**▶내장형 함수(BUILT-IN Function)**

1. 내장형 함수

- 형변환 함수, 문자열 및 숫자형 함수, 날짜형 함수 등

- 모든 데이터베이스는 내장형 함수를 가지고 있으며, DBMS별로 차이는 있으나 거의 비슷한   
 방법으로 사용 가능

- 관계형 데이터베이스는 1970년대에 첫 소개, 1980년대에 상용화됨

2. DUAL 테이블

- Oracle 데이터베이스에 의해 자동으로 생성되는 테이블(Dummy table)

- Oracle 데이터베이스 사용자가 임시로 사용할 수 있는 테이블

3. 내장형 함수의 종류

- DUAL 테이블에 문자형 내장형 함수를 사용하는 경우

SELECT ASCII(‘a’), SUBSTR(‘ABC’, 1, 2),   
 LENGTH(‘A BC’), LTRIM(‘ ABC’),  
 LENGTH(LTRIM(‘ ABC’))  
 FROM DUAL

1. 문자열 함수

|  |  |
| --- | --- |
| **문자열 함수** | **설 명** |
| **ASCII(문자)** | 문자 혹은 숫자를 ASCII 코드값으로 변환 |
| **CHAR(ASCII 코드값)** | ASCII 코드값을 문자로 변환 |
| **SUBSTR(문자열,m,n)** | 문자열에서 m번째 위치부터 n개 자른다 |
| **CONCAT(문자열1, 문자열2)** | - 문자열1번과 문자열2번을 결합 - ORACLE은 ‘||’, MS SQL은 ‘+’를 사용할 수 있음 |
| **LOWER(문자열)** | 영문 대문자를 소문자로 변환 |
| **UPPER(문자열)** | 영문 소문자를 대문자로 변환 |
| **LENGTH 혹은 LEN(문자열)** | **공백 포함하여** 문자열의 길이를 반환 |
| **LTRIM(문자열, 지정문자)** | - 문자열 왼쪽에서 지정된 문자와 동일한 문자를 삭제 - 지정된 문자를 생략 시 공백을 삭제 |
| **RTRIM(문자열, 지정문자)** | - 문자열 오른쪽에서 지정된 문자와 동일한 문자 삭제 - 지정된 문자 생략 시 공백 삭제 |
| **TRIM(문자열, 지정문자)** | - 문자열 양쪽에서 지정된 문자와 동일한 문자 삭제 - 지정된 문자 생략 시 공백 삭제 |

2) 날짜형 함수

|  |  |
| --- | --- |
| **비교 연산자** | **설 명** |
| **SYSDATE** | 오늘의 날짜를 날짜 타입으로 알려줌 |
| **EXTRACT(‘YEAR’ | ‘MONTH’ | ‘DAY’ from dual)** | 날짜에서 년,월,일을 조회 |

※ TO\_CHAR 함수는 형변환 함수 중에서 가장 많이 사용하는 것으로,   
 숫자나 날짜를 원하는 포맷의 문자열로 변환  
 예) SELECT TO\_CHAR(SYSDATE, ‘YYYYMMDD’)  
 FROM DUAL

3) 숫자형 함수

|  |  |
| --- | --- |
| **숫자형 함수** | **설 명** |
| **ABS(숫자)** | 절대값을 반환 |
| **SIGN(숫자)** | 양수이면 1, 음수이면 -1, 0이면 0을 반환 |
| **MOD(숫자1, 숫자2)** | - 숫자1(나눠질값)을 숫자2로 나누기한 **나머지값**을 반환 - %를 사용해도 됨 |
| **CEIL/CEILING(숫자)** | - 숫자보다 **크거나 같은 최소의 정수**를 반환  예) 3.14 → 4 |
| **FLOOR(숫자)** | - **소수점 이하의 수를 버림** - 숫자보다 작거나 같은 최대의 정수를 반환 예) 3.14 → 3 |
| **ROUND(숫자, m)** | - **소수점 m 자리에서 반올림**  - m의 기본값(Default Value)은 0이다 |
| **TRUNC(숫자, m)** | - 소수점 m 자리에서 절삭  - m의 기본값(Default Value)은 0이다 |

**[Quiz]**

**숫자형 함수 중에서 숫자보다 작거나 같은 최대의 저수를 반환하는 것은?**

① CEIL( )  
② TRUNC( )  
③ FLOOR( )  
④ ABS( )

**▶DECODE와 CASE문**

1. DECODE

- DECODE문으로 IF문 구현 가능

SELECT DECODE(EMPNO, 1000, ‘TRUE’, ‘FALSE’)

FROM EMP ;

