Table of Contents

Table of Figures

Introduction

This report dives into the work life of a Database Administrator (DBA) in a midsized company, the report starts with common DBA requirements that are needed in the job and shows how to implement such requirements. After that, some problems that might face the DBA will be discussed along with possible solutions for such problems, and then the report examines some backup modes that must be within the knowledge of a DBA.

The report also includes a case study about database failures and what is the aftermath of such failures in real life situations and how to best protect the database against such failures, the report concludes with an evaluation of the database system discussed for the midsized company.

DBA Requirements Tasks

Creating Tablespaces

a. Creating the INV_TSP Tablespace

The code used to create the INV_TSP tablespace is as follows:

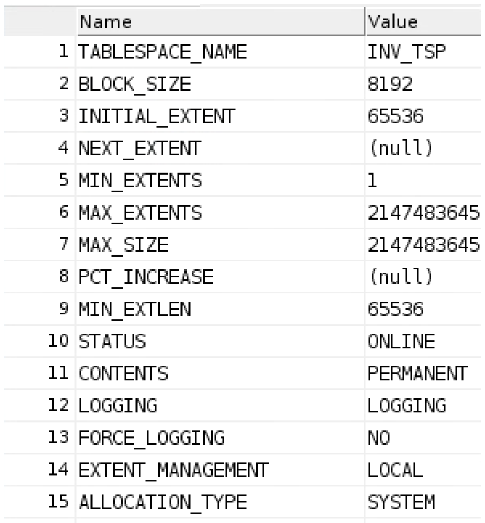
```
Create tablespace inv_tsp
datafile '/u01/app/oracle/oradata/ZK0456L/inv_tsp.df'
size 100M
Autoextend on
extent management local
segment space management Auto;
```

The size of the tablespace was set to 100 MBs to accommodate at least a week's data into it without the need of extending the size whether automatically or manually, approximately 70 MBs of data would be inserted into the tablespace in a week's time, with 100 MBs the storage would be more than sufficient for that. As for the "autoextend" property, it was set to on to make the tablespace more flexible when more data is inserted into it and not showing any errors because of storage insufficiency which could impact the productivity of the users working on the tablespace.

The "extent management local" property allows the tablespace to manage free extents within it automatically by keeping a bitmap that records free extents, this allows the database to save more space in the tablespace. The "Segment space management" is set to auto, which acts as a complimentary property for the "extent management" that manages the extents locally within the tablespace.

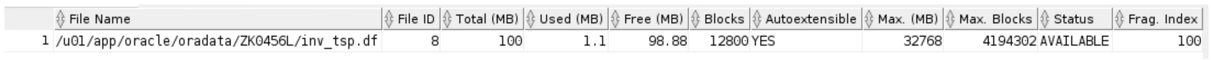
I did not specify a maximum size for the tablespace which means it can keep on auto-extending until it reaches database runs out of storage, which could be a risky approach if not properly monitored to ensure that no user is exceeding the storage limit.

The successful execution of the code results in creating the tablespace, which can be proved in figures 1 and 2 that show the tablespace attributes in the database, meaning that it exists and was created successfully:



	Name	Value
1	TABLESPACE_NAME	INV_TSP
2	BLOCK_SIZE	8192
3	INITIAL_EXTENT	65536
4	NEXT_EXTENT	(null)
5	MIN_EXTENTS	1
6	MAX_EXTENTS	2147483645
7	MAX_SIZE	2147483645
8	PCT_INCREASE	(null)
9	MIN_EXTLEN	65536
10	STATUS	ONLINE
11	CONTENTS	PERMANENT
12	LOGGING	LOGGING
13	FORCE_LOGGING	NO
14	EXTENT_MANAGEMENT	LOCAL
15	ALLOCATION_TYPE	SYSTEM

Figure 1 INV_TSP Attributes



	File Name	File ID	Total (MB)	Used (MB)	Free (MB)	Blocks	Autoextensible	Max. (MB)	Max. Blocks	Status	Frag. Index
1	/u01/app/oracle/oradata/ZK0456L/inv_tsp.df	8	100	1.1	98.88	12800	YES	32768	4194302	AVAILABLE	100

