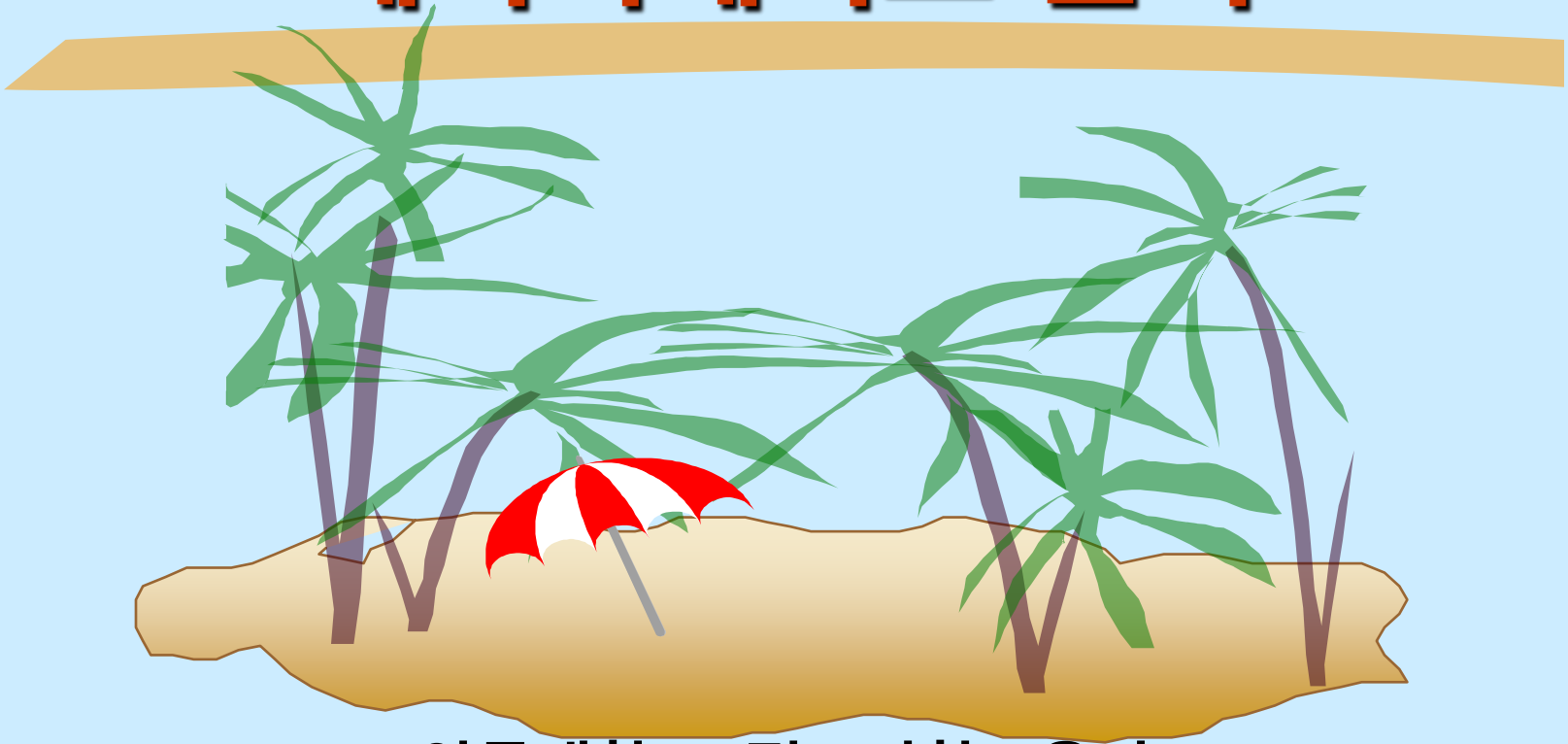
데이터베이스 언어



안동대학교 정보과학교육과

배울 내용

- 데이터 정의 언어
- 데이터 조작 언어
  - Database Modification
  - Database Manipulation
- SQL 뷰
- 삽입 SQL

SQL

- SEQUEL(Structured English Query Language)
  - 1974, IBM
  - SYSTEM R
  - 고급 데이터 언어
- SQL(Structured Query Language)
  - ANSI와 ISO의 데이타베이스 표준언어
  - SQL/92, SQL-92, SQL2
- 종합 DB언어
  - 데이터 정의, 데이터 조작, 데이터 제어
  - 대화식 질의어면서 데이터 부속어로 사용
  - 테이블, 행, 열로 표현

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 COMEDU AUTHORIZATION HHSONG;

※ 실제로 CREATE SCHEMA 보다 CREATE DATABASE 명령문을 씀

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

도메인 정의문(I)

- 일반 형식
  - CREATE DOMAIN 도메인_이름 데이터타입
    [묵시값_정의]
    [도메인_제약조건_정의리스트];

