Abdullah Bin Zubair| abdullahbinzubair570@gmail.com/ AZ5645A@GREENWICH.AC.UK

Oracle Database Administration

Project Report

Contents

[The implementation and the final report to support the following case study and requirements 2](#_Toc69701958)

[DBA requirements: 2](#_Toc69701959)

[Problem solving requirements: 9](#_Toc69701960)

[Evaluation: 13](#_Toc69701961)

[References: 14](#_Toc69701962)

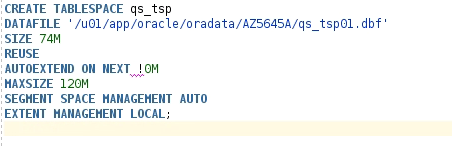
# The implementation and the final report to support the following case study and requirements

## DBA requirements:

**Requirement 1:** Create new permanent tablespace qs\_tsp, size 70Mb to store extra inventory information. Space management within this tablespace should be locally managed and automatically extendable by 10Mb to a maximum size 120Mb. You need to decide on other properties of this tablespace and explain why you made those choices, their benefits and drawbacks.

**Implementation:**

The SQL query to create the required tablespace is given below:



NOTE: Correction in SQL above: “AUTOEXTEND ON NEXT 10M”. I ran the above code after correcting the SQL command. It ran without any error.

* All required requirements have been implemented, other parameters used are listed below.
* REUSE specifies that if the created data file already exists then it should be reused and replaced. If it does not exists, database ignores this keyword.
* SEGEMENT SPACE MANAGEMENT AUTO mitigates the requirement to manage freelists and freelists groups by the use of bitmaps which describes the usage of space by each block within a segment.

Some of the important default values which were used:

* LOGGING specifies the logging attributes of all the objects for example tables, indexes, views etc. in the log files. This clause is not valid for temporary and undo tablespaces as they do not require separate logging mechanism.
* ONLINE specifies that the tablespace created should be available/live so that users can query it right away.
* PERMANENT states that the tablespace created is permanent, not temporary or undo.
* NO COMPRESS specifies that the objects do not need any compression. This COMPRESS feature is suitable for data warehouse systems.

**Requirement 2:** Create several new users with the following login requirements:

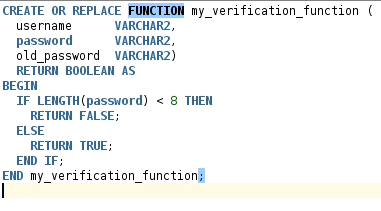
* + 1. Two users for the Customer Service department and one user for the Inventory department. Default tablespace for all users should be qs\_tsp and temporary tablespace should be set to temp.
    2. All of the above users should have unique usernames and passwords. Password should be at least 8 characters long and users should not be allowed to have a password that is shorter than 8 characters. There shouldn’t be any other enforcement on the password.

* + 1. The users should be forced to change their passwords every 6 months and not allowed to reuse the same password.
    2. If during the login they specified their password incorrectly 3 times, the account should be locked for 30 hours.
    3. All new users should be allowed to leave their session without interacting with it for no more than 15 min.

Make sure you include an explanation on how these requirements are implemented.

**Implementation:**

First the verification function for password is compiled which checks the length of password. Password length has to be equal or greater than 8.

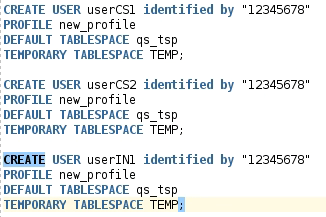


Then profile “new\_profile” is made according to the requirements:



* PASSWORD\_VERIFY\_FUNCTION takes in the function which we created in the first step. It checks the length of password.
* PASSWORD\_REUSE\_TIME and PASSWORD\_REUSE\_MAX works in conjunction. If any one of them is set to unlimited and the other is set to an integer then user is not able to reuse the password.
* PASSWORD\_LIFE\_TIME and PASSWORD\_GRACE\_TIME works in conjunction. In this example, after 173 days, user will be issued with a warning every time they log in to the database saying that they should change their password. After 7 days of that warning and password not being changed, password will expire.
* FAILED\_LOGIN\_ATTEMPTS specified as 3 states that after 3 failed login attempt, account will be blocked for 30 days as we have specified 30 for PASSWORD\_LOCK\_TIME.
* IDLE\_TIME specifies the amount of minutes spent without activity after which the system will logout the user. In this case its 15.

