CSE235
데이터베이스 시스템
(Database Systems)
Lecture 08: SQL I

담당교수: 전강욱(컴퓨터공학부)
kw.chon@koreatech.ac.kr

지난시간 복습

- **관계 대수**
  - 관계 모델을 위한 기본적인 연산들의 집합

- **SQL**
  - 실제적인 관계 데이터베이스의 표준 언어
  - 질의 결과를 기술함으로써 관계 조작을 정의
  - 관계 대수 개념에 기반

- **관계 데이터베이스시스템(RDBMS)의 질의 처리와 최적화 모듈에서 질의를 구현하고 최적화하기 위해 관계 대수 연산들을 사용**
  - SQL은 관계 연산들로 변환된 후, 질의 처리와 최적화가 수행됨

지난시간 복습 (계속)

- **피 연산자와 연산 결과가 모두 릴레이션**

- **관계 연산들은 관계 대수식으로 기본적인 검색 질의를 명시할 수 있음**
  - 일련의 관계 대수 연산들은 관계 대수식을 형성
  - 이 식의 결과는 데이터베이스 질의 결과를 나타내는 릴레이션

지난시간 복습 (계속)

- **셀렉션(selection)**
  - 특정한 조건을 만족하는 튜플들을 선택
- **프로젝션(projection)**
  - 특정 애트리뷰트의 선택
- **집합 연산(set)**
  - 튜플의 결합과 비교를 통해 유사한 2개의 튜플 집합을 조작
  - 합집합(union), 교집합(intersect), 차집합(difference)
- **조인(join), 프로덕트(product)**
  - 두 릴레이션의 튜플들을 하나로 묶음

복습 문제

- **셀렉션 연산에 대한 아래 설명 중에서 올바른 것은?**
  - ① 결과 릴레이션의 카디날리티와 차수는 입력 릴레이션의 카디날리티와 차수에 의존하지 않는다.
  - ② 결과 릴레이션의 카디날리티와 차수는 입력 릴레이션의 카디날리티와 차수와 동일하다.
  - ③ 결과 릴레이션의 차수는 입력 릴레이션의 차수와 같지만 결과 릴레이션의 카디날리티는 입력 릴레이션의 카디날리티를 초과할 수 없다.
- **동일한 스키마를 갖는 두 릴레이션 R(A , B) 와 S(A , B)가 있다. 릴레이션 R과 S의 유일한 키는 A이다. 릴레이션 T (A , B) 는 R과 S의 합집합, 즉 R ∪ S이다. T의 키는 무엇인가?**

복습 문제 (계속)

- **아래의 릴레이션 R에 대하여 물음에 답하라**

R

A	B	C
1	b	d
3	a	f
2	b	e

- 셀렉션의 결과를 보여라: $\sigma_{B=b \text{ AND } A>1}(R)$

- 프로젝션의 결과를 보여라: $\Pi_{A,C}(R)$, $\Pi_{B}(R)$

복습 문제 (계속)

- **아래의 두 릴레이션 R과 S를 보고 물음에 답하라**

R

A	B	C
a1	b1	c7
a2	b1	c5
a3	b4	c3

S

B	D
b1	d1
b5	d5

- R과 S의 카티션 곱의 결과를 보여라

- R과 S의 동등 조인의 결과를 보여라(조인 컬럼은 B).

복습 문제 (계속)

- **아래의 두 릴레이션 R과 S를 보고 물음에 답하라**

R

A	B	C
a1	b1	c7
a2	b1	c5
a3	b4	c3

S

B	D
b1	d1
b5	d5

  - R과 S의 자연 조인의 결과를 보여라

  - R과 S의 왼쪽 외부 조인의 결과를 보여라 (조인 컬럼은 B)

SQL (Structured Query Language)

- **SQL 개요**
  - SQL은 현재 DBMS 시장에서 관계 DBMS가 압도적인 우위를 차지하는데 중요한 요인의 하나
  - SQL은 IBM 연구소에서 1974년에 System R이라는 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 강의에서 SQL2 위주로 설명

SQL (계속)

- **발전 역사**
  - SEQUEL (Structured English Query Language) 유래
    - SEQUEL: 연구용 관계 데이터베이스 관리시스템인 SYSTEM R을 위한 언어
  - 미국 표준 연구소인 ANSI와 국제 표준화 기구인 ISO에서 표준화 작업을 진행
    - SQL-86: 1986년 미국 ANSI에서 표준으로 채택; 1987년에 ISO에서 표준으로 채택
    - SQL-89: 무결성 제약조건 기능이 강화
    - SQL2(SQL-92): 새로운 데이터 정의어와 데이터 조작어 기능이 추가
    - SQL3(SQL-99): 객체 지향과 순환, 멀티미디어 기능 등이 추가; 현재의 표준

