شُروع اَللہ کے پاک نام سے جو بڑا مہر بان نہایت رحم والا ہے

Dr. Abid Sohail Bhutta

abidbhutta@cuilahore.edu.pk

Department of Computer Science,

COMSATS University , Lahore Campus

Database Systems

Lecture 23
Denormailzation, Transaction Management and Concurrency Control

Today's Lecture

- **Denormailzation**
- **Transaction Management**
- **Concurrency Control**

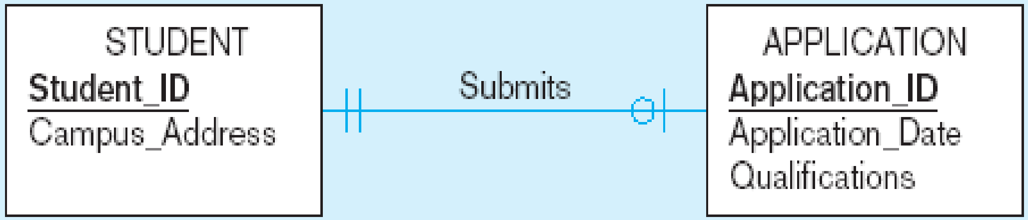
Denormalization

- Normalization is one of many database design goals
- Normalized table requirements
  - Additional processing
  - Loss of system speed

Denormalization

- Transforming ***normalized*** relations into ***unnormalized*** physical record specifications
- Benefits:
  - Can improve performance (speed) by reducing number of table lookups (i.e. *reduce number of necessary join queries*)
- Costs (due to data duplication)
  - Wasted storage space
  - Data integrity/consistency threats

Denormalization situation: two entities with one-to-one relationship



Normalized relations:

STUDENT

Student_ID	Campus_Address

APPLICATION

Application_ID	Application_Date	Qualifications	Student_ID

Denormalized relation:

STUDENT

Student_ID	Campus_Address	Application_Date	Qualifications

and Application_Date and Qualifications may be null

What is a Transaction?

- Transaction changes the contents of the database must alter the database from one consistent state to another.
- Consistent database state is one in which all data integrity constraints are satisfied.
- All transactions are controlled and executed by the DBMS to guarantee database integrity.

Transaction Example

Consider an airline reservation example with the relations:

FLIGHT(FNO, DATE, SRC, DEST, STSOLD, CAP)

CUST(CNAME, ADDR, BAL)

FC(FNO, DATE, CNAME,SPECIAL)

```
Begin_transaction Reservation
begin
    input(flight_no, date, customer_name);
    EXEC SQL  UPDATE   FLIGHT
              SET      STSOLD = STSOLD + 1
              WHERE    FNO = flight_no AND DATE = date;
    EXEC SQL  INSERT
              INTO     FC(FNO, DATE, CNAME, SPECIAL);
              VALUES   (flight_no, date, customer_name, null);
    output("reservation completed")
end . {Reservation}
```

Properties of Transactions

ATOMICITY

- all or nothing

CONSISTENCY

- no violation of integrity constraints

ISOLATION

- concurrent changes invisible È serializable

DURABILITY

- committed updates persist

Transaction Properties

Atomicity

- Either all or none of the transaction's operations are performed.
- Atomicity requires that if a transaction is interrupted by a failure, its partial results must be undone.

Consistency

- Internal consistency
  - A transaction which executes *alone* against a *consistent* database leaves it in a consistent state.
  - Transactions do not violate database integrity constraints.
- Transactions are correct programs

Transaction Properties

Isolation

- Serializability
  - If several transactions are executed concurrently, the results must be the same as if they were executed serially in some order.
- Incomplete results
  - An incomplete transaction cannot reveal its results to other transactions before its commitment.
  - Necessary to avoid cascading aborts.

