COMS W4111-002/V02, Spring 22:
Take Home Midterm

Information and Instructions

- The midterm exam is due on 04-NOV at 11:59 PM. **You may not use late days.**
- See the Ed post #403 (https://edstem.org/us/courses/27172/discussion/2058572) for submission instructions.
- Students should periodically check Ed post #404 (https://edstem.org/us/courses/27172/discussion/2058585) for clarifications.
- You may use lecture notes, lecture slides, ... to help answer questions. You may also use online sources of information. If you use an online source,
  1. You must provide a link to the source.
  2. You are still responsible for ensuring the answer is correct. Not everything on the web is correct.
  3. You **MUST NOT** simply cut and paste, copy verbatim, ... You can use the information for guidance but must provide the answer in your own words and own code.
- You **MUST NOT** collaborate with other students or other people in any way. You may discuss the exam with TAs and instructors.

Environment Setup

Notes:

1. This section tests your environment.
2. You will need to change the MySQL userID and password in some of the cells below to match your configuration.
3. You may need to load data and copy databases. The relevant questions provide information.
4. You will need to:
   A. Install the Classic Models (https://www.mysqltutorial.org/mysql-sample-database.aspx) database if you have not already done so.

B. Install the sample database (https://www.db-book.com/university-lab-dir/sample_tables-dir/index.html) that comes with the recommended textbook if you have not already done so.

In [1]:

```
%load_ext sql
```

In [2]:

```
%sql mysql+pymysql://root:dbuserdbuser@localhost
```

In [3]:

```
%sql select * from classicmodels.customers where country='Spain'
```

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

Out[3]:

customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1	ad
141	Euro+ Shopping Channel	Freyre	Diego	(91) 555 94 44	C/ Moralzarzal, 86	
216	Enaco Distributors	Saavedra	Eduardo	(93) 203 4555	Rambla de Cataluña, 23	
237	ANG Resellers	Camino	Alejandra	(91) 745 6555	Gran Vía, 1	
344	CAF Imports	Fernandez	Jesus	+34 913 728 555	Merchants House	
458	Corrida Auto Replicas, Ltd	Sommer	Martín	(91) 555 22 82	C/ Araquil, 67	
465	Anton Designs, Ltd.	Anton	Carmen	+34 913 728555	c/ Gobelas, 19-1 Urb. La Florida	
484	Iberia Gift Imports, Corp.	Roel	José Pedro	(95) 555 82 82	C/ Romero, 33	

In [4]:

```
from sqlalchemy import create_engine
```

In [5]:

```
sql_engine = create_engine("mysql+pymysql://root:dbuserdbuser@localhost")
```

In [6]:

```
import pandas as pd
```

In [7]:

```
sql = """
    select customerName, customerNumber, city, country from classicmodels.customers
        where country = 'Spain'
"""

res = pd.read_sql(sql, con=sql_engine)
```

In [8]:

```
res
```

Out[8]:

	customerName	customerNumber	city	country
0	Euro+ Shopping Channel	141	Madrid	Spain
1	Enaco Distributors	216	Barcelona	Spain
2	ANG Resellers	237	Madrid	Spain
3	CAF Imports	344	Madrid	Spain
4	Corrida Auto Replicas, Ltd	458	Madrid	Spain
5	Anton Designs, Ltd.	465	Madrid	Spain
6	Iberia Gift Imports, Corp.	484	Sevilla	Spain

In [9]:

```
import pymysql
```

In [10]:

```
sql_conn = pymysql.connect(
    user="root",
    password='dbuserdbuser',
    host="localhost",
    port=3306,
    cursorclass=pymysql.cursors.DictCursor,
    autocommit=True)
```

In [11]:

```
try:
    cur = sql_conn.cursor()
    res = cur.execute(sql)
    res = cur.fetchall()
except Exception as e:
    print("Exception ", e, "is probably NOT good.")
```

In [12]:

```
res
```

Out[12]:

```
[{'customerName': 'Euro+ Shopping Channel',
  'customerNumber': 141,
  'city': 'Madrid',
  'country': 'Spain'},
 {'customerName': 'Enaco Distributors',
  'customerNumber': 216,
  'city': 'Barcelona',
  'country': 'Spain'},
 {'customerName': 'ANG Resellers',
  'customerNumber': 237,
  'city': 'Madrid',
  'country': 'Spain'},
 {'customerName': 'CAF Imports',
  'customerNumber': 344,
  'city': 'Madrid',
  'country': 'Spain'},
 {'customerName': 'Corrida Auto Replicas, Ltd',
  'customerNumber': 458,
  'city': 'Madrid',
  'country': 'Spain'},
 {'customerName': 'Anton Designs, Ltd.',
  'customerNumber': 465,
  'city': 'Madrid',
  'country': 'Spain'},
 {'customerName': 'Iberia Gift Imports, Corp.',
  'customerNumber': 484,
  'city': 'Sevilla',
  'country': 'Spain'}]
```

In [13]:

```
cur.close()
```

Written Questions

Note:

"If you can't explain something in a few words, try fewer." – Robert Brault

"Professor Ferguson has the patience of a ferret that just drank a double espresso. If your answer is long, he gets bored and cranky, and deducts points." - Anonymous TA advising students in a previous semester.

- We expect brief, succinct answers.
- We deduct points for bloviating.

