**내용 개요**

**1. 데이터베이스의 개념**

**2. 데이터베이스에서 핵심이라고 할 수 있는 관계 데이터베이스 모델 즉 관계 데이터베이스 설명**

**3. 트랜잭션**

**4. 뷰**

**데이터베이스(Database, DB)**

- 개념: **여러 사람이 공유하여 사용할 목적으로 체계화해 통합, 관리하는 데이터의 집합**

    통합, 저장, 공용 데이터

- 특징

① 실시간 접근성 : 질의 또는 조회에 대해 즉각적 응답

② 계속적 변화 : 새로운 데이터의 삽입, 삭제, 갱신으로 항상 최신의 데이터를 유지하면서 변화

③ 동시 공유 : 여러 사용자가 동시에 같은 데이터에 접근 가능

④ 내용 참조 : 데이터를 참조할 때 주소나 위치가 아닌 내용으로 접근

- 장점

① 데이터 중복 최소화

② 데이터 공유

③ 최신의 데이터 유지

④ 용이한 데이터 접근

⑤ 데이터의 표준화 가능

⑥ 데이터의 논리적, 물리적 독립성

⑦ 데이터 저장 공간 절약

⑧ 일관성, 무결성(데이터의 정확성과 일관성 유지 보증하는 것을 뜻함), 보안성 유지

- 단점

① 데이터베이스 전문가 필요

② 많은 비용 부담

③ 데이터 백업과 복구가 어려움

④ 시스템의 복잡함

⑤ 대용량 디스트로 엑세스가 집중되면 과부하 발생

- 구성요소

① 속성(attribute) : 데이터베이스의 가장 작은 논리적 단위

 개체의 구성 원소로서 그 개체의 특성이나 상태를 나타냄

② 개체(entity) : 데이터베이스가 표현하려고 하는 유형, 무형의 정보 객체

    서로 연관된 몇 개의 속성들로 구성

③ 관계(relationship) : 개체 간의 어떤 의미를 표현함

관계 유형

⒜ 일대일(1:1)

⒝ 일대다(1:N)

⒞ 다대다(N:N)

**데이터베이스 관리 시스템(DataBase Management System, DBMS)**

- 개념: 모든 응용프로그램들이 데이터베이스를 공용으로 사용할 수 있게끔 관리해 주는 시스템 소프트웨어

- 기능

① 정의기능 : 데이터베이스에 저장될 데이터형과 구조의 정의, 이용방법, 제약조건을 명세

 응용프로그램과 데이터베이스 간의 상호작용 수단

② 조작기능 : 데이터베이스에 접근하여 데이터 검색, 삽입, 삭제, 갱신 등을 수행

 사용자와 데이터베이스 간의 상호작용 수단

③ 제어기능 : 보안 유지, 권한 검사, 여러 사용자가 같은 데이터를 동시에 처리할 때 병행 제어

**데이터베이스 언어(Database Language)**

1. **데이터 정의어** (Data Definition Language, DDL)

- 데이터베이스 구조, 데이터 형식, 접근방식 등 데이터베이스를 생성하거나 수정할 목적으로 이용하는 언어

- SQL ex)  CREATE(생성), ALTER(변경), DROP(삭제)

2. **데이터 조작어**(Data Manipulation Langauge, DML)

- 사용자와 DBMS 간의 인터페이스를 제공하는 언어, 사용자가 원하는 데이터를 처리하는데 이용하는 언어

- SQL ex) Select(튜플 검색),Insert(튜플 삽입),Update(튜플 갱신),Delete(튜플 삭제)

             DROP != Delete

3. **데이터 제어어**(Data Control Language, DCL)

- 불법 사용자로부터 데이터를 보호하기 위해 이용하는 언어

- SQL ex) COMMIT(수행된 결과를 실제 물리적 디스트로 저장), ROLLBACK(비정상적으로 종료되었을때 복구),

  GRANT(권한 부여), REVOKE(권한 취소)

 \* SQL (Structured Query Language)

