CHAPTER 3

Data Definition, Part I

This short chapter is the first one about data definition with SQL. It's intended to get you started using SQL for data retrieval as soon as possible. Therefore, this chapter covers only the data definition basics, such as how to create simple tables using standard datatypes. In Chapter 7, we will revisit data definition with SQL and explore topics such as indexes, synonyms, and constraints.

This chapter is mainly theoretical in nature in that it still offers no hands-on exercises and only a few examples. In the next chapter, you will start writing SQL commands yourself.

The first section introduces the concept of database schemas and database users. In an Oracle database, tables always belong to a schema, and, in general, a schema has a database user as its owner. The second section explains how you can create simple tables, and the most common Oracle datatypes are covered in the third section. To illustrate the contents of the first three sections, the fourth section shows the `CREATE TABLE` commands to create the sample tables used in the examples in this book (introduced in the previous chapter), without bothering about constraints yet.

The last section of this chapter covers the Oracle data dictionary. It provides a global overview of the data dictionary, lists some typical examples of data dictionary tables, and shows how to execute some simple queries against some of those data dictionary tables.

3.1 Schemas and Users

Before you can start creating and populating tables with SQL, you need to understand how data stored in an Oracle database is organized internally. In the previous chapter, you learned that you cannot do anything in an Oracle database if you do not identify yourself first by specifying a *username* and a *password*. This process identifies you as a certain *database user*.

In an Oracle database there is, in general, a one-to-one relationship between database *users* and database *schemas* with the same name. Briefly, these are the differences between a database user and a database schema:

- A *database user* has a password and certain database privileges.
- A *database schema* is a logical collection of database objects (such as tables, indexes, views, and so on) that is usually owned by the user of the same name.

Normally, when you log on to an Oracle database, you are automatically connected with the corresponding database schema with the same name. However, it is also possible that certain database users don't have their own schema; in other words, they don't have any

database objects of their own, and they don't have the privileges to create them either. These "schema-less" users are, for example, authorized only to retrieve or manipulate data in a different database schema.

For example, in SQL*Plus, you can use the CONNECT command to establish a new connection with a different schema, provided you are able to enter a valid combination of a database name and a corresponding password. With the ALTER SESSION SET CURRENT_SCHEMA command, you can "visit" a different schema in SQL*Plus without changing your identity as database user, and therefore without changing any of your privileges.

All of the examples and exercises in this book assume the presence of a database user BOOK, with the password BOOK, and a schema BOOK that contains the seven case tables introduced in the previous chapter. You can find all of the scripts to create the BOOK schema, to create the seven tables, and to insert the rows on my web site at http://www.naturaljoin.nl or the Downloads section of the Apress web site (http://www.apress.com).

3.2 Table Creation

The SQL command to create tables is CREATE TABLE. If you create a table, you must specify a name for the new table, followed by a specification of all table columns. The columns must be specified as a comma-separated list between parentheses.

■ **Note** The right to create tables in an Oracle database is not granted to everyone; you need some additional system privileges. If you get error messages when you try to create tables, contact your database administrator or check *Oracle Database Administrator's Guide* in the online documentation.

The basic syntax of the CREATE TABLE command is shown in Figure 3-1.

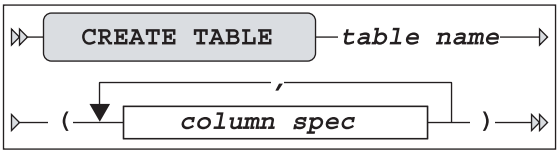

Figure 3-1. *CREATE TABLE basic command syntax diagram*

■ **Note** Figure 3-1 does *not* show the complete syntax of the CREATE TABLE command. Just for fun, check out *Oracle SQL Reference* for the amount of documentation describing the CREATE TABLE command. Chapter 7 of this book will revisit this command with the full syntax and more details.

Column specifications normally consist of several components. Figure 3-2 shows the column specification syntax.