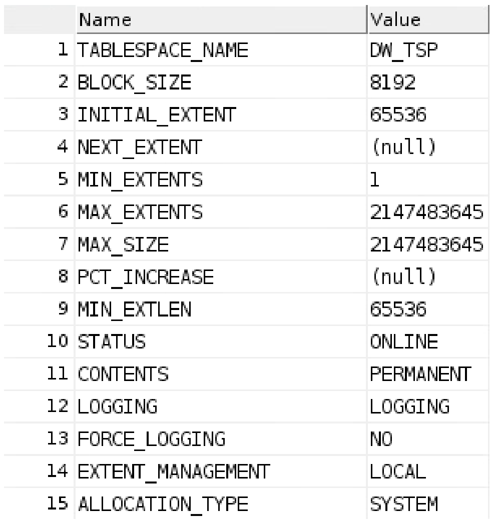
Figure 2 INV_TSP Location and storage attributes

b. Creating the DW_TSP Tablespace

The following code was executed to create the DW_TSP:

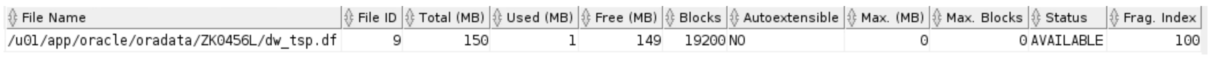
```
Create tablespace dw_tsp
datafile '/u01/app/oracle/oradata/ZK0456L/dw_tsp.df'
size 150M
Autoextend off
extent management local
segment space management Auto;
```

The same properties were used to create this tablespace, but "autoextend" was set to off because this tablespace is created for a data warehousing application and it is not supposed to be changing frequently, this allows the DBA to take more control of the size of the tablespace. The data inserted into the tablespace is approximated to be 120 MBs, meaning that 150 MBs takes a precautionary measure to ensure the tablespace runs efficiently even if the estimated data is fully inserted.



	Name	Value
1	TABLESPACE_NAME	DW_TSP
2	BLOCK_SIZE	8192
3	INITIAL_EXTENT	65536
4	NEXT_EXTENT	(null)
5	MIN_EXTENTS	1
6	MAX_EXTENTS	2147483645
7	MAX_SIZE	2147483645
8	PCT_INCREASE	(null)
9	MIN_EXTLEN	65536
10	STATUS	ONLINE
11	CONTENTS	PERMANENT
12	LOGGING	LOGGING
13	FORCE_LOGGING	NO
14	EXTENT_MANAGEMENT	LOCAL
15	ALLOCATION_TYPE	SYSTEM

Figure 3 DW_TSP Attributes



File Name	File ID	Total (MB)	Used (MB)	Free (MB)	Blocks	Autoextensible	Max. (MB)	Max. Blocks	Status	Frag. Index
/u01/app/oracle/oradata/ZK0456L/dw_tsp.df	9	150	1	149	19200	NO	0	0	AVAILABLE	100

Figure 4 DW_TSP Location and storage attributes

Creating the users and the function, profile, and roles associated with them

The DBA is asked to create new users that comply with specific login requirements. To meet these requirements, a password function, a profile, and a couple of roles were created as well to ensure that the users comply with the requirements easily and efficiently.

1. Creating the password verification function

Oracle provides pre-defined password functions that are suitable for password complexity measures that go beyond the length of the password, which can be used as an alternative as it adds more security measures to the passwords. By looking at these functions and examining how they are implemented, a password function was created that acts similarly to these functions, but it only implements the specified requirement which is to ensure the passwords created by the new users are always at least 8 characters long.

The function must accept three parameters representing the username, new password, and old password (Shavadze, 2017). The function must also return a Boolean value that returns "True" value if the password meets the function requirements and "False" otherwise. The code below was executed to compile and create the function:

```
create function pswd_length_max(
  username      VARCHAR2,
  password      VARCHAR2,
  old_password  VARCHAR2)
  return boolean is
begin
  if length(password) < 8 then
    return false;
    else return true;
    end if;
end;
/
```

This code is a PL/SQL code that allows the DBA to add decision making arguments while using SQL, the if statement in this code is an example of decision making as it returns "True" or "False" based on a value entered by the user and evaluated by the compiler without the interference of the DBA.

