F24-W4111-03: Introduction to Databases: Homework 1, Part B

Nathan Gong

Submission Instructions

Note to TAs: Please complete this information, create GradeScope entries, etc.

Environment Setup

This section tests your environment for HW1B.

If you successfuly completed HW0, you should not have any problems.

Please make sure you set your MySQL user id and password correctly.

```
In [1]: # %pip install pandas
        import pandas
```

```
In [2]: import sqlalchemy
```

```
In [3]: import pymysql
```

```
In [4]: import json
```

```
In [5]: %load_ext sql
```

```
In [6]: %sql mysql+pymysql://root:dbuserdbuser@localhost
```

```
In [7]: engine = sqlalchemy.create_engine("mysql+pymysql://root:dbuserdbuser@localhost")
```

```
In [8]: from IPython.display import Image
```

Entity Relationship Modeling

Top-Down Modeling

The ability to prduce an ER diagram from a "human" description of the data model is an import skill. In this process, you may have to make and document assumptions or explain decisions. There is no single, correct answer. As long as your assumptions and decisions are reasonable, and your model accurately reflects requirements and decisions, your model answer is "correct."

In this scenario, there are four entity types/entity sets:

1. `Person(id, last_name, first_name, middle_name, created_timestamp, last_modified_timestamp)` : Basic information about a person. The type has properties/attributes:
   - `id` uniquely indentifies the `Person`
   - `last_name`
   - `first_name`
   - `middle_name`
   - `created_timestamp` : When the the entity was created for the first time.
   - `last_modified_timestamp` : The last time the entity's information changed.
2. `Contact_Information(contact_type, contact_value)` : Represents a mechanism for contacting a person.
   - `id` : A unique ID for the `Contact_Information` .
   - `contact_type` : Indicates the type of contact, e.g. "primary phone," "email," etc.
   - `contact_value:` The value for the contact. This is simply a text string for both types of contact. For example, "bilbo.baggins@shire.org" or "+1 212-555-1212."
3. `Order(id, product_name, order_date, description)` : Represents someone having placed an order to purchase something. Order has the properties:
   - `id` : Uniquely identifies the `Order`
   - `product_name` : The name of the product, e.g. "Strawbery Poptarts," "Cross Pen."
   - `order_date` : The date the order was placed
   - `description` : A text description of the order
4. `Comment(id, comment, comment_timestamp)` : Represent a user's comment on an order. Comment has three properties:
   - `id` : Uniquely identifies the `Comment`
   - `comment` : Text of the comment
   - `comment_timestamp` : Timestamp when the comment was made.

The model has the following relationships/entity sets:

- `Person-Comment` is a relationship the represents the fact that the `Person` made the `Comment` . A `Person` may make many `Comments` , but a `Comment` is made by exactly one user.
- `Order-Comment` associates the `Comment` with the `Order.` There may be many `Comments` on an `Order` but a `Comment` has one `Order.`
- `Person-Contact-Info` is between `Person` and `Contact-Info.` A `Person` may have multiple `Contact-Info` entries. A `Contact-Info` relates to exactly one `Person` .

The model must represent the fact that `Contact-Info` is valid between a start timestamp and end timestamp.

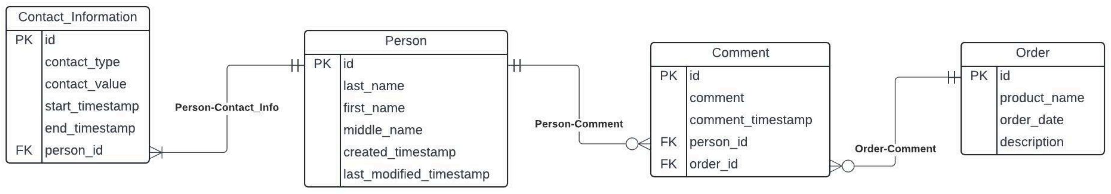
The system never deletes any information.

