 경우 **DROP TABLE** 명령을 사용하여 제거 가능
  - 하지만, 원래의 저장된 릴레이션이 갱신될 경우 변경이 반영되지 않은 상태가 될 수 있음
  - 변경을 반영한 최신의 정보를 이용하기 위해서는 **뷰를 사용해야 함**

[삽입 요청] 한 프로젝트에 일하는 각 사원들에 대해 사원의 성, 프로젝트 이름, 주당 근무 시간을 갖는 임시 테이블을 생성.

```
CREATE TABLE    WORKS_ON_INFO(Emp_name  VARCHAR(15), Proj_name  VARCHAR(15),
                Hours_per_week    DECIMAL(3,1));
INSERT INTO     WORKS_ON_INFO (Emp_name, Proj_name, Hours_per_week)
SELECT          E.lname, P.Pname, W.Hours
FROM            PROJECT AS P, WORKS_ON AS W, EMPLOYEE AS E
WHERE           P.Pnumber=W.Pno AND W.Essn=E.Ssn;
```

SQL 데이터 삭제 및 갱신

➢ DELETE 명령

- 한 릴레이션에서 투플들을 삭제
- SQL 질의에서 사용한 것과 유사하게 삭제할 투플들의 조건을 나타내는 **WHERE** 절을 포함할 수 있음
  - 한 번에 **한 테이블** 내의 투플들만 삭제
    - 데이터 정의어에서 참조 무결성 제약조건 내에 참조 트리거된 동작이 명시되어 있다면 삭제는 다른 릴레이션 내의 투플들도 연쇄적으로 삭제할 수 있음
  - **WHERE** 절을 생략한 경우 테이블 내의 모든 투플을 삭제해 빈 테이블이 됨
  - 참고로, 테이블 정의 자체를 제거하려면 **DROPTABLE** 명령을 사용하야 함

➢ UPDATE 명령

- 선택된 하나 이상의 투플에서 애트리뷰트 값들을 수정하기 위해 사용
- **WHERE** 절은 **한 릴레이션**에서 수정할 투플들을 선택
  - 기본키 값이 수정된 경우에 참조 무결성 제약 조건 내에 참조 트리거된 동작이 명시되어 있다면 다른 릴레이션들에 있는 투플들의 외래키 값도 연쇄적으로 변경시킬 수 있음
- **SET** 절: 변경할 애트리뷰트의 새로운 값을 명시
- 단일 **UPDATE** 명령을 이용해 여러개의 투플을 수정 가능
  - `UPDATE EMPLOYEE / SET Salary=Salary*1.1 / WHERE Dno=5;`