**데이터 베이스 설계 순서**

**사용자의 요구 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 데이터베이스 구현**

개념적 설계 : 개체와 개체들 간의 관계에서 ER다이어그램\*(아래에서 설명)을 만드는 과정

논리적 설계 : ER다이어그램을 사용하여 논리적 모델 만드는 과정

물리적 설계 : 논리적 모델을 실제 컴퓨터의 저장 장치에 저장하기 위해 물리적 구조를 정의하고 구현하는 과정

**데이터 모델**

- 개념: 현실세계의 데이터 구조를 컴퓨터 세계의 데이터 구조로 논리적으로 표현한 것

- 종류

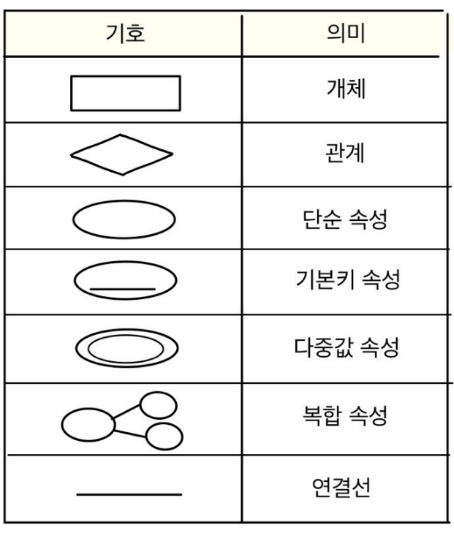
① **개념적 모델** : 인간이 이해하기 쉽게 개념적으로 표현하는 방법

**E-R 모델** : 개체 타입과 관계 타입을 이용해 현실 세계를 개념적으로 표현한 모델

 개체, 속성, 관계로 묘사

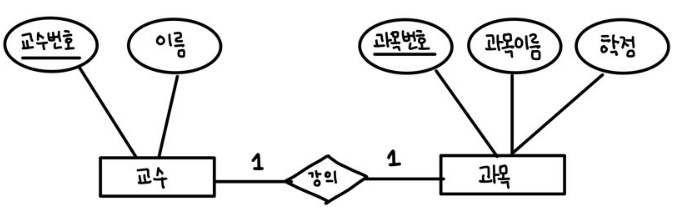
 E-R 다이어그램으로 표현

E-R 다이어그램 : E-R모델을 시각적으로 표현하기 위해 사용되는 도형, 여러 기호로 구성

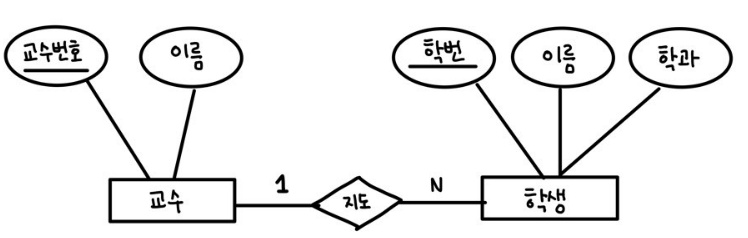


 예를 들어서 E-R모델을 만들어보자

교수와 과목은 1:1 일대일 관계이다.



교수와 학생은 1:N 일대다 관계이다.



② **논리적 모델** : 개념적 구조를 컴퓨터가 이해하고 처리할 수 있게 표현하는 방법

**관계 데이터 모델** : 행과 열의 테이블 형태

  상용 DBMS : ORACLE, SQL/DS

**계층 데이터 모델** : 트리 형태

  상용 DBMS : IMS, SYSTEM2000

**네트워크 데이터 모델** : 그래프 형태

  상용 DBMS : DBTG, IDMS, TOTAL

**관계데이터베이스**

**릴레이션(Relation)** 으로 표현됨

 - **릴레이션** 이란? 데이터 구조를 테이블 형태로 표현한 것

릴레이션 스키마와 릴레이션 인스턴스로 구성

① 릴레이션 스키마 : 릴레이션의 논리적 구조를 정의하는 정적 구조

② 릴레이션 인스턴스 : 어느 한 시점의 릴레이션 상태, 즉 튜플 전체를 의미하며 동적 구조

 - 릴레이션 관련 용어

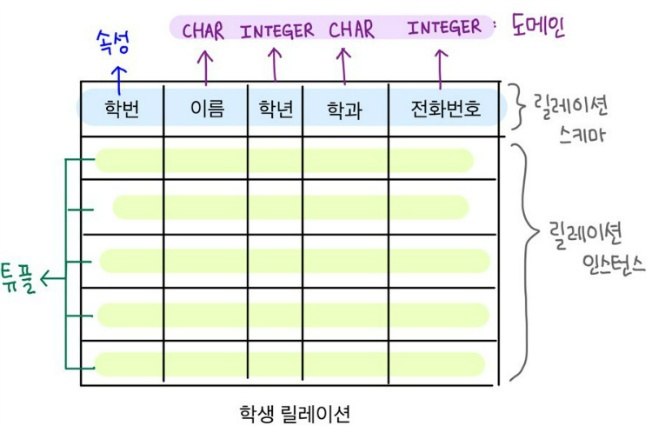
① 튜플(Tuple) : 릴레이션에서 각각의 행(row)을 의미

② 속성(Attiribute) : 릴레이션에서 각각의 열(column)을 의미

③ 도메인(Domain) : 한 속성이 가질 수 있는 값의 집합

④ 차수(Degree) : 릴레이션에서 속성의 개수

⑤ 카디널리티(Cardinality) : 릴레이션에서 튜플의 개수



이 학생 릴레이션의 degree는 학번, 이름, 학년, 학과, 전화번호 = 5

cardinality는 튜플의 수 = 5 이다.

