Introduction

Introduction to Database Management

CS348 Spring 2024

Instructor: **Sujaya Maiyya**

Sections: **002 and 003 only**

Outline For Today

1. Overview of DBMSs

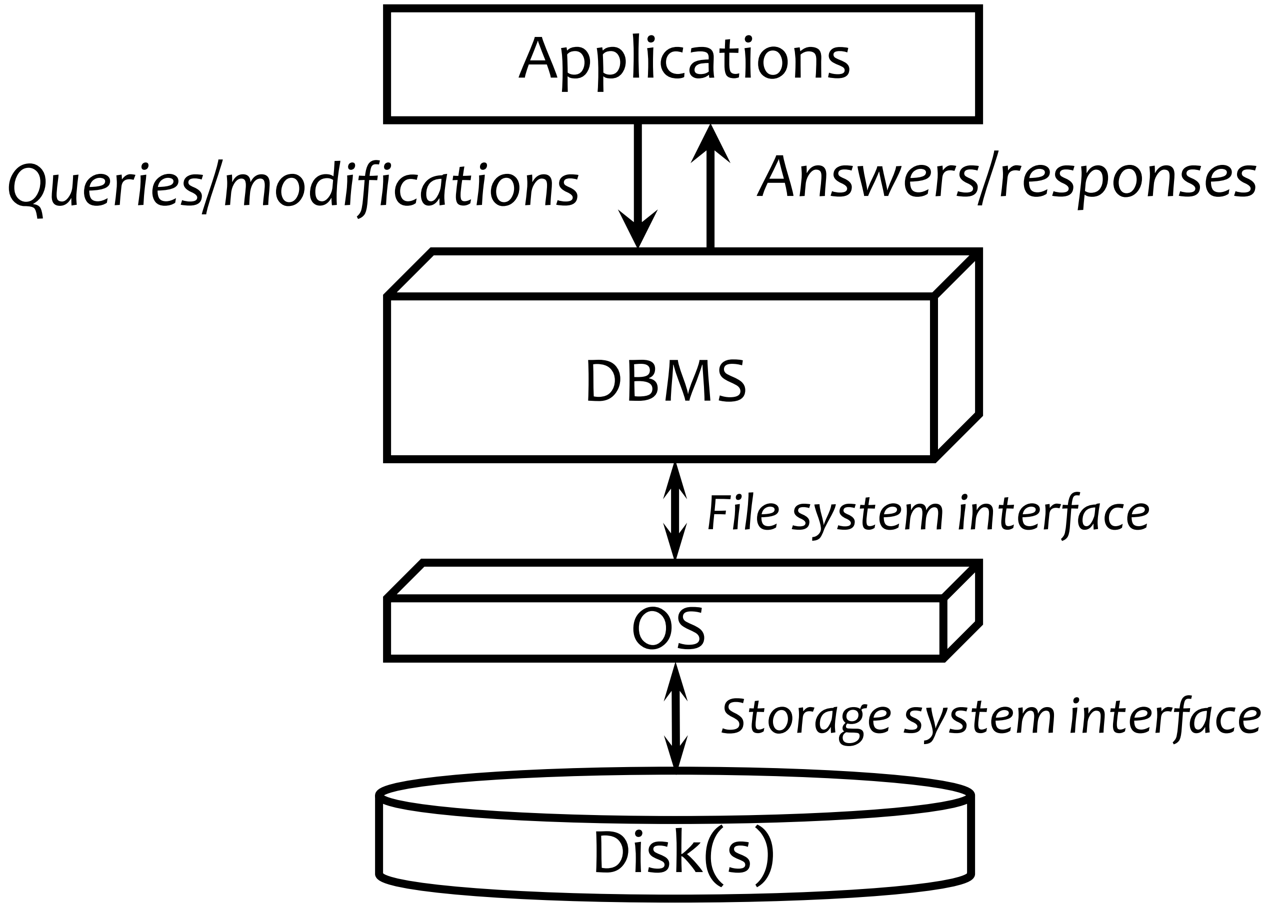
2. Course & Administrative Information

Outline For Today

1. Overview of DBMSs:
    1. Challenges with data management
    2. How DBMSs help overcome these challenges

2. Course & Administrative Information

What is a Database Management System (DBMS)?



Why Do App Developers Need a DBMS?

Challenges to overcome if we do NOT use a DBMS:

1. Physical record design
2. Efficient query algorithms
3. Scalability
4. Data integrity/consistency
5. Concurrent requests
6. Failure & recovery

Why Do App Developers Need a DBMS?

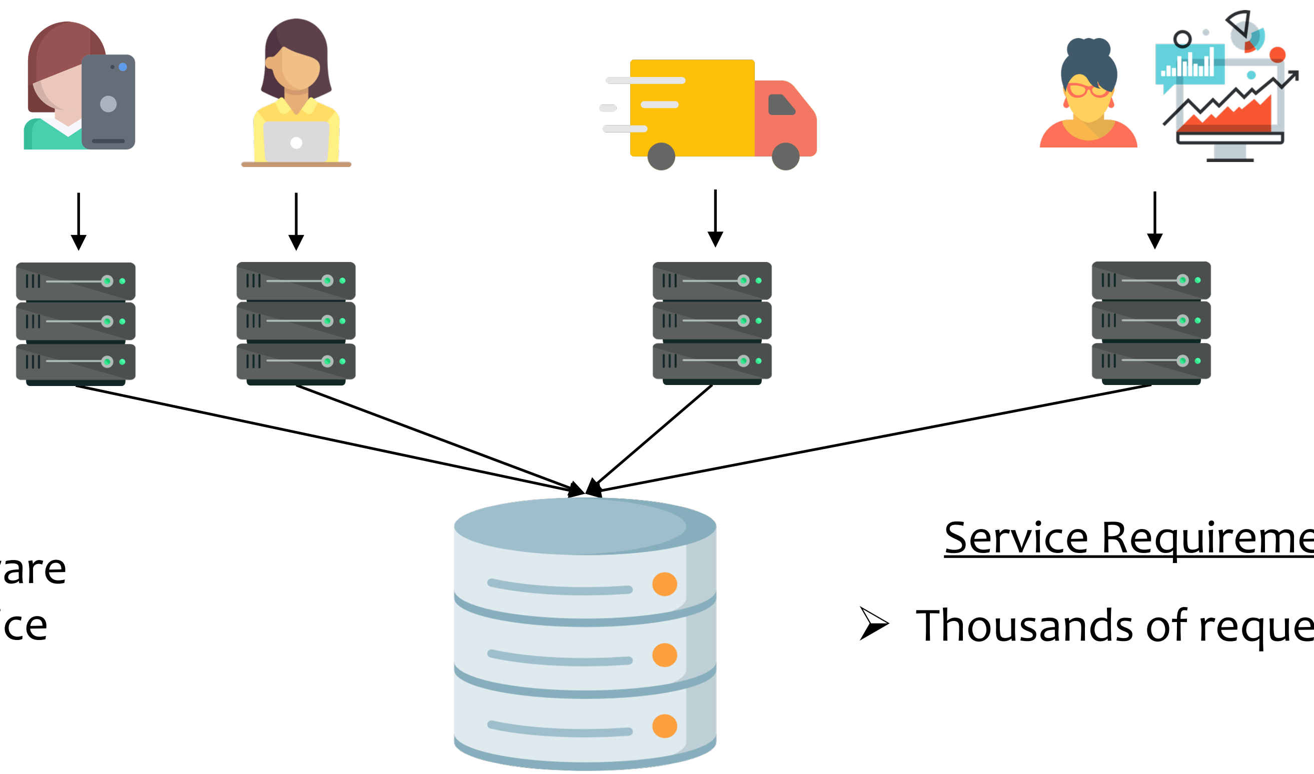
- Application: Order & Inventory Management in E-commerce
  - E.g.: Amazon or Alibaba

Customers & End Devices

Product Shipments & Arrivals

Managers & Analytics Apps

App Software/ Servers



Storage Software Server & Device

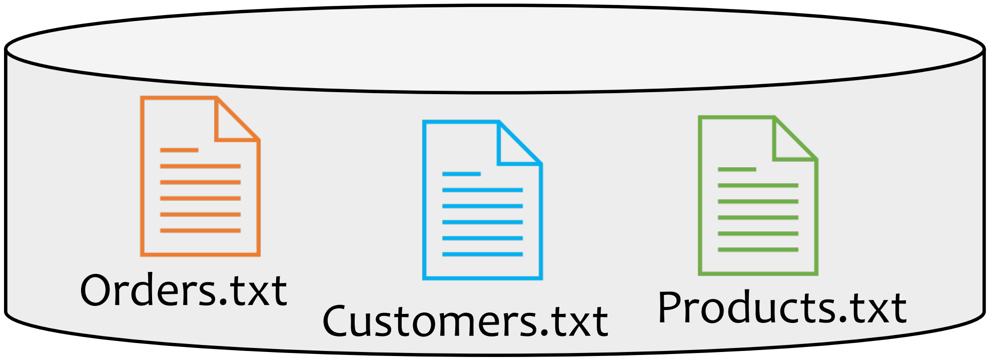
<u>Service Requirement</u>

- Thousands of requests/sec

Let's simplify the design: assume a single server will accept requests from app software to keep track of and serve your records: orders, new products, etc.

