Char8. SQL응용

DDL Data Define Language 데이터 정의어

CREATE, ALTER, DROP

```
CREATE SCHEMA 스키마명 AUTHORIZATION 사용자_id;
```

```
CREATE TABLE 테이블명
    (속성명 데이터_타입 [DEFAULT 기본값] [NOT NULL], …
    [, PRIMARY KEY(기본키_속성명, …)]
    [, UNIQUE(대체키_속성명, …)]
    [, FOREIGN KEY(외래키_속성명, …)]
        [REFERENCES 참조테이블(기본키_속성명, …)]
        [ON DELETE 옵션]
        [ON UPDATE 옵션]
    [, CONSTRAINT 제약조건명] [CHECK (조건식)]);
```

```
다른 테이블을 이용한 테이블 정의
CREATE TABLE 신규테이블명 AS SELECT 속성명[, 속성명,...] FROM 기존테이블명;
```

```
CREATE DOMAIN 도메인명 [AS] 데이터_타입
        [DEFAULT 기본값]
        [CONSTRAINT 제약조건명 CHECK (범위값)];
```

```
CREATE VIEW 뷰명[(속성명[, 속성명, …])]
AS SELECT문;
```

```
CREATE [UNIQUE] INDEX 인덱스명
ON 테이블명(속성명 [ASC|DESC] [,속성명 [ASC|DESC]])
[CLUSTER];
```

```
ALTER TABLE 테이블명 ADD 속성명 데이터_타입 [DEFAULT '기본값'];
ALTER TABLE 테이블명 ALTER | MODIFY 속성명 [SET DEFAULT '기본값'];
ALTER TABLE 테이블명 DROP COLUMN 속성명 [CASCADE];
```

```
DROP TABLE 테이블명 [CASCADE | RESTRICT];
```

DCL Data Control Language 데이터 제어어

GRANT, REVOKE, COMMIT, ROLLBACK, SAVEPOINT

- 데이터베이스 관리자가 데이터베이스 사용자에게 권한을 부여하거나 취소하기 위한 명령어이다.
- GRANT : 권한 부여를 위한 명령어
- REVOKE : 권한 취소를 위한 명령어
- **사용자등급 지정 및 해제**

```
– GRANT 사용자등급 TO 사용자_ID_리스트 [IDENTIFIED BY 암호];
– REVOKE 사용자등급 FROM 사용자_ID_리스트;
```

- **테이블 및 속성에 대한 권한 부여 및 취소**

```
– GRANT 권한_리스트 ON 개체 TO 사용자 [WITH GRANT OPTION];
– REVOKE [GRANT OPTION FOR] 권한_리스트 ON 개체 FROM 사용자 [CASCADE];
```

  - 권한 종류 : ALL, SELECT, INSERT, DELETE, UPDATE, ALTER 등
  - WITH GRANT OPTION : 부여받은 권한을 다른 사용자에게 다시 부여할 수 있는 권한을 부여함
  - GRANT OPTION FOR : 다른 사용자에게 권한을 부여할 수 있는 권한을 취소함
  - CASCADE : 권한 취소 시 권한을 부여받았던 사용자가 다른 사용자에게 부여한 권한도 연쇄적으로 취소함

COMMIT	트랜잭션이 성공적으로 끝나면 데이터베이스가 새로운 일관성(Consistency) 상태를 가지기 위해 변경된 모든 내용을 데이터베이스에 반영하여야 하는데, 이때 사용하는 명령어
ROLLBACK	아직 COMMIT되지 않은 변경된 모든 내용들을 취소하고 데이터베이스를 이전 상태로 되돌리는 명령어
SAVEPOINT	트랜잭션 내에 ROLLBACK 할 위치인 저장점을 지정하는 명령어로, 저장점을 지정할 때는 이름을 부여하며, ROLLBACK 시 지정된 저장점까지의 트랜잭션 처리 내용이 취소됨