 - 릴레이션의 특징

① 릴레이션의 모든 튜플은 중복되지 않는다. → 튜플의 유일성

② 릴레이션에서 튜플 사이의 순서는 정의되지 않는다. → 튜플의 무순서

③ 릴레이션을 구성하는 속성 사이에는 순서가 없다. → 속성의 무순서

④ 릴레이션의 속성값은 더이상 분해할 수 없는 원자값이다. → 속성값의 원자성

⑤ 릴레이션을 구성하는 속성명은 유일한 이름이여야 한다. → 속성명의 유일성

 - 릴레이션의 키

**키(Key)** : 릴레이션의 모든 튜플들을 유일하게 식별할 수 있는 특성을 가진 속성이나 속성의 집합을 의미

⒜ **슈퍼키(super key)** : 릴레이션 내의 모든 튜플에 대하여 **유일성**만 만족하는 속성

  릴레이션 내에서 여러 개의 슈퍼키가 정의 가능

⒝ **후보키(candidate key)** : 릴레이션에서 튜플을 유일하게 구별하기 위해 사용하는 속성 혹은 속성들의 집합

 릴레이션 내에서 **유일성 뿐만 아니라 최소성**도 만족해야 하는 속성

 릴레이션 내에서 여러 개의 후보키 정의 가능

* + - * **유일성(uniqueness) : 릴레이션에 있는 모든 튜플에 대해 유일하게 식별되어야 한다.**
      * **최소성(minimality) : 유일성을 가진 키를 구성하는 속성(Attribute) 중 하나라도 제외하는 경우 유일성이 깨지는 것을 의미한다. 즉, 릴레이션의 모든 튜플을 유일하게 식별하는 데 꼭 필요한 속성들로만 구성되어야 한다.**

**인터넷에서 가져온 문제로 후보키를 구해 보겠다.**

(ex)  아래와 같은 학생들의 인적사항이 주어졌을 때, 후보 키의 최대 개수를 구하라.



위의 예를 설명하면, 학생의 인적사항 릴레이션에서 모든 학생은 각자 유일한 “학번”을 가지고 있다.

따라서 “학번”은 릴레이션의 후보 키가 될 수 있다.

그다음 “이름”에 대해서는 같은 이름(“apeach”)을 사용하는 학생이 있기 때문에, “이름”은 후보 키가 될 수 없다.

그러나, 만약 [“이름”, “전공”]을 함께 사용한다면 릴레이션의 모든 튜플을 유일하게 식별 가능하므로 후보 키가 될 수 있게 된다.

물론 [“이름”, “전공”, “학년”]을 함께 사용해도 릴레이션의 모든 튜플을 유일하게 식별할 수 있지만,

최소성을 만족하지 못하기 때문에 후보 키가 될 수 없다.

따라서, 위의 학생 인적사항의 후보키는 “학번”, [“이름”, “전공”] 두 개가 된다.

+ 슈퍼키는 "학번", [“학번”, “이름”] ,["학번“,이름”, “전공”],[“이름”, “전공”], [“이름”, “전공”, “학년”] ...등 튜플의 유일성만 만족하면 되므로 다양하다

⒞ **기본키(primary key)** : 여러 개의 후보키 중 선택된 하나의 주키

   기본키 속성값으로 NULL이나 중복되는 값은 결코 가질 수 없음

   \*NULL : 공백(space)도 아니고 0(zero)도 아니고 부재정보를 나타냄

   정의되지 않았거나 알려지지 않는 모르는 값

⒟ **대체키(alternate key)** : 여러 후보키 중 기본키로 선택되지 못한 후보키

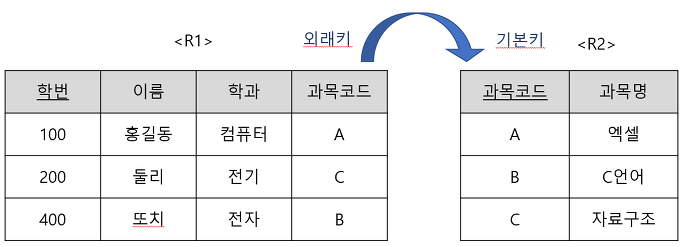
⒠ **외래키(foreign key)** : 릴레이션 R1, R2에서 R1의 속성값이 릴레이션 R2의 기본키 값들과 일치함을 요구하는  R1의 속성값을 외래키 라고한다.

   이 경우 외래키는 릴레이션 R2를 참조한다고 하고, R1을 참조 릴레이션, R2를 참조된 릴레이션이라고 한다.

