0405

SEQUENC (순서 , 연속)

- 순차적으로 번호를 발생시켜주는 객체이다

사용 목적

- 주로 PRIMARY KEY 제약조건에 활용하는 값으로 주로 사용된다.
  - 1 , 2 , 3 , 4 .....

[작성법]
CREATE SEQUENCE 시퀀스이름
[STRAT WITH 숫자] -- 처음 발생시킬 시작값 지정, 생략하면 자동 1이 기본
[INCREMENT BY 숫자] -- 다음 값에 대한 증가치, 생략하면 자동 1이 기본
[MAXVALUE 숫자 | NOMAXVALUE] -- 발생시킬 최대값 지정 (10의 27승, -1)
[MINVALUE 숫자 | NOMINVALUE] -- 최소값 지정 (-10의 26승)
[CYCLE | NOCYCLE] -- 값 순환 여부 지정
[CACHE 바이트크기 | NOCACHE] -- 캐쉬메모리 기본값은 20바이트, 최소값은 2바이트

SEQUENCE 사용방법

1) 시퀀스명 . NEDXTVAL : 다음 시퀀스 번호를 얻어옴(INCREMENT BY만큼 증가된 값)
단 시퀀스 생성 후 첫 호출인 경우 START WHIT의 값을 얻어온다

2) 시퀀스명 . CURRVAL : 현재 시퀀스 번호를 얻어옴
단 시퀀스 생성 후 NEXTVAL 호출 없이CURRVAL을 호출하면 오류 발생 .

```
CREATE SEQUENCE SEQ_EMP_ID
START WITH 223
INCREMENT BY 5;
```

- SEQ_EMP_ID 시퀀스를 만든다
- 시작 값은 223 , 5씩 증가한다

```
SELECT SEQ_EMP_ID.NEXTVAL FROM DUAL;  --최초 호출 START WITH값이 조회
--223
```

- 최초 호출시 시작값부터 호출된다

```
--CURRVAL 호출
SELECT SEQ_EMP_ID.CURRVAL FROM DUAL;
```

- 현재 SEQUENCE의 수를 보여준다

- ROLLBACK을 해도 증가된 시퀀스의 번호는 돌아가지 않는다

SEQUENCE 변경 _ALTER

```
-- [ 작성법 ]  CREATE = > ALTER  /  CREATE 삭제
--   ALTER SEQUENCE 시퀀스이름
--  [INCREMENT BY 숫자] -- 다음 값에 대한 증가치, 생략하면 자동 1이 기본
--  [MAXVALUE 숫자 | NOMAXVALUE] -- 발생시킬 최대값 지정 (10의 27승, -1)
--  [MINVALUE 숫자 | NOMINVALUE] -- 최소값 지정 (-10의 26승)
--  [CYCLE | NOCYCLE] -- 값 순환 여부 지정
--  [CACHE 바이트크기 | NOCACHE] -- 캐쉬메모리 기본값은 20바이트, 최소값은 2바이트
```

- CREATE 와 작성법이 완전 동일한대 CREATE가 아니라 ALTER으로 적는다
- 만약 시퀀스를 잘못 다뤄 시퀀스번호에 문제가 생길 경우 삭제 후 재 생성하는 게 유일한 길이다

INDEX (색인, 목차)

- SELECT의 처리속도를 향상시키기 위해 컬럼에 대하여 생성하는 객체
- 인덱스의 내부 구조는 B* 트리 형식으로 되어있다 (2진트리)

장점

- 이진트리 형식이라 자동 정렬및 검색 속도가 빠르다
- 조회시 전체 테이블이 아닌 인덱스가 있는 컬럼만을 조회하기 때문에 시스템 부하가 낮아져 성능이 향상된다

단점

- 데이터 변경 작업이 빈번한 경우 오히려 성능이 저하되는 문제가 발생한다
- 인덱스도 하나의 객체이다 보니 저장하기 위한 별도으 | 공간

```
[작성법]
   CREATE [UNIQUE] INDEX 인덱스명
   ON 테이블명 (컬럼명, 컬럼명, ... | 함수명, 함수계산식);
```

