Systems and Database Administration

C19704439



**Scenario: HR**

You may use the following template for your answers. Some example headings have been given. You may choose to use them or use your own.

# Security

**Reminder: Questions to answer in this section**

What are the major risks/challenges?

What measures have been taken to combat these?

What policies should be put in place after deployment to maintain security?

## Discussion

### Major Risks and Challenges

1. Data Breaches: A data breach is a major risk associated with the HR database, as it contains sensitive personal information of over a million employees. Data breaches can occur due to various reasons, such as a weak password, unsecured data storage, or a cyber-attack. A data breach can lead to the loss of sensitive data, identity theft, financial fraud, reputational damage, and legal liabilities.
2. Unauthorized Access: The HR database is accessible by HR staff from HQ and employees from their respective store locations. This makes it vulnerable to unauthorized access by internal employees or external parties. Unauthorized access can lead to data theft, data tampering, or data loss.
3. Compliance: The HR database must comply with various legal and regulatory requirements, such as the General Data Protection Regulation (GDPR). Failure to comply with these regulations can lead to legal liabilities, reputational damage, and financial penalties.
4. Data Accuracy: The HR database contains employee records, including their position, social security number, and current salary. It is important that this data is accurate and up-to-date, as it can impact the employee's performance, compensation, and benefits.
5. Roles & Permissions: Assigning the correct roles and permissions is critical for ensuring the security of an organization's systems and data. It helps protect against data breaches, reduces the risk of insider threats, and helps maintain compliance with relevant regulations.

### Measures taken to combat risks and challenges

Employees have been granted read-only access to their relevant views, which means they can view their own records, payments, and performance reviews. This ensures that employees are only able to access their own data and not that of their colleagues or other employees. This helps maintain privacy and prevent data breaches caused by employees accessing data that they are not authorized to view.

The HR Staff have been granted greater access to the database. They can view all the data, insert, modify, or delete rows as necessary. This ensures that they have the necessary access to manage the performance reviews table and add payments to the payments table.

This follows the principle of least privilege. This means that employees should only be granted access to the data they need to perform their job duties, and nothing more. This can help prevent accidental or intentional data breaches caused by employees who have been granted excessive access to the database.

### Recommended post-deployment policies

1. Regular Access Reviews: Implement a policy to regularly review and audit the access controls and permissions assigned to employees. This can help ensure that employees only have access to the data they need to perform their job duties, and nothing more.
2. Password Policy: Implement a password policy that requires employees to use strong, complex passwords that are changed regularly. This can help prevent unauthorized access to the database and reduce the risk of data breaches caused by weak passwords.
3. Data Retention Policy: Establish a data retention policy that defines how long data should be stored in the database. This can help prevent the accumulation of unnecessary data and reduce the risk of data breaches caused by outdated or unnecessary data.
4. Data Backup Policy: Implement a data backup policy that ensures that the database is regularly backed up and that backups are stored securely offsite. This can help prevent data loss in the event of a disaster or other unexpected event.
5. Incident Response Plan: Develop an incident response plan that outlines the steps to be taken in the event of a security incident or data breach. This can help ensure that incidents are handled quickly and effectively, minimizing the potential impact on the organization.

## Step-by-step

### Enabling SSL

Edit the postgresql.conf configuration file, which is usually located in the data directory of your PostgreSQL installation.

Look for the following lines in the file:

#ssl = off

#ssl\_cert\_file = 'server.crt'

#ssl\_key\_file = 'server.key'

Uncomment these lines by removing the # symbol at the beginning of each line.

Set the ssl parameter to on and specify the paths to your SSL certificate and key files.

ssl = on

ssl\_cert\_file = '/path/to/server.crt'

ssl\_key\_file = '/path/to/server.key'

Save the changes to postgresql.conf and exit the file.

Restart the PostgreSQL server to apply the changes.

# Auditing

**Reminder: Questions to answer in this section**

Why is auditing important (role of auditing) **for this database**?

What are the objectives of the auditing policy for this database?

How will the database be configured to support these objectives? (high-level discussion)

What options are available?

What have you chosen? Why?

Auditing is important in this HR database for several reasons.

1. Auditing helps to ensure that the data in the database is accurate and reliable. By tracking changes to the data, you can quickly identify any discrepancies or errors, and take steps to correct them before they become larger issues.
2. Helps to identify potential security breaches or unauthorized access to the database. By tracking who has accessed the database and when, you can quickly identify any suspicious activity, such as attempts to access data that the user is not authorized to view or attempts to modify or delete data in the database.
3. Ensures compliance with regulatory requirements. Many industries are subject to regulations that require them to maintain accurate and secure records, and to report on any changes or access to those records. By auditing the database, you can ensure that you are meeting these regulatory requirements and avoid potential fines or penalties for non-compliance.

PostgreSQL 15 offers several options for auditing, including:

1. Logging: PostgreSQL can be configured to log all database activity to files on disk. This includes login attempts, queries, updates, and more.
2. Audit Extension: PostgreSQL provides a third-party extension called "pgAudit" that can be used to capture and log detailed information about database activity. This extension is highly configurable and can be used to track specific events or user actions.
3. Triggers: PostgreSQL's triggers can be used to capture specific events and actions in the database. For example, a trigger can be set up to capture all updates to a specific table.

## Discussion

I have chosen to use an audit extension known as PGAudit[[1]](#footnote-1). This can be configured to capture a wide range of events and actions in the database, including logins, queries, updates, and deletes. You can also configure it to capture specific events based on user, role, or database.

