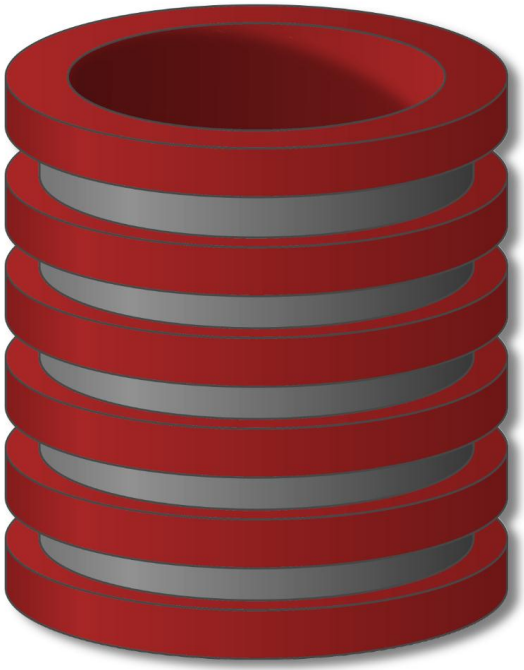


Transactions

Properties

Solution for both concurrency and failures

Transactions

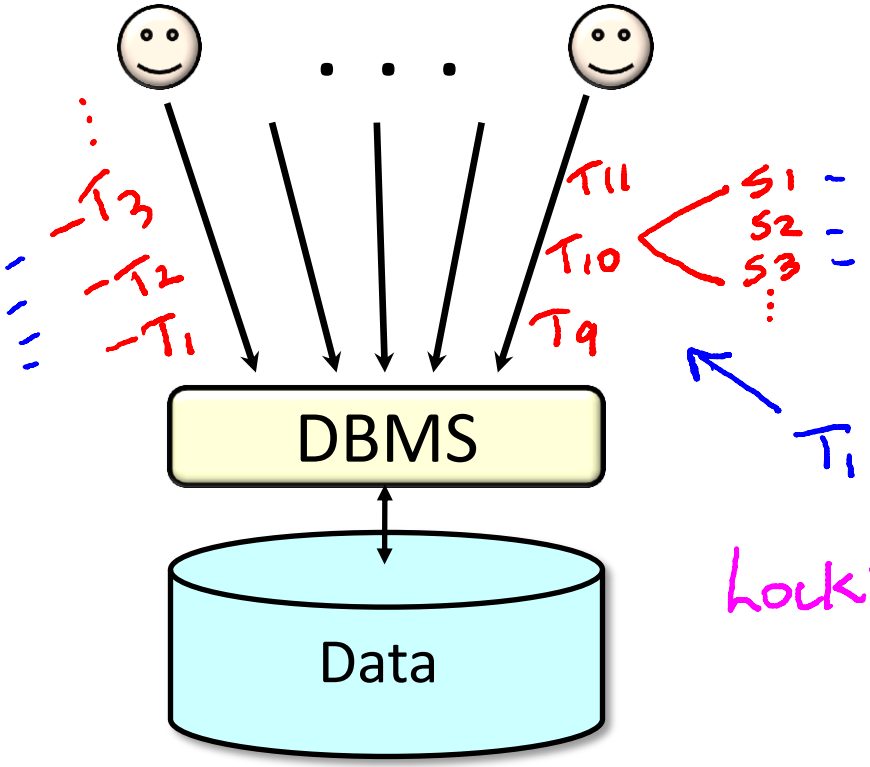
A transaction is a sequence of one or more SQL operations treated as a unit

- Transactions appear to run in isolation
- If the system fails, each transaction's changes are reflected either entirely or not at all

ACID Properties

3. **A**tomicity
4. **C**onsistency
1. **I**solation
2. **D**urability

(ACID Properties) **Isolation**



→ Serializability

Operations may be interleaved, but execution must be equivalent to *some* sequential (serial) order of all transactions

T_1 T_2 T_9 T_{10} T_3 . . .

Locking.

