

Chapter 10

Creating Tables: Data Definition Language (DDL)

Creating and Managing Tables

Objectives

- Describe the main database objects
- Create tables
- Describe the datatypes that can be used when specifying column definition
- Alter table definitions
- Drop, rename, and truncate tables

Database Objects

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Generates primary key values
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

The CREATE TABLE Statement

- You must have :
 - CREATE TABLE privilege
 - A storage area

```
CREATE TABLE [schema.]table  
(column datatype [DEFAULT expr][, ...]);
```

- You specify:
 - Table name
 - Column name, column datatype, and column size

Database Creation

- CREATE TABLE is used to describe the layout of a table.
- Typical restrictions are:
 - The table or column name can be no longer than 18 characters.
 - The name must start with a letter.
 - The name can contain letters, numbers, and underscores (_).
 - The name cannot contain spaces.

Oracle Naming Rules

1. Names 1 to 30 characters
2. Names cannot contain quotes
3. Names are not case sensitive
4. Names must begin with an alphabetic character unless surrounded by quotes
5. Names can only contain alphanumerics
 - i. Avoid use of \$ and #
 - ii. Names of database links can contain (.) and (@)
6. Name cannot be a reserve word (e.g. ALTER, CREATE, DATE, TABLE)
7. Avoid using DUAL
8. Avoid use of non reserve keywords (e.g. BACKUP, EXECUTE, ROLES)
9. Name must be unique across name space

Namespaces for Schema Objects

Namespace	Schema Object
Tables	Indexes
Views	
Sequences	Constraints
Private Synonyms	
Stored Procedures	Clusters
Stored Functions	
Packages	Database Triggers
Snapshots	
	Private Database Links

Each schema in the database has its own namespace for objects it contains.

Two tables in different schemas are in different namespaces and can have the same name.

10. Name can be enclosed in double quotation marks. If use quotes, can ignore rules 3-7 but is best to follow them anyway.
 - If use quotes must always use them
 - Use quotes if you wish the name to:
 - Contain spaces
 - Be case sensitive
 - Begin with non-alphabetic
 - Include special characters
 - Use a reserve word

Referencing Another User's Tables

- Tables belonging to other users are not in the user's schema.
- You should use the owner's name as a prefix to the table.

Creating Tables

- Create the table.

```
SQL> CREATE TABLE dept
2          (deptno NUMBER(2),
3          dname  VARCHAR2(14),
4          loc    VARCHAR2(13));
Table created.
```

- Confirm table creation

```
SQL> DESCRIBE dept
```

Name	Null?	Type
DEPTNO		NUMBER (2)
DNAME		VARCHAR2 (14)
LOC		VARCHAR2 (13)

Datatype	Description
VARCHAR2(size)	Variable-length character data
CHAR(size)	Fixed-length character data
NUMBER(p,s)	Variable-length numeric data
DATE	Date and time values
LONG	Variable-length character data up to 2 gigabytes
CLOB	Single-byte character data up to 4 gigabytes
RAW and LONG RAW	Raw binary data
BLOB	Binary data up to 4 gigabytes
BFILE	Binary data stored in an external file; up to 4 gigabytes

The DEFAULT Option

- Specify a default value for a column during an insert.

```
... hiredate DATE DEFAULT SYSDATE, ...
```

- Legal values are literal value, expression, or SQL function.
- Illegal values are another column's name or pseudocolumn.
- The default datatype must match the column datatype.

Tables in the Oracle Database

- User Tables
 - Collection of tables created and maintained by the user
 - Contain user information
- Data Dictionary
 - Collection of tables created and maintained by the Oracle server
 - Contain database information

Querying the Data Dictionary

- Describe tables owned by the user.

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

- View distinct objects owned by the user

```
SQL> SELECT DISTINCT object_type
      2 FROM user_objects;
```

- View tables, views, synonyms and sequences owned by the user

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

Creating a Table by Using a Subquery

- Create a table and insert rows by combining the CREATE TABLE statement and AS *subquery* option.

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

- Match the number of specified columns to the number of subquery columns.
- Define columns with column names and default values.

```
SQL> CREATE TABLE dept30
      2 AS
      3 SELECT empno, ename, sal*12 ANNSAL, hiredate
      4 FROM emp
      5 WHERE deptno = 30;
Table created.
```

```
SQL> DESCRIBE dept30
```

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
ANNSAL		NUMBER
HIREDATE		DATE

The ALTER TABLE Statement

- Use the ALTER TABLE statement to:
 - Add a new column
 - Modify an existing column
 - Define a default value for the new column

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

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

Adding a Column

DEPT30

EMPNO	ENAME	ANNSAL	HIREDATE	JOB
7698	BLAKE	34200	01-MAY-81	
7654	MARTIN	15000	28-SEP-81	
7499	ALLEN	19200	20-FEB-81	
7844	TURNER	18000	08-SEP-81	

Adding a Column

- You use the ADD clause to add columns.

```
SQL> ALTER TABLE dept30
      2 ADD          (job VARCHAR2(9));
Table altered.
```

- The new column becomes the last column

EMPNO	ENAME	ANNSAL	HIREDATE	JOB
7698	BLAKE	34200	01-MAY-81	
7654	MARTIN	15000	28-SEP-81	
7499	ALLEN	19200	20-FEB-81	
7844	TURNER	18000	08-SEP-81	

...