2. Creating the profile for all users

Profiles are used to enforce restrictions on users, with each user only allowed one active profile. Creating the profile at this stage is important because the users must have a required usage limitation. The following code was executed to create the ALL_USERS profile:

```
Create profile All_Users Limit
password_verify_function pswd_length_max
idle_time 15
password_life_time 180
failed_login_attempts 3
password_lock_time 1.25
password_reuse_max 1
password_reuse_time unlimited;
```

To implement the function created in the previous step, a “password_verify_function” statement must be included before the name of the function so that the Oracle environment would understand that this function is used as a password verification measure.

The rest of the limitations are pre-defined by oracle; “idle_time” sets the number of minutes that users are allowed when leaving their session without any interactions, “password_life_time” sets the number of days allowed before a password change is required for the users, “failed_login_attempts” sets the number of failed logins allowed before the user account is locked by the database, “password_lock_time” sets the number of days a database lock would take place; the value 1.25 means that the account would be locked for exactly 30 hours if not manually unlocked by the DBA, “password_reuse_max” and “password_reuse_time” are used in this profile as a combination of 1 and “unlimited” values, this combination results in ensuring that the users can never use the same password twice in the database.

3. Creating the users

The following code was executed to create all three users as required, two users for the customer service department and one user of the inventory department:

```
create user CSUSER1
identified by cspasswrd1
default tablespace inv_tsp
quota 15M on inv_tsp
profile All_users
password expire;

create user CSUSER2 identified by cspasswrd2
default tablespace inv_tsp
quota 15M on inv_tsp
profile All_users
password expire;

create user Invuser identified by invpassword12
default tablespace inv_tsp
quota 50M on inv_tsp
profile All_users
password expire;
```

All three users were created with the newly created INV_TSP tablespace as a default tablespace and assigned the “All_users” profile created previously to implement the resource limits on the users, the customer service department users were given a quota of 15 MBs each as they would not need much storage because of the nature of their work as they are only allowed to read data from tables, while the inventory department user was given 50 MBs quota on the tablespace. The “password expire” statement in the end of each user creation block will force the users to change their password on first login to the database.

Please note that the user INVUSER was used to test on, therefore the password has changed from the one originally used to create it, “password expire” was re-executed on the user after the tests.

4. Creating the roles

The following roles were created to simplify the process of granting access rights and privileges to the users, it also simplifies the process of expanding the database when creating new users by assigning them the roles instead of granting them specific access rights or privileges. The following code was executed to create two roles, a customer service department role, and an inventory department role:

```
create role customer_Service;
create role inventory;
```

Creating a materialized view for the customer service department

A materialized view was created for the customer service users under the name CS, this materialized view provides information about all the orders from the tables they have access to, selecting the essential columns from the tables and joining them together to ensure that there is no misuse of the data, the materialized view was chosen over a normal view because it is more efficient and saves more time as all the tables used are indexed and the data from these tables can be fetched quickly. The following code was executed to create the materialized view:

```
Create Materialized View CS as
select c.customer_id, c.cust_first_name, c.cust_last_name, c.cust_email,
o.order_id, o.order_date, o.order_mode, o.order_status,
a.product_id, a.unit_price, a.quantity,
p.product_name, p.product_description, p.warranty_period, p.product_status, p.list_price
from oe.customers c, oe.orders o, oe.order_items a, oe.product_information p
where c.customer_id = o.customer_id and o.order_id = a.order_id and a.product_id = p.product_id;
```

The customer service users would need certain privileges to access the CS materialized view, such access would be granted later in the report, Customer service users can easily fetch data for a specific customer by running the following code and replacing the "customer_id" value to the desired customer id:

```
Select * from sys.CS where customer_id = 3228;
```

Granting necessary access rights and privileges to the users

The DBA needs to maintain the principle of least privileges when granting access rights and privileges, this guarantees that the system is more protected from misuse of privileges.

1. Granting the roles to their respective users

At first, roles need to granted to the users that belong to these roles, this can be done by granting the "customer_service" role to the customer service department users, as well as granting the "inventory" role to the inventory department user. The following code does this process:

```
grant customer_service to CSUSER1, CSUSER2;
grant inventory to Invuser;
```

2. Granting necessary system privileges to the users

System privileges allows the users to perform actions in the database that are not specified on specific objects in the database, such privileges are implemented in the database by granting all users the "create_session" system privilege which allows the users to log into the database. Other system privileges have been granted to the inventory department user allowing it to create tables, and views as required, in addition to those, creating materialized views was granted as well, these privileges allow the inventory department users to only create objects under their own schema.