## Step-by-step

To achieve auditing for the HumanResources database follow these steps:

Note – PostgreSQL v15

git clone <https://github.com/pgaudit/pgaudit.git>

cd pgaudit

git checkout REL\_15\_STABLE

make install USE\_PGXS=1 PG\_CONFIG=/usr/pgsql-15/bin/pg\_config

sudo nano /etc/postgresql/15/main/postgresql.conf

logging\_collector = on

log\_destination = 'csvlog'

shared\_preload\_libraries = 'pgaudit'

pgaudit.log = 'ddl, read, write'

sudo systemctl restart postgresql

log in with postgres user

CREATE SCHEMA pgaudit;

CREATE TABLE pgaudit.audit\_log (

event\_time timestamp with time zone,

user\_name text,

command\_tag text,

object\_schema text,

object\_name text,

audit\_type text,

session\_id text,

server\_address text,

client\_address text,

application\_name text,

backend\_type text,

backend\_pid integer,

statement text,

tags text[]

);

GRANT INSERT ON pgaudit.audit\_log TO postgres;

ALTER DATABASE humanresources SET pgaudit.log='all';

SELECT \* FROM pgaudit.audit\_log;

sudo su postgres

cd /var/lib/postgresql/15/main/

cat current\_logfiles

nano into current logfile

# Performance Optimisation

**Reminder: Questions to answer in this section**

What are the most likely performance bottlenecks and potential performance issues in your database?

What options are available to optimise performance?

What have you chosen to implement and why?

What are the benefits and drawbacks of your chosen methods?

## Discussion

Bottlenecks and potential performance issues:

1. Triggers: The use of triggers can slow down database performance, especially if they are complex or run frequently. In this database, there are two triggers (insert\_employee\_trigger) and (create\_store\_view\_trigger) that create views and update other tables. If these triggers are executed frequently or on large numbers of records, they could cause performance issues.
2. Views: The creation of views can also slow down performance, especially if they are complex or involve multiple tables. In this database, the create\_store\_employees\_view function creates a view for each store that shows the employees in that store. If there are a large number of stores or employees, this could cause performance issues.
3. Sequences: There are two sequences (employee\_id\_seq) and (store\_id\_seq) in this database. If these sequences are not set up correctly, they could cause performance issues.

Options available to optimise performance:

1. Use indexing: Indexes can help to speed up queries by allowing the database to quickly find the data it needs. You could create indexes on the columns used most frequently in WHERE clauses or JOIN statements.
2. Upgrade hardware: This could involve increasing the RAM or CPU resources or moving to a faster storage system such as solid-state drives (SSDs).
3. Optimise queries: Query optimisation can significantly improve the performance of a database by reducing the amount of time it takes to retrieve data. This is especially important for large databases with millions of records or when dealing with complex queries.
4. Use caching: Caching can help to improve performance by storing frequently accessed data in memory, rather than querying the database every time.

For my database I have chosen to use both partitioning and indexing. Without an index, the database would have to scan every row of the table which can be time consuming and resource intensive especially for large datasets. Partitioning can be useful in improving the performance of the database because it can reduce the amount of data that needs to be scanned when executing queries.

## Step-by-step

To create Indexes on employees, payments and performance reviews:

CREATE INDEX employees\_store\_id\_idx ON employees (store\_id);  
CREATE INDEX payments\_employee\_id\_idx ON payments (employee\_id);  
CREATE INDEX performance\_reviews\_employee\_id\_idx ON performance\_reviews (employee\_id);

To partition the performance review tables by the year the review was created at:

CREATE TABLE performance\_reviews (  
 employee\_id INT REFERENCES employees(employee\_id),  
 text TEXT,  
 rating INT,  
 bonus INT,  
 created\_at TIMESTAMP NOT NULL DEFAULT *NOW*()  
)  
PARTITION BY RANGE(*EXTRACT*(YEAR FROM created\_at));  
  
-- Create the child tables for each year  
CREATE TABLE performance\_reviews\_2020 PARTITION OF performance\_reviews  
 FOR VALUES FROM (2020) TO (2021);  
  
CREATE TABLE performance\_reviews\_2021 PARTITION OF performance\_reviews  
 FOR VALUES FROM (2021) TO (2022);  
  
CREATE TABLE performance\_reviews\_2022 PARTITION OF performance\_reviews  
 FOR VALUES FROM (2022) TO (2023);  
  
CREATE TABLE performance\_reviews\_2023 PARTITION OF performance\_reviews  
 FOR VALUES FROM (2023) TO (2024);

# Backup/Recovery/Availability

**Reminder: Questions to answer in this section**

What are the objectives of the backup / recovery policy in your database?

What options are available for backup/recovery?

What have you chosen and why?

What are the benefits and drawbacks of your chosen methods?

## Discussion

Options for backup/recovery include:

1. pg\_dump: This is a utility provided with PostgreSQL that allows you to create a text file containing SQL commands that can be used to recreate the database objects and data. It is a simple backup solution, but it may not be suitable for very large databases.
2. pg\_basebackup: This utility allows you to create a backup of a PostgreSQL database cluster. This backup includes all the database objects, data, and configuration files. It can be used for both backup and replication purposes.
3. WAL archiving[[2]](#footnote-2): PostgreSQL provides the ability to archive transaction logs for point-in-time recovery. This requires setting up a backup server and archiving the transaction logs to the backup server.
4. Third-party backup solutions: There are also several third-party backup solutions available for PostgreSQL, such as Barman [[3]](#footnote-3)and pgBackRest[[4]](#footnote-4), that offer additional features and functionalities.

## Step-by-step

1. https://www.pgaudit.org/ [↑](#footnote-ref-1)
2. https://www.postgresql.org/docs/current/continuous-archiving.html [↑](#footnote-ref-2)
3. https://pgbarman.org/ [↑](#footnote-ref-3)
4. https://pgbackrest.org/ [↑](#footnote-ref-4)