Introduction to Database Systems
IDBS – Spring 2024

Functional Dependencies

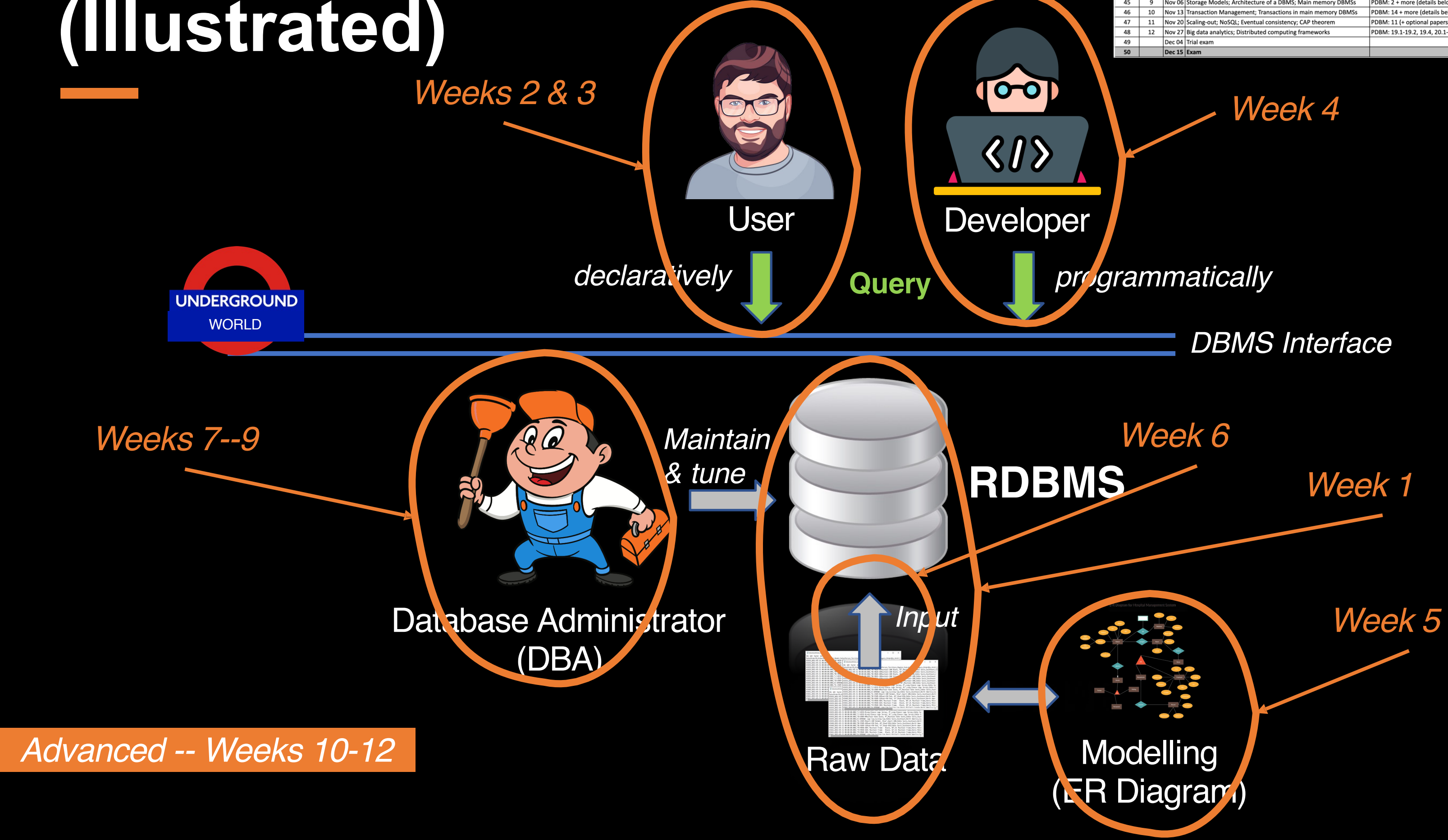
Readings
PDBM 6.2

Eleni Tzirita Zacharatou

Based on slides by Andy Pavlo

Course Schedule (Illustrated)

