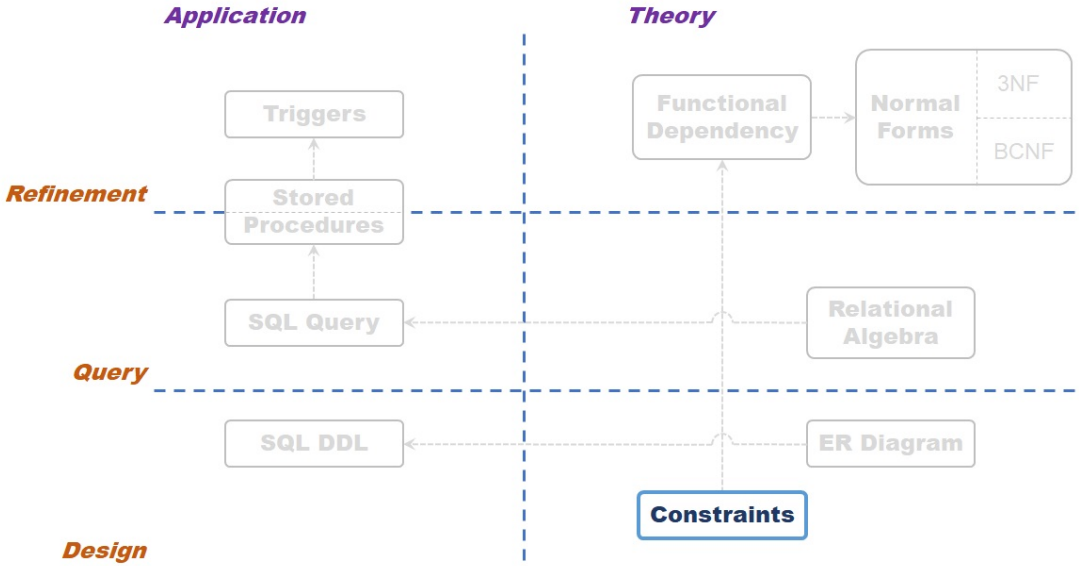
CS2102: Database Systems

Introduction

Roadmap


Application
Theory
Triggers
Functional Dependency
Normal Forms
3NF
BCNF
Refinement
Stored Procedures
SQL Query
Relational Algebra
Query
SQL DDL
ER Diagram
Constraints
Design

Roadmap

Overview

Overview

DBMS

- Database
- Challenges
- File to DBMS
- Core Concepts

Relational Model

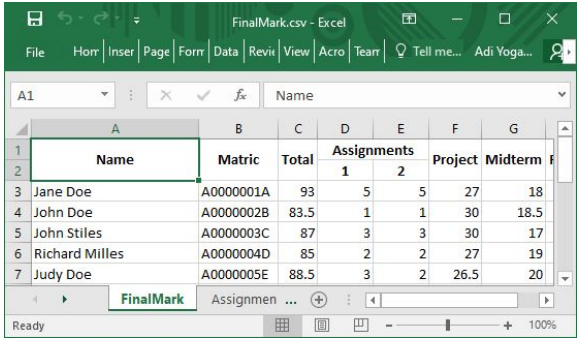
- Motivation & History
- Core Concepts
- Constraints

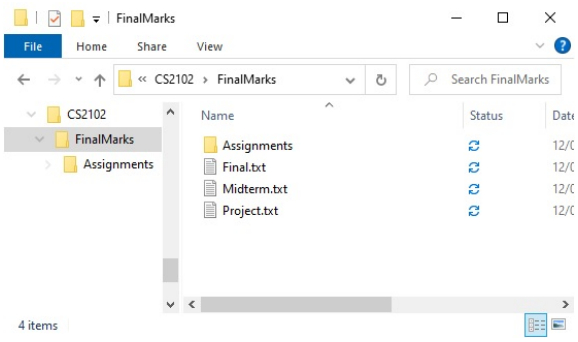
DBMS

What is a Database?

Definition

A **database** is a collection of *organized* data, information, and records.





Common Challenges

Data-Intensive Applications

Efficiency

Fast access to information in **huge** volumes of data.

Transactions

"All-or-nothing" changes to data *(e.g., bank transfer)*.

Data Integrity

Parallel access and **changes** to data.

Recovery

Fast and **reliable** handling of failures.

Security

Fine-grained data **access** rights.

