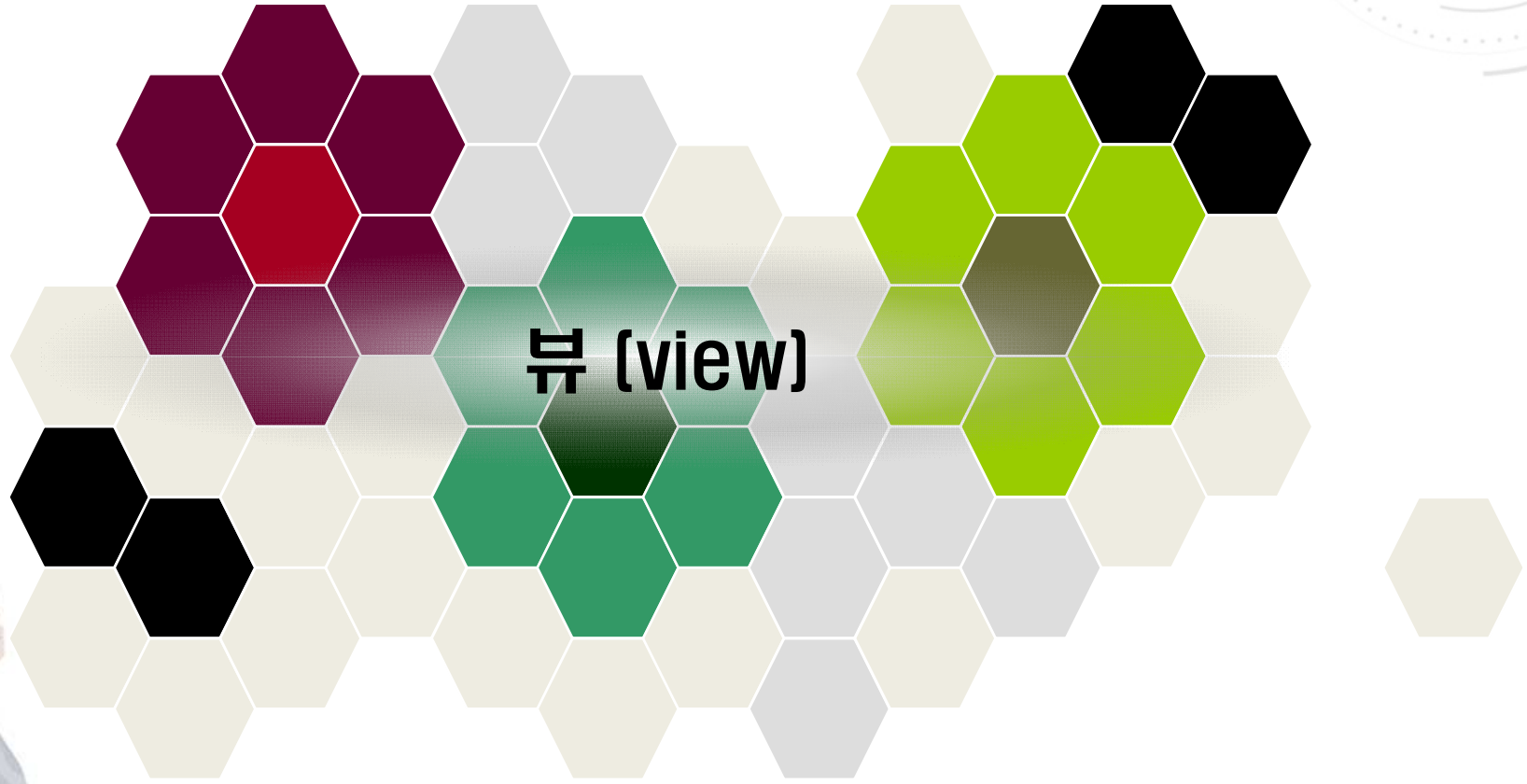
뷰 (view)



뷰 (view)

1. 뷰(View)란?

- 테이블 또는 다른 뷰를 기반으로 하는 논리 테이블
- 데이터의 논리적인 부분 집합 또는 조합

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81/11/17	5000		10
7698	BLAKE	MANAGER	7839	81/05/01	2850		30
7782	CLARK	MANAGER	7839	81/06/09	2450		10
7566	JONES	MANAGER	7839	81/04/02	2975		20
7902	FORD	ANALYST	7566	81/10/03	3000		20
7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30
7521	WARD	SALESMAN	7698	81/02/22	1250	500	30
7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30
7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30
7900	JAMES	CLERK	7698	81/10/03	950		30
7934	MILLER	CLERK	7782	82/01/23	1300		10
7566			7566	82/10/09	3000		20
7902			7902	80/10/17	800		20
7788			7788	83/01/12	1100		20

EMP

EMPNO	ENAME	SAL
7839	KING	5000
7782	CLARK	2450
7934	MILLER	1300

emp_v10

뷰 (view)

