데이터베이스 및 설계

Chap 6. SQL

2012.05.11.

오 병 우

컴퓨터공학과

kit 국립금오공과대학교
Kumoh National Institute of Technology

SQL의 역사

- SEQUEL(Structured English QUEry Language)에 연유
  - 1974년, IBM 연구소에서 발표
  - IBM은 'SYSTEM R' 의 인터페이스로 설계 구현
    - 실험적 관계 데이터베이스 시스템 인터페이스
- 표준 SQL
  - 1986년, SQL-86 또는 SQL1
  - 1992년 개정, SQL/92, SQL-92 또는 SQL2
  - 다음 버전 : SQL3, SQL-99
- 현재
  - 상용 DBMS인 DB2와 SQL/DS의 데이터 언어로 사용
  - ORACLE, INFORMIX, SYBASE 등과 같은 다른 회사에서도 채택
  - 미국 표준 연구소(ANSI)와 국제 표준 기구(ISO)에서 관계 데이터베이스의 표준 언어로 채택

SQL(Structured Query Language)

- SQL의 의미
  - 구조화 질의어
  - 종합 데이터베이스 언어 역할
    - 단순히 검색만을 위한 데이터 질의어가 아님
  - 데이터 정의어(DDL), 데이터 조작어(DML), 데이터 제어어(DCL)의 기능 모두 제공

SQL의 특징

- Relational algebra의 특징포함 + 확장된 relational calculus 기초
- 고급 비 절차적 데이터 언어
  - 사용자 친화적인 인터페이스 제공
- SQL의 표준화
  - 상용 RDBMS간의 전환 용이
  - 관계 데이터베이스를 접근하는 데이터베이스 응용 프로그램을 작성 기능 제공
- 온라인 터미널을 통해 대화식 질의어로 사용

SQL의 특징(2)

- 응용 프로그램에 삽입된 형태로도 사용 가능
  - Java, COBOL, C/C++ 등과 같은 범용 프로그래밍 언어로 된 응용 프로그램
- 개개의 레코드 단위로 처리하기 보다는 레코드 집합 단위로 처리
- 선언적 언어
  - SQL 명령문에는 데이터 처리를 위한 접근 경로(access path)에 대한 명세가 불필요

Note

관계 모델의 공식적 용어 대신 일반적인 용어 사용

릴레이션 – 테이블, 투플 – 행, 애트리뷰트 – 열

SQL 데이터 정의문

- 스키마와 카탈로그
  - 스키마
    - 하나의 응용(사용자)에 속하는 테이블과 기타 구성요소 등의 그룹
    - 스키마 이름, 스키마 소유자나 허가권자, (테이블, 뷰, 도메인, 기타 내용) 포함
  - **CREATE SCHEMA UNIVERSITY AUTHORIZATION SHLEE ;**

※ 실제로 **CREATE SCHEMA** 보다 **CREATE DATABASE** 명령문을 씀

use information_schema;

show tables;

  - 카탈로그
    - 한 SQL 시스템에서의 스키마들의 집합
    - Information_schema : 그 카탈로그에 속한 모든 스키마에 대한 정보 제공

도메인 정의문 (1)

- 일반 형식
  - **CREATE DOMAIN** 도메인_이름 데이터_타입
    [기정 값_정의]
    [도메인_제약조건_정의리스트];
    - ex) **CREATE DOMAIN** Dept **CHAR**(4)
      DEFAULT '???'
      CONSTRAINT VALID-DEPT
      CHECK(VALUE IN
      ('COMP', 'ME', 'EE', 'ARCH', '???'));

이것 중 하나

  - **ALTER DOMAIN** 도메인_이름 <변경 내용>

참조하는 곳이 없을 때

따라가서 삭제

  - **DROP DOMAIN** 도메인_이름 **RESTRICT** | **CASCADE** ;

도메인 정의문 (2)

- 데이터 타입
- 시스템 데이터 타입만 사용
  - 숫자
    - INTEGER, SMALLINT : 정수
    - FLOAT(n), REAL, DOUBLE PRECISION : 실수
    - DECIMAL(i, j), NUMERIC(i, j) : 정형 숫자
  - 문자 스트링
    - CHAR(n) : 고정 길이 문자
    - VARCHAR(n) : 가변 길이 문자
  - 비트 스트링
    - BIT(n), BIT VARYING(n)
  - 날짜
    - DATE : YY-MM-DD
  - 시간
    - TIME : hh:mm:ss
    - TIMESTAMP : DATE와 TIME 포함
    - INTERVAL : DATE, TIME, TIMESTAMP 포함

