

---

## 5. CHANGING STRUCTURE AND DATA

- ❑ Altering the structure of a table
- ❑ Dropping a table
- ❑ Manipulating data
- ❑ Transaction
- ❑ Locking
- ❑ Read Consistency
- ❑ Summary
- ❑ Exercises

### Altering the structure of a table

It is possible to modify the structure of the table even after the table is created. ALTER TABLE command is used to alter the structure of the table.

The following are the possible alterations

- ❑ Adding a new column
- ❑ Adding a new table level constraint
- ❑ Increasing and decreasing width of a column
- ❑ Changing data type of the column
- ❑ Dropping a column
- ❑ Dropping a constraint
- ❑ Disabling a constraint
- ❑ Enabling a disabled constraint

The following is the syntax of ALTER TABLE command.

**ALTER TABLE Syntax**

```
ALTER TABLE tablename
  [ ADD      ( column specification ) ]
  [ MODIFY  ( column specification ) ]
  [ DROP    constraint-name [CASCADE]
      | column [CASCADE CONSTRAINTS] ]
  [ DROP UNUSED COLUMN [column] ]
  [SET UNUSED column]
  [ ENABLE | DISABLE constraint-name ]
```

Let us now examine various examples using ALTER TABLE command. It is important to note that some possibilities are applicable only when certain conditions are met. These if and buts are to be remembered while modifying the structure of the table.

**Adding a new column or constraint**

It is always possible to add a new column to an existing table. However, if column is to be added to a table that already contains rows, then certain options are not available.

**To add a new column CAT (category) to COURSES table, enter the following:**

```
ALTER TABLE courses
  ADD (cat varchar2(5));
```

It is not possible to given any constraint that is not satisfied by existing data. For instance, it is not possible to add CAT as a NOT NULL column as Oracle initializes CAT column in all rows with NULL value. See the following snapshot of the screen.

```
SQL> alter table courses
      2  add ( cat varchar2(5) not null);
alter table courses
      *
ERROR at line 1:
ORA-01758: table must be empty to add mandatory (NOT NULL) column
```

However, it is possible to have NOT NULL constraint in the following cases:

- ❑ If DEFAULT option is used to specify the default value for the column
- ❑ When table is empty.

The following example will work as DEFAULT option is used to specify the default value.

```
alter table courses add ( cat varchar2(5) default 'prog' not null);
```

You can add a table constraint. Once table is created, it is not possible to add constraints other than NOT NULL to columns of the table. However, it is possible to add any constraint at the table level as follows:

```
alter table courses
    add ( constraint courses_cat_chk check (length(cat) >= 2))
```

We will see more about functions such as **length** later in this book. The above constraint specifies that the column CAT should have at least two characters. It is added as a table constraint as it not possible to add CHECK constraint at the column level to column that already exists.

### Modifying attributes of existing columns

It is also possible to modify the certain attributes of an existing column. The following are possible modifications.

- ❑ Increasing the length of the column
- ❑ Decrease the length of the column only when column is empty.
- ❑ Change the datatype of the column only when column is empty.
- ❑ Adding NOT NULL constraint. No other constraint can be added to column. However, it is possible to add constraints at table level. See the previous section.

To increase the size of the column CAT, enter the following:

```
alter table courses
    modify (cat varchar2(10));
```

---

You can decrease the width and even change the datatype of the column if the column is empty. That means if no row has any value for the column being altered.

### Dropping a constraint

Used to drop the constraints defined on the table.

To drop a constraint, the name of the constraint is required. You may use USER\_CONSTRAINTS data dictionary view to get the list of constraints.

```
alter table courses drop constraint courses_cat_chk;
```

### CASCADE Option

You cannot drop a UNIQUE or PRIMARY KEY constraint that is part of a referential integrity constraint without also dropping the corresponding foreign key. To drop PRIMARY KEY or UNIQUE constraint along with REFERENCES constraint use CASCADE option.

To drop PRIMARY KEY constraint of STUDENTS along with related constraint, do the following.

```
alter table courses
    drop primary key cascade;
```

---

**Note:** You can get information about all the constraint using USER\_CONSTRAINTS data dictionary view.

---

### Dropping a column

For the first time Oracle8i has provided a command to drop a column from the table. Till Oracle8, dropping a column is very lengthy task as there was no direct way to drop a column. Oracle8i has provided a new option with ALTER TABLE command to drop a column – DROP COLUMN.

