CS2102: Database Systems

Lecture 1 — Introduction & Relational Model

Overview

- **Why Database Management Systems (DBMS)?**
  - **Challenges for data-intensive applications**
  - From file-based data management to DBMS
  - Core concepts of DBMS (transactions, data abstraction)

- Relational Database Model
  - Motivation & history
  - Core concepts: relation, domain, schema, etc.
  - Integrity constraints

Common Challenges for Data-Intensive Applications

- Fast access to information in huge volumes of data

  ➜ **Efficiency**

- "All-or-nothing" changes to data
  (e.g. bank transfer: debit + credit)

  ➜ **Transactions**

- Parallel access and changes to data

  ➜ **Data Integrity**

VISA

5,000 tps*

amadeus
(global travel booking platform)

100,000 tps*

Alibaba.com
(on Singles Day 2020)

544,000 tps*

* transactions per seconds; self-reported peak values

Common Challenges for Data-Intensive Applications

- Fast and reliable handling of failures
  (e.g., HDD/SDD/system crash, power outage, network disruption)

  ➜ **Recovery**

- Fine-grained data access rights

  ➜ **Security**

Only HR & Management

EmpID	Name	Office	Phone	DOB	Salary
1	Alice	02-05	4520	10-08-1988	7,500
2	Bob	02-10	4530	06-11-2001	4,800
3	Carol	01-06	4540	25-02-1995	5,500

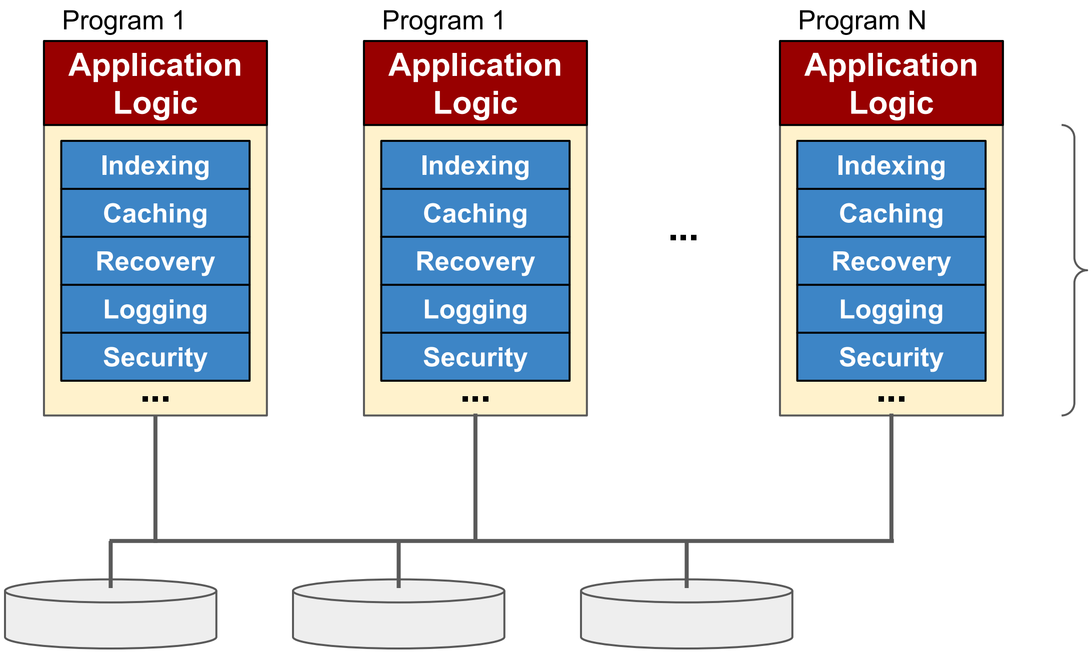
All employees

Overview

- **Why Database Management Systems (DBMS)?**
  - Challenges for data-intensive applications
  - **From file-based data management to DBMS**
  - Core concepts of DBMS (transactions, data abstraction)

- Relational Database Model
  - Motivation & history
  - Core concepts: relation, domain, schema, etc.
  - Integrity constraints