Now we create the users with qs\_tsp and TEMP as default and temporary tablespace respectively and assign new\_profile to them:



USERIN1 would be given privileges (in next requirement) to create tables and views and also insert into specific tables. For this cause we have to assign quota to this user:



**Requirement** **3:** New users should have following access rights:

* + 1. All of the above users should be able to login to the database.
    2. Two users for the Customer Service department should have **Read/Only** access to PRODUCT\_INFORMATION, PRODUCT\_DESCRIPTIONS, ORDERS, ORDER\_ITEMS and CUSTOMERS tables, owned by user **oe**.
    3. Both Customer Service users should be able to query easily all orders for a particular customer.
    4. The user for the Inventory department should have **Read/Write access** to the tables: INVENTORIES and WAREHOUSES as well as **Read/Only access** to ORDERS, ORDER\_ITEMS and CUSTOMERS tables, owned by user **oe**.

The user for the Inventory department should also be allowed to create new tables and new views.

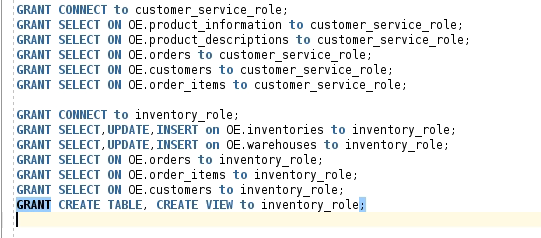
Make sure you include an explanation on how these requirements are implemented.

**Implementation:**

First we create 2 roles for each department:



Then we grant necessary privileges to both roles:





* CONNECT role is granted so that both roles can create session and connect to the database.
* SELECT is granted for the specified tables to fulfil READ only access condition.
* UPDATE, INSERT and DELETE is granted for the specified tables in order to satisfy the condition of READ/WRITE access.
* CREATE TABLE and CREATE VIEW is granted so that particular role can make new tables and views.

Lastly we assign roles to newly created users:



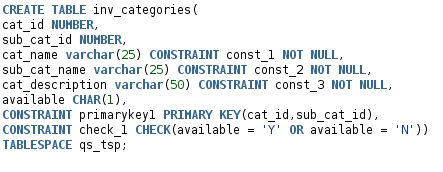
**Requirement** **4:** Create a table inv\_categories in the schema of the newly created Inventory department with the structure described in the table below. Make sure all constraints have user-defined names and the table is stored in qs\_tsp tablespace.

|  |  |  |
| --- | --- | --- |
| **Column name** | **Data type** | **Constraint** |
| cat\_id | NUMBER | Composite Primary Key |
| sub\_cat\_id | NUMBER |
| cat\_name | VARCHAR2(25) | Not Null |
| sub\_cat\_name | VARCHAR2(25) | Not Null |
| cat\_description | VARCHAR2(50) | Not Null |
| available | CHAR(1) | Check: this column should contain only values ‘Y’ or ‘N’ |

Make sure you include an explanation on how this requirement is implemented.

**Implementation:**

SQL query to create the required table is:



* NOT NULL constraints have been given names and defined inline.
* Combined PRIMARY KEY has been defined and named.
* CHECK constraint for AVAILABLE column has been defined and named.
* TABLESPACE in which this TABLE is being made is also mentioned at the end of SQL statement.

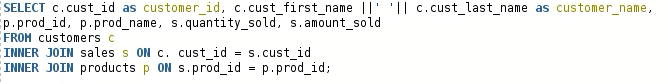
**Requirement** **5:** The **sh** user must be able to run a report with the information about all products sold for a particular customer, based on the provided customer id.

You need to implement the best way for this user to produce such reports. The output should include customer id and customer name as well as information about the products that were sold to this customer- product id, product name, quantity sold, and amount sold.

You need to include a rationale for your implementation discussing alternatives, their benefits and drawbacks.

