CSC 355 Database Systems
Lecture 10

Eric J. Schwabe

School of Computing, DePaul University

Spring 2020

Today:

- Finish Transactions
- Midterm Exam Information
- Introduction to Relational Database Design

Transactions

- A *transaction* is a collection of SQL statements that must be executed as a unit
- Transactions have *"ACID"* properties:
  - Atomicity: Execute completely or not at all
  - Consistency: Satisfy all database constraints
  - Isolation: Execute "separately" from others
  - Durability: Once completed, results are permanent

Serializable Isolation

- Transactions must behave as though they were run serially (first one, then the other)
- Usually implemented by “locking” the tables (or parts of tables) used by a transaction
  - Other transactions using the same tables will have to wait until they are released
  - Other transactions using other tables could run at the same time

Read Committed Isolation

- Transaction operations can be interleaved
- If one transaction tries to read data that were written by another, it can only see the changes that have been committed
  - Can’t be rolled back, but could be changed later
  - Multiple queries of the same table might not yield the same results... “non-repeatable reads”, including “phantoms”…

Read Uncommitted Isolation

- Transaction operations can be interleaved
- If one transaction tries to read data that were written by another, it can see all changes, even if they have not been committed
  - Could be rolled back by the other transaction!
  - Transaction might make decisions based on values that are later rolled back and so were never really there … “dirty reads”…

Isolation Levels

- **SERIALIZABLE:** Transactions must appear to run serially – cannot read any changes from others
- **[REPEATABLE READ:** Can read committed changes that only add data (allows only phantoms)]
- **READ COMMITTED:** Can read all committed changes (allows all non-repeatable reads)
- **READ UNCOMMITTED:** Can read all changes (allows all non-repeatable reads and dirty reads)

Transactions in Oracle

- SET TRANSACTION to start a transaction
  - Specify transaction NAME
  - Can also specify ISOLATION LEVEL
    - READ COMMITTED (default) or SERIALIZABLE
  - COMMIT or ROLLBACK ends transaction
  - SQL statements that modify data start a transaction implicitly… COMMIT to end it

Midterm Exam Information

- Exam Monday 5/4, at regular class time (90 minutes)
- Exam will be given as a quiz in d2l
- Lecture slides will be available, but no other sources of information (electronic, printed, or human) may be consulted
  - I will have you sign a statement agreeing to this as part of exam
- Sections 1.1-1.3, 2.1-2.3, 6.1-6.6, 7.1-7.3 covered
- Multiple choice, short answers, writing and/or evaluating SQL queries and transactions
- Review outline has been posted
- Optional Q&A session Friday 5/1, via Zoom

Relational Database Design

- Start with a set of attributes

  $R = \{A_1, A_2, \ldots, A_n\}$

  - (Can also be written as a *universal relation*

    $R(A_1, A_2, \ldots, A_n) \ldots)$

- Construct a *decomposition* of R into relations

  $D = \{R_1, R_2, \ldots, R_m\}$

  - Each R_i is a subset of R

Relational Database Design

- The decomposition D = $\{R_1, R_2, \ldots, R_m\}$ should satisfy the following conditions:
  1. The union of the R_i’s is R
  2. Redundancy has been removed from the R_i’s
  3. Dependencies among attributes in R are preserved
  4. The original relation R can be recovered from D
- Conditions 2.-4. have to be formalized…

Redundancy

- *Redundancy* occurs when more than one record in a table stores the same information
  - Wastes space
  - Allows *update* and *deletion* anomalies
- Remove redundancy by identifying (and removing) *functional dependencies* in R

Functional Dependencies

- A set of attributes Y = $\{Y_1, Y_2, \ldots, Y_n\}$ is *functionally dependent* on a set of attributes X = $\{X_1, X_2, \ldots, X_m\}$ if and only if every pair of tuples that have the same values for X must also have the same values for Y
  - Also "X functionally determines Y" or "X → Y"
  - X is called the *determinant*
- (Less formally, "the values of X uniquely determine the values of Y"…)

Functional Dependencies

- "Every pair of tuples that have the same values on X also have the same values on Y"
  - For X to functionally determine Y, this condition must be *satisfied by every possible relation state*
  - If *some relation state does not satisfy* the condition because two tuples have the same values on X but different values on Y, then X does not functionally determine Y

Finding Functional Dependencies

- DVD (DVDID , MovieID , Title , Genre , Length , Rating)
- GRADING (CNumber , CTitle , SID , SName , Grade)
- (We do not typically include *trivial* functional dependencies, where Y is a subset of X…)

Okay … but why?

- Redundancy comes from functional dependencies whose determinants do not include a complete candidate key of R
- We use the functional dependencies to construct decompositions of R
- How do we measure the quality of the resulting decompositions?

Next:

- Midterm exam Monday 5/4
  - Q&A session Friday 5/1
- Next lecture will be posted Wednesday 5/6