**Systems and Database Administration – Assignment**

**Jake Bolger, C18395341**

In this assignment I have set up and configured a production ready database for the stockbroker scenario. I have created tables and database objects and have created a report outlining the decisions I have taken.

This report is split into four sections, each section opens with a discussion aimed at stakeholders and describes the motivation behind the decisions along with a step-by-step guide on how to implement them for the database administrators.

Student Number: **C18395341**

**Scenario**

|  |  |
| --- | --- |
| Final Digit of Student Number | Scenario |
| 0, 1, 2, 3, 4 | **Stockbroker** |
| 5, 6, 7, 8, 9 | Human Resources |

First step of this process was to create the database. A new directory called ‘assignment’ was created and where the server could be run, and the database could be created.

Database Creation Steps:

Text

Description automatically generated

Text

Description automatically generated

Database: C18395341\_stockbroker\_db

Text

Description automatically generated

Next, an SQL script called ‘stockbroker\_script.sql’ was created for generating the database. Once this was done five tables were created inside the script for the stockbroker scenario. The tables below were the initial tables created and were modified and shown later in this report.

Text

Description automatically generated

Figure - first table draft

These table were sorted into two different schemas.

To run this SQL script the command below should be run:

Graphical user interface, text

Description automatically generated with medium confidence

## 1. Security (20%)

Nowadays there are quite a few risks and challenges that pose a threat to maintaining the security of a database. In this scenario, a stockbroker database that will be used to back up a trading application could be at risk to the following threats.

**Discuss the major risks and challenges posed to maintaining the security of the database.**

The first risk that could pose a threat to this database is excessive privileges. Excessive privileges are when users are granted privileges that surpass the requirements of their jobs. If a user is granted these privileges, they can be abused. Also, users’ jobs might change, and they might need more privileges and the database might not be updated to fit their needs.

The next risk is legitimate privilege abuse, this is when users abuse the database privileges that are legitimate for nefarious use.

Next is Unmanaged sensitive data. Most of the time, database inventories aren’t accurate. Lots of these databases are forgotten about, and the sensitive data is susceptible to threats as the permissions aren’t implemented correctly or updated.

The final risk or challenge is the human factor. At least 30% of the data breaches are because of human carelessness. This is normally due to lack of experience to implement security on the database.

**Describe the measures that have been taken to combat these.**

For this database, there are two types of users. this means that they will be able to perform different operations than each other on the database. They will have different permissions and privileges to combat the risk of excessive privileges and legitimate privilege abuse.

In this database there were two different schemas created, one called the ‘Internal\_schema’ and another called ‘Public\_info’. These schemas were created to manage which users could access certain information on a table for a particular schema.

Two different roles were then created, one for the traders and one for the customers, this means privilege security could be implemented based on the role a user has. the customers were given a LOGIN which allowed them to be the initial role name for the database connection.

The next step was to alter the users’ privileges. this was done so their privileges don’t surpass the requirements of their job and their privileges can’t be abused, making the database more secure. Both the traders and customers privileges were revoked to begin with, this means that any previously granted privileges are removed on all the tables. This was done for both schemas. After this was done, the users’ privileges were granted. The traders were granted privileges to select on all tables in the ‘Internal\_schema’. The traders were also granted select privileges on the ‘Public\_info’ schema. This is because they need access to view information on all tables. The traders were then granted privileges to insert on tables in the ‘Interna\_schema’, granting them access to the traders table and the positions table. This gives the traders the ability to make trades using company funds. Finally, the customers were granted select privileges on tables from the ‘Public\_info’ schema so they could view information form the companies, portfolio, and prices tables. This means that the customers have read only access to the current positions for any portfolios they have invested in.

The remote access was also blocked to traders to help with the risk of the human factor.

**Outline the policies and procedures which should be put in place after the database has been deployed to maintain the security of the system.**

The policies and procedures that should be put in place making sure the database is checked regular to monitor the different privileges the users are able to use. Some privilege security might have been overlooked and users like the customers might have more privileges and access to the database than they need.

Privileges should be updated regularly, because if someone leaves the company, they shouldn’t be able to access the database. When someone is leaving the company, the privileges need to be change and revoked.

Real time database monitoring should also be put in place to scan for breach attempts so attacks can be reacted to or prevented. Software can be implemented to alert us to any breaches. This can also be done by auditing the database regularly and using penetration tests.

**Include a step-by-step guide to implement the security policies you have chosen.**

Below is a step-by-step guide for implementing the security.

