

TCL

(Transaction Control Language)

1. TCL이란?
2. 트랜잭션 이해하기
3. COMMIT 과 ROLLBACK
4. SAVEPOINT
5. 실습용 문제 풀이

1. TCL이란?

SQL 문법의 종류

SELECT

테이블에서 원하는 데이터를 조회한다.

DML

테이블에 데이터를 입력/삭제/수정한다.

DDL

테이블 같은 데이터 저장소 객체를 만들거나 수정한다.

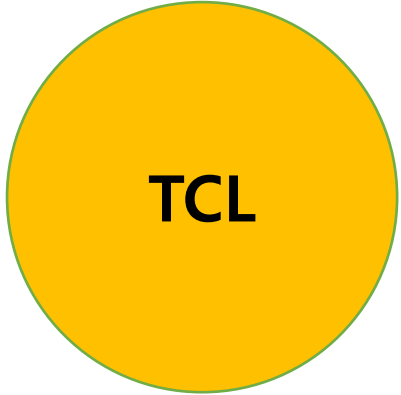
TCL

트랜잭션을 제어한다.

DCL

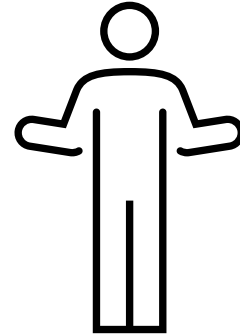
객체에 권한을 부여한다.

1. TCL이란?



트랜잭션을 제어한다.

트랜잭션은 어떤 업무를
수행하기 위한 일련의 단계
를 의미합니다.



