7-2. 트랜잭션 처리, Merge문, 뷰, 데이터 딕셔너리

홍형경
chariehong@gmail.com
2020.06

1. 트랜잭션 (Transaction) 처리

- 트랜잭션은 거래라는 뜻

- 오류가 났을 경우는 거래 자체가 없었던 것으로 처리
- 입금 계좌에 돈이 확인된 다음에야 거래를 성사시킴

1. 트랜잭션 (Transaction) 처리

· SQL에서는 COMMIT, ROLLBACK 문장으로 트랜잭션 처리

· 거래 성공 ➔ **COMMIT** : 변경된 데이터 최종 저장

· 거래 실패 ➔ **ROLLBACK** : 변경 이전 상태로 회귀

· INSERT, UPDATE, DELETE, MERGE 문 실행 후 오류 없을 경우 반드시 COMMIT 문 실행

· 데이터 가공 작업 실패나 기타 사유 (예, WHERE 절 없이 DELETE 했을 경우)로 인해
작업 전 상태로 가고 싶다면 ROLLBACK 문 실행

1. 트랜잭션 (Transaction) 처리

· COMMIT, ROLLBACK은 마지막 COMMIT, ROLLBACK 문을 실행한 이후 내역에 대해 적용

```
INSERT INTO 테이블1
… ;

COMMIT;

UPDATE 테이블1 SET 컬럼1 = 값
WHERE …;

ROLLBACK;
```

테이블1에 대한 INSERT 작업 반영

테이블1에 대한 UPDATE 작업 취소

INSERT 작업은 이미 반영됨

1. 트랜잭션 실습

· 테이블 복제

```
CREATE TABLE emp_tran AS
SELECT *
FROM emp1;
```

· emp_tran 테이블 **생성** +
  emp1 테이블 데이터 **복사**

```
SELECT *
  FROM emp_tran;
```

	EMP_NO	EMP_NAME	SALARY	HIRE_DATE	DEPT_ID
1	100	Steven King	24000	2003-06-17 00:00:00	90
2	101	Neena Kochhar	17000	2005-09-21 00:00:00	90
3	102	Lex De Haan	17000	2001-01-13 00:00:00	90
4	103	Alexander Hunold	9000	2006-01-03 00:00:00	60
5	104	Bruce Ernst	6000	2007-05-21 00:00:00	60
6	105	David Austin	4800	2005-06-25 00:00:00	60
7	106	Valli Pataballa	4800	2006-02-05 00:00:00	60
8	107	Diana Lorentz	4200	2007-02-07 00:00:00	60
9	108	Nancy Greenberg	12008	2002-08-17 00:00:00	100
10	109	Daniel Faviet	9000	2002-08-16 00:00:00	100
11	111	Ismael Sciarra	7700	2005-09-30 00:00:00	100
12	113	Luis Popp	6900	2007-12-07 00:00:00	100
13	114	Den Raphaely	11000	2002-12-07 00:00:00	30
14	115	Alexander Khoo	3100	2003-05-18 00:00:00	30
15	116	Shelli Baida	2900	2005-12-24 00:00:00	30
16	117	Sigal Tobias	2800	2005-07-24 00:00:00	30
17	118	Guy Himuro	2600	2006-11-15 00:00:00	30
18	119	Karen Colmenares	2500	2007-08-10 00:00:00	30
19	120	Matthew Weiss	8000	2004-07-18 00:00:00	50
20	121	Adam Fripp	8200	2005-04-10 00:00:00	50
21	122	Payam Kaufling	7900	2003-05-01 00:00:00	50
22	123	Shanta Vollman	6500	2005-10-10 00:00:00	50
23	124	Kevin Mourgos	5800	2007-11-16 00:00:00	50
24	126	Irene Mikkilineni	2700	2006-09-28 00:00:00	50
25	129	Laura Bissot	3300	2005-08-20 00:00:00	50

1. 트랜잭션 실습

```
DELETE emp_tran
WHERE dept_id = 90;

COMMIT;

SELECT * FROM emp_tran;

UPDATE emp_tran
      SET emp_name = 'HAHA'
  WHERE dept_id = 60;

ROLLBACK;

SELECT * FROM emp_tran;
```

	EMP_NO	EMP_NAME	SALARY	HIRE_DATE	DEPT_ID
1	103	Alexander Hunold	9000	2006-01-03 00:00:00	60
2	104	Bruce Ernst	6000	2007-05-21 00:00:00	60
3	105	David Austin	4800	2005-06-25 00:00:00	60
4	106	Valli Pataballa	4800	2006-02-05 00:00:00	60
5	107	Diana Lorentz	4200	2007-02-07 00:00:00	60
6	108	Nancy Greenberg	12008	2002-08-17 00:00:00	100
7	109	Daniel Faviet	9000	2002-08-16 00:00:00	100
8	111	Ismael Sciarra	7700	2005-09-30 00:00:00	100
9	113	Luis Popp	6900	2007-12-07 00:00:00	100
10	114	Den Raphaely	11000	2002-12-07 00:00:00	30
11	115	Alexander Khoo	3100	2003-05-18 00:00:00	30
12	116	Shelli Baida	2900	2005-12-24 00:00:00	30
13	117	Sigal Tobias	2800	2005-07-24 00:00:00	30

