Partitioning Strategies

Oracle Partitioning offers three fundamental data distribution methods as basic partitioning strategies that control how data is placed into individual partitions:

* Range
* Hash
* List

Using these data distribution methods, a table can either be partitioned as a single list or as a composite partitioned table:

* [Single-Level Partitioning](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#CACBDEIE)
* [Composite Partitioning](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#i460895)

Each partitioning strategy has different advantages and design considerations. Thus, each strategy is more appropriate for a particular situation.

Single-Level Partitioning

A table is defined by specifying one of the following data distribution methodologies, using one or more columns as the partitioning key:

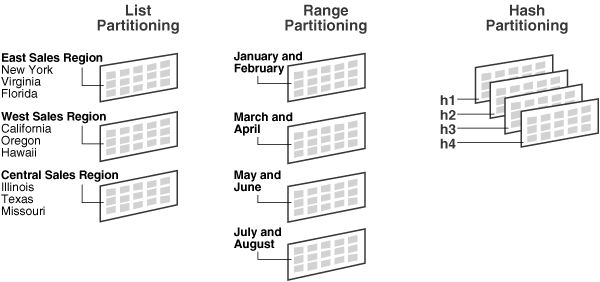
* [Range Partitioning](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#i468016)
* [Hash Partitioning](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#i462869)
* [List Partitioning](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#i460951)

For example, consider a table with a column of type NUMBER as the partitioning key and two partitions less\_than\_five\_hundred and less\_than\_one\_thousand. The less\_than\_one\_thousand partition contains rows where the following condition is true:

500 <= partitioning key < 1000

[Figure 2-2](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#i468083) offers a graphical view of the basic partitioning strategies for a single-level partitioned table.

***Figure 2-2 List, Range, and Hash Partitioning***

  
[Description of "Figure 2-2 List, Range, and Hash Partitioning"](https://docs.oracle.com/cd/B28359_01/server.111/b32024/img_text/cncpt158.htm)

Range Partitioning

Range partitioning maps data to partitions based on ranges of values of the partitioning key that you establish for each partition. It is the most common type of partitioning and is often used with dates. For a table with a date column as the partitioning key, the January-2005partition would contain rows with partitioning key values from 01-Jan-2005 to 31-Jan-2005.

Each partition has a VALUES LESS THAN clause, which specifies a non-inclusive upper bound for the partitions. Any values of the partitioning key equal to or higher than this literal are added to the next higher partition. All partitions, except the first, have an implicit lower bound specified by the VALUES LESS THAN clause of the previous partition.

A MAXVALUE literal can be defined for the highest partition. MAXVALUE represents a virtual infinite value that sorts higher than any other possible value for the partitioning key, including the NULL value.

Hash Partitioning

Hash partitioning maps data to partitions based on a hashing algorithm that Oracle applies to the partitioning key that you identify. The hashing algorithm evenly distributes rows among partitions, giving partitions approximately the same size.

Hash partitioning is the ideal method for distributing data evenly across devices. Hash partitioning is also an easy-to-use alternative to range partitioning, especially when the data to be partitioned is not historical or has no obvious partitioning key.

**Note:**

You cannot change the hashing algorithms used by partitioning.

List Partitioning

List partitioning enables you to explicitly control how rows map to partitions by specifying a list of discrete values for the partitioning key in the description for each partition. The advantage of list partitioning is that you can group and organize unordered and unrelated sets of data in a natural way. For a table with a region column as the partitioning key, the North America partition might contain values Canada, USA, and Mexico.

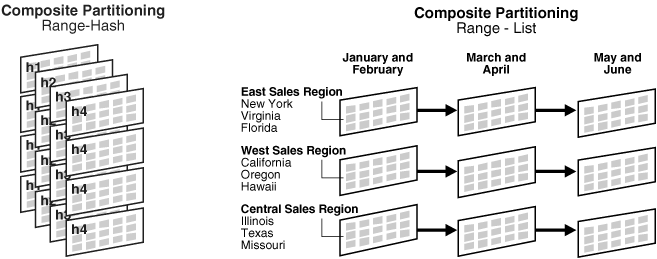
The DEFAULT partition enables you to avoid specifying all possible values for a list-partitioned table by using a default partition, so that all rows that do not map to any other partition do not generate an error.

Composite Partitioning

Composite partitioning is a combination of the basic data distribution methods; a table is partitioned by one data distribution method and then each partition is further subdivided into subpartitions using a second data distribution method. All subpartitions for a given partition together represent a logical subset of the data.

