**TUNING THE INSTANCE**

**What is Instance Tuning**

When considering instance tuning, take care in the initial design of the database to avoid bottlenecks that could lead to performance problems. In addition, you must consider:

1. Allocating memory to database structures
2. Determining I/O requirements of different parts of the database
3. Tuning the operating system for optimal performance of the database

After the database instance has been installed and configured, you must monitor the database as it is running to check for performance-related problems.

**Performance Principles**

Performance tuning requires a different, although related, method to the initial configuration of a system. Configuring a system involves allocating resources in an ordered manner so that the initial system configuration is functional. Tuning is driven by identifying the most significant bottleneck and making the appropriate changes to reduce or eliminate the effect of that bottleneck. Usually, tuning is performed reactively, either while the system is in preproduction or after it is live.

**Baselines**

The most effective way to tune is to have an established performance baseline that you can use for comparison if a performance issue arises. Most database administrators (DBAs) know their system well and can easily identify peak usage periods.

**Example:**

The peak periods could be between 10.00am and 12.00pm and also between 1.30pm and 3.00pm. This could include a batch window of 12.00am midnight to 6am. It is important to identify these peak periods at the site and install a monitoring tool that gathers performance data for those high-load times. Optimally, data gathering should be configured from when the application is in its initial trial phase during the Quality Assurance cycle. Otherwise, this should be configured when the system is first in production. Ideally, baseline data gathered should include the following:

1. Application statistics (transaction volumes, response time)
2. Database statistics
3. Operating system statistics
4. Disk I/O statistics
5. Network statistics

In the Automatic Workload Repository, baselines are identified by a range of snapshots that are preserved for future comparisons.

There are two automatic tuning features that are available:

* **Automatic plan correction** identifies problematic query execution plans and fixes query execution plan performance problems. **Applies to**: SQL Server (Starting with SQL Server 2017 (14.x)) and Azure SQL Database
* **Automatic index management** identifies indexes that should be added in your database, and indexes that should be removed. **Applies to**: Azure SQL Database

### How does automatic tuning work?

Automatic tuning is a continuous monitoring and analysis process that constantly learns about the characteristics of your workload and identify potential issues and improvements.

ALTER DATABASE current

SET AUTOMATIC\_TUNING ( FORCE\_LAST\_GOOD\_PLAN = ON );

DYNAMIC PERFORMACE VIEW

**Dynamic Performance Views**.

The **Oracle** Server contains a set of underlying**views** that are maintained by the server and accessible to the database administrator user SYS. ... These **views** provide data on internal disk structures and memory structures. These **views** can be selected from, but never updated or altered by the user.

## V$ Views

The actual dynamic performance views are identified by the prefix V\_$. Public synonyms for these views have the prefix V$. Database administrators and other users should access only the V$ objects, not the V\_$ objects.

The dynamic performance views are used by Oracle Enterprise Manager, which is the primary interface for accessing information about system performance. After an instance is started, the V$ views that read from memory are accessible. Views that read data from disk require that the database be mounted, and some require that the database be open.

## GV$ Views

For almost every V$ view described in this chapter, Oracle has a corresponding GV$ (global V$) view. In Real Application Clusters, querying a GV$ view retrieves the V$ view information from all qualified instances. In addition to the V$ information, each GV$ view contains an extra column named INST\_IDof datatype NUMBER. The INST\_ID column displays the instance number from which the associated V$ view information was obtained. The INST\_IDcolumn can be used as a filter to retrieve V$ information from a subset of available instances. For example, the following query retrieves the information from the V$LOCK view on instances 2 and 5:

SQL> SELECT \* FROM GV$LOCK WHERE INST\_ID = 2 OR INST\_ID = 5;