

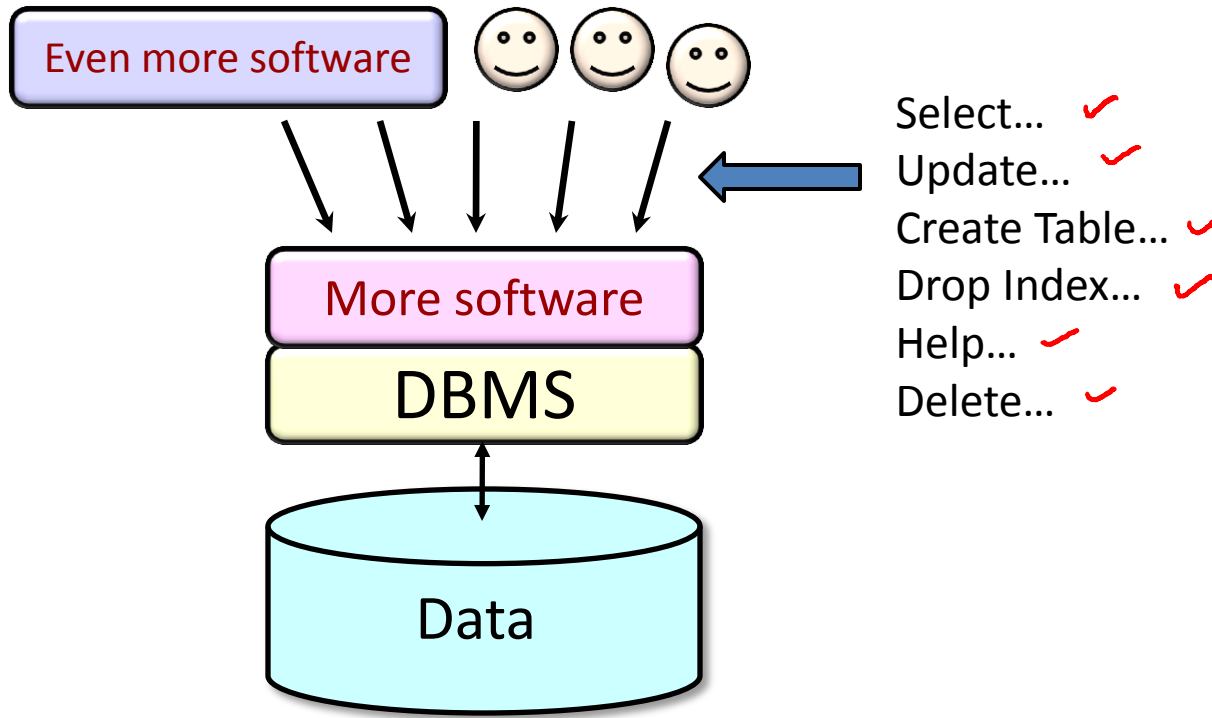
Transactions

Introduction

Motivated by two independent requirements

- Concurrent database access
- Resilience to system failures

Concurrent Database Access

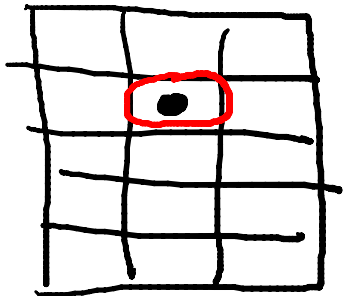


Concurrent Access: Attribute-level Inconsistency

S1 Update college Set enrollment = enrollment + 1000
where cName = 'Stanford'

concurrent with ...

S2 Update college Set enrollment = enrollment + 1500
where cName = 'Stanford'



get ; modify ; put ✓

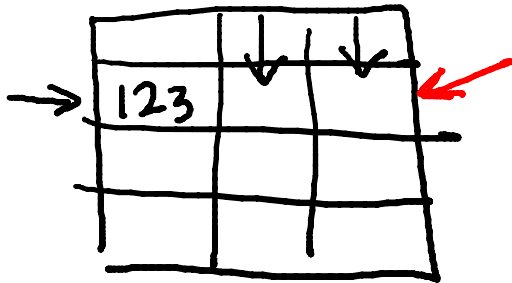
$$\begin{aligned} 15,000 + 2500 &= 17,500 \\ &+ 1000 \\ &+ 1500 \end{aligned}$$

Concurrent Access: Tuple-level Inconsistency

51 Update Apply Set major = 'CS' where SID = 123

concurrent with ...