  참조관계인 외래키와 기본키의 속성명은 다를 수 있지만 정의된 도메인은 같아야한다.

  외래키는 참조하는 릴레이션의 기본키에 없는 값은 가질 수 없다.

  외래키는 릴레이션과 릴레이션을 연결(join)하는 도구로 사용된다.



 - **무결성 제약조건**

① 개체 무결성 제약조건 : 기본키(primary key)의 값은 결코 널(null)값이나 중복값을 가질 수 없다.

② 참조 무결성 제약조건 : 외래키(foreign key) 값은 참조하는 릴레이션의 기본키 값과 일치하거나 NULL 값만 가져야 하고, 참조할 수 없는 값은 가질 수 없다.

**관계 데이터베이스의 연산자**

**1. 관계 대수** : 원하는 정보를 얻기 위해 일련의 연산순서를 명세하는 절차적 언어

**일반 집합 연산자**

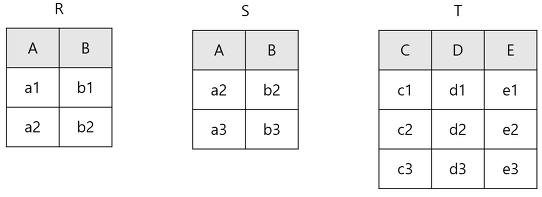
① 합집합(∪)

② 교집합(∩)

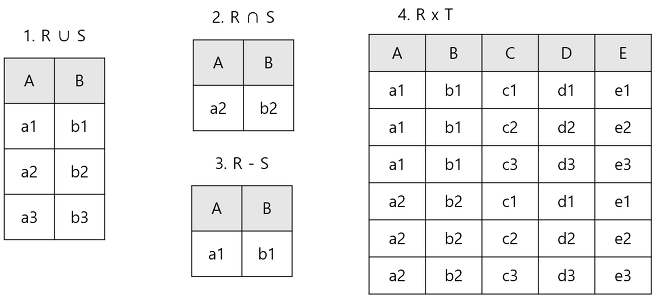
③ 차집합(－)

④ 교차곱(×)

ex) 예를 들어 릴레이션 R,S,T가 있다고 할 때



합집합, 교집합, 차집합, 교차곱을 계산하면 아래와 같다

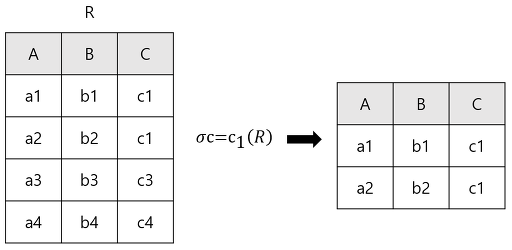


**순수 관계 연산자**

① Select(σ) : 릴레이션에서 조건식을 만족하는 튜플을 검색하는 연산자

행에 해당하는 튜플만 구함

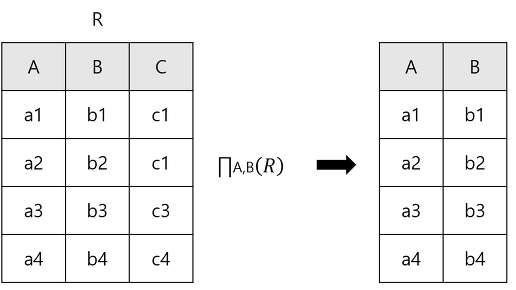
(ex) 릴레이션 R에서 속성 C값이 c1인 튜플만 추출할 시



② Project(π) : 릴레이션에서 기술된 속성리스트를 추출하는 연산자

  열에 해당하는 속성만 구함

(ex) 릴레이션 R에서 속성 A,B 값만 추출할 시



(ex) 학생 릴레이션에서 컴퓨터 학과 학생의 이름을 추출할 시 관계 대수?

-> **∏**이름(**σ**학과=컴퓨터(학생))

③ Join(**▷◁**) : 두 릴레이션의 공통 속성을 기준으로 하나로 합쳐서 새로운 릴레이션을 만드는 연산자

카디션 곱(교차곱, x) + 선택연산(select)의 형태

조건에 따라 3 가지로구별됨

세타조인(theta join) : 선택연산의 비교연산자가 {＝, ＜, ＞, ≤, ≥ }등이 사용되는 경우

동등조인(equi join) : 세타 조인 중 특별히 비교 연산자가 = 인 경우

자연조인(natural join) : 동등 조인에서 중복 속성 중 하나가 제거 된 경우

(ex) 학생 릴레이션과 과목 릴레이션이 있을 때



'학생’릴레이션과 ‘과목’ 릴레이션을 **동등 조인** 할 경우

(학생 **⋈학생.과목코드 = 과목.과목코드**과목)