SQL (계속)

- **SQL은 비절차적 언어(선언적 언어)**
  - 자신이 원하는 데이터만을 명시
  - 처리하는 방법은 명시 불가
  - 관계 DBMS는 사용자가 입력한 SQL문을 번역하여 사용자가 요구한 데이터를 찾는데 필요한 모든 과정을 담당
  - 두가지 사용방식
    - 대화식 SQL: 직접 데이터베이스 관리 시스템에 접근하여 질의를 실행
    - 삽입식 SQL: 응용 프로그램에 삽입

예제 데이터베이스 스키마

EMPLOYEE

FNAME	MINT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPARTMENT_NAME	SEX	BDATE	RELATIONSHIP

EMPLOYEE	FNAME	MINT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	09-Jan-55	Houston	M	30000	333445555	5
	Franklin	T	Wong	333445555	08-Dec-45	Houston	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	19-Jul-58	Spring	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	20-Jun-31	Bellaire	F	43000	888665555	4
	Ramesh	K	Narayn	666884444	15-Sep-52	Oak	M	38000	333445555	5
	Joyce	A	English	453453453	31-Jul-62	Houston	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	29-Mar-59	Houston	M	25000	987654321	4
	James	E	Borg	888665555	10-Nov-27	Houston	M	55000	NULL	1

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40
	453453453	1	20
	453453453	2	20
	333445555	2	10
	333445555	3	10
	333445555	10	10
	333445555	20	10
	999887777	30	30
	999887777	10	10
	987987987	10	35
	987987987	30	5
	987654321	30	20
	987654321	20	15
	888665555	20	null

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	22-May-78
	Administration	4	987654321	01-Jan-85
	Headquarters	1	888665555	19-Jun-71

DEPT_LOCATION	DNUMBER	DLOCATION
	1	서울
	4	천안
	5	서울
	5	부산
	5	대전

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	DEPARTMENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	05-Apr-76	DAUGHTER
	333445555	Theodore	M	25-Oct-73	SON
	333445555	Joy	F	03-May-48	SPOUSE
	987654321	Abner	M	29-Feb-32	SPOUSE
	123456789	Michael	M	01-Jan-78	SON
	123456789	Allice	F	32-DEC-78	DAUGHTER
	123456789	Elizabeth	F	05-May-57	SPOUS

스키마와 카탈로그

- **SQL 스키마**
  - 스키마 이름으로 식별
  - 스키마 각 원소에 대한 기술 뿐 아니라 스키마의 권한부여 식별자도 포함
    - 스키마 원소: 테이블, 뷰, 인덱스, 도메인 등
    - 권한부여 식별자: 사용자나 계정관련 권한부여
  - CREATE SCHEMA 명령어를 이용하여 생성
    - 데이터 뿐 아니라 메타 데이터를 위한 공간도 마련
    - e.g., JSMITH란 사용자 소유의 스키마 COMPANY를 생성하라

      CREATE SCHEMA COMPANY **AUTHRIZATION** JSMITH;
- **카탈로그**
  - 특별한 스키마인 "INFORMATION_SCHEMA"에 포함
  - 카탈로그 내 모든 스키마들과 이들 스키마 내 모든 원소 기술문들에 대한 정보 제공

CREATE TABLE 명령

- **새로운 릴레이션(table)을 생성하는 명령어**
- **릴레이션의 이름과 함께 각 속성 및 데이터 유형을 명시**
  - 속성명, 값집합, 제약조건 등
  - 키 속성, 엔티티무결성, 참조 무결성 등 제약 조건 명시
- **데이터 유형: INTEGER, FLOAT, DECIMAL (i,j), CHAR(n), VARCHAR(n) 등**
- **e.g., CREATE TABLE** COMPANY.DEPARTMENT **(**

  DNAME **VARCHAR(10) NOT NULL,**

  DNUMBER **INTEGER NOT NULL,**

  MGRSSN **CHAR(9),**

  MGRSTARTDATE **CHAR(9));**

CREATE TABLE 명령(계속)

- **PRIMARY KEY(기본 키)절은 릴레이션의 기본 키를 구성하는 하나 이상의 속성들을 명시**
- **UNIQUE 절은 대체키를 명시**
- **FOREIGN KEY(외래 키)절은 참조 무결성을 지정**
  - 참조 무결성을 위반시 취할 동작도 함께 정의함
    - 참조 무결성 위반: ON DELETE나 ON UPDATE 명령
    - 동작: SET NULL, CASCADE, SET DEFAULT 등의 동작을 지정 가능