VISA

5,000 tps*

amadeus

100,000 tps*

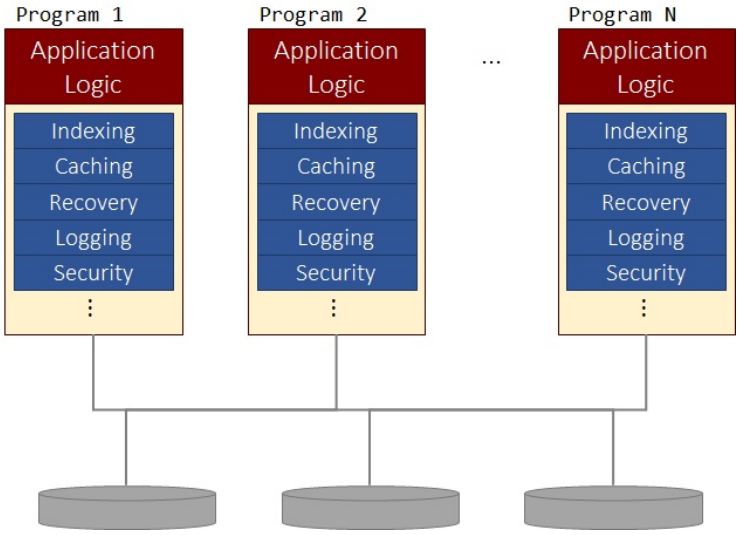
Alibaba.com

544,000 tps*

***tps:** *transactions per seconds (self-reported peak values).*

Data Management

File-Based Data Management



Note

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

Problems

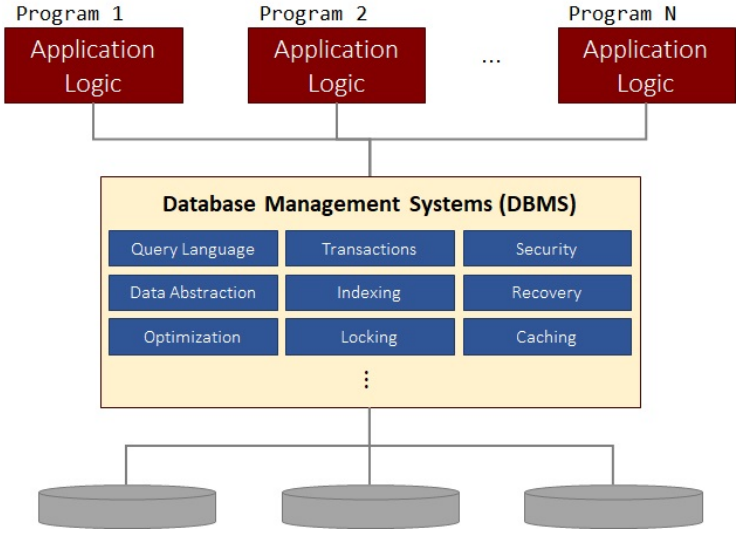
- High development effort
- Long development times
- Higher risk of errors

DBMS

Database
Challenges
Management
- File-Based
- DBMS
- Comparison
Transactions
Serialization
Architecture
Study

Data Management

Data Management with DBMS



Note

- Application logic moved to DBMS
- Other set of universal and powerful functionalities

Benefits

- Faster application development
- Increased productivity
- Higher stability

Database
Challenges
Management
- File-Based
- DBMS
- Comparison
Transactions
Serialization
Architecture
Study

Data Management

Comparison

File-Based

CS2102.py - C:/Users/adi-yoga/Documents/OneDrive - National U...

File Edit Format Run Options Window Help

```
import file_io

finals   = file_io.read_file('Final.txt')
midterms = file_io.read_file('Midterm.txt')
projects = file_io.read_file('Project.txt')
assignments = file_io.read_folder('Assignments')

def find_student(min_total):
  total = get_total(finals, midterms, projects, assi
  res = {}
  for matric in total:
    if total[matric] >= min_total:
      res[matric] = total[matric]
  return res

def get_total(finals, midterms, projects, assignment
  total = {}
  for matric in assignments:
    total_mark = 0
    total_mark += assignments[matric][0]
    total_mark += assignments[matric][1]
    total_mark += finals[matric][0]
    total_mark += midterms[matric][0]
    total_mark += projects[matric][0]
    total[matric] = total_mark
```

Ln: 20 Col: 0

DBMS

SQL Shell (psql)

```
Username [postgres]:
psql (11.1)
WARNING: Console code page (437) differs from Wind
ows code page (1252)
         8-bit characters might not work correctly
. See psql reference
         page "Notes for Windows users" for detail
s.
Type "help" for help.

postgres=# SELECT matric, total
postgres-# FROM   FinalMarks
postgres-# WHERE  total >= 93;
  matric   | total
-----------+-------
 A0000001A |    93
(1 row)


postgres=#
```

Transactions

Definition

A **finite sequence** of database operations *(reads and/or writes)* and constitutes the *smallest* logical unit of work from the application perspective.

Properties

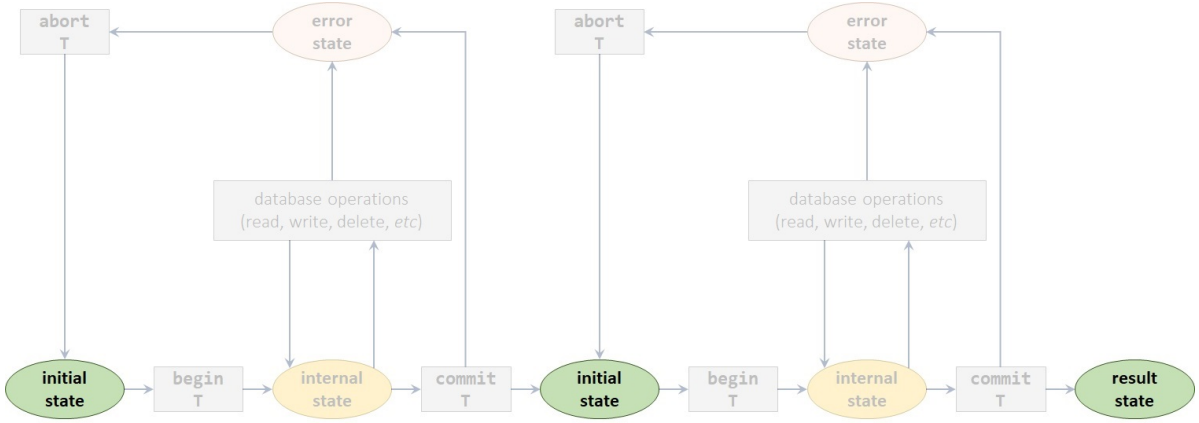
Each transaction *T* has the following properties:

- ***A*tomicity** either all effects of *T* are reflected in the database or none
- ***C*onsistency** *T* guarantees to yield the correct state of the database
- ***I*solation** *T* is isolated from the effect of concurrent transactions
- ***D*urability** after a commit of *T*, its effects are permanent

ACID properties of transactions

Transactions

Transition Graph



Addendum

What you will see as a database user is the sequence of **consistent** state depicted in green. In between, the database may go into a *temporary* inconsistent state. Here, we define consistency as satisfying all user specified constraints.

Transactions

Sequential

```
define transaction T(U, Amount) {
  BEGIN
  Bal1 = READ(U)       -- read   balance
  Bal2 = Bal1 + Amount -- update balance
  WRITE(U, Bal2)       -- write  balance
  COMMIT
}
```

Note

- Initial balance of U = 'A' to be $1000
- Two transactions:
  A. `T1 = T('A', 500)`
  B. `T2 = T('A', 100)`

Serial

T('A', 500)	T('A', 100)
BEGIN	
Bal1=READ('A')	
Bal2=Bal1+500	
WRITE('A',Bal2)	
COMMIT	
	BEGIN
	Bal1=READ('A')
	Bal2=Bal1+100
	WRITE('A',Bal2)
	COMMIT
⋮	⋮

☑ **Expected** $1600
☑ **Actual** $1600

Transactions

Addendum

There is no need to fully understand this. This is simply to illustrate what the database has made easier for us as database user.