1. First create the schemas for public and internal info.
2. Create the two roles for traders and customers.
3. Revoke all privileges on the two schemas for all users on all tables.
4. Grant desired privileges to customers and traders.

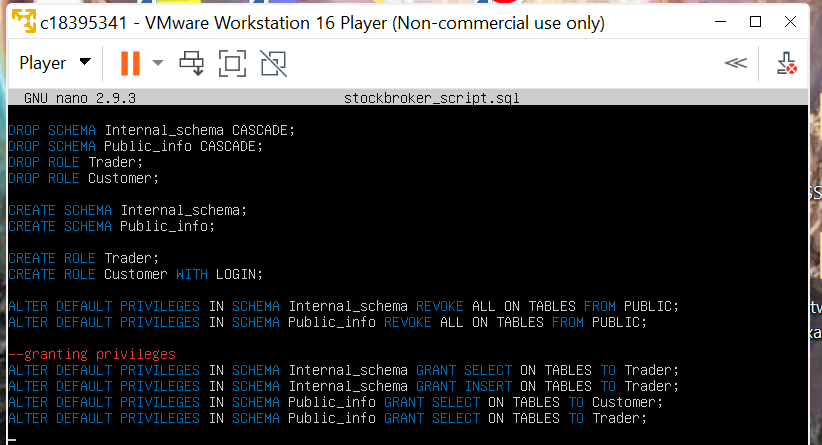


Figure - Schemas, Roles, and Privileges

Assign tables to specific schemas:

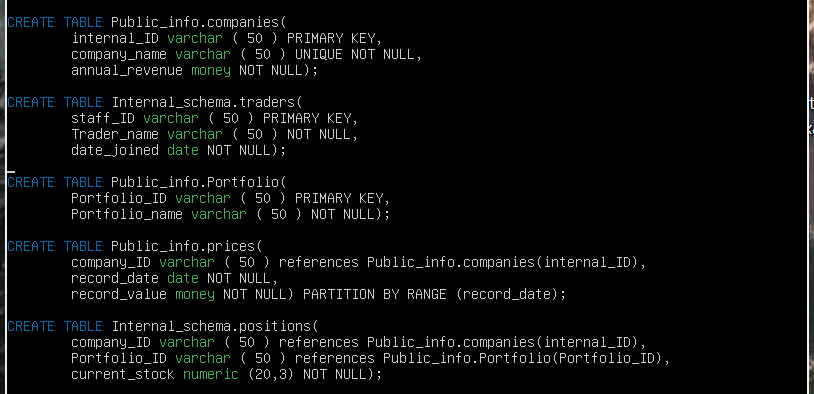


Figure - Tables with Correct Schemas

Now when we sign into the database and check the privileges, we can see them:

**Graphical user interface, text

Description automatically generated**

**Text

Description automatically generated**

## 2. Auditing (20%)

**Discuss the role of auditing within the database.**

In this database auditing needs to be implemented. This will allow us to see what has happened in the system database and examine controls within. In this case the audit will be run by an independent auditor which is us. This allows us to see what the Customers and Traders have been doing such as view data, inserting data and all statements run on the database.

**Discuss the objectives of the auditing policy and how the database will be configured to support these objectives.**

The objectives of the auditing policy that we will implement will provide user accountability, data leakage protection, and the ability to trace actions. The policy will mean that the database actions will be monitored and recorded, such as individual actions, combinations of actions and info, and successful and failed activities.

**Describe the auditing options available, state which option(s) you have chosen and justify your choice.**

In our case we will be looking at the log files which are important in the auditing process. The log files can tell us what was done whoever did it and when they did it. The best way to do this is to use automatic auditing by logging all SQL statements.

This was done by logging the statements to a specific file. First configuration file needed to be found so the database was connected to and the ‘SHOW config\_file;’ command was ran. Once the file was found it was edited and parameters were changed in the file so that we could log the SQL to the ‘postgresql.conf’ file. Once this was done the configuration was reloaded so our database could start logging using ‘SELECT pg\_reload\_conf();’. Once this was done, we are now able to go into the postrgesql csv file and see all the statements and activity of the database.

Next to give us more control over the auditing in the database we installed PgAudit. This will give us better control over the auditing.