기본 테이블의 생성 (1)

- 테이블의 종류
  - 기본 테이블 (base table)
    - 원래 DDL에 의해 만들어지는 테이블
    - 독자적으로 존재 가능
  - 뷰 (view)
    - DDL로 만들어지지만 독자적으로 존재 불가
    - 어떤 기본 테이블로부터 유도(derived)되어 만들어지는 가상 테이블 (virtual table)
  - 임시 테이블 (temporary table)
    - DDL에 의해 만들어지는 것이 아님
    - 질의문 처리 과정의 중간 결과로 만들어지는 테이블

기본 테이블의 생성 (2)

- 일반형식
  - **CREATE TABLE** 기본테이블

    ({열이름 데이터타입 [**NOT NULL**] [**DEFAULT** 값],}$^{+}$

    [**PRIMARY KEY** (열이름_리스트),]

    {[**UNIQUE** (열이름_리스트),]}*

    {[**FOREIGN KEY**(열이름_리스트)

    REFERENCES 기본테이블[(열이름_리스트)]

    [**ON DELETE** 옵션]

    [**ON UPDATE** 옵션] ,]}*

    [**CONSTRAINT** 이름] [**CHECK**(조건식)]);

기본 테이블의 생성 (3)

- 예제

```
CREATE TABLE ENROL
        ( Sno   DSNO   NOT NULL,
          Cno   DCNO   NOT NULL,
          Grade  INTEGER,
          PRIMARY KEY(Sno,Cno),
          FOREIGN KEY(Sno) REFERENCES STUDENT(sno)
                          ON DELETE  CASCADE
                          ON UPDATE  CASCADE,
          FOREIGN KEY(Cno) REFERENCES COURSE
                          ON DELETE  CASCADE
                          ON UPDATE  CASCADE,
          CHECK(Grade □ 0 AND Grade □ 100));
```

기본 테이블의 제거와 변경 (1)

- 기본 테이블의 제거
  - ◈ 일반 형식

DROP TABLE 기본_테이블_이름
{ **RESTRICT** | **CASCADE** } ;

DROP TABLE COURSE **CASCADE**;

- 스키마 제거

  - ◈ 일반형식

DROP SCHEMA 스키마_이름
{ **RESTRICT** | **CASCADE** };

DROP SCHEMA UNIVERCITY **CASCADE**;

기본 테이블의 제거와 변경 (2)

- 기본 테이블의 변경
  - 일반형식

ALTER TABLE 기본_테이블_이름
([**ADD** 열_이름 데이터_타입][**DEFAULT** 기정 값] |
[**DROP** 열_이름] [**CASCADE**] |
[**ALTER** 열_이름 (**DROP DEFAULT** |
SET DEFAULT 기정 값)]);

  - 예제
    - **ALTER TABLE** ENROL
      ADD Final **CHAR** **DEFAULT** 'F';
    - **ALTER TABLE** ENROL
      DROP Grade **CASCADE**;

Example

- 대학(University) 관계 데이터베이스

학생 (STUDENT)

학번 (Sno)	이름 (Sname)	학년 (Year)	학과 (Dept)
100	나 수 영	4	컴퓨터
200	이 찬 수	3	전기
300	정 기 태	1	컴퓨터
400	송 병 길	4	컴퓨터
500	박 종 화	2	산공

과목 (COURSE)

과목번호 (Cno)	과목이름 (Cname)	학점 (Credit)	학과 (Dept)	담당교수 (PRname)
C123	프로그래밍	3	컴퓨터	김성국
C312	자료구조	3	컴퓨터	황수관
C324	화일구조	3	컴퓨터	이규찬
C413	데이터베이스	3	컴퓨터	이일로
E412	반 도 체	3	전자	홍봉진

Example (cont'd)

- 대학(University) 관계 데이터베이스

등록 (ENROL)

학번 (Sno)	과목번호 (Cno)	성적 (Grade)	중간성적 (Midterm)	기말성적 (Final)
100	C413	A	90	95
100	E412	A	95	95
200	C123	B	85	80
300	C312	A	90	95
300	C324	C	75	75
300	C413	A	95	90
400	C312	A	90	95
400	C324	A	95	90
400	C413	B	80	85
400	E412	C	65	75
500	C312	B	85	80