Actual you have two options when you want to drop a column.

- ❑ Either you can drop unwanted column straight away. All the data related to the column being dropped will be removed immediately.
- ❑ Or you can mark the column for deletion and delete the column at a later stage. Since the column is marked for dropping, it is considered to be dropped. But the data of the column will remain until the column is physically removed.

The second options is especially useful considering the fact that dropping of a column does take a lot of time and if it is done when the load on the system is high then it will severely effect performance of the system. So you can mark a column for dropping and then drop the column when load on the system is low.

To drop column CAT of COURSES table, enter the following:

```
alter table courses drop column cat;
```

If column being dropped is either a PRIMARY KEY or UNIQUE key that is referenced by a foreign key, then it is not possible to drop the column. But it is possible if CASCADE CONSTRAINTS option is used. CASCADE CONSTRAINTS option drops all constraints that depend on the column being dropped.

To drop column FCODE column of FACULTY table along with all depending constraints, enter:

```
alter table faculty drop column fcode cascade constraints;
```

---

**Note:** When you drop a UNIQUE or PRIMARY KEY column then Oracle automatically drops the index that it creates to enforce uniqueness.

---

SET UNUSED option of ALTER TABLE command is used to mark a column for dropping. But the column is not physically removed. However, the column is treated as deleted. Once a column is marked as UNUSED then it cannot be accessed.

The following example marks column CAT or COURSES table as unused.

```
alter table courses set unused column cat;
```

Columns that are marked for deletion can be physically deleted using DROP UNUSED COLUMNS option of ALTER TABLE command as follows:

```
alter table courses drop unused columns;
```

---

**Note:** We can view the number of columns that are marked for deletion using USER\_UNUSED\_COL\_TABS data dictionary view. USER\_TAB\_COLUMNS gives information about existing columns of a table.

---

---

**Note:** Until a column is physically dropped from the table it is counted as a column of the table and counted towards the absolute limit of 1000 columns per table.

---

### Enabling and Disabling Constraints

ALTER TABLE can be used to enable and disable constraints without dropping constraints. When a constraint is disabled, Oracle does not enforce the rule defined by constraint. This may be useful when you need to insert a large number of rows and does not want Oracle to apply constraints as it takes a lot of time.

To disable PRIMARY KEY on SUBJECTS table:

```
ALTER TABLE courses DISABLE PRIMARY KEY;
```

Or you can drop any constraint by giving its name as follows:

```
alter table courses disable constraint courses_cat_chk;
```

If the constraint has depending constraints then you must use CASCADE clause to disable dependent constraints.

You can enable a disabled constraints using ENABLE clause as follows:

```
alter table courses disable constraint courses_cat_chk;
```

---

**Note:** You can find out status of a constraint by using STATUS column of USER\_CONSTRAINTS data dictionary view.

---

### Dropping a table

To drop a table, use DDL command DROP TABLE. It removes the data as well as structure of the table. The following is the syntax of DROP TABLE command.

```
DROP TABLE tablename [CASCADE CONSTRAINTS];
```

**CASCADE CONSTRAINTS** clause is used to drop constraints that refer to primary and unique keys in the dropped table. If you do not give this clause and if referential integrity (references constraint) constraints exist then Oracle displays an error and doesn't drop the table.

The following command will drop FACULTY table.

```
DROP TABLE faculty;
```

---

**Note:** When table is dropped, Views, and Synonyms based on the table will be made invalid, though they remain in the system. Also note, dropping a table cannot be undone.

---

## Manipulating data

As we have seen in the first chapter, SQL commands are divided into DML commands and DDL commands. DDL commands such as CREATE TABLE, ALTER TABLE, and DROP TABLE are dealing with definition of the table or structure of the table.

DML commands such as INSERT, DELETE and UPDATE are used to manipulate the data of the table. We have already seen how to use INSERT command to insert rows into table. Now let us see two other DML commands.

### Updating rows using UPDATE command

UPDATE command is used to modify existing data in the rows. The following is the syntax of UPDATE command.

```
UPDATE table SET column = {expression | subquery}
                [, column = {expression | subquery}] ...
[WHERE condition];
```

If WHERE clause is not given then all the rows of the table will be effected by the change. In fact, it is more often the result of an error than intentional.

