Khaula Molapo

Database Week 12

1. Full Logical Backup using pg_dump
To take a full backup of the PostgreSQL sample database 'dvdrental', use this command:

- pg_dump -U postgres -F c -b -v -f dvdrental_backup.sql dvdrental

This command creates a full logical backup file called dvdrental_backup.sql.

2. Create a Table and Insert Rows
After backing up, connect to the database:

- psql -U postgres -d dvdrental

Then run:

```
CREATE TABLE test_table (
   id SERIAL PRIMARY KEY,
   name VARCHAR(50),
   age INT
);

INSERT INTO test_table (name, age)
VALUES ('John', 25), ('Mary', 30), ('Alex', 28);
```

Check data with: SELECT * FROM test_table;

3. Simulate Failure
Now disconnect and delete the database to simulate data loss: dropdb -U postgres dvdrental.

4. Restore Database from Backup
To restore the database from the backup file, use these commands:

- createdb -U postgres dvdrental
- pg_restore -U postgres -d dvdrental -v dvdrental_backup.sql

After restoring, check the tables. The new table 'test_table' is gone because it was created after the backup.

Point-in-Time Recovery

Steps to set up PITR in PostgreSQL:

1. 1. Enable WAL Archiving in postgresql.conf:

```
wal_level = replica
archive_mode = on
archive_command = 'cp %p /var/lib/postgresql/wal_archive/%f'
```

2. 2. Take a Base Backup:

```
pg_basebackup -U postgres -D /var/lib/postgresql/base_backup -Ft -z -P
```

3. 3. Keep WAL files stored safely.
4. 4. Restore to a Point in Time:

Stop PostgreSQL, replace data directory with base backup, add recovery.signal, and set recovery_target_time. Then restart the service.

Performance Analysis

To analyze performance, PostgreSQL provides the pg_stat_statements view, which records information about executed queries. After enabling it in postgresql.conf using shared_preload_libraries = 'pg_stat_statements', run:

```
SELECT query, total_exec_time, calls FROM pg_stat_statements ORDER BY total_exec_time
DESC LIMIT 3;
```

Query	Total Time	Calls
SELECT * FROM rental WHERE customer_id IN (SELECT ...)	5000 ms	10
SELECT * FROM film ORDER BY title	4200 ms	20
UPDATE payment SET amount = amount + 1 WHERE customer_id = 100	3100 ms	5

Example top 3 slow queries:

Optimization Suggestions

5. 1. Query 1 (Nested SELECT):

Problem: Subquery causes slow performance. Fix: Use a JOIN instead.

Example: SELECT r.* FROM rental r JOIN customer c ON r.customer_id = c.customer_id;

6. 2. Query 2 (Sorting large tables):

Problem: Sorting without an index is slow. Fix: Create an index on title.

CREATE INDEX idx_film_title ON film(title);

7. 3. Query 3 (Frequent updates):

Problem: Too many updates on same rows. Fix: Batch updates or use triggers.

Using pg_stat_statements helps identify slow queries. Optimizing with indexing, query rewriting, and batching improves performance, reduces load time, and ensures smoother database operation.