DML Data Manipulation Language 데이터 조작어

SELECT	테이블에서 튜플을 검색한다.	
INSERT	테이블에 새로운 튜플을 삽입	**INSERT INTO 테이블명([속성명1, 속성명2, ...]) VALUES (데이터1, 데이터2, ...);**
DELETE	테이블에서 튜플을 삭제	**DELETE FROM 테이블명 [WHERE 조건];**
UPDATE	테이블에서 튜플의 내용을 갱신	**UPDATE 테이블명 SET 속성명=데이터[,속성명=데이터] [WHERE 조건];**

SELECT

```
SELECT [PREDICATE] [테이블명.]속성명 [AS 별칭][, [테이블명.]속성명, ...]
[, 그룹함수(속성명) [AS 별칭]]
[, Window 함수 OVER (PARTITION BY 속성명 1, 속성명 2, ...
                     ORDER BY 속성명 3, 속성명 4, ...)]
FROM 테이블명[, 테이블명, ...]
[WHERE 조건]
[GROUP BY 속성명, 속성명, ...]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];
```

잠깐만요 SELEC문의 실행 작동 순서

FROM → WHERE → GROUP BY → HAVING → SELECT → DISTINCT → ORDER BY

INNER JOIN

가장 일반적인 JOIN의 형태로, 관계가 설정된 두 테이블에서 조인된 필드가 일치하는 행만을 표시한다.

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, ···
FROM 테이블명1, 테이블명2, ···
WHERE 테이블명1.속성명 = 테이블명2.속성명;
```

〈학생〉

학번	이름	학과코드	선배	성적
15	고길동	com		83
16	이순신	han		96
17	김선달	com	15	95
19	아무개	han	16	75
37	박치민		17	55

〈학과〉

학과코드	학과명
com	컴퓨터
han	국어
eng	영어

예제 〈학생〉 테이블과 〈학과〉 테이블에서 학과코드 값이 같은 튜플을 JOIN하여 학번, 이름, 학과코드, 학과명을 출력하는 SQL문을 작성하시오.

```
SELECT 학번, 이름, 학생.학과코드, 학과명
FROM 학생, 학과
WHERE 학생.학과코드 = 학과.학과코드;
```

〈결과〉

학번	이름	학과코드	학과명
15	고길동	com	컴퓨터
16	이순신	han	국어
17	김선달	com	컴퓨터
19	아무개	han	국어

OUTER JOIN

- 릴레이션에서 JOIN 조건에 만족하지 않는 튜플도 결과로 출력하기 위한 JOIN 방법으로, LEFT OUTER JOIN, RIGHT OUTER JOIN 등이 있다.
- LEFT OUTER JOIN : INNER JOIN의 결과를 구한 후, 우측 항 릴레이션의 어떤 튜플과도 맞지 않는 좌측 항의 릴레이션에 있는 튜플들에 NULL 값을 붙여서 INNER JOIN의 결과에 추가함

  – 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, ···
FROM 테이블명1 LEFT OUTER JOIN 테이블명2
ON 테이블명1.속성명 = 테이블명2.속성명;
```

- RIGHT OUTER JOIN : INNER JOIN의 결과를 구한 후, 좌측 항 릴레이션의 어떤 튜플과도 맞지 않는 우측 항의 릴레이션에 있는 튜플들에 NULL 값을 붙여서 INNER JOIN의 결과에 추가함

  – 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, ···
FROM 테이블명1 RIGHT OUTER JOIN 테이블명2
ON 테이블명1.속성명 = 테이블명2.속성명;
```

〈학생〉

학번	이름	학과코드	선배	성적
15	고길동	com		83
16	이순신	han		96
17	김선달	com	15	95
19	아무개	han	16	75
37	박치민		17	55

〈학과〉

학과코드	학과명
com	컴퓨터
han	국어
eng	영어

예제 〈학생〉 테이블과 〈학과〉 테이블에서 학과코드 값이 같은 튜플을 JOIN하여 학번, 이름, 학과코드, 학과명을 출력하는 SQL문을 작성하시오. 이때, 학과코드가 입력되지 않은 학생도 출력하시오.