COMMIT

ROLLBACK

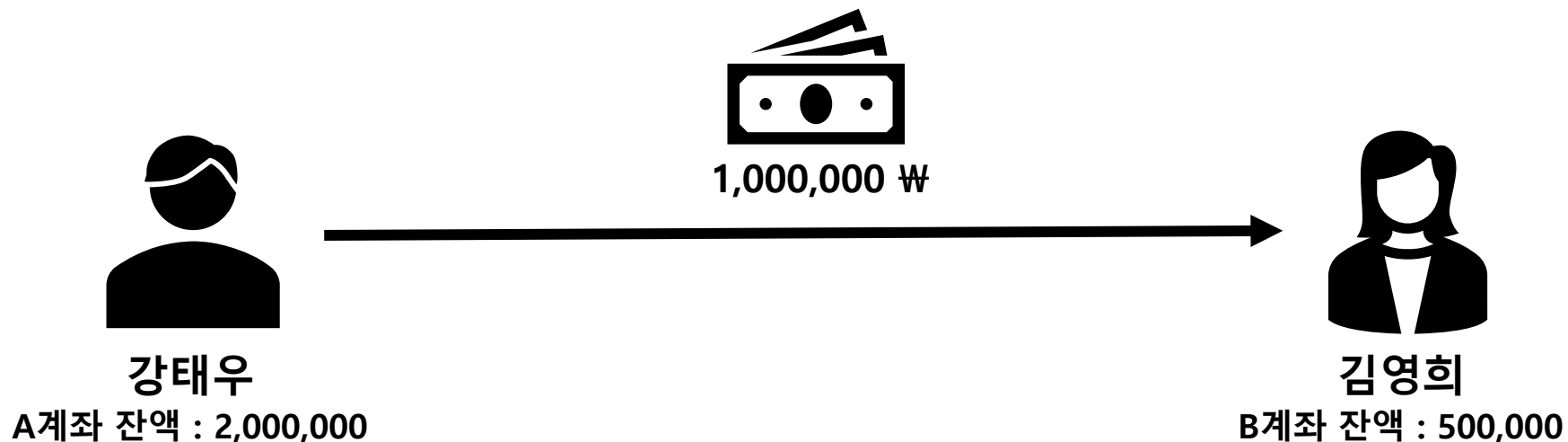
SAVEPOINT

2. 트랜잭션 이해하기

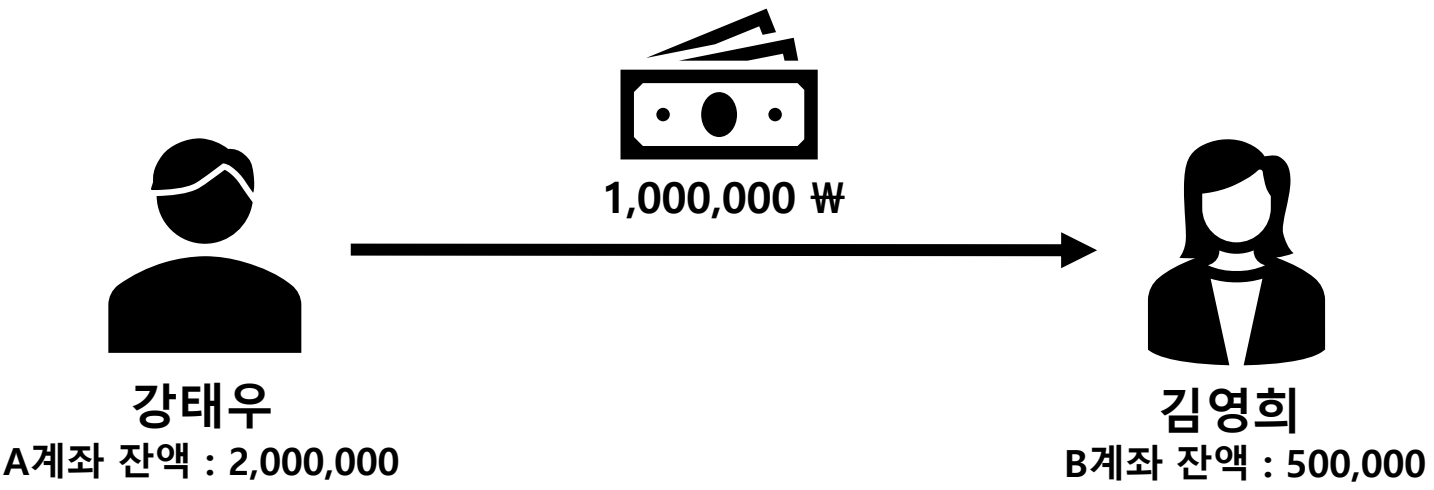
트랜잭션을 이해해봅시다.

강태우(A계좌)가 김영희(B계좌)에게 1,000,000원을 송금하려고 합니다.

어떤 **일련의 과정**을 거쳐 송금 업무가 이루어지는지 생각해봅시다.



2. 트랜잭션 이해하기



계좌정보 테이블

계좌번호	이름	잔액
A계좌	강태우	2000000
B계좌	김영희	500000
...

- 1. 강태우의 A계좌에 잔액이 일백만원 이상인지 확인합니다.
- 2. 강태우의 A계좌 잔액에서 일백만원을 차감합니다.
- 3. 김영희의 B계좌 잔액에서 일백만원을 더합니다.
- 4. 송금 업무가 완료되었습니다.

```
SELECT 잔액 FROM 계좌정보 WHERE 계좌번호 = 'A계좌' AND 잔액 >= 1,000,000 ;

UPDATE 계좌정보 SET 잔액 = 잔액 -1,000,000 WHERE 계좌번호 = 'A계좌' ;

UPDATE 계좌정보 SET 잔액 = 잔액+1,000,000 WHERE 계좌번호 = 'B계좌' ;

COMMIT;
```

3. COMMIT 과 ROLLBACK

송금을 위한 트랜잭션이 정상 처리되면

데이터를 영구 반영하기 위해 COMMIT합니다.

계좌정보 테이블

계좌번호	이름	잔액
A계좌	강태우	1000000
B계좌	김영희	1500000
...