* EMPNO = 1000이면 TRUE, 아니면 FALSE를 반환

2. CASE문

- 조건을 WHEN구에 사용하고 THEN은 해당 조건이 참이면 실행되며 거짓이면 ELSE구가 실행

SELECT CASE  
 WHEN EMPNO = 1000 THEN ‘A’  
 WHEN EMPNO = 1001 THEN ‘B’ ELSE ‘C’   
 END  
 FROM EMP ;

🡪 EMPNO가 1000이면 A, 1001이면 B, 둘 다 아니면 C를 출력

**[Quiz]**

**성별이 남자이면 M, 여자이면 F를 출력하는 DECODE문으로 올바른 것은?  
(단, 성별 컬럼에는 남자와 여자만 있음)**

① DECODE(성별, ‘남자’, ‘M’, ‘F’)  
② DECODE(‘남자’, ‘성별’, ‘F’, ‘M’)  
③ DECODE(성별, ‘여자’, ‘M’, ‘F’)  
④ DECODE(성별, ‘남자’, ‘F’, NULL)

**▶ROWNUM과 ROWID**

1) ROWNUM

- ROWNUM은 오라클 데이터베이스의 SELECT문 결과에 대해서 논리적인 일련번호를 부여

- ROWNUM은 조회되는 행의 수를 제한할 때 자주 사용

- ROWNUM은 화면에 데이터를 출력할 때 부여되는 논리적 순번이며, 만약 ROWNUM을 사용  
 해서 페이지단위 출력을 하기 위해서는 인라인 뷰(Inline view)를 사용해야 함

SELECT \* FROM EMP  
 WHERE ROWNUM <= 10 ; 🡨 10개의 ROW만 조회(출력)

인라인 뷰를 사용하고 ROWNUM에 별칭을 사용한 경우

SELECT \*   
 FROM ( SELECT ROWNUM list , ENAME  
 FROM EMP ) **--🡪 FROM절에서 SELECT문을 사용하면 Inline view라고 함**  
 WHERE list <= 5

**※ [TIP] SQL Server의 TOP 구문과 MySQL의 limit 구문**

Oracle의 ROWNUM = SQL Server는 TOP문을 사용 = MySQL은 LIMIT구 사용

예) 10개의 컬럼만 인출(Fetch)하고자 하는 경우

- SQL Server : select TOP(10) from EMP ;

- MySQL : select \* from EMP LIMIT 10 ;

2) ROWID

- ROWID는 ORACLE 데이터베이스 내에서 데이터를 구분할 수 있는 유일한 값이다.

- SELECT ROWID, EMPNO FROM EMP 와 같은 SELECT문으로 확인 가능

- ROWID를 통해 데이터가 어떤 데이터 파일의 어느 블록에 저장되어 있는지 알 수 있음

- ROWID의 구조

|  |  |  |
| --- | --- | --- |
| **구 조** | **길 이** | **설 명** |
| 오브젝트 번호 | 1~6 | 오브젝트(Object)별로 유일한 값을 가지고 있으며, 해당 오브젝트가 속해 있는 값이다 |
| 상대 파일 번호 | 7~9 | 테이블스페이스(Tablespace)에 속해 있는 데이터 파일에 대한 상대 파일번호이다 |
| 블록 번호 | 10~15 | 데이터 파일 내부에서 어느 블록에 데이터가 있는지 알려준다 |
| 데이터 번호 | 16~18 | 데이터 블록에 데이터가 저장되어 있는 순서를 의미한다 |

**[Quiz]**

**다음 중 ROWID에 대한 설명으로 올바르지 않은 것은?**

① ROWID는 데이터베이스에 저장되어 있는 데이터를 구분할 수 있는 유일한 값이다.  
② ROWID의 데이터 번호는 데이터 블록에 데이터가 저장되어 있는 순서를 의마한다.  
③ 테이블을 생성하고 데이터를 입력할 때 ROWID를 입력해야 사용할 수 있다.  
④ ROWID는 데이터가 어떤 데이터 파일의 어떤 블록에 저장되어 있는지 확인할 수 있다.