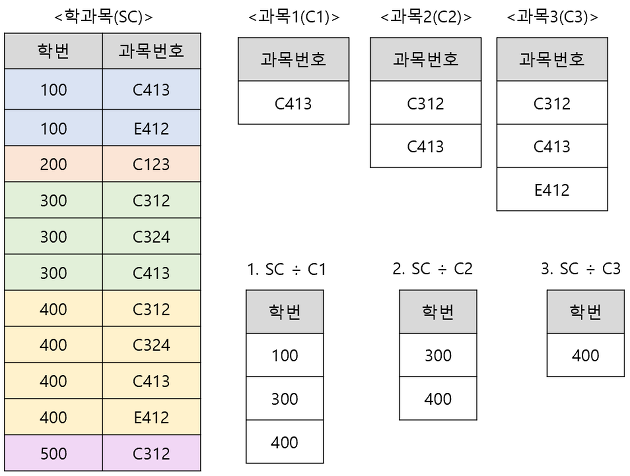


학생’릴레이션과 ‘과목’ 릴레이션을 **자연 조인** 할 경우(중복 속성 중 하나 제거)



④ Division(÷) : 두 릴레이션 R(X), S(Y) 에 대하여 X ⊇ Y 일때, R의 속성이 S의 속성값을 모두 가지는 튜플에서 S가 가진 속성을 제외한 속성값만을 구하는 연산자

(ex) 학과목 릴레이션과 과목릴레이션 3개가 있을때 Division예시를 보면



﻿

1. SC ÷ C1

나누려는 릴레이션의 원소를 가지고 있어야하므로 과목1(C1)의 원소 C413을 가진 학번은

100, 300, 400 번.

2. SC ÷ C2

과목2(C2)의 원소 C312을 가진 학번은 300, 400, 500

C413을 가진 학번은 100, 300, 400

두 원소 공통으로 가지고 있는 학번은 300, 400 번.

3. SC ÷ C3

과목3(C3)의 원소 C312, C413, E412 모두 가진 학번은 400 번.

**2. 관계 해석** : 원하는 정보가 무엇이라는 것만 선언하는 비절차적 언어

|  |  |
| --- | --- |
| 정량자 | ∀ : 모든 것에 대하여 ∃ : 존재하는 것에 대하여 |
| 논리연산자 | ∧ : and ∨ : or ￢ : not |
| 비교 연산자 | ＜,＞,≤,≥,＝,≠ |
| 기타 연산자 | ∈ : 원소, ⊂ : ~부분집합,  : 공집합 |

**관계데이터베이스 언어**

- SQL(Structed Query Language) : 관계 대수및 관계 해석에 기초한 데이텁이스 언어

|  |  |  |
| --- | --- | --- |
| 구분 | 명령어 | 기능 |
| DDL(데이터 정의어) | CREATE  ALTER  DROP | 테이블정의  테이블변경  테이블삭제 |
| DML(데이터 조작어) | INSERT  DELETE  UPDATE  SELECT | 데이터삽입  데이터삭제  데이터변경  데이터검색 |
| DCL(데이터 제어어) | GRANT  REVOKE  COMMIT  ROLLBACK | 권한부여  권한해제  트랜잭션 완료  트랜잭션 취소 |

**1. 데이터 정의어**

① **CREATE** : 테이블 정의

   형식

**CREATE TABLE <테이블명>(**

**속성명 데이터타입 [NOT NULL] [DEFAULT <값>] [PRIMARY KEY]**

**[UNIQUE (속성명)]**

**[FOREIGN KEY (외래키) REFERENCES <참조테이블 이름> (기본키)]**

**[CHECK (조건식)]**

**);**

UNIQUE : 속성값의 중복을 배제하기 위해 사용

FOREIGN KEY : 외래키 지정시 사용

CHECK : 테이블의 행 삽입이나 수정시 유지해야할 무결성 제약조건

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | /\*예를 들어 다음 조건에 맞게 '학생'테이블을 만든다고 할 때  조건)  학번은 정수형으로 NULL을 가질 수 없고, 기본키이다.  이름은 가변길이 문자열 10자리  학년은 정수형으로 1~4 사이만 입력되는 제약조건을 가진다.  학과는 고정길이 문자열 15자리  과목코드는 고정길이 문자열 10자리로 과목테이블의 과목코드를 참조하는 외래키이다.\*/    CREATE TABLE 학생(        학번 INT NOT NULL,      이름 VARCHAR(10),      학년 INT,      학과 CHAR(15),      과목코드 CHAR(10),      PRIMARY KEY(학번),      FOREIGN KEY(과목코드) REFERENCES 과목(과목코드),      CHECK(학년 >= 1 AND 학년 <= 4)    );  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**SQL의 데이터 타입**