영구 반영

T1 -> SELECT 잔액 FROM 계좌정보 WHERE 계좌번호 = 'A계좌' AND 잔액 >= 1,000,000 ;

T2 -> UPDATE 계좌정보 SET 잔액 = 잔액 -1,000,000 WHERE 계좌번호 = 'A계좌' ;

T3 -> UPDATE 계좌정보 SET 잔액 = 잔액+1,000,000 WHERE 계좌번호 = 'B계좌' ;

COMMIT; (송금 업무를 위한 일련의 절차가 정상 처리되었으니 데이터를 영구반영합니다.)

3. COMMIT 과 ROLLBACK

트랜잭션 도중에 오류가 생기면 어떻게 될까요?

계좌정보 테이블

계좌번호	이름	잔액
A계좌	강태우	1000000
B계좌	김영희	500000
...

1. 강태우의 A계좌에 잔액이 일백만원 이상인지 확인합니다.

```
SELECT 잔액 FROM 계좌정보 WHERE 계좌번호 = 'A계좌' AND 잔액 >= 1,000,000 ;
```

2. 강태우의 A계좌 잔액에서 일백만원을 차감합니다.

```
UPDATE 계좌정보 SET 잔액 = 잔액 - 1,000,000 WHERE 계좌번호 = 'A계좌' ;
```

~~3. 김영희의 B계좌 잔액에서 일백만원을 더합니다.~~

```
UPDATE 계좌정보 SET 잔액 = 잔액 + 1,000,000 WHERE 계좌번호 = 'B계좌' ;
```

ERROR!!!!

4. 송금 업무가 완료되었습니다.

COMMIT 불가! (잘못된 데이터가 발생합니다)

3. COMMIT 과 ROLLBACK

COMMIT / ROLLBACK 작동 예시

TAB3

COL1	COL2
------	------

```
INSERT INTO TAB3 (COL1 , COL2) VALUES ( 'A' , 'A') ;  
INSERT INTO TAB3 (COL1 , COL2) VALUES ( 'B' , 'B') ;  
COMMIT;  
INSERT INTO TAB3 (COL1 , COL2) VALUES ('C' , 'C') ;  
ROLLBACK ;  
INSERT INTO TAB3(COL1 ,COL2) VALUES ('D' , 'D') ;  
INSERT INTO TAB3(COL1 ,COL2) VALUES ('E' , 'E') ;  
COMMIT;  
  
SELECT COUNT(*) FROM TAB3 ;
```


3. COMMIT 과 ROLLBACK

COMMIT 실습을 해봅시다.

SQL DEVELOPER 를 하나 더 실행해봅시다. (총 2개)

Oracle SQL Developer : SHOPPING

파일(F) 편집(E) 보기(V) 이동(N) 실행(R) 소스 팀(M) 도구(T) 창(W) 도움말(H)

제목 없음3.sql x 제목 없음4.sql x 시작 페이지 x SHOPPING x

워크시트 질의 작성기

```
--1. PRD_ID 가 'P0001' 인 상품의 가격을 조회한다.  
SELECT PRD_NAME , PRD_PRICE --30000원  
FROM TB_PRD  
WHERE PRD_ID = 'P0001' ;  
  
--2. PRD_ID 가 'P0001' 인 상품의 가격을 5000원 올린다.  
UPDATE TB_PRD  
SET PRD_PRICE = PRD_PRICE + 5000  
WHERE PRD_ID = 'P0001' ;  
  
--3. 가격이 5000원 올랐는지 조회를 해본다.  
SELECT PRD_NAME , PRD_PRICE --35000원  
FROM TB_PRD  
WHERE PRD_ID = 'P0001' ;
```

스크립트 출력 x 질의 결과 x

PRD_NAME	PRD_PRICE
1 헤어드라이기	35000

Oracle SQL Developer : SHOPPING

파일(F) 편집(E) 보기(V) 이동(N) 실행(R) 소스 팀(M) 도구(T) 창(W) 도움말(H)