	EMP_NO	EMP_NAME	SALARY	HIRE_DATE	DEPT_ID
1	103	HAHA	9000	2006-01-03 00:00:00	60
2	104	HAHA	6000	2007-05-21 00:00:00	60
3	105	HAHA	4800	2005-06-25 00:00:00	60
4	106	HAHA	4800	2006-02-05 00:00:00	60
5	107	HAHA	4200	2007-02-07 00:00:00	60

	EMP_NO	EMP_NAME	SALARY	HIRE_DATE	DEPT_ID
1	103	Alexander Hunold	9000	2006-01-03 00:00:00	60
2	104	Bruce Ernst	6000	2007-05-21 00:00:00	60
3	105	David Austin	4800	2005-06-25 00:00:00	60
4	106	Valli Pataballa	4800	2006-02-05 00:00:00	60
5	107	Diana Lorentz	4200	2007-02-07 00:00:00	60
6	108	Nancy Greenberg	12008	2002-08-17 00:00:00	100
7	109	Daniel Faviet	9000	2002-08-16 00:00:00	100

2. MERGE 문

· INSERT와 UPDATE를 한 번에 처리

· 대상 테이블에 대해 조건에 따라 INSERT 나 UPDATE 를 수행

· 일반적으로 테이블의 주요 키 값을 체크, 해당 값이 존재하면 UPDATE,
존재하지 않으면 INSERT 수행

2. Merge 문

· 구문

- MERGE INTO 대상테이블명
  USING 참조테이블 or 서브쿼리
  ON 조인조건
  WHEN MATCHED THEN
      UPDATE SET 컬럼1 = 값1, 컬럼2 = 값2, …
  WHEN NOT MATCHED THEN
  INSERT (컬럼1, 컬럼2, …)
  VALUES (값1, 값2, …);

2. Merge문 실습

· 테이블 복제

```
CREATE TABLE dept_mgr AS
SELECT *
FROM departments;
```

	DEPARTME...	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700
8	80	Sales	145	2500
9	90	Executive	100	1700
10	100	Finance	108	1700
11	110	Accounting	205	1700
12	120	Treasury	(null)	1700
13	130	Corporate Tax	(null)	1700

```
ALTER TABLE dept_mgr
ADD CONSTRAINTS dept_mgr_pk PRIMARY KEY (department_id);

SELECT *
  FROM dept_mgr;
```


2. Merge문 실습

```
MERGE INTO dept_mgr a
USING ( SELECT 280 dept_id, '영업부(Merge)' dept_name
                FROM dual
               UNION ALL
              SELECT 285 dept_id, '경리부(Merge)' dept_name
                FROM dual
            ) b
  ON ( a.department_id = b.dept_id )
WHEN MATCHED THEN  -- ON 조건에 만족하는 건이 있으면
UPDATE SET a.department_name = b.dept_name
WHEN NOT MATCHED THEN  --일치하는 건이 없으면
INSERT (a.department_id, a.department_name)
VALUES (b.dept_id, b.dept_name);


SELECT *
  FROM dept_mgr;
```

```
2개 행 이(가) 병합되었습니다.
```

21	210	IT Support	(null)	1700
22	220	NOC	(null)	1700
23	230	IT Helpdesk	(null)	1700
24	240	Government Sales	(null)	1700
25	250	Retail Sales	(null)	1700
26	260	Recruiting	(null)	1700
27	280	영업부(Merge)	(null)	(null)
28	285	경리부(Merge)	(null)	(null)

2. Merge문 실습

```
MERGE INTO dept_mgr a
USING ( SELECT 280 dept_id, '영업부(Merge)2' dept_name
          FROM dual
         UNION ALL
        SELECT 285 dept_id, '경리부(Merge)2' dept_name
          FROM dual
      ) b
  ON ( a.department_id = b.dept_id )
WHEN MATCHED THEN  -- 일치하는 건이 있으면
UPDATE SET a.department_name = b.dept_name
WHEN NOT MATCHED THEN  --일치하는 건이 없으면
INSERT (a.department_id, a.department_name)
VALUES (b.dept_id, b.dept_name);
```

```
2개 행 이(가) 병합되었습니다.
```

```
SELECT *
  FROM dept_mgr;
```

21	210	IT Support	(null)	1700
22	220	NOC	(null)	1700
23	230	IT Helpdesk	(null)	1700
24	240	Government Sales	(null)	1700
25	250	Retail Sales	(null)	1700
26	260	Recruiting	(null)	1700
27	280	영업부(Merge)2	(null)	(null)
28	285	경리부(Merge)2	(null)	(null)

2. Merge문 실습

```
MERGE INTO dept_mgr a
USING ( SELECT 280 dept_id, '영업부(Merge)3' dept_name
          FROM dual
         UNION ALL
        SELECT 290 dept_id, '전산팀(Merge)' dept_name
          FROM dual
      ) b
  ON ( a.department_id = b.dept_id )
WHEN MATCHED THEN  -- 일치하는 건이 있으면
UPDATE SET a.department_name = b.dept_name
WHEN NOT MATCHED THEN  --일치하는 건이 없으면
INSERT (a.department_id, a.department_name)
VALUES (b.dept_id, b.dept_name);
```

```
2개 행 이(가) 병합되었습니다.
```

```
SELECT *
  FROM dept_mgr;
```