**Include a step-by-step guide to implement the auditing steps you have taken.**

Determining the location of the postgresql configuration file:

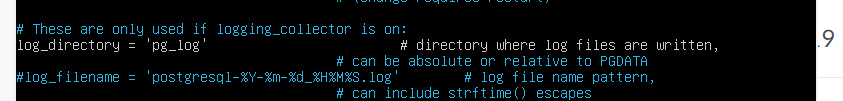
**Text

Description automatically generated**

Now we needed to edit the parameters:

**A screenshot of a computer

Description automatically generated with medium confidence**

****

****

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer

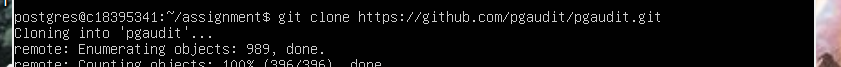
Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated**

Figure - SQL statements being logged

**Installing PgAudit:**

****

## 3. Performance Optimisation (30%)

**Identify the most likely performance bottlenecks and potential performance issues in your database.**

In this database some read operations might be slow when users are querying the database. The data for the customers and traders can be hard to manage and archive, and performance might not be good.

**Discuss the options available to optimise performance.**

To optimise performance and make the read operations faster, indexes can be created. Partitions can improve performance a lot and make it easier to manage and archive data for the customers and traders. Partitioning lets us break down the tables based on a specific value in each row. Most databases will be able to support two types of indexes which are Hash indexing and B-Tree indexing. Hash indexing is cheap and quick to run, and B-Trees are a good way of storing data and retrieving sorted data. the hash index is only suitable for equality operations. B-Tree needs more maintenance and is suitable for any operation use of sorted information.

**State which option(s) you have chosen and justify your choice, clearly stating the benefits and drawbacks of your choice.**

The options I have chosen are hash indexes on the traders table for the staff\_id and the portfolio table for the portfolio id because it allows for quick lookups on the data stored. This is useful if there is a lot of input with similar values or duplicates. It only needs to compare the keys instead of searching through every record.

The options I have chosen for the B-Tree are companies tables for the annual\_revenue and company\_name, and the traders table for date\_joined, the prices table for the record\_values and the positions for the current\_stock. I have chosen these options because it is a good way of retrieving and storing the sorted data for these tables. The B-Tree allows the database to find a leaf node quickly. It is almost instant, making this index very efficient

In this database a ranged partition has been implemented for specific dates of record\_values for the prices table. This uses a different partition for each range band in the columns that were partitioned. In this case the tables split into four different partitions. The steps to implement this are shown in the step-by-step guide.

A tablespace was also created to tell the database what system directory to use to store the files needed for the tables.

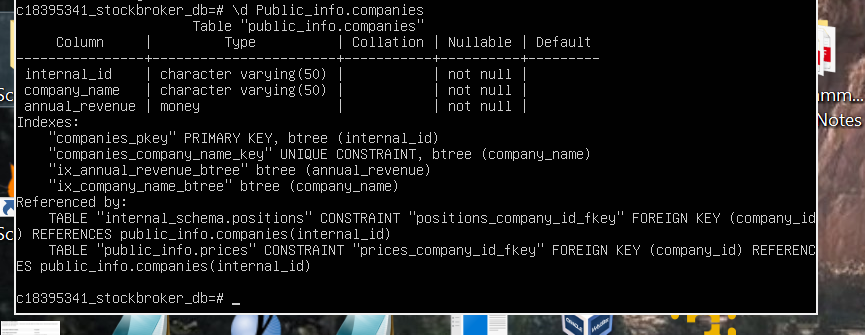
**Include a step-by-step guide to implement the auditing steps you have taken.**

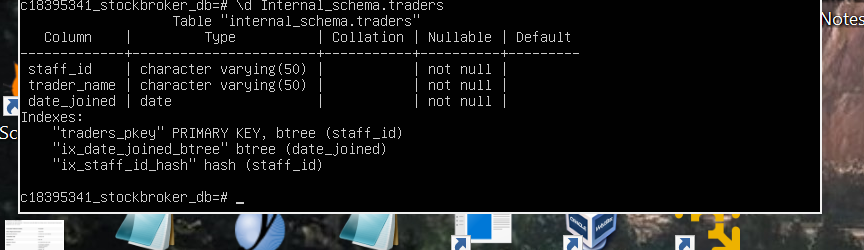
Indexing:

**Graphical user interface, text

Description automatically generated**

Showing indexes:

****

****

**Text

Description automatically generated**

**Graphical user interface

Description automatically generated**

**Graphical user interface

Description automatically generated**

Partitioning table spaces and query optimisation. Creating a tablespace.

Viewing existing tablespaces:

**Text

Description automatically generated**

Create directory to hold the files:

**Text

Description automatically generated**

Setting ownership:

****

Creating the tablespace pointing to the directory I just created.