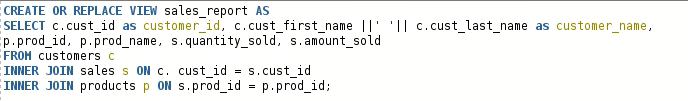
**Implementation:**

The SH user needs to query sales report for particular customers by referencing their customer ID. First obvious method is to write a very lengthy query each time to get sales report for the customers. This can be a time consuming and non-efficient way of doing this particular task. So whenever the SH user needs the sales report, he/she would be required to write the following query:



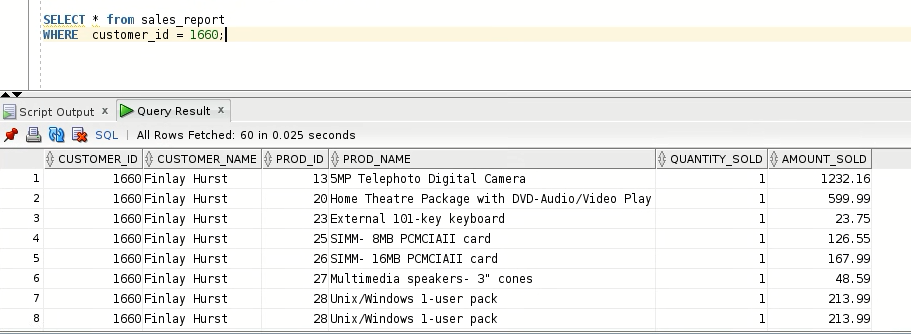
* The above query lists down all the customers’ sales data as we did not query data for a particular customer by referencing their customer’s ID.
* All the required columns as specified by the requirement are obtained in the query.
* It joins 3 tables - customers, sales and products.
* Customer name is concatenated as it was defined separately in two columns as first and last name in customers table.
* CUST\_ID is also aliased as CUSTOMER\_ID for clear understanding and easy referencing.

This process can be simplified and it can be made very easy to carry out by creating a VIEW. Making a view is as lengthy as writing a SQL statement, but the benefit of VIEW is that you do not have to write the same query over and over again. Below you will see how to create a view for the sales report of customers:



* This view also display results for all the customers.
* Just 1 line addition (first line) to the previous query let us make the VIEW.

Now whenever SH user would want to get sales report for any particular customer, he/she would not need to write the whole query again, rather he/she would just need to query the SALES\_REPORT view by referencing the CUSTOMER\_ID in WHERE clause. An example is given below:



## Problem solving requirements:

**Requirement 6:** If you accidently dropped an important schema (for example**,** hr)in your database, discuss the best functionality to use in order to recover it quickly and safely. Provide an explanation of the steps needed to recover data including the SQL commands for each step.

**Implementation:**

If I accidently drop an important schema in my database, then the first thing I would think of to get that schema back would be my option of flashing back the database to the point when I had not dropped the schema. Then immediately I would look at the current time and subtract 3 to 5 minutes from it in order to get the time when schema was still in the database. For the sake of demonstration, let’s assume that it was 07:43 am when I realised that I have made a big mistake. Now I will subtract 5 minutes which will bring time back to 7:38 am and that’s the information (including current date) I would write it down somewhere safe. Now I will do all the necessary things in order to start my flashback process.

Now there are 2 pre requisites for this task and those are:

* Database must be in archive log mode

ALTER DATABASE ARCHIVELOG;

* Flashback functionality should be enabled

ALTER DATABASE FLASHBACK ON;

Then I would shut down my database and start it in mount mode:

* SHUTDOWN IMMEDIATE;
* STARTUP MOUNT;

After performing the above steps, my database is ready to carry the flashback process. For that cause I will execute the following SQL command which will flashback the database to the point in time which we noted down earlier:

* FLASHBACK database to timestamp TO\_TIMESTAMP( ‘2020-11-18 07:38:00′,’YYYY-MM-DD HH24:MI:SS’);

Now I could also have used SCN number to get back but in real life I would never query for current SCN unless I know that I’m going to perform data movement. In that case the flashback query would go like:

* FLASHACK TO SCN 12345;

After flashing back the database, I would open the database in read only mode by executing the following command:

* ALTER DATABASE OPEN READ ONLY;

Then I would reset the logs:

* ALTER DATABASE OPEN RESETLOGS;

After completing these steps, I would have successfully managed to retain the schema that I accidently dropped. I could also flashback the database an hour before or maybe a day before but that really depends on the scenario. I could also have queried the last activity of user on their particular schema and would have noticed the timestamp of that activity and then flashback database to that timestamp. It largely depends on the scenario and only after analysing all the variables, a DBA could decide exactly what approach to carry out.