2. 뷰 사용 목적

- 데이터 액세스를 제한하기 위해
- 복잡한 질의를 쉽게 작성하기 위해
- 데이터 독립성을 제공하기 위해
- 동일한 데이터로부터 다양한 결과를 얻기 위해

뷰 (view)

3. 뷰 생성과 뷰를 통한 데이터 조회

3.1 뷰 생성

- **CREATE VIEW** 문장 내에서 서브쿼리를 내장
- 뷰를 정의하는 서브쿼리는 조인, 그룹, 서브쿼리 등의 복합 **SELECT** 구문을 포함할 수 있음
- 뷰를 정의하는 서브쿼리는 **ORDER BY** 절을 포함할 수 없음
- **WITH CHECK OPTION**을 사용하여 생성한 뷰의 제약조건 이름을 지정하지 않으면 **SYS_C*n*** 형식의 기본 이름이 지정

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
  [(alias[, alias]...)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

뷰 (view)

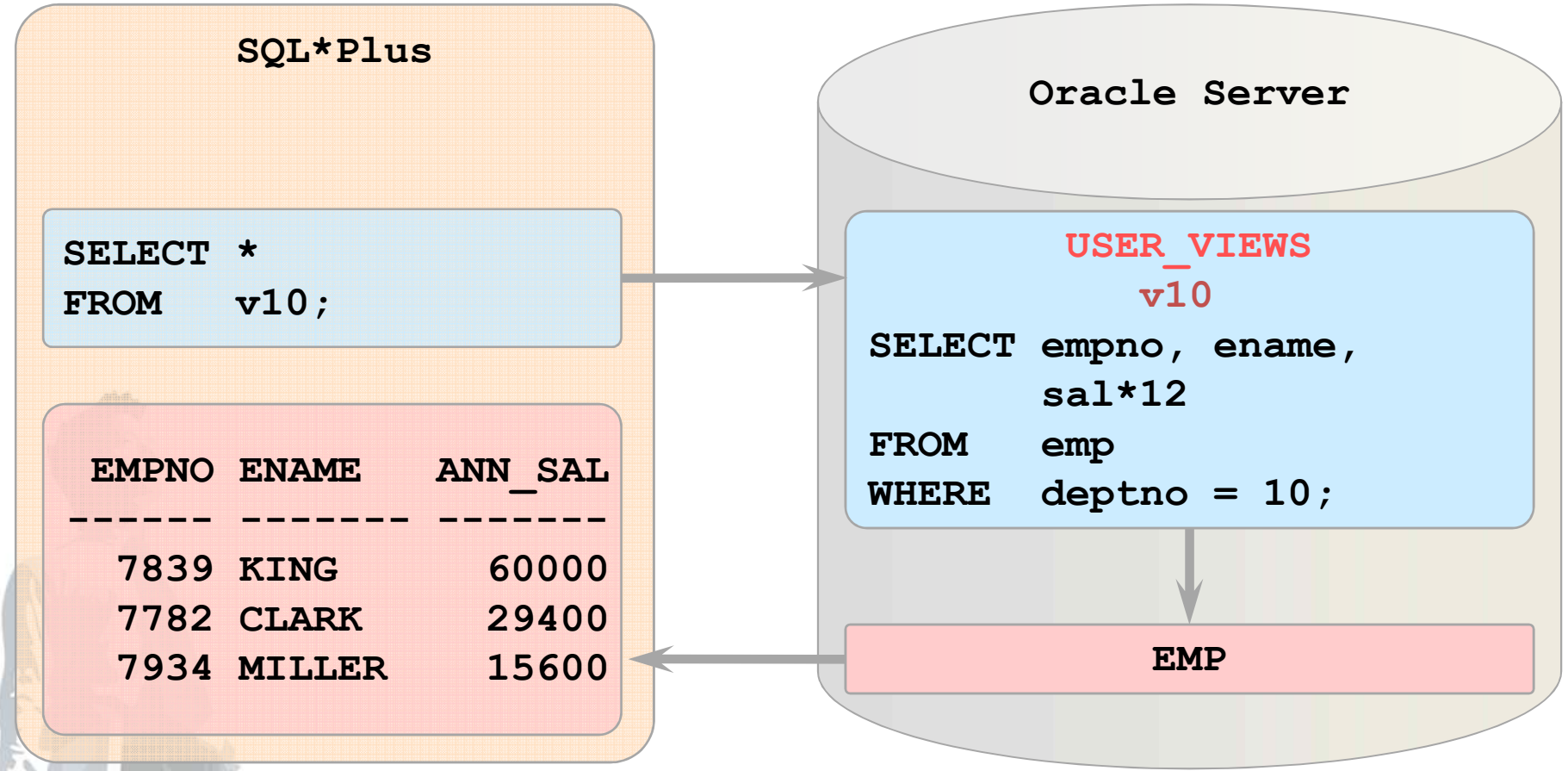
3.1 뷰 생성 (계속)

```
SQL> CREATE VIEW v10 (empno, ename, ann_sal)
  2  AS SELECT empno, ename, sal*12
  3     FROM   emp
  4     WHERE  deptno = 10;

뷰가 생성되었습니다.
```

```
SQL> CREATE VIEW v10_1
  2  AS SELECT empno, ename, (sal*12) AS ann_sal
  3     FROM   emp
  4     WHERE  deptno = 10;

뷰가 생성되었습니다.
```