Week	Lecture	Date	Lecture Topic	Readings	Exercises / Homeworks	Notes
35	1	Aug 28	Introduction; Course overview; Relational model	PDBM: 1, 6.1, 7.1-7.2	Exercise L1	Lecturer: Eleni Tzirita Zacharatou
36	2	Sep 04	SQL DDL/DML; Basic SQL queries; Joins; Aggregations	PDBM: 7.3	Exercise L2	Lecturer: Eleni Tzirita Zacharatou
37	3	Sep 11	Complex SQL queries; Subqueries; Views	PDBM: 7.3 - 7.4	Homework 1 (Sep 2 - 18)	Lecturer: Omar Shahbaz Khan
38	4	Sep 18	Triggers; Transactions; Using SQL from Python	PDBM: 9.2, 14.1, 14.2.1, 14.5 + more (details below)	[BSc only] Exercise L4B	Lecturer: Omar Shahbaz Khan
39		Sep 25	**No lecture - only exercise session for MSc students**		[MSc only] Exercise L4M	
40	5	Oct 02	ER modeling	PDBM: 3.0-3.3, 6.3-6.4	Exercise L5	Lecturer: Omar Shahbaz Khan
41	6	Oct 9	Normalization	PDBM: 6.2-6.4	Homework 2 (Oct 1 - 16)	Lecturer: Eleni Tzirita Zacharatou
42		Oct 16	**Semester Break**			
43	7	Oct 23	Storage hierarchy; Physical database design; Indexing	PDBM: 12.1, 12.2, 12.3.1 - 12.3.7 + more (details below)	Exercise L7	Lecturer: Eleni Tzirita Zacharatou
44	8	Oct 30	B-trees; Performance tuning; Access methods; Join implementation	PDBM: 12.3.8, 13.1	Exercise L8	Lecturer: Omar Shahbaz Khan
45	9	Nov 06	Storage Models; Architecture of a DBMS; Main memory DBMSs	PDBM: 2 + more (details below)	Homework 3 (Oct 28 - Nov 13)	Lecturer: Eleni Tzirita Zacharatou
46	10	Nov 13	Transaction Management; Transactions in main memory DBMSs	PDBM: 14 + more (details below)	Old exercises/exams/homeworks	Lecturer: Omar Shahbaz Khan
47	11	Nov 20	Scaling-out; NoSQL; Eventual consistency; CAP theorem	PDBM: 11 (+ optional papers)	Old exercises/exams/homeworks	Lecturer: Omar Shahbaz Khan
48	12	Nov 27	Big data analytics; Distributed computing frameworks	PDBM: 19.1-19.2, 19.4, 20.1-20.3	Homework 4 (Nov 18 - Dec 4)	Lecturer: Eleni Tzirita Zacharatou + guest
49		Dec 04	Trial exam			Online
50		**Dec 15**	**Exam**			



DATABASE DESIGN

How do we design a “good” database schema?

We want to ensure the integrity of the data.

We also want to get good performance.

EXAMPLE DATABASE

student(sid,cid,room,grade,name,address)

sid	cid	room	grade	name	address
123	15-445	GHC 6115	A	Andy	Pittsburgh
456	15-721	GHC 8102	B	Tupac	Los Angeles
789	15-445	GHC 6115	A	Obama	Chicago
012	15-445	GHC 6115	C	Waka Flocka	Atlanta
789	15-721	GHC 8102	A	Obama	Chicago

EXAMPLE DATABASE

student(<u>sid</u>,<u>cid</u>,room,grade,name,address)

sid	cid	room	grade	name	address
123	15-445	GHC 6115	A	Andy	Pittsburgh
456	15-721	GHC 8102	B	Tupac	Los Angeles
789	15-445	GHC 6115	A	Obama	Chicago
012	15-445	GHC 6115	C	Waka Flocka	Atlanta
789	15-721	GHC 8102	A	Obama	Chicago

REDUNDANCY PROBLEMS

Update Anomalies

→ If the room number changes, we need to make sure that we change all students records.

Insert Anomalies

→ May not be possible to add a student unless they're enrolled in a course.

Delete Anomalies

→ If all the students enrolled in a course are deleted, then we lose the room number.

EXAMPLE DATABASE

student(<u>sid</u>,name,address)

sid	name	address
123	Andy	Pittsburgh
456	Tupac	Los Angeles
789	Obama	Chicago
012	Waka Flocka	Atlanta

courses(<u>sid</u>,<u>cid</u>,grade)

sid	cid	grade
123	15-415	A
456	15-721	B
789	15-415	A
012	15-415	C
789	15-721	A

rooms(<u>cid</u>,room)

cid	room
15-415	GHC 6115
15-721	GHC 8102

Why this decomposition is better and how to find it.

THIS VIDEO

Functional Dependencies: Constraints between attributes

FUNCTIONAL DEPENDENCIES

A functional dependency (FD) is a form of a constraint.

Part of a relation's schema to define a valid instance.

Definition: $X \rightarrow Y$

→ The value of X functionally defines the value of Y.

FUNCTIONAL DEPENDENCIES

Formal Definition:

→ $X \rightarrow Y \Rightarrow (t_1[x]=t_2[x] \Rightarrow t_1[y]=t_2[y])$

If two tuples (t_1, t_2) agree on the **X** attribute, then they must agree on the **Y** attribute too.

