

테이블 관리-DDL



배재대학교 컴퓨터공학과
김 창 수

Contents



테이블 생성



테이블 구조 변경



데이터 사전

테이블 생성

❖ 개요

- 테이블 생성은 테이블에 대한 구조를 정의하고, 데이터를 저장하기 위한 공간을 할당하는 과정
- 테이블에 대한 구조 정의는 테이블을 구성하는 칼럼의 데이터 타입과 무결성 제약조건을 정의하는 과정

❖ 테이블 이름 정의 방법

- 문자(A-Z, a-z)로 시작, 30자 이내
- 문자(a-z, A-Z), 숫자(0-9), 특수문자(underscore, dollar, hash) 사용 가능
- 대소문자 구별 없음, 소문자로 저장하려면 단일 인용부호 이용
- 동일 사용자가 소유한 다른 객체의 이름과 중복 불가
- 서로 다른 테이블에서 동일한 데이터를 저장하는 칼럼 이름은 가능하면 같은 이름을 사용
- 필요에 따라 언제든지 테이블 생성 가능
- 완성된 설계도에 따라 테이블을 생성 권장

테이블 생성 방법

❖ 사용법

```
CREATE [GLOBAL TEMPORARY] TABLE [schema.] table  
(column datatype[DEFAULT expression][column_constraint clause]  
[,...]);
```

- GLOBAL TEMPORARY : 임시 테이블을 만들기 위한 키워드로서 테이블 구조는 모든 세션에서 볼 수 있지만, 데이터는 테이블을 생성한 세션에서만 조회 가능
- schema : 데이터베이스 사용자 계정과 같은 의미
- table : 생성하고자 하는 테이블 이름
- column : 테이블에 포함되는 칼럼 이름
- datatype : 칼럼에 대한 데이터 타입과 길이
- DEFAULT expression : 데이터 입력 시 값이 생략된 경우에 입력되는 기본 값
- column_constraint_clause : 칼럼에 대해 정의되는 무결성 제약 조건

테이블 생성 방법

❖ 사용 예

- 연락처 정보를 저장하기 위한 주소록(address)테이블을 생성하여라.

```
SQL> CREATE TABLE address  
2 (id NUMBER(3),  
3 name VARCHAR2(50),  
4 addr VARCHAR2(100),  
5 phone VARCHAR2(30),  
6 email VARCHAR2(100));
```

테이블이 생성되었습니다.

CREATE 명령문을 사용하여 주소록 테이블을 생성한다.

테이블 생성 방법

```
SQL> SELECT * FROM tab;
```

TNAME	TABTYPE	CLUSTERID
ADDRESS	TABLE	
DEPARTMENT	TABLE	
HEIGHT_INFO	TABLE	
PROFESSOR	TABLE	
PROFESSOR_TEMP	TABLE	
SALES	TABLE	
SALES_DATA	TABLE	
SALGRADE	TABLE	
STUDENT	TABLE	
STUD_HEAVY	TABLE	
WEIGHT_INFO	TABLE	

주소록 테이블의 생성을 확인한다.

11 개의 행이 선택되었습니다.

주소록 테이블의 구조를 확인한다.

```
SQL> DESC address
```

이름	길이?	유형
ID		NUMBER(3)
NAME		VARCHAR2(50)
ADDR		VARCHAR2(100)
PHONE		VARCHAR2(30)
EMAIL		VARCHAR2(100)

DEFAULT 옵션

❖ 기능

- 칼럼의 입력 값이 생략될 경우에 NULL 대신에 입력되는 기본 값을 지정하기 위한 기능
- 기본값 : 리터럴 값, 표현식, SQL함수 , SYSDATE, USER를 사용
- 칼럼이나 의사칼럼(NEXTVAL, CURRVAL)은 사용할 수 없음

❖ 칼럼 정의 시 기본 값 설정 예

```
addr      varchar2(100)  DEFAULT 'KOREA'
```

테이블 생성 확인

❖ DESC[RIBE] 명령어

- 테이블의 생성 여부와 테이블의 구조를 확인하기 위한 명령어
- 칼럼 이름, 데이터 타입과 크기, NOT NULL 무결성 제약조건

❖ 사용법

```
DESC[RIBE] table
```


서브쿼리를 이용한 테이블 생성

❖ 개요

- CREATE TABLE 명령문에서 서브쿼리 절을 이용하여 다른 테이블의 구조와 데이터를 복사하여 새로운 테이블 생성 가능
- 서브쿼리의 출력 결과가 테이블의 초기 데이터로 삽입

