SQL

CISC637, Lecture #3

Ben Carterette

Homework 1 Notes

- Assignment due Feb 24th at midnight
- Remember to use `engine=innodb` with *all* your `CREATE TABLE` statements
  - Ensure that MySQL will enforce foreign key constraints
- For #3, don't copy sample data from the book
  - Come up with your own
- Use mysqldump to export your database to a text file for submission on Sakai
  - Be sure you have completed #1 and #3 before exporting
- Work by yourself

SQL Queries

- Basic form of a SQL query:

```
SELECT [DISTINCT] fields
FROM tables
WHERE qualification
```

- *fields* is a comma-separated list of attributes/fields
- *tables* is a comma-separated list of tables to get data from
- *qualification* is a Boolean logic (T/F; AND/OR/NOT) sentence about fields in the tables
- DISTINCT is an optional keyword for dropping duplicates

- Result of query is an anonymous table (a relation with no name)

SELECT

```
SELECT field1, field2, …
FROM Table
```

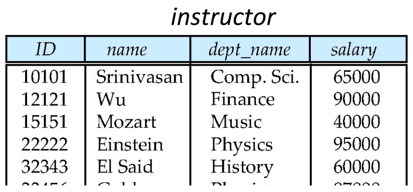
- Use `SELECT *` to get all fields

- Can include arithmetic expressions or functions
  - `SELECT 10+field1 FROM Table`
  - `SELECT field1*field2 FROM Table`
  - `SELECT max(field1) FROM Table`
  - `SELECT count(DISTINCT field1) FROM Table`

- String functions can be applied to character fields
  - `SELECT upper(field1) FROM Table`
  - `SELECT substring(field1, 0, 4) FROM Table`
  - `SELECT concat(field1, " ", field2) FROM Table`

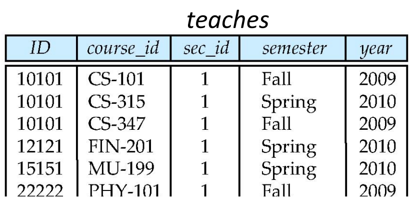
SELECT Examples

```
SELECT field1, field2, … FROM Table
```



instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
[illegible]	[illegible]	[illegible]	[illegible]



teaches

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009

What are the monthly salaries of instructors?

What unique course IDs are taught?

WHERE

```
SELECT field1, field2, …
FROM Table
WHERE condition
```

- `condition` is a Boolean logical sentence of comparisons on fields and constants, linked with ANDs, ORs, NOTs
  - Usually will be something like `field1 = value AND field2 = value`
- Some useful keywords to use in WHERE clauses:
  - `field1 BETWEEN x AND y`
    - equivalent to `field1 >= x AND field1 <= y`
  - `field1 LIKE 'str%'`
    - % indicates wildcard match
    - match strings starting with "str"
  - `field1 IS NULL` / `field1 IS NOT NULL`

WHERE Examples

```
SELECT field1, field2, … FROM Table WHERE condition
```

instructor

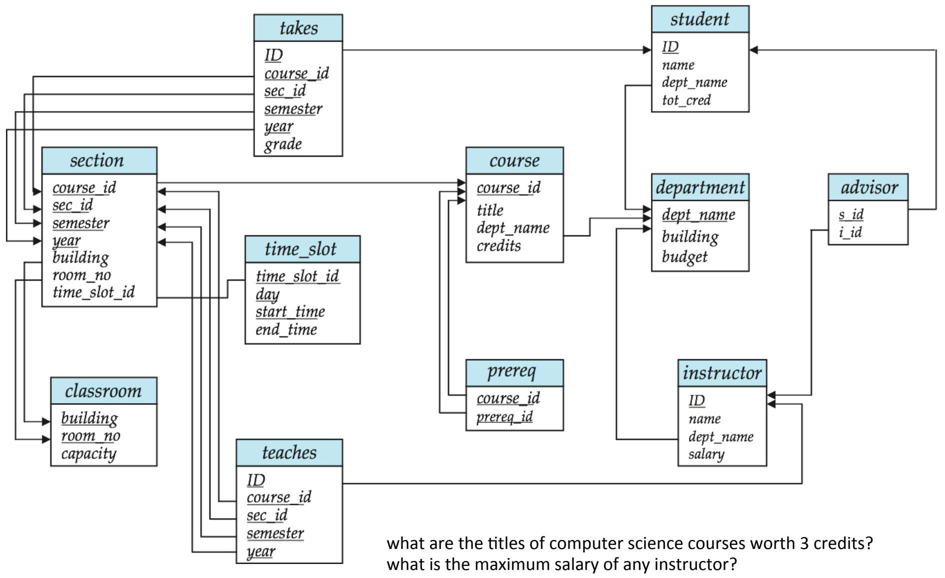
ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000

teaches

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009

What are the monthly salaries of professors of CS?

What unique courses were taught in either the Spring of 2010 or the Fall of 2011?