You must create a Crow's Foot Notation *logical model* that is your model that satisfies the requirements. You may have to add unspecified attributes to entity types. You can add comments and notes.

Show your diagram below. You can add notes to your diagram or add explanatory text. You can take a screenshot of your diagram and include below. The "Implement ER Diagram" question has an example of embedding an image in the notebook.

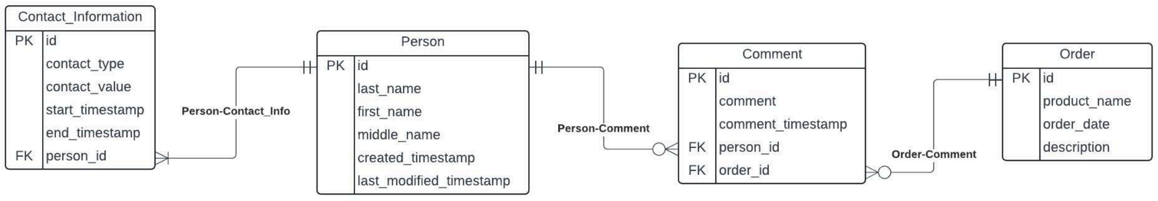
There is no single correct answer.

Diagram:



```
In [9]: Image("ER_logical.jpeg")
```

Out[9]:

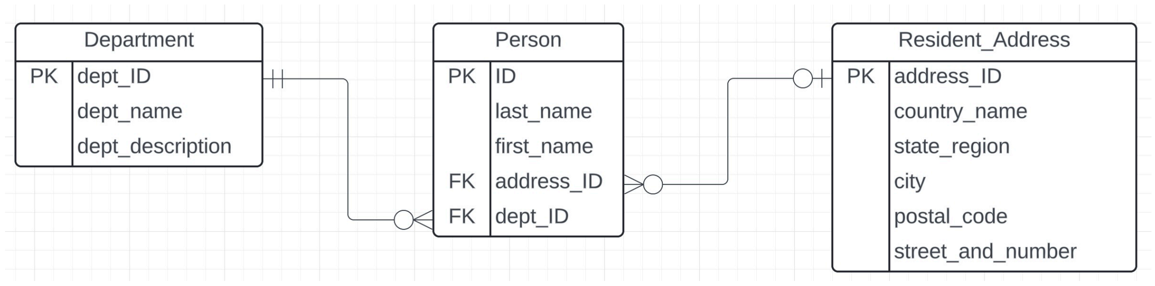


Comments:

- `Comment` contains a `person_id` which is a foreign key that relates it to a `Person`.
- Assume that a `Person` may make 0 or many `Comment` s.
- `Comment` contains an `order_id` which is a foreign key that relates it to an `Order`.
- Assume that an `Order` may have 0 or many `Comment` s.
- `Contact_Information` contains a `person_id` which is a foreign key that relates it to a `Person`.
- `Contact_Information` contains a `start_timestamp` and an `end_timestamp` which represent the timestamps that it is valid between.

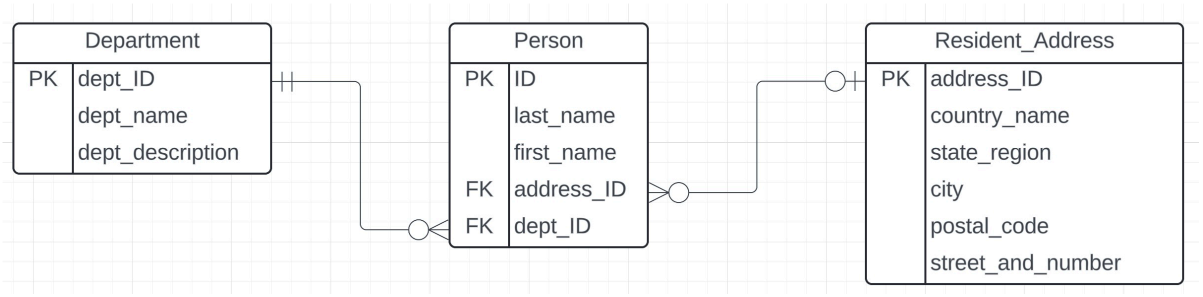
- Assume that a `Person` must have at least 1 `Contact_Information`.