The code was executed to be applied on the roles which would then be automatically applied to any user with these roles, the code is as follows:

```
grant create session to customer_service, inventory;
grant create table, create view, create materialized view to Inventory;
```

3. Granting necessary object privileges to the users

Object privileges allow the users to access and alter specific objects within the database, and they vary from object to object, where the owner of the object would have all privileges on that object and can grant any privileges on that object to any user or role. As per the requirements, object privileges were granted to the users, with each object having a separate line of code because they cannot be granted on multiple objects. The following code grants required object privileges:

```
grant select on oe.PRODUCT_INFORMATION to customer_Service;
grant select on oe.PRODUCT_DESCRIPTIONS to customer_Service;
grant select on oe.ORDERS to customer_Service, inventory;
grant select on oe.ORDER_ITEMS to customer_Service, inventory;
grant select on oe.customers to customer_Service, inventory;
```

The above lines show the process of granting the read/only object privileges to the customer service and inventory departments, the code below shows the process of granting read/write privileges to the inventory department:

```
grant select, update, insert, delete on oe.Inventories to Inventory;
grant select, update, insert, delete on oe.Warehouses to Inventory;
```

The customer service department needs access on the created materialized view, the role was given read/only access on the materialized view using the following code:

```
grant select on sys.cs to customer_service;
```

Creating the table in inventory department schema

The table required should belong to the schema of the user INVUSER, it also should meet all the specified requirements and constraints.

Creating the INV_CATEGORIES table

The following code was used to create a table under the name “INV_CATEGORIES” that meets all the specified requirements:

```
create table invuser.inv_categories(
cat_id number,
sub_cat_id number,
cat_name varchar(25) constraint catname_NN not null,
sub_cat_name varchar(25) constraint subcatname_NN not null,
cat_description varchar(50) constraint catdesc_NN not null,
available char(1),
constraint invcat_check_avl check (available = 'Y' or available = 'N'),
constraint PK_invcat primary key (cat_id, sub_cat_id))
tablespace inv_tsp;
```

The table was created in the INVUSER schema as indicated by referencing the schema before the table name, the structure of the table was created as requested, with all constraints having user-defined names to make it easier to differentiate the error if there is any constraint that is not met in the future. The tablespace statement after the create table round brackets adds the table created in the specified tablespace which is “INV_TSP”.

Not Null constraints were created by adding the constraint clause before "NOT NULL" as an in-line constraint and adding a name to that constraint to make it user-defined, this was done to all "NOT NULL" constraints. Out-of-line constraints were also added to the table, which are the composite primary key constraint under the name "PK_INVCAT" and the check constraint under the name "INVCAT_CHECK_AVL" which checks whether the "AVAILABLE" column has a "Y" or "N" in it or not, rejecting all other possibilities.

SH user report implementation

The requirement is to provide the SH user with a method to run a report that provides certain information about the products sold for a particular customer based on the customer id entered by the user.

The most suitable option would be to create a "SQL Developer" report and make it accessible to the SH user, by just clicking on it and specifying the customer id the result would appear on the screen. Alternatively, a materialized view can be created where the SH user would need to do a select on and produce the desired output. Both solutions are discussed in this report.

Creating a SQL developer report was preferred because it gives the user the ability to save the report in multiple formats, unlike a materialized view where the user would be forced to save the output script, which is not exclusive to the materialized view and might include other outputs produced previously. Both options are available to the user in the database.

Creating and using the SQL Developer report

SQL developer provides the reports function that can be accessed through the view tab, the creation of this report includes adding a SQL query within the report as well as a bind variable for the customer id to make it flexible to the entry of the SH user. This report is a user-defined master report accessible to all users but only users with access to the table would be able to run it.