File-Based Data Management

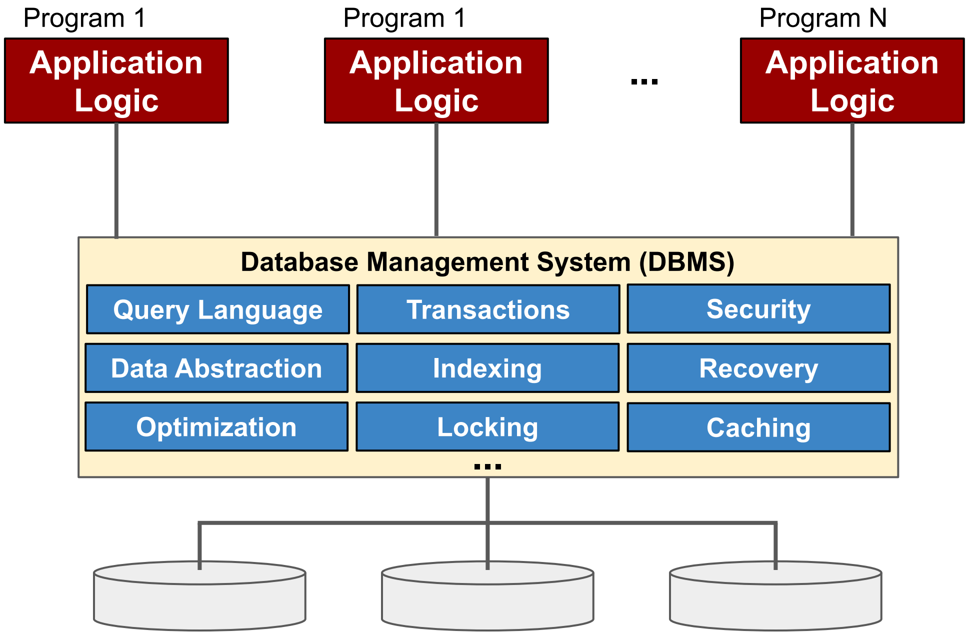


- Complex, low-level code
- Often similar requirements across different programs

➜ Problems / Challenges:

- High development effort
- Long development times
- Higher risk of (critical) errors

Data Management with DBMS



- Complex, low-level code moved from application logic to DBMS
- DBMS = set of universal and powerful functionalities for data management

➜ Benefits:

- Faster application development
- Increased productivity
- Higher stability / less errors

Overview

- **Why Database Management Systems (DBMS)?**
  - Challenges for data-intensive applications
  - From file-based data management to DBMS
  - **Core concepts of DBMS** (transactions, data abstraction)

- Relational Database Model
  - Motivation & history
  - Core concepts: relation, domain, schema, etc.
  - Integrity constraints

