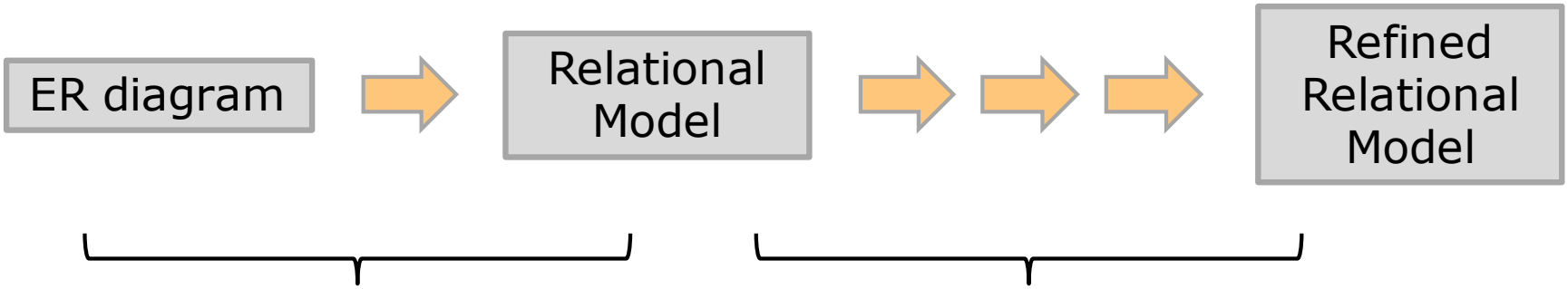
Normalization 1: Functional dependencies

Database design process



we discussed how to create an ER design, and how to map it to a relational design

now we discuss how to refine the initial relational design to a final one

Is this a good schema?

inst_dept(ID, name, salary,
dept_name, building, budget)

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

This schema obtained by joining 'instructor' and 'department'.

(table from Database System Concepts, Silbershatz et al)

Pros and Cons

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Good news:

Fewer joins needed in queries.

Bad news 1:

If Physics budget changes, it must be changed in multiple places.

Bad news 2:

How to create a new department if no instructors yet for the department?

Bad news 3:

If the last instructor in a department leaves, other dept. info is lost

(table from Database System Concepts, Silbershatz et al)

Improving a schema

If we were given a schema like this, would we know how to improve it?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

It might be hard to spot these kinds of things

(table from Database System Concepts, Silbershatz et al)

Improving a schema

To improve the schema, it would be helpful if we knew this:

dept_name → budget

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

With this functional dependency above, we can identify some redundancy.

(table from Database System Concepts, Silbershatz et al)

Decomposing a schema

If these functional dependencies were given:

dept_name → budget

dept_name → building

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

we could see the problem, and could fix it by splitting the schema in two

How to do this systematically?

(table from Database System Concepts, Silbershatz et al)

Summarizing

- Redundancy in relational schemas is bad
  - storage waste
  - leads to inconsistencies
- It's hard to spot redundancy by looking at a table, especially if it's big
- If data designer supplies functional dependencies, we can spot redundancies
- It can be fixed by splitting schemas – but how exactly?

Functional dependencies

- a kind of integrity constraint
- examples:
  - students are uniquely identified by their ID
  - each instructor has only one name
  - each instructor is associated with just one department
  - each department has only one budget value
- dept_name → budget
- a table satisfies dept_name → budget if:
  - if two rows have the same dept_name, they have the same budget

Functional dependencies: definition

- suppose r(R) is a relational schema (R is the set of attributes in schema r)
- a functional dependency has the form

$$X \rightarrow Y$$

  where *X, Y* are subsets of R
- an instance of r(R) satisfies $X \rightarrow Y$ if:
  - all pairs of rows with the same values for the X attributes have the same values for the Y attributes

Examples

- students are uniquely identified by their ID

  ID → name

- each instructor has only one name

  ID → name

- each instructor is associated with just one department

  ?

- each department has only one budget value

  ?

Functional dependencies and keys

Question: Are keys just a kind of functional dependency?

Examples:

- department(dept_name, building, budget)
- section(course_id, sec_id, semester, year, building, room_number, time_slot_id)

Remember:
- you can't find functional dependencies by looking at a table

All the FDs of a schema

When we talk about the functional dependencies of a schema, we mean:

- the FDs we may have explicitly identified

+

- "trivial" FDs, like $X \rightarrow Y$ and $X, Y \rightarrow Y$

+

- FDs derivable from other FDs of the schema

"derivable" means by one of these rules:

1) if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$
2) if $X \rightarrow Y$ then $X \cup Z \rightarrow Y \cup Z$

So if ID → name, and name → dept_name, then ID → dept_name

Summary

- We want relational schemas that:
  - don’t have redundancy
  - can be queried efficiently
- Redundancy is addressed by splitting a relation into smaller relations
- To do this in a systematic way, we use functional dependencies
- Functional dependencies are integrity constraints
- Keys are a special kind of functional dependency