1. Introduction

Oracle 12 c provides many features to promote reliability, security, performance and transparency. Many companies use the Oracle database to store and manage mission critical data. Having a means to guarantee the effective and expedient processing of data into information in today’s age provides a point of leverage that makes or breaks businesses. Analyzed in the coming sections are detailed points summarized for topics such as user provisioning and security, indexing, object transparency and performance for the modern database.

1. Analysis and Discussion of Topics

* Basic user security (password security, and assigning system and object privileges, and roles).

As an Oracle database is created there are in default several database users created for the purpose of accessing and modifying data. Due to the fact that applications and users need access to the database and need to be provisioned accordingly setting up users their passwords and privileges are seen as critical steps in database administration. SQL statements allow for the creation of users by using the *create user* statement. Accordingly when a user is provisioned the user is provisioned with a password. The statement *alter user identified by password* allows for the setting of a password associated with a user. This allows for a layer of security in the database and allows for restricted access to tables and data accordingly. When a user is identified with a password he or she is granted a level of privilege that defines what views and tables they can access and modify. A schema is a collection of database objects such as tables and indexes. A schema is owned by a user and has the same name as a user. The SYS schema is the super user of the database and owns all internal data dictionary objects and is used for tasks such as creating a database, starting or stopping the instance, backup and recovery and adding or moving data files. (Kuhn 128) If a person is working in a pluggable database environment then all users can be looked up by querying cdb\_users. To keep the database secure over time it is advised that password for every default account must be changed periodically. User accounts can be locked so that users won’t be able to access data unless they are unlocked. Similarly passwords can expire which force the user to change the password after they logon with the expired password. All these tasks are important for making sure that data is operated upon a secure and reliable manner.

* Describe the purpose of an index.

An Oracle database uses several types of indexes to secure fast access to commonly viewed data. Judicious use of indexes can make processing time much faster for a database. It is also important not to overuse indexes as after a certain threshold is violated indexes follow the law of diminishing returns. A well indexed database is especially important in data warehousing scenarios where databases are very large and pinpoints of reference need to be established for quick and efficient access. The first type of default index used is a B-tree indexes. This index stores information in a hierarchical tree structure where each node contains a part of the information. The second type of index used is a bitmap indexes which is used when the number of distinct values is very less. The bitmap index is also mostly used on tables that updated non-frequently. The third type of index is known as a reverse key indexes which are used in RAC to prevent hotspots. The fourth type of index is the partitioned indexes which can be local or global. The local partitioned indexes use the same partitioning scheme that the table is using and global partitioned indexes can have a different indexing scheme than the parent tables. Finally there are the fifth type of indexes which are cluster indexes which are used on clustered tables.

* Using primary key and unique key constraints to manage indexes on primary key and unique key columns.

Primary keys are used to identify entities in a database by utilizing a combination of specific columns. A table can have multiple unique and foreign keys but it can only have one primary key. Unique keys can have null values but primary keys have to be non-null. A foreign key can reference a non-primary unique key in an entity. Primary keys are by default automatically indexed. However in setting indexes for unique keys care should be taken that these values can become null and potentially non-advantageous to index. For unique key columns can be assigned as a special case of index, acting like a regular index with added checking for uniqueness. Using *show indexes from* statement you can see your unique keys are in fact B-tree type indexes.

* Describe why or why not foreign key columns should be indexed.

For primary keys indexes are automatically generated when they are created. But for foreign keys indexes are not automatically created. The foreign key may be part of a composite primary key for the table for this a clustered index would exist with the foreign key as part of the clustering key. We can also have queries that may require an index that includes the foreign key and one or more additional columns in the table, which in effect would mean that a non-clustered index would be used. In general indexes on foreign keys can provide performance benefits for table joins involving the foreign key and the primary keys.

* Use of views and synonyms to provide application transparency.

In an oracle database there exist several views which are virtual tables. These can be created on a table or another views. The views amount to windows via which we can access or change base table data. Views take data from the base table and are stored as a query in data dictionary. The main advantages of views is that it can restrict access to a defined set of rows of a table. We can hide the complexity of querying and calculation by using views. These snapshots of data are a secure and transparent way to let specific users’ access specific subsets of data shared across multiple entities. Similarly synonyms provide a mechanism for creating an alternate name or alias for an object. Creating a synonym that points to another object eliminates the need to specify a schema owner and also allows you to specify a name for the synonym that does not match the object name. This allows for a layer of abstraction between object and user, often referred to as object transparency. (Kuhn 247) Synonyms let you to manage objects transparently and separately from the users that access the objects. You can also relocate objects to different schemas. The good thing is that the application code that references the synonym doesn’t change, only the definition of the synonym does.

* Uses of materialized views.

A materialized view allows you to execute a query at a point in time and store the result in a table. After the materialized view is initially populated you can later rerun the materialized view query and store fresh results in the underlying table. (Kuhn 403) The two main uses of materialized views are first to replicate data to offload query workloads to separate reporting databases and second to improve performance of queries by periodically computing and storing results of complex aggregations of data, which lets users query point in time results of complex aggregations. These are useful in computing historical data to see how a process calculated an algorithm over time and how the variables of that algorithm changed with each point in time. Secondly it helps produce a backtracking for complex changes to the database that occur over a set of specific transactions by providing snapshots of changes to the entity columns. Materialized vies allow for speeding up for write operations. Since indexes can be created on MV’s they are quite fast. To reduce index maintenance overhead materialized views can be created with indexes specified on them. MV’s also help in speeding read operations for complex joins; pivots can be speed up by creating indexes on the materialized views. It is particularly useful in reporting scenarios.

* Describe two database features new to Oracle Database 12c that you’ve encountered so far in the class.

The Oracle 12 c database has a new feature of CDB views for pluggable databases. This view lets you see at the root container out of which the pluggable databases are tenanted out. We can see the *cdb root* when we do a query on the *cdb view*. A second new feature in 12 c is the in memory column store. This feature allows for objects that are tables or partitions to be stored in memory in a columnar format. A database administrator has to simply allocate memory to In-Memory Column Store. As we discussed in class for the database statistics discussion the optimizer is aware of in-memory Column Store so queries that are related to columnar data are routed accordingly. This improves spontaneous querying without causing a performance bottleneck in the database.(Oracle Docs)

III. Conclusion

In conclusion we see that not only does Oracle 12 c provide secure means of setting up users and secure access to data within a database, it also promotes the sound architectural practices of securing views and synonyms for greater object transparency. Judicious use of indexes and materialized views further add to the performance oriented offering of the database which is not the only new feature in the 12 c offering that also boasts a multi-tenant architecture which can be viewed as a set of pluggable databases spreading from the root container and having advanced computation features like in memory column stores which work with the optimizer to generate useful database statistics to make the database constantly improved.

References:

Kuhn D. (2013) Pro Oracle Database 12c Administration: Second Edition: New York, NY.Apress

Oracle Docs. (2012). [Oracle Database Online Documentation 12c Release 1 (12.1)](https://docs.oracle.com/database/121/index.html) Retrieved from: https://docs.oracle.com/database/121/NEWFT/chapter12102.htm#NEWFT003