Database System Tutorial 1

Database TA Team
The University of Auckland

THE UNIVERSITY OF
AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND

Tutorial Structure

- Covers additional material if needed
- Questions that you can try
- Answering questions

(Top-left corner: QR code of this slide)

Compare SQL Joins

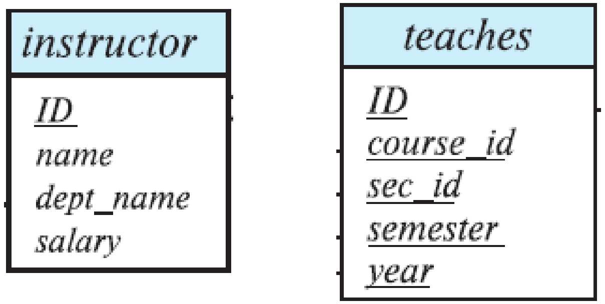
- Cartesian Product (×)
- Theta Join ($\bowtie_\theta$)
- Natural Join ($\bowtie$)
- Key Differences

Cartesian Product (×)

- Join with no condition
  - Yields every combination of values
  - All attributes concatenated

instructor X *teaches*

instructor.ID



instructor

ID
name
dept_name
salary

teaches

ID
course_id
sec_id
semester
year

teaches.id

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018

Theta Join ($\bowtie_{\theta}$)

- Cartesian Product, but with extra predicate

$$r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$$

$$\sigma_{\text{instructor.ID=teaches.ID}}(\text{instructor} \times \text{teaches})$$

$$=$$

$$\text{instructor} \bowtie_{\text{instructor.ID=teaches.ID}} \text{teaches}$$

Natural Join (⋈)

- Cartesian Product, but automatically uses = for each column with the same name

$$\text{instructor} \bowtie \text{teaches}$$

$$=$$

$$\sigma_{\text{building}=303 \wedge \text{year}=2024} (\text{instructor} \times \text{teaches} \times \text{department})$$

Key differences

- Cartesian Product (×): No condition, pairs all
- Theta Join ($\bowtie_{\theta}$): plus predicate
- Natural Join ($\bowtie$): automatically join matching cols

Exercises

- *branch*(*branch name*, *branch city, assets*)
- *customer* (*ID*, *customer name*, *customer street, customer city*)
- *loan* (*loan number*, *branch name, amount*)
- *borrower* (*ID*, *loan number*)
- *account* (*account number*, *branch name, balance*)
- *depositor* (*ID*, *account number*)

1. Consider this bank database. Assume that branch names and customer names uniquely identify branches and customers, but loans and accounts can be associated with more than one customer.

- What are the appropriate primary keys?
- Given your choice of primary keys, identify appropriate foreign keys.

Exercises

- *branch*(*branch name*, *branch city, assets*)
- *customer* (*ID*, *customer name*, *customer street, customer city*)
- *loan* (*loan number*, *branch name, amount*)
- *borrower* (*ID*, *loan number*)
- *account* (*account number*, *branch name, balance*)
- *depositor* (*ID*, *account number*)

2. Consider this bank database. Give an expression in the relational algebra for each of the following:

- Find each loan number with a loan amount greater than $10000.
- Find the ID of each depositor who has an account with a balance greater than $6000.
- Find the ID of each depositor who has an account with a balance greater than $6000 at the “Uptown” branch.

Exercises

3. Consider the university database (see Appendix).

Write the following queries in relational algebra:

- Find the ID and name of each instructor in the Physics department.
- Find the ID and name of each instructor in a department located in the building “Watson”.
- Find the ID and name of each student who has taken at least one course in the “Comp. Sci.” department.
- Find the ID and name of each student who has taken at least one course section in the year 2018.
- Find the ID and name of each student who has not taken any course section in the year 2018.

Exercises

4. Write the following queries in SQL, using the university schema:

- Find the titles of courses in the Comp. Sci. department that have 3 credits.
- Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.
- Find the highest salary of any instructor.
- Find all instructors earning the highest salary (there may be more than one with the same salary).
- Find the enrollment of each section that was offered in Fall 2017.
- Find the maximum enrollment, across all sections, in Fall 2017.
- Find the sections that had the maximum enrollment in Fall 2017.