SQL 데이터 조작문

- 데이터 검색
  - 기본 구조

```
SELECT    열_리스트
 FROM     테이블_리스트
 WHERE    조건;
```

  - 예

```
SELECT    Sname, Sno
 FROM     STUDENT
 WHERE    Dept = '컴퓨터';
```

실행 결과

Sname	Sno
나수영	100
정기태	300
송병길	400

```
SELECT    STUDENT.Sname, STUDENT.Sno
FROM      STUDENT
 WHERE    STUDENT.Dept = '컴퓨터';
```

데이터 검색 (1)

- 폐쇄 시스템 (closed system)
  - 기존 테이블 처리 결과가 또 다른 테이블이 되는 시스템
  - 중첩 질의문(nested query)의 이론적 기초

- SQL과 이론적 관계 모델의 차이점
  - SQL의 테이블
    - 한 테이블 내에 똑같은 레코드(행) 중복 가능
    - Primary key를 반드시 가져야 하는 것은 아님
    - 이론상 SQL의 테이블 ≠ 투플의 집합
    - 같은 원소의 중복을 허용하는 다중 집합(multiset) 또는 백(bag)
      → DISTINCT 명세 : 집합과 같은 결과를 만듦

데이터 검색 (2)

- 일반적인 형식
  - **SELECT** [**ALL** | **DISTINCT**] 열_리스트
    FROM 테이블_리스트
    [**WHERE** 조건]
    [**GROUP BY** 열_리스트
    [**HAVING** 조건]]
    [**ORDER BY** 열_리스트 [**ASC** | **DESC**]];
- 검색 결과에 레코드의 중복 제거
  - **SELECT DISTINCT** Dept
    FROM STUDENT;
- 테이블의 열 전부를 검색하는 경우
  - **SELECT** *
    FROM STUDENT;

데이터 검색 (3)

- 조건 검색
  - **SELECT** Sno,Sname
    FROM STUDENT
    WHERE Dept = '컴퓨터' **AND** Year = 4;
- 순서를 명세하는 검색
  - **SELECT** Sno, Cno
    FROM ENROL
    WHERE Midterm □ 90
    ORDER BY Sno **DESC**, Cno **ASC**;

교재 p.142
(구교재 p.183)

- 산술식과 문자 스트링이 명세된 검색
  - **SELECT** Sno **AS** 학번, '중간시험 = ' **AS** 시험, Midterm + 3 **AS** 점수
    FROM ENROL
    WHERE Cno = 'C312';

데이터 검색 (4)

- 복수 테이블로부터의 검색(조인)
  - **SELECT** S.Sname, S.Dept, E.Grade
    FROM STUDENT S, ENROL E
    WHERE S.Sno = E.Sno **AND** E.Cno = 'C413';

- 자기 자신의 테이블에 조인하는 검색
  - **SELECT** S1.Sno, S2.Sno
    FROM STUDENT S1, STUDENT S2
    WHERE S1.Dept = S2.Dept
    AND S1.Sno < S2.Sno;

데이터 검색 (5)

- **FROM** 절에 조인 명세
  - **SELECT** Sname, Dept, Grade
    FROM STUDENT **JOIN** ENROL **ON**
    (STUDENT.Sno=ENROL.Sno)
    WHERE ENROL.Cno = 'C413';
  - **SELECT** Sname, Dept, Grade
    FROM STUDENT **JOIN** ENROL **USING**(Sno)
    WHERE ENROL.Cno = 'C413';
  - **SELECT** Sname, Dept, Grade
    FROM STUDENT **NATURAL JOIN** ENROL
    WHERE ENROL.Cno = 'C413';

데이터 검색 (6)

- 집계 함수(aggregate function)를 이용한 검색
  - 집계 함수: **COUNT, SUM, AVG, MAX, MIN**

```
– SELECT   COUNT(*)  AS  학생수
  FROM     STUDENT;
```

```
– SELECT   COUNT(DISTINCT Cno)
  FROM     ENROL
  WHERE    Sno = 300;
```

```
– SELECT   AVG(Midterm) AS 중간평균
  FROM     ENROL
  WHERE    Cno = 'C413';
```

데이터 검색 (7)