52 Update Apply Set decision = 'Y' where SID = 123



get; modify; put

both changes

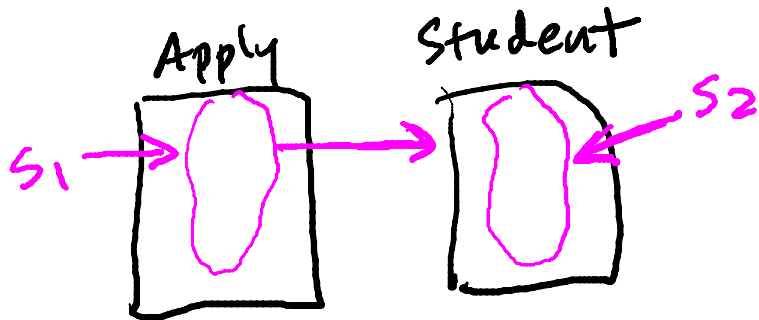
one of the two changes

Concurrent Access: Table-level Inconsistency

S₁ Update Apply Set decision = 'Y'
where SID In (Select SID From Student Where GPA > 3.9)

concurrent with ...

S₂ Update Student Set GPA = (1.1) * GPA Where sizeHS > 2500



Concurrent Access: Multi-statement inconsistency

Insert Into Archive ←

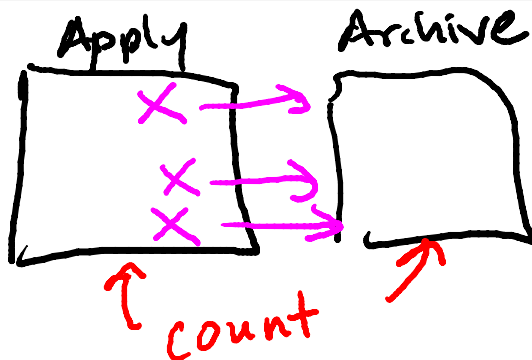
C1 Select * From Apply where decision = 'N';

Delete From Apply where decision = 'N'; ←

concurrent with ...

C2 Select Count(*) From Apply;

Select Count(*) From Archive;



Concurrency Goal

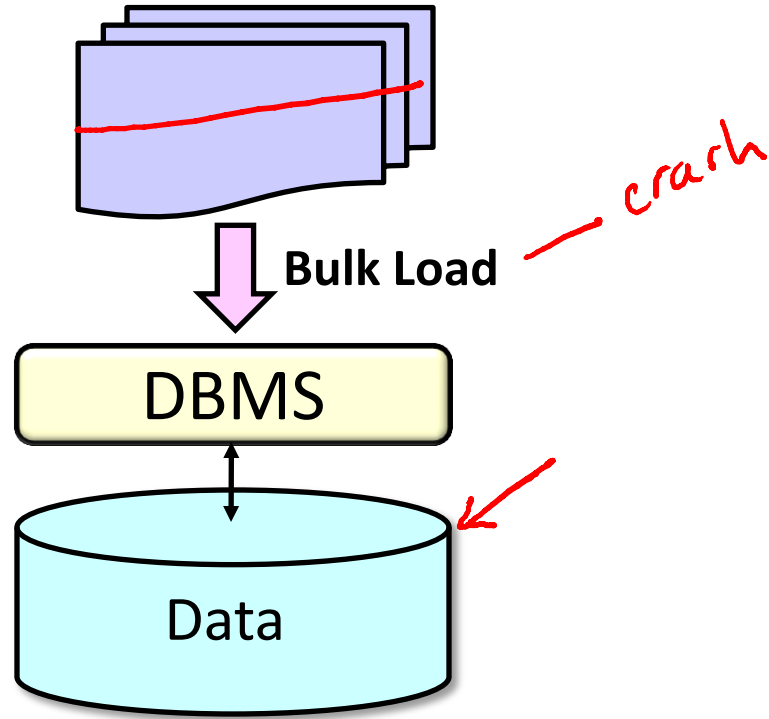
Execute sequence of SQL statements so they appear to be running in isolation