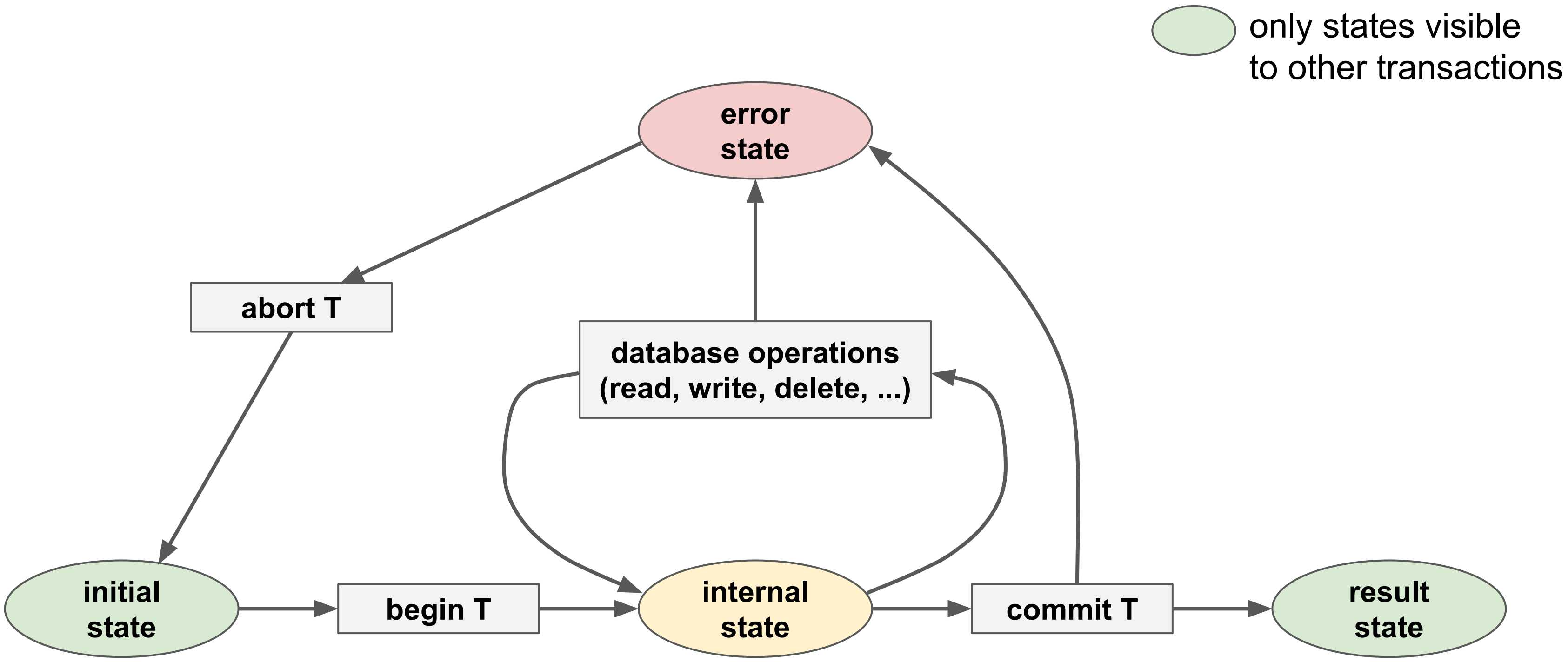
Separating "Files Only" from DBMS: Transactions

- **Transaction**
  - Finite sequence of database operations (reads and/or writes)
  - Smallest logical unit of work from from an application perspective

- Each transaction T has the following properties:
  - **Atomicity**: either all effects of T are reflected in the database or none ("all or nothing")
  - **Consistency**: the execution of T guarantees to yield a correct state of the database
  - **Isolation**: the execution of T is isolated from the effects of concurrent transactions
  - **Durability**: after the commit of T, its effects are permanent even in case of failures

➜ **ACID** properties of transactions

Transition Graph of a Transaction T



Transactions — Example: Update Bank Account Balance

- Very simple transaction

Transaction update(X, amount)

```
begin:
        read(X)
        X = X + amount
        write(X)
commit
```

- Assume 2 transactions

(initial balance B: 1,000)

  - T_1(B, 500)
  - T_2(B, 100)

Serial execution of T_1 and T_2

T_1(B, 500)	T_2(B, 100)
begin	
read(B)	
B = B + 500	
write(B)	
commit	
	begin
	read(B)
	B = B + 100
	write(B)
	commit

+ Correct final result (by definition)

− Less resource utilization and low throughput

Concurrent Execution — Common Problems

T_1(B, 500)	T_2(B, 100)
begin	
read(B)	
B = B + 500	
	begin
	read(B)
	B = B + 100
write(B)	
commit	
	write(B)
	commit

Final balance B = 1,100
(effect of T_1 overwritten)

➜ Lost Update

T_1(B, 500)	T_2(B, 100)
begin	
read(B)	
B = B + 500	
write(B)	
	begin
	read(B)
	B = B + 100
	write(B)
	commit
abort	

Final balance B = 1,600
(when it should be 1,100)

➜ Dirty Read

T_1(B, 500)	T_2(B, 100)
begin	
read(B)	
	begin
	read(B)
	B = B + 100
	write(B)
	commit
read(B)	
...	