Bad Idea: Write Storage Software in Java/C++

- Possible Approach: Directly use the file system of the OS.
  - E.g: one or more files for orders, customers, products etc.



- Problem: Physical Record Design?
  - For each customer store **name, birthdate**
  - How many bytes for each fact?
    - E.g.: Encoding of string names? Fixed or variable length?
  - Many sub-problems: E.g.: How to quickly find a record?

PR1: Example Physical Record Designs (1)

- Variable-length design

name-len (bytes)	name payload	birthdate (fixed 4 bytes)

11	Alice Smith	2001/09/08	19	Alexander Desdemona	2002/05/20
6	Ali Jo	1992/02/25	26	Montgomery Cambridgeshire	1992/02/25
…	…	…	…	…	…
…	…	…			

Customers.txt

- Fixed-length design

Overflow ptr	len	name (16 byte)	birthdate (4 bytes)

null	11	Alice Smith ----------	2001/09/08
0	19	Alexander Desdem	2002/05/20
null	19	Ali Jo ----------------	1992/02/25
1	26	Montgomery Cambr	1992/02/25

Customers.txt

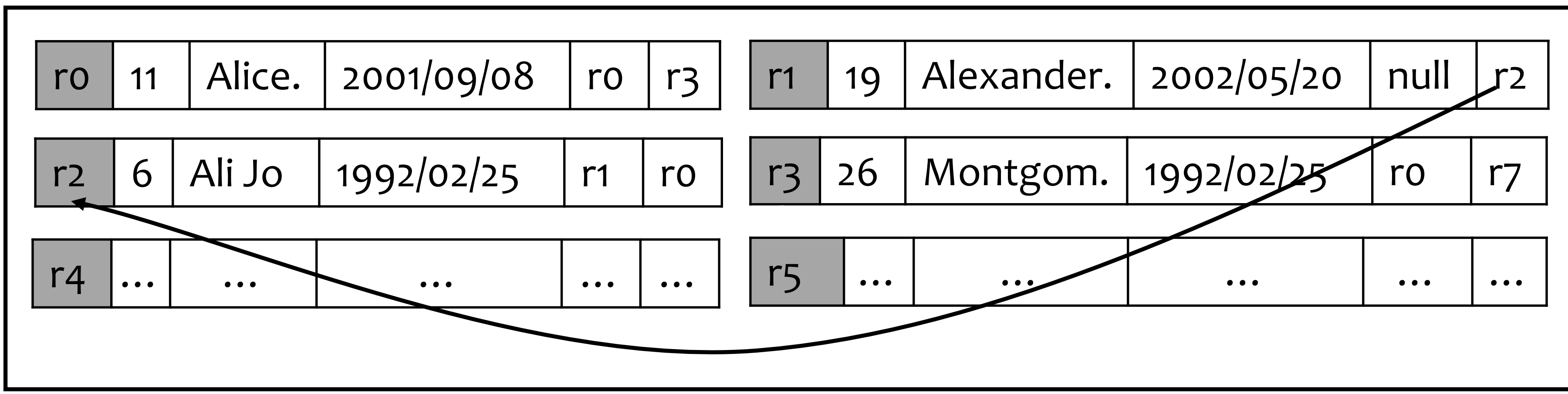
ona	idgeshire	….
…	…	….

Customer-overflow.txt

PR1: Example Physical Record Designs (2)

- Chained Design: Maybe to keep in sorted alphabetical order

name-leng (bytes)	name payload	birthdate (fixes 4 bytes)	prev ptr	next ptr



Customers.txt

Takeaway 1: Many design options & difficult for app developers!

Takeaway 2: Bytes not the right data abstraction to program apps.

PR2: Efficient Query/Analytics Algorithms

- Managers Ask: Who are top paying customers?
  - Task: Compute total sales by customer (assume fixed len records)
  - Problem: App developer needs to implement an algorithm.

Possible Algorithm 1:

```
file = open("Orders.txt")
HashTable ht;
for each line in file:
// some code to parse custID and price
 if (ht.contains(custID))
  ht.put(custID, ht.get(custID) + price)
 else: ht.put(custID, price);
file.close();
```

O1	Cust1	BookA	$20
O2	Cust2	WatchA	$120
O3	Cust1	DiapersB	$30
O4	Cust3	MasksA	$15
…	…	…	…
…	…	…	…

Orders.txt

Should one parallelize this? How?
Do this again if query is repeated?

PR2: Efficient Query/Analytics Algorithms

- That is only for 1 question. There will be many questions:
  - List of Orders that bought a product that cost > $500
  - Last Order from Cust4?
  - Who are closest co-purchasers of Cust4?
  - Many many more (thousands) important business questions:
    - For each question numerous possible algorithms and implementations.

Takeaway 1: Many algs & implementations. Difficult to choose.

Takeaway 2: Writing an algorithm for each task won’t scale!

PR3: Scalability

- Large-scale Data: Data > Memory
  - E.g. Orders.txt grows to terabytes & does not fit in memory.
  - Often the case for data-intensive applications
  - Need disk to scale
  - Hard to write such algorithms. Challenge:
    - *Read in batches? Where to store intermediate results?*

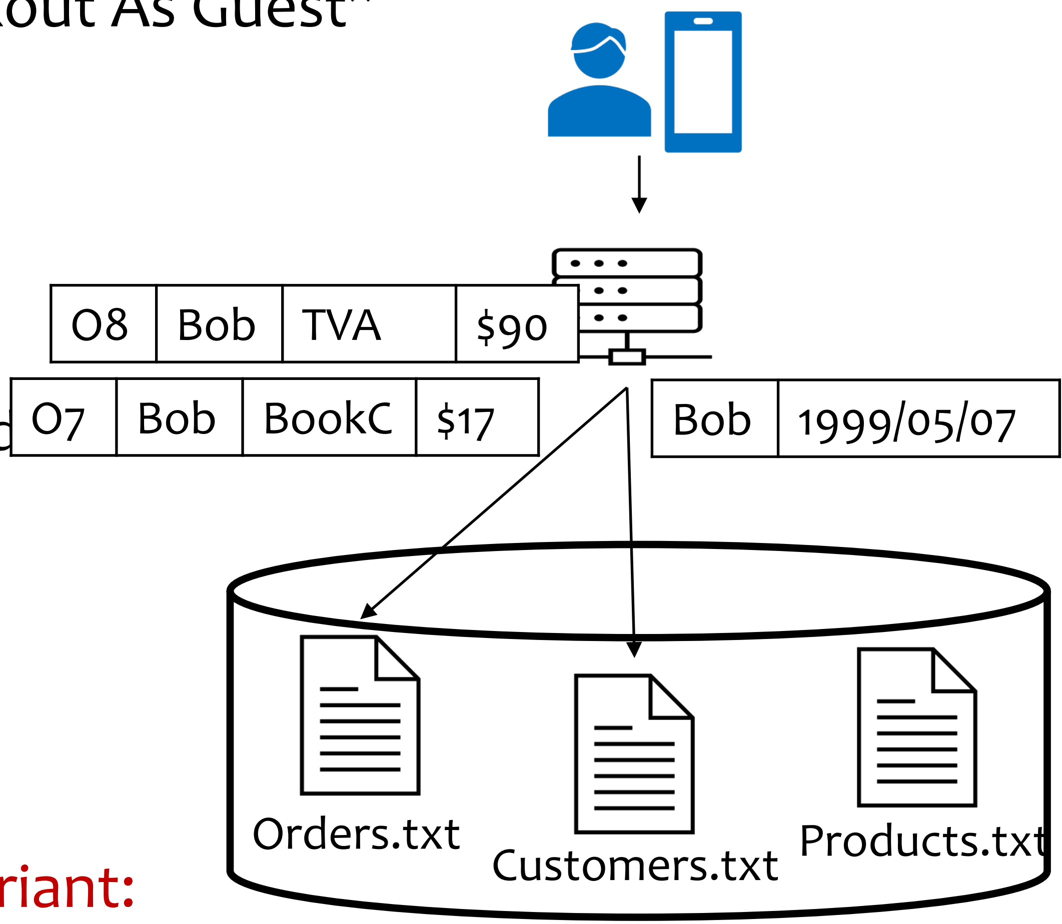
- Scale to: 10K~100Ks of requests/sec
  - Hard to write code that efficiently supports such workloads.

Takeaway: Hard to implement & has nothing to do w/ the app logic!

App developers should focus on the app!

