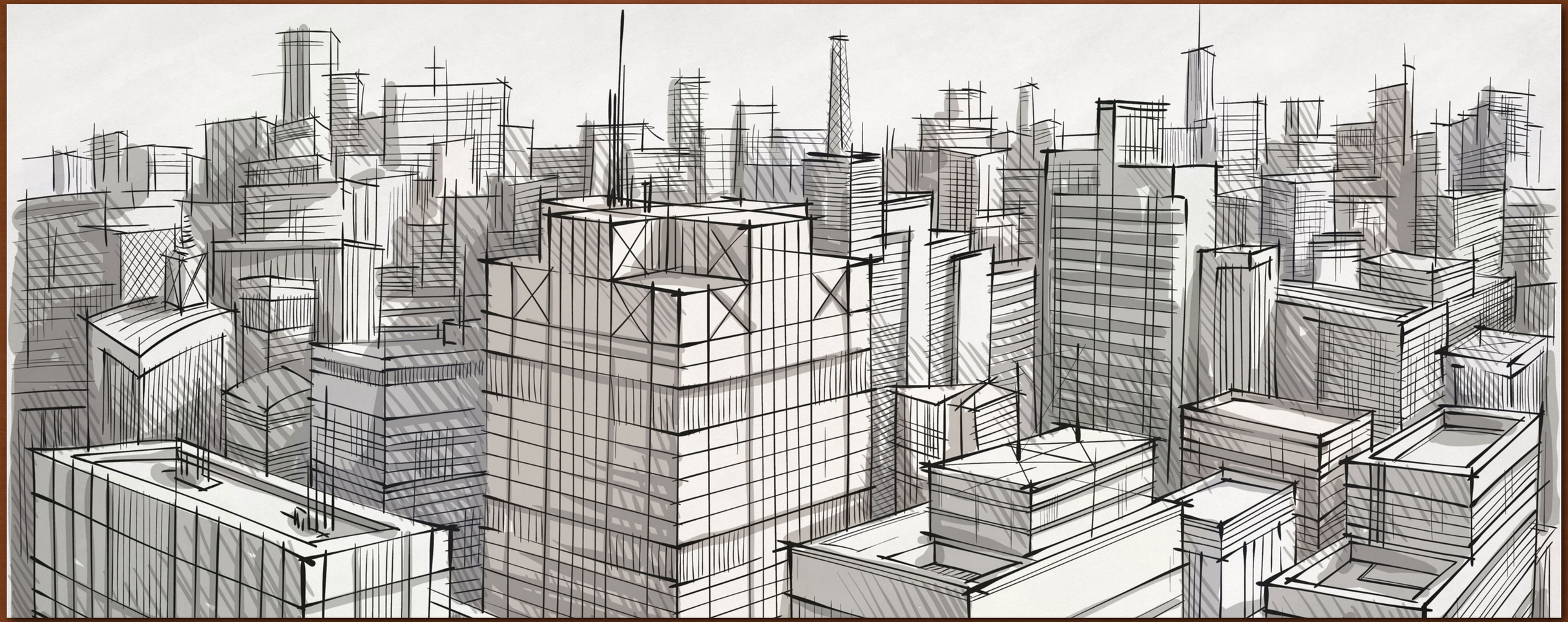
COSC 061

TRANSACTIONS



ATTRIBUTE LEVEL INCONSISTENCY

```
1  /* Client 1 - Deposit */
2  UPDATE CheckingAccount
3  SET Balance = Balance+100
4  WHERE AccountNumber=123456;
5  --------
6  /* Client 2 - Deposit */
7  UPDATE CheckingAccount
8  SET Balance = Balance+150
9  WHERE AccountNumber=123456;
```

RELATION-LEVEL INCONSISTENCY

```
/* Client 1 - Give High Cost Area increases to top performers */
UPDATE Employees
SET Hours = Hours * 1.25
WHERE Rating>90;

/* Client 2 - Tri-State hourly rate Increases for full-time workers */
UPDATE Paychecks
SET Rate = Rate * 1.05
WHERE (State=NY OR State=CT OR State=NJ) AND Empid IN
   (SELECT Empid FROM Employees WHERE Hours>=40);
```

MULTIPLE-STATEMENT INCONSISTENCY

```
/* Client 1 - promote students based on hours earned */
INSERT INTO Seniors (SELECT * FROM Juniors WHERE Hours > 90);
DELETE FROM Juniors WHERE Hours > 90;
...

/* Client 2 - Calculate class sizes */
SELECT COUNT(*) FROM SENIORS;
SELECT COUNT(*) FROM JUNIORS;
```

TRANSACTIONS

SELECT → UPDATE → DELETE → COMMIT → SELECT

A **Transaction** is a Logical unit of work that must be entirely completed or completely aborted.

A Transaction typically begins automatically as the first SQL statement is run.

A COMMIT ends the Transaction.

SELECT → UPDATE → DELETE → COMMIT → SELECT

DON'T DELAY THE TRANSACTION!

```
START TRANSACTION;
-- get input from something or someone
Do some SQL commands using that input;
-- Confirm the results with something or someone
IF (OK?) THEN Commit; ELSE Undo;
```

WHAT DO TRANSACTIONS NEED?

- Atomicity
- Consistency
- Isolation
- Durability

… AND …

- **Serializability**

TRANSACTION SERIALIZABILITY

Serializability means that the actions of transactions may be interleaved, but the end result must be the same as if the transactions were run in some sequential order.

UNCOMMITTED DATABASE UPDATES

If a transaction has written some data to the DB but not yet committed it, that data is *Dirty Data*.

If then another transaction reads that data, that is called a *Dirty Read*.

ISOLATION LEVELS

READ UNCOMMITTED - Dirty Reads are ok; In general, SQL assumes transactions are READ WRITE, except when you allow Dirty Reads. Since that's so risky, SQL assume the transaction is READ ONLY unless you specifically override it.

READ COMMITTED - Forbids reading Dirty Data

REPEATABLE READ - Implies that repeated reads of the same tuple will have identical results. Reasonable, except that the query might return phantom tuples due to some other updates to the database.

TRANSACTION WITH *REPEATABLE READ*

```
---Transaction log
START TRANSACTION;
---
SELECT ... ;
--- Begin some complex calculation that uses the following result
SELECT COUNT (*) FROM ENROLLMENT WHERE ClassDept = "CompSci";
--- do some other stuff, then get that same result again to
--- finish the calculation, and this count had better be the
--- same as before!
SELECT COUNT (*) FROM ENROLLMENT WHERE ClassDept = "CompSci";
--- more stuff
COMMIT; -- This ends the transaction
```

MAYBE TICKET PURCHASING APP USED *READ COMMITTED*

1. Transactions with READ COMMITTED begins
2. Customer chooses seats
3. Transaction commits the selection with the PENDING value set TRUE.
4. A Trigger fires because of the COMMIT and PENDING=TRUE that sets a timer to run while updating the little "Time left" window.
5. Other transactions cannot see those seats because of the COMMIT.
6. If the timer completes, another TRIGGER is initiated which reverses the transaction and COMMITS.