Composite partitioning supports historical operations, such as adding new range partitions, but also provides higher degrees of potential partition pruning and finer granularity of data placement through subpartitioning. [Figure 2-3](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#CACFJBDF) offers a graphical view of range-hash and range-list composite partitioning, as an example.

***Figure 2-3 Composite Partitioning***

  
[Description of "Figure 2-3 Composite Partitioning"](https://docs.oracle.com/cd/B28359_01/server.111/b32024/img_text/cncpt168.htm)

* [Composite Range-Range Partitioning](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#CACFIDEG)
* [Composite Range-Hash Partitioning](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#CACBHCEA)
* [Composite Range-List Partitioning](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#CACHFHGG)
* [Composite List-Range Partitioning](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#CACBEGAH)
* [Composite List-Hash Partitioning](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#CACGBJBB)
* [Composite List-List Partitioning](https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm#CACHGFGD)

# [**Different types of keys in RDBMS?**](http://stackoverflow.com/questions/1711492/what-are-the-different-types-of-keys-in-rdbms)

* Alternate key - An alternate key is any candidate key which is not selected to be the primary key
* Candidate key - A candidate key is a field or combination of fields that can act as a primary key field for that table to uniquely identify each record in that table.
* Compound key - compound key (also called a composite key or concatenated key) is a key that consists of 2 or more attributes.
* Primary key - a primary key is a value that can be used to identify a unique row in a table. Attributes are associated with it. Examples of primary keys are Social Security numbers (associated to a specific person) or ISBNs (associated to a specific book). In the relational model of data, a primary key is a candidate key chosen as the main method of uniquely identifying a tuple in a relation.
* Superkey - A superkey is defined in the relational model as a set of attributes of a relation variable (relvar) for which it holds that in all relations assigned to that variable there are no two distinct tuples (rows) that have the same values for the attributes in this set. Equivalently a superkey can also be defined as a set of attributes of a relvar upon which all attributes of the relvar are functionally dependent.
* Foreign key - a foreign key (FK) is a field or group of fields in a database record that points to a key field or group of fields forming a key of another database record in some (usually different) table. Usually a foreign key in one table refers to the primary key (PK) of another table. This way references can be made to link information together and it is an essential part of database normalization

# **PL/SQL - Cursors**

Oracle creates a memory area, known as context area, for processing an SQL statement, which contains all information needed for processing the statement, for example, number of rows processed, etc.

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors:

* Implicit cursors
* Explicit cursors

## Implicit Cursors

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the **SQL cursor**, which always has the attributes like %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT. The SQL cursor has additional attributes, %BULK\_ROWCOUNT and %BULK\_EXCEPTIONS, designed for use with the FORALL statement. The following table provides the description of the most used attributes:

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| %FOUND | Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE. |
| %NOTFOUND | The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE. |
| %ISOPEN | Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement. |
| %ROWCOUNT | Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement. |

## Explicit Cursors

Explicit cursors are programmer defined cursors for gaining more control over the **context area**. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is :

CURSOR cursor\_name IS select\_statement;

Working with an explicit cursor involves four steps:

* Declaring the cursor for initializing in the memory
* Opening the cursor for allocating memory
* Fetching the cursor for retrieving data
* Closing the cursor to release allocated memory

## Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example:

CURSOR c\_customers IS

SELECT id, name, address FROM customers;

## Opening the Cursor

Opening the cursor allocates memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open above-defined cursor as follows:

OPEN c\_customers;

## Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example we will fetch rows from the above-opened cursor as follows:

FETCH c\_customers INTO c\_id, c\_name, c\_addr;

## Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close above-opened cursor as follows:

## Types of Subqueries

**Single Row Sub Query:**Sub query which returns single row output. They mark the usage of single row comparison operators, when used in WHERE conditions.

**Multiple row sub query:**Sub query returning multiple row output. They make use of multiple row comparison operators like IN, ANY, ALL. There can be sub queries returning multiple columns also.

**Correlated Sub Query:** Correlated subqueries depend on data provided by the outer query.This type of subquery also includes subqueries that use the EXISTS operator to test the existence of data rows satisfying specified criteria.