Implement ER Diagram



In [10]:
```
Image("er-to-sql.jpg")
```

Out[10]:



Write SQL DDL that creates the tables and relationships in the preceding diagram

You can pick `VARCHAR(32)` for the type of each column.

You must specify keys and foreign keys.

Create a new database that you name `hw1b_<uni>` and replace `<uni>` with your UNI. For example, mine would be `hw1b_dff9.`

You must enter and successfully execute your SQL in the code cell below.

In [11]:
```
%%sql

/* Your create and alter table statements. */

CREATE DATABASE IF NOT EXISTS hw1b_ng2695;
USE hw1b_ng2695;

CREATE TABLE IF NOT EXISTS Department (
    dept_ID VARCHAR(32),
    dept_name VARCHAR(32),
    dept_description VARCHAR(32),
    PRIMARY KEY (dept_ID)
);

CREATE TABLE IF NOT EXISTS Resident_Address (
    address_ID VARCHAR(32),
    country_name VARCHAR(32),
    state_region VARCHAR(32),
    city VARCHAR(32),
    postal_code VARCHAR(32),
    street_and_number VARCHAR(32),
    PRIMARY KEY(address_ID)
);

CREATE TABLE IF NOT EXISTS Person (
    ID VARCHAR(32),
    last_name VARCHAR(32),
    first_name VARCHAR(32),
    address_ID VARCHAR(32),
    dept_ID VARCHAR(32),
    PRIMARY KEY (ID),
    FOREIGN KEY (address_ID) REFERENCES Resident_Address(address_ID),
    FOREIGN KEY (dept_ID) REFERENCES Department(dept_ID)
);
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

```
Out[11]: []
```

Relational Algebra

You will use the Relax calulator and the schema associated with the text book for this question.

https://dbis-uibk.github.io/relax/calc/gist/4f7866c17624ca9dfa85ed2482078be8/relax-silberschatz-english.txt/0

Problem 1

Write a relational algebra expression that produces a result table with the following format:

`(student_id, student_name, course_title, course_id, sec_id, semester, year, instructor_id, instructor_name)`

- `student_id` is a student's ID (`student.ID`)
- `student_name` is a student's name (`student.name`)
- `course_title` (`course.title`)
- The following columns are common to `section, takes, teaches:`
  - `course_id`
  - `sec_id`
  - `semester`
  - `year`

- `instructor_id` is an instructor's ID (`instructor.ID`)
- `instructor_name` is an instructor's name (`instructor.name`)

This derived relation represents student that took a section and the instructor taught the section.

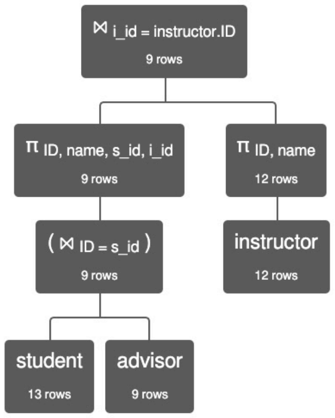
Cut and paste your query in the markdown cell below.

Past relational algebra here. The following is an example of pasting a relational algebra expression. Replace the following with your expression.

```
/*
This query produces students and their advisors.
(student ⨝ ID=s_id advisor) ⨝ i_id=instructor.ID instructor
*/