```
SELECT 학번, 이름, 학생.학과코드, 학과명
FROM 학생 LEFT OUTER JOIN 학과
ON 학생.학과코드 = 학과.학과코드;
```

해설 INNER JOIN을 하면 학과코드가 입력되지 않은 박치민은 출력되지 않는다. 그러므로 JOIN 구문을 기준으로 왼쪽 테이블, 즉 〈학생〉의 자료는 모두 출력되는 LEFT JOIN을 사용한 것이다. 다음과 같이 JOIN 구문을 기준으로 테이블의 위치를 교환하여 RIGHT JOIN을 사용해도 결과는 같다.

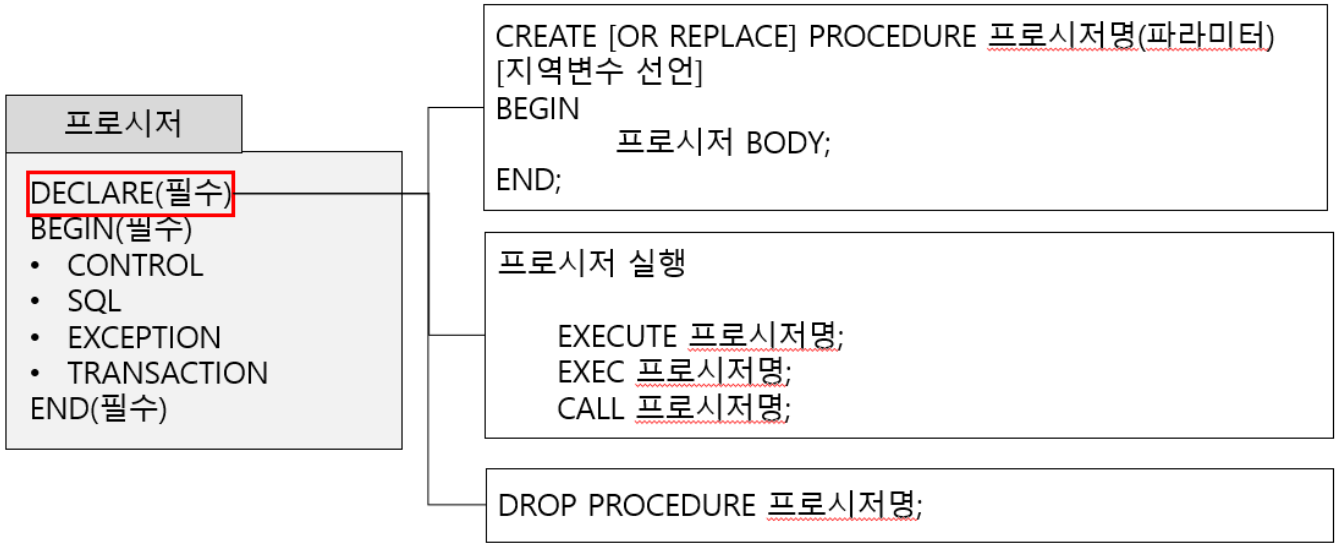
〈결과〉

학번	이름	학과코드	학과명
15	고길동	com	컴퓨터
16	이순신	han	국어
17	김선달	com	컴퓨터
19	아무개	han	국어
37	박치민		

프로시저 Procedure

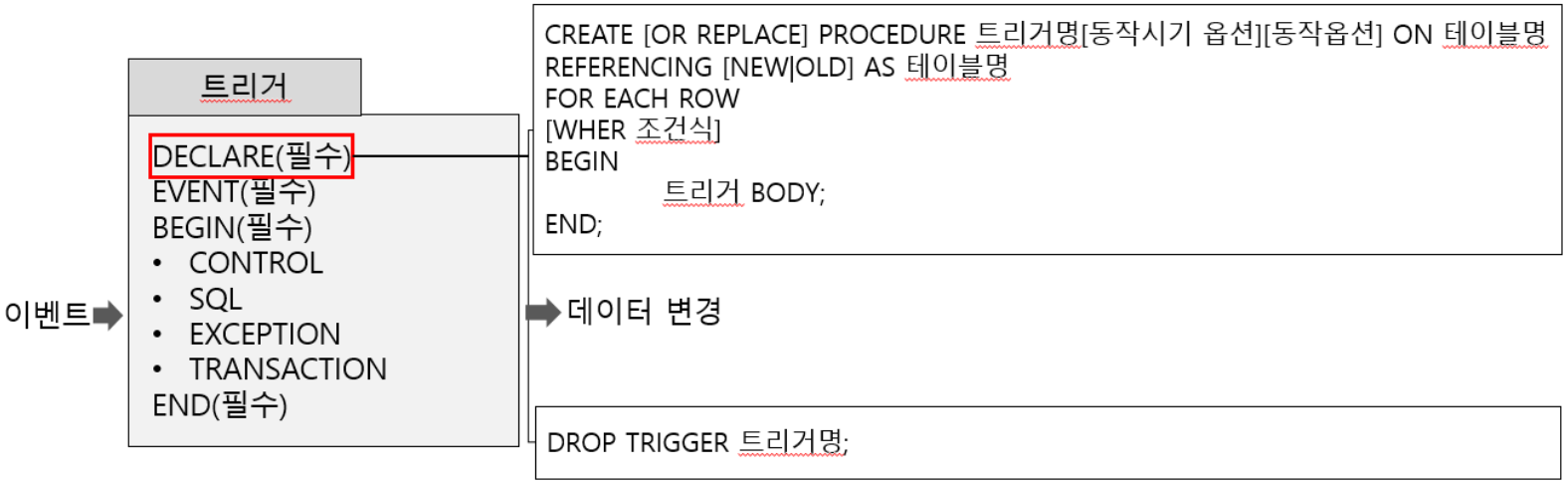
절차형 SQL을 활용하여 특정 기능을 수행하는 일종의 트랜잭션 언어로 호출을 통해 실행되어 미리 저장해 놓은 SQL 작업을 수행한다.

프로시저는 데이터베이스에 저장되어 수행되기 때문에 스토어드 프로시저 Stored Procedure라고 불린다.



트리거 Trigger

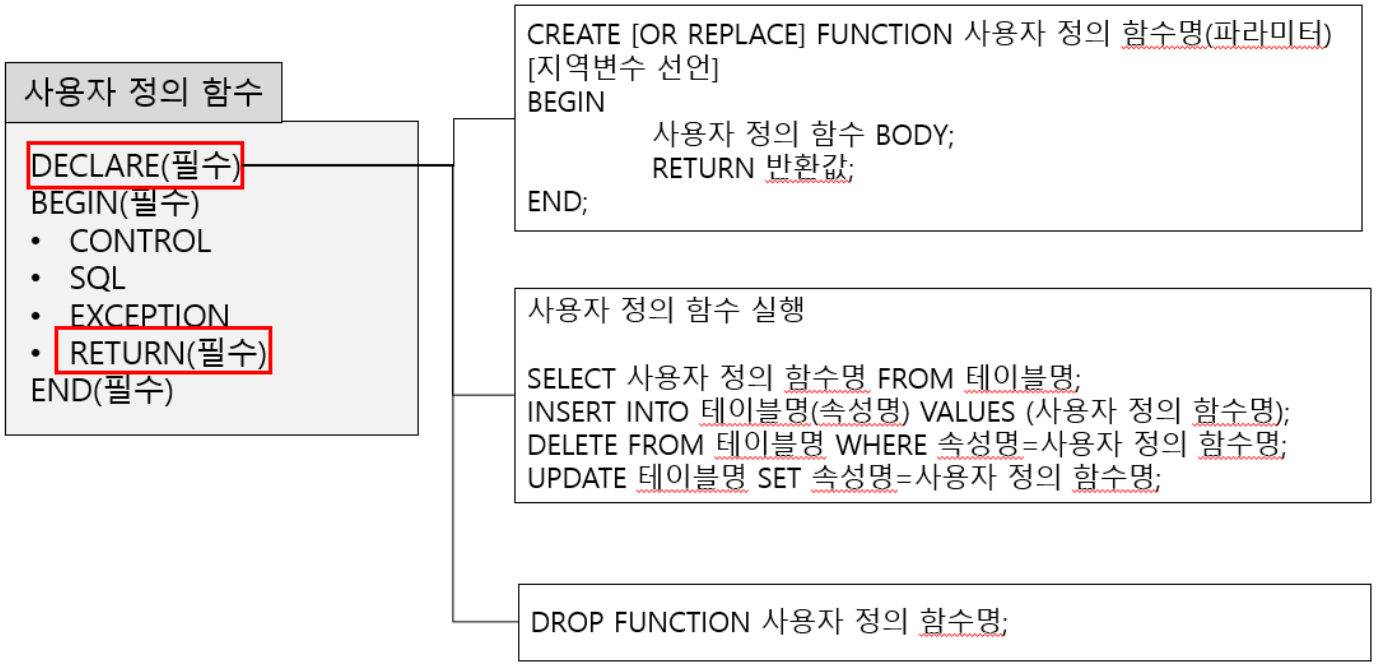
