10. SQL

SQL

- SEQUEL(Structured English Query Language)
  - 1974, IBM
  - SYSTEM R

- SQL(Structured Query Language)
  - ANSI와 ISO의 데이타베이스 표준언어
  - DB2, SQL/DS

- Select-From-Where의 블럭 사상 (block mapping)을 이용

10.1 SQL 데이터 정의문

◻ 스키마와 카탈로그 (SQL2에서의 정의)

- 스키마
  - 하나의 응용(사용자)에 속하는 테이블과 기타 구성요소 등의 그룹
  - 스키마 이름, 스키마 소유자나 허가권자, (테이블, 뷰, 도메인, 기타 내용) 포함
- CREATE SCHEMA 명령문
  - CREATE SCHEMA UNIVERSITY AUTHORIZATION SHLEE;
  - 현재는 CREATE SCHEMA 보다 CREATE DATABASE 명령문을 씀
- 카탈로그
  - 한 SQL 환경에서의 스키마들의 집합

도메인 정의문

□ 일반 형식

- CREATE DOMAIN 도메인_이름 데이타타입
  [묵시값_정의]
  [도메인_제약조건_정의리스트];
  - CREATE DOMAIN DEPT CHAR(4)
    DEFAULT '???'
    CONSTRAINT VALID-DEPT
    CHECK (VALUE IN 'COMP', 'ME', 'EE', 'ARCH', '???'));

- DROP DOMAIN 도메인_이름 RESTRICT | CASCADE;
  - RESTRICT : 삭제할 도메인을 참조하는 곳이 없을 때만 삭제 실행
  - CASCADE : 삭제할 도메인을 참조하는 모든 곳에서 삭제 실행

SQL 데이타 타입

- 숫자 타입
  - INT, SMALLINT : 정수
  - FLOAT(n), REAL, DOUBLE PRECISION : 실수
  - DECIMAL(i, j), NUMERIC(i, j) : 정형 숫자
- 문자 스트링
  - CHAR(n), VARCHAR(n) : 문자
- 비트 스트링
  - BIT(n), BIT VARYING(n)
- 날짜, 시간
  - DATE : 날짜 (YY-MM-DD)
  - TIME : 시간 (hh:mm:ss)
  - TIMESTAMP : DATE & TIME
  - INTERVAL : DATE, TIME & TIMESTAMP

기본 테이블의 생성

□ 일반형식

- CREATE TABLE 기본테이블_이름
  ({열이름 도메인명|타입 [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 조건이름] [CHECK (조건식)]);
- Options
  - PRIMARY: primary key 명세
  - UNIQUE: alternate key 명세
  - FOREIGN: foreign key 명세
    - 옵션: NO ACTION, CASCADE, SET NULL, SET DEFAULT
  - CONSTRAINT: 테이블의 행이 갱신될때 유지되야할 제약조건 명세

□ 예

```
- CREATE  TABLE  ENROL
          ( SNO  DSNO  NOT NULL,
            CNO  DCNO  NOT NULL,
            GRADE  INTEGER,
            PRIMARY KEY(SNO,CNO),
            FOREIGN KEY(SNO) REFERENCES STUDENT
                        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

◘ 기본 테이블의 제거

- 일반형식
  DROP TABLE 기본테이블_이름 {RESTRICT | CASCADE};

  - ◆ RESTRICT: 기본 테이블이 다른 곳에서 참조되는한 제거되지 않음
  - ◆ CASCADE: 기본 테이블을 참조하는 다른 테이블도 함께 제거

  - ◆ DROP TABLE COURSE CASCADE;
    - ENROL 테이블도 함께 제거됨.

◘ 스키마 제거

- 일반형식
  DROP SCHEMA 스키마_이름 {RESTRICT | CASCADE};

◘ 기본 테이블의 변경

- 변경내용
  - ◆ 열의 첨가/삭제, 묵시값 명세 첨가/삭제, 기본테이블에 새로운 무결성 조건 명세 첨가/삭제 등
- 일반형식

  ALTER TABLE 기본테이블_이름
  ([ADD 열_이름 데이타_타입] [DEFAULT 묵시값] |
  [DROP 열_이름] [CASCADE] |
  [ALTER 열_이름 (DROP DEFAULT | SET DEFAULT 묵시값)]);

  - ◆ ALTER TABLE ENROL
    ADD FINAL CHAR DEFAULT 'F';
  - ◆ ALTER TABLE ENROL
    DROP GRADE CASCADE;

10.2 SQL 데이타 조작문

- 데이타 검색(retrieve, query)
- 데이터 갱신(update)
  - 데이터 수정(modify)
  - 데이터 삽입(insert)
  - 데이터 삭제(delete)

데이타 검색

□ 데이타 검색

- 기본구조

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

- 예