Concurrent Access: Attribute-level Inconsistency

T_1

```
Update College Set enrollment = enrollment + 1000
Where cName = 'Stanford'
```

concurrent with …

T_2

```
Update College Set enrollment = enrollment + 1500
Where cName = 'Stanford'
```

T_1 ; T_2 15,000 → 17,500

T_2 ; T_1

Concurrent Access: Tuple-level Inconsistency

T_1

```
Update Apply Set major='CS' Where sID=123
```

concurrent with …

T_2

```
Update Apply Set decision='Y' Where sID=123
```

$T_1 ; T_2$

$T_2 ; T_1$

both changes

Concurrent Access: Table-level Inconsistency

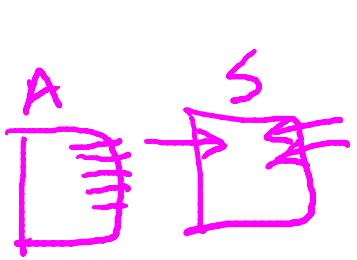
T_1

```
Update Apply Set decision = 'Y'
Where sID In (Select sID From Student Where GPA > 3.9)
```

concurrent with …

T_2

```
Update Student Set GPA = (1.1) * GPA Where sizeHS > 2500
```



$[T_1 ; T_2$ ←
$T_2 ; T_1$ ← order matter

Some

Concurrent Access: Multi-statement inconsistency

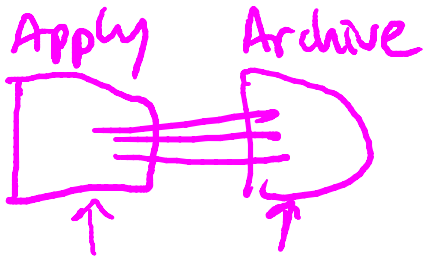
T1

```
Insert Into Archive
  Select * From Apply Where decision='N';
Delete From Apply Where decision='N';
```

concurrent with …

T2

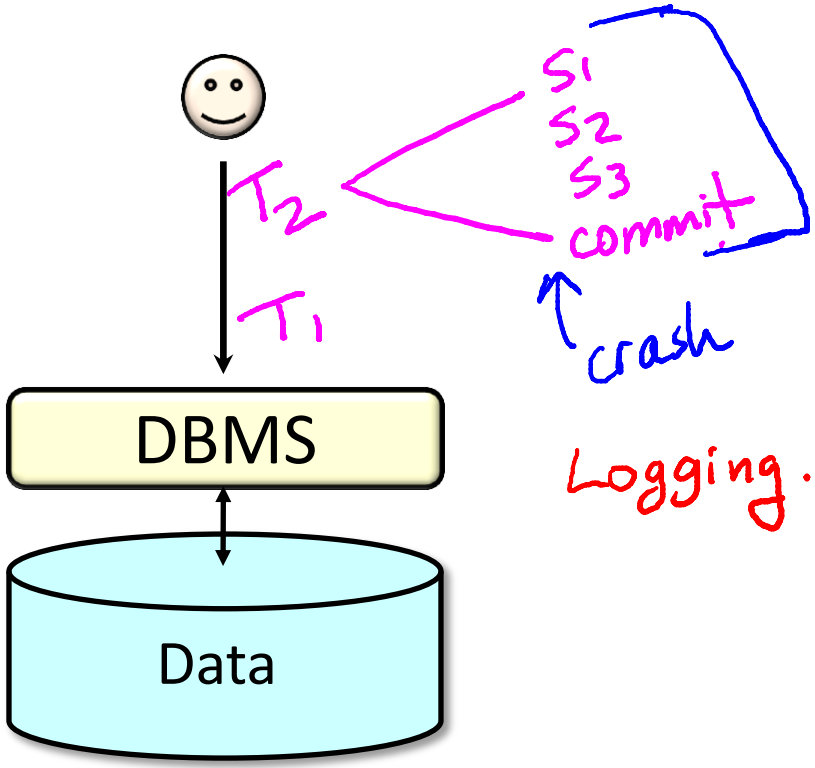
```
Select Count(*) From Apply;
Select Count(*) From Archive;
```