22	220	NOC	(null)	1700
23	230	IT Helpdesk	(null)	1700
24	240	Government Sales	(null)	1700
25	250	Retail Sales	(null)	1700
26	260	Recruiting	(null)	1700
27	280	영업부(Merge)3	(null)	(null)
28	285	경리부(Merge)2	(null)	(null)
29	290	전산팀(Merge)	(null)	(null)

3. 뷰 (View)

· 하나 혹은 그 이상의 다른 테이블이나 뷰로 구성된 논리적 객체 (테이블처럼 동작)

· 뷰 자체에는 데이터가 저장되어 있지 않음

· 하나의 뷰가 또 다른 뷰에서 사용 될 수 있음

· 뷰의 용도

- 테이블 데이터 보안 강화 ➔ 컬럼이나 ROW 접근 제한
- 데이터 복잡성 숨김 ➔ 복잡하게 얽힌 쿼리를 뷰로 만들어 사용
- 테이블 구조 변경에 따른 영향도 감소 ➔ 신규 컬럼 추가 시에도 영향 받지 않음

3. 뷰 (View)

· 뷰 생성

```
CREATE OR REPLACE VIEW 뷰이름 AS
 SELECT 문;
```

· 뷰 수정

```
CREATE OR REPLACE VIEW 뷰이름 AS
 SELECT 문;
```

· 뷰 삭제

```
DROP VIEW 뷰이름;
```


3. View 실습

```
SELECT a.employee_id,
        a.first_name || ' ' || a.last_name emp_names,
       b.department_id ,b.department_name
 FROM employees a,
      departments b
WHERE a.department_id = b.department_id
ORDER BY 1;
```

	EMPLOYEE_ID	EMP_NAMES	DEPARTMENT_ID	DEPARTMENT_NAME
1	100	Steven King	90	Executive
2	101	Neena Kochhar	90	Executive
3	102	Lex De Haan	90	Executive
4	103	Alexander Hunold	60	IT
5	104	Bruce Ernst	60	IT
6	105	David Austin	60	IT
7	106	Valli Pataballa	60	IT
8	107	Diana Lorentz	60	IT
9	108	Nancy Greenberg	100	Finance
10	109	Daniel Faviet	100	Finance
11	110	John Chen	100	Finance
12	111	Ismael Sciarra	100	Finance
13	112	Jose Manuel Urman	100	Finance
14	113	Luis Popp	100	Finance
15	114	Den Raphaely	30	Purchasing
16	115	Alexander Khoo	30	Purchasing
17	116	Shelli Baida	30	Purchasing
18	117	Sigal Tobias	30	Purchasing
19	118	Guy Himuro	30	Purchasing
20	119	Karen Colmenares	30	Purchasing

3. View 실습

```
CREATE OR REPLACE VIEW emp_dept_v AS
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_names,
       b.department_id ,b.department_name
  FROM employees a,
       departments b
 WHERE a.department_id = b.department_id
 ORDER BY 1;

SELECT *
  FROM emp_dept_v;
```

```
View EMP_DEPT_V이(가) 생성되었습니다.
```

	EMPLOYEE_ID	EMP_NAMES	DEPARTMENT_ID	DEPARTMENT_NAME
1	100	Steven King	90	Executive
2	101	Neena Kochhar	90	Executive
3	102	Lex De Haan	90	Executive
4	103	Alexander Hunold	60	IT
5	104	Bruce Ernst	60	IT
6	105	David Austin	60	IT
7	106	Valli Pataballa	60	IT
8	107	Diana Lorentz	60	IT
9	108	Nancy Greenberg	100	Finance
10	109	Daniel Faviet	100	Finance
11	110	John Chen	100	Finance
12	111	Ismael Sciarra	100	Finance
13	112	Jose Manuel Urman	100	Finance
14	113	Luis Popp	100	Finance
15	114	Den Raphaely	30	Purchasing
16	115	Alexander Khoo	30	Purchasing
17	116	Shelli Baida	30	Purchasing
18	117	Sigal Tobias	30	Purchasing
19	118	Guy Himuro	30	Purchasing
20	119	Karen Colmenares	30	Purchasing

3. View 실습

```
CREATE OR REPLACE VIEW emp_dept_v AS
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_names,
       a.salary,
       b.department_id ,b.department_name
  FROM employees a,
       departments b
 WHERE a.department_id = b.department_id
 ORDER BY 1;

SELECT *
  FROM emp_dept_v;
```

```
View EMP_DEPT_V이(가) 생성되었습니다.
```

	EMPLOYEE_ID	EMP_NAMES	SALARY	DEPARTMENT_ID	DEPARTMENT_NAME
1	100	Steven King	24000	90	Executive
2	101	Neena Kochhar	17000	90	Executive
3	102	Lex De Haan	17000	90	Executive
4	103	Alexander Hunold	9000	60	IT
5	104	Bruce Ernst	6000	60	IT
6	105	David Austin	4800	60	IT
7	106	Valli Pataballa	4800	60	IT
8	107	Diana Lorentz	4200	60	IT
9	108	Nancy Greenberg	12008	100	Finance
10	109	Daniel Faviet	9000	100	Finance
11	110	John Chen	8200	100	Finance
12	111	Ismael Sciarra	7700	100	Finance
13	112	Jose Manuel Urman	7800	100	Finance
14	113	Luis Popp	6900	100	Finance
15	114	Den Raphaely	11000	30	Purchasing
16	115	Alexander Khoo	3100	30	Purchasing
17	116	Shelli Baida	2900	30	Purchasing
18	117	Sigal Tobias	2800	30	Purchasing
19	118	Guy Himuro	2600	30	Purchasing
20	119	Karen Colmenares	2500	30	Purchasing