* Simple solution: execute them in isolation

But want to enable concurrency whenever safe to do so

Multiprocessor
Multithreaded
Asynchronous I/O

Resilience to System Failures

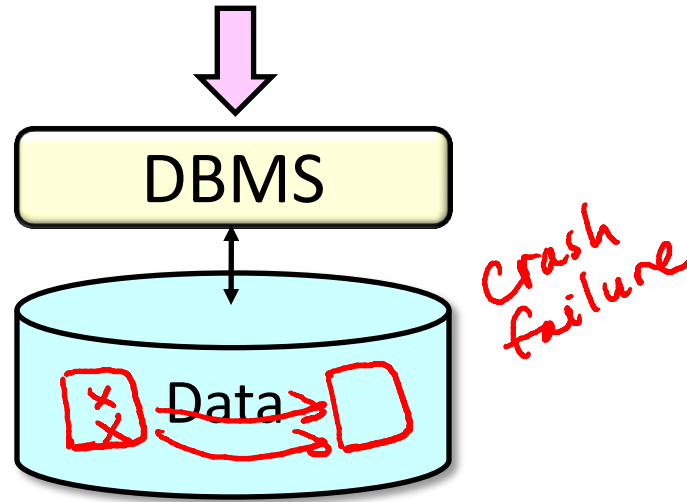


Resilience to System Failures

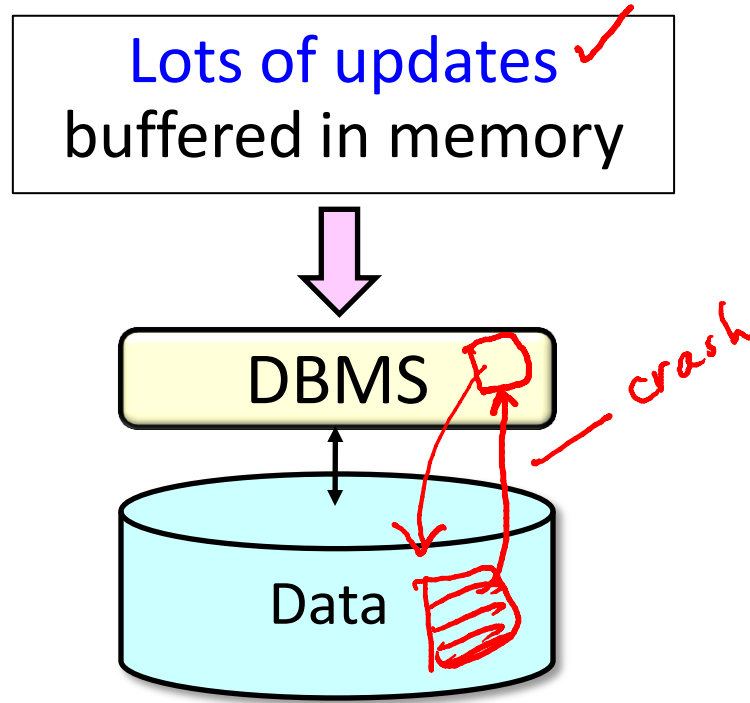
Insert Into **Archive**

Select * From **Apply** where **decision = 'N'** ;

Delete From **Apply** where **decision = 'N'** ;

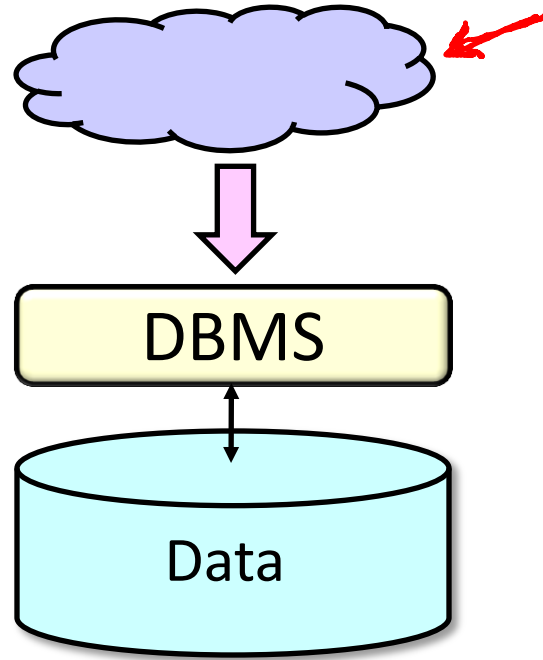


Resilience to System Failures



System-Failure Goal

Guarantee all-or-nothing execution, regardless of failures



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

Solution for both concurrency and failures

Transactions

A transaction is a sequence of one or more SQL operations treated as a unit. **SQL standard:**

- Transaction begins automatically on first SQL statement
- On “**commit**” transaction ends and new one begins
- Current transaction ends on session termination
- “**Autocommit**” turns each statement into transaction

Solution for both concurrency and failures

Transactions

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

