Recovery and Concurrency Control

Anne Denton

Department of Computer Science
North Dakota State University

Outline

Table of Contents

The concept of a transaction

- Expectations of a DBMS
  - Should handle failures, expected or unexpected
  - Should allow many users to make changes without interfering
    - Concurrency control is topic of next section
- Central concept: Transactions
  - COMMIT ends the transaction and ensures changes are permanent
  - ROLLBACK undoes the transaction and removes all changes
- Log files keep track of active transactions

Failures from which recovery is needed

- Most common types of failures
  - Transaction failures that may nor may not be anticipated
  - Result in ROLLBACK of transaction
- Less common types of failures
  - System failures that leave disk unharmed
  - Includes disk failures for which regular redundancy features allow recovery (e.g. swapping out RAIDed disk)
  - Result in recovery from log file
- Highly uncommon types of failures
  - Catastrophic failures like fire, theft, etc.
  - Recovery from database backup (not simply disk backup) theoretically possible
  - Normally, commercial applications rely on database replication for recovery from catastrophic failure

Question 1 (Multiple answers can be correct)

What is the typical strategy for addressing the problem than clients may walk away from their computer to have a cup of coffee

1. ROLLBACK of a transaction
2. Recovery from a disk log
3. Using a replicated database

COMMIT and ROLLBACK of transactions

- Database programming involves identifying transactions
  - The database programmer when to end (commit) a transaction
  - Should be “atomic" unit of work that is either completely done or not at all
- Committing a transaction
  - SQL statement for ending a transaction is COMMIT
  - The psql client and php use “autocommit" by default, i.e. any individual statement is assumed to be followed by COMMIT
  - Python psycopg2 by default requires an explicit COMMIT statement
  - COMMIT is also done before and after any Data Definition Language statement and when disconnecting from the database
- Rollback can happen for multiple reasons
  - System may abort or rollback a transaction for many reasons
  - ROLLBACK is also a standard SQL statement

Question 2 (Multiple answers can be correct)

Assume that you are using an update statement via psycopg2 (autocommit off by default) to modify a table in a PostgreSQL database. Immediately after that update, can you see the changes through a query using psql (autocommit on by default)?

1. Yes
2. No

Question 3 (Multiple answers can be correct)

Assume that you are using an update statement via psql (autocommit on by default) to modify a table in a PostgreSQL database. Immediately after that update, can you see the changes through a query using psycopg2 (autocommit off by default)?

1. Yes
2. No

Recovery from serious failures

- Recovery from system failures
  - Most system failures result in recovery from the disk log file
    - Also heavily depends on transaction concept
    - Only works when disk is unharmed or RAID recovery possible
  - Backup strategies that are specific to DBMSs can allow recovery from a separate backup
  - Normal backup applied to the database files is not suitable because locking and indexing of data may make files inaccessible!
- Catastrophic failures require contingency planning
  - Typically involves replication of some sort
    - Could be solved as part of a distributed database
    - Could be achieved through cloud deployment
  - Copies of data should be stored in geographically distant locations

Table of Contents

Need for Concurrency Control

- Requests may happen around the same time
- Interleaved processing necessary to achieve performance in multi-user system
- Transaction processing designed to prevent inconsistencies
  - Discussed using concept of schedules
  - Schedules include the read and write statements of all concurrent transactions
- Desirable properties of schedules are
  - Conflict serializability: Prevents problems discussed here
  - Recoverability: Enables the recovery discussed previously

Potential problems requiring concurrency control

- Lost update
  - Change that one transaction makes is overwritten by another
- Dirty read
  - Un-commited information is used
- Incorrect summary
  - Aggregate information is corrupted
- Nonrepeatable read
  - Reading the same record twice may give different results
  - Allows lost update and incorrect summary
- Phantom problem
  - The set of relevant rows changes within a transaction

Lost update

- Result of committed transaction was lost
- Most immediate problem requiring concurrency control
- Consider spreadsheets for tallying grades
    - Instructor records exam grades
    - Instructor sends spreadsheet to grader
    - Grader records assignment grades
    - Instructor records late grade for a student
    - Grader returns spreadsheet
    - Copy with late grade overwritten
- Different solutions for different contexts
    - Blackboard: Uses DBMS (previously Oracle, now PostgreSQL)
    - Writing: Google Docs
    - Programming: Version control (Git, SVN)

Dirty read

- A dirty read accesses uncommitted transaction
- Example: Flight reservation
  - Search says flight is full,
  - You select a different flight
  - Information that flight is full was due to an ongoing other reservation
  - Other transaction aborted
  - You do not find out, that you could have taken flight after all

Incorrect summary

- Inconsistent aggregate information is returned through inappropriate sequence of reads and writes
- Example: Net-worth query during money transfer
  - Money is withdrawn from one of your accounts
  - Transaction that queries net worth reads old and new account
  - Money is credited to a different account of yours
  - Money that was in transit is not accounted for in net worth