PR4: Integrity/Consistency of The Data (1)

- Many ways data can be corrupted:
  - Often: Wrong application logic or bugs in application
- E.g: Checkout App's "Checkout As Guest"
  - Writes the Order record
  - And the Customer record
  - Assume Bob shops again
  - (Bob, 1999/05/07) is duplicated



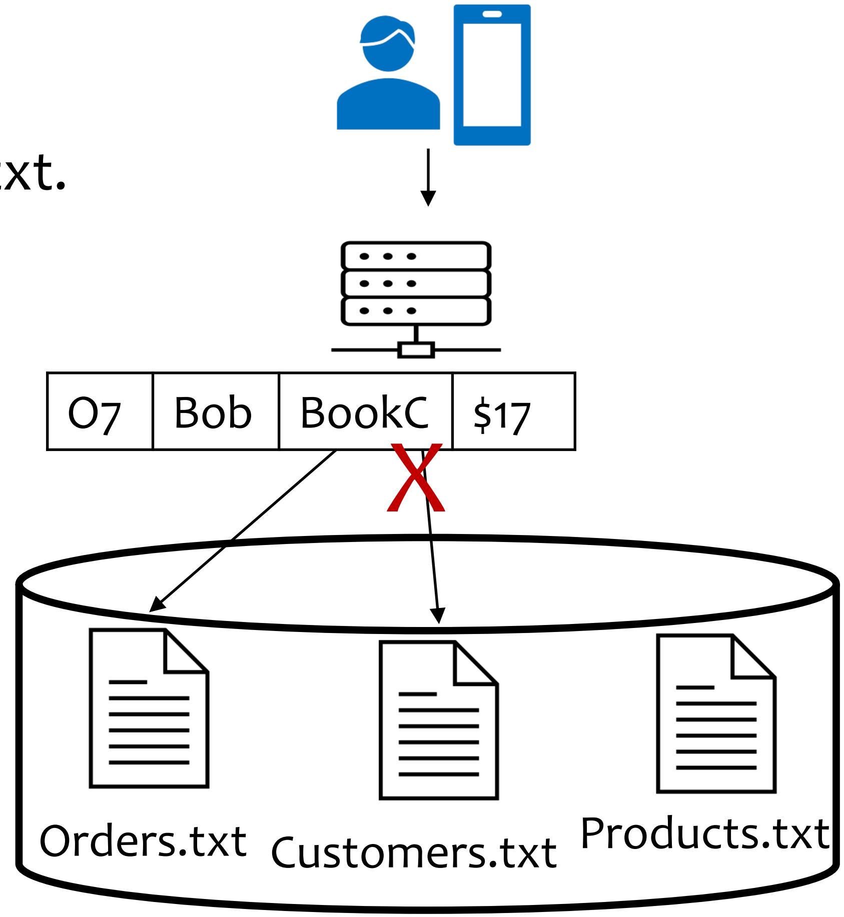
Likely an inconsistency.

We'd want to enforce the invariant:

No duplicate cust records!

PR4: Integrity/Consistency of The Data (2)

- E.g: Checkout App's "Checkout As Guest"
  - Writes the Order record
  - But not the Customer record
  - (Bob, 1999/05/07) is not in Customers.txt.



Likely an inconsistency.
We'd want to enforce the invariant:
Every order's cust record exists!

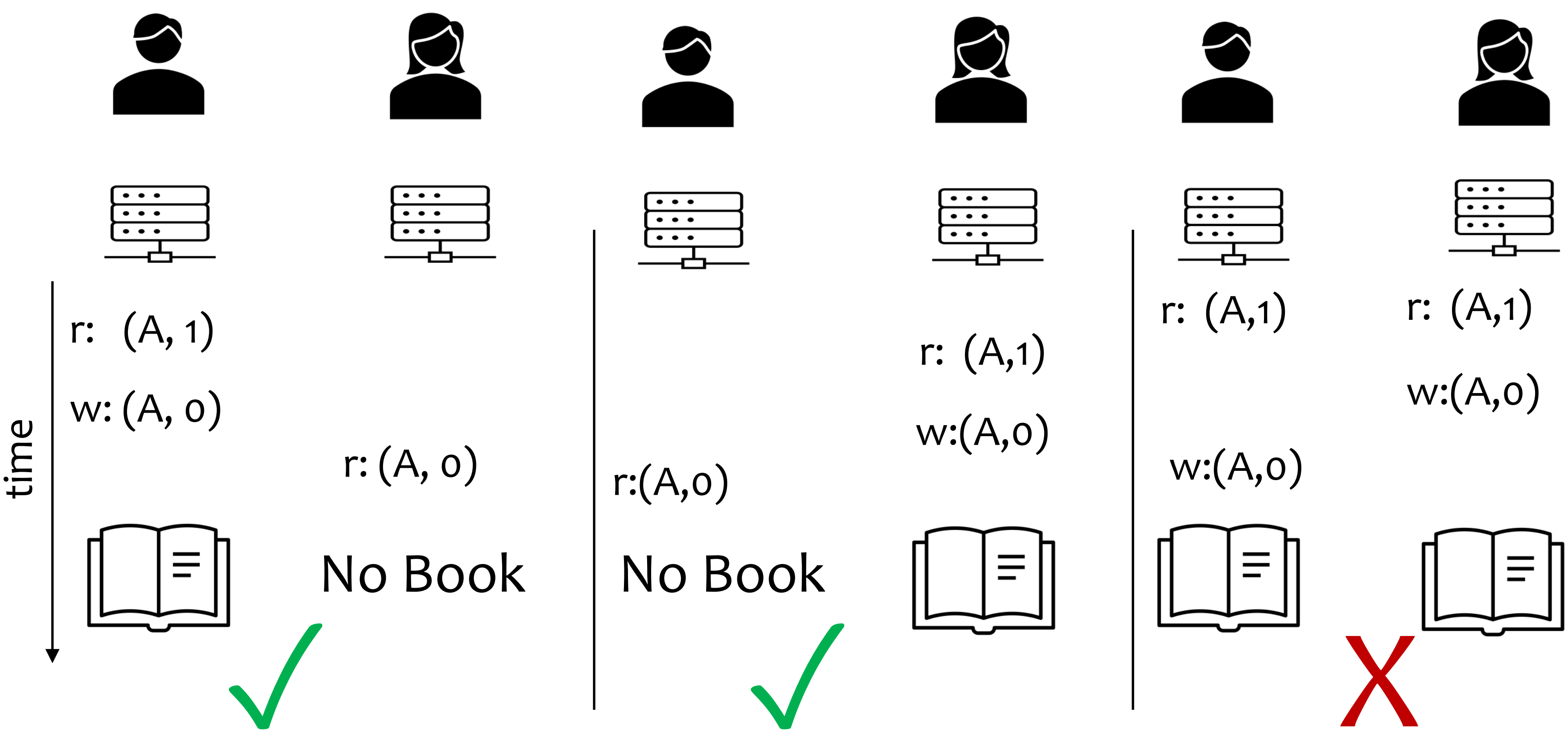
Take away: Incorrectly handling consistency requirements *violates data integrity/consistency!*

PR5: Concurrency: Multiple Conflicting Requests

- Alice & Bob concurrently order BookA: suppose 1 left in stock.

Product	NumInStock
…	…
BookA	1
…	…

```
Buy_Product_Subroutine(string prodName):
(prod, numInStock) = readProduct(prodName)
if (numInStock > 0):
        writeProduct(prod, numInStock - 1)
else throw("Cannot buy product!");
```



Concurrency Questions

- What is a correct/incorrect state upon concurrent updates?
  - Theoretical formalism to explain these states: Serializability
- What protocols/algorithms can ensure a correct state?
  - Locking-based protocols
    - Acquire locks to prevent bad state (Pessimistic protocols)
  - Optimistic protocols
    - Detect bad state and recover from it

Concurrency Avoidance Ex: Global DB Lock

- Alice and Bob order BookA

time ↓

Bob	Alice
lock DB X Wait	lock DB ✓
	r (A, 1)
	w (A, 0)
	release lock
lock DB ✓	

Product	NumInStock
…	…
BookA	1
BookB	7

Safe but inefficient. Why?

Concurrency Avoidance Ex: Global DB Lock

- Alice orders BookA, Bob orders BookB

Bob	Alice
lock DB X Wait	lock DB ✓
	r (A, 1)
	w (A, 0)
	release lock
lock DB ✓	
r (B, 7)	
…	

time

Product	NumInStock
…	…
BookA	1
BookB	7

Bob had no conflicts; so was "unnecessarily" blocked.