- **GROUP BY**를 이용한 검색
  - **SELECT** Cno, **AVG**(Final) **AS** 기말평균
    FROM ENROL
    GROUP BY Cno;

- **HAVING**을 사용한 검색
  - **SELECT** Cno, **AVG**(Final) **AS** 평균
    FROM ENROL
    GROUP BY Cno
    HAVING **COUNT**(*) >= 3;

데이터 검색 (8)

- 부속 질의문(Subquery)을 사용한 검색

  - 부속 질의문
    - 다른 질의문에 중첩(nested)되어 사용된 검색문
    - 형태 : **SELECT-FROM-WHERE-GROUP BY-HAVING**
    - 중첩 질의문 : 부속 질의문을 포함하고 있는 질의문
    - **IN** 다음에 사용 : 집합의 멤버십 연산자(∈)로 해석 가능

  - **SELECT** Sname
    FROM STUDENT
    WHERE Sno **IN**
    (**SELECT** Sno
    FROM ENROL
    WHERE Cno = 'C413');

데이터 검색 (9)

- 부속 질의문을 사용한 검색(cont'd)
  - **SELECT** Sname
    FROM STUDENT
    WHERE Sno **NOT IN**
    (**SELECT** Sno
    FROM ENROL
    WHERE Cno = 'C413');
  - **SELECT** Sname, Dept
    FROM STUDENT
    WHERE Dept =
    (**SELECT** Dept
    FROM STUDENT
    WHERE Sname = '정기태');

데이터 검색 (10)

- 부속 질의문을 사용한 검색(cont'd)
  - **SELECT** Sno, Cno
    FROM ENROL
    WHERE Final > **ALL**
    (**SELECT** Final
    FROM ENROL
    WHERE Sno = 500);

데이터 검색 (11)

- **LIKE**를 사용하는 검색
  - **LIKE**
    - 서브 스트링 패턴(substring pattern) 비교 연산자
  - **%**
    - 어떤 길이의 어떤 문자 스트링도 관계 없음을 의미
  - **_ (underbar)**
    - 문자 하나를 의미

```
SELECT    Cno, Cname
FROM      COURSE
WHERE     Cno LIKE  'C%';
```

데이터 검색 (12)

- **NULL**을 사용한 검색
  - **NULL**
    - 누락된 정보(missing information)
    - 값은 있지만 모르는 값(unknown value)
    - 해당되지 않는 값(unapplicable value)
    - 의도적으로 유보한 값(withheld value)
  - **NULL**이 추가된 3-값 논리(3-VL: 3-value logic)
    - 논리값으로 보면 참(true)도 거짓(false)도 아닌 미정(unknown)

AND	T	F	U
T	T	F	U
F	F	F	U
U	U	F	U

OR	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

NOT	
T	F
F	T
U	U

데이터 검색 (13)

- **NULL**을 사용한 검색 (cont'd)
  - **SELECT** Sno, Sname
    FROM STUDENT
    WHERE Dept **IS NULL**;
  - "열_이름 **IS [NOT] NULL**"의 형식만 허용
    - "열_이름 = **NULL**"의 형식은 불법적 형식
  - 널값 : 조건식에서 비교연산자와 같이 사용 → 항상 거짓

Year의 값이 널인 경우 다음은 모두 거짓이거나 불법

Year > 3
Year ≤ 3
Year = 3
Year ≠ 3
Year = NULL (불법적 형식)
Year ≠ NULL (불법적 형식)

데이터 검색 (14)

- **EXISTS**를 사용하는 검색
  - 과목 'C413'에 등록한 학생의 이름을 검색하라.

```
SELECT  Sname
FROM    STUDENT
WHERE   EXISTS
          (SELECT      *
           FROM        ENROL
           WHERE       Sno = STUDENT.Sno
                       AND     Cno = 'C413');
```

Note : **EXISTS** 이하 **SELECT** 문이 참(공집합이 아님)일 때 본 **SELECT** 문을 실행

데이터 검색 (15)

- **EXISTS**를 사용하는 검색(cont'd)
  - 과목 'C413'에 등록하지 않은 학생의 이름을 검색하라.
  -
```
SELECT Sname
FROM   STUDENT
WHERE  NOT EXISTS
          (SELECT     *
           FROM       ENROL
           WHERE      Sno = STUDENT.Sno
                      AND    Cno = 'C413');
```