제목 없음3.sql x 제목 없음4.sql x SHOPPING x

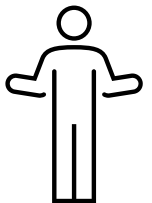
워크시트 질의 작성기

```
--4. 여기에서도 PRD_ID 가 'P0001' 인 상품의 가격을 조회한다.  
  
SELECT PRD_NAME , PRD_PRICE --가격은 얼마??  
FROM TB_PRD  
WHERE PRD_ID = 'P0001' ;
```

질의 결과 x

PRD_NAME	PRD_PRICE
1 헤어드라이기	30000

결과가 다른 이유는?



Oracle SQL Developer : SHOPPING

파일(F) 편집(E) 보기(V) 이동(N) 실행(R) 소스 팀(M) 도구(D) 창(W) 도움말(H)

제목 없음3.sql x 제목 없음4.sql x 시작 페이지 x SHOPPING x

워크시트 | 질의 작성기

```
--3. 가격이 5000원 올랐는지 조회를 해본다.  
SELECT PRD_NAME , PRD_PRICE --35000원  
FROM TB_PRD  
WHERE PRD_ID = 'P0001' ;
```

--5. COMMIT (영구반영) 을 실행해서
-- 데이터베이스가 변화가 영구 저장 되도록 한다.
COMMIT;

스크립트 출력 x | 질의 결과 x

작업이 완료되었습니다.(0.04초)

1 행 이 (가) 업데이트되었습니다.

커밋 완료.

Oracle SQL Developer : SHOPPING

파일(F) 편집(E) 보기(V) 이동(N) 실행(R) 소스 팀(M) 도구(D) 창(W) 도움말(H)

제목 없음3.sql x 제목 없음4.sql x SHOPPING x

워크시트 | 질의 작성기

```
--4. 여기에서도 PRD_ID 가 'P0001' 인 상품의 가격을 조회한다.  
  
SELECT PRD_NAME , PRD_PRICE --가격은 얼마???  
FROM TB_PRD  
WHERE PRD_ID = 'P0001' ;
```

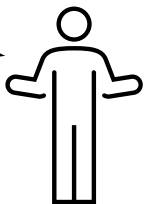
--6. 다시 한번 조회해본다. (영구저장 되었으므로 조회가능)
SELECT PRD_NAME , PRD_PRICE --35000원
FROM TB_PRD
WHERE PRD_ID = 'P0001' ;

질의 결과 x

SQL | 인출된 모든 행: 1(0.004초)

PRD_NAME	PRD_PRICE
헤어드라이기	35000

COMMIT으로 데이터베이스에
데이터가 영구반영되었습니다.

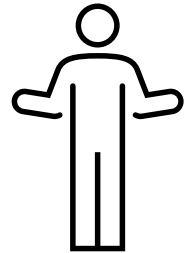


4. SAVEPOINT

ROLLBACK 을 특정 지점까지만

원상 복구를 하도록 조절해주는 문법입니다

실무에서 자주 쓰이지는 않지만
테스트할 때 사용할 수 있습니다.



4. SAVEPOINT

```
UPDATE TB_PRD SET PRD_PRICE = PRD_PRICE + 5000 WHERE PRD_ID = 'P0001' ; --현재 40000원

SAVEPOINT SV1 ;

UPDATE TB_PRD SET PRD_PRICE = PRD_PRICE + 5000 WHERE PRD_ID = 'P0001' ; --현재 45000원

SAVEPOINT SV2 ;

UPDATE TB_PRD SET PRD_PRICE = PRD_PRICE + 5000 WHERE PRD_ID = 'P0001' ; --현재 50000원

ROLLBACK TO SV2 ;
COMMIT; -- 상품의 가격은 얼마 ??
```

ROLLBACK을 SV2 지점까지만 되돌린다.