❖ 기능

- CREATE TABLE 명령문에서 지정한 칼럼 수와 데이터 타입과 반드시 일치
- 칼럼 이름을 명시하지 않을 경우 서브쿼리 칼럼 이름과 동일
- 무결성 제약조건은 NOT NULL 조건만 복사
 - 기본 키, 참조 키와 같은 무결성 제약조건은 사용자의 재정의 필요
- 디폴트 옵션에서 정의한 값은 그대로 복사

❖ 사용법

```
CREATE TABLE table
    [column[, column,...]]
AS subquery;
```

서브쿼리를 이용한 테이블 생성

❖ 예제 데이터 입력

```
SQL> INSERT INTO address
  2  VALUES(1, 'HGDONG', 'SEOUL', '123-4567', 'gdhong@cwunet.ac.kr');
```

1 개의 행이 만들어졌습니다.

```
SQL> COMMIT;
```

커밋이 완료되었습니다.

```
SQL> SELECT * FROM address;
```

ID	NAME
1	HGDONG
	SEOUL
	123-4567
	gdhong@cwunet.ac.kr

서브쿼리를 이용한 테이블 생성

❖ 사용 예

- 서브쿼리 절을 이용하여 주소록 테이블의 구조와 데이터를 복사하여 addr_second 테이블을 생성하여라.

```
SQL> CREATE TABLE addr_second(id, name, addr, phone, e_mail)
2 AS SELECT * FROM address;
```

테이블이 생성되었습니다.

기존 테이블 구조와 데이터를 복사하여 새로운 테이블 생성하고 e_mail 컬럼처럼 이름을 변경할 수 있다.

```
SQL> DESC addr_second;
```

이름	널?	유형
ID		NUMBER(3)
NAME		VARCHAR2(50)
ADDR		VARCHAR2(100)
PHONE		VARCHAR2(30)
E_MAIL		VARCHAR2(100)

테이블 구조 복사

❖ 기존 테이블의 구조만 복사

- 서브쿼리를 이용한 테이블 생성시 데이터는 복사하지 않고 기존 테이블의 구조만 복사 가능
- 서브쿼리의 WHERE 조건절에 거짓이 되는 조건을 지정하여 출력 결과 집합이 생성되지 않도록 지정

❖ 사용법

```
CREATE TABLE    table
AS SELECT *
FROM            source table
WHERE           condition;
```

- condition : 출력 결과가 항상 거짓인 조건을 명시.
 - 예) WHERE 1=2

테이블 구조 복사

❖ 사용 예

- 주소록 테이블에서 id, name 칼럼만 복사하여 addr_fourth 테이블을 생성하여라. 단, 데이터는 복사하지 않는다.

```
SQL> CREATE TABLE addr_fourth
2 AS SELECT id, name FROM address
3 WHERE 1=2;
```

테이블이 생성되었습니다.

```
SQL> DESC addr_fourth
```

이름	널?	유형
ID		NUMBER(3)
NAME		VARCHAR2(50)

id, name 칼럼만 복사하여 테이블을 생성하고, 데이터는 복사하지 않는다.

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

선택된 레코드가 없습니다.

서브쿼리를 이용한 테이블 생성

❖ 사용 예

- 주소록 테이블에서 id, name 칼럼만 복사하여 addr_third 테이블을 생성하여라.

```
SQL> CREATE TABLE addr_third
2 AS SELECT id, name FROM address;
```

테이블이 생성되었습니다.

```
SQL> DESC addr_third;
```

이름	길이?	유형
ID		NUMBER(3)
NAME		VARCHAR2(50)

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

ID	NAME
1	HGDONG



테이블 구조 변경

❖ 개요

- ALTER TABLE 명령문 이용
- 칼럼 추가, 삭제, 타입이나 길이의 재정의와 같은 작업

❖ 칼럼추가

- ALTER TABLE ... ADD 명령문 사용
- 추가된 칼럼은 테이블의 마지막 부분에 생성, 위치 지정 불가능
- 추가된 칼럼에도 기본 값을 지정 가능
- 수정할 테이블에 기존 데이터가 존재하면 칼럼 값은 NULL로 입력

❖ 사용법

```
ALTER TABLE table
ADD (column datatype [DEFAULT expression]
[, column datatype]...);
```