Lost Update

T('A', 500)	T('A', 100)
BEGIN	
Bal1=READ('A')	
Bal2=Bal1+500	
	BEGIN
	Bal1=READ('A')
	Bal2=Bal1+100
	WRITE('A',Bal2)
	COMMIT
WRITE('A',Bal2)	
COMMIT	
⋮	⋮

☑ **Expected** $1600
☒ **Actual** **$1500**

Dirty Read

T('A', 500)	T('A', 100)
BEGIN	
Bal1=READ('A')	
Bal2=Bal1+500	
WRITE('A',Bal2)	
	BEGIN
	Bal1=READ('A')
	Bal2=Bal1+100
	WRITE('A',Bal2)
	COMMIT
ABORT	
⋮	⋮

☑ **Expected** $1100
☒ **Actual** **$1600**

Unrepeatable Read

T('A', 500)	T('A', 100)
BEGIN	
Bal1=READ('A')	
	BEGIN
	Bal1=READ('A')
	Bal2=Bal1+100
	WRITE('A',Bal2)
	COMMIT
Bal1=READ('A')	
Bal2=Bal1+500	
WRITE('A',Bal2)	
⋮	⋮

1st read **$1000**
2nd read **$1100**

Serialization

Equivalent Transaction

Definition

Two executions are **equivalent** if they have the ***same effect*** on the data.

Serializability

Definition

A concurrent execution of a set of transaction is **serializable** if this execution is ***equivalent to some*** serial execution of the same set of transactions.

DBMS

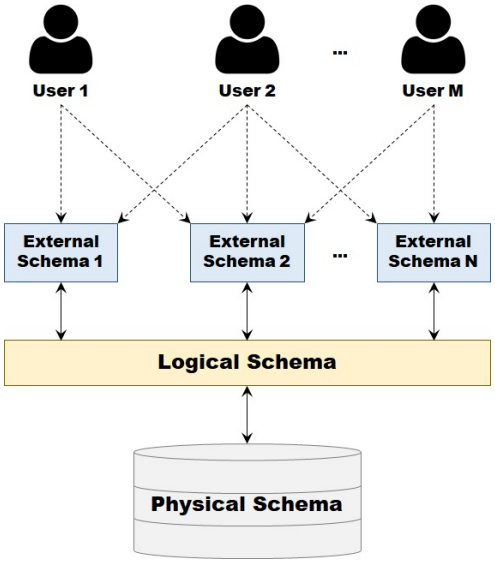
- Support concurrent executions of transactions to optimize performance
- Enforce serializability of concurrent executions to ensure integrity of data

DBMS

Database
Challenges
Management
Transactions
Serialization
Architecture
- Abstraction
- Independence
Study

3-Tier Architecture of DBMS

Levels of Data Abstraction



External Schema

- User or group-specific view on the data

Logical Schema

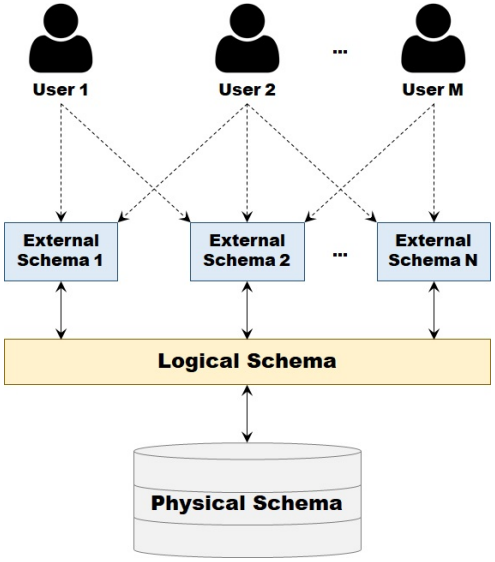
- Logical organization of data *(e.g., relations/tables, objects, graphs)*
- Unified representation of all data
- Support of **logical** data independence and **physical** data independence

Physical Schema

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

3-Tier Architecture of DBMS

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

Study of DBMS

Scope	Topic
Database Design	
• How to model the data requirements • How to organize data using a DBMS	Relational Model ER Model Schema Refinement
Database Programming	
• How to create, query, and update a database • How to specify data constraints • How to use SQL in applications	Relational Algebra SQL
DBMS Implementation	CS3223
• How to build a DBMS	*out of scope*

Relational Model

Relational Model

History
Popularity
Model
Integrity
Key
Foreign Key
Considerations

History of DBMS

"Historical" Models

- Hierarchical Model
- Network Model

Relational Model

- Commercial RDBMS
- Open-Source RDBMS

Object-Oriented Model

- Native OO Model *(e.g., Objectstore, 1988)*
- Object-Relational Model*

More Recent Development

- NoSQL Model; In-Memory DBMS

Commercial Systems

ORACLE IBM DB2 Microsoft SQL Server SAP SYBASE

Open-Source Systems

PostgreSQL MariaDB MySQL CUBRID SQLite

*Supported by most DBMS

Relational Model

History
Popularity
- Table
- Graph
- Job
Model
Integrity
Key
Foreign Key
Considerations

Popularity

Table

Rank			DBMS	Database Model	Score		
Jun 2022	May 2022	Jun 2021			Jun 2022	May 2022	Jun 2021
1.	1.	1.	**Oracle**	**Relational**	1287.74	+24.92	+16.80
2.	2.	2.	**MySQL**	**Relational**	1189.21	-12.89	-38.65
3.	3.	3.	**Microsoft SQL Server**	**Relational**	933.83	-7.37	-57.25
4.	4.	4.	**PostgreSQL**	**Relational**	620.84	+5.55	+52.32
5.	5.	5.	MongoDB	Document	480.73	+2.49	-7.49
6.	6.	7. ↑	Redis	Key-value	175.31	-3.71	+10.06
7.	7.	6. ↓	**IBM DB2**	**Relational**	159.19	-1.14	-7.85
8.	8.	8.	Elasticsearch	Search engine	156.00	-1.70	+1.29
9.	9.	10. ↑	**Microsoft Access**	**Relational**	141.82	-1.62	+26.88
10.	10.	9. ↓	**SQLite**	**Relational**	135.44	+0.70	+4.90