Concurrency Avoidance Ex: Record-level Lock

- Alice, Bob as before want BookA, Carmen orders Book B

Bob	Alice	Carmen
lock: (A) ✓	lock: (A) X Wait	lock: (B) ✓
r (A, 1)		r (B, 7)
w (A, 0)		w (B, 6)
	...	

time

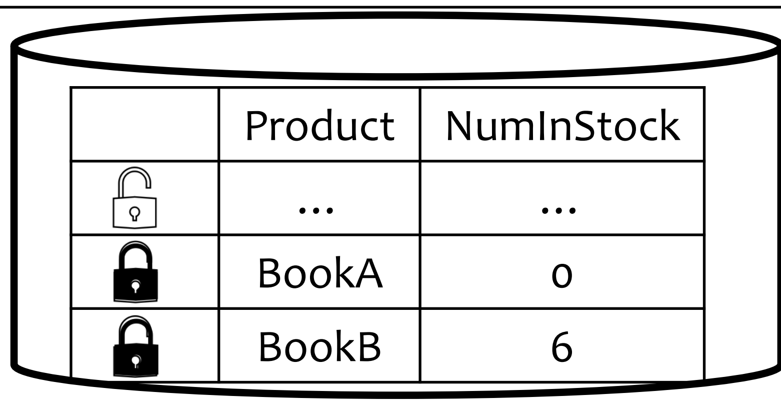
	Product	NumInStock
	...	...
	BookA	1
	BookB	7

Concurrency Avoidance Ex: Record-level Lock

- Alice, Bob as before want BookA, Carmen orders Book B

time ↓

Bob	Alice	Carmen
lock: (A) ✓	lock: (A) X Wait	lock: (B) ✓
r (A, 1)		r (B, 7)
w (A, 0)		w (B, 6)
release lock A		release lock B
	lock: (A) ✓	
…		

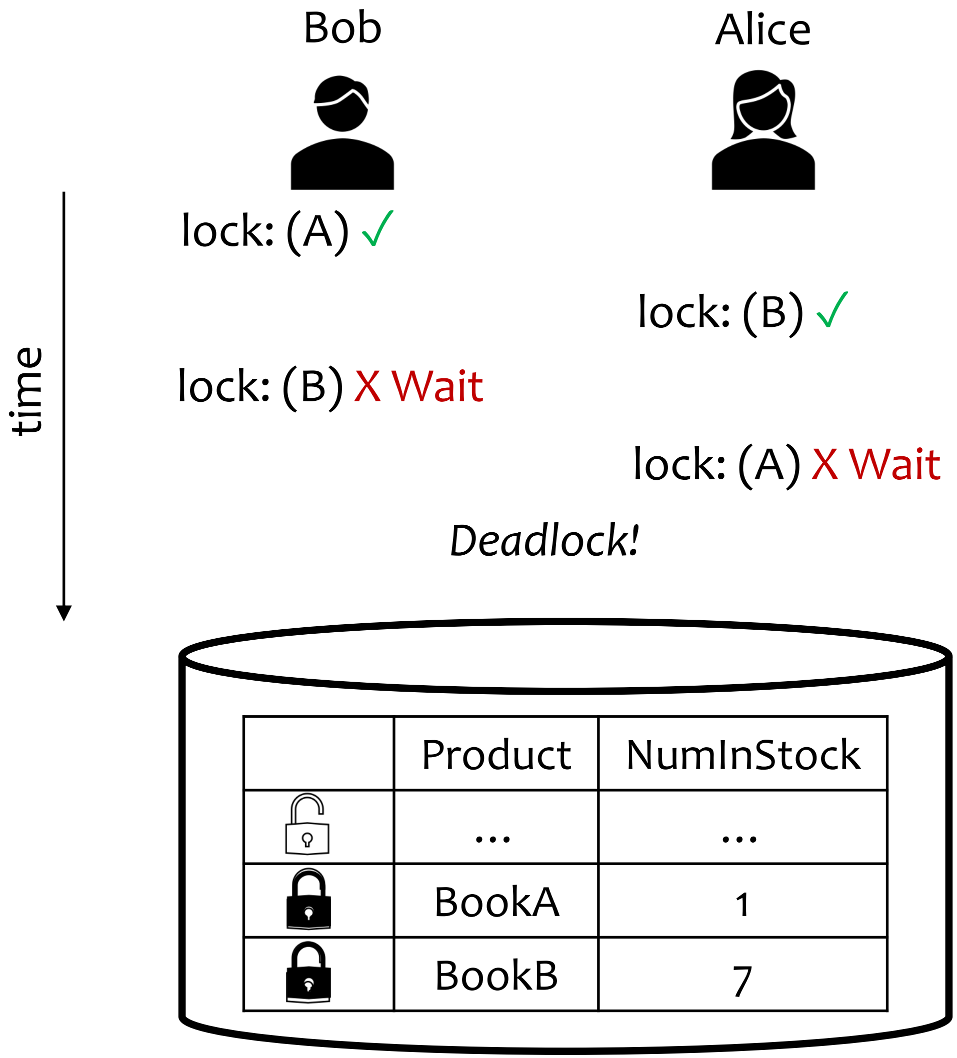


	Product	NumInStock
(unlocked)	…	…
(locked)	BookA	0
(locked)	BookB	6

Safe and achieves parallelism. What can go wrong?

Where There is Locking, There is Deadlocks!

- Alice, Bob both order both BookA and BookB together

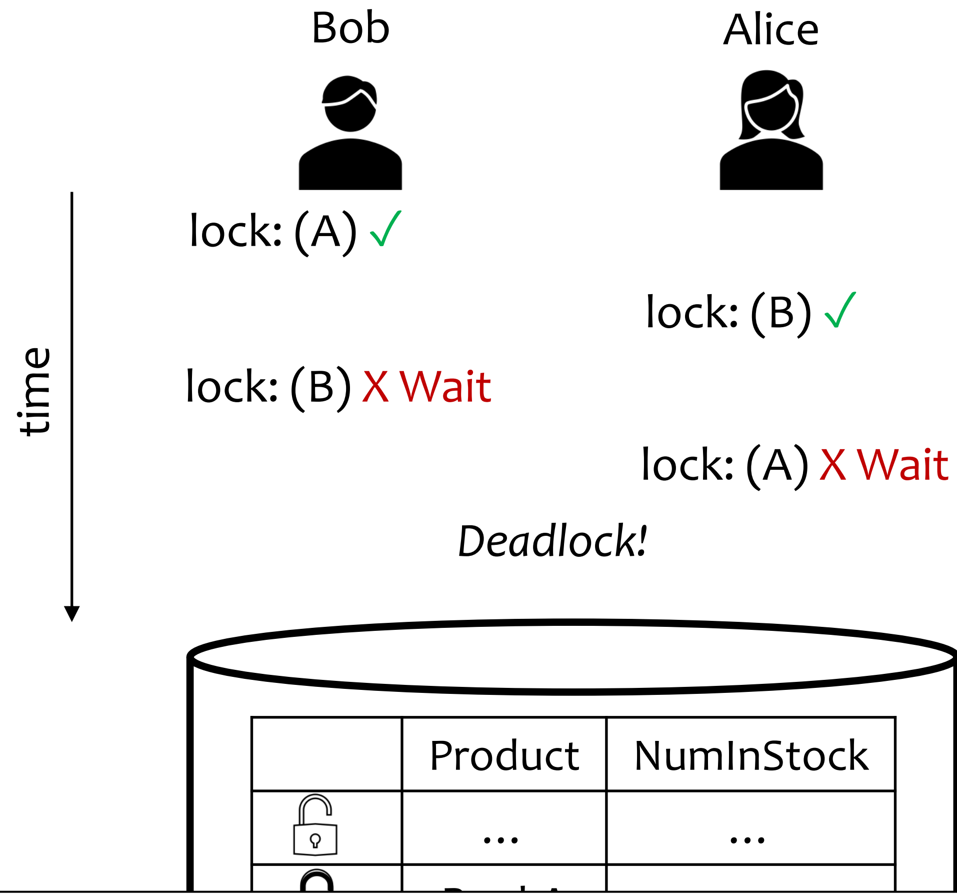


	Product	NumInStock
	…	…
	BookA	1
	BookB	7

How can we detect & avoid deadlocks?

Where There is Locking, There is Deadlocks!

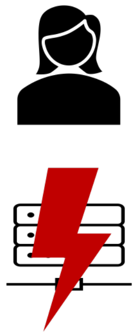
- Alice, Bob both order both BookA and BookB together



Take away: Handling concurrent requests is one of the biggest challenges in data management!

PR6: Failure & Recovery

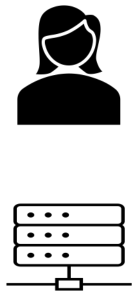
- What if your disk fails in the middle of an order?
- What if your server software fails due to a bug?
- What if there is a power outage in the machine storing files?



Product	NumInStock
…	…
BookA	1
BookB	7

Failure & Recovery

- What if your disk fails in the middle of an order?
- What if your server software fails due to a bug?
- What if there is a power outage in the machine storing files?
- Suppose Alice orders both BookA and BookB



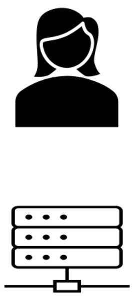
w (A, 0)

w (B, 6)

Product	NumInStock
…	…
BookA	1
BookB	7

Failure & Recovery

- What if your disk fails in the middle of an order?
- What if your server software fails due to a bug?
- What if there is a power outage in the machine storing files?
- Suppose Alice orders both BookA and BookB



w (A, 0)

w (B, 6)

Before (B, 6) is written failure!
Inconsistent data state!

Take away: How to recover from inconsistent state?