테이블에 칼럼 추가

❖ 사용 예

- 주소록 테이블에 날짜 타입을 가지는 birth 칼럼을 추가하여라.

```
SQL> ALTER TABLE address
2 ADD (birth DATE);
```

테이블이 변경되었습니다.

```
SQL> DESC address
```

이름	날?	유형
ID		NUMBER(3)
NAME		VARCHAR2(50)
ADDR		VARCHAR2(100)
PHONE		VARCHAR2(30)
EMAIL		VARCHAR2(100)
BIRTH		DATE

테이블에 칼럼 추가

❖ 실습 예

- 주소록 테이블에 문자 타입을 가지는 comment 칼럼을 추가하여라.

```
SQL> ALTER TABLE address
2 ADD (comments VARCHAR2(200) DEFAULT 'No Comment');
```

테이블이 변경되었습니다.

```
SQL> DESC address
```

이름	널?	유형
ID		NUMBER(3)
NAME		VARCHAR2(50)
ADDR		VARCHAR2(100)
PHONE		VARCHAR2(30)
EMAIL		VARCHAR2(100)
BIRTH		DATE
COMMENTS		VARCHAR2(200)

테이블 칼럼 삭제

❖ 기능

- 테이블 내의 특정 칼럼과 칼럼의 데이터를 삭제
- ALTER TABLE ... DROP COLUMN 명령문 사용
- 2개 이상의 칼럼이 존재하는 테이블에서만 삭제 가능
- 하나의 칼럼 삭제 명령문은 하나의 칼럼만 삭제 가능

❖ 사용법

```
ALTER TABLE table DROP column;
```

테이블 칼럼 삭제

❖ 사용 예

- 주소록 테이블에서 comment 칼럼을 삭제하여라.

```
SQL> ALTER TABLE address DROP COLUMN comments;
```

테이블이 변경되었습니다.

```
SQL> DESC address
```

이름	길이?	유형

ID		NUMBER(3)
NAME		VARCHAR2(50)
ADDR		VARCHAR2(100)
PHONE		VARCHAR2(30)
EMAIL		VARCHAR2(100)
BIRTH		DATE

테이블 칼럼 변경

❖ 기능

- 테이블에서 칼럼의 타입, 크기, 기본 값 변경 가능
- ALTER TABLE ... MODIFY 명령문 이용
- 기존 칼럼에 데이터가 없는 경우
 - 칼럼 타입이나 크기 변경이 자유로움
- 기존 데이터가 존재하는 경우
 - 타입 변경은 CHAR와 VARCHAR2만 허용
 - 변경한 칼럼의 크기가 저장된 데이터의 크기보다 같거나 클 경우 변경 가능
 - 숫자 타입에서는 정밀도 증가 가능
- 기본 값의 변경은 변경 후에 입력되는 데이터부터 적용

❖ 사용법

```
ALTER TABLE table
MODIFY (column datatype [DEFAULT expression]
[, column datatype]...);
```

테이블 칼럼 변경

❖ 사용 예

- 주소록 테이블에서 phone 칼럼의 데이터 타입의 크기를 50으로 증가하여라.

```
SQL> ALTER TABLE address
2 MODIFY phone VARCHAR2(50);
```

데이터가 저장되어 있어도 데이터 타입의 크기를 증가 시킬 수 있다

테이블이 변경되었습니다.

```
SQL> ALTER TABLE address
2 MODIFY phone VARCHAR2(5);
MODIFY phone VARCHAR2(5)
*
```

변경 칼럼의 크기를 기존에 저장된 데이터 크기보다 작게 지정하면 오류가 발생한다.

2행에 오류:

ORA-01441: 일부 값이 너무 커서 열 길이를 줄일 수 없음

```
SQL> DESC address
```

이름	널?	유형
ID		NUMBER(3)
NAME		VARCHAR2(50)
ADDR		VARCHAR2(100)
PHONE		VARCHAR2(50)
EMAIL		VARCHAR2(100)
BIRTH		DATE

테이블 이름 변경

❖ 기능

- RENAME 명령문 사용
 - 객체의 이름을 변경하는 DDL 명령문
 - 뷰, 시퀀스, 동의어 등과 같은 데이터베이스 객체의 이름 변경 가능

❖ 사용법

```
RENAME    old_table    TO    new_table
```

테이블 이름 변경

❖ 사용 예

- addr_second 테이블 이름을 client_address로 변경하여라.

```
SQL> RENAME addr_second TO client_address;
```