π student.ID->student_id, student.name->student_name, course.title-
>course_title, teaches.course_id->course_id, teaches.sec_id-
>sec_id, teaches.semester->semester, teaches.year->year,
instructor.ID->instructor_id, instructor.name->instructor_name
((((student ⨝ student.ID=takes.ID takes) ⨝
takes.course_id=course.course_id course) ⨝
course.course_id=teaches.course_id teaches) ⨝
teaches.ID=instructor.ID instructor)
```

Execute your query on the Relax calculator and show an image of the first page of your result below. The following shows an example of the format of the answer applied to the above example.



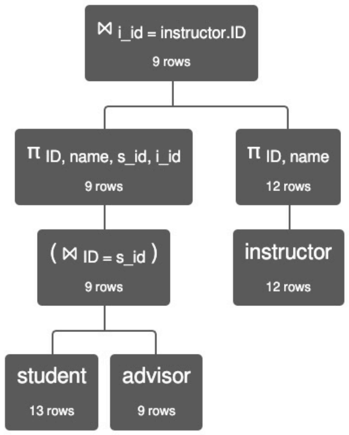
$(\pi_{\text{ID, name, s\_id, i\_id}} (\text{student} \bowtie_{\text{ID = s\_id}} \text{advisor})) \bowtie_{\text{i\_id = instructor.ID}} \pi_{\text{ID, name}} (\text{instructor})$

Execution time: 2 ms

student.ID	student.name	advisor.s_id	advisor.i_id	instructor.ID	instructor.name
128	'Zhang'	128	45565	45565	'Katz'
12345	'Shankar'	12345	10101	10101	'Srinivasan'
23121	'Chavez'	23121	76543	76543	'Singh'
44553	'Peltier'	44553	22222	22222	'Einstein'
45678	'Levy'	45678	22222	22222	'Einstein'
76543	'Brown'	76543	45565	45565	'Katz'
76653	'Aoi'	76653	98345	98345	'Kim'
98765	'Bourikas'	98765	98345	98345	'Kim'
98988	'Tanaka'	98988	76766	76766	'Crick'

In [12]:
```
Image("relational1.jpg")
```

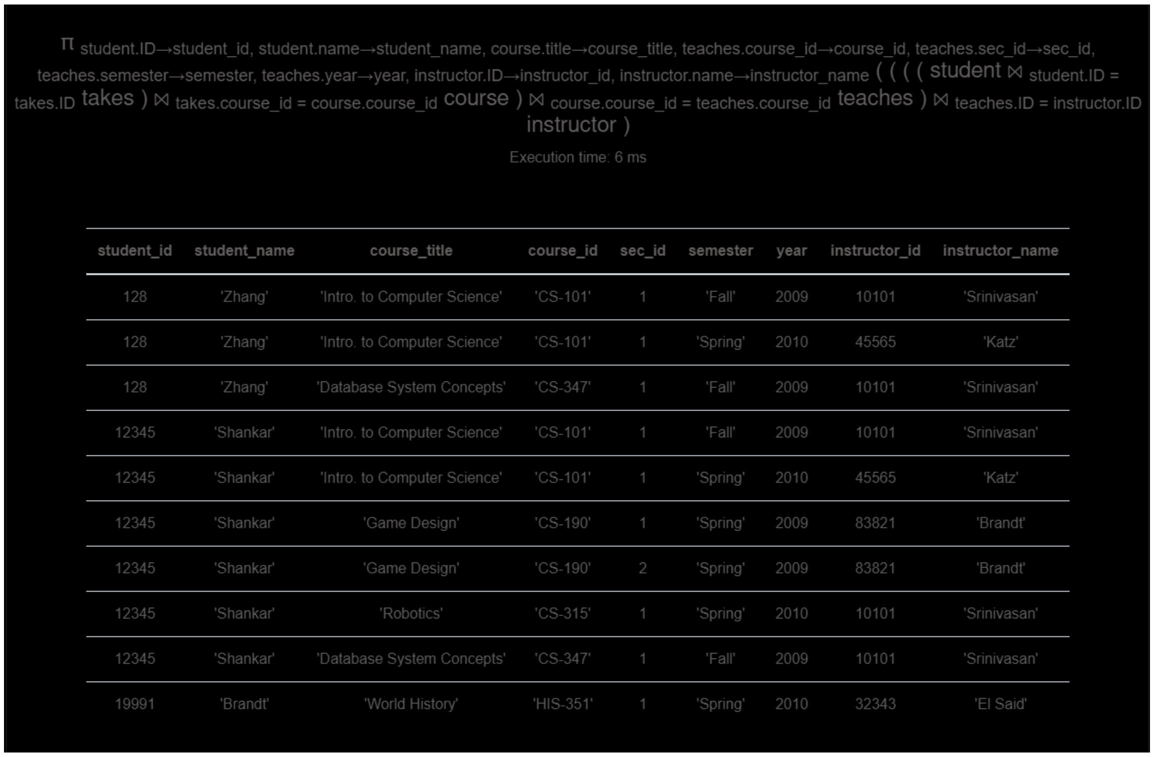
Out[12]:



$(\pi_{\text{ID, name, s\_id, i\_id}} (\text{student} \bowtie_{\text{ID = s\_id}} \text{advisor})) \bowtie_{\text{i\_id = instructor.ID}} \pi_{\text{ID, name}} (\text{instructor})$

Execution time: 2 ms

student.ID	student.name	advisor.s_id	advisor.i_id	instructor.ID	instructor.name
128	'Zhang'	128	45565	45565	'Katz'
12345	'Shankar'	12345	10101	10101	'Srinivasan'