The following command will change course fee of ASP to 6000.

```
Update courses set fee = 6000
Where ccode = 'asp';
```

It is also possible to change more than one column at a time as follows:

```
update courses set fee = 6000, duration=30
where ccdoe = 'asp';
```

---

**Note:** We will discuss how to use **subquery** in UPDATE command later in this book.

---

## Deleting rows using DELETE command

DELETE command is used to delete rows from a table. The following is the syntax of DELETE command.

```
DELETE FROM table
      [WHERE condition;]
```

If WHERE clause is not given then all rows of the table will be deleted.

The following command will delete row where CCODE is "c".

```
Delete from courses
Where ccode = 'c';
```

It is not possible to delete a parent row while it has child rows. For example, it is not possible to delete a row from COURSES table if the row has dependent rows in BATCHES table or in COURSE\_FACULTY table.

However, it is possible to delete parent row along with its child rows provided ON DELETE CASCADE option is given at the time of create foreign key constraint in child table. Please see chapter 4 for more information.

Changes made using INSERT, UPDATE and DELETE are not made permanent until explicitly or implicitly they are committed. See next section for more information.

## Transaction

A transaction is a collection of statements used to perform a single task. These statements are logically related as they perform a single task. All these statements must be executed to successfully complete the task. If any of the statements fails then the all the statements that were executed prior to the statement that failed should be undone otherwise data in the database becomes invalid and inconsistent.

The following example will illustrate the process.

Assume that faculty with code **kl** (Kevin Loney) is leaving the institute. So all his batches are to be assigned to **jc** (Jason Couchman). For this the following steps are to be taken.

- ❑ Change FCODE of all the batches that are currently being handled by **kl** to **jc**.
- ❑ Delete rows from COURSE\_FACULTY where FCODE is **kl**.

- 
- Delete row from FACULTY where FCODE is **kl**.

That means the following are the commands to be executed to perform the above-mentioned task.

```
update batches set fcode = 'jc' where fcode = 'kl';  
delete from course_faculty where fcode = 'kl';  
delete from faculty where fcode = 'kl';
```

It is important to make sure that all three statements are either successfully completed or all of them are rolled back. To ensure this Oracle provides transaction mechanism.

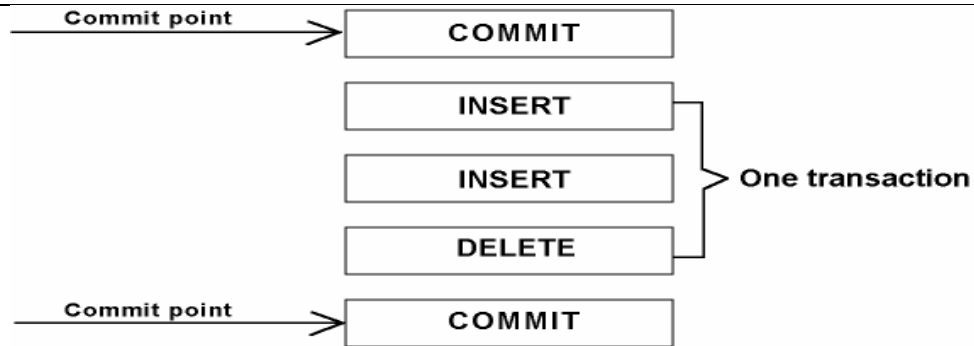
If UPDATE command in the above set of commands begins the transaction then only COMMIT command is given after the second DELETE is executed, the changes are committed. If ROLLBACK command is given then all the changes up to UPDATE will be rolled back.

So COMMIT and ROLLBACK command are used to ensure either everything is committed or everything is rolled back.

A transaction is a collection of statements which is to be either completely done or not done at all. In other words the process should not be *half-done*. That means ALL or NOTHING.

A transaction begins when previous transaction ends or when the session begins. A transaction ends when COMMIT or ROLLBACK is issued. See figure 1.

A new session starts when you connect to Oracle. For example, when you log on using SQL\*PLUS you start a new session. When you exit SQL\*PLUS the session is terminated.



**Figure 1:** A transaction is a collection of commands given between two commit points.

### COMMIT Command

Used to commit all changes made since the beginning of the transaction. It does the following.

- ❑ Makes all the changes made to database during transaction permanent
- ❑ Unlocks the rows that are locked during the transaction.
- ❑ Ends the transaction.
- ❑ Erases all savepoints in the current transaction(covered later).

