CSE460/560 DATA MODELS AND QUERY LANGUAGES

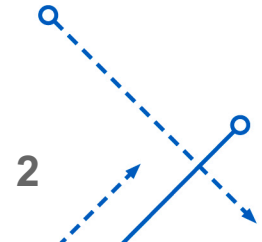
Structured Query Language (SQL) – Advanced

Cheng-En Chuang
(Slides Adopted from Jan Chomicki and Ning Deng)

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

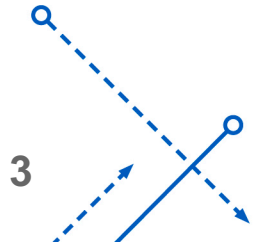
Outline

1. Recursion
2. Transaction
3. Indexes



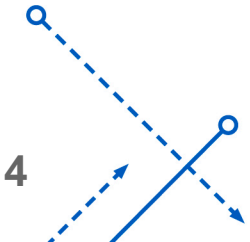
Outline

1. **Recursion**
2. Transaction
3. Indexes



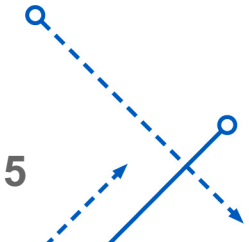
Recursion

- Limitations of Relational Query Languages
  - Cannot express queries involving transitive closure of binary relations
    - List all the ancestors of David
    - Find all the buildings reachable from Davis Hall without going outside
- Solution
  - Recursion



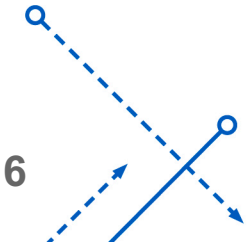
Recursion

- A relations *R* depends on a relation *S* if *S* is used, directly or indirectly, in the definition of *R*
  - In a recursive definition, **a relation may depend on itself!**
- Recursive Views in SQL
  - SQL3 supported only in some DBMS
  - Recursively defined relations should be preceded by RECURSIVE
  - Syntax
    - Given Q1, Q2, Q3 are SQL Query
    - WITH RECURSIVE cte_tbl_name (col1, col2, …) AS (
      Q1
      UNION (ALL|DISTINCT)
      Q2
) Q3



Recursion (Algorithm)

- Initially, the contents of all view are empty
- Evaluate the non-recursive term
- Compute the new-contents of views
  - Using database relations and current contents of the views
    - As the working table is not empty, evaluate the recursive term
- Repeat the previous step until no changes in view contents occur

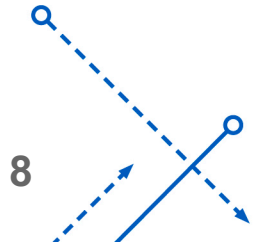


Recursion (Example)

- Parent(parent, child): Find all the ancestors of David
- WITH RECURSIVE Anc(Upper, Lower) AS (
  (SELECT * FROM Parent)
  UNION ALL
  (SELECT P.parent, A.Lower
  FROM Parent P, Anc A
  WHERE P.child=A.upper))
  SELECT Anc. Upper
  FROM Anc
  WHERE Aanc.Lower='David')
- Run until no view can depend on itslf through EXCEPT or aggregation

Outline

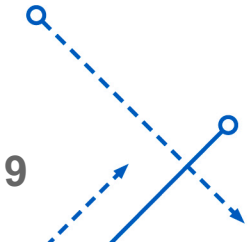
1. Recursion
2. **Transaction**
3. Indexes