R1(sid,name,address)

sid	name	address
123	Andy	Pittsburgh
456	Tupac	Los Angeles
789	Obama	Chicago
012	Waka Flocka	Atlanta

FUNCTIONAL DEPENDENCIES

Formal Definition:

→ $X \rightarrow Y \Rightarrow (t_1[x]=t_2[x] \Rightarrow t_1[y]=t_2[y])$

If two tuples (t_1, t_2) agree on the X attribute, then they must agree on the Y attribute too.

R1(sid,name,address)

sid	name	address
123	Andy	Pittsburgh
456	Tupac	Los Angeles
789	Obama	Chicago
012	Waka Flocka	Atlanta

X Y

✓ sid→name

FUNCTIONAL DEPENDENCIES

FD is a constraint that allows instances for which the FD holds.

You can check if an FD is violated by an instance, but you cannot prove that an FD is part of the schema using an instance.

R1(sid,name,address)

sid	name	address
123	Andy	Pittsburgh
456	Tupac	Los Angeles
789	Obama	Chicago
012	Waka Flocka	Atlanta
555	Andy	Providence

??? name→address

FUNCTIONAL DEPENDENCIES

Two FDs **X→Y** and **X→Z** can be written in shorthand as **X→YZ**.

But **XY→Z** is not the same as the two FDs **X→Z** and **Y→Z**.

WHY SHOULD I CARE?

FDs seem important, but what can we actually do with them?

They allow us to decide whether a database design is correct.

→ Note that this different then the question of whether it's a good idea for performance…

IMPLIED DEPENDENCIES

student(sid,cid,room,grade,name,address)

sid	cid	room	grade	name	address
123	15-445	GHC 6115	A	Andy	Pittsburgh
456	15-721	GHC 8102	B	Tupac	Los Angeles
789	15-445	GHC 6115	A	Obama	Chicago
012	15-445	GHC 6115	A	Waka Flocka	Atlanta

Provided FDs

sid → name,address
sid,cid → grade

Implied FDs

sid,cid → grade
sid,cid → sid
sid,cid → cid

IMPLIED DEPENDENCIES

Given a set of FDs $\{f_1,\ldots,f_n\}$, how do we decide whether FD g holds?

Compute the closure using Armstrong's Axioms
→ This is the set of all implied FDs.

ARMSTRONG'S AXIOMS

Reflexivity:

$\longrightarrow$ $X \supseteq Y \Rightarrow X \to Y$

- This is a **trivial functional dependency**

Augmentation:

$\longrightarrow$ $X \to Y \Rightarrow XZ \to YZ$

Transitivity:

$\longrightarrow$ $(X \to Y) \wedge (Y \to Z) \Rightarrow X \to Z$

Union:

$\longrightarrow$ $(X \to Y) \wedge (X \to Z) \Rightarrow X \to YZ$

Decomposition:

$\longrightarrow$ $X \to YZ \Rightarrow (X \to Y) \wedge (X \to Z)$

Pseudo-transitivity:

$\longrightarrow$ $(X \to Y) \wedge (YW \to Z) \Rightarrow XW \to Z$

CLOSURES

Given a set **F** of FDs $\{f_1,\ldots,f_n\}$, we define the closure **F+** is the set of all implied FDs.

`student(sid,cid,room,grade,name,address)`

F

```
sid,cid → grade
sid → name,address
```

CLOSURES

Given a set F of FDs $\{f_1, \ldots, f_n\}$, we define the closure F+ is the set of all implied FDs.

student(<u>sid</u>,<u>cid</u>,room,grade,name,address)

F	sid,cid → grade	F+
F	sid → name,address	F+
Reflexivity →	sid,name → sid	F+
Decomposition →	sid → name	F+
Decomposition →	sid → address	F+
Augmentation →	sid,grade → name,grade	F+
Union →	sid,cid → sid,grade	F+
Transitivity →	sid,cid → grade,name	F+
	…	

UNAVOIDABLE AND REDUNDANT FDs

- FDs with a superkey on the left-hand side are **unavoidable**
  - If A is a candidate key for a relation, then clearly $A \rightarrow B$ for any attribute B
  - Similarly, if $\{A_1, A_2\}$ forms a superkey, we have $A_1A_2 \rightarrow B$ for any B
  - Etc

- FDs that can be computed from others are **redundant**
  - They do not require decomposition!

- Only decompose when not trivial, unavoidable, or redundant

CONCLUSION

Functional dependencies are simple to understand.

They will allow us to reason about schema decompositions.

- **This week's lecture**