23121	'Chavez'	23121	76543	76543	'Singh'
44553	'Peltier'	44553	22222	22222	'Einstein'
45678	'Levy'	45678	22222	22222	'Einstein'
76543	'Brown'	76543	45565	45565	'Katz'
76653	'Aoi'	76653	98345	98345	'Kim'
98765	'Bourikas'	98765	98345	98345	'Kim'
98988	'Tanaka'	98988	76766	76766	'Crick'



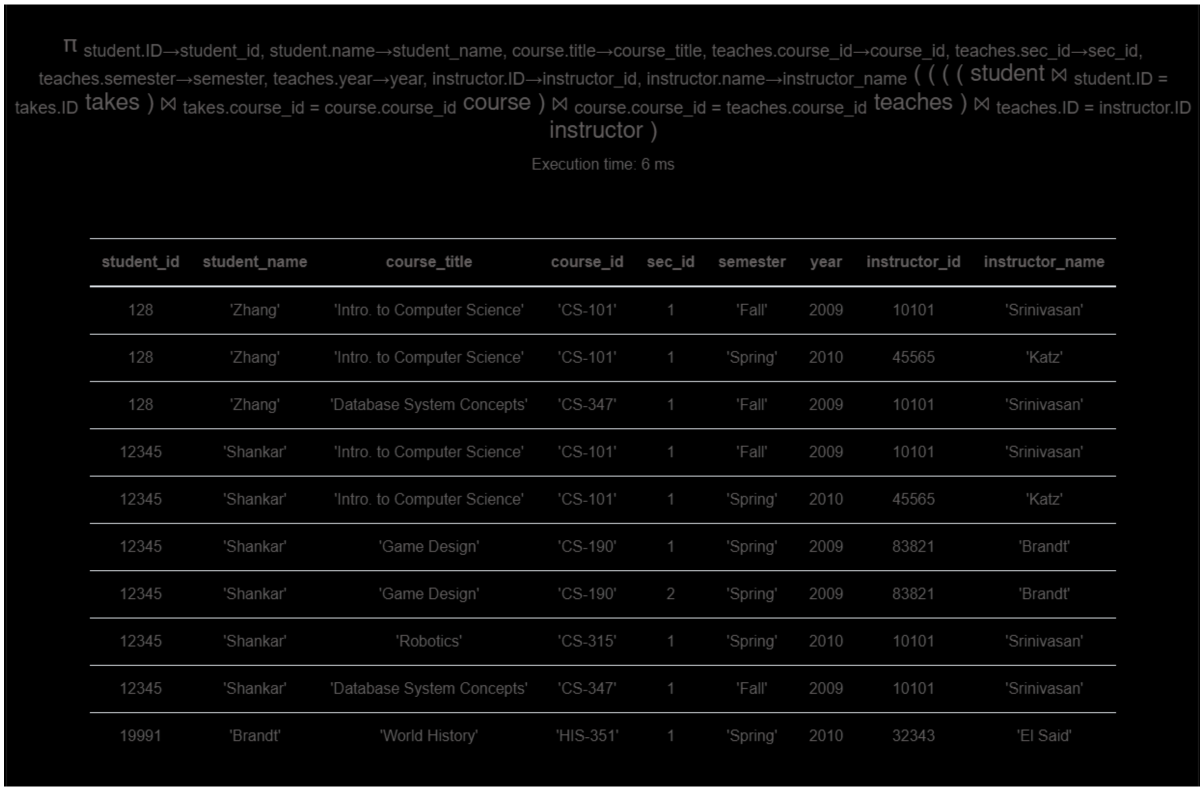
π student.ID→student_id, student.name→student_name, course.title→course_title, teaches.course_id→course_id, teaches.sec_id→sec_id, teaches.semester→semester, teaches.year→year, instructor.ID→instructor_id, instructor.name→instructor_name ((((student ⋈ student.ID = takes.ID takes) ⋈ takes.course_id = course.course_id course) ⋈ course.course_id = teaches.course_id teaches) ⋈ teaches.ID = instructor.ID instructor)

Execution time: 6 ms

student_id	student_name	course_title	course_id	sec_id	semester	year	instructor_id	instructor_name
128	'Zhang'	'Intro. to Computer Science'	'CS-101'	1	'Fall'	2009	10101	'Srinivasan'
128	'Zhang'	'Intro. to Computer Science'	'CS-101'	1	'Spring'	2010	45565	'Katz'
128	'Zhang'	'Database System Concepts'	'CS-347'	1	'Fall'	2009	10101	'Srinivasan'
12345	'Shankar'	'Intro. to Computer Science'	'CS-101'	1	'Fall'	2009	10101	'Srinivasan'
12345	'Shankar'	'Intro. to Computer Science'	'CS-101'	1	'Spring'	2010	45565	'Katz'
12345	'Shankar'	'Game Design'	'CS-190'	1	'Spring'	2009	83821	'Brandt'
12345	'Shankar'	'Game Design'	'CS-190'	2	'Spring'	2009	83821	'Brandt'
12345	'Shankar'	'Robotics'	'CS-315'	1	'Spring'	2010	10101	'Srinivasan'
12345	'Shankar'	'Database System Concepts'	'CS-347'	1	'Fall'	2009	10101	'Srinivasan'
19991	'Brandt'	'World History'	'HIS-351'	1	'Spring'	2010	32343	'El Said'

In [13]:
```
Image("relational2.png")
```

Out[13]:



π student.ID→student_id, student.name→student_name, course.title→course_title, teaches.course_id→course_id, teaches.sec_id→sec_id, teaches.semester→semester, teaches.year→year, instructor.ID→instructor_id, instructor.name→instructor_name ((((student ⋈ student.ID = takes.ID takes) ⋈ takes.course_id = course.course_id course) ⋈ course.course_id = teaches.course_id teaches) ⋈ teaches.ID = instructor.ID instructor)

Execution time: 6 ms

student_id	student_name	course_title	course_id	sec_id	semester	year	instructor_id	instructor_name
128	'Zhang'	'Intro. to Computer Science'	'CS-101'	1	'Fall'	2009	10101	'Srinivasan'
128	'Zhang'	'Intro. to Computer Science'	'CS-101'	1	'Spring'	2010	45565	'Katz'
128	'Zhang'	'Database System Concepts'	'CS-347'	1	'Fall'	2009	10101	'Srinivasan'
12345	'Shankar'	'Intro. to Computer Science'	'CS-101'	1	'Fall'	2009	10101	'Srinivasan'
12345	'Shankar'	'Intro. to Computer Science'	'CS-101'	1	'Spring'	2010	45565	'Katz'
12345	'Shankar'	'Game Design'	'CS-190'	1	'Spring'	2009	83821	'Brandt'
12345	'Shankar'	'Game Design'	'CS-190'	2	'Spring'	2009	83821	'Brandt'
12345	'Shankar'	'Robotics'	'CS-315'	1	'Spring'	2010	10101	'Srinivasan'
12345	'Shankar'	'Database System Concepts'	'CS-347'	1	'Fall'	2009	10101	'Srinivasan'
19991	'Brandt'	'World History'	'HIS-351'	1	'Spring'	2010	32343	'El Said'

Problem 2

Write a relational algebra expression that produces a result table with the following format:

`(dept_name, building, classroom, capacity)`

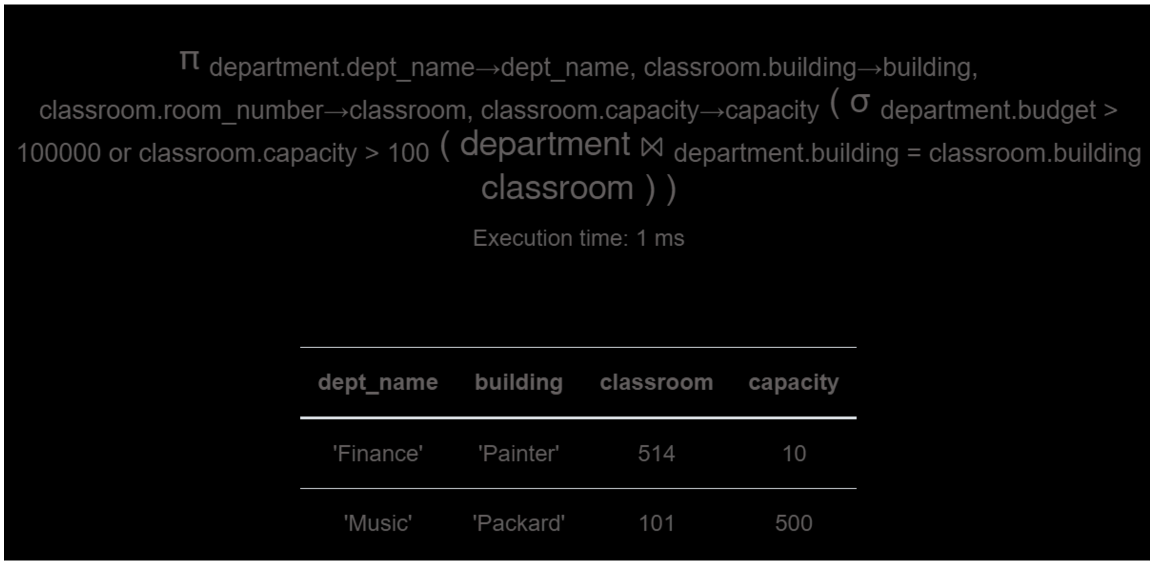
This contains tuples where:

- The department is in the building, e.g. there is a tuple in `department` that has the `dept_name` and `building`.
- The `classroom` is in the `building`.
- The result ONLY contains entries for which the department's `budget` is greather than 100,000 or the classroom's `capacity` is greater than 100.

Past relational algebra here.