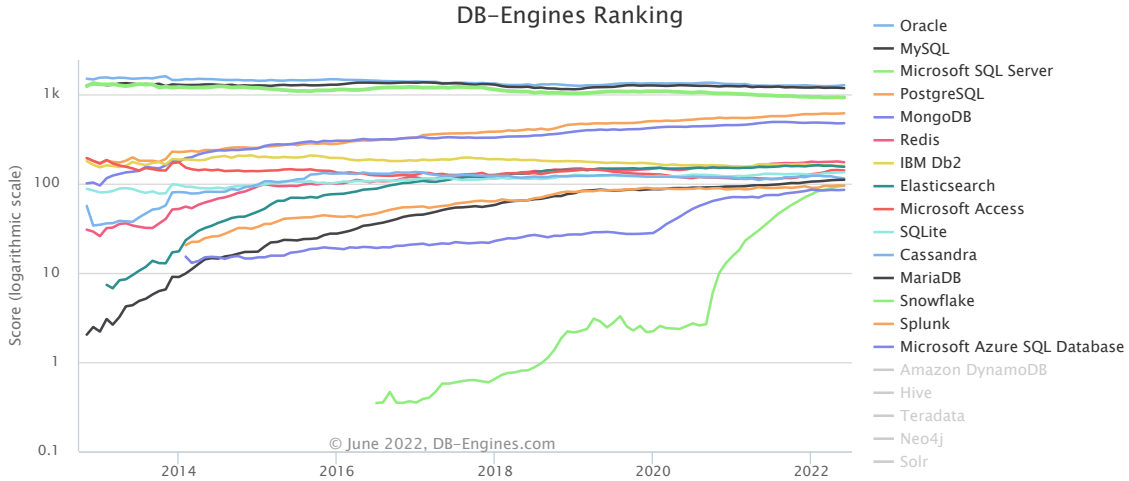
[#]Source: DB-Engines

Relational Model

History
Popularity
- Table
- Graph
- Job
Model
Integrity
Key
Foreign Key
Considerations

Popularity

Graph



[#]Source: DB-Engines

Relational Model

History
Popularity
- Table
- Graph
- Job
Model
Integrity
Key
Foreign Key
Considerations

Popularity

Analysis of Job Descriptions

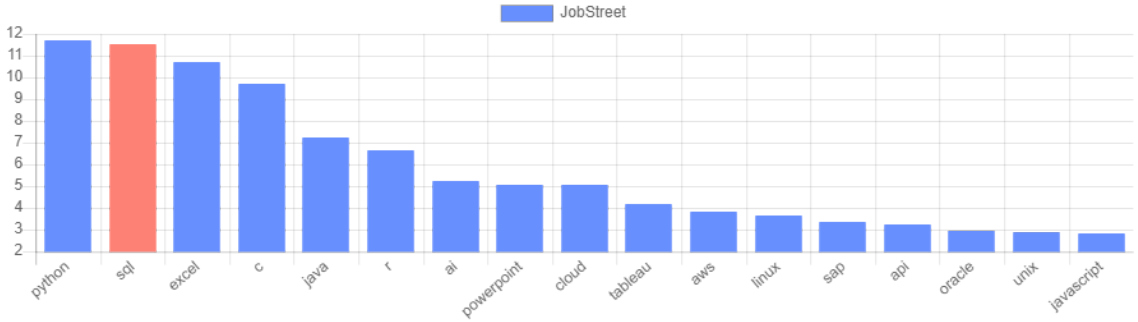
- 15k+ job offers from JobStreet
  (data analyst, data engineer, data scientist)
- Quick & dirty keyword extraction
  (by Dr. Christian Von Der WETH)

Anecdotes

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.



Relational Model

History
Popularity
Model
- Preliminary
- Domain & Relation
- Relational Schema
- Relational Database
Integrity
Key
Foreign Key
Considerations

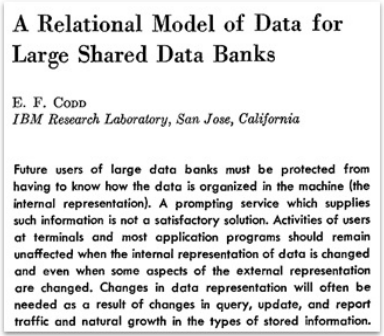
The Relational Model

Preliminary

- 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



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.

- **Relation Schema:** *Definition of a Relation*
  - Specifies the attributes *(columns)* and data constraints *(e.g., domain constraints)*

```
Employees(id:INT, name:TEXT, dob:DATE, salary:NUMERIC)
```

Addendum

When it is clear from the context, we often omit the data type *(e.g., `INT` or `TEXT`)*. Additionally, we may omit the data type in favor of simplicity and assume that the data type is the most "*logical*" data type.

Relational Model

History
Popularity
Model
- *Preliminary*
- ***Domain & Relation***
- *Relational Schema*
- *Relational Database*

Integrity
Key
Foreign Key
Considerations

The Relational Model

Domain

Definition

A **domain** is a set of *atomic* values *(e.g., `INT`, `NUMERIC`, `TEXT`)*.

- We denote the domain of attribute A_i as $\boldsymbol{dom(A_i)}$ *(i.e., the set of possible values of A_i)*
- Each value v of attribute A_i is either: $v \in dom(A_i)$ or $v =$ `NULL`
  - `NULL` is a special value indicating that the value v is *unknown*, *invalid*, *etc*

Relation

Definition

A **relation** is a set of tuple *(i.e., records)*.