CREATE TABLE 예제

```
CREATE TABLE EMPLOYEE
        (  FNAME                VARCHAR(15)          NOT NULL,
           MINIT                CHAR,
           LNAME                VARCHAR(15)          NOT NULL,
           SSN                  CHAR(9)              NOT NULL,
           BDATE                DATE,
           ADDRESS              VARCHAR(30),
           SEX                  CHAR,
           SALARY               DECIMAL(10, 2),
           SUPERSSN             CHAR(9),
           DNO                  INT                  NOT NULL,
        PRIMARY KEY (SSN),
        FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN),
        FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNUMBER)) ;
```

EMPLOYEE

FNAME	MINT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

CREATE TABLE 예제(계속)

```
CREATE TABLE DEPARTMENT
        ( DNAME                 VARCHAR(15)          NOT NULL,
          DNUMBER               INT                  NOT NULL,
          MGRSSN                CHAR(9)              NOT NULL,
          MGRSTARTDATE          DATE,
        PRIMARY KEY (DNUMBER),
        UNIQUE (DNAME),
        FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE (SSN)) ;
```

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

```
CREATE TABLE DEPT_LOCATONS
        (  DNUMBER              INT                  NOT NULL,
           DLOCATION            VARCHAR(15)          NOT NULL,
         PRIMARY KEY (DNUMBER, DLOCATON),
         FOREIGN KEY (DNUMBER) REFERENCES DEPARTMENT (DNUMBER));
```

DEPT_LOCATIONS

DNUMBER	DLOCATION

CREATE TABLE 예제 (계속)

```
CREATE TABLE PROJECT
        (  PNAME                    VARCHAR(15)          NOT NULL,
           PNUMBER                  INT                  NOT NULL,
           PLOCATION                VARCHAR(15),
           DNUM                     INT                  NOT NULL,
         PRIMARY KEY (PNUMBER),
         UNIQUE (PNAME),
         FOREIGN KEY (DNUM) REFERENCES DEPARTMENT (DNUMBER)) ;
```

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

```
CREATE TABLE WORKS_ON
        (  ESSN                     CHAR(9)              NOT NULL,
           PNO                      INT                  NOT NULL,
           HOURS                    DECIMAL(3, 1)        NOT NULL,
         PRIMARY KEY (ESSN, PNO),
         FOREIGN KEY (ESSN) REFERENCES EMPLOYEE (SSN),
         FOREIGN KEY (PNO) REFERENCES PROJECT (PNUMBER)) ;
```

WORKS_ON

ESSN	PNO	HOURS

CREATE TABLE 예제 (계속)

```
CREATE TABLE DEPENDENT
        (  ESSN                         CHAR(9)                 NOT NULL,
           DEPENDENT_NAME               VARCHAR(15)             NOT NULL,
           SEX                          CHAR,
           BDATE                        DATE,
           REATIONSHIP                  VARCHAR(8),
         PRIMARY KEY (ESSN, DEPENDENT_NAME),
         FOREIGN KEY (ESSN)  REFERENCES EMPLOYEE (SSN)) ;
```

DEPENDENT

ESSN	DEPARTMENT_NAME	SEX	BDATE	RELATIONSHIP

속성 데이터 타입과 도메인

- **애트리뷰트의 데이터 타입**
  - 숫자: INTEGER, INT, SMALLINT, FLOAT, REAl, DOUBLE PRECISION
  - 문자열: CHAR(n), CHARACTER(n), VARCHAR(n), CHAR VARYING(n), CHARACTER VARYING(n), ||: 문자열 연결연산
  - 비트열: BIT(n), BOT VARYING(n)
  - 불리언
  - 날짜와 시간: DATE(YYYY-MM-DD), TIME(HH:MM:SS)
  - 타임스탬프: DATE+TIME+초의 소수점 이하 자리 (예: 2002-09-27 09:12:47 648302)

- **CREATE DOMAIN**
  - 새로운 릴레이션을 생성하는 데 사용
  - 도메인의 이름과 데이터 타입 명시
  - 예: CRAETE DOMAIN SSN_TYPE AS CHAR(9)

속성 제약조건 및 디폴트 값 명시

- **SQL은 속성 값으로 NULL을 허용함**
  - NOT NULL 제약조건을 지정하여 속성에 NULL값을 허용하지 않을 수도 있음
  - e.g., CREATE TABLE DEPARTMENT
    (DNUMBER INT NOT NULL, …) ;

- **디폴트 값 명시**
  - DEFAULT문을 이용하여 애트리뷰트의 디폴트 값을 명시
  - e.g., CREATE TABLE EMPLOYEE
    (DNO INT DEFAULT 1, …) ;

- **도메인 값 제약**
  - CHECK문을 이용하여 애트리뷰트의 값집합을 제약
  - e.g., CREATE TABLE DEPARTMENT
    (DNUMBER INT CHECK(DNUMBER > 0 AND DNUMBER < 21), …) ;