뷰 (view)

3.2 뷰 질의


SQL*Plus
SELECT *
FROM v10;
EMPNO ENAME ANN_SAL
7839 KING 60000
7782 CLARK 29400
7934 MILLER 15600
Oracle Server
USER_VIEWS
v10
SELECT empno, ename,
sal*12
FROM emp
WHERE deptno = 10;
EMP

뷰 (view)

3.2 뷰 질의 (계속)

```
SQL> SELECT *
  2  FROM   v10;
```

```
 EMPNO ENAME                    ANN_SAL
------ -------------------- -----------
  7839 KING                       60000
  7782 CLARK                      29400
  7934 MILLER                     15600
```

뷰 (view)

4. 뷰 수정과 삭제

4.1 뷰 수정

```
SQL> CREATE OR REPLACE VIEW v10
  2                         (id_number, name, sal, department_id)
  3  AS SELECT empno, ename, sal, deptno
  4     FROM   emp
  5     WHERE  deptno = 20;

뷰가 생성되었습니다.
```

4.2 뷰 삭제

```
SQL> DROP VIEW v10;

뷰가 삭제되었습니다.
```

5. 뷰를 통한 데이터 조작

- 뷰를 통한 **DELETE** 불가능
  - 그룹 함수
  - **GROUP BY** 절
  - **DISTINCT** 키워드
  - 의사 열 **ROWNUM** 키워드
- 뷰를 통한 **UPDATE** 불가능
  - 뷰를 통한 **DELETE**가 불가능한 모든 경우
  - 표현식에 의해 정의된 열
- 뷰를 통한 **INSERT** 불가능
  - 뷰를 통한 **UPDATE**가 불가능한 모든 경우
  - 기본 테이블에서 뷰에 의해 선택되지 않은 열에 **NOT NULL** 제약조건이 있는 경우

뷰 (view)

6. 뷰의 제약조건

6.1 WITH CHECK OPTION

- WITH CHECK OPTION을 사용하면 뷰를 통한 DML 작업이 뷰의 도메인 내에서 수행

```
SQL> CREATE OR REPLACE VIEW empvu20
  2  AS SELECT *
  3      FROM   emp
  4      WHERE  deptno = 20
  5      WITH CHECK OPTION CONSTRAINT empvu20_ck;

뷰가 생성되었습니다.
```

뷰 (view)

6.1 WITH CHECK OPTION (계속)

```
EMPNO ENAME      JOB          MGR HIREDATE     SAL       COMM     DEPTNO
----- ---------- -------- ------ -------- ------ ---------- ----------
 7566 JONES      MANAGER    7839 81/04/02   2975                    20
 7902 FORD       ANALYST    7566 81/10/03   3000                    20
 7788 SCOTT      ANALYST    7566 82/10/09   3000                    20
 7369 SMITH      CLERK      7902 80/10/17    800                    20
 7876 ADAMS      CLERK      7788 83/01/12   1100                    20
```

```
SQL> UPDATE empvu20
  2  SET    deptno = 10
  3  WHERE  empno = 7788;
UPDATE empvu21
       *
1행에 오류:
ORA-01402: 뷰의 WITH CHECK OPTION의 조건에 위배 됩니다
```

뷰 (view)

6.1 WITH CHECK OPTION (계속)

- WITH CHECK OPTION 사용 여부에 따른 구분

뷰를 통한 DML	WITH CHECK OPTION 미사용	WITH CHECK OPTION 사용
범위 안 → 범위 안	성공	성공
범위 안 → 범위 밖	성공	에러
범위 밖 (INSERT)	성공	에러
범위 밖 (UPDATE)	0 행	0 행
범위 밖 (DELETE)	0 행	0 행

뷰 (view)

6.2 WITH READ ONLY

- WITH READ ONLY를 추가하면 DML 작업 거부
- 뷰를 통해 행에 DML 작업을 수행하면 Oracle Server 오류가 발생

```
SQL> CREATE OR REPLACE VIEW empvu20
  2  AS SELECT *
  3      FROM   emp
  4      WHERE  deptno = 20
  5      WITH READ ONLY;

뷰가 생성되었습니다.
```

뷰 (view)

6.2 WITH READ ONLY (계속)

```
SQL> DELETE empvu20
  2  WHERE  empno = 7788;
DELETE empvu20
       *
1행에 오류:
ORA-01752: 뷰으로 부터 정확하게 하나의 키-보전된 테이블 없이 삭제할 수 없습니다
```

```
SQL> UPDATE empvu20
  2  SET    deptno = 10
  3  WHERE  empno = 7788;
SET    deptno = 10
       *
2행에 오류:
ORA-01733: 가상 열은 사용할 수 없습니다
```