데이터 검색 (16)

- **UNION**이 관련된 검색
  - ```
    SELECT Sno
    FROM   STUDENT
    WHERE  Year = 1
       UNION
    SELECT Sno
    FROM   ENROL
    WHERE  Cno = 'C324';
    ```

Note : 중복되는 투플은 제거
    ```

데이터의 갱신 (1)

- 일반적인 형식
  - **UPDATE** 테이블
    SET { 열_이름 = 산술식}’+
    [**WHERE** 조건];

- 하나의 레코드 변경
  - **UPDATE** STUDENT
    SET Year = 2
    WHERE Sno = 300;

- 복수의 레코드 변경
  - **UPDATE** COURSE
    SET Credit = Credit + 1
    WHERE Dept = '컴퓨터';

데이터의 갱신 (2)

- 부속 질의문을 이용한 변경
  - **UPDATE** ENROL
    SET Final = Final + 5
    WHERE Sno **IN**
    (**SELECT** Sno
    FROM STUDENT
    WHERE Dept = '컴퓨터');

  - **UPDATE** STUDENT
    SET Dept = (**SELECT** Dept
    FROM COURSE
    WHERE Cname = '데이터베이스')
    WHERE Year = 4;

데이터의 삽입 (1)

- 일반 형식
  - **INSERT**
    INTO 테이블 [(열_이름_리스트)]
    VALUES (열값_리스트);
  - **INSERT**
    INTO 테이블 [(열_이름_리스트)]
    SELECT문;

데이터의 삽입 (2)

- 레코드의 직접 삽입
  - **INSERT**
    INTO STUDENT(Sno, Sname, Year, Dept)
    VALUES (600, '박상철', 1, '컴퓨터');
  - **INSERT**
    INTO STUDENT
    VALUES (600, '박상철', 1, '컴퓨터');
  - **INSERT**
    INTO STUDENT(Sno, Sname, Year)
    VALUES (600, '박상철', 1);

데이터의 삽입 (3)

- 부속 질의문을 이용한 레코드 삽입
  - **INSERT**
    INTO COMPUTER(Sno, Sname, Year)
    SELECT Sno, Sname, Year
    FROM STUDENT
    WHERE Dept = '컴퓨터';

데이터의 삭제 (1)

- 일반 형식
  - **DELETE**
    FROM 테이블
    [WHERE 조건];

- 하나의 레코드 삭제
  - **DELETE**
    FROM STUDENT
    WHERE Sno = 100;

Note : Primary key와 referential integrity(참조 무결성) 문제

데이터의 삭제 (2)

- 복수의 레코드 삭제
  - **DELETE**
    FROM ENROL;

- 부속 질의문을 사용한 삭제
  - **DELETE**
    FROM ENROL
    WHERE Cno = 'C413' **AND** Final < 60
    AND ENROL.Sno **IN**
    (**SELECT** Sno
    FROM STUDENT
    WHERE Dept = '컴퓨터');

SQL 뷰

- 하나 또는 둘 이상의 기본 테이블(base table)로부터 유도되어 만들어지는 가상 테이블 (Virtual table)
- 외부 스키마는 뷰와 기본 테이블들의 정의로 구성됨
- 기본 테이블을 들여다보는 '유리창' (window)
  - 동적임 (dynamic)
    - cf.) Snapshot: static
- 물리적인 구현이 아님
  - 뷰의 정의만 시스템 카탈로그(SYSVIEWS)에 **SELECT-FROM-WHERE**의 형태로 저장됨
- 뷰에 대한 변경 → 테이블에 대한 변경 (제한적임)
  - 불가능할 수 있음 (derived attribute)
  - Error가 발생할 수 있음 (with check option)
  - Null로 채워질 수 있음 (뷰에 없는 attribute)

뷰의 생성 (1)

- 일반형식
  - **CREATE VIEW** 뷰_이름[(열_이름 리스트)]
    AS SELECT문
    [**WITH CHECK OPTION**];

- **WITH CHECK OPTION**
  - 갱신이나 삽입 연산 시 조건 확인
  - **CREATE VIEW** CSTUDENT2(Sno, Sname, Dept)
    AS SELECT Sno, Sname, Dept
    FROM STUDENT
    WHERE Dept = '컴퓨터'
    WITH CHECK OPTION;
  - insert into CSTUDENT2 values (600, '아무로', '메카트로닉스')
    - Error 발생

뷰의 생성 (2)