테이블명이 바뀌었습니다.

```
SQL> SELECT * FROM tab;
```

TNAME	TABTYPE	CLUSTERID
-----	-----	-----
ADDRESS	TABLE	
ADDR_FOURTH	TABLE	
ADDR_THIRD	TABLE	
CLIENT_ADDRESS	TABLE	
DEPARTMENT	TABLE	
HEIGHT_INFO	TABLE	
PROFESSOR	TABLE	
PROFESSOR_TEMP	TABLE	

테이블 삭제

❖ 기능

- 기존 테이블과 데이터를 모두 삭제
- DROP TABLE 명령문 사용
- 삭제된 테이블 칼럼에 대해 생성된 인덱스도 함께 삭제
- 삭제된 테이블과 관련된 뷰와 동의어 'invalid' 상태
- 삭제할 테이블의 기본 키나 고유 키를 다른 테이블에서 참조하고 있는 경우 삭제 불가능
 - 참조하는 테이블(자식 테이블)을 먼저 삭제
 - DROP TABLE 명령문 마지막에 CASCADE CONSTRAINTS 옵션을 사용하여 무결성 제약조건을 동시에 삭제

❖ 사용법

```
DROP      TABLE      [schema.] table [cascade constraints]
```

- cascade constraints : 삭제 대상 테이블의 기본 키나 고유 키를 참조하는 무결성 제약조건을 동시에 삭제하기 위한 옵션

테이블 삭제

❖ 사용 예

- addr_third 테이블을 삭제 하여라.

```
SQL> SELECT * FROM tab;
```

TNAME	TABTYPE	CLUSTERID
ADDRESS	TABLE	
ADDR_FOURTH	TABLE	
ADDR_THIRD	TABLE	
CLIENT_ADDRESS	TABLE	
DEPARTMENT	TABLE	
HEIGHT_INFO	TABLE	
PROFESSOR	TABLE	
PROFESSOR_TEMP	TABLE	
SALES	TABLE	
SALES_DATA	TABLE	
SALGRADE	TABLE	
STUDENT	TABLE	
STUD_HEAVY	TABLE	
WEIGHT_INFO	TABLE	

addr_third 테이블 확인

14 개의 행이 선택되었습니다.

```
SQL> DROP TABLE addr_third;
```

테이블이 삭제되었습니다.

addr_third 테이블 삭제

```
SQL> SELECT * FROM tab
2 WHERE tname = 'ADDR_THIRD';
```

addr_third 테이블 삭제 확인

선택된 레코드가 없습니다.

TRUNCATE 명령문

❖ 기능

- 테이블 구조는 그대로 유지하고, 테이블의 데이터와 할당된 공간만 삭제
- 테이블에 생성된 제약조건과 연관된 인덱스, 뷰, 동의어는 유지

❖ DELETE 명령문과 차이

- DELETE 명령문
 - 기존 데이터만 삭제하는 명령이며, ROLLBACK 가능
 - WHERE 절을 이용하여 특정 행만 삭제 가능
- TRUNCATE 명령문
 - 기존 데이터 삭제뿐 아니라, 물리적인 저장 공간까지 반환
 - DDL 문이므로 ROLLBACK 이 불가능
 - WHERE 절을 이용하여 특정 행만 삭제하는 것이 불가능

❖ 사용법

```
TRUNCATE TABLE [schema.] table
```

TRUNCATE 명령문

❖ 사용 예

- client_address 테이블의 데이터와 할당된 공간을 삭제하여라.

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

client_address 데이터 확인

ID NAME

ADDR

PHONE

E_MAIL

1 HGDONG
SEOUL
123-4567
gdhong@cwunet.ac.kr

TRUNCATE 명령문

```
SQL> TRUNCATE TABLE client_address;
```

TRUNCATE 명령문 실행

테이블이 잘렸습니다.

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

데이터 삭제 여부 확인

선택된 레코드가 없습니다.

```
SQL> ROLLBACK;
```

롤백이 완료되었습니다.

ROLLBACK을 수행하여도 삭제된 데이터는
복구 불가능

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

선택된 레코드가 없습니다.



