Week 6 Tutoring

CSE 180

Midterm 2

- Midterm 2 on Monday, 11/12
- Extra tutoring session on Thursday, 11/9, 2:00-3:00
- Includes Lecture 7 (and earlier) and Lecture 8 (except Triggers)
- Should be able to do everything on the practice midterm, but more will be included
    - Questions that were not on MT1: 5, 7, 9, 10, 11, 13, 16

ACID

Atomicity: all or nothing; all changes in a transaction are performed or none

Consistency: domain of attribute must have certain restrictions and constraints

Isolation: whether or not multiple transactions trip over each other

Durability: after a transaction completes, changes to data persist

Transactions

A transaction is a group of operations that should be executed atomically

The DBMS will execute each transaction in its entirely or not at all, “without transactions interfering with each other”.

START/BEGIN TRANSACTION <ISOLATION LEVEL >

<transaction>

COMMIT;

Dirty Reads

Dirty data refers to data that is written by a transaction but has not yet been committed by the transaction.

A **dirty read** refers to the read of dirty data written by another transaction.

Consider the following transaction T that transfers an amount of money ($X) from one account to another:

1) Add $X to Account 2.
2) Test if Account 1 exists and has at least $X in it.
    a) If there is insufficient money, remove $X from Account 2.
    b) Otherwise, subtract $X from Account 1.

- Transaction T1: Transfers $150 from A1 to A2.
- Transaction T2: Transfers $250 from A2 to A3.
- Initial values: A1: $100, A2: $200, A3: $300.

1) A1: $100 , A2: $200, A3: $300
2) A1: $100 , A2: $200, A3: $300
3) A1: $100 , A2: $350, A3: $300
4) A1: $100 , A2: $350, A3: $550
5) A1: $100 , A2: $100, A3: $550

Isolation Levels

READ UNCOMMITTED

- Allows dirty reads

READ COMMITTED (default)

- Only clean reads
- May read data committed by different transactions (no repeatable reads)

REPEATABLE READ (default)

- Repeated queries will retrieve the same value even if it changed by another transaction

Isolation Levels Cont.

SERIALIZABLE

- As if one-by-one
- Preserves consistency
- Worse response time and throughput

SNAPSHOT ISOLATION

- Along with read committed and repeatable read, them most common
- Better performance but worse consistency than SERIALIZABLE
- Transaction read data as it existed when transaction began

Isolation Levels

Isolation Level	Clean Reads	Repeatable Reads	Simultaneous Existence
READ UNCOMMITTED	N	N	N
READ COMMITTED	Y	N	N
REPEATABLE READ	Y	Y	N
SERIALIZABLE	Y	Y	Y

Snapshot Isolation

- Transaction reads data as it existed when transaction began (hence reads are repeatable).
- Prevents Write conflicts
  - ... but not on Read/Write conflicts between transactions (which Serializability DOES do)

SQL Languages

SQL Languages

Data Manipulation Language (DML)

- Access and modify data
- SELECT, INSERT, DELETE, UPDATE

Data Definition Language (DDL)

- Modify structure of data
- CREATE, DROP, ALTER

Data Control Language (DCL)

- Control access to the data (security)
- GRANT, REVOKE

CREATE/DROP TABLE

CREATE TABLE <table name> (

    Attributes ...

)

DROP TABLE <table name>;

ALTER TABLE

• Adding a column to a table:

ALTER TABLE <table name> ADD <attribute name> <attribute type> <DEFAULT/NOT NULL>;

Dropping a column from a table:

ALTER TABLE <table name> DROP <column name>;

- In some SQL systems, dropping a column isn't allowed.

VIEWS

- Views help with logical data independence, allowing you to retrieve it if it matches the description in the view
- Advantages:
  - short-hand/encapsulation
  - Re-use
  - Authorization
  - Logical data independence

CREATE VIEW <view name>(<attribute names>) AS

SELECT <attributes>

FROM R...

WHERE <conditions>

INDEX

- We use indexes so we can easily search tables
  - Can also make INSERT, DELETE, UPDATE slower
- If a table is updated, all indexes are immediately updated within the transaction
- Preserves physical data independence
  - SQL statements can be executed regardless of which indexes (if any) exist in the database
- What gets impacted when indexes are created or dropped?
  - Performance of SQL statements
  - Some may run faster, some may run slower

Constraints and Triggers

* Triggers will not be included on Midterm 2

Kinds of Constraints

- Primary Key/Unique constraints
- Foreign Key, or referential-integrity constraints
- Attribute-based constraints
  - Constrain values of a particular attribute
- Tuple-based constraints
  - Relationship among components of tuple
- Assertions
  - Any SQL boolean expression (not implemented in most relational DBMS, not discussed in this lecture)

Adding/Dropping FK Constraints

ALTER TABLE R

ADD CONSTRAINT <foreign key name> FOREIGN KEY (<attributes>)

REFERENCES R(<attributes>)

ALTER TABLE R

DROP CONSTRAINT <foreign key name>

INSERT/UPDATE with FK Constraints

Options:

1. RESTRICT: Reject the modification; the default
2. CASCADE: Makes the same changes in all tables that are referenced
3. SET NULL: Change the corresponding tuples that are referenced to NULL

CHECK Constraints

ALTER TABLE R

ADD CONSTRAINT <constraint name>

CHECK (<condition>);