To create the report, under the reports tab, click on the "new report" button as shown in figure 5 below:

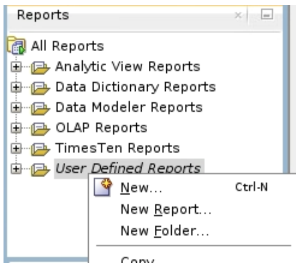
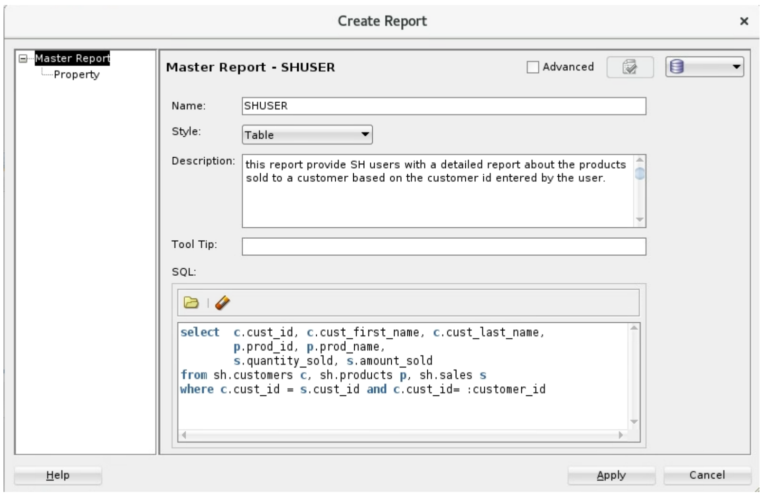


Figure 5 Creating a report

The report would include the following code as an SQL query:

```
select  c.cust_id, c.cust_first_name, c.cust_last_name,
        p.prod_id, p.prod_name,
        s.quantity_sold, s.amount_sold
from sh.customers c, sh.products p, sh.sales s
where c.cust_id = s.cust_id and c.cust_id = :customer_id;
```

The ":" symbol before "customer_id" in the where clause indicates that this variable is a bind variable and the user running the report must input the value manually (Cunningham, 2006). The window of creating the report is as shown in figure 6 below:



Master Report - SHUSER

Name: SHUSER

Style: Table

Description: this report provide SH users with a detailed report about the products sold to a customer based on the customer id entered by the user.

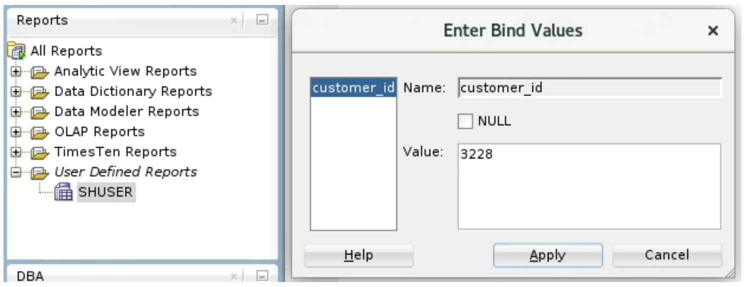
Tool Tip:

SQL:

```sql
select  c.cust_id, c.cust_first_name, c.cust_last_name,
        p.prod_id, p.prod_name,
        s.quantity_sold, s.amount_sold
from sh.customers c, sh.products p, sh.sales s
where c.cust_id = s.cust_id and c.cust_id= :customer_id
```

Figure 6 Create Report Window

After clicking on the report created, a window would appear for the user to enter the value of the bind variable, this window is shown in figure 7 below:



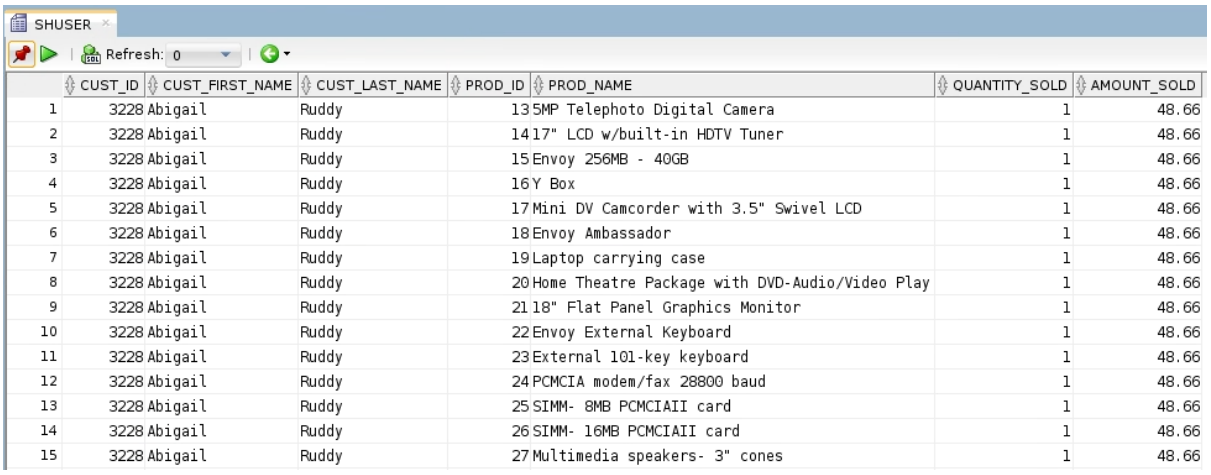
Name: customer_id

NULL

Value: 3228

Figure 7 Bind Variable Window

The user can also save the report in any format desired by right clicking on the report name and clicking on "save as" which would open a window to specify the directory and type of the file. After the execution of the report, the output on the screen is shown in figure 8 below, note that this output was produced by connecting as the SH user:



	CUST_ID	CUST_FIRST_NAME	CUST_LAST_NAME	PROD_ID	PROD_NAME	QUANTITY_SOLD	AMOUNT_SOLD
1	3228	Abigail	Ruddy	13	5MP Telephoto Digital Camera	1	48.66
2	3228	Abigail	Ruddy	14	17" LCD w/built-in HDTV Tuner	1	48.66
3	3228	Abigail	Ruddy	15	Envoy 256MB - 40GB	1	48.66
4	3228	Abigail	Ruddy	16	Y Box	1	48.66
5	3228	Abigail	Ruddy	17	Mini DV Camcorder with 3.5" Swivel LCD	1	48.66
6	3228	Abigail	Ruddy	18	Envoy Ambassador	1	48.66
7	3228	Abigail	Ruddy	19	Laptop carrying case	1	48.66
8	3228	Abigail	Ruddy	20	Home Theatre Package with DVD-Audio/Video Play	1	48.66
9	3228	Abigail	Ruddy	21	18" Flat Panel Graphics Monitor	1	48.66
10	3228	Abigail	Ruddy	22	Envoy External Keyboard	1	48.66
11	3228	Abigail	Ruddy	23	External 101-key keyboard	1	48.66
12	3228	Abigail	Ruddy	24	PCMCIA modem/fax 28800 baud	1	48.66
13	3228	Abigail	Ruddy	25	SIMM- 8MB PCMCIAII card	1	48.66
14	3228	Abigail	Ruddy	26	SIMM- 16MB PCMCIAII card	1	48.66
15	3228	Abigail	Ruddy	27	Multimedia speakers- 3" cones	1	48.66

Figure 8 Report Output

Alternative: Creating the SHUSER materialized view

An alternative to using "SQL Developer" would be to create a materialized view, the materialized view was created in the schema of the SH user by executing the code below:

```
Create Materialized View sh.shuser as
select    c.cust_id, c.cust_first_name, c.cust_last_name,
          p.prod_id, p.prod_name,
          s.quantity_sold, s.amount_sold
from sh.customers c, sh.products p, sh.sales s
where c.cust_id = s.cust_id;
```

For the user to be able to input the customer id, the select statement should be executed as follows:

```
Select * from shuser where cust_id = &Customer_id;
```

The “&” symbol would act as a “:” symbol in the report mentioned previously, the user running the select statement would be asked to enter a value for “customer_id”. Or by entering the value in the where clause, both ways would work efficiently so it is up to the user’s preference.

Problem Solving Requirements

Dropping an important schema problem

This problem addresses a scenario where a user accidently dropped an important schema in the database, dropping a schema is considered as a logical issue that might cause a logical corruption in the database, even if it does not affect the physical datafiles of the database.

The best solution to a logical data failure would be to flashback the database to the last memorable timestamp when the schema was available. Flashback database requires several configurations of the database before it can be implemented; the database must be configured for archiving using the “ARCHIVELOG” clause, it also must have the flashback option enabled, the database must be in “mount” stage when performing the flashback on it. This can be done as follows:

Setting the database in mount stage

This can be done by either executing the code “Alter database mount force;” or shutting down the database then starting it up in mount stage by executing the following code:

```
Shutdown immediate;
startup mount;
```

The second option is preferred as the database would be allowed the time to shutdown and then startup again into the mount stage without the need to force it down to the mount stage which may cause the flashback to not run correctly.