- Simplified notation of ***relation schema*** where the domain is clear from context is $\boldsymbol{R(A_1, A_2, ..., A_n)}$
- Each ***instance*** of schema R is a relation which is a subset of $\boldsymbol{\{(a_1, a_2, ..., a_n) \mid a_i \in dom(A_i) \cup \{\texttt{NULL}\}\}}$
- We write $\boldsymbol{R.A_i}$ to refer to the attribute A_i of the relation R

Relational Model

History
Popularity
Model
- Preliminary
- Domain & Relation
- Relational Schema
- Relational Database
Integrity
Key
Foreign Key
Considerations

The Relational Model

Domain & Relation

Relational Schema

- *Modules(course, mc, exam)*
  - *dom(course)* = {cs2102, cs3223, cs4221}
  - *dom(mc)* = {2, 4}
  - *dom(exam)* = {yes, no}

Instance

- Subset of:
  {cs2102, cs3223, cs4221, NULL}
  × {2, 4, NULL}
  × {yes, no, NULL}

course	mc	exam
cs2102	2	yes
cs2102	2	no
cs2102	2	NULL
cs2102	4	yes
cs2102	4	no
⋮	⋮	⋮
NULL	NULL	NULL

Addendum

The symbol × on set is a Cartesian product. Do not forget to include the NULL value when computing the maximum possible cardinality.

Relational Model

History
Popularity
Model
- Preliminary
- Domain & Relation
- Relational Schema
- Relational Database
Integrity
Key
Foreign Key
Considerations

The Relational Model

Domain & Relation

Exercise #1

Assume a relation *R(A, B)* with

- *dom(A) = {x, y, z}*
- *dom(B) = {1, 2, 3, 4}*

Which tuples on the right **violate** the definition of relation *R*?

A	B	Row
x	4	*Row 1*
z	4	*Row 2*
NULL	2	*Row 3*
NULL	0	*Row 4*

	Choice	Comment	
A	Row 1		?
B	Row 2		?
C	Row 3		?
D	Row 4		?

Relational Model

History
Popularity
Model
- *Preliminary*
- ***Domain & Relation***
- *Relational Schema*
- *Relational Database*

Integrity
Key
Foreign Key
Considerations

The Relational Model

Domain & Relation

Exercise #2

Assume a relation *R(A, B)* with

- *dom(A) = {x, y, z}*
- *dom(B) = {1, 2, 3, 4}*

Which tuples on the right **violate** the definition of relation *R*?

A	B	Row
NULL	NULL	*Row 1*
x	4	*Row 2*
x	y	*Row 3*
z	1	*Row 4*

	Choice	Comment	
A	Row 1		?
B	Row 2		?
C	Row 3		?
D	Row 4		?

Relational Model

History
Popularity
Model
- *Preliminary*
- ***Domain & Relation***
- *Relational Schema*
- *Relational Database*

Integrity
Key
Foreign Key
Considerations

The Relational Model

Domain & Relation

Exercise #3

Assume a relation *R(A, B)* with

- *dom(A) = {x, y, z}*
- *dom(B) = {1, 2, 3, 4}*

Which tuples on the right **violate** the definition of relation *R*?

A	B	Row
x	4	*Row 1*
y	1	*Row 2*
x	1	*Row 3*
x	4	*Row 4*

	Choice	Comment	
A	Row 1		?
B	Row 2		?
C	Row 3		?
D	Row 4		?

Relational Model

History
Popularity
Model
- Preliminary
- Domain & Relation
- Relational Schema
- Relational Database
Integrity
Key
Foreign Key
Considerations

The Relational Model

Relational Database Schema

Definition

A **relational database schema** is a *set of relation schemas* + *data constraints*.

Example

Movies(

id	*: **INT***
title	*: **TEXT***
genre	*: **TEXT***
opened	*: **DATE***

)

Casts(

movie_id	*: **INT***
actor_id	*: **INT***
role	*: **TEXT***

)

Actors(

id	*: **INT***
name	*: **TEXT***
dob	*: **DATE***

)

[#]When the data type is clear from context, we often omit. If it is omitted, we choose the most "*logical*" type.

Relational Model

History
Popularity
Model
- *Preliminary*
- *Domain & Relation*
- *Relational Schema*
- ***Relational Database***

Integrity
Key
Foreign Key
Considerations

The Relational Model

Relational Database

Definition

A **relational database** is a *collections of tables*.

Addendum

Remember, this is a relation *(or relation instances)*. The name "relation" comes from the fact that each row *(or tuple)*, relate the values in the tuple. So 101 is related to Aliens, action, and 1986.

Example

Table "Movies"

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

Table "Casts"

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
⋮	⋮	⋮

Relational Model

History
Popularity
Model
Integrity
- Motivation
- Constraints
Key
Foreign Key
Considerations

Data Integrity

Motivation

Problem

Consider the *relational database* below. What are the problems with the tables?

Relational Database

Table "Movies"

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

Table "Casts"

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	08-10-2049
21	Hugh Jackman	12-10-1968
NULL	Tom Hanks	09-07-1956
23	Bill Paxton	17-05-1955

Addendum

1. The id 101 appears multiple times in Movies *(logically, the name id should be unique)*.
2. The movie_id 102 in Casts has no corresponding id in Movies.
3. Sigourney Weaver is born in the future?
4. Tom Hanks has no id?

[#]In this example, what you see is the entire database.

Relational Model

History
Popularity
Model
Integrity
- Motivation
- Constraints
Key
Foreign Key
Considerations

Data Integrity

Integrity Constraints

Definition

Integrity constraints are *conditions* that restrict what constitutes valid data.

Structural

- Independent of application
- Domain constraints *(e.g., cannot store `TEXT` in `INT` column)**
- Key constraints
- Foreign key constraints

General

- Dependent of application
- Check constraints
- Triggers

Table "Movies"

id	title	genre	opened
101	Aliens	Action	1986
101	Logan	Drama	2017

Table "Actors"

id	name	dob
20	Sigourney Weaver	**08-10-2049**
NULL	Tom Hanks	09-07-1956

*Some automatic type conversion may be performed, but for simplicity, we often ignore this.

Relational Model

History
Popularity
Model
Integrity
Key
- Superkey & Key
- Candidate & Primary
- Exercise
Foreign Key
Considerations

Key Constraints

Superkey

Definition

A **superkey** is a *subset* of attributes that *uniquely identifies* a tuple in a relation.

Key

Definition

A **key** is a ***superkey*** that is also *minimal* *(i.e., no proper subset of the key is a superkey)*.