6 rows selected.

Modifying a Column

- You can change a column's datatype, size, and default value.

```
ALTER TABLE dept30
MODIFY      (ename VARCHAR2(15));
Table altered.
```

- A change to the default value affects only subsequent insertions to the table.

Dropping a Table

- All data and structure in the table is deleted.
- Any pending transactions are committed.
- All indexes are dropped.
- You *cannot* roll back this statement.

```
SQL> DROP TABLE dept30;  
Table dropped.
```

Changing the Name of an Object

- To change the name of a table, view, sequence, or synonym, you execute the RENAME statement.

```
SQL> RENAME dept TO department;  
Table renamed.
```

- You must be the owner of the object.

Truncating a Table

- The TRUNCATE TABLE statement:
 - Removes all rows from a table
 - Releases the storage space used by that table

```
SQL> TRUNCATE TABLE department;  
Table truncated.
```

- You cannot roll back row removal when using TRUNCATE.
- Alternatively, you can remove rows by using the DELETE statement.

Adding Comments to a Table

- You can add comments to a table or column by using the COMMENT statement.

```
SQL> COMMENT ON TABLE emp  
2 IS 'Employee Information';  
Comment created.
```

- Comments can be viewed through the data dictionary views.
 - ALL_COL_COMMENTS
 - USER_COL_COMMENTS
 - ALL_TAB_COMMENTS
 - USER_TAB_COMMENTS

Summary

Statement	Description
CREATE TABLE	Creates a table
ALTER TABLE	Modifies table structures
DROP TABLE	Removes the rows and table structure
RENAME	Changes the name of a table, view, sequence, or synonym
TRUNCATE	Removes all rows from a table and releases the storage space
COMMENT	Adds comments to a table or view

Objectives

- Describe constraints
- Create and maintain constraints

What Are Constraints?

- Constraints enforce rules at the table level.
- Constraints prevent the deletion of a table if there are dependencies.
- The following constraint types are valid in Oracle:
 - NOT NULL
 - UNIQUE
 - PRIMARY KEY
 - FOREIGN KEY
 - CHECK

Constraint Guidelines

- Name a constraint or the Oracle Server will generate a name by using the SYS_Cn format.
- Create a constraint:
 - At the same time as the table is created
 - After the table has been created
- Define a constraint at the column or table level.
- View a constraint in the data dictionary.

Defining Constraints

```
CREATE TABLE [schema.]table
    (column datatype [DEFAULT expr]
    [column_constraint],
    ...
    [table_constraint][, ...]);
```

```
CREATE TABLE emp(
    empno    NUMBER(4),
    ename    VARCHAR2(10),
    ...
    deptno   NUMBER(7,2) NOT NULL,
    CONSTRAINT emp_empno_pk
        PRIMARY KEY (EMPNO));
```

- Column constraint level
column [CONSTRAINT *constraint_name*] *constraint_type*,

– Table constraint level

```
column,...
[CONSTRAINT constraint_name] constraint_type
(column, ...),
```

The NOT NULL Constraint

- Ensures that null values are not permitted for the column

EMPNO	ENAME	JOB	...	COMM	DEPTNO
7839	KING	PRESIDENT			10
7698	BLAKE	MANAGER			30
7782	CLARK	MANAGER			10
7566	JONES	MANAGER			20
...					

↑ NOT NULL constraint (no row can contain a null value for this column)
 ↑ Absence of NOT NULL constraint (any row can contain null for this column)
 ↑ NOT NULL constraint

- Defined at the column level

```
SQL> CREATE TABLE emp(
2      empno      NUMBER(4),
3      ename      VARCHAR2(10) NOT NULL,
4      job        VARCHAR2(9),
5      mgr        NUMBER(4),
6      hiredate   DATE,
7      sal        NUMBER(7,2),
8      comm       NUMBER(7,2),
9      deptno     NUMBER(7,2) NOT NULL);
```

The UNIQUE Key Constraint

DEPTNO	DNAME	LOC	
10	ACCOUNTING	NEW YORK	<- DNAME has UNIQUE key constraint
20	RESEARCH	DALLAS	
30	SALES	CHICAGO	
40	OPERATIONS	BOSTON	

Want to insert

```
50 SALES      DETROIT    <- Not allowed
60            BOSTON      <- Allowed
```

- Defined at either the table level or the column level

```
SQL> CREATE TABLE dept(
2      deptno          NUMBER(2),
3      dname           VARCHAR2(14),
4      loc             VARCHAR2(13),
5      CONSTRAINT dept_dname_uk UNIQUE(dname));
```

The PRIMARY KEY Constraint

DEPTNO	DNAME	LOC	
10	ACCOUNTING	NEW YORK	<-DEPTNO is the Primary Key
20	RESEARCH	DALLAS	
30	SALES	CHICAGO	
40	OPERATIONS	BOSTON	