Isolation Example

- Consider the following two transactions:

T_1:	Read(x)	T_2:	Read(x)
	$x \leftarrow x+1$		$x \leftarrow x+1$
	Write(x)		Write(x)
	Commit		Commit

- Possible execution sequences:

T_1:	Read(x)	T_1:	Read(x)
T_1:	$x \leftarrow x+1$	T_1:	$x \leftarrow x+1$
T_1:	Write(x)	T_2:	Read(x)
T_1:	Commit	T_1:	Write(x)
T_2:	Read(x)	T_2:	$x \leftarrow x+1$
T_2:	$x \leftarrow x+1$	T_2:	Write(x)
T_2:	Write(x)	T_1:	Commit
T_2:	Commit	T_2:	Commit

Transaction Properties

Durability

- Once a transaction commits, the system must guarantee that the results of its operations will never be lost, in spite of subsequent failures.
- Database recovery

Characterization of Transactions

Based on

- Application areas
  - non-distributed vs. distributed
  - compensating transactions
  - heterogeneous transactions
- Timing
  - on-line (short-life) vs batch (long-life)
- Organization of read and write actions
  - two-step
  - restricted
  - action model
- Structure
  - flat (or simple) transactions
  - nested transactions
  - workflows

Concurrency Control

- Coordinates simultaneous transaction execution in multiprocessing database
  - Objective of concurrent control is to ensure serializability of transactions in multi-user database environment
  - Simultaneous execution of transactions over a shared database can create several problems in data integrity and consistency.
  - Potential three problems in multi-user environments
    - Lost updates
    - Uncommitted data
    - Inconsistent retrievals

Concurrency Control

- Lost updates
  - Observation of PRODUCT table (PROD_QOH as a attribute), current value is 35
  - Two transactions occur (T1 and T2)

T1: purchase 100 units → PROD_QOH= PROD_QOH+100
T2: sell 30 units → PROD_QOH= PROD_QOH-30

Lost Updates

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	PROD_QOH = 35 + 100	
3	T1	Write PROD_QOH	135
4	T2	Read PROD_QOH	135
5	T2	PROD_QOH = 135 - 30	
6	T2	Write PROD_QOH	105

If transaction T2 also reads the original value, we have the following case: T1 has not yet been committed when the T2 is executed.

Lost Updates

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T2	Read PROD_QOH	35
3	T1	PROD_QOH = 35 + 100	
4	T2	PROD_QOH = 35 - 30	
5	T1	Write PROD_QOH **(Lost update)**	135
6	T2	Write PROD_QOH	5

Uncommitted Data

- During two transactions, T1 and T2 are executed concurrently and T1 is rollback after the T2 has already accessed the uncommitted data
- Example: T1 rollback to eliminate adding 100 units

T1: purchase 100 units → PROD_QOH= PROD_QOH+100 (rollback)
T2: sell 30 units → PROD_QOH= PROD_QOH-30

Uncommitted Data

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	PROD_QOH = 35 + 100	
3	T1	Write PROD_QOH	135
4	T1	*******ROLLBACK *******	35
5	T2	Read PROD_QOH	35
6	T2	PROD_QOH = 35 - 30	
7	T2	Write PROD_QOH	5

Uncommitted Data

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	PROD_QOH = 35 + 100	
3	T1	Write PROD_QOH	135
4	T2	Read PROD_QOH (Read uncommitted data)	135
5	T2	PROD_QOH = 135 - 30	
6	T1	***** **ROLLBACK** *****	35
7	T2	Write PROD_QOH	105

Uncommitted data problem can arise when the ROLLBACK is complete at T2 has begin its execution.

Inconsistent Retrievals

- Occur when a transaction calculates some summary functions over a set of data while other transactions are updating the data.

T1: calculates the total quantity on hand

T2: at the same time updates the quantity fro two of PRODUCT table's product

Inconsistent Retrievals

Objective: user accidentally added 30 to 345TYX PROD_CODE. We need to correct it. Suppose to add to 125TYZ. ---- Jun Ni's mistake he made!

He promised to correct. Watch what he did!