3. View 실습

· 시나리오

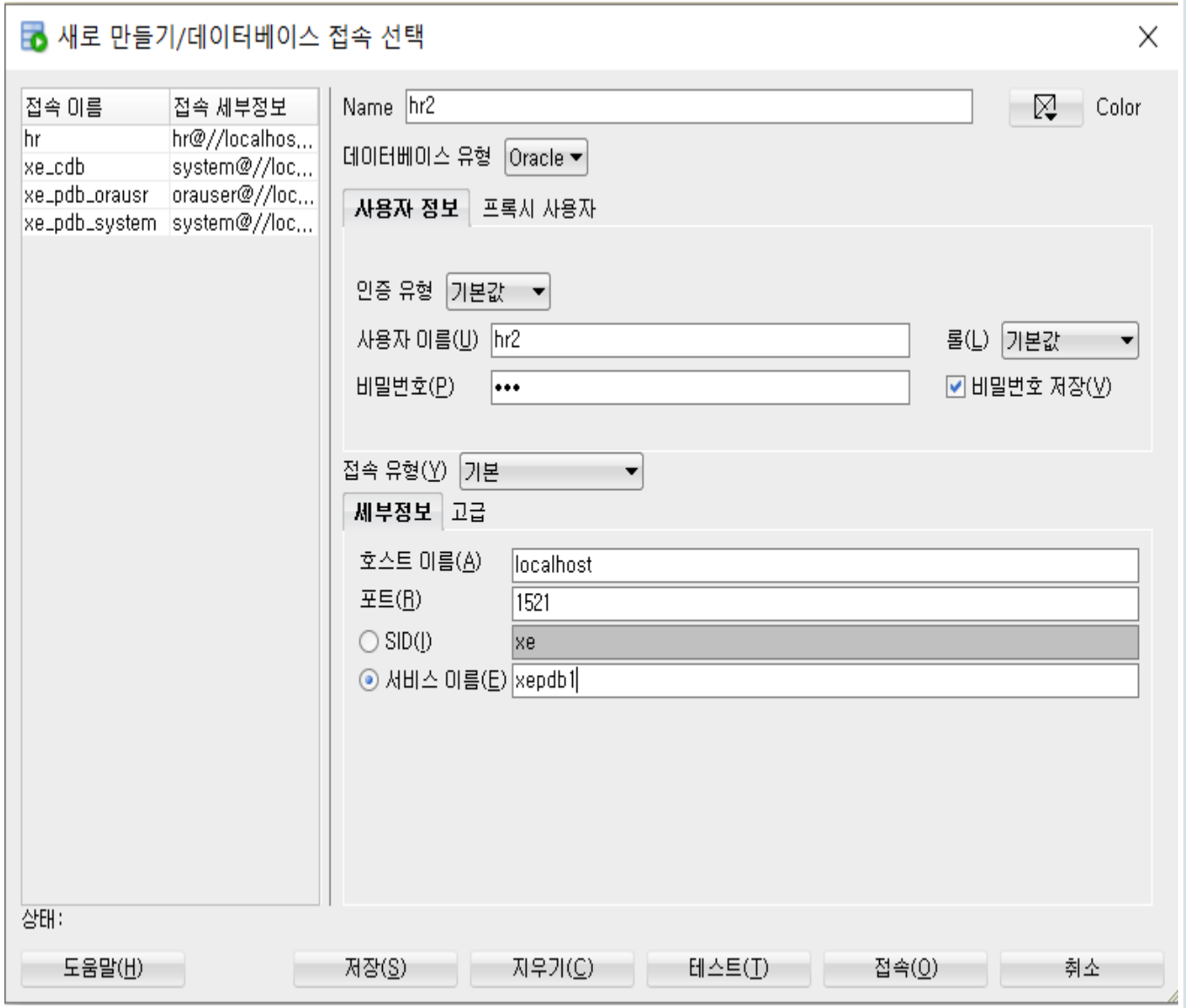
- HR2 사용자 생성 (ORAUSER로 로그인)

  CREATE USER hr2 IDENTIFIED BY hr2;

- HR2에 접속 권한 설정

  GRANT CREATE SESSION TO hr2;

- SQL Developer에서 hr2 사용자 접속 생성



3. View 실습

· 시나리오

- HR2 사용자가 HR 스키마에 있는 사원과 부서 정보를 보고싶다고 요청
- 하지만 사원의 급여는 개인정보 이므로 HR 이외의 사용자에게는 공개하지 못함
- 어떻게 해야 할까?