Want to insert

20	MARKETING	DALLAS	<- Not allowed
	FINANCE	NEW YORK	<- Not allowed

- Defined at either the table level or the column level

```
SQL> CREATE TABLE dept(
2      deptno          NUMBER(2),
3      dname           VARCHAR2(14),
4      loc             VARCHAR2(13),
5      CONSTRAINT dept_dname_uk UNIQUE (dname),
6      CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno));
```

The FOREIGN KEY Constraint

DEPT table

DEPTNO	DNAME	LOC	
10	ACCOUNTING	NEW YORK	<- DEPTNO=Primary Key
20	RESEARCH	DALLAS	

EMP table

EMPNO	ENAME	JOB	...	COMM	DEPTNO	
7839	KING	PRESIDENT			10	<- DEPTNO=Foreign Key
7698	BLAKE	MANAGER			30	
...						

7571	FORD	MANAGER	...	200	9	<- not allowed
7571	FORD	MANAGER	...	200	20	<- allowed

↑ Insert into

- Defined at either the table level or the column level

```
SQL> CREATE TABLE emp(
2      empno          NUMBER(4),
3      ename          VARCHAR2(10) NOT NULL,
4      job            VARCHAR2(9),
5      mgr            NUMBER(4),
6      hiredate       DATE,
7      sal            NUMBER(7,2),
8      comm           NUMBER(7,2),
9      deptno         NUMBER(7,2) NOT NULL,
10     CONSTRAINT emp_deptno_fk FOREIGN KEY (deptno)
11                REFERENCES dept (deptno));
```

Keywords

- **FOREIGN KEY**
Defines the column in the child table at the table constraint level
- **REFERENCES**
Identifies the table and column in the parent table
- **ON DELETE CASCADE**
Allows deletion in the parent table and deletion of the dependent rows in the child table

The CHECK Constraint

- Defines a condition that each row must satisfy
- Expressions that are not allowed:
 - References to CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudocolumns
 - Calls to SYSDATE, UID, USER, and USERENV functions
 - Queries that refer to other values in other rows

```
..., deptno NUMBER(2),  
    CONSTRAINT emp_deptno_ck  
    CHECK (DEPTNO BETWEEN 10 AND 99),...
```

Adding a Constraint

```
ALTER TABLE table ADD [CONSTRAINT constraint] type (column);
```

- Add or drop, but not modify, a constraint
 - Enable or disable constraints
 - Add a NOT NULL constraint by using the MODIFY clause
- Add a FOREIGN KEY constraint to the EMP table indicating that a manager must already exist as a valid employee in the EMP table.

```
SQL> ALTER TABLE emp  
2 ADD CONSTRAINT emp_mgr_fk  
3 FOREIGN KEY(mgr) REFERENCES emp(empno);  
Table altered.
```

Dropping a Constraint

- Remove the manager constraint from the EMP table.

```
SQL> ALTER TABLE emp  
2 DROP CONSTRAINT emp_mgr_fk;  
Table altered.
```

- Remove the PRIMARY KEY constraint on the DEPT table and drop the associated FOREIGN KEY constraint on the EMP.DEPTNO column.

```
SQL> ALTER TABLE dept  
2 DROP PRIMARY KEY CASCADE;  
Table altered.
```

Disabling Constraints

- Execute the DISABLE clause of the ALTER TABLE statement to deactivate an integrity constraint.
- Apply the CASCADE option to disable dependent integrity constraints.

```
SQL> ALTER TABLE emp  
2 DISABLE CONSTRAINT emp_empno_pk CASCADE;  
Table altered.
```

Enabling Constraints

- Activate an integrity constraint currently disabled in the table definition by using the ENABLE clause.

```
SQL> ALTER TABLE      emp
      2  ENABLE CONSTRAINT emp_empno_pk;
Table altered.
```

- A UNIQUE or PRIMARY KEY index is automatically created if you enable a UNIQUE key or PRIMARY KEY constraint.

Viewing Constraints

- Query the USER_CONSTRAINTS table to view all constraint definitions and names.

```
SQL>  SELECT      constraint_name, constraint_type,
      2            search_condition
      3  FROM      user_constraints
      4  WHERE table_name = 'EMP';
```

CONSTRAINT_NAME	C SEARCH_CONDITION
SYS_C00674	C EMPNO IS NOT NULL
SYS_C00675	C DEPTNO IS NOT NULL
EMP_EMPNO_PK	P
...	

Viewing the Columns Associated with Constraints

- View the columns associated with the constraint names in the USER_CONS_COLUMNS view.

```
SQL> SELECT  constraint_name, column_name
      2  FROM    user_cons_columns
      3  WHERE   table_name = 'EMP';
```

CONSTRAINT_NAME	COLUMN_NAME
EMP_DEPTNO_FK	DEPTNO
EMP_EMPNO_PK	EMPNO
EMP_MGR_FK	MGR
SYS_C00674	EMPNO
SYS_C00675	DEPTNO

Summary

- Create the following types of constraints:
 - NOT NULL
 - UNIQUE
 - PRIMARY KEY
 - FOREIGN KEY
 - CHECK
- Query the USER_CONSTRAINTS table to view all constraint definitions and names.