속성 제약조건 및 디폴트 값 명시 (계속)

```
CREATE TABLE EMPLOYEE
        (       FNAME           VARCHAR(15) NOT NULL,
                ...
                DNO             INT         NOT NULL   DEFAULT 1,
        CONSTRAINT EMPPK
                PRIMARY KEY (SSN),
        CONSTRAINT EMPSUPERFK
                FOREIGN KEY (SUPERSSN) 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) ;
```

널 값 허용 안함 (기본값 1)

해당 튜플을 삭제할 경우, 그 튜플을 참조하고 있는 모든 튜플을 널 값으로 설정

속성 제약조건 및 디폴트 값 명시 (계속)

```
CREATE TABLE DEPARTMENT
        (DNAME                  VARCHAR(15)                     NOT NULL,
         DNUMBER                INT                             NOT NULL,
         MGRSSN                 CHAR(9)  NOT NULL  DEFAULT "888665555"
         MGRSTARTDATE           DATE,
        CONSTRAINT DEPTPK       PRIMARY KEY (DNUMBER),
        CONSTRAINT DEPTSK       UNIQUE (DNAME),
        CONSTRAINT DEPTMGRFK    FOREIGN KEY (MGRSSN) REFERENCE EMPLOYEE (SSN)
                                ON DELETE SET DEFAULT ON UPDATE CASCADE) ;
```

해당 튜플을 삭제할 경우, 그 튜플을 참조하고 있는 모든 튜플을 기본 값으로 변경

키와 참조 무결성 제약조건의 명시

- **PRIMARY KEY 절**
  - 릴레이션의 기본 키를 구성하는 하나 이상의 속성들을 명시
    e.g., CREATE TABLE DEPARTMENT
    (DNUMBER INTEGER PRIMARY KEY, …) ;
- **UNIQUE 절**
  - 대체키(또는 보조키)를 명시

- **FOREIGN KEY 절**
  - 참조 무결성 지정
  - 외래 키를 정의할 때 참조 무결성의 위반 시 취할 동작을 명시 가능
  - 위반의 종류: ON DELETE, ON UPDATE 등
  - 동작의 종류: SET NULL, SET DEFAULT, CASCADE 등
  - e.g., ON DELETE CASCADE: 삭제되는 튜플을 참조하고 있는 모든 튜플을 삭제
    ON UPDATE CASCADE: 수정되는 튜플을 참조하는 모든 튜플의 외래키 값을 수정된 튜플의 새롭게 갱신된 기본키 값으로 수정

DROP 명령어

- **DROP SCHEMA**
  - 스키마(a.k.a., 데이터베이스)를 제거하는 명령
    - e.g., **DROP TABLE** COMPANY **[CASCADE | RESTRICT]**;
  - CASCADE: 제거되는 스키마(데이터베이스)에 포함된 테이블, 도메인 등을 모두 제거
  - RESTRICT: 비어 있는 스키마를 제거할 때 사용
    - 비어 있지 않으면 제거 불가
- **DROP TABLE**
  - 릴레이션(a.k.a., 테이블)을 제거하는 명령
    - e.g., DROP TABLE DEPENDENT **[CASCADE | RESTRICT]**;
  - CASCADE: 릴레이션과 릴레이션을 참조하는 모든 제약조건 및 뷰 등을 자동적으로 제거
  - RESTRICT: 제약조건이 없는 릴레이션만을 제거
    - 다른 릴레이션의 외래키에서 참조
    - 뷰에서 참조

ALTER 명령어

- **기본 테이블의 정의를 변경하는 명령어**
  - 속성의 추가/제거, 열 정의 변경, 테이블 제약 조건 추가/제거
  - CASCADE: 기존 속성 제거시, 참조하는 모든 제약조건과 뷰를 제거
  - RESTRICT: 제거되는 속성을 참조하는 뷰와 제약조건이 없는 경우에만 속성을 제거
  - e.g., EMPLOYEE 테이블에 TITLE 속성 추가
    - **ALTER TABLE** COMPANY.EMPLOYEE **ADD** TITLE VARCHAR(10);
  - e.g., EMPLOYEE 테이블에 FNAME 속성 제거
    - **ALTER TABLE** COMPANY.EMPLOYEE **DROP** FNAME **CASCADE**;
  - e.g., EMPLOYEE 테이블에 DNO 속성 변경
    - **ALTER TABLE** COMPANY.EMPLOYEE **ALTER** DNO **DROP** DEFAULT;
    - **ALTER TABLE** COMPANY.EMPLOYEE **ALTER** DNO **SET** DEFAULT "9999"