PRD_PRICE
45000

추가 지식!

오라클에서 DML (INSERT , UPDATE , DELETE)는
우리가 직접 COMMIT 을 해야 하지만

오라클에서 **DDL** (CREATE , ALTER , DROP 등)은
자동으로 COMMIT 이 됩니다.

이를 전문용어로 AUTO COMMIT 이라고 합니다.

(DML => AUTO COMMIT : FALSE
DDL => AUTO COMMIT : TRUE)

--1. 아까 헤어드라이기의 값을 30000으로 변경 (DML 는 자동 COMMIT 되지 않음!)

```
UPDATE TB_PRD |
    SET PRD_PRICE = 30000
WHERE PRD_ID = 'P0001' ;
```

--2. 임의의 테이블 CREATE 로 DDL 문장 실행 (DDL 은 자동 COMMIT 이 됨!)

```
CREATE TABLE TEST_222 (
    COL1 VARCHAR2(10) ,
    COL2 VARCHAR2(10) );
```

--3. ROLLBACK 으로 되돌리기 실행

```
ROLLBACK ;
```

--4. 헤어드라이기의 가격은 30000? 혹은?

```
SELECT PRD_PRICE
FROM TB_PRD
WHERE PRD_ID = 'P0001' ;
```

5. 실습용 문제풀이

(1). 아래와 같이 TAB3 이라는 테이블에는 COL1 , COL2 컬럼이 있습니다. 아직 아무런 데이터가 없습니다.

오른쪽 쿼리를 위에서 아래로 순서대로 실행했을 때, TAB3 테이블에는 총 몇 개의 행이 있을까요?

(단, COL1 은 PRIMARY KEY)

TAB3	
COL1	COL2

```
INSERT INTO TAB3 (COL1 , COL2) VALUES ( 'A' , 'A') ;  
INSERT INTO TAB3 (COL1 , COL2) VALUES ( 'B' , 'B') ;
```

COMMIT;

```
INSERT INTO TAB3 (COL1 , COL2) VALUES ('C' , 'C') ;
```

ROLLBACK ;

```
INSERT INTO TAB3(COL1 ,COL2) VALUES ('D' , 'D') ;
```

SAVEPOINT SV1 ;

```
INSERT INTO TAB3(COL1 ,COL2) VALUES ('E' , 'E') ;
```

ROLLBACK TO SV1 ; COMMIT;

5. 실습용 문제풀이

(2). 아래와 같이 TAB4 이라는 테이블에는 COL1 , COL2 컬럼이 있습니다. 아직 아무런 데이터가 없습니다.

오른쪽 쿼리를 위에서 아래로 순서대로 실행했을 때, TAB4 테이블에는 총 몇 개의 행이 있을까요?

(단, COL1 은 PRIMARY KEY)

TAB4

COL1	COL2
------	------

```
INSERT INTO TAB4 (COL1 , COL2) VALUES ( 'A' , 'A') ;
```

```
COMMIT;
```

```
INSERT INTO TAB4 (COL1 , COL2) VALUES ( 'B' , 'B') ;
```

```
INSERT INTO TAB4 (COL1 , COL2) VALUES ( 'C' , 'C') ;
```

```
CREATE TABLE HELLO( COL1 VARCHAR2(10) , COL2 VARCHAR2(10) ) ;
```

```
INSERT INTO TAB4(COL1 ,COL2) VALUES ( 'D' , 'D') ;
```

```
ROLLBACK ;
```

```
INSERT INTO TAB4(COL1 ,COL2) VALUES ( 'E' , 'E') ;
```

```
COMMIT;
```


DCL

(Data Control Language)

1. DCL이란?
2. GRANT , REVOKE , ROLE
3. 실제 예시

1. DCL이란?

SQL 문법의 종류

SELECT

테이블에서 원하는 데이터를 조회한다.