Product	NumInStock
…	…
BookA	0
BookB	7

X

✓

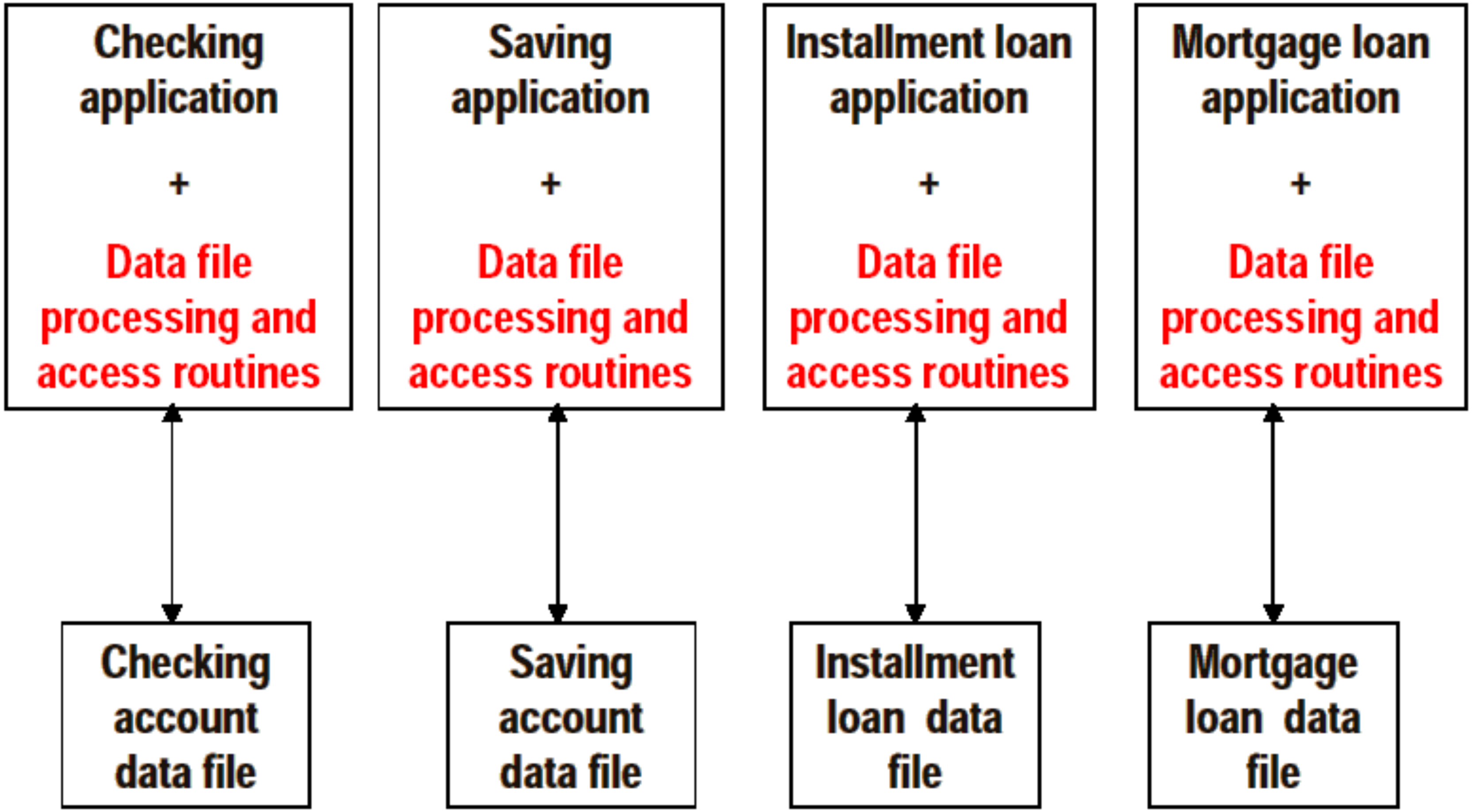
Product	NumInStock
…	…
BookA	0
BookB	6

Summary of challenges

1. Physical record design
2. Efficient query algorithms
3. Scalability
4. Data integrity/consistency
5. Concurrent requests
6. Failure & recovery

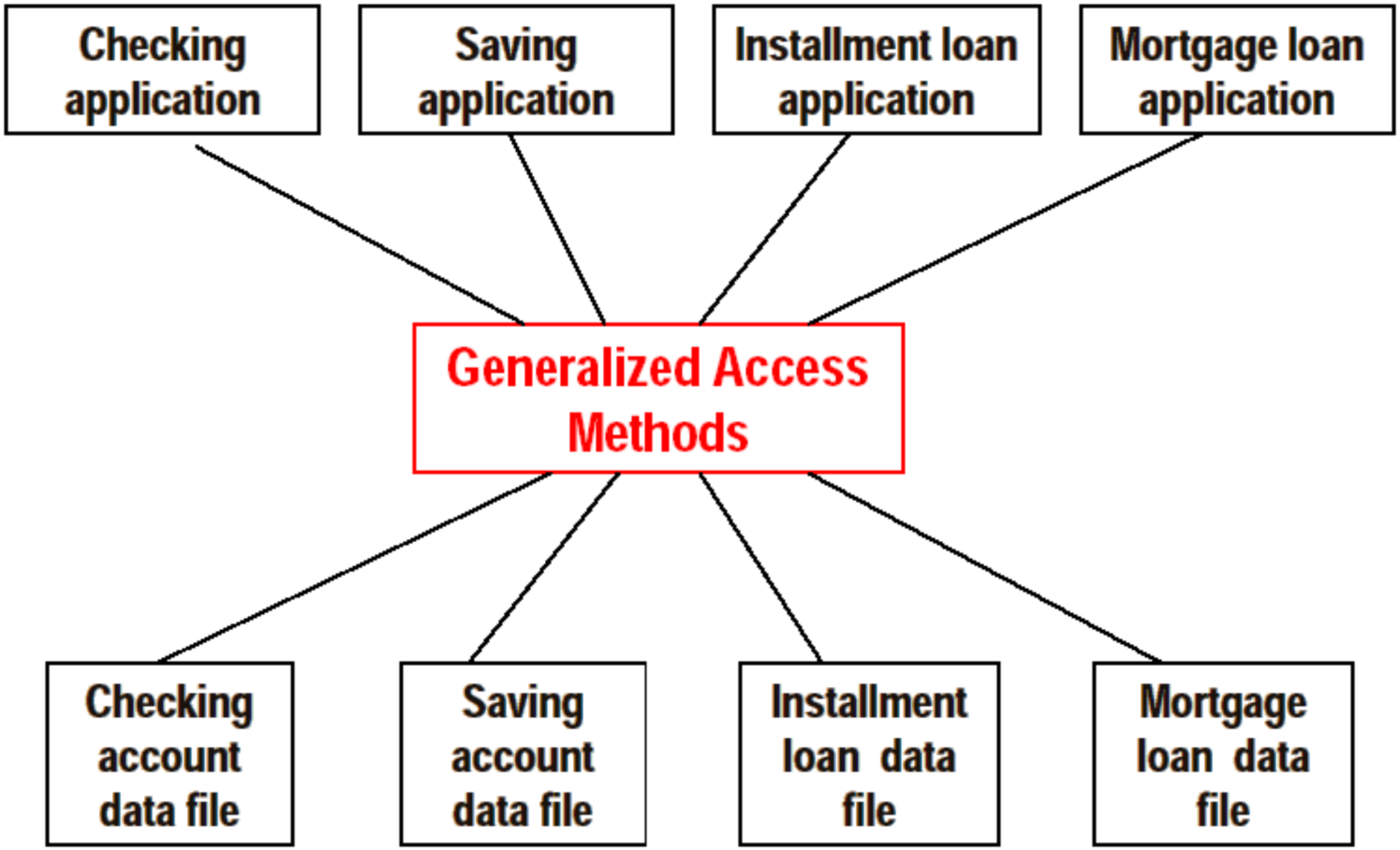
A database management system (DBMS) helps us solve all the discussed problems