SELECT 문

- **SELECT 문은 데이터베이스에서 정보를 검색하는 기본 명령어**
  - SQL의 SELECT문의 결과 릴레이션(테이블)은 속성 값의 중복이 허용
    - i.e., 튜플들의 집합이 아닌 튜플들의 다중집합을 결과로 출력
  - 사용자들은 키 제약조건이나 DISTINCT 조건을 활용하여 SQL의 결과 릴레이션들을 집합으로 표현 가능함

SQL 질의의 구조

- **SELECT 문의 구조**

  SELECT <속성 목록>

  FROM <테이블 목록>

  WHERE <조건>

  - 속성 목록: 질의 결과에 나타나는 속성명 목록
  - 테이블 목록: 질의의 대상이 되는 테이블 목록
  - 조건: 질의 결과의 튜플들이 만족해야 하는 조건식

DEPT_LOCATION	DNUMBER	DLOCATION
	1	서울
	4	천안
	5	서울
	5	부산
	5	대전

```
SELECT DNUMBER, DLOCATION
FROM DEPT_LOCATION
WHERE DNUMBER=5;
```

SQL 질의 예제

- **이름이 'John B. Smith'인 사원의 생일(BDATE)과 주소(ADDRESS)를 검색하시오**

```
SELECT    BDATE, ADDRESS
FROM      EMPLOYEE
WHERE     FNAME='John' AND MINIT='B' AND LNAME='Smith' ;
```

- **관계대수 표현**

$$\Pi_{\text{BDATE, ADDRESS}}(\sigma_{\text{NAME='John' AND MINIT='B' AND LNAME='Smith'}}(\text{EMPLOYEE}))$$

SQL 질의 예제 (계속)

- **'Research' 부서에서 일하는 모든 사원의 이름(FNAME, LNAME)과 주소를 검색하시오.**

```
SELECT      FNAME, LNAME, ADDRESS
FROM        EMPLOYEE, DEPARTMENT
WHERE       DNAME='Research' AND DNUMBER=DNO ;
```

  - 관계대수 연산 SELECT-PROJECT-JOIN과 유사
  - SELECT 절은 관계 대수의 PROJECT 연산에 해당
  - WHERE 절에서 DNAME='Research'은 선택조건, 관계대수에서 SELECT 연산에 해당
  - WHERE 절에서 DNUMBER=DNO는 조인조건, 관계대수의 JOIN 연산에 해당

SQL 질의 예제 (계속)

- ‘Stafford’에 위치한 모든 프로젝트에 대하여 프로젝트 번호, 담당 부서 번호, 부서 관리자의 성, 주소, 생일을 검색하라.

```
SELECT    PNUMBER, DNUM, LNAME, ADDRESS, BDATE
FROM      PROJECT, DEPARTMENT, EMPLOYEE
WHERE     DNUM=DNUMBER AND MGRSSN=SSN AND
          PLOCATION=‘Stafford’ ;
```

- 두 개의 조인조건이 존재
  - 조인조건 DNUM=DNUMBER는 프로젝트와 담당 부서를 조인
  - 조인조건 MGRSSN=SSN은 부서와 담당 관리자를 조인

모호한 애트리뷰트 이름과 별명의 사용

- **동일한 이름을 갖는 애트리뷰트의 사용**
  - 서로 다른 릴레이션에서 동일한 이름을 갖는 애트리뷰트가 사용될 수 있음
  - 릴레이션 이름과 함께 애트리뷰트 이름을 사용함으로써 모호함을 방지해야 함
  - 질의 작성시 릴레이션 이름 다음에 점(.)을 두고 애트리뷰트 이름을 명시함

- **질의 1A**
  - 'Research' 부서에서 일하는 모든 종업원들의 이름과 주소를 검색하시오.

```
SELECT   FNAME, EMPLOYEE.NAME, ADDRESS
FROM     EMPLOYEE, DEPARTMENT
WHERE    DEPARTMENT.NAME='Research' AND
         DEPARTMENT.DNUMBER=EMPLOYEE.DNUMBER ;
```

모호한 애트리뷰트 이름과 별명의 사용 (계속)

- **동일한 릴레이션을 두 번 참조하는 경우**
  - 질의 작성에서 모호성은 동일한 릴레이션을 두 번 참조하는 경우에도 발생함
  - 이 경우에도 모호함을 방지하기 위하여 릴레이션 이름의 별명을 애트리뷰트 이름 앞에 붙여서 사용함

- **질의 8**
  - 종업원에 대해, 종업원의 성과 이름, 직속 감독자의 성과 이름을 검색하시오.

```
SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM     EMPLOYEE E, EMPLOYEE S      // EMPLOYEE에 대한 별명
WHERE    E.SUPERSSN=S.SSN ;
```