기타 객체

기타 객체

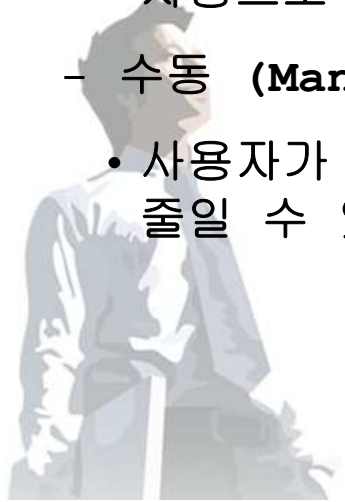
1. 인덱스 (Index)

1.1 인덱스 개념

- 포인터를 사용하여 행 검색 속도를 높일 수 있는 스키마 객체
- 데이터 위치를 빠르게 찾는 신속한 경로 액세스 방법을 사용하여 디스크 I/O를 감소
- 논리적 또는 물리적으로 인덱스화 된 테이블과 독립되어 존재
- **Oracle Server**에 의해 사용되며 자동으로 유지 관리
- 테이블을 삭제하면 해당하는 인덱스도 삭제

1.2 인덱스 생성

- 인덱스 생성 방법
  - 자동 **(Automatically)**
    - 테이블 정의에 **PRIMARY KEY** 또는 **UNIQUE** 제약 조건을 정의하면 고유 인덱스가 자동으로 생성
    - 인덱스의 이름은 제약조건에 지정한 이름과 동일
    - 자동으로 생성된 인덱스는 수동으로 삭제할 수 없음
  - 수동 **(Manually)**
    - 사용자가 열에 고유하지 않은 인덱스를 생성하여 행에 대한 액세스 시간을 줄일 수 있음



기타 객체

1.2 인덱스 생성 (계속)

```
CREATE INDEX index
ON table (column[, column]...);
```

```
SQL> CREATE INDEX sal_comm_idx
  2  ON emp (sal, comm);

인덱스가 생성되었습니다.
```

```
SQL> CREATE INDEX dept_dname_idx
  2  ON dept (dname);

인덱스가 생성되었습니다.
```

1.2 인덱스 생성 (계속)

- 인덱스 생성 지침
  - 인덱스를 생성하는 경우
    - 열이 **WHERE** 절이나 조인 조건에서 자주 사용되는 경우
    - 열에 광범위한 값이 포함된 경우
    - 열에 많은 수의 **NULL** 값이 포함된 경우
    - **WHERE** 절 또는 조인 조건에서 두개 이상의 열이 함께 자주 사용되는 경우
    - 큰 테이블에서 대부분의 질의에 의해 검색되는 범위가 **2~4%** 미만인 경우

1.2 인덱스 생성 (계속)

- 인덱스 생성 지침 (계속)
  - 인덱스를 생성하지 않는 경우
    - 테이블이 작은 경우
    - 열이 질의의 조건으로 자주 사용되지 않는 경우
    - 대부분의 질의에 의해 검색되는 행이 2-4% 이상인 경우
    - 테이블이 자주 갱신되는 경우

기타 객체

1.3 인덱스 정보 확인

- **USER_INDEXES** 데이터 딕셔너리 뷰는 인덱스 이름 및 고유성을 포함
- **USER_IND_COLUMNS** 뷰는 인덱스 이름, 테이블 이름 및 열 이름을 포함

```
SQL> SELECT   ic.index_name, ic.column_name,
  2           ic.column_position, ix.uniqueness
  3  FROM     user_indexes ix, user_ind_columns ic
  4  WHERE    ic.index_name = ix.index_name
  5  AND      ic.table_name = 'EMP';
```

기타 객체

1.4 함수 기반 인덱스

- 함수 기반 인덱스는 표현식을 기반으로 하는 인덱스
- 인덱스 표현식은 테이블 열, 상수, SQL 함수 및 사용자가 정의한 함수에서 생성

```
SQL> CREATE INDEX upper_ename_idx
  2  ON emp (UPPER(ename));
```

기타 객체

1.5 인덱스 삭제

- 데이터 딕셔너리에서 인덱스를 제거
- 인덱스를 제거하려면 인덱스 소유자이거나 **DROP ANY INDEX** 권한을 가지고 있어야 함

```
SQL> DROP INDEX index;
```

```
SQL> DROP INDEX upper_ename_idx;

인덱스가 삭제되었습니다.
```

기타 객체

2. 시퀀스 (Sequence)

2.1 시퀀스 개요

- 고유 번호를 자동으로 생성
- 테이블과 별도로 저장 및 생성
- 공유 가능한 객체
- 일반적으로 기본 키 값을 생성하는데 사용
- 응용 프로그램 코드를 대체
- 시퀀스 값을 메모리에 캐시하면 액세스 효율이 향상