Balance B is retrieved twice
but the values differ

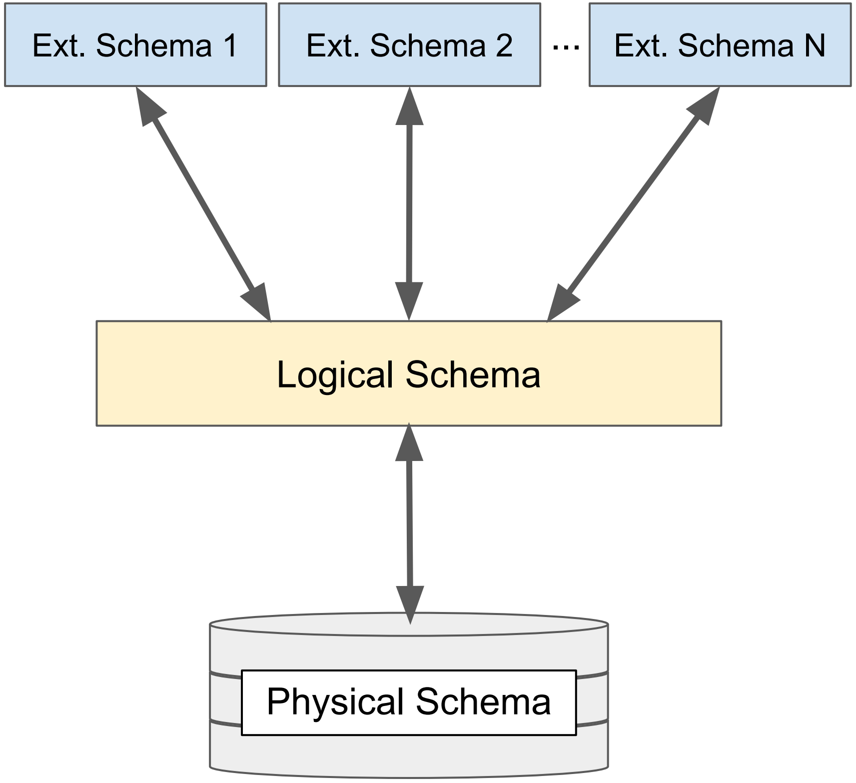
➜ Unrepeatable Read

Requirement for Concurrent Transactions: Serializability

- Serializable transaction execution
  - A concurrent execution of a set of transactions is **serializable** if this execution is equivalent to some serial execution of the same set of transactions
  - Two executions are equivalent if they have the same effect on the data

- Core tasks of DBMS
  - **Support concurrent executions** of transactions to optimize performance
  - **Enforce serializability** of concurrent executions to ensure integrity of data

3-Tier Architecture of DBMS — Levels of Data Abstraction

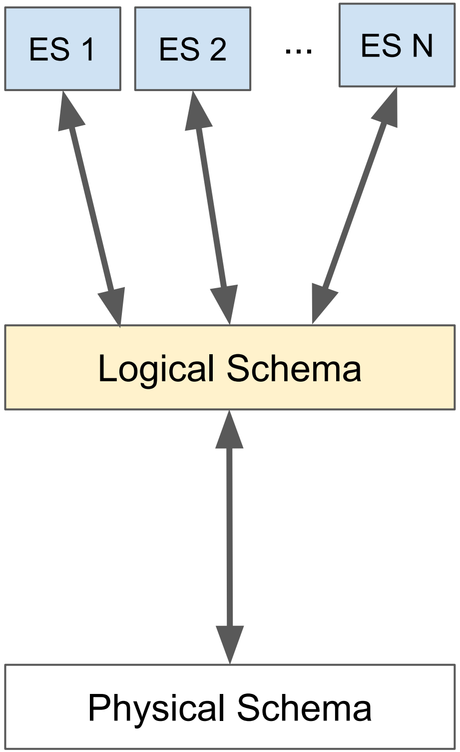


- User or group-specific view on the data

- Logical organization of data ➜ **data model**
  (e.g., <u>relations/tables</u>, objects, graphs)
- Unified representation of all data
- Support of **physical data independence** and **logical data independence**

- Organization of data on disk and in memory
- Database as collection of fields, arrays, records, files, pages, etc.