**- 고정길이 문자 : CHAR(n), n은 문자 수**

**- 가변길이 문자 : VARCHAR(n)**

**- 정수 : INT, SMALL INT**

**- 실수 :  FLOAT, REAL**

**- 날짜 : DATE**

② **ALTER** : 테이블 변경

  형식

**ALTER TABLE <테이블명>**

**ADD 추가할 속성명 데이터타입;**

**DROP COLUMN 삭제할 속성명;**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5 | -- 예를 들어 '학생' 테이블에 새로운 속성, 두문자 크기를 갖는 학점을 추가할 시  ALTER TABLE 학생 ADD 학점 CHAR(2);    --학생 테이블에 추가된 학점 속성을 삭제할 시  ALTER TABLE 학생 DROP COLUMN 학점; | [cs](http://colorscripter.com/info#e) |

③ **DROP** : 테이블 삭제

형식

**DROP TABLE <테이블명> [CASCADE|RESTRICT];**

CASCADE : 테이블과 테이블을 참조하는 모든 제약조건과 뷰들이 자동 삭제

RESTRICT : 제약조건이나 뷰들이 참조하지 않는 테이블인 경우만 삭제

|  |  |  |
| --- | --- | --- |
| 1  2 | --예를 들어 학생 테이블을 삭제하면서 학생테이블을 참조하는 모든 테이블도 같이 삭제할 시   DROP TABLE 학생 CASCADE; | [cs](http://colorscripter.com/info#e) |

**2. 데이터 조작어**

① **INSERT** : 튜플 삽입

형식

**INSERT INTO 테이블명(속성목록) VALUES(상수목록)**

**INSERT INTO 테이블명(속성목록) SELECT 문 ~**

속성목록 작성안하면 테이블의 모든 속성에 값을 삽입한다는 뜻

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | --예를 들어 학생 테이블의 학번, 이름 속성에 (9004, 박길동)을 삽입할 시  INSERT INTO 학생(학번, 이름) VALUES(9004, 박길동);    --학생 테이블의 각 속성에 대응하여 튜플 (9005, 차길동, 3, 보안)을 삽입할 시  INSERT INTO 학생 VALUES (9005, 차길동, 3, 보안);    --학생 테이블에서 전자과 학생의 학번,이름 학년을 검색하여 클래스 테이블로 삽입할 시    INSERT INTO 클래스(학번, 이름, 학년)  SELECT 학번,이름,학년 FROM 학생 WHERE 학과 ='전자'; | [cs](http://colorscripter.com/info#e) |

② **DELETE** : 튜플 삭제

형식

**DELETE**

**FROM <테이블명>**

**[WHERE <조건>]**

조건을 기술하지 않으면 테이블의 모든 튜플들이 삭제된다는 뜻

|  |  |  |
| --- | --- | --- |
| 1  2 | --예를 들어 학생테이블에서 학번이 9004인 학생을 삭제할시  DELETE FROM 학생 WHERE 학번 = 9004; | [cs](http://colorscripter.com/info#e) |

③ **UPDATE** : 튜플 갱신

형식

**UPDATE <테이블명>**

**SET 속성 = 변경값**

**[WHERE] <조건>**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6 | --예를 들어 모든 학생의 점수를 1점씩 더할시  UPDATE 학생 SET 점수 = 점수 + 1;      --학번이 9003인 학생의 이름을 박지성으로 바꿀시  UPDATE 학생 SET 이름 = '박지성' WHERE 학번 = 9003; | [cs](http://colorscripter.com/info#e) |

④ **SELECT** : 튜플 검색

형식

**SELECT [DISTINCT] <속성목록>**

**FROM <테이블 리스트>**

**[WHERE <[LIKE] 조건>]**

**[OREDER BY <속성> [ASC] 또는 [DESC] ]**

**[GROUP BY <그룹화 필드>]**

**[HAVING <그룹조건>]**

DISTINCT : 중복된 데이터를 제거하고 표시할 때 사용

LIKE : 문자 상수를 명세한 검색 조건

   %는 임의개수 문자

   \_는 한 개의 문자

ORDER BY :  검색결과 에 대해 속성별로 오름차순, 내림차순 지정

GROUP BY : 그룹화 할 기준 속성

HAVING : 그룹들 중에서 선택할 조건 기술

예를 들어 학생 테이블과 성적 테이블이 있다고 할때

학생

|  |  |  |  |
| --- | --- | --- | --- |
| 학번 | 이름 | 학년 | 학과 |
| 9001 | 박찬호 | 3 | 전자 |
| ... | ... | ... | .. |

.

.

성적

|  |  |  |  |
| --- | --- | --- | --- |
| 학번 | 과목코드 | 중간고사 | 기말고사 |
|  |  |  |  |

.

