

SQL (Structured Query Language)

SQL 기능

- 데이터 정의어 (DDL)
- 데이터 조작어 (DML) CRUD (insert, select, update, delete)
- 뷰 정의어
- 보안과 권한 관리
- 무결성 제약 조건 명시
- 트랜잭션 제어

Embedded SQL (DSL : Data Server Language)은 Java, COBOL, C/C++(Host language)에 Embedding 가능

CREATE – 테이블 생성

DROP – 테이블 삭제

INSERT – 행 삽입

DELETE – 행 삭제

DDL, DML

CREATE SCHEMA

- 동일한 데이터베이스에 속하는 테이블이나 기타 구성요소들을 그룹화하기 위해 사용
- 스키마의 생성은 스키마의 이름과 함께 기술
- catalog > schema > table
- CREATE SCHEMA Company AUTHORIZATION 'Jsmith';

CREATE TABLE

- 새로운 기본 테이블을 생성하는 데 사용하며, 릴레이션의 이름과 함께 각 애트리뷰트와

데이터 유형을 기술함

- 데이터 유형 : (INTEGER, FLOAT, DECIMAL(i, j), CHAR(n), VARCHAR(n))
- NOT NULL 제약조건을 각 애트리뷰트에 명시할 수 있음

- Example

```
CREATE TABLE DEPARTMENT (  
  DNAME          VARCHAR(10) NOT NULL,  
  DNUMBER        INTEGER NOT NULL,  
  MGRSSN         CHAR(9),  
  MGRSTARTDATE   CHAR(9)  
);
```

CREATE TABLE – Key and Foreign Key

- CREATE TABLE 명령은 Primary Key와 Secondary Keys, 그리고 참조 무결성 제약(Foreign Keys)을 명시할 수 있음
- Key 애트리뷰트들은 Primary Key와 UNIQUE 절을 통해 명시할 수 있음

- Example

```
CREATE TABLE DEPT (  
  DNAME          VARCHAR(10) NOT NULL,  
  DNUMBER        INTEGER NOT NULL,  
  MGRSSN         CHAR(9),  
  MGRSTARTDATE   DATE,  
  PRIMARY KEY (DNUMBER),  
  UNIQUE (DNAME),  
  FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(Ssn)  
);
```

트랜잭션 지원하는 엔진 - InnoDB

%sql : 한 줄 입력

%%sql : 여러 줄 입력

%sql show databases;

- Information_schema : 사용할 수 있는 저장된 meta data를 표시

%sql show tables;

- database에서 만들어진 것들을 출력

%sql desc EMPLOYEE;

- EMPLOYEE의 schema를 출력
- KEY는 PRI (Primary key), MUL (foreign key)

순환 참조 (Cyclic References)의 문제점

- 생성되지 않은 테이블을 참조하면 외래키에 오류 발생
- 행 삽입에도 동일한 문제 발생
- 테이블 행들을 모두 삽입 후 **alter table** 문을 사용하여 외래키 선언을 추가하여 스키마를 변경
- 예) alter table EMPLOYEE **ADD** (FOREIGN KEY (dno) REFERENCES DEPARTMENT (dnumber) ON DELETE SET NULL);
- Alter : add, modify, drop 사용 가능

애트리뷰트 데이터 타입

숫자

- 정수형 : INTEGER, INT, SMALLINT
- 부동소수점수 : FLOAT, REAL, DOUBLE PRECISION
- 형식화된 수 : **DECIMAL(i,j), NUMERIC(i,j)** (i : 숫자의 총 개수. 유효숫자, j : 소수점 뒤에 오는 숫자의 개수). DECIMAL(5,2) : -999.99 ~ 999.99

문자열

- **고정** 길이 - **CHAR**(n)
- **가변** 길이 - **VARCHAR**(n). 애트리뷰트의 크기를 예상할 수 없을 때, 애트리뷰트의 update가 일어나지 않을 때 사용
- CLOB : Character Large Object

비트열

- 고정길이 – BIT(n)
- 가변길이 – BIT VARYING(n)
- BLOB – Binary Large Object

BOOLEAN (TRUE/FALSE)

DATE, TIME, TIMESTAMP 데이터 타입을 추가로 가짐

- DATE :
 - yyyy-mm-dd 형식으로 year-month-day을 표현
 - DATE '2008-09-27'
- TIME
 - hh:mm:ss 형식으로 hour:minute:second을 표현함
 - TIME '09:12:27'
- TIMESTAMP
 - DATE와 TIME 구성요소를 포함
 - TIMESTAMP '2002-09-27 09:12:47 648302'

SQL에서 기본제약조건의 명시

- 애틀리뷰트 제약조건과 디폴트값 명시
 - 애틀리뷰트에 NULL 값의 허용여부 명시 : NOT NULL
 - 애틀리뷰트에 디폴트 값 지정 : DEFAULT <값>
 - 애틀리뷰트나 도메인 값을 특정한 값(들)으로 한정 : CHECK (값의 범위)
- 키와 참조 무결성 제약조건의 명시
 - CREATE TABLE 명령에서 PRIMARY KEY 절은 릴레이션의 기본 키를 구성하는 하나 이상의 애틀리뷰트들을 명시하고, UNIQUE 절은 대체키(또는 보조키)를 명시하며, FOREIGN KEY 절에서는 참조 무결성을 지정함
 - 외래 키(FOREIGN KEY)를 정의할 때, 참조 무결성의 위반시 취할 동작을 명시할 수 있음.

