

CA420 Databases II

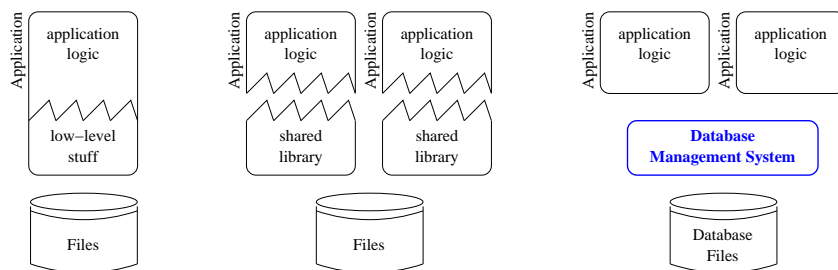
Stephen Blott
 School of Computing
 Dublin City University
 blott@computing.dcu.ie

<http://ca420.computing.dcu.ie/>

Key Database Concepts / Abstractions

data model (relational)
 high-level query language (SQL)
 data independence
 automatic query optimisation
 ⋮
transactional processing

A Brief, Idiosyncratic History of Database Systems



DBMSs provide several key abstractions to support data-management applications

What are they?

Transactions for Programmers

Example 1:

```
BEGIN TRANSACTION
DELETE OrderItems WHERE order_num = 12345
DELETE Orders WHERE order_num = 12345
COMMIT TRANSACTION
```

Example 2:

```
BEGIN TRANSACTION
DELETE OrderItems WHERE order_num = 12345
DELETE Orders WHERE order_num = 12345
ROLLBACK
```

Example 3:

```
BEGIN TRANSACTION
DELETE OrderItems WHERE order_num = 12345
– system failure! –
```

Interleaved Transactions

However, generally transaction execution is interleaved

Why?

Also, some interleavings are obviously incorrect

How can we implement transactional systems that are provably correct, even in the presence of failures?

Concurrency

Concurrent execution includes interleaved executions, parallel executions on multiprocessor systems, and parallel execution across distributed systems

Common problem cases:

Lost updates:

- interleaving causes some updates to be lost
- $r_1[x] \rightarrow r_2[x] \rightarrow w_2[x] \rightarrow w_1[x]$

Dirty reads:

- abort causes read of incorrect data
- $w_1[x] \rightarrow r_2[x] \rightarrow a_1 \rightarrow w_2[y]$

We will also encounter other problem cases as we go along

ACID Transactions

ACID transactions:

Atomicity: transactions are all or nothing:

- either they commit and their effects become permanent,
- or they abort and their effects are removed from the database

(Consistency): in isolation, transactions transform the stable database from one logically-consistent state to another

Isolation: transactions appear to execute in isolation:

- they are isolated from the effects of concurrently-executing transactions

Durability: the effects of committed transactions are permanent

Two mechanisms are provided by DBMSs to ensure that transaction execution is ACID, and hence correct:

- *concurrency control*, and
- *recovery*

Recovery

Database recovery subsystems must deal with many types of failure:

- system crashes
- transaction or system error:
 - e.g. divide by zero, or disk failure
- transaction aborts
- concurrency control enforcement:
 - no way to proceed without violating isolation
- catastrophe (e.g. building burns down)
- . . . and many other reasons

A Simple Model – 1

A database:

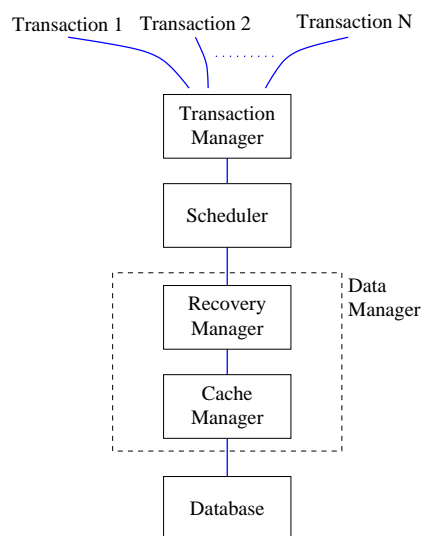
- consists of a set of *distinct items*:
 - denoted x, y, \dots
- supports *transaction operations*:
 - *Start*:
 - * create a new transaction (usually denoted T_1, T_2 , etc.)
 - *Commit*: (usually denoted c_1)
 - * (attempt to) make a transaction's effects permanent
 - *Abort*: (usually denoted a_1)
 - * remove the effects of a transaction from the database
- offers some set of operations on items:
 - usually $Read(x)$, $Write(y)$ (usually denoted $r_1[x]$ and $w_2[y]$)
- transactions are self contained:
 - no messaging between transaction

But this is not just database management systems . . .



. . . it's anywhere data is managed

A Simple System Model – 2



Transaction Manager (TM):

- issue transaction identifiers, coordinate transaction execution, possibly together with other TMs

Scheduler:

- controls order of execution:
 - execute operation immediately
 - reject operation
 - delay operation

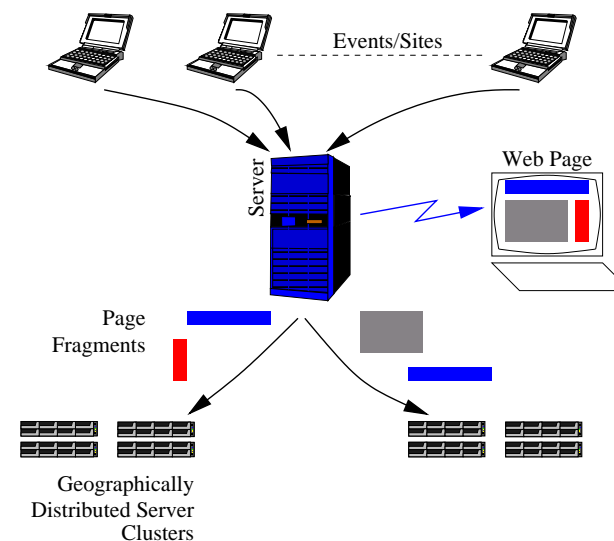
Recovery Manager (RM):

- ensure stable database (upon recovery) contains *all* and *only* the effects of committed transactions, and handle transaction abort

Cache Manager (CM):

- manage data movement between volatile (main) memory and the stable database

Distribution Architecture



Other examples

Transactions are now integral to many software systems:

- workflow systems
 - reliable messaging
 - TP-monitors
 - Corba, SOAP, web services, etc.
-

Textbooks

Concurrency Control and Recovery in Database Systems

Bernstein, Hadzilacos and Goodman

Addison-Wesley

<http://research.microsoft.com/~philbe/ccontrol/>

Transaction Processing: Concepts and Technologies

Jim Gray and Andreas Reuter

Morgan Kaufmann

ISBN: 1-55860-190-2

Databases and Transaction Processing: An Application-Oriented Approach

Lewis, Bernstein and Kifer

Addison-Wesley

ISBN: 0-201-70872-8