기타 객체

2.2 시퀀스 생성

```
CREATE SEQUENCE sequence
          [INCREMENT BY n]
          [START WITH n]
          [{MAXVALUE n | NOMAXVALUE}]
          [{MINVALUE n | NOMINVALUE}]
          [{CYCLE | NOCYCLE}]
          [{CACHE n | NOCACHE}];
```

```
SQL> CREATE SEQUENCE dept_deptno_seq
  2          INCREMENT BY 10
  3          START WITH 50
  4          MAXVALUE 110
  5          NOCYCLE
  6          NOCACHE;

주문번호가 생성되었습니다.
```

기타 객체

2.3 시퀀스 사용

- **NEXTVAL** 및 **CURRVAL** 의사 열
  - **NEXTVAL**은 사용 가능한 다음 시퀀스 값을 반환
  - **NEXTVAL**은 참조될 때마다 고유한 값을 반환
  - **CURRVAL**은 현재 시퀀스 값을 반환
  - **CURRVAL**이 값을 포함하려면 먼저 해당 시퀀스에 대해 **NEXTVAL**이 실행되어야 함

기타 객체

2.3 시퀀스 사용 (계속)

- **NEXTVAL** 및 **CURRVAL** 사용 규칙
  - **NEXTVAL** 및 **CURRVAL**을 사용할 수 있는 경우
    - 서브쿼리에 속하지 않은 **SELECT** 문의 **SELECT** 목록
    - **INSERT** 문에 있는 서브쿼리의 **SELECT** 문
    - **INSERT** 문의 **VALUES** 절
    - **UPDATE** 문의 **SET** 절

기타 객체

2.3 시퀀스 사용 (계속)

- **NEXTVAL 및 CURRVAL 사용 규칙 (계속)**
  - **NEXTVAL 및 CURRVAL**을 사용할 수 없는 경우
    - 뷰의 **SELECT** 목록
    - **DISTINCT** 키워드가 있는 **SELECT** 문
    - **GROUP BY, HAVING, ORDER BY** 절이 있는 **SELECT** 문
    - **SELECT, DELETE, UPDATE** 문의 서브쿼리
    - **CREATE TABLE, ALTER TABLE** 문의 **DEFAULT** 표현식

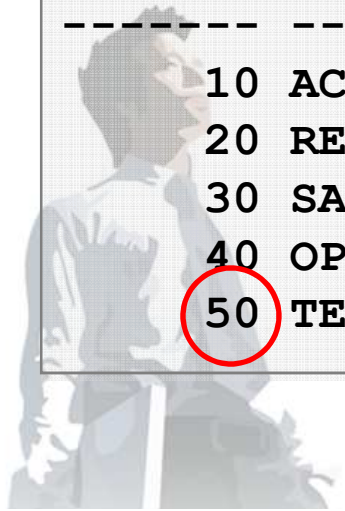
기타 객체

2.3 시퀀스 사용 (계속)

```
SQL> INSERT INTO dept
  2  VALUES (dept_deptno_seq.NEXTVAL, 'TESTING', 'SEOUL');
```

```
SQL> SELECT *
  2  FROM   dept;
```

```
    DEPTNO DNAME                          LOC
---------- ------------------------------ ---------
        10 ACCOUNTING                     NEW YORK
        20 RESEARCH                       DALLAS
        30 SALES                          CHICAGO
        40 OPERATIONS                     BOSTON
        50 TESTING                        SEOUL
```



기타 객체

2.3 시퀀스 사용 (계속)

- 시퀀스 값 사이에 공백(GAP)이 발생하는 경우
  - 롤백(ROLLBACK)이 발생하는 경우
  - 시스템이 고장 난 경우
  - 시퀀스가 다른 테이블에서 사용되는 경우
- NOCACHE로 시퀀스를 생성한 경우 USER_SEQUENCES 테이블을 질의하여 사용 가능한 다음 값을 볼 수 있음

기타 객체

2.4 시퀀스 정보 확인

- **USER_SEQUENCES** 데이터 딕셔너리 테이블에서 시퀀스 값을 확인

```
SQL> SELECT sequence_name, min_value, max_value,
  2         increment_by, last_number
  3  FROM   user_sequences;
```

```
SEQUENCE_NAME    MIN_VALUE  MAX_VALUE INCREMENT_BY LAST_NUMBER
---------------- --------- ---------- ------------ -----------
DEPT_DEPTNO_SEQ          1        110           10          60
```

- **NOCACHE**가 지정된 경우 **LAST_NUMBER** 열에는 사용 가능한 다음 시퀀스 번호가 표시

기타 객체

2.5 시퀀스 수정

```
ALTER SEQUENCE sequence
      [INCREMENT BY n]
      [{MAXVALUE n | NOMAXVALUE}]
      [{MINVALUE n | NOMINVALUE}]
      [{CYCLE | NOCYCLE}]
      [{CACHE n | NOCACHE}];
```

```
SQL> ALTER SEQUENCE dept_deptno_seq
  2          INCREMENT BY 20
  3          MAXVALUE 99999
  4          NOCYCLE
  5          NOCACHE;

주문번호가 변경되었습니다.
```