DML

테이블에 데이터를 입력/삭제/수정한다.

DDL

테이블 같은 데이터 저장소 객체를 만들거나 수정한다.

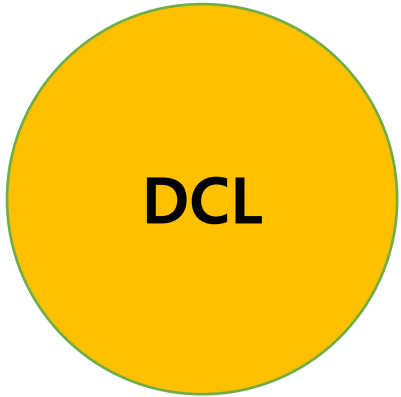
TCL

트랜잭션을 제어한다.

DCL

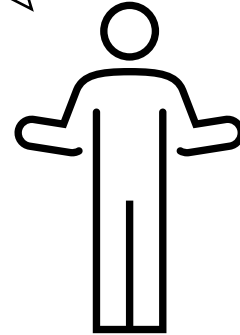
객체에 권한을 부여한다.

1. DCL이란?



객체에 권한을 부여한다.

데이터베이스는 권한을
부여/회수 하면서
객체를 보호합니다.



GRANT

REVOKE

ROLE

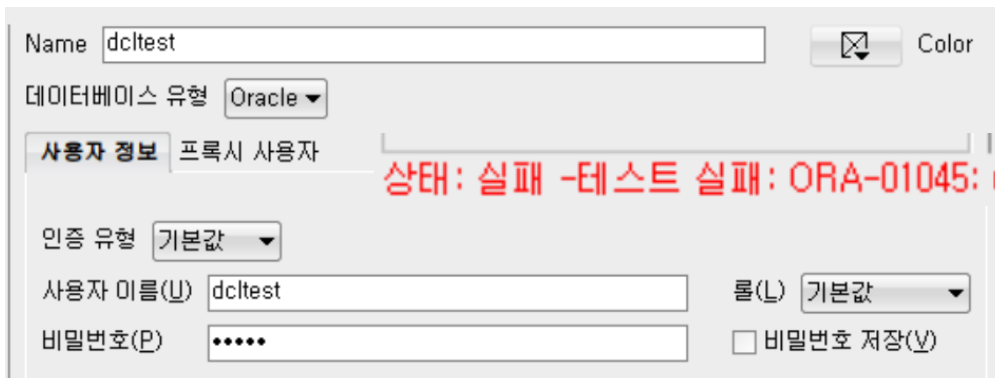
2. GRANT (권한부여)를 할 때 사용하는 DCL문법)

아래 계정을 생성해 SQL DEVELOPER로 접속을 시도해봅시다.

(1) SQL COMMAND LINE 에서 DCLTEST 라는 계정을 생성 (SYSTEM 관리자계정으로 접속)

```
SQL> CONN SYSTEM/12345 ;  
Connected.  
SQL> CREATE USER DCLTEST IDENTIFIED BY 12345 ;  
  
User created.
```

(2) SQL DEVELOPER 에서 작성 후 테스트를 클릭



상태: 실패 -테스트 실패: ORA-01045: user DCLTEST lacks CREATE SESSION privilege; logon denied

DCLTEST 계정은 CREATE SESSION 권한이 부족합니다.

2. GRANT (권한부여)를 할 때 사용하는 DCL문법)

GRANT 권한 [ON 대상테이블] TO 권한을부여할계정

(3) GRANT (부여합니다) CREATE SESSION (이 권한을) TO DCLTEST (DCLTEST계정에게)

```
SQL> GRANT CREATE SESSION TO DCLTEST ;  
Grant succeeded.
```

(4) DCLTEST 계정으로 다시 테스트를 시도해봅시다.

