Fernanda Abad

Week 3: Assignment 2

Instructions

Answer the following questions regarding the major topics covered in Chapter 9 in your textbook. This is also useful in preparing for the Midterm Exam this week.

1. What is a transaction?

A transaction is a sequence of operations performed (using one or more SQL statements) on a database as a single logical unit of work. The effects of all the SQL statements in a transaction can be either all committed (applied to the database) or all rolled back (undone from the database). A database transaction must be atomic, consistent, isolated, and durable.

1. Define the four key phrases that control logical changes to the database.

BEGIN starts a transaction;

SAVEPOINT savepointname asks the server to remember the current state of the transaction. This statement can be used only after a BEGIN and before a COMMIT or ROLLBACK; that is, while a transaction is being performed.

COMMIT says that all the elements of the transaction are complete and should now be made persistent and accessible to all concurrent and subsequent transactions.

ROLLBACK [TO savepointname] says that the transaction is to be abandoned, and all changes made to data by that SQL transaction are canceled. The database should appear to all users as if none of the changes had ever occurred since the previous BEGIN, and the transaction is closed. The alternative version, with the addition of the TO clause, allows rollback to a named savepoint and does not complete a transaction.

1. Briefly define the four properties that a transaction must-have.

Atomicity: (all or nothing), a transaction is said to be atomic if a transaction always executes all its actions in one step or does not execute any actions at all It means either all or none of the operations of the transaction are performed.

Consistency: (No violation of integrity constraints), a transaction must preserve the consistency of a database after the execution. The DBMS assumes that this property holds for each transaction. Ensuring this property of a transaction is the responsibility of the user.

Isolation: (concurrent changes invisible), the transactions must behave as if they are executed in isolation. It means that if several transactions are executed concurrently the results must be the same as if they were executed serially in some order. The data used during the execution of a transaction cannot be used by a second transaction until the first one is completed.

Durability: (committed update persist), the effect of completed or committed transactions should persist even after a crash. It means once a transaction commits, the system must guarantee that the result of its operations will never be lost, despite subsequent failures.

1. What is a transaction log and what is it used for?

Every SQL Server database has a transaction log that records all transactions and the database modifications made by each transaction.

The transaction log is a critical component of the database. If there is a system failure, you will need that log to bring your database back to a consistent state.

1. What is a savepoint and what is it used for?

Savepoint is a command in SQL that is used with the rollback command. It is a command in Transaction Control Language that is used to mark the transaction in a table. Savepoint is helpful when we want to roll back only a small part of a table and not the whole table. In simple words, we can say savepoint is a bookmark in SQL.

1. Briefly describe the three major transaction limitations.

**Nesting:** We cannot nest transactions in PostgreSQL (or most other relational databases, for that matter), if you try to execute a BEGIN statement while it’s already in a transaction, a warning message will be produced, telling you a transaction is already in progress.

**Transaction Size:** It is advisable to keep transactions small, a consequence of this is that the parts of a database involved in a transaction frequently need to become locked, to ensure that transactions are kept separate. Therefore, we should try to make sure that each transaction is no larger than it needs to be. Including large amounts of unnecessary changes in each transaction will result in excessive amounts of locking taking place in the database, impacting both performance and other users’ ability to access data.

**Transaction Duration:** Transactions should not be kept open over extended periods. Although PostgreSQL locks the database automatically for you, a long-running transaction usually prevents other users from accessing data involved in the transaction until the transaction is committed or rolled back. Therefore, you should also avoid having a transaction in progress when any user dialogue is required. It is advisable to collect all the information required from the user first, and then process the information in a transaction, unhindered by unpredictable user-response times.

1. Define the different types of reads:
   1. Dirty read occurs when some SQL in a transaction reads data that has been changed by another transaction, but the transaction changing the data has not yet committed its block of work.
   2. Unrepeatable read occurs where a transaction reads a set of data, then later rereads the data and discovers it has changed. This is much less serious than a dirty read, but not quite ideal.
   3. Phantom read occurs when a new row appears in a table while a different transaction is updating the table, and the new row should have been updated but was not.
2. What is a lost update?

Lost updates are slightly different from the previous three cases, which are generally an application-level problems and not related to the way the relational database works. A lost update, on the other hand, occurs when two different changes are written to the database, and the second update causes the first to be lost.

1. What is one way to resolve a lost update?

As a first step, applications should keep transactions as short as possible, never holding them in progress for longer than is necessary. As a second step, applications should write back only data that they have changed. These two steps will prevent many occurrences of lost updates. A more comprehensive way to prevent lost updates is to encode the value you are trying to change in the UPDATE statement.

1. What is the difference between explicit and implicit transactions?

The implicit transaction is auto-commit, there are no beginning and end of the transaction and also no roll back, while the explicit transaction has a beginning and end and rollback command, if an error occurs between transactions, then it can be rollback.

1. Briefly describe the concept of locking.

Locks occur when a process accesses a piece of data where there is a possibility that another concurrent process also needs that data at the same time. When performing the data lock, it is guaranteed that the operation will act as expected, such as a select without using NOLOCK, or inserts, updates, and deletes. The lock guarantees that, when the query is executed, no information covered by it will be changed by another process that uses the data involved in that query.

1. What is a deadlock? Is this good or bad?

A deadlock occurs when a process enters the block and waits for a second process to complete its work and release the locks, while the second process at the same time enters the block and waits for the first process to release the lock, that is, there is a dependency cross between them, which will never be resolved. Deadlocks are considered a critical situation in the database world because processes are being killed automatically. Deadlocks are resolved by the DBMS and do not need manual intervention.

1. What is explicit locking and what is it used for? Should the use of this be encouraged or discouraged and why?

In some cases, you may need to explicitly lock some rows or perhaps an entire table. You should avoid explicit locking if at all possible. It is possible to lock rows or tables only inside a transaction. Once the transaction completes, either with a COMMIT or ROLLBACK, all locks acquired during the transaction will be automatically released.