Changes made to database such as inserting rows, deleting rows, and updating rows are not made permanent until they are committed.

---

## Implicit Commit

Oracle Implicitly issues commit in the following cases.

- ❑ Before a DDL command.
- ❑ After a DDL command.
- ❑ At normal disconnect from the database. For example, when you exit SQL\*PLUS using EXIT command.

---

**Note:** *It is very important to remember not to mix DML commands with DDL command as the later commands are going to issue COMMIT, which might commit incomplete changes also.*

---

## ROLLBACK Command

Used in transaction processing to undo changes made since the beginning of the transaction.

- ❑ Undoes the changes made to database in the current transaction
- ❑ Releases locks held on rows duration transaction.
- ❑ Ends the transaction.
- ❑ Erases all savepoints in the current transaction (covered later).

## ROLLBACK SEGMENT

In order to rollback changes that were made, Oracle has to store the data that was existing prior to the change so that previous data can be restored when user rolls back the changes. ROLLBACK SEGMENT is the area where Oracle stores data that will be used to roll back the changes. Every transaction is associated with a ROLLBACK SEGMENT.

Generally a few ROLLBACK SEGMENTS are created at the time of creating database. Database Administrator can create some more rollback segments depending upon the requirement using CREATE ROLLBACK SEGMENT command. Oracle assigns one rollback segment to each transaction. It is also possible to associate a transaction with a particular rollback segment using SET ROLLBACK SEGMENT command. It is not relevant to discuss more about the way ROLLBACK SEGMENT functions in a book like this. Please see Oracle Concepts for more information about ROLLBACK SEGEMENT.

## SAVEPOINT Command

Savepoint is used to mark a location in the transaction so that we can rollback up to that mark and not to the very beginning of the transaction.

The following is the syntax of SAVEPOINT.

```
SAVEPOINT savepoint_name;
```

A single transaction may also have multiple savepoints. The following example illustrates how to use savepoints.

```
SQL> update . . .
SQL> savepoint s1;
Savepoint created.
SQL> insert .... ;
SQL> insert .... ;
SQL> savepoint s2;
Savepoint created.
SQL> delete ...;
SQL> rollback to s1;
Rollback complete.
SQL> update...
SQL> commit;
```

In the above example, ROLLBACK TO S1; will undo changes made from savepoint S1 to the point of rollback. That means it will undo INSERT, INSERT, and even DELETE given after second savepoint. It doesn't undo UPDATE given before first savepoint. The COMMIT given after last UPDATE is going to commit first UPDATE and last UPDATE. Because all the remaining are already rolled back by ROLLBACK TO S1 command.

If we have given ROLLBACK TO S2; then it would have undone only DELETE given after the second savepoint and remaining statements (update, insert, insert, and update) would have been committed.

## Locking

It is important from database management system's point of view to ensure that two user are not modifying the same data at the time in a destructive manner.

Let us consider the following example to understand what will happen if two users are trying to update the same data at the same time.

Assume we have PRODUCTS table with details of products. Each product is having quantity on hand (QOH). Transactions such as sales and purchases are going to modify QOH column of the table.

The following are the steps that might take place when two transactions – one sale and one purchase – are taking place.

1. Assume QOH of product 10 is 20 units.
2. At 10:00 USER1 has sold 5 units and updated QOH as follows but has not committed the change. After UPDATE command QOH will be 15.  

```
update products set qoh = qoh - 5 where prodid = 10;
```
3. At 10:11 USER2 has purchased 10 units and updated QOH as follows and committed. After UPDATE command QOH will be 25 as 10 is added to 15.  

```
update products set qoh = qoh + 10 where prodid = 10;
```
4. If at 10:12 USER1 has rolled back the UPDATE then data that was there before the UPDATE should be restored i.e.20. But the actual data should be 30 as we added 10 units to it at 10:11 and committed.

As you can see in the above example, if two users are trying to update the same row at the same time the data may be corrupted. As shown in the above example at the end of the process QOH should be actually 30 but it is not only 20.

It is possible to ensure that two transactions are interfering with each other by locking the rows that are being modified so that only one transaction at a time can make the change.

Oracle ensures that only one transaction can modify a row at a time by locking the row once the row is updated. The lock will remain until the transaction is completed.