Appendix: University Database

classroom(*building*, *room number*, *capacity*)

department(*dept_name*, *building*, *budget*)

course(*course_id*, *title*, *dept_name*, *credits*)

instructor(*ID*, *name*, *dept_name*, *salary*)

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room number*, *time_slot_id*)

teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)

student(*ID*, *name*, *dept_name*, *tot cred*)

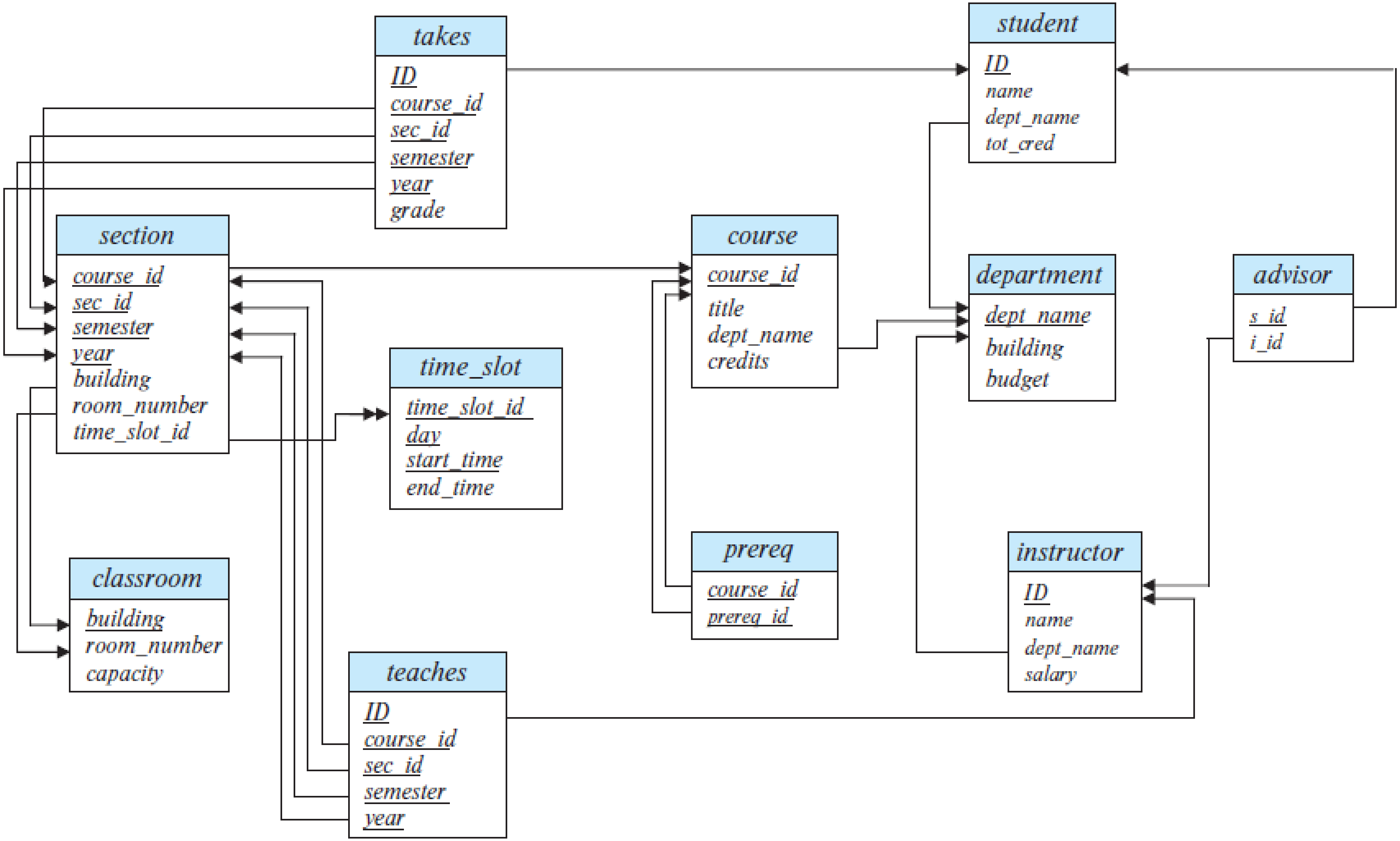
takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)

advisor(*s_ID*, *i_ID*)

time_slot(*time_slot_id*, *day*, *start time*, *end time*)

prereq(*course_id*, *prereq_id*)

Appendix: University Database



takes
ID
course_id
sec_id
semester
year
grade

student
ID
name
dept_name
tot_cred

section
course_id
sec_id
semester
year
building
room_number
time_slot_id

course
course_id
title
dept_name
credits

department
dept_name
building
budget

advisor
s_id
i_id

time_slot
time_slot_id
day
start_time
end_time

classroom
building
room_number
capacity

prereq
course_id
prereq_id

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

Appendix: University Database

https://www.db-book.com/university-lab-dir/sqljs.html

<- Online SQL interpreter

University database loaded

FIN

Any questions?