Configuring archivelog and flashback

To set the database into archiving the following code must be executed “alter database archivelog;”. To enable the flashback feature in the database, the code “alter database flashback on;” must be executed.

Performing Flashback database

The Flashback database feature takes the number of days in the argument, thus if the flashback is required to a certain number of days then specify the number of days instead of the “2/24” argument in the code below as it assumes that the flashback is needed to be done for 2 hours before the current time. The code to execute the Flashback database would be as follows:

```
Flashback database to timestamp (sysdate - 2/24)
```

This code sets the flashback to a timestamp which can be replaced with a “to SCN” clause, SCN is a number that is auto generated by the Oracle database after every commit, SCN is Oracle’s primary mechanism to maintain data consistency. Using timestamp in this example approaches a more realistic scenario because the DBA would hardly know the specific SCN number before the schema was dropped. The timestamp used was “(sysdate – 2/24)”, where “sysdate” is the current timestamp and “2/24” means 2 hours.

There are other options that can be used in the timestamp argument, some of which would be typing the timestamp in the system’s format or specifying an interval of certain minutes or hours to subtract from the current timestamp using the “(sysdate – interval ‘XX’ minutes)”.

ORA-1555: Snapshot Too Old error

This error occurs when the read data is changed and at the same time overwritten in the undo segment, in other words, the undo retention period is too short for the database to pick up data from the undo tablespace. This can be avoided by simply increasing the “undo retention” initialization parameter which takes the undo retention period in seconds, this can be done in several ways, either through “SQL PLUS” or “SQL Developer”.

SQL PLUS

To view the current value in the “undo_retention” parameter, run the code:

```
Show parameter undo_retention;
```

The value can be adjusted afterwards to have a value of 1000 for example, specifying the scope in the code to “spfile” means that the change in the parameter will be at a system level. Without specifying the scope, the system would by default make the change in the “spfile” and the memory meaning the scope is set to “both” by default. The execution is shown in the following code:

```
Alter system set undo_retention = 1000;
```

SQL DEVELOPER

The parameter can be viewed and changed in SQL developer by expanding “database configuration” under the DBA tab, clicking on “Initialization Parameters”, and then finding and editing the “undo_retention” parameter, and then committing the change.

Difference between Hot and Cold backup modes

There are two main backup modes for physical backup in the Oracle database system. The first being the online backup mode which is known as hot backup, and the second is the offline backup mode which is known as cold backup. Both backup modes have their benefits and drawbacks as well as their own usage.

A hot backup allows users to do changes on the data while the backup is running, log files that contain the changes made into the system while the backup was running are saved and applied to synchronize the database with the backup after the online backup process finishes. While a cold backup does not allow users to modify the database, making the backup copy and the database consistent and synchronized at all times (Oracle, 2021). A cold backup is recommended to ensure synchronization, a complete comparison between the two backup modes is shown in table 2 below:

	Hot Backup (Online)	**Cold Backup (Offline)**
Consistency	Inconsistent	Consistent
Modifying data during backup	Possible	Not Possible
System Status	Online and available	Offline and not available

Usage	When the system is not allowed downtime	When the system is allowed downtime

Table 1 Hot vs Cold Backup Modes

Database failures case study

There are several types of database failures that can occur in the database, some of these failures are caused on the user level like a "statement failure" due to a constraint violation or any other SQL syntax error, and a "user process failure" that can happen because of an abrupt session end or a terminal failure, the second type of failure specifically is beyond the scope of the DBA.

There are more severe types of database failures such as "user error failure" which is a logical error in the structure, "media failure" where the physical disk or file is corrupted or damaged, and "Instance failure" that can occur because of a power outage or a hardware issue.

These types of failures can have a huge impact on the business side of any organization and might cause financial losses to the organization. There is no evidence of an Oracle database failure available for research, as Oracle claims that the database architecture is completely reliable. This report will examine the financial and productivity costs of a database failure that struck the undisclosed database of the software company "Salesforce" which occurred on May 11, 2016.