- 인덱스가 자동으로 생성되는 경우 == PK 또는 UNIQUE 제약조건이 설정되는 경우
- WHERE 문에서 INDEX를 '언급 ' 할 경우 자동으로 인덱스를 활용한다

    - WHERE EMP_NAME != '0' ;

시퀀스로 중복되지 않는 목록을 만들고
인덱스로 빠르게 찾아낸다

DCL (DATE CONTROL LANGUAGE)

계정

- 관리자 계정
    - DB의 생성과 관리를 담당하는 계정
    - 모든 '권한' , '책임'을 가지는 계정
    - SYS(관리자 계정) , SYSTEM (SYS에서 몇가지를 제외)

- 사용자 계정
    - DB에 대하여 질의 / 갱신 / 보고서 작성등의
    - 작업을 수행하는 계정
    - 업무에 필요한 '최소한'의 권한만을 소유하는 것을 원칙으로 한다

**DCL 이란 : 계정에 DB , DB객체에 대한 접근 권한을 부여하고 회수 할 수있는 DATA CONTROL LANGUAGE(데이터 제어 언어)이다

GRANT : 권한 부여
REVOKE : 권한 회수

시스템 권한

- 테이블에 생성 혹은 삭제와 관련된 명령

```
CRETAE SESSION   : 데이터베이스 접속 권한
CREATE TABLE     : 테이블 생성 권한
CREATE VIEW      : 뷰 생성 권한
CREATE SEQUENCE  : 시퀀스 생성 권한
CREATE PROCEDURE : 함수(프로시져) 생성 권한
CREATE USER      : 사용자(계정) 생성 권한
DROP USER        : 사용자(계정) 삭제 권한
DROP ANY TABLE   : 임의 테이블 삭제 권한
```

객체 권한

- 특정 객체를 조작(DML) 할 수 있는 권한

```
SELECT              TABLE, VIEW, SEQUENCE
INSERT              TABLE, VIEW
UPDATE             TABLE, VIEW
```

```
    DELETE              TABLE, VIEW
    ALTER               TABLE, SEQUENCE
    REFERENCES           TABLE
    INDEX               TABLE
    EXECUTE            PROCEDURE
```

쉽게 권한을 부여하기 위한 방법

- ROLE (역할)
- 묶어둔 권한(ROLE)을 특정계정에 부여 --> 부여받은 계정을 특정 권한을 가진 역할을 갖게 된다
- ROLE을 사용해 편하게 한번에 많은 권한을 부여 혹은 회수 할 수 있다

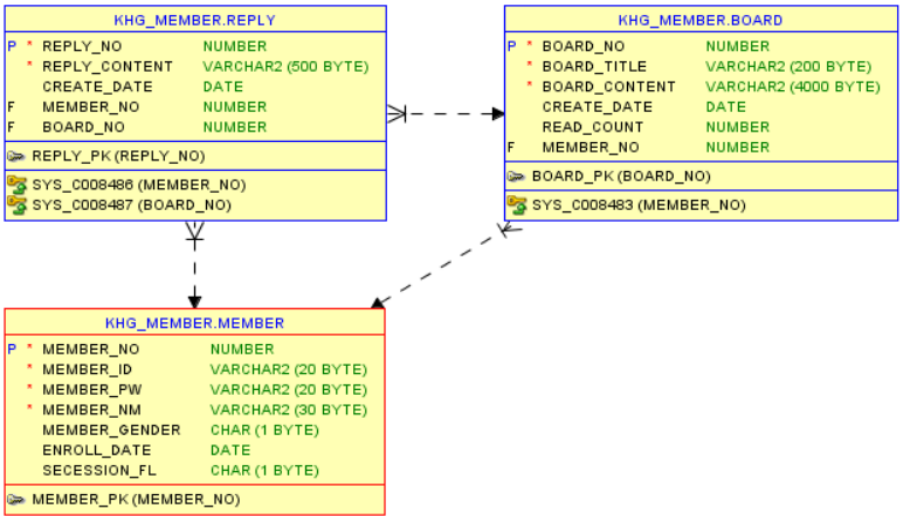
- CONNECT : DB 접속 권한 (= CREATE SESSION)
- RESOURCE : DB 사용을 위한 기본 객체 생성 권한 묶음