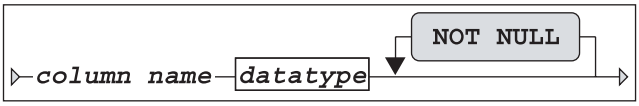

Figure 3-2. *Column specification syntax diagram*

Each column specification starts with a column name, followed by the datatype (discussed in the next section). If you add the optional expression `NOT NULL` to a column definition, each future row of the table you are creating *must* have a value specified for this column, and you will not be able to update future rows by removing a value for this column. In other words, you define the column to be a mandatory attribute.

The `NOT NULL` addition is an example of a constraint. You can specify many additional constraints in the `CREATE TABLE` command, such as `UNIQUE`, `CHECK`, `PRIMARY KEY`, and `FOREIGN KEY`. Chapter 7 will discuss these options of the `CREATE TABLE` command.

3.3 Datatypes

Oracle supports many standard datatypes, as you will see if you take a look at the Oracle documentation. Some Oracle datatypes look very similar; some are even synonyms for each other. These datatypes are supported for compatibility purposes of Oracle with other DBMSs or with the ANSI/ISO SQL standard. For example, `INT` and `INTEGER` are synonyms for `NUMBER(38)`. Some datatypes are very specific in nature, making them irrelevant for us at this point in time. This section covers only the most common and widely used Oracle datatypes.

In general, there are three categories of column data: numbers (numeric data), text (alphanumeric data), and time-related data. The most important corresponding Oracle datatypes are `NUMBER`, `VARCHAR` or `VARCHAR2`, and `DATE`, respectively.

Table 3-1 shows some examples of the `NUMBER` datatype.

Table 3-1. *NUMBER Datatype Examples*

Example	Description
`NUMBER(4)`	An integer with a maximum length of four digits
`NUMBER(6,2)`	A number with a maximum precision of six digits; at most two digits behind the decimal point
`NUMBER(7,-3)`	A multiple of thousand with at most seven digits
`NUMBER`	Identical to `NUMBER(38,*)`
`NUMBER(*,5)`	Identical to `NUMBER(38,5)`

Oracle offers a number of alphanumeric datatypes. Depending on the Oracle version you are using, there are some differences due to the evolution of the ANSI/ISO SQL standard over the years. For example, since Oracle7, the two datatypes `VARCHAR` and `VARCHAR2` are identical, but this could change in a future Oracle release. If you create a table and you use the `VARCHAR` datatype, the Oracle DBMS translates `VARCHAR` to `VARCHAR2` on the fly. Therefore, this book refers to only the `VARCHAR2` datatype. In cases where the maximum size of the `VARCHAR2` datatype (4000) is insufficient for a specific column, you can use the `CLOB` (Character Large OBject) datatype.

Table 3-2 shows some simple examples of character datatypes.

Table 3-2. *Character Datatype Examples*

Example	Description
`VARCHAR2(25)`	Alphanumeric, *variable* length, up to 25 characters
`CHAR(4)`	Alphanumeric, *fixed* length, four characters
`CLOB`	Alphanumeric, larger than the maximum size of the `VARCHAR2` datatype

Table 3-3 lists the maximum size values for the datatypes mentioned so far.

■ **Note** The actual units of measure used for the size of `CHAR` and `VARCHAR2` datatypes depend on character semantics (bytes or characters). See Chapter 7 for details.

Table 3-3. *Maximum Datatype Sizes*

Datatype	Maximum Size
`NUMBER`	38 digits precision
`CHAR`	2000
`VARCHAR2`	4000
`CLOB`	4GB

■ **Note** The indicated maximum `CLOB` size (4GB) is not completely correct. Depending on some configuration parameters, `CLOB` columns may contain much more than 4GB worth of data. Refer to *Oracle SQL Reference* for details.

The basic datatype for time-related data is `DATE`. By default, date values are interpreted and displayed according to a standard date format, typically showing only the day, the month, and the last two digits of the year. You can change the default date format for your session or use conversion functions in your SQL commands to display dates in different ways. Internally, Oracle stores dates in such a way that `DATE` column values are allowed from the year 4712 BC until the year 9999. Oracle dates are internally stored with much more precision than you might expect on first consideration.