Data Independence



Logical data independence

- Ability to change logical schema without affecting external schemas
  (e.g., adding/deleting/updating attributes, changing data types, changing data model)

Physical data independence

- Representation of data independent from physical scheme
- Physical schema can be changed without affecting logical schema
  (e.g., creating indexes, new caching strategies, different storage devices)

Study of DBMS — Scope of CS2102

- **Database design**
  - How to model the data requirements
  - How to organize data using a DBMS
- **Database programming**
  - How to create, query and update a database
  - How to specify data constraints
  - How to use SQL in applications
- DBMS implementation
  - How to build a DBMS?

Topics covered in CS2102

Relation Model
ER Model
Schema Refinement

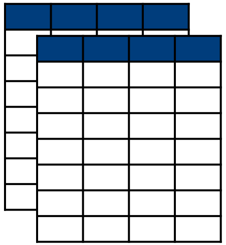
Relational Algebra

SQL

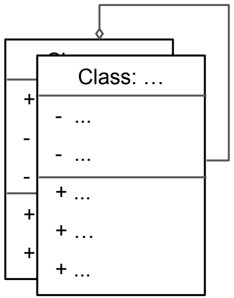
Describing Data in a DBMS

- **Data Model**
  - Set of concepts for describing the data
  - Framework to specify structure of a DB

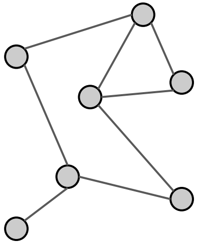
Tables



Objects



Graphs



- **Schema**
  - Description of the structure of a DB using the concepts provided by the data model

Employees (id: **integer**, name: **text**, dob: **date**, salary: **numeric**)

- **Schema Instance**
  - Content of a DB at a particular time

Table "Employees"

ID	Name	DOB	Salary
1	Alice	10-08-1988	7,500
2	Bob	06-11-2001	4,800
3	Carol	25-02-1995	5,500

Overview

- Why Database Management Systems (DBMS)?
  - Challenges for data-intensive applications
  - From file-based data management to DBMS
  - Core concepts of DBMS (transactions, data abstraction)

- **Relational Database Model**
  - **Motivation & history**
  - Core concepts: relation, domain, schema, etc.
  - Integrity constraints

Timeline of DBMS (Regarding the Supported Data Model)

- "Historical" models
  - Hierarchical model
  - Network model

- **Relation Model**

  (early: prototypes 1970+, commercial products: 1980+)
  - Commercial RDBMS
  - Open-source RDBMS

- Object-oriented model
  - Native OO model (e.g., Objectstore, 1988)
  - Object-relational model

    (now supported by most RDBMS)

- More recent development
  - NoSQL models, in-memory DBMS

    (e.g.. Cassandra, 2008; MongoDB, 2009; Redis, 2009)

Commercial systems*

Open-source systems

* some vendors offer free versions with limited functionalities

RDBMS (still) Reign Supreme

Rank			DBMS	Database Model	Score		
Aug 2023	Jul 2023	Aug 2022			Aug 2023	Jul 2023	Aug 2022
1.	1.	1.	Oracle	Relational, Multi-model	1242.10	-13.91	-18.70
2.	2.	2.	MySQL	Relational, Multi-model	1130.45	-19.89	-72.40
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	920.81	-0.78	-24.14
4.	4.	4.	PostgreSQL	Relational, Multi-model	620.38	+2.55	+2.38
5.	5.	5.	MongoDB	Document, Multi-model	434.49	-1.00	-43.17
6.	6.	6.	Redis	Key-value, Multi-model	162.97	-0.80	-13.43
7.	↑8.	↑8.	Elasticsearch	Search engine, Multi-model	139.92	+0.33	-15.16
8.	↓7.	↓7.	IBM Db2	Relational, Multi-model	139.24	-0.58	-17.99
9.	9.	9.	Microsoft Access	Relational	130.34	-0.38	-16.16
10.	10.	10.	SQLite	Relational	129.92	-0.27	-8.95