# **About Clusters**

A **cluster** provides an optional method of storing table data. A cluster is made up of a group of tables that share the same data blocks. The tables are grouped together because they share common columns and are often used together. For example, the emp and dept table share the deptno column. When you cluster the emp and dept tables (see [Figure 20-1](https://docs.oracle.com/cd/B28359_01/server.111/b28310/clustrs001.htm#i1006158)), Oracle Database physically stores all rows for each department from both the emp and dept tables in the same data blocks.

Because clusters store related rows of different tables together in the same data blocks, properly used clusters offer two primary benefits:

* Disk I/O is reduced and access time improves for joins of clustered tables.
* The **cluster key** is the column, or group of columns, that the clustered tables have in common. You specify the columns of the cluster key when creating the cluster. You subsequently specify the same columns when creating every table added to the cluster. Each cluster key value is stored only once each in the cluster and the cluster index, no matter how many rows of different tables contain the value.

# **Difference between function, procedure and trigger**

Explain the difference between trigger and stored procedure.

- A stored procedure can accept parameters while a trigger cannot.  
- A trigger can’t return any value while stored procedures can.  
- A trigger is executed automatically on some event while a stored procedure needs to be explicitly called.  
- Triggers are used for insertions, update and deletions on tables while stored procedures are often using independently in the database.  
- A trigger cannot be written in a stored procedure. However, the reverse is not possible.

**Row Level Trigger**  
Row Level Trigger is fired each time row is affected by Insert, Update or Delete command. If statement doesn’t affect any row, no trigger action happens.  
  
**Statement Level Trigger**  
This kind of trigger fires when a SQL statement affects the rows of the table. The trigger activates and performs its activity irrespective of number of rows affected due to SQL statement.

Explain the categories of oracle processes i.e. user, data writing processes, logging processes and monitoring processes.

**User process -** User process is used in invocation of application software.  
  
**Data writing process -** A database writer process is used to write buffer content into a datafile. They are specifically used to write “dirty block” to data files from the buffer.  
  
**Logging processes -** Log writer is used to write the redo log buffer from system global area to online redo log file. Only those redo entries are written hat have been copied into the buffer since the last time it wrote.  
  
**Monitoring process -** this can be either a system monitor process or a process monitor process. System monitor process is mainly used for crash recovery and cleaning up of temporary segments. Process monitor is used to clean all resources acquired by a failed process.

What is an integrity constraint?

Integrity constraints define a business rule for a column of the table. They are defined with a table and are stored as part of a table’s definition.  
  
Types of integrity constraints supported by oracle are:  
  
NOT NULL integrity constraint   
Unique Key integrity constraint  
Primary Key integrity constraint  
Foreign key integrity constraint  
Check integrity constraint

What is nested table? Explain the purpose of nested table.

A nested table is an unordered set of data elements. These data elements are all of the same datatype. It has a single column whose type is either built in or an object type. It is a table stored within the structure of another table.  
  
**Example:**

NESTED TABLE employee STORE AS employee\_table

**Purpose:**  
Can load the entire nested table into the database as column values. This means that we can store and retrieve nonatomic data in a single column.

What are constraints?

It is the rules that prevent the invalid entry into the table. They are stored in the data dictionary. They can be defined either at column level or table level.  
  
Following are the constraints available in oracle.  
  
**Not Null -** Specifies that column can’t contain a null value.  
  
**Unique -** Enforce unique value for all rows in the table.  
  
**Primary key -** Uniquely identifies each row of the table.  
  
**Foreign key -** Enforces a foreign key relationship between the columns of the referenced table.  
  
**Check -** specifies condition that must be true.

Define referential integrity.

Referential integrity is the rules that governs the relationships between primary keys and foreign keys of the tables and ensure data consistency. It ensures the value of foreign key be matched by the value of a primary key in another table.

What is TCL command?

TCL - Transaction Control: statements used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.   
  
**COMMIT -** save work done  
  
**SAVEPOINT -** identify a point in a transaction to which you can later roll back  
  
**ROLLBACK -** restore database to original since the last COMMIT  
  
**SET TRANSACTION -** Change transaction options like isolation level and what rollback segment to use

**Difference between locks, latches, enqueues and semaphores.**

Latches are used to protect Oracle data structures to be modified or run by more than one process. They are more restrictive than locks. Locking has a similar concept. When a transaction being modified is locked, it cannot be modified until committed. However, locks allow some user interaction. Enqueues are associated with a transaction or a session that serialize access to database resources. When an enqueue is associated with a session, it cannot be held by another session. Semaphores are mainly used for communication between processes. They control the processes by allowing them stop, wait and resume their processing.