NOTE:
- Minimal *(cannot be made smaller)* ≠ Minimum *(the smallest)*

Properties

- If *(A, B, C)* is *definitely* a ***superkey*** then *(A, B, C, D)* is a ***superkey***
- If *(A, B, C)* is *definitely* a ***key*** then *(A, B, C)* is a ***superkey***
- If *(A, B)* is *definitely* a ***superkey*** then *(A, B, C)* is **NOT** a ***key***
- If *(A, B)* is *definitely* a ***key*** then *it is possible that (B, C, D)* is also a ***key***
- Every relations have at least 1 superkey

Each table has exactly 1 primary key.

Relational Model

History
Popularity
Model
Integrity
Key
- *Superkey & Key*
- ***Candidate & Primary***
- *Exercise*

Foreign Key
Considerations

Key Constraints

Candidate Keys

Definition

The **candidate keys** is a the *set* of ***keys*** of a given relation.

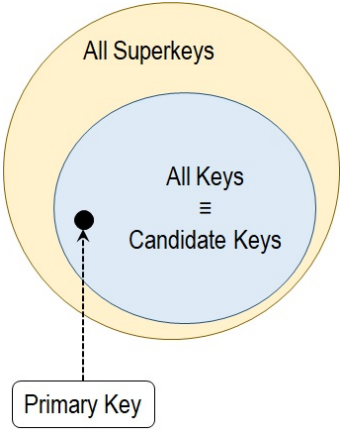
Primary Key

Definition

A **primary key** is a *selected* ***candidate keys***.

- We also add a constraint that primary key attributes cannot be `NULL`
- To simplify our notation, we *underline* the primary key in the schema notation

*Movies(id: **INT**, title: **TEXT**, genre: **TEXT**, opened: **DATE**)*



[#] We often say "*a candidate key*" to also mean "*a key*" and *vice versa*.

Relational Model

History
Popularity
Model
Integrity
Key
- *Superkey & Key*
- *Candidate & Primary*
- ***Exercise***

Foreign Key
Considerations

Key Constraints

Exercise #4

Consider a forum database with the following relation filled with many thousands of users: *Accounts(email:* ***TEXT****, password:* ***TEXT****, name:* ***TEXT****).*

Using reasonable guess, which subsets of attributes are **superkeys** of relation *Accounts*?

	Choice	Comment	
A	*{email}*		?
B	*{pwd}*		?
C	*{name}*		?
D	*{email, pwd}*		?
E	*{email, name}*		?
F	*{pwd, name}*		?
G	*{email, pwd, name}*		?

Relational Model

History
Popularity
Model
Integrity
Key
- *Superkey & Key*
- *Candidate & Primary*
- ***Exercise***
Foreign Key
Considerations

Key Constraints

Exercise #5

Consider a forum database with the following relation filled with many thousands of users: *Accounts(email:* ***TEXT****, password:* ***TEXT****, name:* ***TEXT****).*

Using reasonable guess, which subsets of attributes are **keys** of relation *Accounts*?

	Choice	Comment	
A	*{email}*		?
B	*{pwd}*		?
C	*{name}*		?
D	*{email, pwd}*		?
E	*{email, name}*		?
F	*{pwd, name}*		?
G	*{email, pwd, name}*		?

Relational Model

History
Popularity
Model
Integrity
Key
Foreign Key
- Definition
- Exercise
Considerations

Foreign Key Constraints

Foreign Key

Definition

A **foreign key** is a *subset* of attributes of relation R_1 that *refers to the* ***primary key**** of relation R_2.

- We say that R_1 is the ***referencing relation***
- We say that R_2 is the ***referenced relation***

Informally, we write $(R_1.A_{i1}, R_1.A_{i1}, ...) \dashrightarrow (R_2.A_{j1}, R_2.A_{j2}, ...)$.

Requirement

Each foreign key in R_1 must satisfy one of the following:

- Appear as ***primary key*** in R_2
- Be a NULL value *(or a tuple containing at least one `NULL` value)*

*In most DBMS, it can actually refer to any ***candidate keys***.

Addendum

In the tables below, we have:

- (Cast.movie_id) ⇢ (Movies.id)
- (Cast.actor_id) ⇢ (Actors.id)

Table "Movies"

id	title	genre	opened
101	Aliens	Action	1986
102	Logan	Drama	2017
⋮	⋮	⋮	⋮

Table "Cast"

movie_id	actor_id	role
101	20	Ellen Ripley
102	21	Logan
⋮	⋮	⋮

Table "Actors"

id	name	dob
20	Sigourney Weaver	08-10-1949
21	Hugh Jackman	12-10-1968
⋮	⋮	⋮

Relational Model

History
Popularity
Model
Integrity
Key
Foreign Key
- Definition
- Exercise
Considerations

Foreign Key Constraints

Foreign Key

Definition

A **foreign key** is a *subset* of attributes of relation R_1 that *refers to the* ***primary key****** of relation R_2.

- We say that R_1 is the ***referencing relation***
- We say that R_2 is the ***referenced relation***

Informally, we write $(R_1.A_{i1}, R_1.A_{i1}, ...) \twoheadrightarrow (R_2.A_{j1}, R_2.A_{j2}, ...)$.

Notes

- A ***referencing relation*** can be a ***referenced relation*** for different foreign key
- The ***referencing relation*** and the ***referenced relation*** can be the same relation

Requirement

Each foreign key in R_1 must satisfy one of the following:

- Appear as ***primary key*** in R_2
- Be a NULL value *(or a tuple containing at least one `NULL` value)*

Table "Employees"

id	name	dob	salary	*manager*
1	Alice	10-08-1988	7500	NULL
2	Bob	06-11-2001	4800	3
3	Carol	25-02-1995	5500	1

*In most DBMS, it can actually refer to any ***candidate keys***.

(Employees.manager) ↠ (Employees.id)

Relational Model

History
Popularity
Model
Integrity
Key
Foreign Key
- Definition
- Exercise
Considerations

Foreign Key Constraints

Exercise #6

Consider two tables *R(A, B)* and *S(C, D)* with *S.D* ⇝ *R.A*.

Which tuples in Table "*S* **violate** the **foreign key** constraint?

Table "R"

A	B
x	4
z	2
y	NULL

Table "S"

C	D	Row
a	w	*Row 1*
b	x	*Row 2*
c	y	*Row 3*
d	NULL	*Row 4*

	Choice	Comment	
A	Row 1		?
B	Row 2		?
C	Row 3		?
D	Row 4		?