    - 위의 예는 EMPLOYEE 릴레이션에 대해서 두 개의 별명(alias) E와 S를 선언하여 사용함

모호한 애트리뷰트 이름과 별명의 사용 (계속)

- **릴레이션에 대한 별명**
  - 키워드 AS의 이용
    - From 절의 릴레이션 이름 바로 다음에 오거나, EMPLOYEE AS E 처럼 키워드 AS를 이용해서 릴레이션과 연관시킴
  - 애트리뷰트 이름 재명명
    - 질의 내에서 별명을 주어 릴레이션의 애트리뷰트를 재명명할 수도 있음
    - EMPLOYEE AS E(FN, MI, LN, SSN, BD, ADDR, SEX, SAL, SSSN, DNO) ;
      - 별명 FN은 FNAME, MI은 MINIT, LN은 LNAME 대신 이용

- **질의 1B – 별명을 이용하여 질의 1A를 간단하게 나타낼 수 있음**

```
SELECT    E.FNAME, E.NAME, E.ADDRESS
FROM      EMPLOYEE E, DEPARTMENT D
WHERE     D.NAME='Research' AND D.DNUMBER=E.DNUMBER ;
```

WHERE절의 생략

- **WHERE 절의 생략**
  - SQL에서 WHERE 절을 생략하면 튜플 선택에 대한 조건이 없다는 것을 의미함
  - FROM 절에 있는 테이블의 모든 튜플이 조건을 만족하게 됨

- **질의 9**
  - 데이터베이스에서 EMPLOYEE의 모든 SSN을 선택하시오.

```
SELECT   SSN
FROM     EMPLOYEE ;
```

- **질의 10**
  - EMPLOYEE의 SSN과 DEPARTMENT의 DNAME의 모든 조합을 선택하시오.

```
SELECT   SSN, DNAME
FROM     EMPLOYEE, DEPARTMENT ;
```

'*'의 사용

- **선택된 튜플들의 모든 애트리뷰트 값들을 검색하는 경우**
  - 모든 애트리뷰트 이름을 명시적으로 열거하지 않고 단지 '*'를 사용함

- **질의 1C**
  - 5번 DEPARTMENT에서 일하는 EMPLOYEE 튜플들의 모든 애트리뷰트 값들을 검색하라.

```
SELECT   *
FROM     EMPLOYEE
WHERE    DNO=5 ;
```

'*'의 사용

- **질의 1D**
  - 'Research' 부서에서 일하는 모든 종업원들에 대하여 EMPLOYEE의 모든 애트리뷰트들과 DEPARTMENT의 모든 애트리뷰트들을 검색하라.

```
SELECT   *
FROM     EMPLOYEE, DEPARTMENT
WHERE    DNAME='Research' AND DNO=DNUMBER ;
```

- **질의 10A**
  - 릴레이션 EMPLOYEE와 DEPARTMENT의 카티션 프로덕트 결과를 모두 검색하라.

```
SELECT   *
FROM     EMPLOYEE, DEPARTMENT ;
```

집합으로서의 테이블

- **튜플의 중복허용**
  - 릴레이션이나 질의의 결과로 중복된 튜플들이 나타날 수 있으므로, SQL에서는 일반적으로 릴레이션을 집합으로 취급하지 않음
  - 중복된 튜플의 삭제
    - SELECT 항목에서 키워드 DISTINCT를 사용
    - DISTINCT는 질의 결과에서 유일한 튜플들만 남기라는 의미의 키워드임

- **질의 11**
  - 모든 사원의 급여를 검색하라.

```
SELECT   SALARY
FROM     EMPLOYEE ;
```

- **질의 11A**
  - 모든 사원의 구별되는 급여를 검색하라.

```
SELECT   DISTINCT SALARY
FROM     EMPLOYEE ;
```

집합으로서의 테이블

- **SQL에서의 집합 연산**
  - 합집합(UNION)연산, 차집합(EXCEPT)연산, 교집합(INTERSECT)
  - 릴레이션에 대한 집합 연산의 결과는 튜플들의 집합임 → 중복된 튜플을 결과에서 제거함

- **질의 4**
  - 성이 'Smith'인 종업원(일반 직원 혹은 프로젝트를 담당하는 부서의 관리자)이 참여하는 프로젝트의 프로젝트 번호 목록을 작성하시오.

```
( SELECT    PNUMBER              // Smith가 관리자인 projects
  FROM      PROJECT, DEPARTMENT, EMPLOYEE
  WHERE     DNUM=DNUMBER AND MGRSSN=SSN AND
    LNAME='Smith')
UNION
( SELECT    PNUMBER              // Smith가 참여하는 projects
  FROM      PROJECT, WORKS_ON, EMPLOYEE
  WHERE     PNUMBER=PNO AND ESSN=SSN AND LNAME='Smith') ;
```