```
SELECT  SNAME, SNO
FROM    STUDENT
WHERE   DEPT = '컴퓨터';
```

- 일반 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 중복제거
  - SELECT DISTINCT DEPT
    FROM STUDENT;

- 테이블의 열 전부를 검색하는 경우
  - SELECT *
    FROM STUDENT;

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;

산술식과 문자 스트링이 명세된 검색

- SELECT SNO AS 학번, '중간시험 = ' AS 시험,
  MIDTERM + 3 AS 점수
  FROM ENROL
  WHERE CNO = 'C312';

복수 테이블로부터의 검색

- SELECT STUDENT.SNAME, STUDENT.DEPT, ENROL.GRADE
  FROM STUDENT, ENROL
  WHERE STUDENT.SNO = ENROL.SNO AND ENROL.CNO = 'C413';

자기자신의 테이블에 조인하는 검색

- 범위 변수(range variable) 이용
- SELECT S1.SNO, S2.SNO
  FROM STUDENT S1, STUDENT S2
  WHERE S1.DEPT = S2.DEPT AND S1.SNO < S2.SNO;

□ 집단 함수(Aggregate Function)를 이용한 검색

- 집단 함수: COUNT, SUM, AVG, MAX, MIN
- SELECT COUNT(*) AS 학생수
  FROM STUDENT;
- SELECT COUNT(DISTINCT CNO)
  FROM ENROL
  WHERE SNO = '300';

□ GROUP BY를 이용한 검색

- SELECT CNO, AVG(FINAL) AS 기말평균
  FROM ENROL
  GROUP BY CNO;

HAVING을 사용한 검색

- GROUP BY와 함께 사용되며, 각 그룹의 구성요건을 명세.
  GROUP BY가 생략되면, 전체 테이블을 하나의 그룹으로 취급
- "세사람 이상 등록한 과목의 평균 기말성적을 검색하라"

```
SELECT      CNO, AVG(FINAL)  AS  평균
FROM        ENROL
GROUP BY    CNO
HAVING      COUNT(*) ≥ 3;
```

◘ 부속질의어(Subquery)를 사용한 검색 (조인)

-
```
SELECT SNAME
FROM   STUDENT
WHERE  SNO IN                          //혹은 NOT IN, =
           (SELECT SNO
           FROM    ENROL
           WHERE   CNO = 'C413');
```

- 조인으로 표현 가능

```
SELECT STUDENT.SNAME
FROM   STUDENT, ENROL
WHERE  STUDENT.SNO = ENROL.SNO AND
       ENROL.CNO = 'C413';
```

□ 검색 결과의 임시 저장

- WITH TEMP AS (

