Chapter 6:

The tables and constraints you created in previous chapters are considered database

objects. A database object is anything with a name and a defined structure.

Three other database objects commonly used in Oracle 12c are *sequences*, *indexes*, and *synonyms*, which you examine in this chapter. The following list identifies the role of each object:

• A ***Sequence***: generates sequential integers that organizations can use to assist with internal controls or simply to serve as primary keys for tables.

• A ***Database*** ***index*** serves the same basic purpose as an index in a book, allowing users to locate specific records quickly.

• A ***synonym*** is a simpler name, like a nickname, given to an object with a complex name or to provide an alternative name for identifying database objects. Synonyms can simplify referencing objects with complex names and objects that are moved to different schemas.

# S E Q U E N C E S

1. **CREATE SEQUENCE sequencename**

[INCREMENT BY value]

[START WITH value]

[{MAXVALUE value | NOMAXVALUE}]

[{MINVALUE value | NOMINVALUE}]

[{CYCLE | NOCYCLE}]

[{ORDER | NOORDER}]

[{CACHE value | NOCACHE}];

1. **Alter a sequence ALTER SEQUENCE sequencename**

[INCREMENT BY value]

[{MAXVALUE value | NOMAXVALUE}]

[{MINVALUE value | NOMINVALUE}]

[{CYCLE | NOCYCLE}]

[{ORDER | NOORDER}]

[{CACHE value | NOCACHE}];

1. **Drop a sequence**

DROP SEQUENCE sequencename;

1. **Create a B-tree index**

CREATE INDEX indexname

ON tablename(columnname,. . .);

1. **Create a bitmap index**

CREATE BITMAP INDEX indexname

ON tablename (columnname, . . .);

1. **Create a function-based index**

CREATE INDEX indexname

ON tablename (expression);

A sequence is a database object you can use to generate a series of integers. These integers are most commonly used to generate a unique primary key for each record or for internal control purposes. A brief overview of these two concepts follows. When you use values generated by a sequence as a primary key, there’s no true correlation between the number assigned to a record and the entity it represents. However, depending on the parameters used to create the sequence, database users can be assured that no two records have the same primary key value. Ensuring that primary key values aren’t duplicated is especially important if different users are assigned the task of entering records in a database table because they might attempt to assign the same primary key value to different records. For example, if several customer service representatives are entering new customers at the same time, how are customer numbers assigned? Are all the customer service representatives in the same room, asking one another “What number did your last customer receive?” Not likely. Chances are that they’re using a sequence, so each customer service representative can be certain that every customer number is unique.

A sequence can also be used to provide business and auditing controls. Every organization should have some control mechanisms to avoid problems with transaction auditing, embezzlement, and accounting errors. Most organizations use sequential numbers to track checks, purchase orders, invoices, and anything else used to record financial events. With sequential numbers, an auditor can determine whether items such as checks or invoices are missing, which can reveal accounting problems—unrecorded transactions or employees obtaining blank checks or invoices for their own use.

• Used for internal control purposes by providing sequential integers for auditing

• Used to generate unique value for primary key column –

Surrogate key = no correlation with actual row contents

1. **Creating a Sequence**

•Use the **CREATE SEQUENCE** command

•Various intervals are allowed – **Default**: 1

•You can specify the starting number – Default: 1

•Can specify **MINVALUE** for decreasing sequence and **MAXVALUE** for increasing sequence

•Numbers can be reused if **CYCLE** is specified

•**ORDER** clause is used in application cluster environment

•Use **CACHE** to pre generate integers – Default: 20

•To verify the settings for options of a sequence, query **USER**\_**SEQUENCES** data dictionary view

•**NEXTVAL** – generates integer URRVAL – contains last integer generated by NEXTVAL

•Set column **DEFAULT** value

1. **Altering Sequence Definitions**

•Use **ALTER SEQUENCE** command to change the settings for a sequence

•**START WITH** value cannot be altered – drop the sequence and re-create it

•Changes cannot make current integers invalid

1. **Removing a Sequence**

•Use the **DROP SEQUENCE** command to delete a sequence.

•Previous values generated are not affected by removing a sequence from a database

1. **Create an Identity Column**

* Alternative to using sequences to populate
* primary key columns

I N D E X E S

An Oracle 12c **index** is a database object that **stores a map of column values** and the ROWIDs of matching table rows. A ROWID is the physical address of a table row. **A database index is much like the index at the end of this textbook.** If you look up a topic such as “primary key,” you can scan the handful of alphabetically sorted index pages quickly to determine the page location for this topic. If no index existed, you would need to scan the entire textbook to find references to this topic, which could be time consuming and tedious. In a similar way, **database indexes make data retrieval more efficient**. A common challenge in managing databases is improving data retrieval speed. As tables become populated with many rows, processing the operations involved in query searches (WHERE conditions) and sorting (ORDER BY and joins) takes increasing amounts of time. Much of this increase in execution time is caused by disk I/O or disk reads (reading data from physical disk drives). **CREATE TABLE** statement in Oracle 12c by default creates a heap-organized table, which is an unordered collection of data. As rows of data are inserted, they’re physically added to the table in no particular order. As rows are deleted, the space can be

• An index stores frequently referenced values and ROWIDs

• Can be based on one column, multiple columns, functions, or expressions

Index Organized Tables

• An IOT stores table contents in a B-tree index structure

• Use the “ORGANIZATION INDEX” option in a CREATE TABLE statement to build an IOT

• Use the **USER\_INDEXES** data dictionary view to determine that the index exists

• Use the **USER\_IND\_COLUMNS** data dictionary view to determine the column index information

• Use the **DROP INDEX** command to remove an index