Name: dcltest

데이터베이스 유형: Oracle

사용자 정보: 프로시 사용자

인증 유형: 기본값

사용자 이름(U): dcltest

비밀번호(P):

롤(R): 기본값

☐ 비밀번호 저장(Y)

상태: 성공

2. ROLE (권한을 묶어 놓은 권한 모음집)

ROLE 은 여러 권한을 한번에 부여할 수 있습니다.

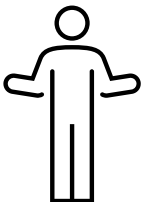
```
GRANT CONNECT , RESOURCE TO DCLTEST;
```

CONNECT	RESOURCE
ALTER SESSION	CREATE CLUSTER
CREATE SESSION	CREATE PROCEDURE
CREATE SYNONYM	CREATE TABLE
CREATE TABLE	CREATE TREIGGER
....

권한을 하나씩 주기엔 종류가 너무 많습니다.

보통은 비슷한 종류의 권한끼리 모아 놓은 **ROLE** 이라는 개념으로 한번에 부여합니다.

대표적인 ROLE :
CONNECT , RESOURCE 등



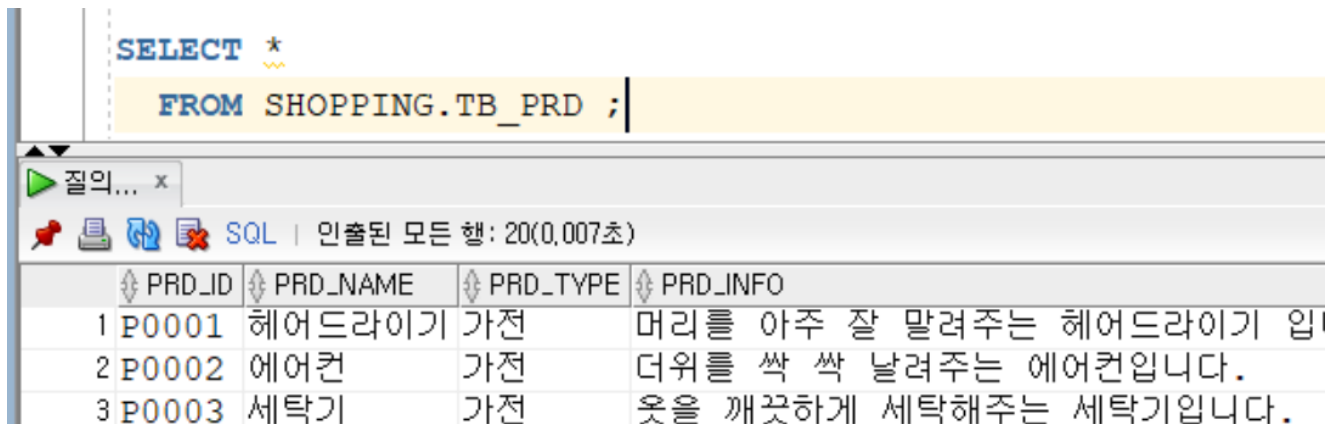
2. GRANT (권한부여)를 할 때 사용하는 DCL문법)

GRANT 권한 [ON 대상테이블] TO 권한을부여할계정

- (5) DCLTEST 계정에서 SHOPPING 계정이 가지고 있는 TB_PRD 테이블을
SELECT , UPDATE 할 권한을 부여합니다. (특정 테이블에 권한 부여시 ON 추가 사용)

```
SQL> GRANT SELECT , UPDATE ON SHOPPING.TB_PRD TO DCLTEST ;  
Grant succeeded.
```

- (6) DCLTEST 계정으로 아래 쿼리를 실행해봅시다.



The screenshot shows a SQL query execution window. The query is `SELECT * FROM SHOPPING.TB_PRD ;`. The results are displayed in a table with 4 columns: PRD_ID, PRD_NAME, PRD_TYPE, and PRD_INFO. There are 3 rows of data.