**▶WITH구문**

- WITH구문은 서브쿼리(Subquery)를 사용해서 임시 테이블이나 뷰처럼 사용할 수 있음

- 서브쿼리 블록에 별칭을 지정할 수 있음

- 옵티마이저는 SQL을 인라인 뷰나 임시 테이블로 판단함

**WITH viewData AS (**

SELECT \* FROM EMP1   
 UNION ALL   
 SELECT \* FROM EMP2  
 **)** SELECT \* FROM viewData WHERE EMPNO = 1000;

**▶DCL(Data Control Language)**

1) GRANT

- GRANT문은 데이터베이스 사용자에게 권한을 부여  
 - 데이터베이스 사용을 위해서는 권한이 필요하며, 연결/ 입력/ 수정/ 삭제/ 조회를 할 수 있다

GRANT privileges ON object TO user ;

🡪 privileges는 권한을 의미하며 object는 테이블명임   
 🡪 user는 Oracle 데이터베이스 사용자를 지정하면 됨

**- privileges(권한)**

|  |  |
| --- | --- |
| **권한** | **설 명** |
| **SELECT** | 지정된 테이블에 대해 SELECT 권한을 부여 |
| **INSERT** | 지정된 테이블에 대해 INSERT 권한을 부여 |
| **UPDATE** | 지정된 테이블에 대해 UPDATE 권한을 부여 |
| **DELETE** | 지정된 테이블에 대해 DELETE 권한을 부여 |
| **REFERENCES** | 지정된 테이블을 참조하는 제약조건을 생성하는 권한을 부여 |
| **ALTER** | 지정된 테이블에 대해서 수정할 수 있는 권한을 부여 |
| **INDEX** | 지정된 테이블에 대해서 인덱스를 생성할 수 있는 권한을 부여 |
| **ALL** | 테이블에 대한 모든 권한을 부여 |

예) GRANT SELECT, INSERT, UPDATE, DELETE   
 ON EMP   
 TO AP59903671   
 🡪 AP59903671에게 EMP테이블에 대해 조회/ 입력/ 수정/ 삭제 권한을 부여함

**- WITH GRANT OPTION**

|  |  |
| --- | --- |
| **GRANT 옵션** | **설 명** |
| **WITH GRANT OPTION** | - 특정 사용자에게 권한을 부여할 수 있는 권한을 부여함 - 권한을 A사용자가 B에게 부여하고 B가 다시 C에게 부여한 후에 권한을 취소(REVOKE)하면 모든 권한이 회수된다 |
| **WITH ADMIN OPTION** | - 태이블에 대한 모든 권한을 부여함 - 권한을 A사용자가 B에게 부여하고, B가 다시 C에게 부여한 후에 권한을 취소(REVOKE)하면 B사용자 권한만 취소된다 |

예) GRANT SELECT , INSERT, UPDATE, DELETE

ON EMP

TO EMP00001 WITH GRANT OPTION ;

🡪 EMP00001 직원에게 **‘권한을 부여할 수 있는 권한’**을 부여한다

2) REVOKE

- REVOKE문은 데이터베이스 사용자에게 부여된 권한을 회수한다

REVOKE privileges ON object FROM user ;