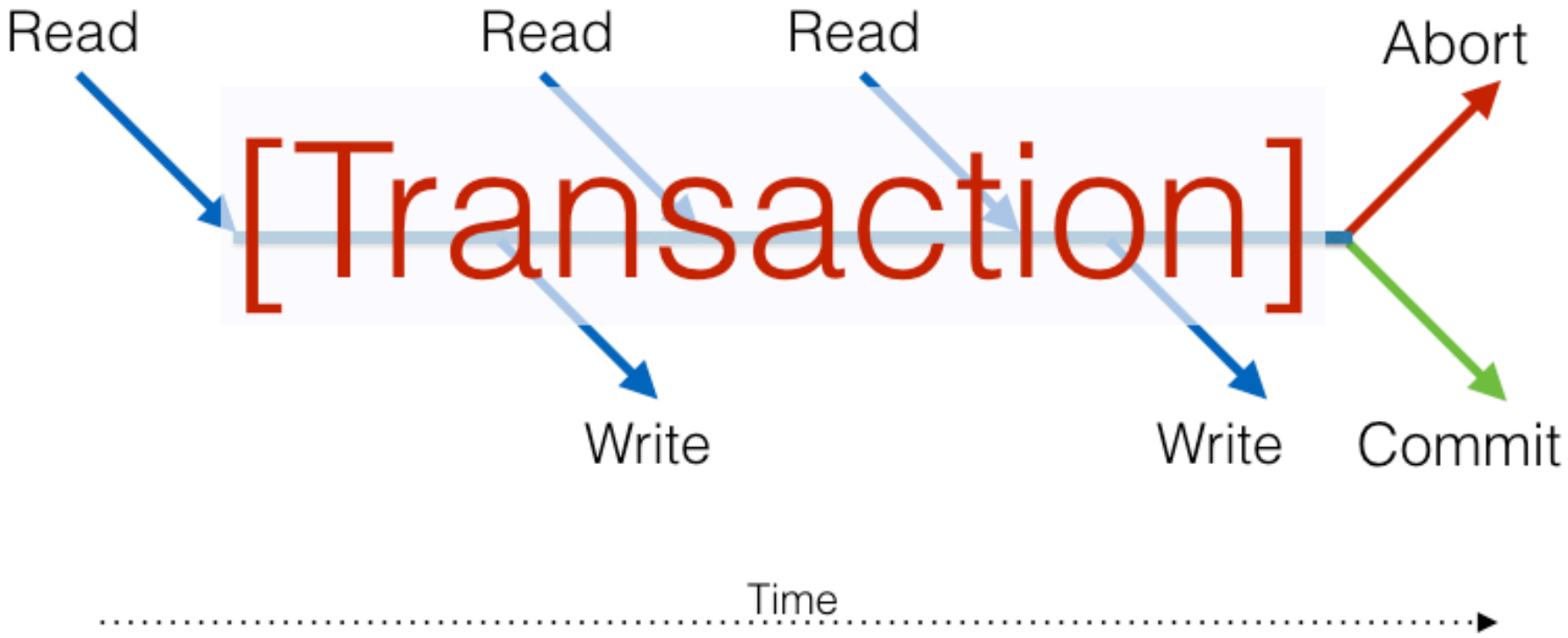
Correctness

- Data Correctness (Constraints)
- Query Correctness (Plan Rewrites)
- Update Correctness (Transactions)

- What could go wrong?
  - Parallelism: What happens if two updates modify the same data?
  - Persistence: What happens if something breaks during an update?



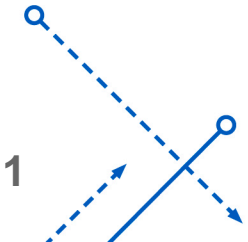
Transaction



A transaction is a collection of one or more operations on the database.

Transaction Correctness

- From a user's perspective
  - Execute fully or none at all (**A**tomicity)
  - Preserve integrity constraints (**C**orrectness)
  - Execute as if on their own (**I**solation)
  - Have their outputs persisted (**D**urability)



Transaction (Example)

- Account(aid, balance)
  - Assume (A, 50) (B, 250)
  - To transfer $100 from B to A
    1. B = B – 100
    2. A = A + 100
  1. In SQL
     1. UPDATE Account SET balance = balance - 100 WHERE aid = 'B';
     2. UPDATE Account SET balance = balance + 100 WHERE aid = 'A';
  - What are possible errors if we use two transfers running concurrently?
    - Ideally (A, 250) (B, 50)
    - Read the same balance of B
      - (A, 250), (B, 150)
    - Read the same balance of A
      - (A, 150), (B, 50)

Transaction (Example)

- Account(aid, balance)
  - Solution – Transaction
  - BEGIN TRANSACTION;
    UPDATE Account SET balance = balance - 100 WHERE aid = 'B';
    UPDATE Account SET balance = balance + 100 WHERE aid = 'A';
    COMMIT;

Outline

1. Recursion
2. Transaction
3. **Indexes**

Indexes

- A data structure that takes the value of one or more fields
- Makes efficient to find records with a value
- Can be “primary” (unique value) or “secondary” (non-unique value)
- Syntax
  - CREATE INDEX idx-name
    ON reln-name(attr1, attr2, ...);
- Example
  - CREATE INDEX idx_pname
    ON Student (LastName, FirstName);

Recommended Reading

Database Systems: The Complete Book
Chapter 6.1 – 6.5