부분 문자열 패턴 비교와 산술 연산자

- **문자열에 대한 비교**
  - SQL은 LIKE 비교 연산자를 사용하여 문자열(혹은 부분 문자열)에 대해 비교조건을 적용할 수 있음
  - 부분 문자열을 표현할 때 '%'는 임의의 개수의 문자를 의미하고, '_'는 임의의 한 문자를 의미함

부분 문자열 패턴 비교와 산술 연산자 (계속)

- **질의 12**
  - 주소가 Houston, Texas인 모든 종업원을 검색하시오.

```
SELECT   FNAME, LNAME
FROM     EMPLOYEE
WHERE    ADDRESS LIKE '%Houston, TX%' ;
[WHERE   ADDRESS LIKE '*Houston, TX*' ;]
```

- **질의 12 A**
  - 1950년대에 태어난 모든 사원을 검색하라.

```
SELECT   FNAME, LNAME
FROM     EMPLOYEE
WHERE    BDATE LIKE '__5_____' ;
[WHERE   BDATE LIKE '??5????????' ;]
```

부분 문자열 패턴 비교와 산술 연산자 (계속)

- **질의내 산술식 허용**

- **질의 13**
  - ‘ProductX’ 프로젝트에 참여하는 모든 사원의 급여를 10% 올린 경우의 급여를 검색하라.

```
SELECT    FNAME, LNAME, 1.1*SALARY
[SELECT   FNAME, LNAME, 1.1*SALARY AS NEWSALARY]
FROM      EMPLOYEE, WORKS_ON, PROJECT
WHERE     SSN=ESSN AND PNO=PNUMBER AND PNAME=‘ProductX’ ;
```

- **질의 14**
  - 급여가 30,000$에서 40,000$ 사이에 있는 5번 부서의 모든 사원을 검색하라.

```
SELECT    *
FROM      EMPLOYEE
WHERE     (SALARY BETWEEN 30000 AND 40000) AND DNO=5
```

질의 결과의 정렬

- **질의 결과의 정렬**
  - ORDER BY 절
    - 하나 이상의 애트리뷰트 값 순서로 질의 결과 튜플을 정렬
  - Default 정렬은 오름차순임
    - 키워드 DESC: 내림차순으로 정렬 - Ascending
    - 키워드 ASC: 오름차순 정렬 - Descending
    - 예: ORDER BY DNAME DESC, LANME ASC, FNAME ASC

- **질의 15**
  - 프로젝트에 참여하는 종업원을 부서의 알파벳 순서대로, 각 부서 내에서는 성과 이름의 알파벳 순서대로 출력하시오.

```
SELECT      DNAME, LNAME, FNAME, PNAME
FROM        DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE       DNUMBER=DNO AND SSN=ESSN AND PNO=PNUMBER
ORDER BY    DNAME, LNAME, FNAME ;
```

삽입, 삭제, 갱신 구문

- INSERT 명령
- DELETE 명령
- UPDATE 명령

INSERT 명령

- **릴레이션에 단일 레코드 삽입**
  - 튜플의 값들은 CREATE TABLE 명령에서 지정한 속성들의 순서와 동일하게 지정
    - e.g., INSERT INTO EMPLOYEE
      VALUES (‘Richard’,‘K’,‘Marini’,‘653298653’,‘30-DEC-52’,
      ‘98 Oak Forest, Katy, TX’,‘M’,37000,‘987654321’,4) ;
  - 속성 순서와 다르게 값을 입력하기 위해서 속성명을 함께 명시하는 것이 필요함
    - 속성명을 명시지 않은 경우, 널 값이나 디폴드 값 설정
    - e.g., INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)
      VALUES (‘Richard’,‘K’,‘Marini’,‘653298653’) ;

INSERT 명령 (계속)

- **SELECT와 결합된 INSERT 명령**
  - 질의의 결과로 생성된 다중 튜플을 또 다른 릴레이션에 삽입하는 경우에 SELECT와 INSERT가 결합된 문장을 사용함
  - e.g.,

```
CREATE TABLE    DEPTS_INFO
       (DEPT_NAME       VARCHAR(15),
        NO_OF_EMPS      INTEGER,
        TOTAL_SAL       INTEGER) ;

INSERT INTO  DEPTS_INFO (DEPT_NAME, NO_OF_EMPS, TOTAL_SAL)
        SELECT      DNAME, COUNT (*), SUM (SALARY)
        FROM        DEPARTMENT, EMPLOYEE
        WHERE       DNUMBER=DNO
        GROUP BY    DNAME ;
```