Relational Model

History
Popularity
Model
Integrity
Key
Foreign Key
Considerations
- Limitations
- Practical

Considerations

Limitations

- Structural integrity constraints covers *application-independent* constraints
  - Limiting domain to valid values
  - Checking existence of values in another relations
- **Not covered:** *application-dependent* constraints derived from deeper semantics of the data
  - Some checking can be done using `CHECK` keyword
  - For example, checking salary not being negative

Table "Employees"

id	name	dob	salary	*manager*
1	Alice	10-08-1988	7500	NULL
2	Bob	06-11-2001	**-4800**	3
3	Carol	25-02-1995	5500	1
4	Dave	18-06-1999	6000	NULL

Relational Model

History
Popularity
Model
Integrity
Key
Foreign Key
Considerations
- Limitations
- Practical

Considerations

Practical Considerations

- Integrity constraints are *optional* and not mandatory
  - It simply allows the check to be done by DBMS instead of application
  - In other words, no matter who uses the database, its entries will always be consistent with respect to the constraints *(assuming they cannot remove the constraints)*

- Integrity constraints may affect *performance*
  - May decrease performance due to additional checking
  - May improve performance due to creation of indexes that speeds up retrieval

Summary

Summary

Cheat Sheet
- Relational Model
- Constraints
Summary

Cheat Sheet

Relational Model

Table "Movies"

id	title	genre	opened	⋯
101	Aliens	action	1986	⋯
102	Logan	drama	2017	⋯
103	Heat	crime	1995	⋯
⋮	⋮	⋮	⋮	⋱

Informal Description

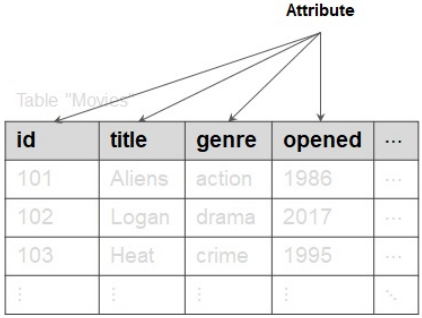
Term	Description
Attribute	column of a table
Domain	possible values
Attribute Value	element of a domain
Relation Schema	set of attributes + constraints + relation name
Tuple	row of a table
Relation	set of tuples
Cardinality	number of rows
Database Schema	set of relation schema
Database	set of relations

Summary

Cheat Sheet
- Relational Model
- Constraints
Summary

Cheat Sheet

Relational Model



Informal Description

> ***Attributes:*** Column of a table

Summary

Cheat Sheet
- Relational Model
- Constraints
Summary

Cheat Sheet

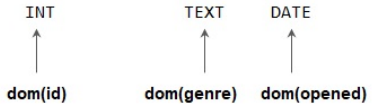
Relational Model

Informal Description

> ***Domain:*** Possible values

Table "Movies"

id		genre	opened	
101	Aliens	action	1986	⋯
102	Logan	drama	2017	⋯
103	Heat	crime	1995	⋯
⋮	⋮	⋮	⋮	⋱



Summary

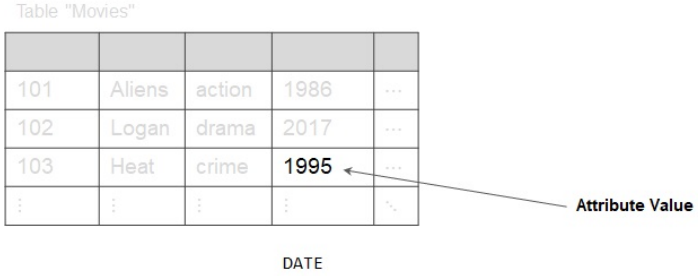
Cheat Sheet
- Relational Model
- Constraints
Summary

Cheat Sheet

Relational Model

Informal Description

> ***Attribute Values:*** Elements of a domain



Summary

Cheat Sheet
- Relational Model
- Constraints
Summary

Cheat Sheet

Relational Model

Full: *Movies(id: INT, title: TEXT, genre: TEXT, opened: DATE)*

Simplified: *Movies(id, title, genre, opened)*

Table "Movies"

id	title	genre	opened	···
101	Aliens	action	1986	···
102	Logan	drama	2017	···
103	Heat	crime	1995	···
⋮	⋮	⋮	⋮	⋱

← **Relation Schema**

Informal Description

> ***Relation Schema:*** Set of attributes + Constraints + Relation name

Summary

Cheat Sheet
- Relational Model
- Constraints
Summary

Cheat Sheet

Relational Model

Informal Description

> ***Tuple:*** Row of a table

Table "Movies"

101	Aliens	action	1986	···
102	Logan	drama	2017	···
103	Heat	crime	1995	···
⋮	⋮	⋮	⋮	⋱

Summary

Cheat Sheet
- Relational Model
- Constraints
Summary

Cheat Sheet

Relational Model

Informal Description

> ***Relation:*** Set of tuples

Table "Movies"

101	Aliens	action	1986	···
102	Logan	drama	2017	···
103	Heat	crime	1995	···
⋮	⋮	⋮	⋮	⋱

← Relation

Summary

Cheat Sheet
- Relational Model
- Constraints
Summary

Cheat Sheet

Relational Model

Informal Description

> ***Cardinality:*** Number of rows

Table "Movies"

101	Aliens	action	1986	···
102	Logan	drama	2017	···
103	Heat	crime	1995	···
⋮	⋮	⋮	⋮	⋱

Cardinality

# of rows

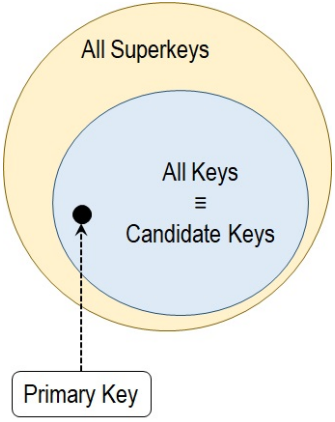
Summary

Cheat Sheet
- Relational Model
- Constraints
Summary

Cheat Sheet

Constraints

Term	Informal Description
Superkey	Set of attributes that *uniquely* identifies a tuple
Key	Minimal superkey *(no subset is a superkey)*
Candidate Keys	Set of **ALL** keys
Primary Key	One selected key
Prime Attribute	Attributes of a candidate key