```
        SELECT   STUDENT.SNO, STUDENT.SNAME,
                 STUDENT.PNO
        FROM     STUDENT, ENROL
        WHERE    STUDENT.SNO = ENROL.SNO AND
                 ENROL.CNO = 'C413')
SELECT  TEMP.SNAME, PROFESSOR.PNAME
FROM    TEMP, PROFESSOR
WHERE   TEMP.PNO = PROFESSOR.PNO;
```

LIKE를 사용하는 검색

- LIKE predicate
  - %: wild string
  - _: wild character
  - SNAME LIKE ‘S%’ 혹은 ‘S__’, ‘%S__’, ‘%S%’ 등
- SELECT CNO, CNAME
  FROM COURSE
  WHERE CNO LIKE 'C%';

NULL을 사용한 검색

- SELECT SNO, SNAME
  FROM STUDENT
  WHERE DEPT IS NULL;
- IS, IS NOT 이외의 NULL을 이용한 모든 조건식은 illegal.
  - YEAR=NULL, YEAR≠NULL 등

□ EXISTS를 사용하는 검색

- 존재 정량자(existential quantifier)
  - ◆ 부속질의문의 결과가 공집합이 아니면 참, 공집합이면 거짓.
- "과목 C413에 등록한 학생의 이름을 검색하라"

```
SELECT  SNAME
FROM    STUDENT
WHERE   EXISTS
        (SELECT *
         FROM    ENROL
         WHERE   SNO = STUDENT.SNO AND
                 CNO = 'C413');
```

UNION이 관련된 검색

- SELECT SNO // Union-compatible
  FROM STUDENT
  WHERE YEAR = 1
  UNION
  SELECT SNO // Union-compatible
  FROM ENROL
  WHERE CNO = 'C324';
- 중복되는 투플은 제거됨.

데이타의 수정

- 일반 형식
  - UPDATE 테이블
    SET { 열_이름 = 산술식,}+
    [WHERE 조건];

- 하나의 레코드 변경
  - UPDATE STUDENT
    SET YEAR = 2
    WHERE SN0 = 300;

복수의 레코드 변경

- UPDATE COURSE
  SET CREDIT = CREDIT + 1
  WHERE DEPT = '컴퓨터';

부속 질의문을 이용한 변경

- UPDATE ENROL
  SET FINAL = FINAL – 5
  WHERE SNO IN
  (SELECT SNO
  FROM STUDENT
  WHERE DEPT = '컴퓨터');

데이타의 삽입

- 일반형식
  - INSERT
    INTO 테이블 [(열_이름_리스트)]
    VALUES (열 값_리스트);
  - INSERT
    INTO 테이블 [(열_이름_리스트)]
    SELECT문;

레코드의 직접 삽입

- INSERT
  INTO STUDENT(SNO, SNAME, YEAR, DEPT)
  VALUES (600, '박 상현', 1, '컴퓨터');

부속 질의문을 이용한 레코드 삽입

- “학생 테이블에서 컴퓨터과 학생의 학번, 이름, 학년을 검색하여, Computer 테이블에 삽입하라.”

```
INSERT
INTO    COMPUTER(SNO, SNAME, YEAR)
SELECT  SNO, SNAME, YEAR
FROM    STUDENT
WHERE   DEPT = '컴퓨터';
```

데이타의 삭제

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

- 하나의 레코드 삭제
  - DELETE
    FROM STUDENT
    WHERE SNO = 100;

- 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');

10.3 SQL 뷰

- 뷰(view)
  - 하나 또는 둘 이상의 기본 테이블(base table)로부터 유도되어 만들어지는 “이름”이 있는 가상 테이블 (virtual table)
  - 기본 테이블을 들여다보는 ’window’
  - 뷰의 정의만 저장, 필요시 동적으로 데이터 생성.
    - 뷰는 실행시간에만 구체화되는 특수한 테이블

- 뷰의 조작
  - 검색: 뷰와 기본 테이블 사이에 차이가 없음
  - 갱신: 뷰의 갱신에는 많은 제약이 따름

- 외부 스키마는 뷰와 기본 테이블들의 정의로 구성됨
  - 뷰의 정의는 시스템 카탈로그(SYSVIEWS)에 SELECT-FROM-WHERE의 형태로 저장됨

뷰의 생성

□ 일반형식

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

- 예