**▶ TCL(Transaction Control Language)**

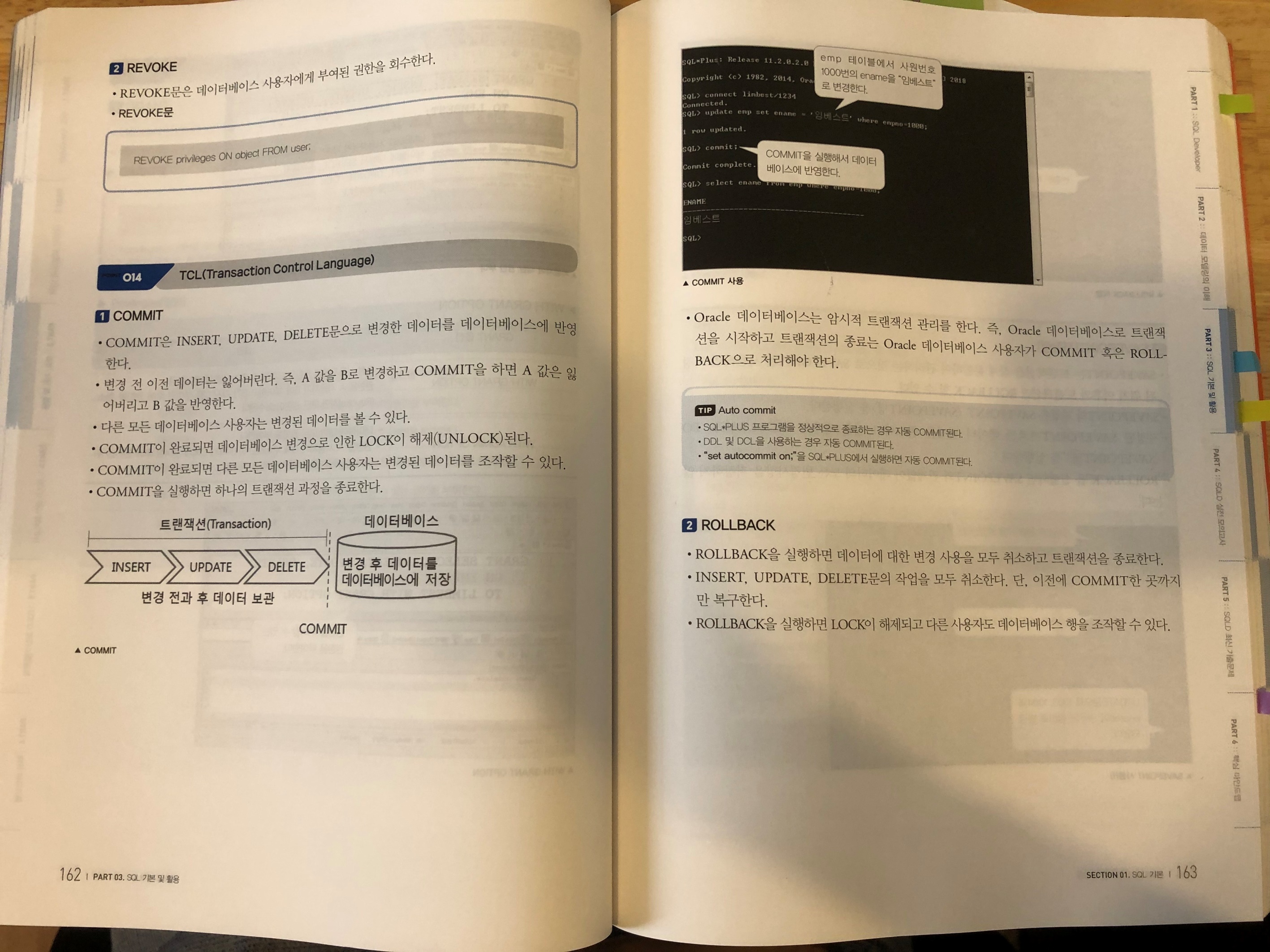
1) COMMIT

- COMMIT은 INSERT, UPDATE, DELETE문으로 변경한 데이터를 데이터베이스에 반영한다

- 변경 전 이전 데이터는 잃어버린다. 즉 A값을 B로 변경하고 COMMIT을 하면   
A값은 잃어버리고 B값을 반영한다

- 다른 모든 데이터베이스 사용자는 변경된 데이터를 볼 수 있다.  
 - COMMIT이 완료되면 데이터베이스 변경으로 인한 LOCK이 해제(UNLOCK)되며,   
 다른 모든 데이터베이스 사용자는 변경된 데이터를 조작할 수 있다.

- COMMIT을 실행하면 하나의 트랜잭션 과정을 종료한다.



**※ [TIP] Auto COMMIT**

- SQL\*PLUS 프로그램을 정상적으로 종료하는 경우 자동 COMMIT 된다.  
 - DDL 및 DCL을 사용하는 경우 자동 COMMIT 된다.  
 - **“set autocommit on;”** 을 SQL\*PLUS에서 실행하면 자동 COMMIT 된다.

2) ROLLBACK

- ROLLBACK 을 실행하면 데이터에 대한 변경 사용을 모두 취소하고 트랜잭션을 종료한다  
 - INSERT, UPDATE, DELETE문의 작업을 모두 취소한다.   
 단, 이전에 COMMIT한 곳까지만 복구한다.   
 - ROLLBACK을 실행하면 LOCK이 해제되고 다른 사용자도 데이터베이스 행을 조작할 수 있다.

3) SAVEPOINT(저장점)

- SAVEPOINT는 트랜잭션을 작게 분할하여 관리하는 것으로, SAVEPOINT를 사용하면   
 지정된 위치 이후의 트랜잭션만 ROLLBACK할 수 있다.   
 - SAVEPOINT의 지정은 SAVEPOINT <SAVEPOINT명>을 실행한다.

- 지정된 SAVEPOINT까지만 데이터 변경을 취소하고 싶은 경우는   
 “ROLLBACK TO <SAVEPOINT명>”을 실행한다.

- ROLLBACK을 실행하면 SAVEPOINT와 관계없이 데이터의 모든 변경사항을 저장하지 않는다.

**[Quiz]**

**다음 보기의 SQL문이 실행되면 조회되는 행 수는?**

SAVEPOINT t1 ;  
 INSERT INTO EMP VALUES(10, 20) ;   
 SAVEPOINT t2 ;  
 INSERT INTO EMP VALUES(20, 30) ;   
 ROLLBACK TO t2 ;   
 COMMIT ;   
 SELECT \* FROM EMP ;