Tablespace Sets

Oracle Sharding creates and manages tablespaces as a unit called a **tablespace set**. The `PARTITIONS AUTO` clause specifies that the number of partitions should be automatically determined. This type of hashing provides more flexibility and efficiency in migrating data between shards, which is important for elastic scalability.

A tablespace is a logical unit of data distribution in an SDB. The distribution of partitions across shards is achieved by automatically creating partitions in tablespaces that reside on different shards. To minimize the number of multi-shard joins, the corresponding partitions of related tables are always stored in the same shard. Each partition of a sharded table is stored in a separate tablespace.

Note:

Only Oracle Managed Files are supported by tablespace sets.

Individual tablespaces cannot be dropped or altered independently of the entire tablespace set.

`TABLESPACE SET` cannot be used with the user-defined sharding method.

Related Topics

- Sharding Architecture
  Oracle Sharding is a database scaling technique based on horizontal partitioning of data across multiple Oracle databases. Applications perceive the pool of databases as a single logical database.
- *Using Oracle Sharding*

Overview of Views

54

A **view** is a logical representation of one or more tables. In essence, a view is a stored query.

A view derives its data from the tables on which it is based, called *base tables*. Base tables can be tables or other views. All operations performed on a view actually affect the base tables. You can use views in most places where tables are used.

Note:

Materialized views use a different data structure from standard views.

Views enable you to tailor the presentation of data to different types of users. Views are often used to:

- Provide an additional level of table security by restricting access to a predetermined set of rows or columns of a table based Java code. At run time, the program can communicate with multi-vendor databases using standard JDBC drivers.

The following example shows a simple SQLJ executable statement:

```
String name;
#sql  { SELECT first_name INTO :name FROM employees WHERE 
employee_id=112 };
System.out.println("Name is " + name + ", employee number = " + 
employee_id);
```

See Also:

- "SQLJ"
- *Oracle Database SQLJ Developer's Guide*

Overview of Triggers

54 A database **trigger** is a compiled stored program unit, written in either PL/SQL or Java, that Oracle Database invokes ("fires") automatically in certain situations.

A trigger fires whenever one of the following operations occurs:

1. DML statements on a particular table or view, issued by any user

   DML statements modify data in schema objects. For example, inserting and deleting rows are DML operations.
2. DDL statements issued either by a particular user or any user

   DDL statements define schema objects. For example, creating a table and adding a column are DDL operations.
3. Database events

   User login or logoff, errors, and database startup or shutdown are events that can invoke triggers.

Triggers are schema objects that are similar to subprograms but differ in the way they are invoked. A subprogram is explicitly run by a user, application, or trigger. Triggers are implicitly invoked by the database when a triggering event occurs.

See Also:

- "Overview of SQL Statements" to learn about DML and DDL
- "Overview of Database Instance Startup and Shutdown"

full index scan

An index scan in which the database reads only the root and left side branch blocks to find the first leaf block, and then reads the leaf blocks in index sorted order using single block I/O.

full outer join

A join between two tables that returns the result of an inner join and the result of a left outer join and a right outer join.

full table scan

A scan of table data in which the database sequentially reads all rows from a table and filters out those that do not meet the selection criteria. The database scans all formatted data blocks under the high water mark (HWM).

54 **function**

A schema object, similar to a PL/SQL procedure, that always returns a single value.

function-based index

An index that computes the value of a function or expression involving one or more columns and stores it in the index.

GDS

See Global Data Services (GDS)

GDS catalog

A metadata repository, located inside an Oracle database, that is associated with a GDS configuration. Every cloud has one and only one catalog.

GDS configuration

A set of databases integrated by the GDS framework into a single virtual server that offers one or more global services while ensuring high performance, availability, and optimal utilization of resources.

See also global service.

GDS pool

A set of databases within a GDS configuration that provides a unique set of global services and belongs to a specific administrative domain.

See also dimension table; fact table.

state object

A session-level structure that contains metadata about the status of database resources such as processes, sessions, and transactions in the SGA.

statement trigger

A trigger that is fired once on behalf of the triggering statement, regardless of the number of rows affected by the triggering statement.

statement-level atomicity

The characteristic of a SQL statement as an atomic unit of work that either completely succeeds or completely fails.

statement-level read consistency

The guarantee that data returned by a single query is committed and consistent for a single point in time.

statement-level rollback

A database operation in which the effects of an unsuccessful SQL statement are rolled back because the statement caused an error during execution.

stored procedure

54 A named PL/SQL block or Java program that Oracle Database stores in the database. Applications can call stored procedures by name.

Streams pool

A memory pool that stores buffered queue messages.

Structured Query Language (SQL)

See SQL.

subquery

A query nested within another SQL statement. Unlike implicit queries, subqueries use a `SELECT` statement to retrieve data.