```
CREATE  VIEW  CSTUDENT(SNO, SNAME, YEAR)
        AS    SELECT      SNO, SNAME, YEAR
              FROM        STUDENT
              WHERE       DEPT = '컴퓨터'
        WITH CHECK OPTION;
```

  - ◆ WITH CHECK OPTION절은 이 뷰에 대한 수정/삽입 연산이 실행될때 뷰의 정의 조건(DEPT = '컴퓨터')이 위배되면 실행을 거절한다는 명세

◘ 열 이름의 상속

- CREATE VIEW DEPTSIZE(DEPT, STNUM)
  AS SELECT DEPT, COUNT(*)
  FROM STUDENT
  GROUP BY DEPT;
- CREATE VIEW DEPTSIZE
  AS SELECT DEPT, COUNT(*) AS STNUM
  FROM STUDENT
  GROUP BY DEPT;

□ 두 개 이상의 테이블에서 유도된 뷰

- CREATE VIEW HONOR(SNAME, DEPT, GRADE)
  AS SELECT STUDENT.SNAME, STUDENT.DEPT, ENROL.GRADE
  FROM STUDENT, ENROL
  WHERE STUDENT.SNO = ENROL.SNO AND ENROL.FINAL > 95;

□ 뷰를 이용한 뷰 정의

- CREATE VIEW COMHONOR
  AS SELECT SNAME, GRADE
  FROM HONOR
  WHERE DEPT = '컴퓨터';

뷰의 제거

◘ 일반형식

- DROP VIEW 뷰_이름 { RESTRICT | CASCADE };

- 예
  DROP VIEW HONOR RESTRICT;
  DROP VIEW HONOR CASCADE; //COMHONOR도 제거

◘ Propagated Destroys

- 기본 테이블이 제거되면 그 위에 만들어진 인덱스나 뷰도 자동적으로 제거됨

뷰의 조작연산

- 기본 테이블에 사용 가능한 어떤 검색문도 뷰에 사용가능

- 변경(삽입, 삭제, 갱신) 연산의 제약

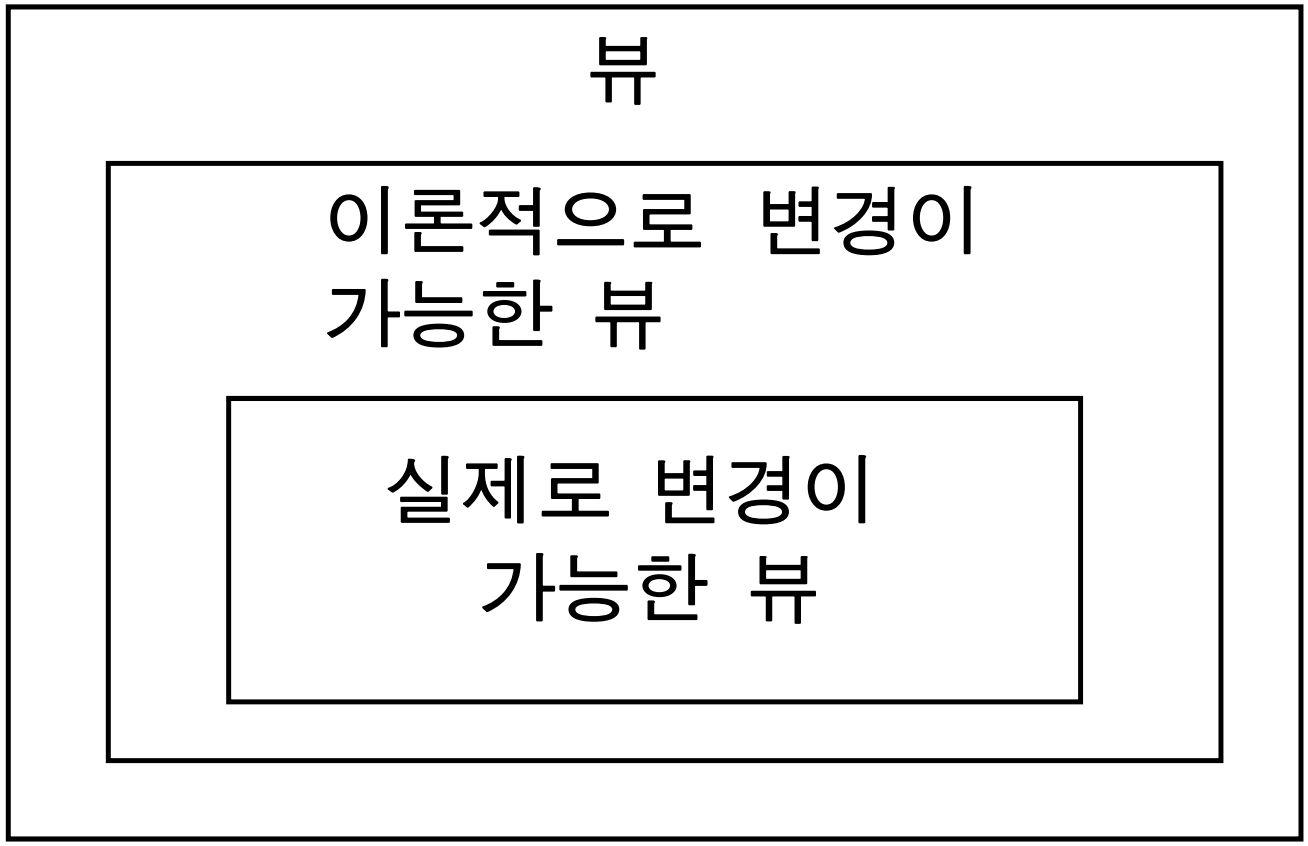
  - 열 부분집합 뷰(column subset view)