· 가능한 시나리오

- 사원, 부서 정보를 hr2에 공개하기 위해서는 employees, departments 테이블 조회 권한 부여
- 하지만 급여(salary)만 비공개로 하기는 불가능
- 따라서 사원과 부서 기본 정보만 조회하는 뷰를 만들고, 이 뷰의 조회권한을 hr2에게 부여

3. View 실습

```
CREATE OR REPLACE VIEW emp_dept_v2 AS
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_names,
       b.department_id ,b.department_name
  FROM employees a,
       departments b
 WHERE a.department_id = b.department_id
 ORDER BY 1;

SELECT *
    FROM emp_dept_v2;
```

	EMPLOYEE_ID	EMP_NAMES	DEPARTMENT_ID	DEPARTMENT_NAME
1	100	Steven King	90	Executive
2	101	Neena Kochhar	90	Executive
3	102	Lex De Haan	90	Executive
4	103	Alexander Hunold	60	IT
5	104	Bruce Ernst	60	IT
6	105	David Austin	60	IT
7	106	Valli Pataballa	60	IT
8	107	Diana Lorentz	60	IT
9	108	Nancy Greenberg	100	Finance
10	109	Daniel Faviet	100	Finance
11	110	John Chen	100	Finance
12	111	Ismael Sciarra	100	Finance
13	112	Jose Manuel Urman	100	Finance
14	113	Luis Popp	100	Finance
15	114	Den Raphaely	30	Purchasing
16	115	Alexander Khoo	30	Purchasing
17	116	Shelli Baida	30	Purchasing

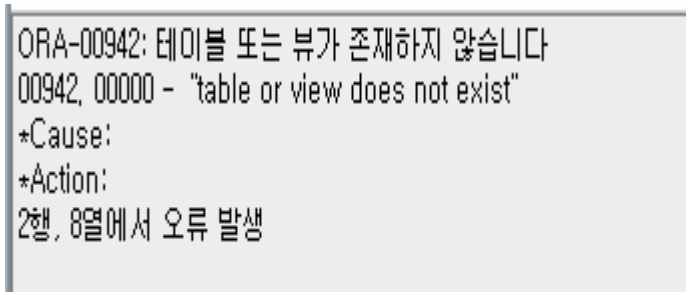
3. View 실습

· hr 사용자는 emp_dept_v2 뷰의 조회권한을 hr2에게 부여

```
GRANT SELECT ON emp_dept_v2 TO hr2;
```

· hr2 사용자로 로그인 한 후, emp_dept_v2 조회

```
SELECT *
   FROM emp_dept_v2;
```



3. View 실습

· 테이블을 포함한 모든 객체를 참조하기 위해서는 **소유자명.객체명** 형태로 사용해야 함

  예) hr.employees, hr.departments, …

· 다만 해당 객체 소유자로 접속한 경우에는 소유자명 생략 가능

· hr이 생성한 emp_dept_v2 뷰를 hr이 아닌 다른 사용자가 참조하려면 소유자명.객체명 으로 접근

  - SELECT * FROM emp_dept_v2 **(X)**

  - SELECT * FROM hr.emp_dept_v2 **(O)**

3. View 실습

· hr2 사용자가 emp_dept_v2 뷰 조회

```
SELECT *
    FROM hr.emp_dept_v2;
```

	EMPLOYEE_ID	EMP_NAMES	DEPARTMENT_ID	DEPARTMENT_NAME
1	100	Steven King	90	Executive
2	101	Neena Kochhar	90	Executive
3	102	Lex De Haan	90	Executive
4	103	Alexander Hunold	60	IT
5	104	Bruce Ernst	60	IT
6	105	David Austin	60	IT
7	106	Valli Pataballa	60	IT
8	107	Diana Lorentz	60	IT
9	108	Nancy Greenberg	100	Finance
10	109	Daniel Faviet	100	Finance
11	110	John Chen	100	Finance
12	111	Ismael Sciarra	100	Finance
13	112	Jose Manuel Urman	100	Finance
14	113	Luis Popp	100	Finance
15	114	Den Raphaely	30	Purchasing
16	115	Alexander Khoo	30	Purchasing
17	116	Shelli Baida	30	Purchasing
18	117	Sigal Tobias	30	Purchasing
19	118	Guy Himuro	30	Purchasing
20	119	Karen Colmenares	30	Purchasing

3. View 실습

· hr2 사용자는 hr의 다른 테이블 접근 불가

```
SELECT *
    FROM hr.employees;
```

```
ORA-00942: 테이블 또는 뷰가 존재하지 않습니다
00942, 00000 - "table or view does not exist"
*Cause:
*Action:
2행, 11열에서 오류 발생
```


4. 데이터 딕셔너리(Data Dictionary)

· 오라클에서 제공하는 데이터베이스 객체(사용자, 테이블, 뷰 등)에 대한 메타정보를 담은 뷰

· 접두어로 용도 구분

- DBA : 데이터베이스 관리자의 뷰 (모든 사용자 스키마가 포함됨)
- ALL : 현재 로그인한 사용자가 접근할 수 있는 뷰
- USER : 현재 로그인한 사용자가 소유자인 데이터베이스 객체

4. 데이터 딕셔너리(Data Dictionary)

· 주요 사용자 객체 정보 뷰

- USER_OBJECTS : 모든 객체 정보
- USER_TABLES : 테이블 정보
- USER_INDEXES : 인덱스 정보
- USER_CONSTRAINTS : 제약조건
- USER_TAB_COLS : 테이블과 해당 컬럼 정보
- USER_VIEWS : 뷰 정보

4. 데이터 딕셔너리 실습

```
SELECT *
    FROM user_objects;
```

	OBJECT_NAME	SUBOBJECT_NAME	OBJECT_ID	DATA_OBJECT_ID	OBJECT_TYPE
1	REGIONS	(null)	73356	73356	TABLE
2	REG_ID_PK	(null)	73357	73357	INDEX
3	COUNTRIES	(null)	73358	(null)	TABLE
4	COUNTRY_C_ID_PK	(null)	73359	73359	INDEX
5	LOCATIONS	(null)	73360	73360	TABLE
6	LOC_ID_PK	(null)	73361	73361	INDEX
7	LOCATIONS_SEQ	(null)	73362	(null)	SEQUENCE
8	DEPARTMENTS	(null)	73363	73363	TABLE
9	DEPT_ID_PK	(null)	73364	73364	INDEX
10	DEPARTMENTS_SEQ	(null)	73365	(null)	SEQUENCE
11	JOBS	(null)	73366	73366	TABLE
12	JOB_ID_PK	(null)	73367	73367	INDEX
13	EMPLOYEES	(null)	73368	73368	TABLE
14	EMP_EMAIL_UK	(null)	73369	73369	INDEX
15	EMP_EMP_ID_PK	(null)	73370	73370	INDEX
16	EMPLOYEES_SEQ	(null)	73371	(null)	SEQUENCE
17	JOB_HISTORY	(null)	73372	73372	TABLE
18	JHIST_EMP_ID_ST_DATE_PK	(null)	73373	73373	INDEX
19	EMP_DETAILS_VIEW	(null)	73374	(null)	VIEW
20	EMP_DEPARTMENT_IX	(null)	73375	73375	INDEX
21	EMP_JOB_IX	(null)	73376	73376	INDEX
22	EMP_MANAGER_IX	(null)	73377	73377	INDEX
23	EMP_NAME_IX	(null)	73378	73378	INDEX
24	DEPT_LOCATION_IX	(null)	73379	73379	INDEX
25	JHIST_JOB_IX	(null)	73380	73380	INDEX
26	JHIST_EMPLOYEE_IX	(null)	73381	73381	INDEX

4. 데이터 딕셔너리 실습

```
SELECT *
    FROM user_tables
ORDER BY 1;
```

	TABLE_NAME	TABLESPACE_NAME	CLUSTER_NAME	IOT_NAME	STATUS
1	BUDGET_TABLE	SYSAUX	(null)	(null)	VALID
2	COUNTRIES	(null)	(null)	(null)	VALID
3	DEPARTMENTS	SYSAUX	(null)	(null)	VALID
4	DEPT_MGR	SYSAUX	(null)	(null)	VALID
5	EMP	SYSAUX	(null)	(null)	VALID
6	EMP1	SYSAUX	(null)	(null)	VALID
7	EMP2	SYSAUX	(null)	(null)	VALID
8	EMP3	SYSAUX	(null)	(null)	VALID
9	EMPLOYEES	SYSAUX	(null)	(null)	VALID
10	EMP_INFO1	SYSAUX	(null)	(null)	VALID
11	EMP_TRAN	SYSAUX	(null)	(null)	VALID
12	GROUPBYMULTIPLY	SYSAUX	(null)	(null)	VALID
13	HONGS	SYSAUX	(null)	(null)	VALID
14	INDEX_TEST	SYSAUX	(null)	(null)	VALID
15	JOBS	SYSAUX	(null)	(null)	VALID
16	JOB_HISTORY	SYSAUX	(null)	(null)	VALID
17	LOCATIONS	SYSAUX	(null)	(null)	VALID
18	REGIONS	SYSAUX	(null)	(null)	VALID
19	SALE_TABLE	SYSAUX	(null)	(null)	VALID
20	SCORE_COL_TABLE	SYSAUX	(null)	(null)	VALID
21	SCORE_TABLE	SYSAUX	(null)	(null)	VALID
22	TEST_SCORE	SYSAUX	(null)	(null)	VALID

4. 데이터 딕셔너리 실습

```
SELECT *
    FROM user_tables
ORDER BY 1;
```

	TABLE_NAME	TABLESPACE_NAME	CLUSTER_NAME	IOT_NAME	STATUS
1	BUDGET_TABLE	SYSAUX	(null)	(null)	VALID
2	COUNTRIES	(null)	(null)	(null)	VALID
3	DEPARTMENTS	SYSAUX	(null)	(null)	VALID
4	DEPT_MGR	SYSAUX	(null)	(null)	VALID
5	EMP	SYSAUX	(null)	(null)	VALID
6	EMP1	SYSAUX	(null)	(null)	VALID
7	EMP2	SYSAUX	(null)	(null)	VALID
8	EMP3	SYSAUX	(null)	(null)	VALID
9	EMPLOYEES	SYSAUX	(null)	(null)	VALID
10	EMP_INFO1	SYSAUX	(null)	(null)	VALID
11	EMP_TRAN	SYSAUX	(null)	(null)	VALID
12	GROUPBYMULTIPLY	SYSAUX	(null)	(null)	VALID
13	HONGS	SYSAUX	(null)	(null)	VALID
14	INDEX_TEST	SYSAUX	(null)	(null)	VALID
15	JOBS	SYSAUX	(null)	(null)	VALID
16	JOB_HISTORY	SYSAUX	(null)	(null)	VALID
17	LOCATIONS	SYSAUX	(null)	(null)	VALID
18	REGIONS	SYSAUX	(null)	(null)	VALID
19	SALE_TABLE	SYSAUX	(null)	(null)	VALID
20	SCORE_COL_TABLE	SYSAUX	(null)	(null)	VALID
21	SCORE_TABLE	SYSAUX	(null)	(null)	VALID
22	TEST_SCORE	SYSAUX	(null)	(null)	VALID

4. 데이터 딕셔너리 실습

```
SELECT *
    FROM user_indexes
ORDER BY 1;
```

	INDEX_NAME	INDEX_TYPE	TABLE_OWNER	TABLE_NAME	TABLE_TYPE	UNIQUENESS
1	COUNTRY_C_ID_PK	IOT - TOP	HR	COUNTRIES	TABLE	UNIQUE
2	DEPT_ID_PK	NORMAL	HR	DEPARTMENTS	TABLE	UNIQUE
3	DEPT_LOCATION_IX	NORMAL	HR	DEPARTMENTS	TABLE	NONUNIQUE
4	DEPT_MGR_PK	NORMAL	HR	DEPT_MGR	TABLE	UNIQUE
5	EMP_PK	NORMAL	HR	EMP	TABLE	UNIQUE
6	EMP1_PK	NORMAL	HR	EMP1	TABLE	UNIQUE
7	EMP2_PK	NORMAL	HR	EMP2	TABLE	UNIQUE
8	EMP3_PK	NORMAL	HR	EMP3	TABLE	UNIQUE
9	EMP_EMAIL_UK	NORMAL	HR	EMPLOYEES	TABLE	UNIQUE
10	EMP_EMP_ID_PK	NORMAL	HR	EMPLOYEES	TABLE	UNIQUE
11	EMP_DEPARTMENT_IX	NORMAL	HR	EMPLOYEES	TABLE	NONUNIQUE
12	EMP_JOB_IX	NORMAL	HR	EMPLOYEES	TABLE	NONUNIQUE
13	EMP_MANAGER_IX	NORMAL	HR	EMPLOYEES	TABLE	NONUNIQUE
14	EMP_NAME_IX	NORMAL	HR	EMPLOYEES	TABLE	NONUNIQUE
15	INDEX_TEST1	NORMAL	HR	INDEX_TEST	TABLE	NONUNIQUE
16	JOB_ID_PK	NORMAL	HR	JOBS	TABLE	UNIQUE
17	JHIST_EMP_ID_ST_DATE_PK	NORMAL	HR	JOB_HISTORY	TABLE	UNIQUE
18	JHIST_JOB_IX	NORMAL	HR	JOB_HISTORY	TABLE	NONUNIQUE
19	JHIST_EMPLOYEE_IX	NORMAL	HR	JOB_HISTORY	TABLE	NONUNIQUE
20	JHIST_DEPARTMENT_IX	NORMAL	HR	JOB_HISTORY	TABLE	NONUNIQUE
21	LOC_ID_PK	NORMAL	HR	LOCATIONS	TABLE	UNIQUE
22	LOC_CITY_IX	NORMAL	HR	LOCATIONS	TABLE	NONUNIQUE
23	LOC_STATE_PROVINCE_IX	NORMAL	HR	LOCATIONS	TABLE	NONUNIQUE
24	LOC_COUNTRY_IX	NORMAL	HR	LOCATIONS	TABLE	NONUNIQUE
25	REG_ID_PK	NORMAL	HR	REGIONS	TABLE	UNIQUE

4. 데이터 딕셔너리 실습

```
SELECT *
    FROM user_constraints
ORDER BY 1;
```

	OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION
1	HR	SYS_C007344	O	EMP_DETAILS_VIEW	(null)
2	HR	COUNTR_REG_FK	R	COUNTRIES	(null)
3	HR	LOC_C_ID_FK	R	LOCATIONS	(null)
4	HR	DEPT_LOC_FK	R	DEPARTMENTS	(null)
5	HR	EMP_DEPT_FK	R	EMPLOYEES	(null)
6	HR	EMP_JOB_FK	R	EMPLOYEES	(null)
7	HR	EMP_MANAGER_FK	R	EMPLOYEES	(null)
8	HR	DEPT_MGR_FK	R	DEPARTMENTS	(null)
9	HR	JHIST_JOB_FK	R	JOB_HISTORY	(null)
10	HR	JHIST_EMP_FK	R	JOB_HISTORY	(null)
11	HR	JHIST_DEPT_FK	R	JOB_HISTORY	(null)
12	HR	REGION_ID_NN	C	REGIONS	"REGION_ID" IS NOT NULL
13	HR	REG_ID_PK	P	REGIONS	(null)
14	HR	COUNTRY_ID_NN	C	COUNTRIES	"COUNTRY_ID" IS NOT NULL
15	HR	COUNTRY_C_ID_PK	P	COUNTRIES	(null)
16	HR	LOC_CITY_NN	C	LOCATIONS	"CITY" IS NOT NULL
17	HR	LOC_ID_PK	P	LOCATIONS	(null)
18	HR	DEPT_NAME_NN	C	DEPARTMENTS	"DEPARTMENT_NAME" IS NOT NULL

4. 데이터 딕셔너리 실습

```
SELECT *
    FROM user_tab_cols
ORDER BY table_name,
         column_id;
```