**Requirement 7:** Your users are reporting “ORA-1555: Snapshot too old” errors. What might be the cause of this? Explain what you need to do in order to solve this issue.

**Implementation:**

This error is related to the read consistency phenomena and undo retention management. Read consistency means that the database have a consistent view of data in a multi user system. Undo retention management refers to how long already committed undo information is to be retained in the undo segments. If a user updates a data block that changed data is not permanent and available to all other users until it is committed. Till the data is committed, it is stored in undo segment or you can say undo tablespace. Whenever any other user wants to read the data that is being changed, database will pick it up from the undo segment, not from the database buffer cache where it is replaced with new data. Now there is this parameter known as ‘UNDO\_RETENTION’ and this specifies the value in seconds about how long data should be in undo segment after it is being committed. Data has to be in undo segment after being committed for several reasons but most important of them all is for the flashback functionality. If this parameter is not set or set to zero, Oracle will keep the data for as long as it can anyway. The algorithm controlling the overwriting of undo data is set to overwrite the oldest bit of data first but it favours transactions over queries.

Now imagine that you query a piece of data block that has been changed since the query started, it will go to undo segment to find the pre-update version of data as query needs to have consistent view and database will try to find the old data. However if that old data been already over written by another old data because transactions are prioritised over queries, then the query fails with the error “ORA-1555: Snapshot too old”. It happens because of insufficient undo storage (small rollback segment size, before oracle 9i, rollback segments had to be managed by DBAs manually) or a too small value for UNDO\_RETENTION parameter.

This error can be resolved by several remedies:

* DBA or users can reschedule long running queries when the system has less DML load.
* DBA can increase the size of rollback segment so that it can hold more data and new data does not need to overwrite any previously changed data. Rollback segment in Oracle is a structure that stores undo information for transactions and queries. (for versions before Oracle 9i)
* Avoid fetching data between commits. This is difficult in a multi user-system.
* DBA needs to analyse the working patterns of his database users and then he should be able to decide the values for undo management parameters such as UNDO\_RETENTION. It takes good analytical skills to determine the values for these parameters. If the database is in automatic undo management mode, increasing the size of UNDO\_RETENTION will help. Otherwise, use larger rollback segments.

**Requirement 8:** Discuss how different database failures can impact a business, including the financial costs, and what can be done to minimise the downtime cost. To illustrate your point, find a real-life example, based on your research, of the financial costs suffered by a company due to an Oracle systems failure; what lessons were learnt and what actions were taken to prevent future problems.

**Implementation:**

Businesses that run mission-critical applications, often make a lot of investments in buying out the resources to complement their processes. Sadly, despite their efforts of investment in infrastructure robustness, many IT organisations still face database, hardware and software failures causing outage and downtime that range from just a few minutes to even days in worst scenarios causing tremendous losses.

Database failures impact businesses in many ways. First thing a data outage does is the disruption in productivity. Lost files may take hours or even days to recover leading to staff downtime and lost sales. According to Strategic Resource Institute, companies who do not recover from data outage within 10 days, might not survive. Downtime also damages the reputation of a business drastically. Clients do not put their trust in such companies which have a history of downtime and data loss incidents in the past, resulting in degradation of business reputation permanently. Some organizations spend hundreds of thousands of pounds repairing their brand after a data loss event. A database failure can be due to a cyber attach which can result in exposure of confidential information. If customer data or employee records are compromised, the information Commissioner’s Office may fine your company up to 500,000 GBP for not complying with the Data Protection Act. If the individuals also start to take legal actions then the resulting fines are compounded. Customer loyalty is also demolished by the exposure of data as current customers would want to shift their business somewhere else and as word spreads, new customers would not be coming to you as an option. Database failure can result in economic and non-financial impacts on your company. Permanent business failure is the worst possible outcome. Every year a handful of companies shut down their business due to a data loss event. Summarizing it out, companies can face tremendous financial losses and non-financial impacts, it is the cautionary practices and data recovery planning that saves the day at the end.