■ **Caution** `DATE` columns also contain a time indication (hours, minutes, and seconds), which may cause problems when comparing two dates. For example, seemingly equal dates could be different due to their invisible time components.

Apart from the DATE datatype, Oracle also supports the related datatypes TIMESTAMP (with or without TIME ZONE) and INTERVAL to store other time-related data in table columns. Chapter 7 provides more details on the time-related datatypes.

This book focuses on the usage of the three standard Oracle datatypes: NUMBER, VARCHAR2, and DATE.

3.4 Commands for Creating the Case Tables

This section lists the SQL commands to create the seven case tables introduced in Chapter 1, as an illustration of the concepts covered in the previous three sections, without much additional explanation. Since the BOOK schema consists of seven tables, this section also shows seven CREATE TABLE commands, presented in Listings 3-1 through 3-7.

■ **Note** As mentioned earlier, constraint definition (and constraint checking) is not taken into consideration in this chapter; therefore, the following listings do *not* show the complete commands to create the case tables.

Listing 3-1. *The EMPLOYEES Table*

```
SQL> create table  EMPLOYEES
  2  ( empno       number(4)    not null
  3  , ename       varchar2(8)  not null
  4  , init        varchar2(5)  not null
  5  , job         varchar2(8)
  6  , mgr         number(4)
  7  , bdate       date         not null
  8  , msal        number(6,2)  not null
  9  , comm        number(6,2)
 10  , deptno      number(2)             );
```

Listing 3-2. *The DEPARTMENTS Table*

```
SQL> create table  DEPARTMENTS
  2  ( deptno      number(2)    not null
  3  , dname       varchar2(10) not null
  4  , location    varchar2(8)  not null
  5  , mgr         number(4)             );
```

Listing 3-3. *The SALGRADES Table*

```
SQL> create table  SALGRADES
  2  ( grade       number(2)    not null
  3  , lowerlimit  number(6,2)  not null
  4  , upperlimit  number(6,2)  not null
  5  , bonus       number(6,2)  not null );
```

Listing 3-4. *The COURSES Table*

```
SQL> create table  COURSES
  2  ( code        varchar2(6)  not null
  3  , description varchar2(30) not null
  4  , category    char(3)      not null
  5  , duration    number(2)    not null );
```

Listing 3-5. *The OFFERINGS Table*

```
SQL> create table  OFFERINGS
  2  ( course      varchar2(6)  not null
  3  , begindate   date         not null
  4  , trainer     number(4)
  5  , location    varchar2(8)           );
```

Listing 3-6. *The REGISTRATIONS Table*

```
SQL> create table  REGISTRATIONS
  2  ( attendee    number(4)    not null
  3  , course      varchar2(6)  not null
  4  , begindate   date         not null
  5  , evaluation  number(1)             );
```

Listing 3-7. *The HISTORY Table*

```
SQL> create table  HISTORY
  2  ( empno       number(4)    not null
  3  , beginyear   number(4)    not null
  4  , begindate   date         not null
  5  , enddate     date
  6  , deptno      number(2)    not null
  7  , msal        number(6,2)  not null
  8  , comments    varchar2(60)          );
```

3.5 The Data Dictionary

If you are interested in knowing which tables are present in your database, which columns they have, whether or not those columns are indexed, which privileges are granted to you, and similar information, you should query the *data dictionary*. Another common term for data dictionary is *catalog*. By the way, we already queried the data dictionary implicitly before, in Chapter 2, when using the SQL*Plus `DESCRIBE` command; this command queries the data dictionary under the hood.

The data dictionary is more or less the internal housekeeping administration of Oracle. The data dictionary stores information about the data, also referred to as *metadata*. The data dictionary is automatically maintained by Oracle; therefore, the data dictionary is always up-to-date.

DBMSs like Oracle store data dictionary data in precisely the same way as they store "regular" data: in tables. This is in compliance with Ted Codd's rule 4 (see Chapter 1). The big advantage of this approach is that you can use the SQL language to query data dictionary data in the same way that you query ordinary data. In other words, if you master the SQL language, you need to know only the names of the data dictionary tables and the names of their columns.