T_1 ; T_2 order

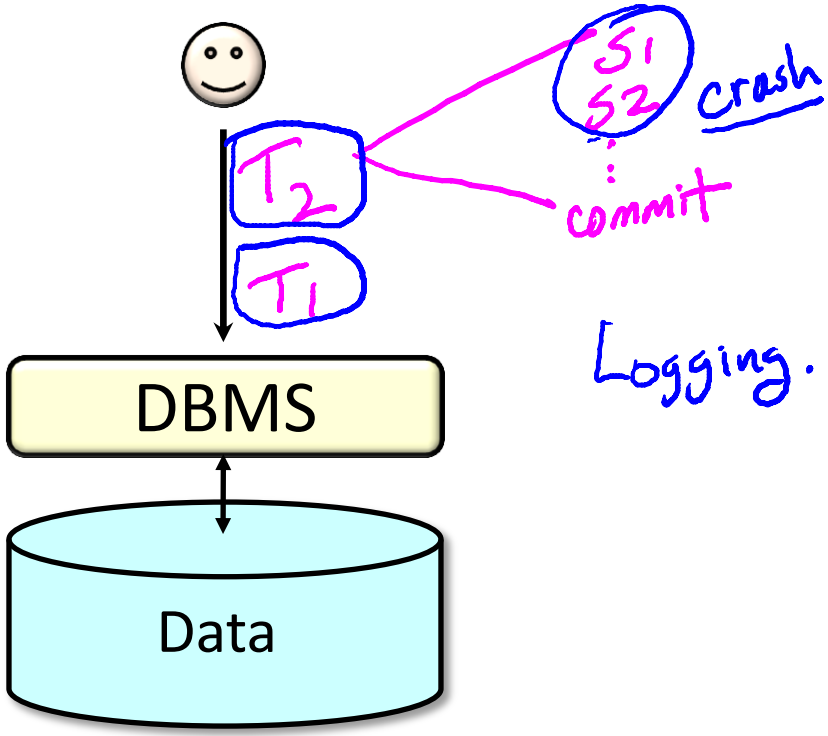
T_2 ; T_1

(ACID Properties) Durability



If system crashes
after transaction commits,
all effects of transaction
remain in database

(ACID Properties) **Atomicity**



Each transaction is "all-or-nothing," never left half done

Transaction Rollback (= Abort)

- Undoes partial effects of transaction
- Can be system- or client-initiated

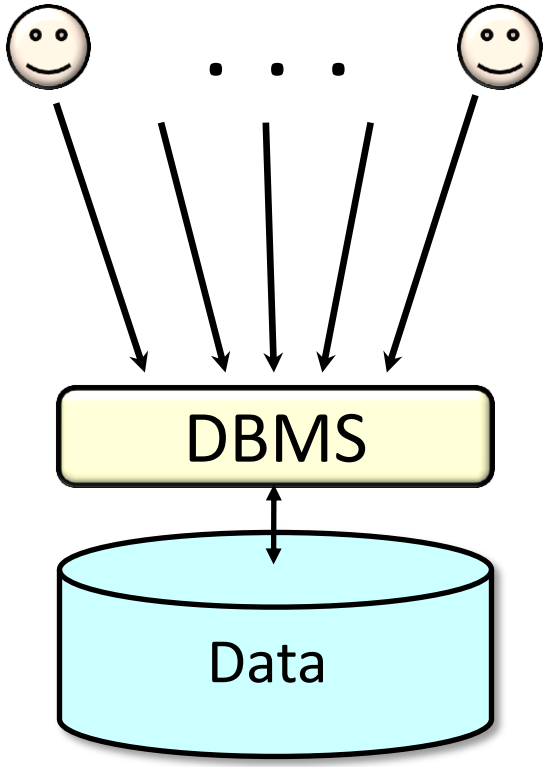
Each transaction is "all-or-nothing," never left half done

```
Begin Transaction;
<get input from user>
SQL commands based on input
<confirm results with user>
If ans='ok' Then Commit; Else Rollback;
```

vars
deliveries
cash

Locking

(ACID Properties) **Consistency**



Each client, each transaction:

- Can assume all constraints hold when transaction begins
- Must guarantee all constraints hold when transaction ends

Serializability ⇒ constraints always hold

Solution for both concurrency and failures

Transactions

- ✓ **A**tomicity
- ✓ **C**onsistency
- ✓ **I**solation
- ✓ **D**urability