기타 객체

2.5 시퀀스 수정 (계속)

- 시퀀스 수정 지침
  - 시퀀스 소유자이거나 시퀀스에 대한 **ALTER** 권한이 있어야 함
  - **ALTER SEQUENCE** 문은 이후 시퀀스 번호에만 영향을 줌
  - 시퀀스를 다른 번호로 다시 시작하려면 시퀀스를 삭제한 후 다시 생성
  - 일부 검증이 수행
    (MAXVALUE를 현재 시퀀스 값 보다 작게 지정할 수 없음**)**

2.6 시퀀스 삭제

```
SQL> DROP SEQUENCE dept_deptno_seq;
주문번호가 삭제되었습니다.
```

기타 객체

3. 동의어 (Synonym)

3.1 동의어 개념

- 동의어(객체의 다른 이름)를 생성하여 객체 액세스를 단순화
- 다른 사용자가 소유한 테이블을 참조
- 긴 객체 이름을 짧게 만듬
- 전용 동의어 이름은 동일한 사용자가 소유한 다른 모든 객체와 구별

기타 객체

3.2 동의어 생성

```
CREATE [PUBLIC] SYNONYM synonym
FOR    object;
```

```
SQL> CREATE SYNONYM e_sum
  2  FOR empvu20;
```

```
SQL> SELECT *
  2  FROM   e_sum;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	81/04/02	2975		20
7902	FORD	ANALYST	7566	81/10/03	3000		20
7788	SCOTT	ANALYST	7566	82/10/09	3000		20
7369	SMITH	CLERK	7902	80/10/17	800		20
7876	ADAMS	CLERK	7788	83/01/12	1100		20

기타 객체

3.3 동의어 정보 확인

```
SQL> SELECT synonym_name, table_owner, table_name
  2  FROM   user_synonyms;
```

```
SYNONYM_NAME      TABLE_OWNER                    TABLE_NAME
----------------- ------------------------------ -----------
E_SUM             SCOTT                          EMPVU20
```

3.4 동의어 삭제

```
SQL> DROP SYNONYM e_sum;

동의어가 삭제되었습니다.
```

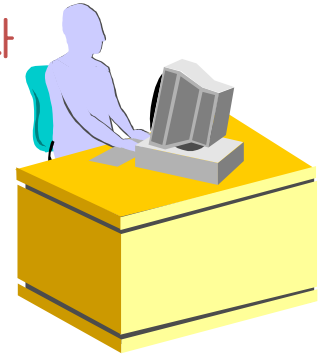
데이터 제어어 (DCL)

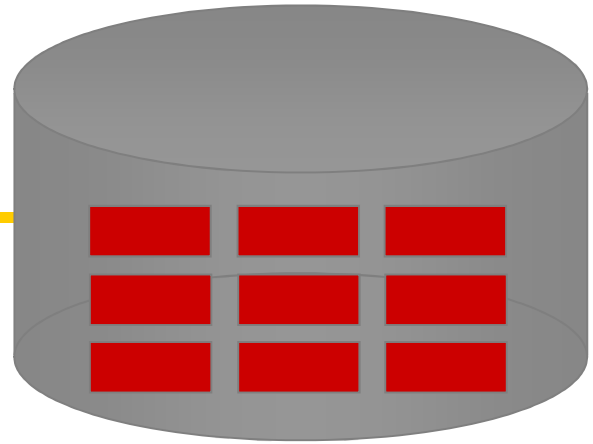
데이터 제어어 (DCL)

1. 사용자 액세스 제어

1.1 사용자 액세스 제어 개요

데이터베이스 관리자
(DBA)

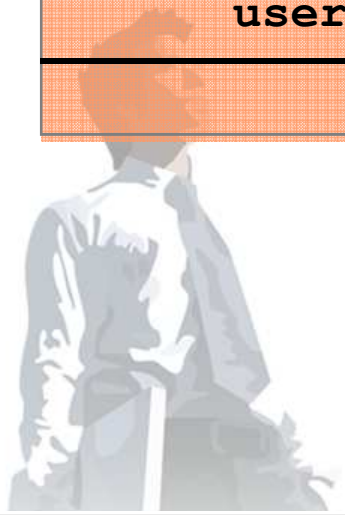




username / password

권한

사용자
(User)



데이터 제어어 (DCL)

1.2 권한

- 데이터베이스 보안
  - 시스템 보안
  - 데이터 보안
- 시스템 권한
  - 데이터베이스를 액세스할 수 있음
- 객체 권한
  - 데이터베이스 객체의 내용을 조작할 수 있음

데이터 제어어 (DCL)

2. 시스템 권한

2.1 개요

- 100가지 이상의 권한을 사용할 수 있음
- 일반적인 DBA 권한
  - 새로운 사용자 생성 (CREATE USER)
  - 사용자 제거 (DROP USER)
  - 테이블 제거 (DROP ANY TABLE)
  - 테이블 백업 (BACKUP ANY TABLE)
  - 스키마 질의 (SELECT ANY TABLE)
  - 테이블 생성 (CREATE ANY TABLE)
- 시스템 권한 확인