Data dictionary access is a potential security risk. That's why the Oracle DBMS offers system privileges and roles to regulate and protect access to the data dictionary. For example, there is a role `SELECT_CATALOG_ROLE`, which contains all privileges that you need to be able to access the data dictionary data. Listing 3-8 demonstrates how Oracle controls data dictionary access.

Listing 3-8. *Needing the SELECT_CATALOG_ROLE Role*

```
SQL> describe dba_sys_privs
ERROR:
ORA-04043: object "SYS"."DBA_SYS_PRIVS" does not exist

SQL> connect / as sysdba
Connected.

SQL> grant select_catalog_role to book;
Grant succeeded.

SQL> connect book/book
Connected.

SQL> desc dba_sys_privs
 Name                          Null?    Type
 ----------------------------- -------- ---------------
 GRANTEE                       NOT NULL VARCHAR2(30)
 PRIVILEGE                     NOT NULL VARCHAR2(40)
 ADMIN_OPTION                           VARCHAR2(3)

SQL>
```

Although the information is stored in data dictionary *tables*, most of the time, you access data dictionary *views* instead. On the other hand, views are tables anyway. See Chapter 10 for details about views.

You can refer to *Oracle Server Reference* in the Oracle documentation to get a complete overview of the Oracle data dictionary. Fortunately, the Oracle data dictionary contains a view that lists all Oracle data dictionary views, with a short description of their contents. This view is called DICTIONARY; DICT is a shorter synonym for the same view. Listing 3-9 shows an abbreviated version of the query results. It's abbreviated for a practical reason: the DICT view contains more than 600 rows!

Listing 3-9. *Using the DICT View*

```
SQL> col COLUMN_NAME format a30
SQL> col COMMENTS    format a40 word
SQL>
SQL> select * from dict order by table_name;

TABLE_NAME           COMMENTS
-------------------- ----------------------------------------
ALL_ALL_TABLES       Description of all object and relational
                     tables accessible to the user
ALL_APPLY            Details about each apply process that
                     dequeues from the queue visible to the
                     current user
...
USER_COL_COMMENTS    Comments on columns of user's tables and
                     views
USER_COL_PRIVS       Grants on columns for which the user is
                     the owner, grantor or grantee
...
V$TIMEZONE_NAMES     Synonym for V_$TIMEZONE_NAMES
V$VERSION            Synonym for V_$VERSION

610 rows selected.

SQL>
```

Data dictionary view names typically have prefixes that suggest the existence of four main categories. In Listing 3-9, you can see the ALL, USER, and V$ prefixes. The fourth common prefix is DBA. The idea behind this is that, most of the time, you are interested in information about a certain subcategory of database objects. By using the appropriate views, you automatically suppress information that is not of interest to you. Also, depending on your database privileges, you will not be allowed to use certain categories of data dictionary views. Table 3-4 lists the most common data dictionary view name prefixes. (Note that not all data dictionary views have one of these prefixes.)

Table 3-4. *Common Data Dictionary View Prefixes*

Prefix	Description
USER_...	Information about your own objects
ALL_...	Information about all objects you can access
DBA_...	All information in the database; for database administrators only
[G]V$...	Dynamic performance views; for database administrators only

The *dynamic performance views* (those with a V$ or GV$ name prefix) are a special category. These views are not based on database tables, but rather on information from other sources such as internal memory structures. They are mainly relevant for, and accessible to, database administrators.

Most data dictionary view names give a clear indication of their contents; however, as a consequence, some of these names are very long. That's why some of the most popular data dictionary views also have alternative (shorter) synonyms, such as CAT, OBJ, IND, TABS, and COLS. The CAT view is an especially useful one, because it lists the objects in the current schema. Listing 3-10 shows an example of using the CAT view with our BOOK schema.