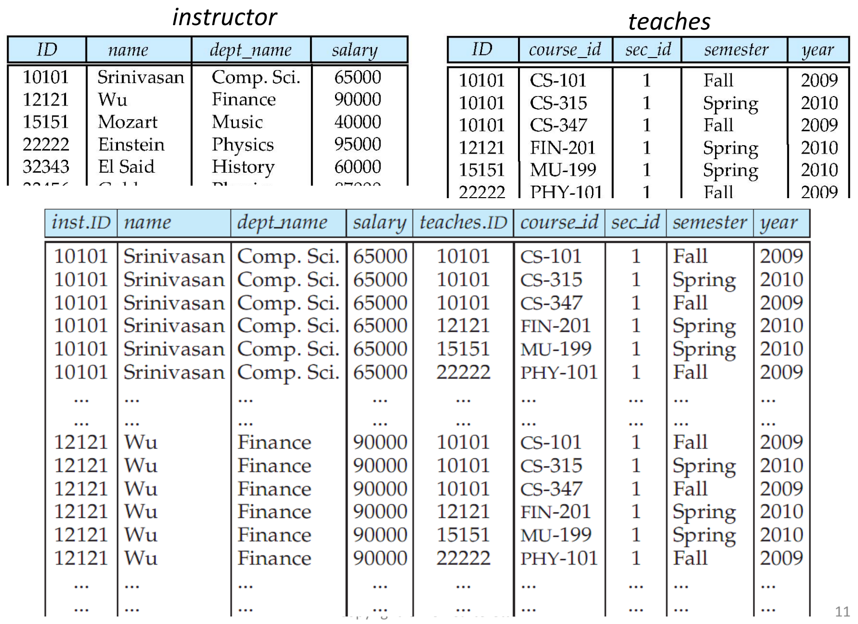


what are the titles of computer science courses worth 3 credits?
what is the maximum salary of any instructor?
which departments have budgets of more than $1,000,000?
how many distinct courses are being offered in the Spring of 2015?
how many distinct sections are being offered in the Spring of 2015?

FROM

```
SELECT *
FROM Table1 T1, Table2 T2
```

- This computes the **Cartesian product** of `Table1` and `Table2`
  - Also known as cross-product
  - Each row in `Table1` concatenated with every row in `Table2`
- Normally you would never need a Cartesian product, but it is useful to understand



instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000

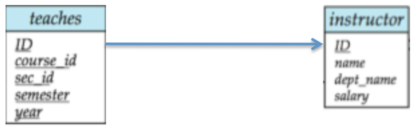
teaches

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009

inst.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2009
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2010
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2010
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

Joins

- Get data from two or more tables linked by common fields

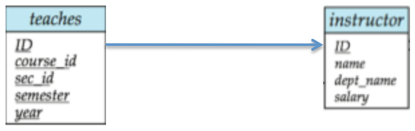


```
SELECT *
FROM Table1, Table2
WHERE Table1.foreignKey = Table2.primaryKey
```

natural join

Joins

- Get data from two or more tables linked by common fields



```
SELECT *
FROM Table1 NATURAL JOIN Table2
```

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000

teaches

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009

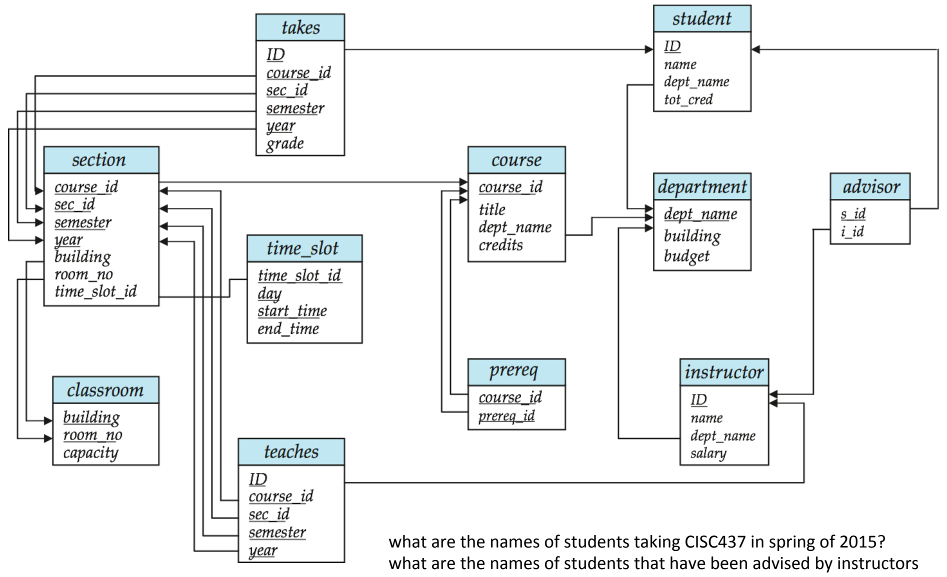
ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010

Don't Do This

- Let's say we want to get names of instructors that have taught CISC437

```
SELECT name
FROM instructor
WHERE ID IN (SELECT ID FROM teaches WHERE course_id = 'CISC437')
```

- It gives the right result, but it is a bad query
- Use joins!!!
  - DBMSs are optimized to do joins
  - Joins are the single most important thing to know how to do in a database



what are the names of students taking CISC437 in spring of 2015?
what are the names of students that have been advised by instructors named Wu?
what are the course_ids that are prerequisites for the course titled "Database Systems"?
what buildings are CIS courses taught in?

NATURAL JOIN

- My recommendation: never use NATURAL JOIN keyword
  - Always specify join conditions in your WHERE clause
- Experiment on Student (ID), Instructor (ID), Advisor (s_id, i_id)
  - Student NATURAL JOIN Instructor:
    - empty set (correct result since IDs have different meaning, but unexpected)
  - Student NATURAL JOIN Advisor:
    - Cartesian product (incorrect result)
  - Instructor NATURAL JOIN Advisor:
    - Cartesian product (incorrect result)
  - Student NATURAL JOIN Advisor NATURAL JOIN Instructor
    - empty set (incorrect result)
- Conclusion: it isn't joining according to foreign key definitions. It isn't joining according to strict field name match. What is it doing?
  - field name match *unless* both fields are primary keys of their respective tables
  - revert to Cartesian product if no field names match *or* matching field names are both primary keys