The failure was caused by a "data integrity failure", which can be categorized as a "user error failure". The database remained down for almost a full day, and the failure caused a loss of data updates for a duration of approximately 3 and half hours (Tunggal, 2021), this data was lost for multiple clients and could not be restored. Although "Salesforce" never released the financial costs of this incident, experts in the field estimated the financial losses to be around $20 million (Tunggal, 2021), which is a huge amount compared to the actual downtime of the servers. This indicates how critical the reliability of the system is to every business around the world.

Predicting the occurrence of such errors is very hard even though an estimation of the financial losses can be provided by an hour-to-hour revenue analysis, downtime is an unacceptable measure for most organizations around the world, making it more important to have a contingency plan as a precautionary extent for a possible downtime. Avoiding the problem would be the best solution for database failures, but there is no refuting the possibility of a failure taking place.

The best approach to avoid such failures would be to ensure all users are provided focused training on the system. The DBA must be fully aware of the database and must have a monitoring software to help avoid such failures, as well as a solid contingency plan to recover the database, reduce the mean time to recover (MTTR), and sufficient backup and recovery tools in case a failure occurs.

Oracle provides the DBA with a couple of tools to increase the reliability of the system and ensuring that the database does not face any failures frequently, which is known as the mean time between failure (MTBF), these tools are the "Real App Clusters (RAC)" and "Oracle Streams". The DBA must be experienced in using such tools to increase the reliability of the system and reduce the system downtime when it occurs.

Strengths and Weaknesses of the system

This section addresses the strengths of the system and explains some of the decisions made while administrating the changes into the database, as well as the weaknesses of the system and what alternatives that could have been used to strengthen these weaknesses.

Strengths of the system

The strengths of the system are having an extra 25% of storage for the DW_TSP data warehousing tablespace as a precautionary measure, taking advantage of both "password_reuse_time" and "password_reuse_max" by having a combination of values that ensures no password re-use is ever done by any user, having a materialized view that supports the customer service department users, and having both a materialized view and a user-defined report available for the SH user.

Utilizing the user-defined report feature in SQL Developer is a great advantage that provides the desired results efficiently and supports several formats, making it very easy for the SH user to create the report quickly, adding a materialized view to support the same query as the report provides experienced SH users with the ability to perform specific queries on the report for analysis purposes.

Opting to choose a materialized view for the customer service department was a choice that prioritizes performance as the tables used are indexed, making joins faster and hence, making a materialized view more efficient than a view.

Granting every inventory department user the ability to create tables and views by granting the privileges to the role instead of the user might be helpful as all inventory users would have the same job description. Adding a materialized view privilege to the requirements helps experienced inventory users decide when to enhance performance using a materialized view, instead of just being limited to normal views as per the requirement.

Weaknesses of the system

The biggest weakness the system is by far the absence of the backup feature and multiplexed files, even though this system is used for a project and will not be released for real-life usage, it is still an important topic to address. The absence of archiving in the database has several consequences aside from the fact that "ARCHIVELOG" is not enabled, as it affects the Flashback feature making it less efficient for example.

The password verification function can be another weakness as the only requirement is having a password of at least 8 characters long, other pre-defined Oracle password functions provide more complexity measures to the password verification process, making sure no simple passwords or related to the username passwords are accepted for access to the database.

Another weakness would be having the customer service department's CS view in the sys schema, an option to avoid this was to add it to one of the department user's schemas, but since the users are not experienced in SQL, it is safer for the materialized view to be created in a shared schema and granting access to the users or role to avoid altering the materialized view.

References

Cunningham, L. R., 2006. *Making the Most of Oracle SQL Developer Reports.* [Online]
Available at: https://www.oracle.com/technical-resources/articles/cunningham-sqldev.html
[Accessed 07 December 2021].

Oracle, 2021. *Oracle Enterprise Performance Management System Supporting Documentation Backup and Recovery Guide.* [Online]
Available at: https://docs.oracle.com/cd/E57185_01/EPMBK/ch01s02s01s01.html
[Accessed 07 December 2021].

Shavadze, O., 2017. *ORACLE SQL PASSWORD FUNCTION.* [Online]
Available at: https://stackoverflow.com/questions/43158508/oracle-sql-password-function
[Accessed 15 November 2021].

Tunggal, A. T., 2021. *Inside Salesforce.com's $20 Million Dollar Firmware Bug.* [Online]
Available at: https://www.upguard.com/blog/inside-salesforces-20-million-dollar-firmware-bug
[Accessed 25 November 2021].