Listing 3-10. *Using the CAT View*

```
SQL> select * from cat;

TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
EMPLOYEES                      TABLE
DEPARTMENTS                    TABLE
SALGRADES                      TABLE
COURSES                        TABLE
OFFERINGS                      TABLE
REGISTRATIONS                  TABLE
HISTORY                        TABLE

7 rows selected.

SQL>
```

Suppose you want to query a specific data dictionary view, and you don't know the actual column names of that view. In that case, you can use the SQL*Plus command DESCRIBE, just as you would do for regular tables. As you can see in Listing 3-11, you can use the DESCRIBE command, or you can query the data dictionary view DICT_COLUMNS.

Listing 3-11. *Using the DESCRIBE Command and the DICT_COLUMNS View*

```
SQL> describe DICT_COLUMNS
 Name                      Null? Type
 ------------------------- ----- --------------
 TABLE_NAME                      VARCHAR2(30)
```

```
 COLUMN_NAME                    VARCHAR2(30)
 COMMENTS                       VARCHAR2(4000)

SQL> select column_name, comments
  2  from   dict_columns
  3  where  table_name = 'ALL_USERS';

COLUMN_NAME                COMMENTS
-------------------------- -------------------------
USERNAME                   Name of the user
USER_ID                    ID number of the user
CREATED                    User creation date

SQL>
```

Listing 3-12 shows a query against the NLS_SESSION_PARAMETERS view (NLS stands for National Language Support). The result shows, for example, the NLS_DATE_FORMAT value used to display dates.

Listing 3-12. *Using the NLS_SESSION_PARAMETERS View*

```
SQL> select * from nls_session_parameters

PARAMETER                VALUE
------------------------ ----------------------
NLS_LANGUAGE             AMERICAN
NLS_TERRITORY            AMERICA
NLS_CURRENCY             $
NLS_ISO_CURRENCY         AMERICA
NLS_NUMERIC_CHARACTERS   .,
NLS_CALENDAR             GREGORIAN
NLS_DATE_FORMAT          DD-MON-YYYY
NLS_DATE_LANGUAGE        AMERICAN
NLS_SORT                 BINARY
NLS_TIME_FORMAT          HH.MI.SSXFF AM
NLS_TIMESTAMP_FORMAT     DD-MON-RR HH.MI.SSXFF AM
NLS_TIME_TZ_FORMAT       HH.MI.SSXFF AM TZR
NLS_TIMESTAMP_TZ_FORMAT DD-MON-RR HH.MI.SSXFF AM TZR
NLS_DUAL_CURRENCY        $
NLS_COMP                 BINARY
NLS_LENGTH_SEMANTICS     BYTE
NLS_NCHAR_CONV_EXCP      FALSE

17 rows selected.

SQL>
```

The NLS features in Oracle are documented in great detail in *Globalization Support Guide*.

Table 3-5 lists a selection of useful Oracle data dictionary tables.

Table 3-5. *Some Useful Oracle Data Dictionary Views*

View	Description
DICTIONARY	Description of the data dictionary itself
DICT_COLUMNS	Data dictionary column descriptions
ALL_USERS	Information about all database users
ALL_INDEXES[1]	All indexes
ALL_SEQUENCES[1]	All sequences
ALL_OBJECTS[1]	All objects
ALL_SYNONYMS[1]	All synonyms
ALL_TABLES[1]	All tables
ALL_VIEWS[1]	All views
USER_INDEXES[2]	Indexes
USER_SEQUENCES[2]	Sequences
USER_OBJECTS[2]	Objects
USER_SYNONYMS[2]	Synonyms
USER_TABLES[2]	Tables
USER_TAB_COLUMNS[2]	Columns
USER_VIEWS[2]	Views
USER_RECYCLEBIN	Dropped objects
CAT	Synonym for USER_CATALOG
COLS	Synonym for USER_TAB_COLUMNS
DICT	Synonym for DICTIONARY
DUAL	Dummy table, with one row and one column
IND	Synonym for USER_INDEXES
OBJ	Synonym for USER_OBJECTS
SYN	Synonym for USER_SYNONYMS
TABS	Synonym for USER_TABLES

[1] *Accessible to the user*
[2] *Owned by the user*

Appendix B provides a more complete description of the data dictionary views, and *Oracle Server Reference* provides all the details you need about the Oracle data dictionary.