예) CREATE DOMAIN DEPT CHAR(4) DEFAULT '???'
CONSTRAINT VALID-DEPT
CHECK(VALUE IN('COMP', 'ME', 'EE', 'ARCH', '???'));

  - ALTER DOMAIN 도메인_이름 <변경내용>

  - DROP DOMAIN 도메인_이름 RESTRICT | CASCADE;

도메인 정의문(II)

- 데이터타입 - 시스템 데이터 타입만 사용
  - 숫자 타입
    - INTEGER, 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
    - DATE : YY-MM-DD
  - 시간
    - TIME : hh:mm:ss

기본 테이블의 생성

- 테이블 : 기본 테이블, 뷰(view), 임시 테이블
- 일반형식
  - CREATE TABLE 기본테이블(

```
{ 열이름 데이터타입 [ NOT NULL ] [ DEFAULT 묵시값 ] , }+
[ PRIMARY KEY (열이름_리스트), ]
{ [ UNIQUE (열이름_리스트) , ] }*
{ [ FOREIGN KEY(열이름_리스트)
        REFERENCES 기본테이블[( )]
        [ ON DELETE RESTRICT | CASCADE ]
        [ ON UPDATE RESTRICT | CASCADE ]
] } ,*
[ CONSTRAINT 이름 ] [ CHECK(조건식) ]
);
```

기본 테이블 생성 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(I)

- 기본 테이블의 제거
  - 일반형식

```
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(II)

- 기본 테이블의 변경
  - 일반형식

```
ALTER TABLE 기본_테이블_이름 (
    [ ADD 열_이름 데이터_타입 ] [ DEFAULT 묵시값 ] |
    [ DROP 열_이름 ] [ CASCADE ] |
    [ ALTER 열_이름 ( DROP DEFAULT | SET DEFAULT 묵시값 ) ]
);
```

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

배울 내용

- 데이터 정의 언어

- 데이터 조작 언어
  - Database Modification
  - Database Manipulation

- SQL 뷰

- 삽입 SQL

데이터 삽입(I)

- 일반 형식

```
INSERT
    INTO  테이블  [ ( 열_이름_리스트 ) ]
    VALUES  ( 열_값_리스트 );

INSERT
    INTO  테이블  [ ( 열_이름_리스트 ) ]
    SELECT문;
```

데이터 삽입(II)

- 레코드의 직접 삽입

```
INSERT
    INTO  STUDENT( SNO, SNAME, YEAR, DEPT )
    VALUES  ( 600, '이승엽', 4, '컴퓨터' );
```

- 부속 질의문을 이용한 레코드 삽입

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

데이터 갱신(I)

- 일반적인 형식

```
UPDATE  테이블
    SET  { 열_이름 = 산술식 }'+
    [ WHERE  조건 ];
```

- 하나의 레코드 변경

```
UPDATE  STUDENT
    SET  YEAR = 2
    WHERE  SN0 = 300;
```

데이터 갱신(II)

- 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'
       );
```

데이터 삭제(I)

- 일반 형식

```
DELETE
    FROM  테이블
    [ WHERE  조건 ];
```

- 하나의 레코드 삭제

```
DELETE
    FROM  STUDENT
    WHERE  SNO = 100;
```

  - 기본키와 참조무결성 문제

데이터 삭제(II)

- 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'
    );
```

데이터 검색 (I)

- 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;
```

데이터 검색 (II-1)

- 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;
```

데이터 검색 (II-2)

- 산술식과 문자 스트링이 명세된 검색

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

데이터 검색(III)

- 복수 테이블로부터의 검색(조인)

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

- 자기자신의 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(IV)

- 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(V)

- 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;
```

데이터 검색(VI)

- HAVING을 사용한 검색

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

- 부속질의어(Subquery)를 사용한 검색

```
SELECT  SNAME
    FROM  STUDENT
    WHERE  SNO  IN  (
            SELECT  SNO
                FROM  ENROL
                WHERE  CNO = 'C413'
        );
```

데이터 검색(VIII)

- LIKE를 사용하는 검색

```
SELECT  CNO, CNAME
   FROM  COURSE
   WHERE  CNO  LIKE  'C%';
```

- NULL을 사용한 검색

```
SELECT  SNO, SNAME
   FROM  STUDENT
   WHERE  DEPT  IS  NULL;
```

Note : NULL 값은 비교연산자와 같이 사용될 수 없음

데이터 검색(VIII)

- EXISTS를 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'
   );
```

  - EXISTS 이하 SELECT문이 참(공집합이 아님)일 때 본 SELECT문을 실행

데이터 검색(IX)

- UNION이 관련된 검색