```
SQL> SELECT *
  2  FROM   system_privilege_map;
```

데이터 제어어 (DCL)

2.2 사용자 생성

- **DBA**는 **CREATE USER** 문을 사용하여 사용자를 생성
- 처음 생성된 사용자는 아무런 권한도 갖지 않음
- **DBA**는 해당 사용자에게 여러 권한을 부여할 수 있으며, 부여하는 권한에 따라 사용자가 데이터베이스 레벨에서 수행할 수 있는 작업이 결정

```
CREATE USER    user
IDENTIFIED BY password;
```

```
SQL> CONN system/(passwd)
```

```
SQL> CREATE USER skj
  2  IDENTIFIED BY test;

사용자가 생성되었습니다.
```

데이터 제어어 (DCL)

2.3 시스템 권한 부여

- 사용자가 생성되었다면 **DBA**는 생성된 사용자에게 특정 시스템 권한을 부여할 수 있음
- 응용 프로그램 개발자는 다음과 같은 시스템 권한을 가짐
  - **CREATE SESSION** (DB로 접속)
  - **CREATE TABLE** (사용자 테이블 생성)
  - **CREATE SEQUENCE** (시퀀스 생성)
  - **CREATE VIEW** (뷰 생성)
  - **CREATE PROCEDURE** (프로시저, 함수, 패키지 생성)

```
GRANT privilege [, privilege...]
TO user [, user...]
[WITH ADMIN OPTION];
```

데이터 제어어 (DCL)

2.3 시스템 권한 부여 (계속)

```
SQL> GRANT create session, create table, create view
  2  TO     skj;

권한이 부여되었습니다.
```

```
SQL> CONN skj/test
연결되었습니다.
```

데이터 제어어 (DCL)

2.4 시스템 권한 철회

- 사용자와 롤에 부여된 시스템 권한 취소
- **WITH ADMIN OPTION**에 의한 연이은 권한은 철회되지 않음

```
REVOKE {privilege [, privilege...]}
FROM   {user[, user...]|role};
```

```
REVOKE create table, create view
FROM   skj;
```

데이터 제어어 (DCL)

2.5 비밀번호 변경

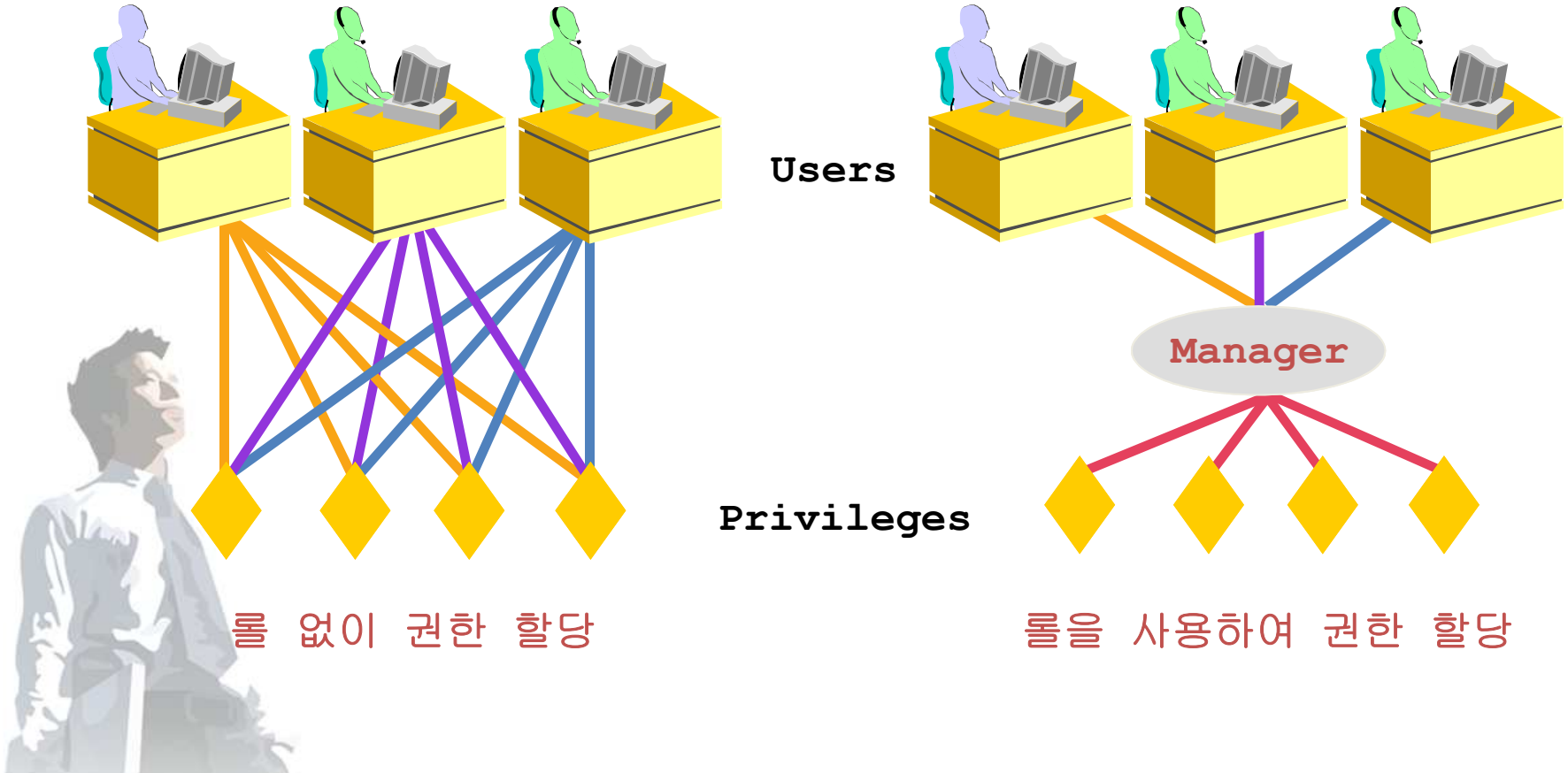
- 사용자 계정이 생성되었을 때 각 사용자는 **DBA**가 초기화 한 비밀번호 보유
- 사용자는 **ALTER USER** 문장을 사용하여 비밀번호를 변경할 수 있음