```
CREATE  VIEW  CSTUDENT(Sno, Sname, Year)
    AS      SELECT      Sno, Sname, Year
              FROM      STUDENT
              WHERE     Dept = '컴퓨터';
```

기본 테이블 학생(STUDENT)의 컴퓨터과 학생(CSTUDENT) 뷰

컴퓨터과 학생 (CSTUDENT)

학번 Sno	이름 Sname	학년 Year	학과 Dept
100	나수영	4	컴퓨터
200	이찬수	3	전기
300	정기태	1	컴퓨터
400	송병길	4	컴퓨터
500	박종화	2	산공

뷰의 생성 (3)

- 예 (cont'd)

```
CREATE  VIEW DEPTSIZE(Dept, Tstdn)
   AS       SELECT  Dept, COUNT(*)
            FROM    STUDENT
            GROUP  BY Dept;
```

  - **AS SELECT** : 열의 이름 상속
    상속 불가한 경우나 열 이름이 중복될 경우 반드시 열 이름 명세

```
CREATE  VIEW  HONOR(Sname, Dept, Grade)
   AS       SELECT      STUDENT.Sname,
                        STUDENT.Dept, ENROL.Final
            FROM        STUDENT, ENROL
            WHERE       STUDENT.Sno = ENROL.Sno
                          AND  ENROL.Final > 90;
```

  - 두 개 이상 테이블 조인

뷰의 생성 (4)

- 예 (cont')