In many cases DBA are under immense pressure and these are some statistics that speak to the unenviable position they often find themselves in:

* Database downtime results in loss of average $7900 per minute (Enterprise level organisations)
* Human error constitutes 80 percent of the outages
* 50 percent of outages occur during release integration or change configuration
* 90 minutes is average duration of database downtime
* On average, companies suffer 97 hours of downtime per year

There are four things every DBA must learn to become a hero in scenarios of data outages:

* Review: Only start your migration after you have reviewed the current systems and thought about the existing environments and the impact it would have on all work flows.
* Schedule smartly: You have to schedule resource-intensive operations during off hours so that programmers or any type of customers do not have to face downtime.
* Keep traffic moving: Facilitate coexistence between old and new systems so that all users can carry on their tasks.
* Have a backup plan: Always keep an eye on your recovery and backup plan to make sure that no data is lost. A DBA should always be ready to perform data restore even in mid-project.

Delta Airlines had to cancel 280 flights due to an outage in 2017, the losses reached over 150 million just due to this one incident. People had reached over to the system alarming them about the outdated legacy IT systems being used. Infrastructure needed to be updated urgently but these systems have to be online round the clock whole year so changing them is one difficult challenge. The company did not provide details about the cause of outage, saying Delta’s “essential IT systems” went down around 6:30 pm Eastern on Sunday. The systems were restored “a few hours later” and were normal shortly after midnight. This is a nightmare situation for any company to be in because an airline just not have to bear expense for repairs and lost reputation, but also have to refund and reschedule affected customers. A data outage incident gives IT professional at least few points to worry about - in this mentioned scenario, upgradation of mission critical systems is a highly delicate and difficult job. These type of applications need to run 24/7 whole year so they do not afford downtime. This can only be done if infrastructure facilitate coexistence of older and newer versions so that no customer have to face database downtime for smooth running of systems.

## Evaluation:

In the first part of this section, I will be evaluating the system whose requirements I had just implemented. In the second part I will be evaluating the proceedings of this whole coursework that I managed to finish within due date.

We have upgraded the database by addition of two new departments - Customer Service and Inventory. There were several requirements to be implemented and I have implemented each and every one of it. New roles were created in order to give only the necessary privileges to users hence following the principle of least privilege. As all the requirements are met at the database level, there needs to be an implementation for client application of any type to consume these services. A software application or web application that connects to this oracle database so that users can modify or read data from it. It is obvious that employees would not be using SQL developer or SQL plus for their daily operations, they would need a sophisticated GUI based implementation which leverages the database. Both of the schemas seems to track specific piece of information. OE schema deals mostly with the orders while the SH schema deals with the sales. Both of these schemas follow poorly managed pattern for database design. If more efficiency has to be taken from them then they should have either star or snowflake schema for reporting and analyses purposes. Most of the data in tables do not change thus the only data that changes most often has to be placed in one table and non-changing data in all others. This way the system takes less time for long running queries hence making the job of reporting fairly easy.

This part of evaluation focuses on my plan and actions in order to build the required system. I had planned the implementation of all 8 requirements in 3 stages. The first stage deals with the first 4 requirements which were fairly easy for me to implement and test. Those were pretty straight forward in which I had run some fairly complex SQL commands in order to fulfil the needs. Creation of new roles, profiles and users should be fairly simple for any DBA as these are really basic things which they might be required to do. The second stage deals with 5th to 7th requirement. In 5th requirement I had to come up with an idea to extract reports from the system without use of same query over and over again and solution of creating personalised view seemed to be the best. For the 6th I had to recall my knowledge of flashback technology and visualised the scenario to the fullest in order to think of a real solution. For 7th requirement the issue was a fairly common one and its solution was also discussed at many forums. The last part dealt with the end discussion which basically wanted us to be aware of all the delicacies and dangers of opting up for the career of DBA as they have immense pressure on their shoulders and their one silly mistake can cause losses of millions of pounds.

## References:

1. <http://www.dba-oracle.com/t_ora_01555_snapshot_old.htm>
2. <https://uk.reuters.com/article/us-tech-ces-delta-air/delta-wants-to-use-tech-to-improve-flying-experience-idUKKBN1Z62KH>
3. <https://www.datacenterknowledge.com/archives/2017/01/30/delta-cancels-280-flights-due-to-it-outage#menu>
4. <https://www.evolven.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html>
5. <https://www.archivestorage.net/the-5-ways-data-loss-impacts-your-business/>