3

Indexes and Index-Organized Tables

Indexes are schema objects that can speed access to table rows. Index-organized tables are tables stored in an index structure.

This chapter contains the following sections:

Introduction to Indexes

53

An **index** is an optional structure, associated with a table or **table cluster**, that can sometimes speed data access.

Indexes are schema objects that are logically and physically independent of the data in the objects with which they are associated. Thus, you can drop or create an index without physically affecting the indexed table.

Note:

If you drop an index, then applications still work. However, access of previously indexed data can be slower.

For an analogy, suppose an HR manager has a shelf of cardboard boxes. Folders containing employee information are inserted randomly in the boxes. The folder for employee Whalen (ID 200) is 10 folders up from the bottom of box 1, whereas the folder for King (ID 100) is at the bottom of box 3. To locate a folder, the manager looks at every folder in box 1 from bottom to top, and then moves from box to box until the folder is found. To speed access, the manager could create an index that sequentially lists every employee ID with its folder location:

```
ID 100: Box 3, position 1 (bottom)
ID 101: Box 7, position 8
ID 200: Box 1, position 10
.
.
.
```

NOT NULL constraints are intended for columns that must not lack values. For example, the hr.employees table requires a value in the email column. An attempt to insert an employee row without an email address generates an error:

```
SQL> INSERT INTO hr.employees (employee_id, last_name) values (999,
'Smith');
.
.
.
ERROR at line 1:
ORA-01400: cannot insert NULL into ("HR"."EMPLOYEES"."EMAIL")
```

You can only add a column with a NOT NULL constraint if the table does not contain any rows or if you specify a default value.

See Also:

- *Oracle Database 2 Day Developer's Guide* for examples of adding NOT NULL constraints to a table
- *Oracle Database SQL Language Reference* for restrictions on using NOT NULL constraints
- *Oracle Database Development Guide* to learn when to use the NOT NULL constraint

Unique Constraints

53

A **unique key constraint** requires that every value in a column or set of columns be unique. No rows of a table may have duplicate values in a single column (the **unique key**) or set of columns (the **composite unique key**) with a unique key constraint.

Note:

The term *key* refers only to the columns defined in the integrity constraint. Because the database enforces a unique constraint by implicitly creating or reusing an index on the key columns, the term *unique key* is sometimes incorrectly used as a synonym for *unique key constraint* or *unique index*.

Unique key constraints are appropriate for any column where duplicate values are not allowed. Unique constraints differ from primary key constraints, whose purpose is to identify each table row uniquely, and typically contain values that have no significance other than being unique. Examples of unique keys include:

- A customer phone number, where the primary key is the customer number
- A department name, where the primary key is the department number

As shown in Example 2-1, a unique key constraint exists on the email column of the hr.employees table. The relevant part of the statement is as follows:

- In some cases, as when you create a primary key with a deferrable constraint, the generated index is not unique.

  Note:

  You can explicitly create a unique index with the `CREATE UNIQUE INDEX` statement.

- If a usable index exists when a primary key constraint is created, then the constraint reuses this index and does not implicitly create one.

By default the name of the implicitly created index is the name of the primary key constraint. You can also specify a user-defined name for an index. You can specify storage options for the index by including the `ENABLE` clause in the `CREATE TABLE` or `ALTER TABLE` statement used to create the constraint.

See Also:

Oracle Database 2 Day Developer's Guide and *Oracle Database Development Guide* to learn how to add primary key constraints to a table

Foreign Key Constraints

53

Whenever two tables contain one or more common columns, Oracle Database can enforce the relationship between the two tables through a **foreign key constraint**, also called a *referential integrity constraint*.

A foreign key constraint requires that for each value in the column on which the constraint is defined, the value in the other specified other table and column must match. An example of a referential integrity rule is an employee can work for only an existing department.

The following table lists terms associated with referential integrity constraints.

Table 5-2 Referential Integrity Constraint Terms

Term	Definition
Foreign key	The column or set of columns included in the definition of the constraint that reference a referenced key. For example, the `department_id` column in `employees` is a foreign key that references the `department_id` column in `departments`. Foreign keys may be defined as multiple columns. However, a composite foreign key must reference a composite primary or unique key with the same number of columns and the same data types. The value of foreign keys can match either the referenced primary or unique key value, or be null. If any column of a composite foreign key is null, then the non-null portions of the key do not have to match any corresponding portion of a parent key.