- **뷰(view)의 필요성**
  - 위의 질의 수행 후, EMPLOYEE 테이블 갱신시, DEPTS_INFO 테이블은 갱신된 값을 반영 못하는 문제가 있음
  - DEPTS_INFO 테이블을 최신 정보로 유지하기 위해서는 뷰로 정의하는 것이 필요

DELETE 명령

- **DELETE 명령은 릴레이션에서 튜플(들)을 제거하는 명령임**
  - 삭제할 튜플에 대한 조건은 WHERE 절에서 명시함
  - 한번의 DELETE 명령으로 WHERE 절의 조건을 만족하는 튜플을 모두 삭제함
    - e.g., DELETE FROM EMPLOYEE WHERE LNAME='Brown' ;
    - e.g., DELETE FROM EMPLOYEE WHERE SSN='123456789' ;
    - e.g., DELETE FROM EMPLOYEE WHERE DNO IN (
      SELECT DNUMBER
      FROM DEPARTMENT
      WHERE DNAME='Research') ;
  - WHERE 절을 생략한 경우에는 테이블내의 모든 튜플을 삭제하며, 테이블은 데이터베이스 내에서 빈 테이블로 남게 됨
    - e.g., DELETE FROM EMPLOYEE;

UPDATE 명령

- **UPDATE 명령은 튜플의 속성 값을 수정하기 위해 사용함**
  - WHERE 절은 한 릴레이션에서 수정할 튜플을 선택하는데 사용됨
  - SET절은 변경할 속성과 그들의 새로운 값을 명시함

  - PROJECT 테이블에서 PNUMBER가 10인 튜플에 대하여 PLOCATION을 'Bellaire'로 변경하고, 담당 부서인 DNUM을 5로 변경하라.
    - e.g.,

```
UPDATE   PROJECT
SET      PLOCATION='Bellaire', DNUM=5
WHERE    PNUMBER=10 ;
```

  - 'Research' 부서에 있는 모든 종업원들의 봉급을 10% 인상하라.
  - e.g.,

```
UPDATE   EMPLOYEE
SET      SALARY=SALARY*1.1
WHERE    DNO IN  (   SELECT  DNUMBER
                     FROM    DEPARTMENT
                     WHERE   DNAME='Research') ;
```

SQL의 기타 기능

- **권한 기능**
  - SQL은 데이터베이스 사용자에게 권한을 부여하고 취소하는 기능을 제공함
- **호스트 언어와 결합되어 사용**
  - SQL은 C, C++, COBOL, JAVA, PASCAL 등과 같은 범용 프로그래밍 언어 내에서 사용될 수 있음
  - Embedded SQL/C, C++, COBOL, JAVA, PASCAL
- **트랜잭션 기능**
  - SQL은 트랜잭션 제어 명령문을 가짐 – 동시성 제어와 회복
- **기타 유용한 명령어**
  - 상용 DBMS는 SQL 명령 이외에도 물리적 데이터베이스 설계 매개변수와 릴레이션들을 위한 파일 구조, 그리고 인덱스와 같은 접근경로를 명시하기 위한 명령어의 집합을 가지고 있음

View

- **View의 특성**
  - SQL에서 뷰는 다른 테이블들에서 유도된 "가상" 테이블 - 실제로 저장되지는 않음
  - 기본 테이블들의 열로 구성
  - 뷰에 대한 질의는 아무런 제한을 받지 않음
  - 몇 개 연산들을 뷰로 표현하여 사용하는데 편리함
  - 데이터 접근제어로 보안성 제공
  - 뷰에 적용할 수 있는 갱신(삽입, 삭제) 연산들은 제한됨 - 물리적인 형태로 저장되지는 않기 때문에, 뷰는 일반적인 Alter 문으로 변경할 수 없음
- **명령어**
  - 뷰를 정의하는 SQL 명령: CREATE VIEW WORKS_ON_NEW;
  - 뷰를 삭제하는 SQL 명령: DROP VIEW WORKS_ON_NEW;

Embedded SQL

- **대부분 SQL 문장들은 COBOL, C, Java와 같은 범용 호스트 프로그래밍 언어에 내포될 수 있음**

- **내포된 SQL 문장은 EXEC SQL 와 END-EXEC (또는 세미콜론(;))에 의해 호스트 프로그래밍 언어 문장으로 구분함**
  - 공유 변수들은 SQL 문장 내에서 사용될 때는 콜론(:)을 그 앞에 붙임

요약

- SQL의 데이터 정의와 데이터 타입
- SQL에서 기본 제약조건의 명시
- SQL에서의 기본 검색 질의
- SQL에서 삽입, 삭제, 갱신문
- SQL의 기타기능

감사합니다!

kw.chon@koreatech.ac.kr