.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69 | --학생 테이블에서 학과가 '게임'인 학생의 학번, 이름, 학과 검색할시    SELECT 학번, 이름, 학과  FROM 학생  WHERE 학과 = '게임';        --학생테이블에서 중복된 학과는 한번만 출력할 시    SELECT DISTINCT 학과  FROM 학생;        --성적테이블에서 과목코드가 b로 시작하는 튜플을 검색할 시    SELECT \*  FROM 성적  WHERE 과목코드 LIKE 'B%';        --학생테이블에서 학과가 NULL이 아닌 학생을 검색할시    SELECT \*  FROM 학생  WHERE 학과 IS NOT NULL;        --학생테이블에서 학번, 이름,학년 검색시 학년은 오름차순으로,  --같은 학번일 때는 학번으로 내림차순 정렬할 시    SELECT 학번, 이름, 학년  FROM 학생  ORDER BY 학년 ASC, 학번 DESC;        --성적테이블에서 중간고사 평균 및 기말 고사 합계 점수를 출력할 시    SELECT AVG(중간고사), SUM(기말고사)  FROM 성적;        --성적테이블에서 과목코드별 기말고사 평균을 검색할시    SELECT AVG(기말고사)  FROM 성적  GROUP BY 과목코드;        --성적테이블에서 두사람이상 등록한 과목의 중간고사 평균을 검색할시    SELECT AVG(중간고사)  FROM 성적  GROUP BY 과목  HAVING COUNT(\*) >= 2;        --성적테이블에서 중간고사 점수가 70이상 90이하인 학생의 학번과 중간고사 점수 검색    SELECT 학번, 중간고사  FROM 성적  WHERE 중간고사 >= 70 AND 중간고사 <=90; | [cs](http://colorscripter.com/info#e) |

**\*\*그룹함수\*\***

**SUM(속성명) : 지정한 속성값의 합계**

**AVG(속성명) : 지정한 속성값의 평균**

**MAX(속성명) : 지정한 속성값 중 최대값**

**MIN(속성명) : 지정한 속성값 중 최소값**

**COUNT(속성명) : 지정한 속성의 튜플 값**

**COUNT(\*) : NULL 값도 포함한 모든 튜플 수**

**3. 데이터 제어어**

① **GRANT :**특정 데이터와 연산을 특정 사용자만이 수행할 수 있게 권한을 부여

형식

**GRANT 권한 ON 데이터 객체 TO 사용자 [WITH GRANT OPTION];**

GRANT OPTION : 다른 사용자에게 권한 부여권을 부여

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5 | --DBA가 사용자1에게 학생 릴레이션에 대한 갱신 권한과 권한 부여권을 부여하는 경우  GRANT UPDATE ON 학생 TO 사용자1 WITH GRANT OPTION;    --DBA가 모든 사용자에게 학생 릴레이션 대한 SELECT 권한을 부여하는 경우  GRANT SELECT ON 학생 TO PUBLIC; | [cs](http://colorscripter.com/info#e) |

**DBA : 데이터베이스관리자 (DataBase Administrator)**

**데이터 정의어와 데이터 제어어를 사용하여 데이터베이스를 설계하고 관리,운영하는 사용자**

② **REVOKE** : 사용자에게 부여된 권한을 취소

형식

**REVOKE 권한 [GRANT OPTION FOR] ON 데이터 객체 FROM 사용자 {CASCADE, RESTRICT};**

CASCADE 사용하면 부여된 권한을 연쇄적으로 취소.

RESTRICT 사용하면 하위 권한 없이 사용자만 권한을 취소.

|  |  |  |
| --- | --- | --- |
| 1  2 | --DBA가 사용자2에게 부여한 권한을 취소하는 경우  REVOKE DELETE ON 학생 FROM 사용자2 CASCADE; | [cs](http://colorscripter.com/info#e) |

**트랜잭션(Transaction)**

- **개념** : **데이터베이스 작업의 논리적인 단위**로 **일련의 연산 집합**을 의미

**데이터베이스의 병행 제어 및 회복 작업의 기본 단위**

     하나의 transaction은 commit되거나 rollback되어야 한다.

**병행 제어(concurrency control)?**

몇 개의 트랜잭셔을 동시에 수행시키는 것

데이터베이스 처리의 성능을 향상시킬 수 있지만 여러 트랜잭션들을 적절히 제어하지 못했을 때는 문제점 발생

**회복(Recovery)?**

장애(failure)가 일어났을 때 데이터베이스를 장애 발생 이전의 일관된 상태로 복원시키는 것

- **특성(ACID)**

① 원자성(Atomicity) : 트랜잭션은 완벽하게 수행 완료되거나, 전혀 수행되지 않아야 한다.

