ORACLE®

DB Management

DML 활용

DML

- DML
  - ✓ Data Manipulation Language
  - ✓ 데이터 조작어
  - ✓ 정의된 데이터베이스에 데이터를 삽입하거나 수정하거나 삭제하는 역할을 수행하는 SQL문
  - ✓ 테이블(Table)이나 뷰(View) 등의 데이터베이스 객체에 행(Row)을 삽입/수정/삭제하는 기능을 담당
  - ✓ 실행 후에 작업의 완료 또는 취소 처리 가능

- DML 종류
  - ① INSERT : 행(Row) 삽입
  - ② UPDATE : 행(Row) 수정
  - ③ DELETE : 행(Row) 삭제

INSERT

- 구문

```
INSERT INTO 테이블_이름
    [(칼럼1, 칼럼2, ...)]
VALUES
    (값1, 값2, ...)
```

칼럼 리스트와 값의 개수가 동일해야 함
칼럼 리스트를 생략하면 전체 칼럼에 값을 입력하는 것임

- NATION 테이블의 CODE, NAME, CONTINENT 칼럼에 새로운 데이터 삽입하기

```
INSERT INTO NATION(CODE, NAME, CONTINENT) VALUES(1, 'KOREA', 'ASIA');
```

실행 전

CODE	NAME	CONTINENT

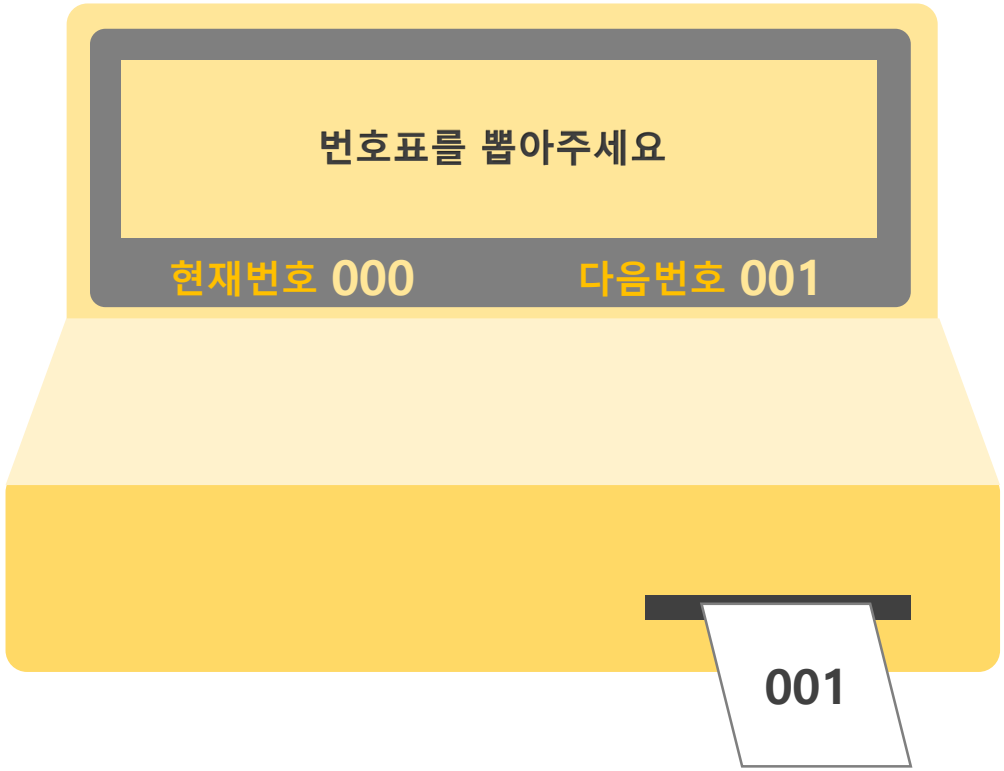
실행 후

CODE	NAME	CONTINENT
1	KOREA	ASIA

시퀀스

- 시퀀스
  - ✓ Sequence
  - ✓ 순차적으로 증가하는 일련번호를 생성하는 데이터베이스 객체
  - ✓ 인공키(Artificial Key)를 기본키로 사용하는 경우 시퀀스를 이용해서 번호를 생성한 뒤 INSERT문으로 삽입함
  - ✓ CREATE문을 이용해 시퀀스 생성 가능