```
π department.dept_name->dept_name, classroom.building->building,
classroom.room_number->classroom, classroom.capacity->capacity (σ
department.budget>100000 or classroom.capacity>100 (department ⨝
department.building=classroom.building classroom))
```

Execute your query on the Relax calculator and show an image of the first page of your result below.



π department.dept_name→dept_name, classroom.building→building, classroom.room_number→classroom, classroom.capacity→capacity (σ department.budget > 100000 or classroom.capacity > 100 (department ⨝ department.building = classroom.building classroom))

Execution time: 1 ms

dept_name	building	classroom	capacity
'Finance'	'Painter'	514	10
'Music'	'Packard'	101	500

```
In [14]: Image("relational3.png")
```

Out[14]:

$$\pi_{\text{department.dept\_name} \rightarrow \text{dept\_name},\ \text{classroom.building} \rightarrow \text{building},\ \text{classroom.room\_number} \rightarrow \text{classroom},\ \text{classroom.capacity} \rightarrow \text{capacity}} \left(\sigma_{\text{department.budget} > 100000 \text{ or classroom.capacity} > 100} \left(\text{department} \bowtie_{\text{department.building} = \text{classroom.building}} \text{classroom} \right) \right)$$

Execution time: 1 ms

dept_name	building	classroom	capacity
'Finance'	'Painter'	514	10
'Music'	'Packard'	101	500

SQL

Use the database that is associated with the recommended textbook for these questions. You loaded this in HW0.

Problem 1

Write a SQL query that produces a table of the form `(student_id, student_name, advisor_id, advisor_name)` that shows the ID and name of a student combined with their advisor. Only include rows where both the student and the advisor are in the `Comp. Sci.` and the student has at least 50 total credits.

Execute your SQL below.

In [15]:

```
%%sql

USE db_book;
SELECT s.ID              as student_id,
       s.name            as student_name,
       s.i_ID            as advisor_id,
       instructor.name as advisor_name
FROM instructor,
     (SELECT student.ID,
             student.name,
             advisor.i_ID
      FROM student,
           advisor
      WHERE student.ID = advisor.s_ID
        AND student.dept_name = "Comp. Sci."
        AND student.tot_cred > 50) as s
WHERE instructor.ID = s.i_ID
  AND instructor.dept_name = "Comp. Sci.";
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
2 rows affected.
```

Out[15]:

student_id	student_name	advisor_id	advisor_name
00128	Zhang	45565	Katz
76543	Brown	45565	Katz

Problem 2

Consider the following query.

In [16]:
```
%%sql
select * from db_book.student where dept_name='Comp. Sci.'
```

```
 * mysql+pymysql://root:***@localhost
4 rows affected.
```

Out[16]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
54321	Williams	Comp. Sci.	54
76543	Brown	Comp. Sci.	58

The following table makes a copy of the student table.

In [17]:
```
%%sql
create table if not exists student_hw1b as select * from student;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

Out[17]: []

In [18]:
```
%sql select * from student_hw1b where dept_name='Comp. Sci.'
```

```
 * mysql+pymysql://root:***@localhost
4 rows affected.
```

Out[18]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
76543	Brown	Comp. Sci.	58
54321	Williams	Comp. Sci.	54

We are now going to make some changes to `student_hw1b`

Write and execute a SQL statement that changes Willliams tot_cred to 75.

In [19]:
```
%%sql
UPDATE student_hw1b SET tot_cred=75 WHERE name="Williams";
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[19]:
```
[]
```

Show the result.

In [20]:
```
%sql select * from student_hw1b where dept_name='Comp. Sci.'
```

```
 * mysql+pymysql://root:***@localhost
4 rows affected.
```

Out[20]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
76543	Brown	Comp. Sci.	58
54321	Williams	Comp. Sci.	75

Write a SQL statement that deletes Williams from the `student_hw1b` table and execute in the cell below.

In [21]:
```
%%sql

delete from student_hw1b where name="Williams";
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[21]:
```
[]
```

Show the resulting table.

In [22]:
```
%sql select * from student_hw1b where dept_name='Comp. Sci.'
```

```
 * mysql+pymysql://root:***@localhost
3 rows affected.
```

Out[22]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
76543	Brown	Comp. Sci.	58

Write and execute SQL statement that puts the original data for Williams back in the table.

In [23]:
```
%%sql

insert into student_hw1b values (54321, "Williams", "Comp. Sci.", 54);
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[23]:
```
[]
```

Show the table.

In [24]:
```
%sql select * from student_hw1b where dept_name='Comp. Sci.'
```

```
 * mysql+pymysql://root:***@localhost
4 rows affected.
```

Out[24]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
76543	Brown	Comp. Sci.	58
54321	Williams	Comp. Sci.	54

In [ ]: