7-2. 트랜잭션 처리, Merge문, 뷰, 데이터 딕셔너리

홍형경 chariehong@gmail.com 2021.06

1. 트랜잭션 (Transaction) 처리

- 트랜잭션은 거래라는 뜻



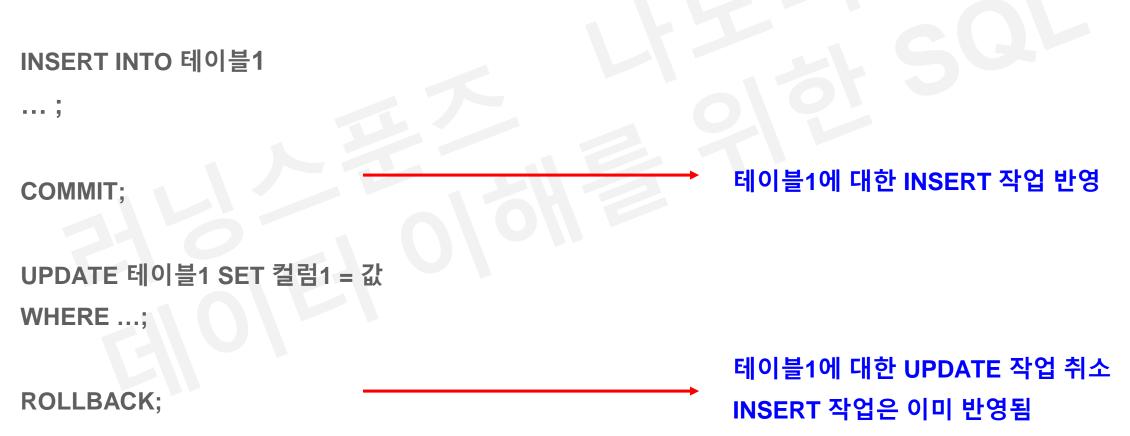
- 오류가 났을 경우는 거래 자체가 없었던 것으로 처리
- 입금 계좌에 돈이 확인된 다음에야 거래를 성사시킴

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



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		
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	
1	103	Alexander Hunold	9000 2006-01-03 00:0	0:00 60
2	104	Bruce Ernst	6000 2007-05-21 00:0	0:00 60
3	105	David Austin	4800 2005-06-25 00:0	0:00 60
4	106	Valli Pataballa	4800 2006-02-05 00:0	0:00 60
5	107	Diana Lorentz	4200 2007-02-07 00:0	0:00 60
6	108	Nancy Greenberg	12008 2002-08-17 00:0	0:00 100
7	109	Daniel Faviet	9000 2002-08-16 00:0	0:00 100
8	111	Ismael Sciarra	7700 2005-09-30 00:0	0:00 100
9	113	Luis Popp	6900 2007-12-07 00:0	0:00 100
10	114	Den Raphaely	11000 2002-12-07 00:0	0:00 30
11	115	Alexander Khoo	3100 2003-05-18 00:0	0:00 30
12	116	Shelli Baida	2900 2005-12-24 00:0	0:00 30
13	117	Sigal Tobias	2800 2005-07-24 00:0	0:00 30

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

⊕ EMP_NO	EMP_NAME		⊕ HIRE_DATE		
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
0 1 1 1	Tambal Caianna	7700	2005 00 20	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, ...);
```

- 테이블 복제

CREATE TABLE dept_mgr AS

SELECT *

FROM departments;

	DEPARTME	⊕ DEPARTMENT_NAME		
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;

```
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;
```

이(가) 병합되었습니다.

21	210 IT Support	(nu⊥⊥)	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 <mark>경리부(Merge)</mark>	(null)	(null)

```
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);
SELECT*
 FROM dept_mgr;
```

이(가) 병합되었습니다.

21	210 IT Support	(nu⊥⊥)	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 <mark>영업부(Merge)2</mark>	(null)	(null)
28	285 <mark>경리부(Merge)2</mark>	(null)	(null)

```
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);
SELECT*
 FROM dept_mgr;
```

이(가) 병합되었습니다.

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	280영업부(Merge)3	(null)	(null)
28	285 <mark>경리부(Merge)2</mark>	(null)	(null)
29	290 <mark>전산팀(Merge)</mark>	(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;

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**;

			⊕ 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

```
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.

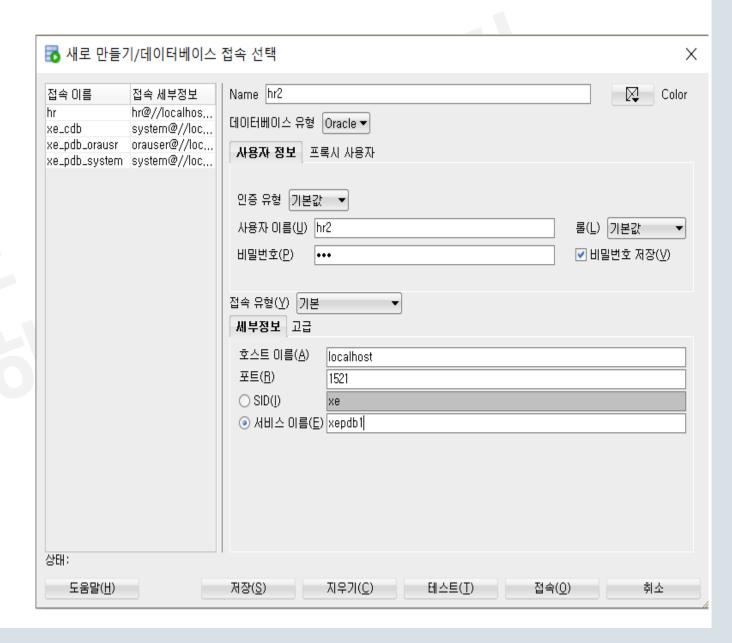
	⊕ EMPLOYEE_ID	EMP_NAMES	⊕ DEPARTMENT_ID	DEPARTMENT_NAME
1	100	Steven King	90	Executive
2	101	Neena Kochhar	90	Executive
3	102	Lex De Haan	90	Executive
4	103	Alexander Hunold	60	IT
5	104	Bruce Ernst	60	IT
6	105	David Austin	60	IT
- 7	106	Valli Pataballa	60	IT
8	107	Diana Lorentz	60	IT
9	108	Nancy Greenberg	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

```
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.

		⊕ EMP_NAMES			
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

- 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



- 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

```
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_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

· 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;

ORA-00942: 테이블 또는 뷰가 존재하지 않습니다 |00942, 00000 - "table or view does not exist"

|∗Cause:

*Action:

2행, 8열에서 오류 발생

- 테이블을 포함한 모든 객체를 참조하기 위해서는 소유자명.객체명 형태로 사용해야 함예) 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)

· hr2 사용자가 emp_dept_v2 뷰 조회

SELECT *

FROM hr.emp_dept_v2;

		EMP_NAMES		
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

· 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

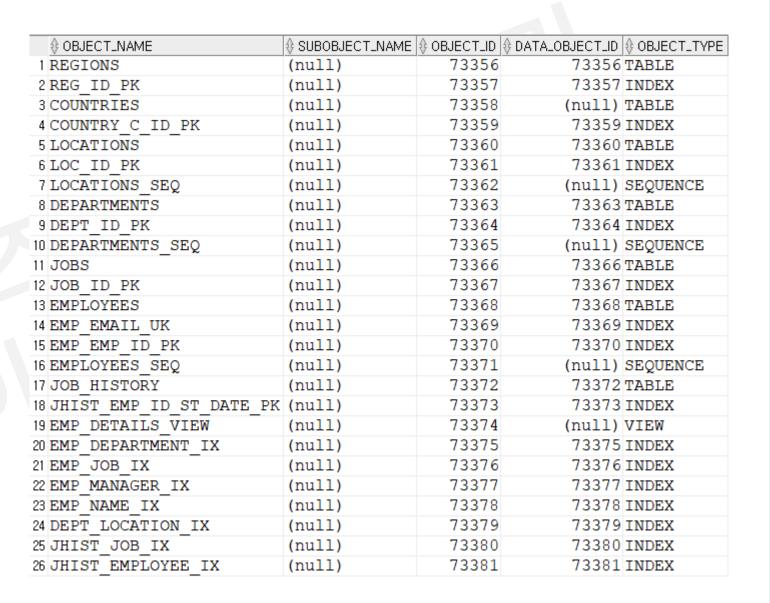
- USER_CONSTRAINTS : 제약조건

- USER_TAB_COLS: 테이블과 해당 컬럼 정보

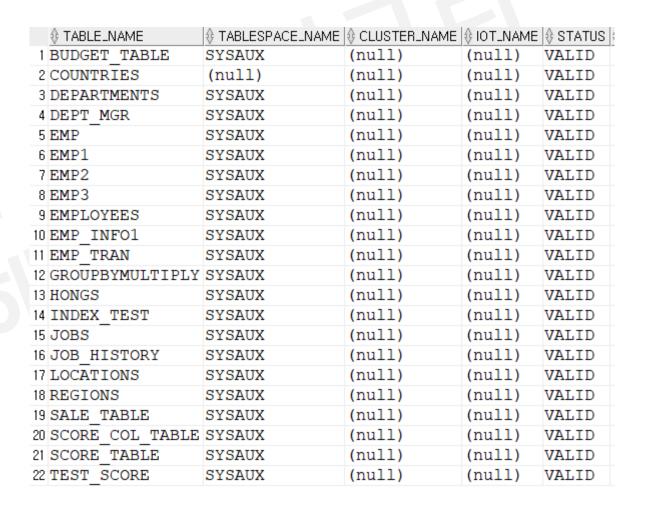
- USER_VIEWS : 뷰 정보

SELECT *

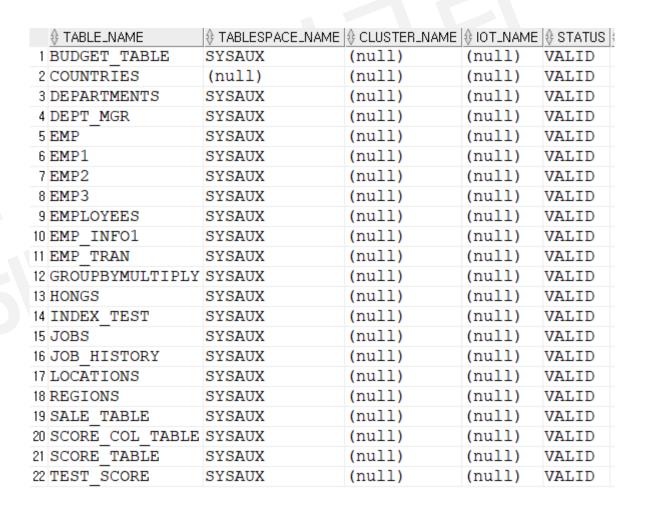
FROM user_objects;



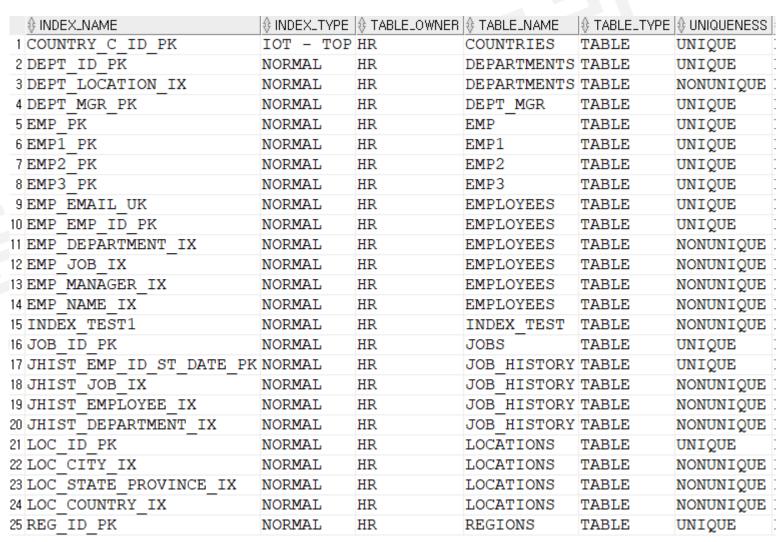
SELECT * FROM user_tables **ORDER BY 1**;



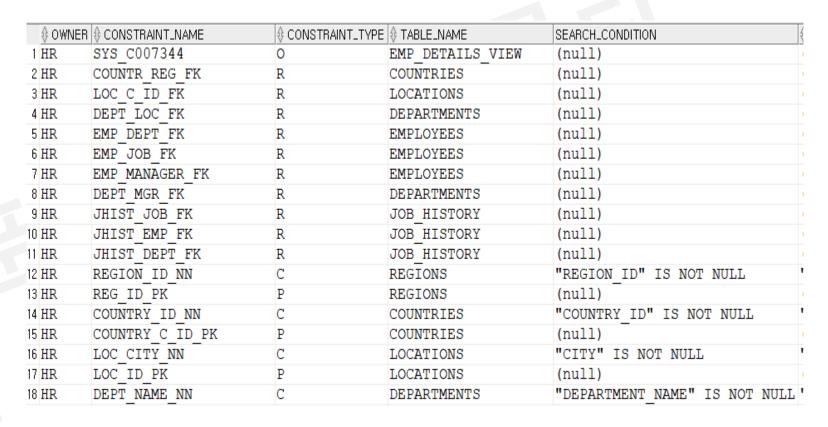
SELECT * FROM user_tables **ORDER BY 1**;



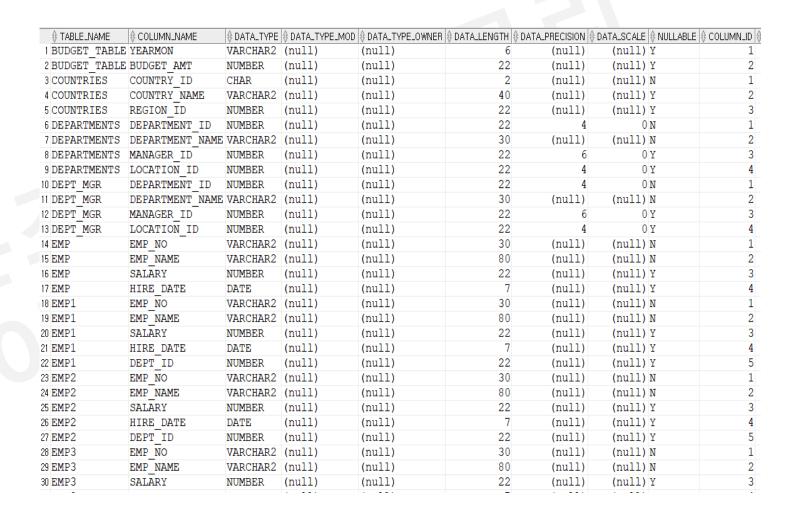
SELECT * FROM user_indexes **ORDER BY 1**;



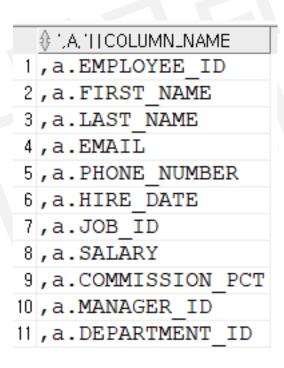
SELECT * FROM user_constraints **ORDER BY 1**;



SELECT * FROM user_tab_cols ORDER BY table_name, column_id;



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