시퀀스는
순서대로 번호를 생성해주는
번호표와 같은 개념



ORACLE

시퀀스 생성

- 시퀀스 생성방법

```
CREATE SEQUENCE 시퀀스_이름
[INCREMENT BY n]
[START WITH n]
[MAXVALUE n | NOMAXVALUE]
[MINVALUE n | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE n | NOCACHE]
[ORDER | NOORDER]
```

- 시퀀스 옵션
  - ① INCREMENT BY n ➢ 증감값 n (기본값 1)
  - ② START WITH n ➢ 시작값 n (기본값 1, 증가인 경우 MINVALUE, 감소인 경우 MAXVALUE와 같은 의미를 가짐)
  - ③ MAXVALUE n ➢ 최대값 n (NOMAXVALUE : 최대값 없음)
  - ④ MINVALUE n ➢ 최소값 n (NOMINVALUE : 최소값 없음)
  - ⑤ CYCLE ➢ 번호 순환 유무 (MAXVALUE 도달 시 MINVALUE부터 다시 시작, NOCYCLE : 순환 없음)
  - ⑥ CACHE n ➢ 메모리 캐시 사용 여부 (CACHE 설정 시 시퀀스 번호가 안 맞을 수 있으므로 NOCACHE 권장)
  - ⑦ ORDER ➢ 순차적 번호 사용 여부 (ORDER : 순차적으로 번호 모두 사용, NOORDER : 번호 건너뛰기 가능)

시퀀스 함수

- CURRVAL
  - ✓ 시퀀스로 생성한 현재 번호를 확인
  - ✓ NEXTVAL 함수 호출이 한 번도 없는 경우 사용할 수 없음
  - ✓ 사용방법 : 시퀀스_이름.CURRVAL

- NEXTVAL
  - ✓ 시퀀스로 새로운 번호를 생성
  - ✓ 최초 시퀀스 사용 시 반드시 NEXTVAL 함수를 먼저 사용해야 함
  - ✓ 사용방법 : 시퀀스_이름.NEXTVAL

- CURRVAL, NEXTVAL
  - INSERT문이나 UPDATE문에서 사용
  - 서브쿼리, GROUP BY, HAVING, ORDER BY, DISTINCT 와 함께 사용할 수 없음

INSERT와 시퀀스

- NATION 테이블의 CODE, NAME, CONTINENT 칼럼에 새로운 데이터 삽입하기

```
CREATE SEQUENCE NATION_SEQ
INCREMENT BY 1
START WITH   1000
NOCACHE;

INSERT INTO NATION(CODE, NAME, CONTINENT) VALUES(NATION_SEQ.NEXTVAL, 'KOREA', 'ASIA');
INSERT INTO NATION(CODE, NAME, CONTINENT) VALUES(NATION_SEQ.NEXTVAL, 'FRANCE', 'EUROPE');
INSERT INTO NATION(CODE, NAME, CONTINENT) VALUES(NATION_SEQ.NEXTVAL, 'JAPAN', 'ASIA');
```

실행 전

CODE	NAME	CONTINENT

실행 후

CODE	NAME	CONTINENT
1000	KOREA	ASIA
1001	FRANCE	EUROPE
1002	JAPAN	ASIA

UPDATE

- 구문

```
UPDATE 테이블_이름
SET 칼럼1 = 값1, 칼럼2 = 값1
[WHERE 조건식]
```

WHERE절을 생략하면 전체 행을 수정하는 것임

- NATION 테이블에서 CODE가 1인 국가의 NAME을 'JAPAN'으로 수정하기

```
UPDATE NATION SET NAME = 'JAPAN' WHERE CODE = 1;
```

실행 전

CODE	NAME	CONTINENT
1	KOREA	ASIA

실행 후

CODE	NAME	CONTINENT
1	JAPAN	ASIA

DELETE

- 구문

```
DELETE
FROM 테이블_이름
[WHERE 조건식]
```

WHERE절을 생략하면 전체 행을 삭제하는 것임

- NATION 테이블에서 CODE가 1인 국가 삭제하기

```
DELETE FROM NATION WHERE CODE = 1;
```

실행 전

CODE	NAME	CONTINENT
1	JAPAN	ASIA

실행 후

CODE	NAME	CONTINENT

트랜잭션(Transaction)

- 트랜잭션(Transaction)
  - ✓ 데이터베이스에서 처리되는 여러 SQL 명령들을 하나의 논리적 작업 단위로 처리하는 것
  - ✓ 작업이 시작되면 중간에 멈추지 않고, 반드시 종료해야 하는 작업 단위
    (은행 이체, 프로그램 설치 등)
  - ✓ 작업 도중 하나라도 실패하면 아무 일도 하지 않은 상태로 되돌아감