The birth of DBMS – 1st gen



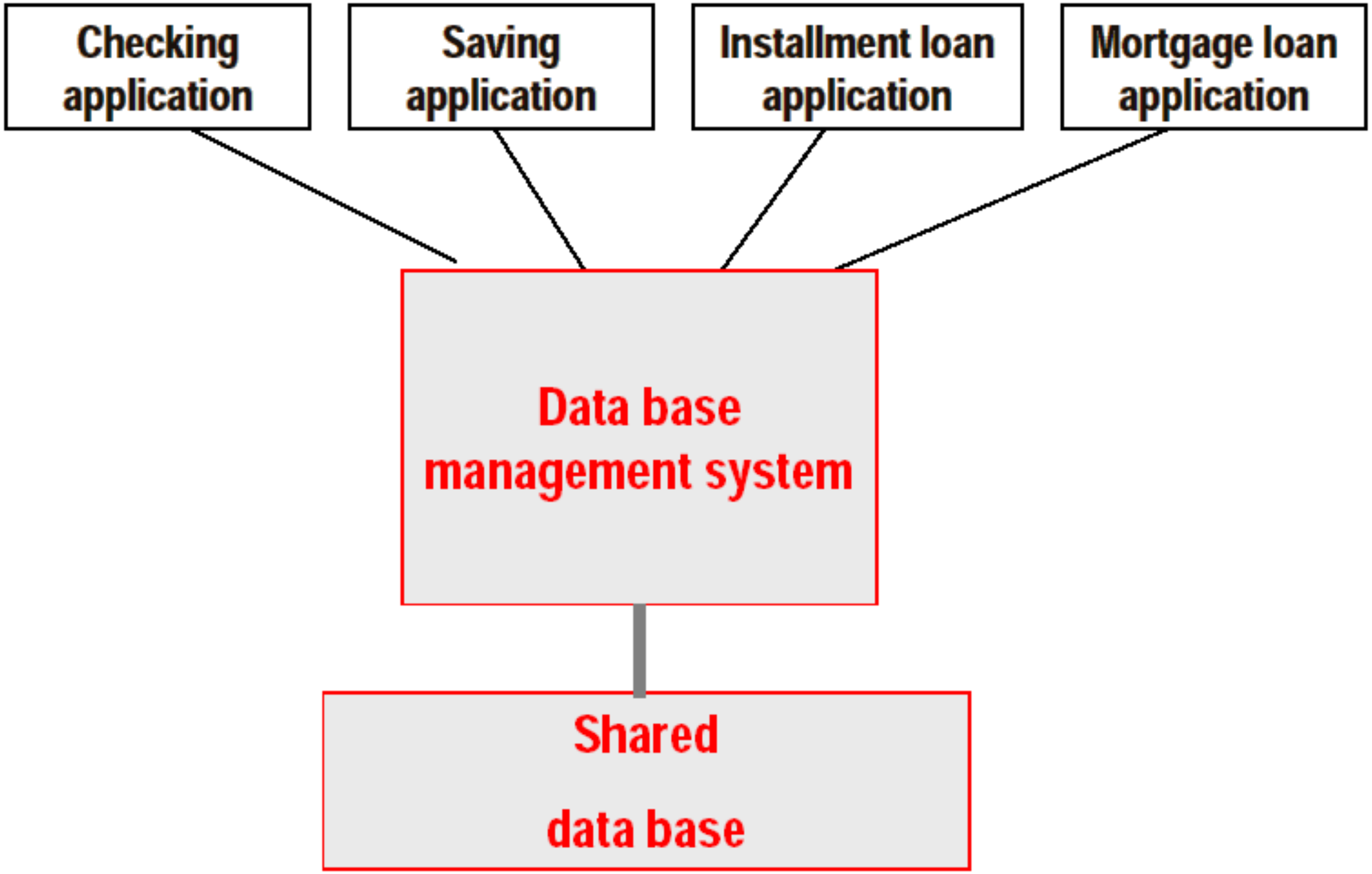
From Hans-J. Schek's *VLDB* 2000 slides

The birth of DBMS – 2nd gen



From Hans-J. Schek's *VLDB* 2000 slides

The birth of DBMS – 3rd gen



From Hans-J. Schek's *VLDB* 2000 slides

Application Development with a DBMS

- Consider the same inventory management application
- We will use a Relational DBMS (RDBMS) but can use other DBMSs too (*e.g., a graph database management system*)
  - Ex: PostgreSQL, Oracle, MySQL, SAP HANA, Snowflake…

1. Data Modeling With an RDBMS (1)

- Relational Model: Data is modeled as a set of tables
  - Much higher-level abstraction than bits/bytes

Customers	
name	birthday
Alice	2001/09/08
Bob	2002/05/20
…	…

Orders			
oID	cust	product	price
O1	2001/09/08	BookA	20
O2	2002/05/20	TVB	100
…	…	…	…

Products	
product	numInStock
BookA	1
TVB	78
…	…

Example SQL Command in an RDBMS:

```
CREATE TABLE Customers
    name varchar(255),
    birthdate DATE;
```

- The RDBMS takes care of physical record design: Fixed-length/var-length, columnar, row, chained etc.
- **The developer need not know the physical record design.**

1. Data Modeling With an RDBMS (2)

- Physical Data Independence:
  - Throughout the lifetime of the app, the RDBMS can change the physical layout for performance or other reasons and the applications is oblivious to this and continues working as-is.
- E.g:
  - A new column can be added that changes the record design
  - A compressed column can be uncompressed

Takeaway: A high-level data model delegates the responsibility of physical record design and access to these records to the DBMS

2. High-level Query Language (1)

- Structured Query Language (SQL)
- SQL is so high-level that it's called a *declarative* language: i.e., one in which you can describe the output of the computation ***but not how to perform the computation***
- Recall managers' question: Who are top paying customers?

```
SELECT cust, sum(price) as sumPay
FROM Orders
ORDER BY sumPay DESC
```

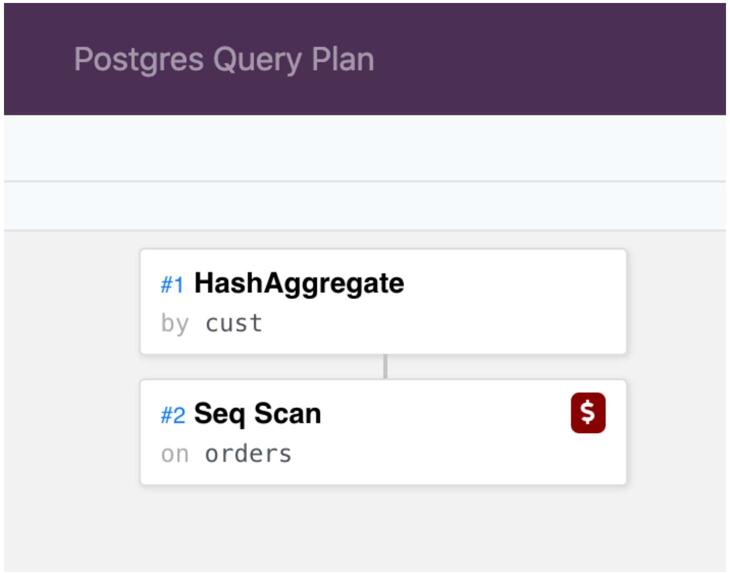
Orders			
oID	cust	product	price

- No procedural description of how to execute the query: hash-based, sort-based, what sorting algorithm to use etc.

2. High-level Query Language (2)

- RDBMS automatically generates an algorithm for the query:
  - We call those algorithms *query plans*

```
SELECT cust, sum(price) as sumPay
FROM Orders
ORDER BY sumPay DESC
```

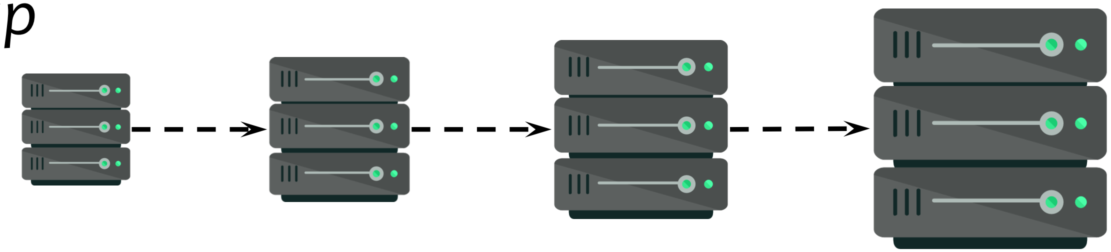


Takeaway: A high-level QL delegates the responsibility of finding an efficient algorithm for queries to the DBMS.

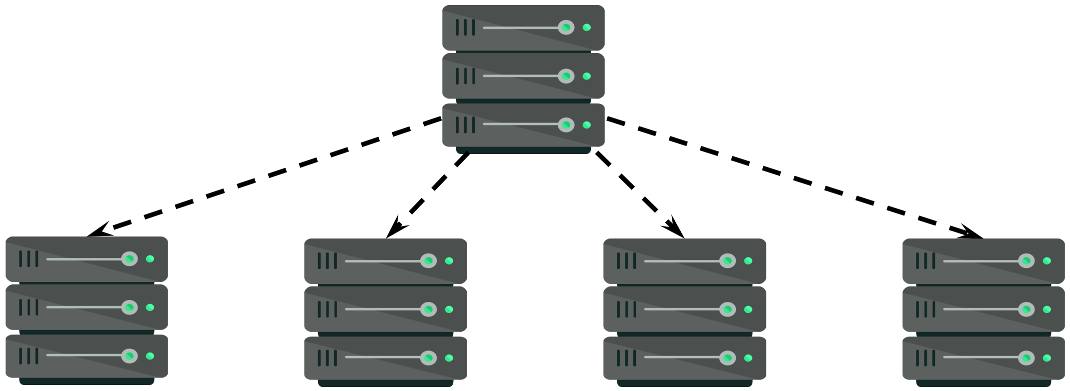
Other efficiency benefits: The DBMS will handle large data and automatically parallelize these algorithms.

3. Scalability

- Two types:
  - *Scale up*



  - *Scale out*



Takeaway: RDBMSs typically support scale out and perform scaling automatically.
App developer need not focus on scalability.

4. Integrity Constraints

- Recall the bug in Checkout App's "Checkout As Guest":
  - Writes the Customer record
  - Assume Bob shops again
  - (Bob, 1999/05/07) is duplicated!
- In RDBMSs: add uniqueness constraints (Primary Key Constraints)

CREATE TABLE Customers (name varchar(255), birthdate DATE, **PRIMARY KEY (name)**);

```
template1=# INSERT INTO Customers Values ('Bob', '1999/05/07');
INSERT 0 1
template1=# INSERT INTO Customers Values ('Bob', '1999/05/07');
ERROR:  duplicate key value violates unique constraint "customers_pkey"
DETAIL:  Key (name)=(Bob) already exists.
```

Takeaway: DBMSs will enforce the constraint and maintain the data's integrity at all times on behalf of the app!