Source: https://db-engines.com/en/ranking

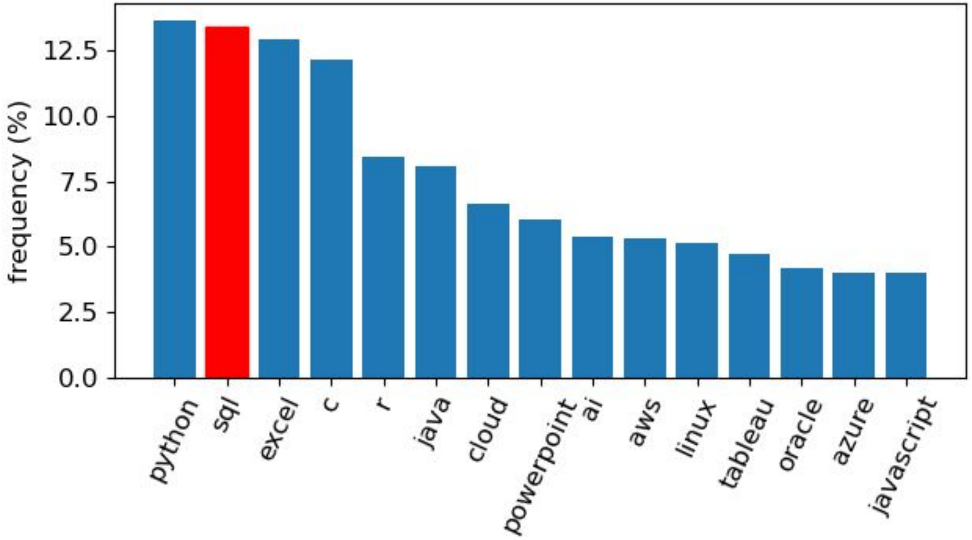
RDBMS (still) Reign Supreme

Java, SQL and Python are the most in-demand digital skills

Key Skill: SQL, Because Companies are Obsessed with Data

Want a Job in Data? Learn SQL.

- Analysis of job descriptions
  - 15k+ job offers from JobStreet
    (data analyst, data engineer, data scientist)
  - Quick-&-dirty keyword extraction
  - ...but check for yourself! :)



Overview

- Why Database Management Systems (DBMS)?
  - Challenges for data-intensive applications
  - From file-based data management to DBMS
  - Core concepts of DBMS (transactions, data abstraction)

- **Relational Database Model**
  - Motivation & history
  - **Core concepts: relation, domain, schema, etc.**
  - Integrity constraints

The Relational Model

- Proposed by Edgar F. Codd in 1970

- Basic concept: **relations**
  (tables with rows and columns)

Table "Employees"

id	name	dob	salary
1	Alice	10-08-1988	7,500
2	Bob	06-11-2001	4,800
3	Carol	25-02-1995	5,500

- Relation schema: definition of a relation
  - Specifies attributes (columns) and data constraints (e.g., domain constraints)

Employees (id: **integer**, name: **text**, dob: **date**, salary: **numeric**)

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

The Relational Model

- **Domain** — set of atomic values (e.g., integer, numeric, text)
  - $dom(A_i)$ domain of attribute A_i = set of possible values of A_i
  - Each value v of attribute A_i : $v \in dom(A_i)$ or $v = null$
  - $null$ — special value indicating the v is not known or not specified

- **Relation** — set of tuples (or records)
  - $R(A_1, A_2, ..., A_n)$ — relation schema with name R and n attributes $A_1, A_2, ..., A_n$
  - Each instance of schema R is a relation which is a subset of

    $\{(a_1, a_2, ..., a_n) \mid a_i \in dom(A_i) \cup \{null\} \}$

Example

- Relational schema: Modules(course, mc, exam) with
  - dom(course) = {cs2102, cs3223, cs4221}
  - dom(mc) = {2, 4}
  - dom(exam) = {yes, no}