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	DATA_TYPE_MOD	DATA_TYPE_OWNER	DATA_LENGTH	DATA_PRECISION	DATA_SCALE	NULLABLE	COLUMN_ID
1	BUDGET_TABLE	YEARMON	VARCHAR2	(null)	(null)	6	(null)	(null)	Y	1
2	BUDGET_TABLE	BUDGET_AMT	NUMBER	(null)	(null)	22	(null)	(null)	Y	2
3	COUNTRIES	COUNTRY_ID	CHAR	(null)	(null)	2	(null)	(null)	N	1
4	COUNTRIES	COUNTRY_NAME	VARCHAR2	(null)	(null)	40	(null)	(null)	Y	2
5	COUNTRIES	REGION_ID	NUMBER	(null)	(null)	22	(null)	(null)	Y	3
6	DEPARTMENTS	DEPARTMENT_ID	NUMBER	(null)	(null)	22	4	0	N	1
7	DEPARTMENTS	DEPARTMENT_NAME	VARCHAR2	(null)	(null)	30	(null)	(null)	N	2
8	DEPARTMENTS	MANAGER_ID	NUMBER	(null)	(null)	22	6	0	Y	3
9	DEPARTMENTS	LOCATION_ID	NUMBER	(null)	(null)	22	4	0	Y	4
10	DEPT_MGR	DEPARTMENT_ID	NUMBER	(null)	(null)	22	4	0	N	1
11	DEPT_MGR	DEPARTMENT_NAME	VARCHAR2	(null)	(null)	30	(null)	(null)	N	2
12	DEPT_MGR	MANAGER_ID	NUMBER	(null)	(null)	22	6	0	Y	3
13	DEPT_MGR	LOCATION_ID	NUMBER	(null)	(null)	22	4	0	Y	4
14	EMP	EMP_NO	VARCHAR2	(null)	(null)	30	(null)	(null)	N	1
15	EMP	EMP_NAME	VARCHAR2	(null)	(null)	80	(null)	(null)	N	2
16	EMP	SALARY	NUMBER	(null)	(null)	22	(null)	(null)	Y	3
17	EMP	HIRE_DATE	DATE	(null)	(null)	7	(null)	(null)	Y	4
18	EMP1	EMP_NO	VARCHAR2	(null)	(null)	30	(null)	(null)	N	1
19	EMP1	EMP_NAME	VARCHAR2	(null)	(null)	80	(null)	(null)	N	2
20	EMP1	SALARY	NUMBER	(null)	(null)	22	(null)	(null)	Y	3
21	EMP1	HIRE_DATE	DATE	(null)	(null)	7	(null)	(null)	Y	4
22	EMP1	DEPT_ID	NUMBER	(null)	(null)	22	(null)	(null)	Y	5
23	EMP2	EMP_NO	VARCHAR2	(null)	(null)	30	(null)	(null)	N	1
24	EMP2	EMP_NAME	VARCHAR2	(null)	(null)	80	(null)	(null)	N	2
25	EMP2	SALARY	NUMBER	(null)	(null)	22	(null)	(null)	Y	3
26	EMP2	HIRE_DATE	DATE	(null)	(null)	7	(null)	(null)	Y	4
27	EMP2	DEPT_ID	NUMBER	(null)	(null)	22	(null)	(null)	Y	5
28	EMP3	EMP_NO	VARCHAR2	(null)	(null)	30	(null)	(null)	N	1
29	EMP3	EMP_NAME	VARCHAR2	(null)	(null)	80	(null)	(null)	N	2
30	EMP3	SALARY	NUMBER	(null)	(null)	22	(null)	(null)	Y	3

4. 데이터 딕셔너리 실습

```
SELECT ',a.' || column_name
  FROM user_tab_cols
 WHERE table_name = 'EMPLOYEES'
 ORDER BY column_id ;
```

	',A.'||COLUMN_NAME
1	,a.EMPLOYEE_ID
2	,a.FIRST_NAME
3	,a.LAST_NAME
4	,a.EMAIL
5	,a.PHONE_NUMBER
6	,a.HIRE_DATE
7	,a.JOB_ID
8	,a.SALARY
9	,a.COMMISSION_PCT
10	,a.MANAGER_ID
11	,a.DEPARTMENT_ID

학습정리

· SQL에서는 COMMIT 과 ROLLBACK 문을 사용해 트랜잭션 처리를 한다.

· 뷰는 한 개 이상의 다른 테이블이나 다른 뷰를 조회하는 쿼리문으로 만든 객체이다.

· 뷰는 테이블처럼 사용할 수 있지만, 데이터를 저장하고 있지 않다.

· 데이터 딕셔너리를 통해 데이터베이스 객체에 대한 다양한 정보를 조회할 수 있다.

Quiz

1. 특정 테이블에 데이터를 INSERT 한 다음, 다시 특정 조건에 따라 어느 컬럼 값을 UPDATE 해야 하는데, 올바른 트랜잭션 처리를 하려면 INSERT 문 실행 후 COMMIT 을 실행하고 다시 UPDATE 문을 실행하고 COMMIT 문을 실행해야 합니다.

Quiz

2. 복잡하게 만들어진 쿼리를 수행하는 것보다 이 쿼리를 기준으로 뷰를 만들고 이 뷰를 조회하는 것이 조회 성능 상 더 유리합니다. 이 말이 맞을까요?