****

**Text

Description automatically generated**

**If we had rows we could add objects.**

**Partitions used for storing historical data in this part I am partitioning the traders table by date.**

**Partitions:**

****

**Graphical user interface, text

Description automatically generated**

Inserting values and selecting the partitioned tables.

**Text

Description automatically generated**

Then ran select statements to show partitioned tables.

**Text

Description automatically generated**

## Backup / Recovery / Availability Policy (30%)

**Discuss the objectives of the backup / recovery policy in your database.**

For this database, one of the most important aspects is to ensure recovery if there is loss of data. this backup policy will allow us to plan for future failure for the stockbroker database. If the database ever goes down which it will, it needs to be back up and running with as little delay as possible. The database administrators should know all the procedures for when the database goes down and be able to get it back up and running. Network performance should be very good as its an important part of availability. Simulation should be run for all failure situations. A full backup of the stockbroker database should be always available.

**Discuss the options available, state which option(s) you have chosen and justify your choice, clearly stating the associated benefits and drawbacks.**

There are a few backup methods in Postgres which are an SQL Dump, File System Backup, and WAL log Archiving.

The PG Dump tool generates all SQL statements that are used to create or recreate a certain database. It uses a technique called transaction isolation; this means that a snapshot which is consistent with the database is taken. This is an easy method to run, and it also make it easy to backup and restore. To do this you need to run the pg\_dump command along with the database name and the file you want it to create.

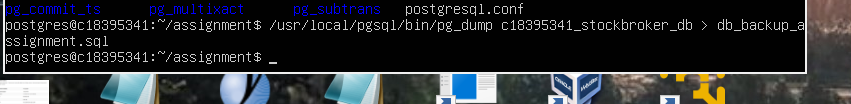
Filesystem level backup is another option available. All the files you need in a database are normally stored in the root directory of Postgres. To do a filesystem level backup we can copy the data directory and get a full backup of the database. To do this you need to use this command: ‘tar -cf backup.tar /usr/local/pgsql/data’. Although this method works there are restrictions such as no partial backups allowed, and the server must be off to allow the snapshot to be consistent. This method is not recommended so it won’t be used in this database.

The next method is WAL File Recovery by archiving it. This can be done by using the archive\_commanf which lets us archive a segment once it is filled. We can use this command to do things like copying the file and putting it on a different location. This method is good as it ensures a consistent snapshot is taken by leveraging the WAL in conjunction with a filesystem backup. However, this method will not be used as it is more complex than the one, we are choosing.

The method that has been chosen to backup the database it the sql\_dump as its efficient, effective, easy, and simple. This is the easiest out of the methods discussed, especially compared to the filesystem backup which is trickier and more prone to errors.

**Describe the configuration steps which have been taken to support this policy.**

**Backup and restoring using pg\_dump:**

****

**Now the backup is created:**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

**Nano:**

****

**Text

Description automatically generated**

Restoring :

**/usr/local/pgsql/bin c18395341\_stockbroker\_db < db\_backup\_assignment.sql**

**Discuss what measures should be taken to increase the availability of the database and prevent downtime. You do not need to provide code or configuration for availability measures but can instead outline the steps the business should take to ensure it.**

The measures that should be taken to increase availability of the database and prevent downtime are to accept or welcome redundancy, automate failover, steer clear of one point of failure, increase the MTBF, decrease the MTTR, and avoiding types of failure.

**Redundancy**

Probably the easiest way to improve data availability is to ensure that your data is redundant, or you have multiple sources of data. As a result, any failure in one of your disks, servers, or databases will not disrupt the availability of your data.

**Automate failover**

A backup component replaces a failing component of your infrastructure automatically when it fails. As a result of automating failover, data availability is minimized or prevented completely as it eliminates the need for humans to detect a problem and switch to a backup system.

**Point of failure**

Keeping single points of failure at a minimum.  which are components and applications that can fail at any time and consequently render your data unavailable

**MTBF and MTTR**

MTBF is the mean time between failures and MTTR is the mean time to recover. MTTR is basically how long the system goes down for after a failure, this can be a disaster as transactions that aren’t recorded can be more dangerous than data loss.

**Failure**

All types of failure should be discussed on a service level agreement. These failures include statement failure, network failure, user errors, media failure, and instance failure.

Statement failure: Avoid invalid data, logic errors, insufficient privilege, and space management.

Network failure: Use load balancing and use more interface cards.

User errors: make sure code is committed.

Media Failure: multiplex datafiles and use archiving like WAL logs.

Instance Failure: make backups to make sure the database is not corrupted when it fails. For example, use an SQL dump.