`http://highscalability.com/blog/2011/2/10/database-isolation-levels-and-their-effects-on-performance-a.html`

Phantom problem

- Additional record is inserted while other transaction is ongoing
- Example:
    - Search for students with GPA above a threshold for scholarship nomination
    - Transfer student inserted while query is ongoing
    - Student is missed because there were not records to lock when query started
- Hardest to prevent because it relates to existence of records in table
- Requires index locking

Isolation levels in SQL

- SERIALIZABLE
    - Does not allow any of the discussed problems
- REPEATABLE READ
    - Allows phantoms
- READ COMMITTED
    - Allows phantoms, incorrect summary, and lost update
- READ UNCOMMITTED
    - Allows all of the above and dirty read

Question 4 (Multiple answers can be correct)

Dirty read problems can happen for the isolation level

1. READ UNCOMMITTED
2. REPEATABLE READ
3. SERIALIZABLE

Table of Contents

ACID Properties

Atomicity:	Transaction should either be done completely or not at all
Consistency Preservation:	Transaction should take database from one consistent state to another
Isolation:	Transaction should appear as if it was the only one in the system
Durability:	A committed transaction should never be undone

Question 5 (Multiple answers can be correct)

You implement money transfer functionality and fail to notice that the environment in which you are working automatically commits each update (autocommit on). Assuming that the correct behavior would have been to treat all updates as single transaction, which of the following guarantees are no longer satisfied considering the autocommit setting:

1. Atomicity
2. Consistency preservation
3. Isolation
4. Durability

Atomicity

- Atomicity: A transaction is an atomic unit of work
- It is either done completely, or if it cannot be committed, it has to be rolled back completely
- Imagine a money transfer that is coded as a withdrawal and credit
  - Then imagine the withdrawal succeeds but the credit fails
  - The money would be lost
  - To prevent that the transaction has to be coded as withdrawal and credit
- Appropriateness of transactions to match atomicity expectation depends on database programmer!
- Notice that “autocommit" may break up transactions too far

Consistency Preservation

- Consistency Preservation: A transaction is expected to take a database from one consistent state to another
- Applies to any types of constraints
  - Key constraints (primary key, uniqueness, foreign keys)
  - Domain constrains, i.e. data types
  - Semantic constraints defined using explicit constraint specification
- Clearly, one transaction cannot be given the expectation of removing all consistency problems
- However, it has to satisfy the expectation of not introducing inconsistencies

Isolation

- Isolation: A transaction should appear as if it was the only one executed at the time
- Relates to different isolation levels
  - Isolation level SERIALIZABLE corresponds to highest expectations
  - Performance reasons may call for lower levels
- Tradeoff between performance and full support of ACID properties particularly noticeable for Isolation

Durability

- Durability: A transaction that was committed, should never be undone
- Example: Depositing money
  - Assume that you deposit money, but after you leave the bank, the system crashes and the transaction is lost although it was committed

Question 6

Which of the 4 ACID properties of transactions is most clearly violated by the following scenario: Two processes, belonging to two different users, concurrently access the file. Each of the two processes modifies a copy of the file. When they write back the results, one of the modifications is overwritten and lost.

1. Atomicity
2. Consistency Preservation
3. Isolation
4. Durability

Question 7

Which of the 4 ACID properties of transactions is most clearly violated by the following scenario: A user enters a new customer. The application normally tests that customer IDs are unique, but due to of a glitch, two customers are entered with the same ID.

1. Atomicity
2. Consistency Preservation
3. Isolation
4. Durability

Question 8

Which of the 4 ACID properties of transactions is most clearly violated by the following scenario: A user initiates a change, and the application program returns a receipt for that change, indicating that the change is fully completed. However, it eventually turns out that, due to a power failure in the server room, the change hasn't been applied after all.

1. Atomicity
2. Consistency Preservation
3. Isolation
4. Durability

Question 9

Which of the 4 ACID properties of transactions is most clearly violated by the following scenario: A user initiates a transaction that amounts to subtracting a quantity from one field and adding it to another. However, after first part, i.e., the subtraction, is completed the application crashes and the quantity is not being included in either the source or the destination field.

1. Atomicity
2. Consistency Preservation
3. Isolation
4. Durability

Table of Contents

Support for ACID Transactions

- ACID Transactions are default for RDBMSs
- Until a couple of years ago, few people would name them that
- Most No-SQL databases offer transactions with fewer guarantees
    - Important in highly distributed systems, e.g., social networking

Preview of graduate content

- Implementation of transactions discussed in next section (graduate content)
  - Introduces concept of schedules and discusses properties that support concurrency control and recovery
- No-SQL databases discussed in lectures about distributed systems
  - Covers some content on conventional replicated and distributed systems
  - Briefly introduces some of the main types of No-SQL databases
- As with any computer science topics:
  - Content perpetually in flux
  - Having heard of concepts helps organizing knowledge
- Undergraduates are welcome to keep listening!