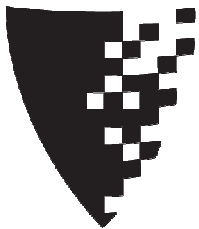


SQL: Transactions

Introduction to Databases

CompSci 316 Fall 2014



DUKE
COMPUTER SCIENCE

Announcements (Thu., Oct. 16)

- **Project Milestone #1** due today
- **Homework #1** grades/feedback released on Sakai
- **Midterm** grades and sample solution to be posted on Sakai by Tuesday

Transactions

- A **transaction** is a sequence of database operations with the following properties (**ACID**):
 - **Atomic**: Operations of a transaction are executed all-or-nothing, and are never left “half-done”
 - **Consistency**: Assume all database constraints are satisfied at the start of a transaction, they should remain satisfied at the end of the transaction
 - **Isolation**: Transactions must behave as if they were executed in complete isolation from each other
 - **Durability**: If the DBMS crashes after a transaction commits, all effects of the transaction must remain in the database when DBMS comes back up

SQL transactions

- A transaction is automatically started when a user executes an SQL statement
- Subsequent statements in the same session are executed as part of this transaction
 - Statements see changes made by earlier ones in the same transaction
 - Statements in other concurrently running transactions do not
- **COMMIT** command commits the transaction
 - Its effects are made final and visible to subsequent transactions
- **ROLLBACK** command aborts the transaction
 - Its effects are undone

Fine prints

- Schema operations (e.g., CREATE TABLE) implicitly commit the current transaction
 - Because it is often difficult to undo a schema operation
- Many DBMS support an **AUTO COMMIT** feature, which automatically commits every single statement
 - You can turn it on/off through the API
 - Examples later in this lecture
 - For PostgreSQL:
 - `psql` command-line processor turns it on by default
 - You can turn it off at the `psql` prompt by typing:
`\set AUTOCOMMIT 'off'`

[Click here to download full PDF material](#)