- Each instance of "Modules" is a subset of

  {cs2102, cs3223, cs4221, null} × {2, 4, null} × {yes, no, null}

max. 36 tuples

course	mc	exam
cs2102	2	yes
cs2102	2	no
cs2102	4	yes
cs2102	4	no
cs3223	2	yes
...	...	...
null	4	no
null	null	no
null	null	null

Quick Quiz

The Relational Model

- **Relational database schema** — set of relation schemas + data constraints

  Movies (id: **integer**, title: **text**, genre: **text**, opened: **date**)

  Cast (movie_id: **integer**, actor_id: **integer**, role: **text**)

  Actors (id: **integer**, name: **text**, dob: **date**)

- **Relational database** — collection of tables

Table "Movies"

id	title	genre	opened
101	Aliens	action	1986
102	Logan	drama	2017
103	Heat	crime	1995
104	Terminator	action	1984
...	...	...	

Table "Cast"

movie_id	actor_id	role
101	20	Ellen Ripley
101	23	Private Hudson
101	54	Corporal Hicks
102	21	Logan
104	23	Punk Leader
...	...	...

Table "Actors"

id	name	dob
20	Sigourney Weaver	08-10-1949
21	Hugh Jackman	12-10-1968
22	Tom Hanks	09-07-1956
23	Bill Paxton	17-05-1955
...	...	...

Challenge: Ensuring Data Integrity

- The definition $R(A_1, A_2, ..., A_n) \subseteq \{(a_1, a_2, ..., a_n) \mid a_i \in dom(A_i) \cup \{null\}\}$ allows:

Table "Movies"

id	title	genre	opened
101	Aliens	action	1986
101	Logan	drama	2017
103	Heat	crime	1995
104	Terminator	action	1984

Table "Cast"

movie_id	actor_id	role
101	20	Ellen Ripley
101	23	Private Hudson
101	**54**	Corporal Hicks
102	21	Logan
abc	23	Punk Leader

Table "Actors"

id	name	dob
20	Sigourney Weaver	**08-10-2049**
21	Hugh Jackman	12-10-1968
null	Tom Hanks	09-07-1956
23	Bill Paxton	17-05-1955

Can we tell the DBMS what are valid tuples and attribute values?

➜ **Integrity Constraints**

Overview

- Why Database Management Systems (DBMS)?
  - Challenges for data-intensive applications
  - From file-based data management to DBMS
  - Core concepts of DBMS (transactions, data abstraction)
- **Relational Database Model**
  - Motivation & history
  - Core concepts: relation, domain, schema, etc.
  - **Integrity constraints**

Integrity Constraints

- **Integrity Constraint** — condition that restricts what constitutes valid data
  - DBMS checks that tables only ever contain valid data ➜ data integrity

- 3 main structural integrity constraints of the Relation Model
  ("structural" = inherent to the data model, independent from the application)
  - Domain constraints (e.g., cannot store a string in a integer column)
  - Key constraints
  - Foreign key constraints

- General constraints
  - Depend on the specific application
  - Covered in later lectures (keyword: triggers)

Table "Cast"

movie_id	actor_id	role
101	20	Ellen Ripley
101	23	Private Hudson
102	21	Logan
abc	23	Punk Leader

Key Constraints

- **Superkey** — subset of attributes that <u>uniquely</u> identifies a tuple in a relation
  - e.g., {id, title}, {id, title, opened}
- **Key** — superkey that is also minimal, i.e., no proper subset of the key is a superkey
  - e.g., {id} (maybe: {title, opened})
- **Candidate keys** — set of all keys for a relation
- **Primary key** — selected candidate key for a relation
  - Important: values of primary key attributes cannot be *null* (**entity integrity constraint**)

Movies (<u>id: **integer**</u>, title: **text**, genre: **text**, opened: **date**)

Table "Movies"

id	title	genre	opened
101	Aliens	action	1986
102	Logan	drama	2017
103	Heat	crime	1995
104	Terminator	action	1984

Quick Quiz

Foreign Key Constraints (also: referential integrity constraints)

- **Foreign key** — subset of attributes of relation A
  if it refers to the (primary) key in a relation B