```
SELECT  SNO
   FROM  STUDENT
   WHERE  YEAR = 1
   UNION
      SELECT  SNO
         FROM  ENROL
         WHERE  CNO = 'C324';
```

  - 중복되는 투플은 제거

배울 내용

- 데이터 정의 언어
- 데이터 조작 언어
  - Database Modification
  - Database Manipulation

- SQL 뷰
- 삽입 SQL

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
- 뷰의 정의는 시스템 카탈로그(SYSVIEWS)에 SELECT-FROM-WHERE의 형태로 저장됨

뷰의 생성(I)

- 일반형식

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

  - 갱신이나 삽입 연산 시 조건 확인

- 예

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

뷰의 생성(II)

- 예 (cont')

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

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;

CREATE  VIEW  COMHONOR
    AS  SELECT  SNAME
            FROM  HONOR
            WHERE  DEPT = '컴퓨터';
```

뷰의 제거

- 일반형식

  DROP VIEW 뷰_이름 { RESTRICT | CASCADE };

- 예

  DROP VIEW DEPTSIZE RESTRICT;

- "Propagated Destroys"
  - 기본 테이블이 제거되면 그 위에 만들어진 인덱스나 뷰도 자동적으로 제거됨

뷰의 조작 연산(I)

- 기본 테이블에 사용 가능한 어떤 검색(SELECT)문도 뷰에 사용가능

- 변경(삽입, 삭제, 갱신) 연산은 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 SNO, SNAME, YEAR, DEPT
        FROM STUDENT WHERE YEAR=4;
```

뷰의 조작 연산(II)

- 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;
```

배울 내용

- 데이터 정의 언어
- 데이터 조작 언어
  - Database Modification
  - Database Manipulation
- SQL 뷰

- 삽입 SQL

삽입 SQL

- 이중 모드(dual mode) 원리
  - 터미널에서 대화식으로 사용할 수 있는 모든 SQL 문

    → 응용 프로그램(Interactive and Embedded form)에서 사용 가능

삽입 SQL 포함하는 응용 프로그램의 특징

- 명령문 앞에 EXEC SQL을 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

응용 프로그램의 특징(cont')

- 호스트변수 SQLSTATE를 포함
  - 피드백 정보
  - SQLSTATE = "00000" : 성공
    ≠ "00000" : 경고 (warning) 또는 에러
- 호스트 변수와 대응하는 필드의 데이터 타입이 일치
- 호스트 변수와 데이터베이스 필드의 이름은 같아도 됨
- SQLSTATE 변수에 반환된 값 검사

응용 프로그램에서의 삽입 SQL

```
EXEC SQL BEGIN DECLARE SECTION;
    int sno;
    char sname[21];
    char dept[7];
    char SQLSTATE[6];
  EXEC SQL END DECLARE SECTION;
  sno = 100;

EXEC SQL SELECT  Sname, Dept
          INTO : sname, : dept
                FROM    STUDENT
                WHERE  Sno = : sno;
       IF SQLSTATE = '00000'
          THEN ... ;
          ELSE  ... ;
```

커서가 필요 없는 데이터 연산(I)

- 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;
```

- 갱신

```
EXEC    SQL    UPDATE    ENROL
        SET    Final = Final + : new
        WHERE    Cno = 'C413';
```

커서가 필요 없는 데이터 연산(II)

- 삭제

```
EXEC    SQL    DELETE
  FROM    ENROL
  WHERE    Sno = :sno;
```

- 삽입

```
EXEC    SQL    INSERT
  INTO    STUDENT(Sno, Sname, Dept)
  VALUES    (:sno, :sname, :dept);
```

커서를 이용하는 데이터 조작(I)

- 커서(cursor)
  - SELECT 문과 호스트 프로그램 사이를 연결
  - SELECT 문으로 검색되는 여러 개의 레코드(투플)에 대해 정의
  - 활동 세트(active set) : SELECT 문으로 검색된 여러 개의 레코드
  - 실행 시에는 활동 세트에 있는 레코드 하나를 지시

커서를 이용하는 데이터 조작(II)

❑ 복수 레코드의 검색 예

```
EXEC   SQL  DECLARE   C1  CURSOR  FOR        /*커서 C1의 정의*/
                SELECT     Sno, Sname, Year
                FROM      STUDENT
                WHERE      Dept = :dept;
EXEC  SQL  OPEN    C1;                /*질의문의 실행*/
        DO /* C1으로 접근되는 모든  STUDENT 레코드에 대해 */
              EXEC SQL FETCH C1  INTO :sno, :sname, :year;
                                          /*다음 학생 레코드의 채취*/
                . . . . . .
        END;
EXEC SQL CLOSE   C1;   /*커서 c1의 활동 종료 */
```

커서를 이용하는 데이터 조작 (III)

- 변경

```
EXEC SQL UPDATE STUDENT
    SET Year = : year
    WHERE CURRENT OF C1;
```

  - CURRENT OF : 커서가 가리키고 있는 특정 레코드

- 삭제

```
EXEC SQL DELETE
    FROM STUDENT
    WHERE CURRENT OF C1;
```

요약

- 데이터 정의 언어
- 데이터 조작 언어
  - Database Modification
  - Database Manipulation
- SQL 뷰
- 삽입 SQL

다음 배울 내용 : 질의어처리 및 최적화