회원게시판을 위한 준비

DB 준비

- 회원의 정보를 저장한 테이블
- 게시판의 테이블을 저장할 테이블
- 댓글의 정보를 저장할 테이블 저장

- 3개의 테이블관 상속 관계



- PK 는 자동적으로 REFERENCES KEY 로 설정됨

- 3개의 테이블 구성 후 이클립스에서 구성
- member_board 자바프로젝트 생성

- XML 파일을 생성해 주는 클래스
  - CreateXML 생성

```
Properties prop = new Properties();

try {
    FileOutputStream fos = new FileOutputStream("driver.xml");
    // 내보낼 파일의 이름
    prop.storeToXML(fos, "DB Connection Data"); // xml 파일 생성

} catch (IOException e) {

    e.printStackTrace();
}
```

XML (eXtensible MarkUp Language)

- 단순화된 데이터 기술 형식
- Key 와 Value로만 이루어져있기 때문에 Map과 동일하다

Properties

- Key 와 Value가 String 자료형으로 제한된 MAP 형식 객체

fos 변수에 파일 출력 스트림을 통해 "driver.xml" 이라고 파일의 이름을 정의

Properties 자료형 prop 변수를 XML파일으로 저장한다(storeToXML)

prop.storeToXML(fos, "DB Connection Data"); -- xml 파일 생성

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<!-- 마크업 언어 주석 -->
<properties>
        <!-- comment : properties XML 파일 설명 -->
        <comment>DB Connection Data</comment>
        <!-- entry : Map의 Key , Value를 묶어 부른 명칭 -->
        <entry key = "driver">oracle.jdbc.driver.OracleDriver</entry>
        <!-- 공용계정 -->
        <!-- <entry key="url">jdbc:oracle:thin:@115.90.212.22:9000:xe</entry>
        <entry key="user">member</entry> -->
        <!-- 기본 계정 -->
        <entry key = "url">jdbc:oracle:thin:@localhost:1521:xe</entry>
```

```
        <entry key = "user">khg_member</entry>
        <entry key = "password">member1234</entry>
</properties>
```

- <!-- 까지 치면 자동으로 주석 처리 된다

entry : Map에서 key와 value를 묶어 부르는 명칭

- <entry key = "검색될 값"> 내용 (띄어쓰기도 인식될 수 있음으로 주의) </entry>
- 이때 메소드에서 호출할 때 entry옆에 선언된 key으로 호출해 data를 가져온다.

DB와 연결을 너무 자주 하기 때문에 자주 사용하고 공통적으로 사용하는 기술들을 JAVA TEMPLATE으로 묶어둔다

- DB연결
- JDBC 자원 반환
- 트랜잭션 제어

등 자주 사용하는 JDBC의 관련 코드들을 하나로 묶어 둔다

```java
// 필드
// static 메서드세어
// 해당 필드를 사용할려면
// 필드도 static이여야 한다.
private static Connection conn = null; // 초기값 null

// 메소드

// DB연결 정보를 담고 있느 Connection 객체 반환 메서드
public static Connection getConnection() {

// cloes( ) 메서드 작성
// Connection 반환 메소드
public static void close(Connection conn) {

// Statement(부모) , PreparedStatement 반환 메소드
public static void close(Statement stmt) {

// Statement(부모) , PreparedStatement 반환 메소드
public static void close(ResultSet rs) {

// 트랜잭션 제어 메소드
// commit 메소드
public static void commit(Connection conn) {

// rollbakc 메소드
public static void rollbakc(Connection conn) {
```