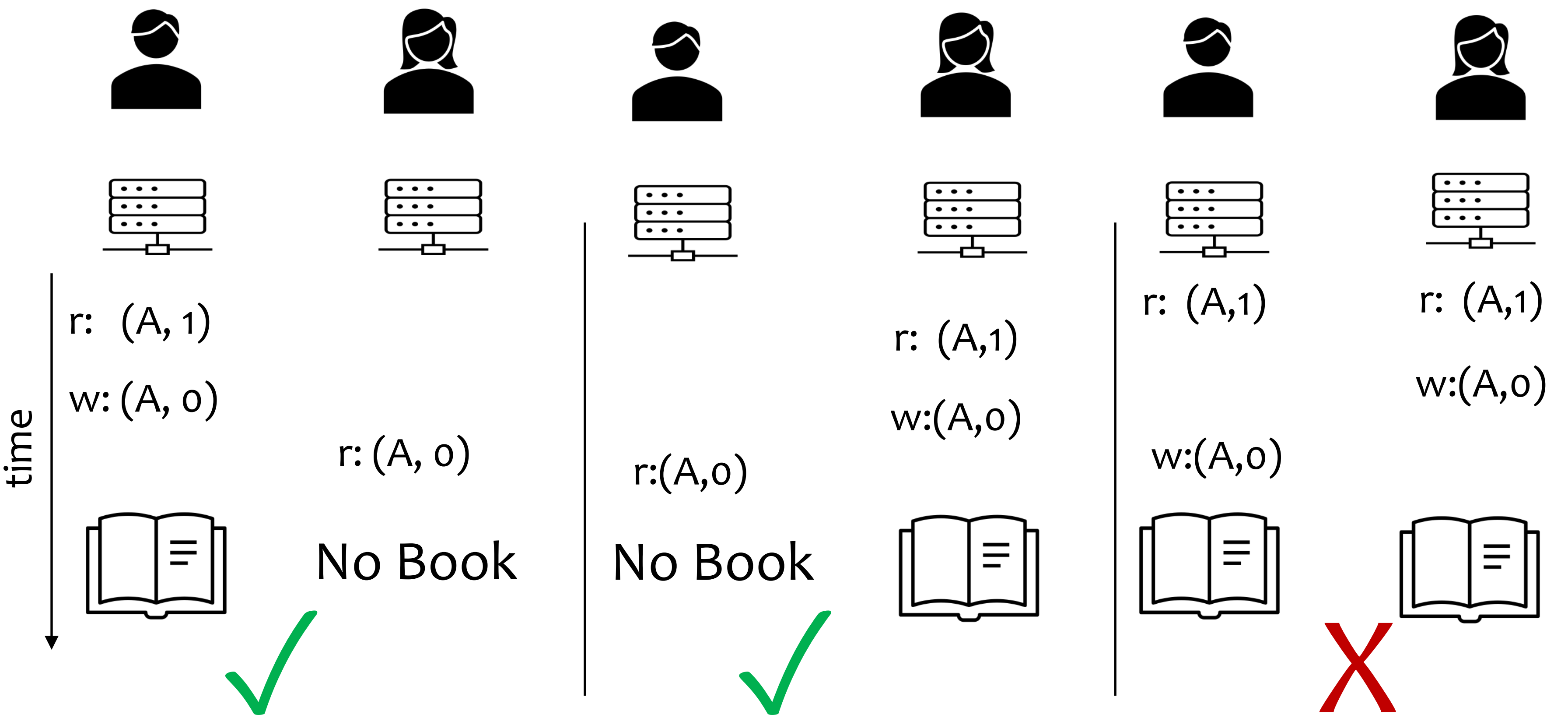
- Can enforce other integrity constraints (e.g., foreign key)

5. Concurrency When Using an RDBMS (1)

- Recall Alice & Bob concurrently ordering BookA:

Product	NumInStock
…	…
BookA	1
…	…

```
Buy_Product_Subroutine(string prodName):
(prod, numInStock) = readProduct(prodName)
if (numInStock > 0):
        writeProduct((prod, numInStock - 1)
else throw("Cannot buy product!");
```



5. Concurrency When Using an RDBMS (2)

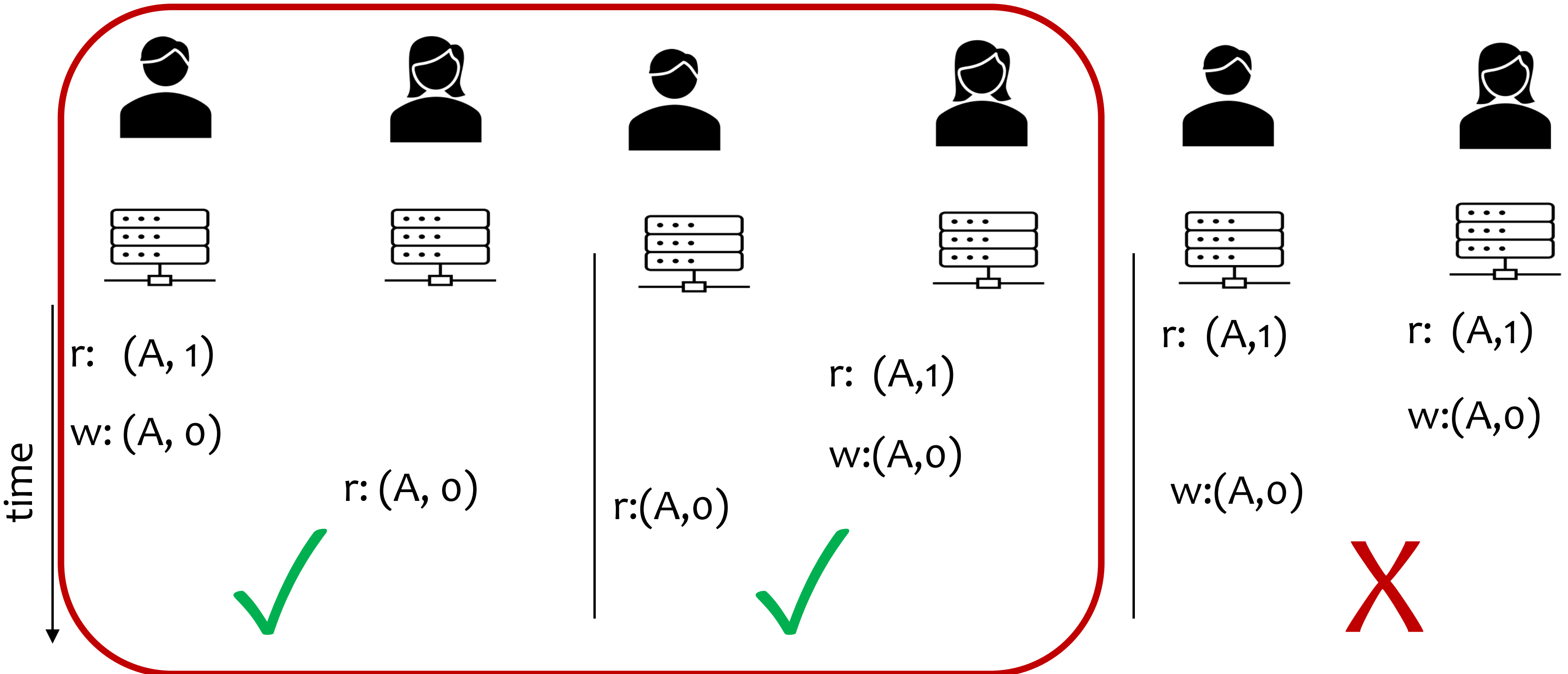
(Simplified) SQL:

```
BEGIN TRANSACTION
UPDATE Products
SET numInStock = numInStock - 1
WHERE name = "BookA"

INSERT INTO Orders
VALUES ("Alice", "BookA", $20)
COMMIT
```

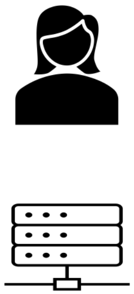
- Will ensure a correct end state
- Will avoid any deadlocks
- Will error for Alice or Bob

Take away: DBMS ensures safe concurrency.



6. Backup and Recovery

- Recall failure scenario: Alice orders both BookA and BookB
- Suppose a power failure occurs and the DBMS fails in the middle of committing the transaction



DBMSs use checkpointing and logging to undo partial changes and
revert back to a consistent state

Take away: DBMSs handle failure recovery

Product	NumInStock
…	…
BookA	0
BookB	7

X

Product	NumInStock
…	…
BookA	1
BookB	7

✓

Summary

DBMS is an indispensable core system software to develop any application that stores, queries, or processes data.