    - CREATE VIEW SVIEW1
      AS SELECT SNO, DEPT FROM STUDENT;
    - CREATE VIEW SVIEW2
      AS SELECT SNAME, DEPT FROM STUDENT;

  - 행 부분집합 뷰(row subset view)
    - CREATE VIEW SVIEW3
      AS SELECT *
      FROM STUDENT WHERE YEAR=4;

- 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
    AS SELECT CNO, AVG(MIDTERM)
    FROM ENROL
    GROUP BY CNO;

□ 뷰는 제한적인 갱신만 가능함



뷰의 장단점

◘ 뷰의 장점

- 논리적 독립성을 제공 (확장, 구조 변경)
- 데이타의 접근을 제어 (보안)
- 사용자의 데이타 관리를 간단히 함
- 여러 사용자에 다양한 뷰를 제공

◘ 뷰의 단점

- 독자적인 인덱스를 가질 수 없음
- 정의를 변경할 수 없음
- 삽입, 삭제, 수정 연산에 제한이 많음

10.4 삽입 SQL (ESQL, Embedded SQL)

- SQL은 이중 모드(dual mode)
  - 터미널을 통해 대화식으로 사용
  - 응용 프로그램에 삽입시켜 사용

- 커서(cursor)
  - SQL과 호스트언어 사이의 교량 역할
    - SQL: 레코드 집합 단위 처리
    - 호스트언어: 개별 레코드 단위 처리
  - 호스트 언어에서 SQL이 돌려주는
    레코드 집합을 처리하는데 사용하는 일종의 포인터

□ 응용 프로그램의 특징

- EXEC SQL을 앞에 붙임
- 삽입 SQL 실행문은 호스트 실행문이 나타나는 어느 곳에서나 사용 가능
- SQL문에 사용되는 호스트 변수는 콜론(:)을 앞에 붙임
- EXEC SQL DECLARE문으로 사용할 테이블을 선언
- 호스트 변수와 대응하는 필드의 데이타 타입이 일치
- 호스트변수 SQLSTATE
  - SQLSTATE='00000' : 성공적으로 실행됨.
  - SQLSTATE='02000' : 실행됐지만 검색된 데이터 없음.
  - 기타 : 경고 혹은 에러 표시

삽입 SQL문을 가진 PL/I 응용프로그램의 예

◆