선언, 이벤트, 시작, 종료로 구성되며, 시작과 종료 구문 사이에는 제어CONTROL, SQL, 예외EXCEPTION가 포함



구분	**프로시저**	**사용자 정의 함수**
반환값	없거나 1 개 이상 가능	1 개
파라미터	입출력 가능	입력만 가능
사용 가능 명령문	DML, DCL	SELECT
호출	프로시저, 사용자 정의 함수	사용자 정의 함수
사용 방법	실행문	DML 에 포함

사용자 정의 함수

- 프로시저와 유사하게 SQL을 사용하여 일련의 작업을 연속적을 처리, 종료 시 처리결과를 단일값으로 반환하는 절차형 SQL
- 예약어 RETURN을 통해 값을 반환하기 때문에 출력 파라미터가 없다.
- 사용자 정의 함수는 INSERT, DELETE,, UPDATE를 통한 테이블 조작은 할 수 없고 SELECT를 통한 조회만 가능
- 내장함수처럼 DML문에서 반환 값을 활용하기 위한 용도로 사용한다.



커서 CURSOR

쿼리문의 처리 결과가 저장되어 있는 메모리 공간을 가리키는 포인터

3단계로 구성: 열기 Open, 패치 Fetch, 닫기 Close

묵시적 커서 Implicit Cursor

DBMS 자체적으로 열리고 패치되어 사용이 끝나면 닫히지만 커서의 속성을 조회하여 사용된 커서 정보를 열람하는 것이 가능하다.