Oracle also ensures that other users will not see the changes that are not committed. That means if transaction T1 has updated a row then until the transaction is committed no other transaction in the system will see the changes made by T1. Instead other transactions will see only the data that was existing before the change was made.

The following scenario will illustrate the process in detail.

1. Assume QOH of product 10 is 20.
2. Transaction T1 has issued UPDATE command to update QOH of product 10. Oracle locks the row that is updated and does not allow other transactions to update the row. However, it is possible to read the data from the row.

```
update products set qoh = qoh + 5 where prodid = 10;
```

3. If T1 has issued SELECT to retrieve data then it will get 25 in QOH of product 10.
4. If T2 has issued SELECT command, it will see only 20 in QOH of product 10. This is because no uncommitted changes will be available to other transactions.
5. If T2 is trying to update product 10 then Oracle will cause transaction T2 to wait until transaction T1 (that holds lock on this row) is completed. Oracle will wait for lock to be released *indefinitely*.
6. If transaction T1 is committed then change is made permanent and lock will be released. Now it is possible for other transactions to see the updated data and even update the row if required.

**The following are the important points to be remembered about Oracle's locking mechanism.**

- ❑ Locking in Oracle is automatic and transparent. That means we never need to ask Oracle to lock row that is being modified. Locking is transparent means user is not aware of the locking process. It happens automatically and in the background.
- ❑ Oracle locks the row that is being updated. That means locking is row-level. Other levels that are in user are - page-level and table-level.
- ❑ Oracle releases locks held by a transaction when transaction is completed either successfully – using COMMIT – or unsuccessfully – using ROLLBACK.
- ❑ If a transaction is trying to update a row and if row is already locked then Oracle will wait for the row that is locked to be unlocked indefinitely. It is because of the fact that rows are locked for a small duration in a typical production system. So Oracle prefers to wait to cause any error.
- ❑ It is possible to lock table manually using LOCK TABLE command.

Locking the rows that are being updated is an important part of Oracle. It ensures that no two transactions can update the same row at the same time. Locking mechanism is followed by all database management systems. But some smaller database management systems follow page-level locking where not the exact row that is being modified is locked instead the entire page in which the row exists is locked.

## Read Consistency

When one user is trying to read data while other user is updating it, we might encounter a scenario where the data to be read by reader has already been modified by writer. In such cases Oracle provides read consistency to reader.

Read consistency ensures that whatever a user is reading is consistent to the point of starting the query. The following example will illustrate this mechanism.

- ❑ USER1 has issues a query to read data from SALES table at 10:00. The table contains 20000 rows and it may take 4 minutes to retrieve data completely. As user is only reading the data no lock is obtained on the table.
- ❑ USER2 has updated a row that is at 15000<sup>th</sup> row in the table at 10:01 and committed the change.
- ❑ USER1 has reached 15000<sup>th</sup> row at 10:03. Now USER1 get the data that was before the change as by the time query started the change didn't take place. That means Oracle has to provide the data to USER1 that existed at the time of starting the query – 10:00.
- ❑ Any changes whether committed or not made after 10:00 should not be retrieved by USER1.

Oracle uses ROLLBACK SEGMENT to store the data that was before the change to provide read consistency.

## Summary

ALTER TABLE command is used to alter the structure of the table. It is possible to add column or constraint, modify attributes such as datatype and length of existing columns, and drop column or constraint.

UPDATE and DELETE commands are used to update and delete rows of the table respectively. Changes made using DML commands are not committed until they are explicitly (using COMMIT command) or implicitly committed. ROLLBACK command is used to roll back the changes since the beginning of the transaction, which is a collection of statements to perform a single task. SAVEPOINT command is used to mark a point in the transaction so that changes made after the savepoint can be rolled back.

Oracle locks the rows that are being update to ensure only one transaction can update a row at a time. Oracle also ensures that the data retrieved by query is consistent to the point of starting the query – read consistency.

## **Exercises**

1. How do you add a check constraint to an existing column?
2. How do you drop a constraint?
3. Is it possible to know the name of the constraint? If yes, how?
4. How do you give primary key constraint if two or more columns are part of the primary key?
5. \_\_\_\_\_ command is used to mark a location in a transaction.
6. What is ROLLBACK SEGMENT?
7. When does a transaction begin?
8. If a row that to be updated is already locked then what happens?
9. What happens if a row is update and not committed or rolled back?
10. What is read consistency?