② 일관성(Consistency) : 트랜잭션 수행 전과 수행 후의 데이터베이스 상태가 일치해야 한다.

③ 격리성(Isolation) : 병행수행 환경에서 하나의 트랜잭션이 실행 중일 때 다른 트랜잭션은 같은 데이터에 접근할 수 없다.

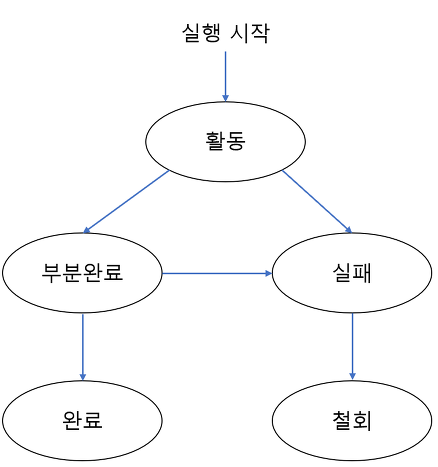
④ 영속성(Durability) : 완료된 트랜잭션의 결과는 영구적으로 유지되어야 한다.

-**연산 (이전에 말한 DCL, database control language)**

① Commit : 트랜잭션의 실행이 성공적으로 종료되었음을 선언하는 연산

② Rollback : 트랜잭션의 실행이 실패하였음을 선언하는 연산

- **트랜잭션의 상태전이**



⒜ 활동(active) : 트랜잭션이 실행을 시작하였거나 실행 중인 상태

⒝ 부분완료(partially committed) : 트랜잭션이 마지막 명령문을 실행한 직후의 상태

⒞ 실패(failed) : 정상적 실행을 더 이상 계속할 수 없어서 중단한 상태

⒟ 철회(aborted) : 트랜잭션이 실행에 실패하여 Rollback 연산을 수행한 상태

⒠ 완료(committed) : 트랜잭션이 실행을 성공적으로 완료하여 Commit 연산을 수행한 상태

**뷰 (View)**

**- 개념**

사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된,

이름을 자니는 가상 테이블(virtual table)

**- 특징**

① 저장장치 내에 물리적으로 구현되어 있지 않음

② 필요한 데이터만 뷰로 정의해서 처리할 수 있기 때문에 관리가 용이하고 명령문이 간단해짐

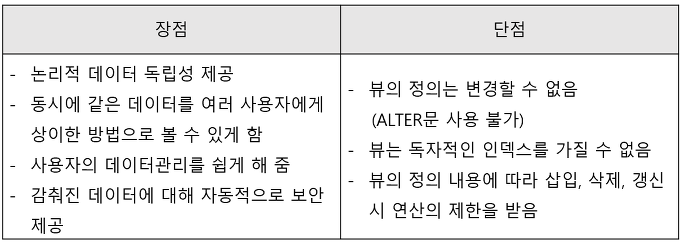
    (뷰를 이용한 검색 연산은 가능하지만 갱신 엽산은 많은 제약점 가짐)

③ 데이터 보정 작업, 처리 과정 시험 등 임시적인 작업을 위한 용도로 활용

④ 뷰는 다른 뷰를 가질 수 있음

⑤ 뷰가 정의된 기본 테이블이나 뷰를 삭제하면 그 테이블이나 뷰를 기초로 정의된 다른 뷰도 자동으로 삭제

**- 장&단점**



**- 뷰 정의**

형식

**CREATE VIEW  뷰\_이름 [(속성목록)] AS SELECT문;**

뷰 이름 뒤에 속성 목록을 작성하지 않으면 SELECT 절의 속성명이 자동 할당 된다.

EX) 학생 테이블에서 '컴퓨터'학과의 학번, 이름, 학년을 '컴공학생' 이라는 뷰로 정의할 시

|  |  |  |
| --- | --- | --- |
| 1  2  3  4 | CREATE VIEW 컴공학생(학번, 이름, 학년)      AS  SELECT 학번, 이름, 학년          FROM 학생          WHERE 부서='컴퓨터'; | [cs](http://colorscripter.com/info#e) |

**- 뷰 삭제**

형식

**DROP VIEW 뷰\_이름{RESTRICT | CASCADE};**

RESTRICT : 뷰가 다른 곳에서 참조되지 있지 않는 한 데이터베이스에서 제거한다.

    만약 다른 곳에서 참조 중이면 삭제 명령은 취소된다.

CASCADE : 뷰를 참조하는 다른 모든 뷰나 제약 조건을 함계 삭제한다.

EX) '컴공학생' 뷰를 제약 조건과 관계없이 삭제할 시

|  |  |  |
| --- | --- | --- |
| 1 | DROP VIEW 컴공학생 CASCADE; | [cs](http://colorscripter.com/info#e) |