DB vs DBS vs DBMS

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

[#]DB = Database ; DBS = Database Systems (not schema) ; DBMS = Database Management System

Summary

Cheat Sheet
Summary

Summary

Advantages of DBMS*

- Transactions with *ACID* properties
- Level of abstraction for *data independence*

Relational Model

- Unified representation of all data as relations *(i.e., tables)*
- *(Structural)* integrity constraints to specify restrictions on what constitutes correct/valid data

*For large-scale data management compared to "files only".

```
postgres=# exit
Press any key to continue . . .
```

Solutions

Exercise #1
Exercise #2
Exercise #3
Exercise #4
Exercise #5
Exercise #6

The Relational Model

Domain & Relation

Exercise #1

Assume a relation *R(A, B)* with

- *dom(A) = {x, y, z}*
- *dom(B) = {1, 2, 3, 4}*

Which tuples on the right **violate** the definition of relation *R*?

A	B	Row
x	4	*Row 1*
z	4	*Row 2*
NULL	2	*Row 3*
NULL	0	*Row 4*

	Choice	Comment	
A	Row 1	*NO:* $x \in dom(A)$ & $4 \in dom(B)$	✗
B	Row 2	*NO:* $z \in dom(A)$ & $4 \in dom(B)$	✗
C	Row 3	*NO:* $A = NULL$ & $2 \in dom(B)$	✗
D	Row 4	*YES:* $A = NULL$ & $0 \notin dom(B) \wedge$ not $NULL$	✓

Solutions

Exercise #1
Exercise #2
Exercise #3
Exercise #4
Exercise #5
Exercise #6

The Relational Model

Domain & Relation

Exercise #2

Assume a relation *R(A, B)* with

- *dom(A) = {x, y, z}*
- *dom(B) = {1, 2, 3, 4}*

Which tuples on the right **violate** the definition of relation *R*?

A	B	Row
NULL	NULL	*Row 1*
x	4	*Row 2*
x	y	*Row 3*
z	1	*Row 4*

	Choice	Comment	
A	Row 1	*NO: **A** = **NULL** & **B** = **NULL***	✗
B	Row 2	*NO: **x** ∈ dom(A) & **4** ∈ dom(B)*	✗
C	Row 3	*YES: **x** ∈ dom(A) & **y** ∉ dom(B) ∧ not **NULL***	✓
D	Row 4	*NO: **z** ∈ dom(A) & **1** ∈ dom(B)*	✗

Solutions

Exercise #1
Exercise #2
Exercise #3
Exercise #4
Exercise #5
Exercise #6

The Relational Model

Domain & Relation

Exercise #3

Assume a relation *R(A, B)* with

- *dom(A) = {x, y, z}*
- *dom(B) = {1, 2, 3, 4}*

Which tuples on the right **violate** the definition of relation *R*?

A	B	Row
x	4	*Row 1*
y	1	*Row 2*
x	1	*Row 3*
x	4	*Row 4*

	Choice	Comment	
A	Row 1	*LIKELY: Duplicate with Row 4*	✓
B	Row 2	*NO: $y \in dom(A)$ & $1 \in dom(B)$*	✗
C	Row 3	*NO: $x \in dom(A)$ & $1 \in dom(B)$*	✗
D	Row 4	*LIKELY: Duplicate with Row 1*	✓

Solutions

Exercise #1
Exercise #2
Exercise #3
Exercise #4
Exercise #5
Exercise #6

Key Constraints

Exercise #4

Consider a forum database with the following relation filled with many thousands of users: *Accounts(email:* ***TEXT****, password:* ***TEXT****, name:* ***TEXT****).*

Using reasonable guess, which subsets of attributes are **superkeys** of relation *Accounts*?

	Choice	Comment	
A	*{email}*	*LIKELY: email is likely unique*	✓
B	*{pwd}*	*UNLIKELY: two people may have the same password*	✗
C	*{name}*	*UNLIKELY: unless this is username*	✗
D	*{email, pwd}*	*LIKELY: it contains email*	✓
E	*{email, name}*	*LIKELY: it contains email*	✓
F	*{pwd, name}*	*UNLIKELY: does not contain email*	✗
G	*{email, pwd, name}*	*LIKELY: it contains email*	✓

Solutions

Exercise #1
Exercise #2
Exercise #3
Exercise #4
Exercise #5
Exercise #6

Key Constraints

Exercise #5

Consider a forum database with the following relation filled with many thousands of users: *Accounts(email:* ***TEXT****, password:* ***TEXT****, name:* ***TEXT****).*

Using reasonable guess, which subsets of attributes are **keys** of relation *Accounts*?

	Choice	Comment	
A	*{email}*	*LIKELY: email is likely unique*	✓
B	*{pwd}*	*UNLIKELY: two people may have the same password*	✗
C	*{name}*	*UNLIKELY: unless this is username*	✗
D	*{email, pwd}*	*UNLIKELY: it contains email*	✗
E	*{email, name}*	*UNLIKELY: it contains email*	✗
F	*{pwd, name}*	*UNLIKELY: does not contain email*	✗
G	*{email, pwd, name}*	*UNLIKELY: it contains email*	✗

Solutions

Exercise #1
Exercise #2
Exercise #3
Exercise #4
Exercise #5
Exercise #6

Foreign Key Constraints

Exercise #6

Consider two tables $R(\underline{A}, B)$ and $S(\underline{C}, D)$ with $S.D \rightsquigarrow R.A$.

Which tuples in Table "S **violate** the **foreign key** constraint?

Table "R"

A	B
x	4
z	2
y	NULL

Table "S"

C	D
a	w
b	x
c	y
d	NULL

Row
Row 1
Row 2
Row 3
Row 4

	Choice	Comment	
A	Row 1	*YES: **w** not in Table "R"*	✓
B	Row 2	*NO: **x** is in Table "R"*	✗
C	Row 3	*NO: **y** is in Table "R"*	✗
D	Row 4	*NO: **NULL** is allowed*	✗