주석 추가

❖ 기능

- 테이블이나 칼럼에 최대 2,000 바이트까지 주석을 추가
- COMMENT ON TABLE ... IS 명령문 이용
- 추가된 주석 확인
 - ALL_COL_COMMENTS, USER_COL_COMMENTS, ALL_TAB_COMMENTS 데이터 사전 질의

❖ 사용법:테이블에 주석 추가

```
COMMENT ON TABLE table  
IS 'content of comment' ;
```

❖ 사용법 : 칼럼에 주석 추가

```
COMMENT ON COLUMN table.column  
IS 'content of comment' ;
```

주석 추가

❖ 사용 예

- 주소록 테이블에서 ‘고객 주소록 관리하기 위한 테이블’ 이라는 주석을 추가하여라.

```
SQL> COMMENT ON TABLE address  
2          IS '고객 주소록을 관리하기 위한 테이블';
```

주석이 생성되었습니다.

❖ 사용 예

- 주소록 테이블의 name 칼럼에 ‘고객이름’ 이라는 주석을 추가하여라.

```
SQL> COMMENT ON COLUMN address.name  
2          IS '고객 이름';
```

주석이 생성되었습니다.

데이터 사전

❖ 개요

- 사용자와 데이터베이스 자원을 효율적으로 관리하기 위한 다양한 정보를 저장하는 시스템 테이블의 집합
- 사전 내용의 수정은 오라클 서버만 가능
 - 오라클 서버는 데이터베이스의 구조, 감사, 사용자 권한, 데이터 등의 변경 사항을 반영하기 위해 지속적 수정 및 관리
- 데이터베이스 관리자나 일반 사용자는 읽기 전용 뷰에 의해 데이터 사전의 내용을 조회만 가능
- 실무에서는 테이블, 칼럼, 뷰 등과 같은 정보를 조회하기 위해 사용

데이터 사전

❖ 데이터 사전의 관리 정보

- 데이터베이스의 물리적 구조와 객체의 논리적 구조
- 오라클 사용자 이름과 스키마 객체 이름
- 사용자에게 부여된 접근 권한과 롤
- 무결성 제약조건에 대한 정보
- 칼럼별로 지정된 기본값
- 스키마 객체에 할당된 공간의 크기와 사용 중인 공간의 크기 정보
- 객체 접근 및 갱신에 대한 감사 정보
- 데이터베이스 이름, 버전, 생성날짜, 시작모드, 인스턴스 이름 정보

데이터 사전의 종류

❖ 개요

- 다수의 사용자가 동일한 데이터를 공유
- 읽기 전용 뷰로 구성
- 데이터베이스 관리자나 사용자에게 데이터 사전에 저장된 정보 조회 허용
- 용도에 따라 USER, ALL, DBA 접두어를 사용하여 분류

접두어 종류에 따른 데이터 사전 뷰

접두어	접근 범위
USER_	객체의 소유자만 접근 가능한 데이터 사전 뷰
ALL_	자기 소유 또는 권한을 부여 받은 객체만 접근 가능한 데이터 사전 뷰
DBA_	데이터베이스 관리자만 접근 가능한 데이터 사전 뷰

USER_ 데이터 사전 뷰

❖ 기능

- 일반 사용자와 가장 밀접하게 관련된 뷰
- 자신이 생성한 테이블, 인덱스, 뷰, 동의어 등의 객체나 해당 사용자에게 부여된 권한 정보 조회

❖ 사용 예

- USER_ 데이터 사전 뷰 조회 예

```
SQL> SELECT table_name FROM user_tables;
```

```
TABLE_NAME
```

```
-----
```

```
ADDRESS
```

```
ADDR_FOURTH
```

```
CLIENT_ADDRESS
```

```
DEPARTMENT
```

```
HEIGHT_INFO
```

```
PROFESSOR
```

```
PROFESSOR_TEMP
```

```
SALES
```

```
SALES_DATA
```

```
SALGRADE
```

```
STUDENT
```

```
STUD_HEAVY
```

```
WEIGHT_INFO
```

user_tables는 사용자가 소유한 테이블에 대한 정보를 조회할 수 있는 데이터 사전 뷰이다.



ALL_ 데이터 사전 뷰

❖ 기능

- 데이터베이스 전체 사용자와 관련된 뷰
- 해당 객체의 소유자를 확인가능
 - OWNER 칼럼 존재
- 사용자는 ALL_ 사전 뷰를 이용하여 접근할 수 있는 모든 객체에 대한 정보 조회 가능

❖ 사용 예

- ALL_ 데이터 사전 뷰 조회 예

```
SQL> SELECT owner, table_name FROM all_tables;
```

OWNER	TABLE_NAME
SYS	DUAL
SYS	SYSTEM_PRIVILEGE_MAP
SYS	TABLE_PRIVILEGE_MAP
SYS	STMT_AUDIT_OPTION_MAP
SYS	AUDIT_ACTIONS
SYS	PSTUPTBL
SYS	ODCI_SECOBJ\$
SYS	ODCI_WARNINGS\$
SYSTEM	DEF\$TEMP\$LOB
WMSYS	WM\$WORKSPACES_TABLE
SYSTEM	HELP
MDSYS	OGIS_SPATIAL_REFERENCE_SYSTEMS
MDSYS	USER_CS_SRS
MDSYS	USER_TRANSFORM_MAP

all_tables는 자기 소유 또는 권한을 부여받은 테이블에 대한 정보를 조회할 수 있는 데이터 사전 뷰이다.

