

[illegible]

Database System Concept (CSE 3103)

Lecture 08-Day 04

Nazmus Sakib, Assistant Professor, Dept. of CSE, AUST

Concurrency Control

- A database must provide a mechanism that will ensure that all possible schedules are both:
 - Conflict serializable.
 - Recoverable and preferably cascadeless
- A policy in which only one transaction can execute at a time generates serial schedules, but provides a poor degree of concurrency
- Concurrency-control schemes tradeoff between the amount of concurrency they allow and the amount of overhead that they incur
- Testing a schedule for serializability *after* it has executed is a little too late!
 - Tests for serializability help us understand why a concurrency control protocol is correct
- **Goal** – to develop concurrency control protocols that will assure serializability.

Weak Levels of Consistency

- Some applications are willing to live with weak levels of consistency, allowing schedules that are not serializable
 - E.g., a read-only transaction that wants to get an approximate total balance of all accounts
 - E.g., database statistics computed for query optimization can be approximate (why?)
 - Such transactions need not be serializable with respect to other transactions
- Tradeoff accuracy for performance

Levels of Consistency in SQL-92

- **Serializable** — default
 - **Repeatable read** — only committed records to be read, repeated reads of same record must return same value. However, a transaction may not be serializable – it may find some records inserted by a transaction but not find others.
 - **Read committed** — only committed records can be read, but successive reads of record may return different (but committed) values.
 - **Read uncommitted** — even uncommitted records may be read.
-
- Lower degrees of consistency useful for gathering approximate information about the database
 - Warning: some database systems do not ensure serializable schedules by default
 - E.g., Oracle and PostgreSQL by default support a level of consistency called snapshot isolation (not part of the SQL standard)

Transaction Definition in SQL

- Data manipulation language must include a construct for specifying the set of actions that comprise a transaction.
- In SQL, a transaction begins implicitly.
- A transaction in SQL ends by:
 - **Commit work** commits current transaction and begins a new one.
 - **Rollback work** causes current transaction to abort.
- In almost all database systems, by default, every SQL statement also commits implicitly if it executes successfully
 - Implicit commit can be turned off by a database directive
 - E.g. in JDBC, `connection.setAutoCommit(false);`