내부에서 자동으로 생성되어 사용

수행된 쿼리문의 정상적인 수행 여부를 확인하기 위해 사용

SQL%FOUND	쿼리 수행의 결과로 패치된 튜플 수가 1개이상이면 TRUE
SQL%NOTFOUND	쿼리 수행의 결과로 패치되 튜플 수가 0개면 TRUE
SQL%ROWFOUND	쿼리 수행의 결과로 패치된 튜플 수 반환
SQL%ISOPEN	커서가 열린 상태 OPEN이면 TRUE 묵시적 커서는 자동으로 생성된 후 자동을 닫히기 때문에 항상 FALSE

명시적 커서 Explicit Cursor

사용자가 직접 정의해서 사용하는 커서

주로 절차형 SQL에 SELECT문의 결과로 반환되는 여러 튜플들을 제어하기 위해 사용

기본적으로 열기 Open - 패치 Fetch - 닫기 Close 순으로 이루어 진다.

사용자가 직접 정의하여 사용

직접 구현 필요

쿼리문의 결과를 저장하여 사용함으로써 동일한 쿼리가 반복 수행되어 데이터베이스 자원 낭비를 방지

선언Declare 형식	CURSOR 커서명(매개변수1, 매개변수 2, ...) IS SELECT문;
열기 Open 형식	OPEN 커서명(매개변수1, 매개변수2, ...);
패치 Fetch 형식	FETCH INTO 변수1, 변수2, ... ;
닫기 Close 형식	CLOSE 커서명;