- TCL(Transaction Control Language)
  - ① **COMMIT**
    트랜잭션내의 모든 SQL 실행으로 인해 변경된 작업 내용을 디스크에 영구적으로 저장하고 트랜잭션을 종료
  - ② **ROLLBACK**
    트랜잭션내의 모든 SQL 실행으로 인해 변경된 작업 내용을 모두 취소하고 트랜잭션을 종료

- 트랜잭션이 필요한 SQL
  - ✓ COMMIT 필요 → INSERT, UPDATE, DELETE
  - ✓ COMMIT 불필요 → CREATE, ALTER, DROP, TRUNCATE, SELECT

예제 데이터베이스 생성

- 부서(DEPARTMENT) 테이블

칼럼명	데이터타입	설명
DEPT_NO	NUMBER	부서번호, PK
DEPT_NAME	VARCHAR2(15 BYTE)	부서이름, NOT NULL
LOCATION	VARCHAR2(15 BYTE)	부서위치, NOT NULL

- 사원(EMPLOYEE) 테이블

칼럼명	데이터타입	설명
EMP_NO	NUMBER	사원번호, PK
NAME	VARCHAR2(20 BYTE)	이름, NOT NULL
DEPART	NUMBER	소속부서번호, FK(부서테이블의 부서번호 칼럼을 참조)
POSITION	VARCHAR2(20 BYTE)	직급
GENDER	CHAR(2 BYTE)	성별
HIRE_DATE	DATE	고용일
SALARY	NUMBER	급여

예제 데이터베이스 생성

- 부서(DEPARTMENT) 테이블

DEPT_NO	DEPT_NAME	LOCATION
1	영업부	대구
2	인사부	서울
3	총무부	대구
4	기획부	서울

- 사원(EMPLOYEE) 테이블

EMP_NO	NAME	DEPART	POSITION	GENDER	HIRE_DAE	SALARY
1001	구창민	1	과장	M	95-05-01	5000000
1002	김민서	1	사원	M	17-09-01	2500000
1003	이은영	2	부장	F	90-09-01	5500000
1004	한성일	2	과장	M	93-04-01	5000000

예제 데이터베이스 생성

- ANSI 문법 - 행(ROW) 하나씩 INSERT

```
INSERT INTO DEPARTMENT VALUES(1, '영업부', '대구');
INSERT INTO DEPARTMENT VALUES(2, '인사부', '서울');
INSERT INTO DEPARTMENT VALUES(3, '총무부', '대구');
INSERT INTO DEPARTMENT VALUES(4, '기획부', '서울');
```

- 오라클 문법 - 여러 행(ROWS) 한 번에 INSERT

```
INSERT ALL
    INTO DEPARTMENT VALUES(1, '영업부', '대구');
    INTO DEPARTMENT VALUES(2, '인사부', '서울');
    INTO DEPARTMENT VALUES(3, '총무부', '대구');
    INTO DEPARTMENT VALUES(4, '기획부', '서울')
SELECT * FROM DUAL;
```

예제 데이터베이스 생성

- ANSI 문법 - 행(ROW) 하나씩 INSERT

```
INSERT INTO EMPLOYEE VALUES(1001, '구창민', 1, '과장', 'M', '95/05/01', 5000000);
INSERT INTO EMPLOYEE VALUES(1002, '김민서', 1, '사원', 'M', '17/09/01', 2500000);
INSERT INTO EMPLOYEE VALUES(1003, '이은영', 2, '부장', 'F', '90/09/01', 5500000);
INSERT INTO EMPLOYEE VALUES(1004, '한성일', 2, '과장', 'M', '93/04/01', 5000000);
```

- 오라클 문법 - 여러 행(ROWS) 한 번에 INSERT

```
INSERT ALL
    INTO EMPLOYEE VALUES(1001, '구창민', 1, '과장', 'M', '95/05/01', 5000000)
    INTO EMPLOYEE VALUES(1002, '김민서', 1, '사원', 'M', '17/09/01', 2500000)
    INTO EMPLOYEE VALUES(1003, '이은영', 2, '부장', 'F', '90/09/01', 5500000)
    INTO EMPLOYEE VALUES(1004, '한성일', 2, '과장', 'M', '93/04/01', 5000000)
SELECT * FROM DUAL;
```

수강신청 데이터베이스