TRANSACTION T1	TRANSACTION T2
SELECT SUM(PROD_QOH) FROM PRODUCT	UPDATE PRODUCT SET PROD_QOH = PROD_QOH + 30 WHERE PROD_CODE = '125TYZ'
	UPDATE PRODUCT SET PROD_QOH = PROD_QOH - 30 WHERE PROD_CODE = '345TYX'
	COMMIT;

The Scheduler

- How is the correct order? Who determines the order?
- Fortunately, DBMS handles that tricky assignment for us by using a built-in scheduler
- Scheduler establishes order of concurrent transaction execution

The Scheduler

- Interleaves execution of database operations to ensure serializability
- Bases actions on concurrency control algorithms
  - Locking
  - Time stamping
- Ensures efficient use of computer's CPU
  - No scheduling, make transaction first comes first
  - Lost CPU cycles during processing of I/O

Concurrency Control with Locking Methods

- Lock guarantees current transaction exclusive use of data item (T2 does not have to access to the data item that currently being used by transaction t1)
- Acquires lock prior to access
- Lock is released (unlocked) when transaction is completed. So other transactions can lock the data

Concurrency Control with Locking Methods

- DBMS automatically initiates and enforces locking procedures
- Managed by lock manager (built-in DBMS), which is responsible fro assigning and policing the locks used by the transactions
- Lock granularity indicates level of lock use

Lock Granularity

- Lock granularity indicates level of lock use
- Locking can take place at the following levels:
  - Database
  - Table
  - Page
  - Row
  - Even field (attribute)

Problems with Locking

- Transaction schedule may not be serializable
  - Managed through two-phase locking
- Schedule may create deadlocks
- Both problems can be managed by using deadlock detection and prevention techniques

Two-Phase Locking

- Defines how transactions acquire and relinquish locks
- Two-phase locking guarantees serializable, but not prevent deadlock.
- It has two phases:
  - Growing phase
  - Shrinking phase

Two-Phase Locking

- Growing phase:
  - in which a transaction acquires all the required locks without unclocking any data;
  - once all locks have been acquired, the transaction is in its locks point.

Deadlocks

- Occurs when two transactions wait for each other to unlock data

T1 ---- access data items X and Y
T2 ---- access data items Y and X

If T1 has not unlocked data items, T2 can not starts.
If T2 ha not unlocked data items t1 can not continue.

deadly embrace

TIME	TRANSACTION	REPLY	LOCK STATUS	
0			Data X	Data Y
			Unlocked	Unlocked
1	T1:LOCK(X)	OK	Locked	Unlocked
2	T2: LOCK(Y)	OK	Locked	Locked
3	T1:LOCK(Y)	WAIT	Locked	Locked
4	T2:LOCK(X)	WAIT	Locked	Locked
5	T1:LOCK(Y)	WAIT	Locked	Locked
6	T2:LOCK(X)	WAIT	Locked	Locked
7	T1:LOCK(Y)	WAIT	Locked	Locked
8	T2:LOCK(X)	WAIT	Locked	Locked
9	T1:LOCK(Y)	WAIT	Locked	Locked
...	....	....	....	....
...	....	....	....	....
...	....	....	....	....
...	....	....	....	....

Deadlock

- Control techniques
  - Deadlock prevention:
    - abort new lock, if there is the possible that deadlock can occur
    - Rollback and all locks obtained by the transaction are released
    - Reschedule transaction
  - Deadlock detection:
    - DBMS periodically tests the database fro deadlocks
    - If it happens, abort, rollback and restart, others continue

- Control techniques
  - Deadlock avoidance
    - Transaction obtains all the locks needed before it can be executed.
- Experience
  - If the probability of deadlock is low, deadlock detection is recommended
  - If the probability of deadlock is high, deadlock prevention is recommended.
  - If response time is nit high on the system priority, deadlock avoidance might be employed

Database Recovery Management

- Restores a database to previously consistent state
- Based on the atomic transaction property
- Level of backup
  - Full backup
  - Differential (only for modification portion)
  - Transaction log (transaction)

Causes of Database Failure

- Software:
  - software induced failures may be traceable to the operating system, DBMS software, application programs, or virus
- Hardware
  - Hardware induced failure may include memory chip errors, disk crashes, bad disk sectors, disk full errors, and so on

Transaction Log

- Tracks all transactions that update database
- May be used by `ROLLBACK` command for recovering
- May be used to recover from system failure
- Some DBMS (ORACLE) use transaction log to recover a database forward to a current consistent state

Transaction Log

- DBMS executes transactions that modify the database, it also automatically updates the transaction log.

Thanks