DBA_ 데이터 사전 뷰

❖ DBA_ 데이터 사전 뷰

- 시스템 관리와 관련된 뷰
- DBA 나 SELECT ANY TABLE 시스템 권한을 가진 사용자
- 사용자 접근 권한, 데이터베이스 자원관리 목적

❖ 사용 예

- DBA_ 데이터 사전 뷰 조회 예

데이터 사전의 종류

데이터베이스 관리를 위해 자주 사용하는 데이터 사전 뷰

데이터 사전	설명
dictionary dict_columns	데이터 사전 테이블, 뷰 및 칼럼에 대한 정보
dba_tables dba_objects dba_tab_columns dba_constraints	테이블, 제약조건, 칼럼, 사용자 객체와 관련된 정보
dba_users dba_sys_privs dba_roles	사용자 권한과 롤에 관한 정보
dba_extents dba_free_space dba_segments	데이터베이스 객체에 대한 공간 할당 정보
dba_rollback_segs dba_data_files dba_tablespaces	데이터베이스 내부 공간의 구조 정보
dba_audit_trail dba_audit_object dba_obj_audit_opts	감사와 관련된 정보

사용자 테이블 정보 조회

❖ USER_TABLES

- 테이블이 저장된 테이블스페이스 이름, 데이터가 저장된 물리적 공간 그리고 블록 파라미터 정보 등과 같은 정보를 저장

❖ 사용 예

- 테이블 이름이 ADDR로 시작하는 테이블의 이름, 테이블이 저장된 테이블스페이스 이름, 최소 확장영역 수와 최대 확장영역 수를 출력하여라.

```
SQL> SELECT table_name, tablespace_name, min_extents, max_extents
2 FROM user_tables
3 WHERE table_name LIKE 'ADDR%';
```

TABLE_NAME	TABLESPACE_NAME	MIN_EXTENTS	MAX_EXTENTS
ADDRESS	SYSTEM	1	2147483645
ADDR_FOURTH	SYSTEM	1	2147483645

USER_OBJECT

❖ 기능

- 사용자가 생성한 테이블 정보와 함께 인덱스, 시퀀스, 동의어, 뷰 같은 객체에 대한 이름, 종류, 생성 날짜 등 정보 저장

❖ 사용 예

- 객체의 종류가 테이블이고 이름이 ADDR로 시작하는 객체의 이름, 종류, 생성날짜를 출력하여라.

```
SQL> SELECT object_name, object_type, created  
2 FROM user_objects  
3 WHERE object_name LIKE 'ADDR%' AND object_type = 'TABLE';
```

OBJECT_NAME

OBJECT_TYPE	CREATED
ADDRESS	
TABLE	06/11/06
ADDR_FOURTH	
TABLE	06/11/06

USER_CATALOG

❖ 기능

- 사용자 소유로 생성된 모든 객체 이름과 객체 종류에 대한 정보 저장
- Table_name : 객체 이름
- Table_type : 객체 종류

```
SQL> DESC user_catalog
```

이름	널?	유형
TABLE_NAME	NOT NULL	VARCHAR2(30)
TABLE_TYPE		VARCHAR2(11)

USER_CATALOG

❖ 사용 예

```
SQL> SELECT * FROM user_catalog;
```

TABLE_NAME	TABLE_TYPE
ADDRESS	TABLE
ADDR_FOURTH	TABLE
CLIENT_ADDRESS	TABLE
DEPARTMENT	TABLE
HEIGHT_INFO	TABLE
PROFESSOR	TABLE
PROFESSOR_TEMP	TABLE
SALES	TABLE
SALES_DATA	TABLE
SALGRADE	TABLE
STUDENT	TABLE
STUD_HEAVY	TABLE
WEIGHT_INFO	TABLE

13 개의 행이 선택되었습니다.