```
CREATE  VIEW        COMHONOR
   AS   SELECT      Sname
        FROM        HONOR
        WHERE       Dept = '컴퓨터';
```

  - 정의된 뷰를 이용하여 또 다른 뷰 정의

뷰의 제거 (1)

- 일반형식
  - **DROP VIEW** 뷰_이름 { **RESTRICT | CASCADE** };
    - **RESTRICT**
      - 다른 곳에서 참조되고 있지 않는 한 데이터베이스에서 제거
    - **CASCADE**
      - 이 뷰가 사용된 다른 모든 뷰나 제약 조건이 함께 제거
- 예
  - **DROP VIEW** DEPTSIZE **RESTRICT**;
- Note : “Propagated Destroys”
  - 기본 테이블이 제거되면 그 위에 만들어진 인덱스나 뷰도 자동적으로 제거됨

뷰의 조작연산 (1)

- 기본 테이블에 사용 가능한 검색(**SELECT**)문도 뷰에 사용가능
- 변경(삽입, 삭제, 갱신) 연산은 제약

◈ 열 부분 집합 뷰(column subset view)

- **CREATE VIEW** STUDENT_VIEW1
  AS SELECT Sno, Dept
  FROM STUDENT;

  → Primary key 포함 : 이론적으로 삽입, 삭제, 갱신, 검색 가능

- **CREATE VIEW** STUDENT_VIEW2
  AS SELECT Sname, Dept
  FROM STUDENT;

  → Primary key 불포함 : 이론적으로 삽입, 삭제, 갱신, 검색 불가

◈ 행 부분 집합 뷰(row subset view)

- **CREATE VIEW** STUDENT_VIEW3
  AS SELECT Sno, Sname, Year, Dept
  FROM STUDENT
  WHERE Year=4;

뷰의 조작연산 (2)

◈ 조인 뷰(join view)

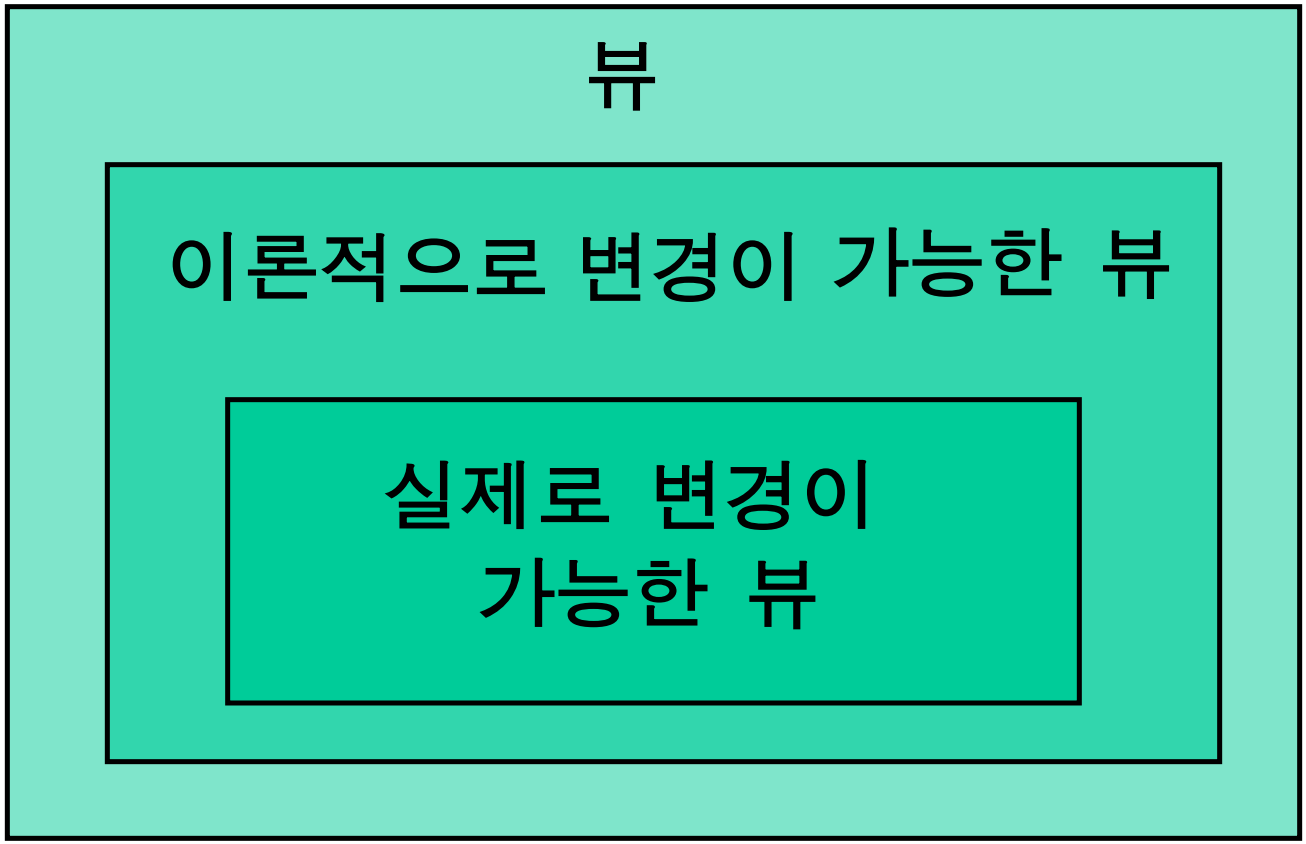
```
– CREATE  VIEW      HONOR(Sname, Dept, Grade)
      AS  SELECT    STUDENT.Sname,
                    STUDENT.Dept, ENROL.Final
          FROM      STUDENT, ENROL
          WHERE     STUDENT.Sno = ENROL.Sno
            AND     ENROL.Final > 95;
```

◈ 통계적 요약 뷰(statistical summary view)

```
– CREATE  VIEW      COSTAT(Cno, Avpoint)
    AS  SELECT      Cno, AVG(Midterm)
          FROM      ENROL
          GROUP BY  Cno;
```

뷰의 조작연산 (3)

- 뷰는 제한적인 갱신만 가능함



뷰의 조작연산 (4)

- 변경 연산이 허용되지 않는 경우
  ① 뷰의 열이 상수나 산술 연산자 또는 함수가 사용된 산술 식으로 만들어질 경우

  ② 집계 함수(**COUNT, SUM, AVG, MAX, MIN**)가 관련되어 정의된 경우

  ③ **DISTINCT, GROUP BY, HAVING**이 사용되어 정의된 경우

  ④ 두 개 이상의 테이블이 관련되어 정의된 경우

  ⑤ 변경할 수 없는 뷰를 기초로 정의된 경우

뷰의 장단점

- 뷰의 장점
  - 논리적 독립성을 제공 (확장, 구조 변경)
  - 데이터의 접근을 제어 (보안)
  - 사용자의 데이터 관리를 단순화
  - 여러 사용자에 다양한 데이터 요구를 지원

- 뷰의 단점
  - 정의를 변경할 수 없음
  - 삽입, 삭제, 갱신 연산에 제한이 많음