```
SQL> ALTER USER skj
  2  IDENTIFIED BY lion;

사용자가 변경되었습니다.
```

데이터 제어어 (DCL)

3. 롤 (Role)

3.1 개요



데이터 제어어 (DCL)

3.2 롤 생성 및 권한 부여

```
SQL> CREATE ROLE manager;

롤이 생성되었습니다.
```

```
SQL> GRANT create session, create table, create view
  2  TO    manager;

권한이 부여되었습니다.
```

```
SQL> GRANT manager
  2  TO    skj;

권한이 부여되었습니다.
```

```
SQL> DROP ROLE manager;

롤이 삭제되었습니다.
```

데이터 제어어 (DCL)

4. 객체 권한

4.1 개요

- 객체 권한은 객체 마다 다름
- 소유자는 객체에 대한 모든 권한을 가짐
- 소유자는 자신의 객체에 대한 특정 권한을 부여할 수 있음

```
GRANT object_priv [(columns)] | ALL
ON    object
TO    {user|role|PUBLIC}
[WITH GRANT OPTION];
```

데이터 제어어 (DCL)

4.1 개요 (계속)

객체 권한	테이블 (Table)	뷰 (View)	시퀀스 (Sequence)	프로시저 (Procedure)
ALTER	✓		✓	
DELETE	✓	✓		
EXECUTE				✓
INDEX	✓			
INSERT	✓	✓		
REFERENCES	✓	✓		
SELECT	✓	✓	✓	
UPDATE	✓	✓		

데이터 제어어 (DCL)

4.2 객체 권한 부여

```
SQL> GRANT select
  2  ON     emp
  3  TO     skj;

권한이 부여되었습니다.
```

```
SQL> GRANT update (deptno, loc)
  2  ON     dept
  3  TO     skj;

권한이 부여되었습니다.
```

데이터 제어어 (DCL)

4.2 객체 권한 부여 (계속)

```
SQL> GRANT select, insert
  2  ON     dept
  3  TO     skj
  4  WITH GRANT OPTION;

권한이 부여되었습니다.
```

```
SQL> GRANT select
  2  ON     scott.dept
  3  TO     PUBLIC;

권한이 부여되었습니다.
```

데이터 제어어 (DCL)

4.3 객체 권한 철회

- **REVOKE** 문을 사용하여 다른 사용자와 롤에 부여된 권한을 취소
- **WITH GRANT OPTION**을 통해 다른 사용자에게 부여한 권한도 취소

```
REVOKE {privilege [, privilege...]|ALL}
ON     object
FROM   {user[, user...]|role|PUBLIC};
```

```
SQL> REVOKE   select, insert
  2  ON       dept
  3  FROM     skj;

권한이 취소되었습니다.
```

데이터 제어어 (DCL)

5. 부여된 권한 정보 확인

데이터 딕셔너리 뷰	설명
ROLL_SYS_PRIVS	롤에 부여된 시스템 권한입니다.
ROLE_TAB_PRIVS	롤에 부여된 테이블 권한입니다.
USER_ROLE_PRIVS	사용자가 액세스할 수 있는 롤입니다.
USER_TAB_PRIVS_MADE	사용자 객체에 대해 부여된 객체 권한입니다.
USER_TAB_PRIVS_RECD	사용자에게 부여된 객체 권한입니다.
USER_COL_PRIVS_MADE	사용자 객체의 열에 대해 부여된 객체 권한입니다.
USER_COL_PRIVS_RECD	특정 열에 대해 사용자에게 부여된 객체 권한입니다.
USER_SYS_PRIVS	사용자에게 부여된 시스템 권한을 나열합니다