```
EXEC SQL BEGIN DECLARE SECTION;
DCL SNO          FIXED BIN(15);
DCL SNAME        CHAR(20);
DCL DEPT         CHAR(6);
DCL SQLSTATE     CHAR(5);
EXEC SQL END DECLARE SECTION;

SNO = 100;

S1:   EXEC SQL SELECT      SNAME, DEPT
               INTO        :SNAME, :DEPT
               FROM        STUDENT
               WHERE       SNO = :SNO;

IF SQLSTATE = '00000'
THEN . . . . . .;
ELSE . . . . . .;
```

커서가 필요 없는 데이타 조작

◘ 단일 레코드 검색(Singleton SELECT)

- 검색된 테이블이 한 개 이하의 행만을 가지는 SELECT문

```
EXEC SQL SELECT     SNAME, DEPT
         INTO       :SNAME, :DEPT
         FROM       STUDENT
         WHERE      SNO = :SNO;
```

여기서 SELECT문은 최대 하나의 레코드만 검색됨.

◘ 수정

-

```
EXEC SQL UPDATE     ENROL
         SET        FINAL = FINAL + :NEW
         WHERE      CNO = 'C413';
```

◘ 삭제

- EXEC SQL DELETE
  FROM ENROL
  WHERE SNO = :SNO;

◘ 삽입

- EXEC SQL INSERT
  INTO STUDENT(SNO,SNAME,DEPT)
  VALUES (:SNO,:SNAME,:DEPT);

커서를 이용하는 데이타 조작

□ 커서(cursor)

- 여러 개의 레코드를 검색하는 SELECT 문
- 검색된 레코드 집합 속의 레코드를 하나하나 접근
- 활동 세트(active set)
  - ◆ 커서가 open 될 때마다 새로운 활동세트가 할당됨.

◘ 커서를 사용한 복수 레코드 검색 예

◆ EXEC SQL DECLARE C1 CURSOR FOR
SELECT SNO, SNAME, YEAR
FROM STUDENT
WHERE DEPT = :DEPT;

EXEC SQL OPEN C1;
DO
EXEC SQL FETCH C1 INTO
:SNO,:SNAME,:YEAR;
. . . . . .
END;

EXEC SQL CLOSE C1;

마지막 레코드를 처리한 후 다음 레코드를 읽을 때
SQLSTATE='02000'이 되어 루프가 종료됨.

□ 변경

- EXEC SQL UPDATE STUDENT
  SET YEAR = :VYEAR
  WHERE CURRENT OF C1;

□ 삭제

- EXEC SQL DELETE
  FROM STUDENT
  WHERE CURRENT OF C1;

Dynamic SQL (DSQL)

DSQL

- 대화식 온라인 응용을 구성할 수 있는 삽입 SQL

온라인 응용의 수행 과정

- 터미널에서 명령문을 접수
- 입력된 명령문 분석
- 데이타베이스에 적절한 SQL문으로 지시
- 터미널에 메시지/결과를 돌려보냄

필요성

- 입력 명령문이 상당히 다양하여 미리 그에 해당하는 SQL문을 만들어 두는 것이 불가능한 경우.
- 필요한 SQL문을 동적으로 만들어 바인딩함.

기본 명령어

- PREPARE
- EXECUTE

예

- DCL SQLSTMNT CHAR(256) VARING;
  SQLSTMNT = 'DELETE FROM ENROL
  WHERE CNO = “C413” AND FINAL < 60';
  EXEC SQL PREPARE SQLOBJ FROM :SQLSTMNT;
  EXEC SQL EXECUTE SQLOBJ;

  SQLOBJ는 SQLSTMNT에 주어진 SQL문의 목적코드를 저장하는데 사용하는 SQL 변수

- SQLSTMNT에는 호스트 변수 사용 불가
  - 매개변수(?)는 사용 가능
  - SQLSTMNT = 'DELETE FROM ENROL
    WHERE CNO = ? AND FINAL < ?';

```
EXEC SQL PREPARE SQLOBJ FROM :SQLSTMNT;
        . . . . . .
CNO='C413';
GRADE=60;
EXEC SQL EXECUTE SQLOBJ USING :CNO,:GRADE;
```