동작의 종류에는 SET NULL, CASCADE, SET DEFAULT가 있으며, 위반의 종류를 나타내는 ON DELETE나 ON UPDATE와 함께 사용

- SQL 시스템들은 참조 무결성과 기본 키를 지정
- 초기 SQL에서는 기본 키를 지정하는 기능이 없고 CREATE UNIQUE INDEX 명령을 통해 지정하고 있음

SQL에서 기본 검색 질의

- SQL은 데이터베이스로부터 정보를 검색하는 문장을 가짐
- 관계 대수의 SELECT 연산과는 무관
- SQL과 관계모델의 중요한 차이점
- SQL의 테이블(릴레이션)은 모든 애트리뷰트 값이 동일한 튜플을 하나 이상 가질 수 있음
- 따라서, SQL 릴레이션(테이블)은 튜플의 집합이 아니라 튜플의 다중집합(multi-set or bag)임
- 키 제약조건을 선언하거나 DISTINCT 선택사항을 사용하여 SQL 릴레이션들을 집합으로 제한
- SQL SELECT문의 기본 형식은 사상(mapping) 또는 SELECT-FROM-WHERE 블록이라고 불림

SELECT <attribute list>

FROM <table list>

WHERE <condition>

- <attribute list> : 질의 결과에 나타나는 애트리뷰트 이름 목록
- <table list> : 질의 대상이 되는 릴레이션 목록
- <condition> : 질의 결과에 포함될 튜플들을 표시하는 조건(부울) 식

간단한 SQL 질의들

- 기본 SQL 질의들은 관계 대수의 SELECT, PROJECT, JOIN 연산으로 표현 가능

- 관계대수 연산의 SELECT-PROJECT 쌍과 유사
 - SELECT 절은 프로젝트 애트리뷰트를 표시하고, WHERE 절은 선택 조건을 표시
 - 질의의 결과는 중복된 튜플 포함

'Research'부서에서 일하는 모든 종업원들의 이름과 주소를 검색하시오

SELECT FNAME, LNAME, ADDRESS

FROM EMPLOYEE, DEPARTMENT EMPLOYEE X DEPARTMENT (카티션 곱)

WHERE DNAME='Research' and DNUMBER=DNO

- DNAME='Research'는 선택 조건이고 관계 대수에서 SELECT 연산
- DNUMBER=DNO는 조인조건이고 관계대수의 JOIN 연산에 해당

SQL 작성시 릴레이션 이름 다음에 (.) 을 두고 애트리뷰트 이름을 명시함

Example) EMPLOYEE.LNAME, DEPARTMENT.DNAME

별명(ALIAS)

- 어떤 질의들은 동일한 릴레이션을 두 번 참조할 필요가 있음
- 릴레이션 이름에 별명을 부여해야 함

Example) 각 사원에 대해 성과 이름, 직속 상사의 성과 이름을 검색

SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME

FROM EMPLOYEE E S

WHERE E.SUPERSSN = S.SSN;

- E는 감독을 받는 사원을 S는 감독을 하는 사원(직속상사)을 나타냄

WHERE 절의 생략

- 튜플 선택에 대한 조건이 없다는 것을 의미함.

- FROM 절에 있는 릴레이션의 모든 튜플이 조건을 만족함.

Example) EMPLOYEE의 모든 SSN을 선택.

SELECT SSN

FROM EMPLOYEE

*의 사용

- 선택된 튜플들의 모든 애트리뷰트 값들을 검색하려면 모든 애트리뷰트 이름을 명시적으로 열거하지 않고 *을 사용함.

DISTINCT 사용

- SQL은 일반적으로 집합으로 취급하지 않음. -> 중복된 튜플들이 나타날 수 있음
- 질의 결과에서 중복된 튜플들을 삭제하려면 키워드 DISTINCT (or UNIQUE)를 사용
- SELECT가 SK이면 DISTINCT가 필요 없음

Example)

- select distinct ssn, salary from EMPLOYEE;
- EMPLOYEE의 ssn이 key이므로 결과값이 중복되지 않음. DISTINCT 필요 없음
- select ssn, dname
- from EMPLOYEE e, DEPARTMENT d, PROJECT p
- where e.ssn = d.mgrssn and d.dnumber = p.dnum;
- d.mgrssn은 e.ssn의 외래키
- p.dnum은 d.dnumber의 외래키
- PROJECT의 key인 pno가 select에 없음. 결과값이 중복. DISTINCT 필요
- select ssn, pno
- from EMPLOYEE e, WORKS_ON, PROJECT p
- EMPLOYEE의 key ssn, PROJECT의 key pno가 있지만 WORKS_ON의 key가 없으므로 결과값이 중복, DISTINCT 필요

집합 연산

- 합집합(UNION)연산, 차집합(EXCEPT) 연산, 교집합(INTERSECT) 연산 제공
- 릴레이션에 대한 집합 연산의 결과는 튜플들의 집합
- 중복된 튜플을 결과에서 제거
- UNION ALL, EXCEPT ALL, INTERSECT ALL 연산은 다중집합 연산임.
- Example) $A=\{1,1,2,2,2,3\}$ $B=\{1,2,2,3,3,4\}$
- $A \text{ UNION ALL } B = \{1,1,1,2,2,2,2,2,3,3,3,4\}$
- $A - B \text{ except all} = \{1,2\}$
- $A \text{ intersect all } B = \{1,2,2,3\}$

부분 문자열 비교

% : wild string

_ : wild character

%sql select ' ' like '%'; True 1

%sql select 'a' like '_'; True 1

%sql select 'ab' like '_'; False 0

- 앞에 %를 붙이면 느리다. 예) %abc
- 정렬하면 abc로 시작하는 것은 빨리 찾을 수 있음
- abc를 포함하는 것은 정렬해서 해결되는 일이 아님