PRD_ID	PRD_NAME	PRD_TYPE	PRD_INFO
1 P0001	헤어드라이기	가전	머리를 아주 잘 말려주는 헤어드라이기 입
2 P0002	에어컨	가전	더위를 싹 싹 날려주는 에어컨입니다.
3 P0003	세탁기	가전	옷을 깨끗하게 세탁해주는 세탁기입니다.

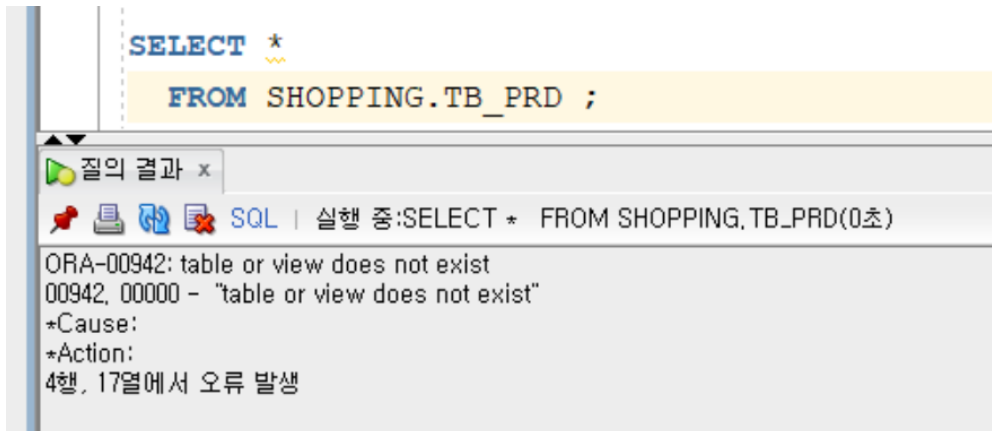
3. REVOKE (권한회수)를 할 때 사용하는 DCL문법)

REVOKE 권한 [ON 대상테이블] FROM 권한을뺏어갈계정

- (7) DCLTEST 계정으로부터 SHOPPING 계정이 가지고 있는 TB_PRD 테이블을 SELECT , UPDATE 할 권한을 다시 회수합니다. (특정 테이블에 권한 부여시 ON 추가 사용)

```
SQL> REVOKE SELECT , UPDATE ON SHOPPING.TB_PRD FROM DCLTEST ;  
Revoke succeeded.
```

- (8) DCLTEST 계정으로 아래 쿼리를 실행해봅시다.



권한 부여 실습

1. SQL COMMAND LINE을 열어 SYSTEM 관리자 계정으로 접속합니다. (ID : SYSTEM , PWD : 12345)
2. hacker 라는 계정을 하나 생성합니다. (비밀번호는 12345)
3. 계정에 다음 권한을 순서대로 부여합니다.
 - (1) 데이터베이스 접속이 가능하도록 CREATE SESSION을 부여
 - (2) 여러 권한을 한번에 처리할 수 있게 CONNECT 와 RESOURCE 라는 ROLE 권한을 부여
 - (3) SHOPPING 계정의 TB_MEMBER 테이블을 조회, 수정할 수 있게 SELECT , UPDATE 권한을 부여
4. HACKER 계정으로 sql developer 에 접속합니다. (name 은 hacker 로 설정)
5. SHOPPING 계정의 TB_MEMBER 테이블에서 PAY_CARD_NO 의 값을 모두 'XXXXX' 로 변경하고 COMMIT을 해버립니다.
(HACKER가 불법적으로 사용자의 결제카드번호를 잘못된 값으로 바꿔버린 상황)
6. 다시 관리자 계정으로 가서 문제를 발생시킨 HACKER 계정으로부터 SHOPPING 계정의 TB_MEMBER 테이블을 SELECT , UPDATE 할 권한을 회수해버립니다.

TCL/DCL END