Table "Movies"

id	title	genre	opened
101	Aliens	action	1986
102	Logan	drama	2017
103	Heat	crime	1995
104	Terminator	action	1984

referenced relation

Table "Cast"

movie_id	actor_id	role
101	20	Ellen Ripley
101	23	Private Hudson
102	21	Logan
104	23	Punk Leader

referencing relation

Table "Actors"

id	name	dob
20	Sigourney Weaver	08-10-1949
21	Hugh Jackman	12-10-1968
22	Tom Hanks	09-07-1956
23	Bill Paxton	17-05-1955

referenced relation

- Requirement: each foreign key in referencing relation must
  - appear as primary key in referenced relation OR
  - be a *null* value

Foreign Key Constraints

- Referencing & referenced relation can be the same relation
  - Example: each employee has at most one manager

Table "Employees"

id	name	dob	salary	manager
1	Alice	10-08-1988	7,500	null
2	Bob	06-11-2001	4,800	3
3	Carol	25-02-1995	5,500	1
4	Dave	18-06-1999	6,000	null
5	Erin	09-05-2000	5,000	1

- A relation can be referencing and referenced relation for different relations

Table "Genre"

genre	description
action	exciting stuff
drama	suspenseful stuff
crime	mysterious stuff

Table "Movies"

id	title	genre	opened
101	Aliens	action	1986
102	Logan	drama	2017
103	Heat	crime	1995
104	Terminator	action	1984

Table "Cast"

movie_id	actor_id	role
101	20	Ellen Ripley
101	23	Private Hudson
102	21	Logan
104	23	Punk Leader

Quick Quiz

Integrity Constraints

- Limitations
  - Structural integrity constraints do cover application-independent constraints
    (e.g., limiting the domain to valid values)
  - Covered later: application-dependent constraints derived from deeper semantics of the data

Table "Actors"

id	name	dob
20	Sigourney Weaver	**08-10-2049**
21	Hugh Jackman	12-10-1968
22	Tom Hanks	09-07-1956
23	Bill Paxton	17-05-1955

- Practical considerations
  - Integrity constraints are optional, not mandatory – but we typically want them!
    (in practice, domain constraints are mandatory, i.e., we need to specify the data type for each attribute)
  - Integrity constraints may affect performance*
    (checking constraints require additional processing steps)

***Sidenote:** Key constraints typically involve the creation indexes which can significantly boost query performance!

Relational Model — Cheat Sheet

Relation name

Attribute

Table "Movies"

id	title	genre	opened	...
101	Aliens	action	1986	...
102	Logan	drama	2017	...
103	Heat	crime	1995	...
104	Terminator	action	1984	...
105	Hot Fuzz	comedy	2007	...
106	Saw	horror	2004	...
...	...	...	...	...

Relation schema

Tuple / Record

Relation

Attribute value

Relational Model — Cheat Sheet

Term	Description (informal)
attribute	Column of a table
domain	Set of possible values for an attribute
attribute value	Element of a domain
relation schema	Set of attributes (with their data types + relation name)
relation	Set of tuples
tuple	Row of a table
database schema	Set of relation schemas
database	Set of relations / tables

Relational Model — Cheat Sheet

Term	Description (informal)
(candidate) key	Minimal set of attributes that uniquely identify a tuple in a relation
primary key	Selected key (in case of multiple candidate keys)
foreign key	Set of attributes that is a key in referenced relation
prime attribute	Attribute of a (candidate) key

- Terminology: DB. vs DBS vs. DBMS

DBS = DBMS + n*DB (n>0)

Summary

- Advantages of DBMS for large-scale data management (compared to "files only")
  - Transactions with ACID properties to guarantee integrity of the data
  - Levels of abstraction for data independence
- Relational Model
  - Unified representation of all data as tables (relations)
  - (Structural) integrity constraints to specify restrictions on what constitutes correct/valid data
- Outlook for next lecture: SQL 1
  - Creating and modifying database schemas (incl. integrity constraints)
  - Inserting, updating, and deleting data

Quick Quiz Solutions

Quick Quiz (Slide 23)

Quick Quiz (Slide 28)

Quick Quiz (Slide 31)