A Glimpse of Current DBMS Market

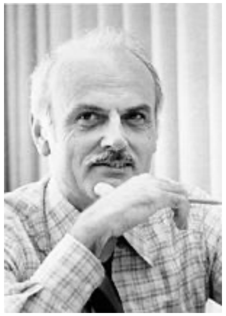
Hundreds of companies producing DBMSs: Many RDBMS/SQL, but also graph, RDF, Document DB, Key-value stores etc..
Not even including companies to tune, ingest, visualize etc..

4 Turing Award Winners!

- Charles Bachman, 1973

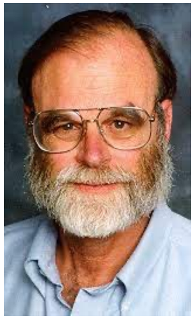
Introduced DB Systems

- Edgar F. Codd, 1981



High-level/Declarative Programming:
Relational Data Model & Algebra

- Jim Gray, 1998



Transactions:
concurrent data-manipulation

- Michael Stonebraker, 2014

Relational DBMS
(e.g. Ingres, Postgres) and
modern DBMSs
(e.g. C-store, H-store, SciDB)

Outline For Today

1. Overview of DBMSs:
    1. Challenges with data management
    2. How DBMSs help overcome these challenges
        - Physical data independence, high level query language, constraints and transactions
2. Course & Administrative Information

Course components

- Relational databases (Lectures 1-10)
  - Relational algebra, SQL, app programming, database design
- Database internals (Lectures 11-18)
  - Storage, indexing, query processing and optimization, transactions, Concurrency & recovery

More about the Teaching Team

- Instructor: Sujaya Maiyya
  - Professor in the CS dept
  - Email: smaiyya@uwaterloo.ca
  - https://cs.uwaterloo.ca/~smaiyya/
- Instructional support coordinator: Sylvie Davies
  - Email: sldavies@uwaterloo.ca
- IAs and TAs
  - Shubhankar Mohapatra (IA)
  - Haseeb Ahmed
  - Alexandar William Caton
  - Chanaka Lakmal Lokupothagamage Don
  - Zhengyuan Dong
  - Harrum Noor
- Office hours will be posted on Learn/Piazza

Who to reach out to?

- Any course content related questions → reach out to me

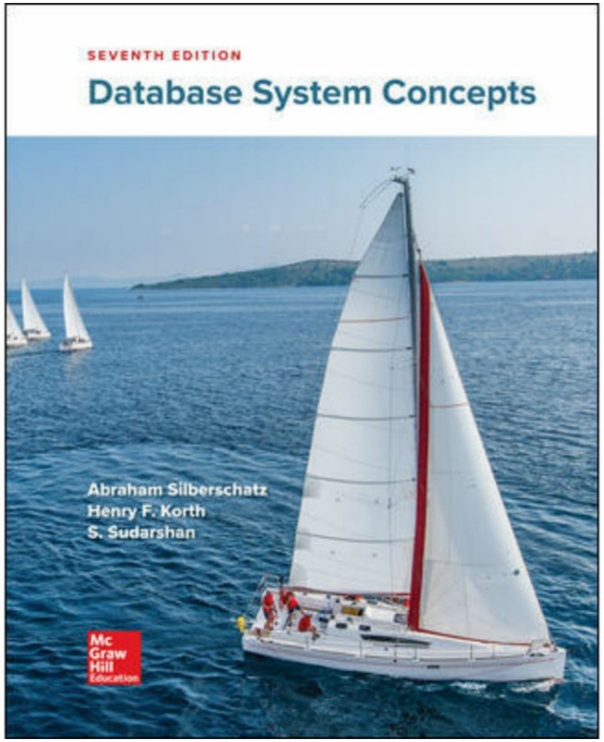
- Regrade requests → To the respective TA and IA

- Approved regrade, late policy, verification of illness → reach out to Sylvie and cc me

- My work hours: 9-6PM Monday to Friday

Textbook

- Database System Concepts (Seventh Edition) Abraham Silberschatz, Henry F. Forth and S.Sudarshan, McGraw Hill.



Logistics

- Course Website:
  - https://cs.uwaterloo.ca/~smaiyya/cs348
  - Course schedule, lecture notes
- Learn:
  - https://learn.uwaterloo.ca/
  - Assignment questions/partial solutions, project info
- Piazza for student discussion, Q&A, TAs info:
  - https://piazza.com/class/lvqzlkkmopm1uv
  - **For student-student discussions**
- Work submission: Crowdmark/Marmoset/Learn
  - Watch your emails for the links

Marking and Late Policies

- Marking and appeals:
  - For everything, there will be an appeal deadline that will be indicated on the front page
  - No appeals will be accepted past this date unless you were sick the entire period until the appeal date

- Late assignments/project deliverables
  - Late assignments will be accepted for 48 hours past the due date, but…
  - For each 24 hour past the due date, a 5% penalty will be applied (cumulatively) for assignments
  - For each 24 hour past the due date, a 25% penalty will be applied (cumulatively) for projects

Assessments

- 3 Assignments
- 1 Midterm Exam
- 1 Final Exam
- Group Project (Optional): Choose 1 mark breakdown
- But both exams are mandatory!

Mark Breakdown	Project-based	Exam-based
3 Assignments	30%	30%
Midterm Exam	10%	30%
Final Exam	20%	40%
Project	40%	-

Any use of GenAI is the assessments must be cited. You are accountable for the content and accuracy of all work you submit in this class.

Lectures

- Lecture slides released on Course Website before Tue/Thur

- Attendance is mandatory

- Lecture format:
  - Important announcements (Don’t miss this!)
  - Key points and examples
  - Exercises with partial solutions

- Will be using lecture materials from Prof. Xi He’s lectures

Project

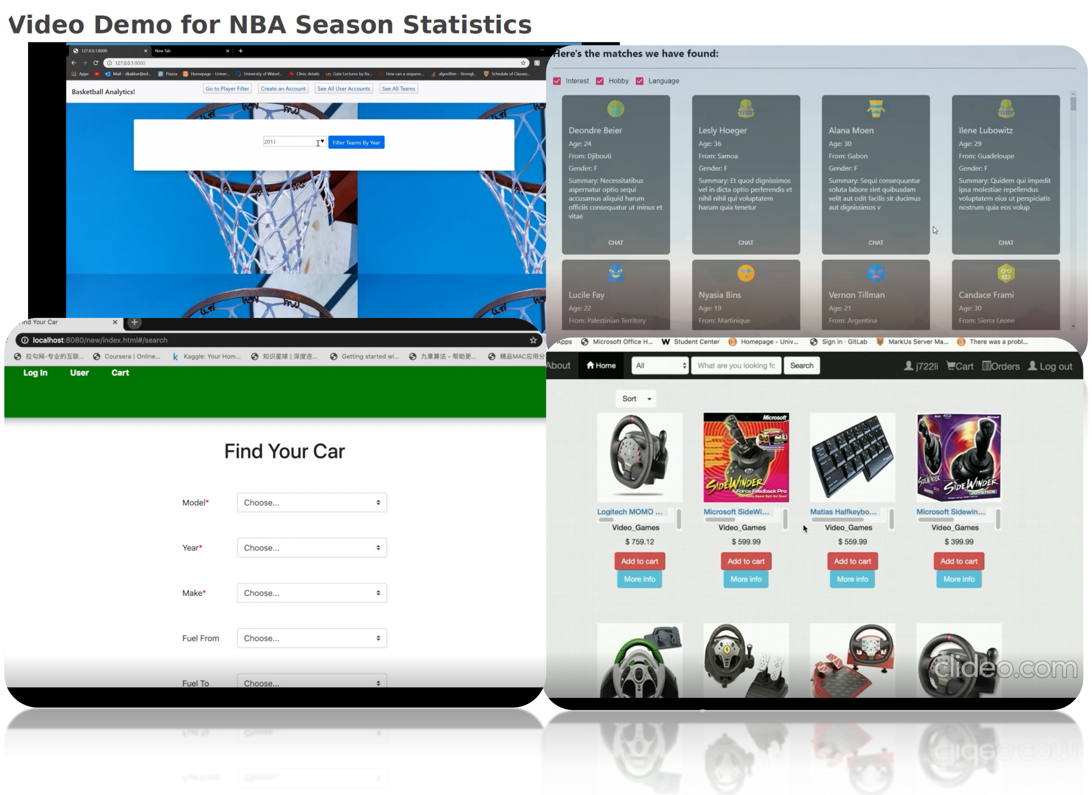
- Team of 4-5 students (minimum 4, maximum 5)
- DB-supported applications
- Project timeline
  - Milestone 0: form a team by Thu, May 23
  - Milestone 1: proposal by Thu, Jun 20
  - Milestone 2: mid-term report by Tue, Jul 9
  - Final: report + demo by Thu, Jul 25

- More details will be released in week 2, but you can start to brainstorm and find your teammates!
  - Members from only **002 and 003** sections are allowed.
  - Piazza is a good place to find teammates.


levels of procrastination
1. non-procrastinator
does work early
2. Sunday-night slacker
slack all weekend
3. super slacker
2-month projects, essays, labs etc.
all in 1 night
4. master procrastinator
still procrastinating after deadline
chibird

Project

- Project demos from previous years



What's next?

- Lecture 2: Relational model and relational algebra