W1

Briefly explain the differences between:

1. *Candidate Key* and *Super Key.*
2. *Primary Key* and *Unique Key.*
3. *Natural Key* and *Surrogate Key.*

Answer

1. Candidate Key is a subset of Super Key. Super Keys may not be Candidate Key but all Candidiate Keys are Super Keys.
2. Primary Key do not allow NULL value, but Unique Key does. One table can only has one Primary Key, but multiple Unique Key.
3. Surrogate Key is generated automatically by the system, is a sequencial integer without business meaning. Natural Key already exsit in the table, which has business meaning. Both keys mean for unique record in a table.

W2

SQL supports the modifier *ON UPDATE* and *ON DELETE* for foreign key definitions. The database engines do not support *ON INSERT*. Why would implementing *ON INSERT* be impossible in most scenarios?

Answer

Because of the foreign key constraints, which prevent insert data to foreign key column, since it must a value contained in the parent table.

W3

Codd's Third Rule states, "Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type."

Consider a table of the form:

```
create table if not exists orders
(
    uni            varchar(12) not null
        primary key,
    last_name      varchar(64) not null,
    first_name     varchar(64) not null,
    age            int         null
)
```

If we do not know the value of `age`, a poor design would use a convention like setting age to `-1` instead of using `NULL`. Give an example of a query for which not following Codd's Thurd Rule would result in an incorrect answer.

Answer

If the user want to query all student whose age are known, then the query will search all NOT NULL rows for ages and all age = -1 will be included, resulting a wrong answer.

W4

The relational model and SQL are *closed* under their operations. Briefly explain why this concept is critical joining three tables?

Answer

When critical joining three tables, the query result should not above the range of joining tables, which has the same concept with the closed relational model and SQL.

W5

Codd's 6th rule states, "All views that are theoretically updatable are also updatable by the system."

Using the following table definition, use SQL (`CREATE VIEW`) to define:

1. Two views of the table for which it is impossible to update the base table through the view.
2. One view for which it is possible to update through the view.

- You do not need to execute the create statement. We are focusing on your understanding.

```
create table S22_W4111_Midterm.midterm_employees
(
    social_security_no char(9) not null
        primary key,
    last_name varchar(64) not null,
    first_name varchar(64) not null,
    dept_no char(4) not null,
    salary double not null
);
```

Answer

1.
(1)

```
CREATE VIEW w5_1 AS
 SELECT
   DISTINCT last_name,
   first_name,
   social_security_no,
   dept_no,
   salary
 FROM
   S22_W4111_Midterm.midterm_employees
```

(2)

```
CREATE VIEW w5_2 AS
 SELECT
   DISTINCT last_name,
   first_name,
   social_security_no,
   DISTINCT dept_no,
   salary
 FROM
   S22_W4111_Midterm.midterm_employees
 GROUP BY
   salary
```

2.

```
CREATE VIEW w5_3 AS
    SELECT *
    FROM
      S22_W4111_Midterm.midterm_employees
```

W6

Consider the following table:

```
create table S22_W4111_Midterm.midterm_employees
(
    phone_number varchar(64) not null primary key,
    last_name varchar(64) not null,
    first_name varchar(64) not null,
);
```

Telephone numbers are of the form `country code` followed by the phone number. Some examples are:

- 01 212-555-1212
- 44 038 717 980 01

Why is storing the number as a single `varchar` a poor design? What problems could that cause? How would you change the table definition.

Answer

The country code should be stored separtely, and the phone number may has different format causing confusion for users.
Add a column country_code to store the first 2 or 3 digits before the first space space and insert rest of string to phone_number, replace all the dashes to space to unified format.

W7

Briefly explain the differences between:

- Database stored procedure
- Database function

- Database trigger

Answer

Database stored procedure is a set of code that stored how to query the database and user can call procedure multiple times.
Database function is more likely than function in C++, which has input parameter and output, user can use database function (like avg(), sum()) in query.
Database trigger is a specific action which will perform before or after the update action (update, delete, insert). User can add subquery in trigger including use function.|

W8

Briefly explain:

- Natural join
- Equi-join
- Theta join
- Self-join

Give a scenario in which a Natural Join would produce an incorrect answer.

Answer

Natural join: join tables based on same column name and datatype. The result will contain all attributes for tables but only one copy.
Equi-join: join tables based on equality operator. The result will contain all equivalent values and columns.
Theta join: join tables based on arbitrary comparison relationships, such as >, <, or <=. The result will contain all values and columns that match the comparison condition.
Self-join: join tables without condition. The result will contain all columns in tables.
If the attribute and datatype is matched but the meaning is diffreent, for example, student.ID and teacher.ID. If natural join for these two tables, then it will produce an incorrent answer.

W9

We have seen examples in SQL of implementing relationships between two tables using an *associative entity* table instead of foreign keys. Give two reasons for using the associative entity design pattern.

Answer

When the relationship between two tables is changing by the propertity of relationship.
Foreign key may caused meaningless for a table.

W10

Professor Ferguson often adds a `LIMIT` to his example queries in Jupyter Notebooks. Assume the table `customers` is very large. Why would the query

`select * from customers`

cause problems for the notebook? How does addining `limit 20` solve this problem for an example?

Answer

Without setting a LIMIT clause may cause the notebook stuck and need a large response time. The limit 20 only shows the first 20 rows of the result, which speed up the notebook.

Relational Algebra

R1

- You can assume that the type for the columns in this question are `varchar(32)`.
- Translate the following relational schema definition into an equivalent SQL `CREATE TABLE` statement.
- You do not need to execute the statement. We are focusing on understanding.

$$(\underline{policy\_type}, \underline{policy\_no}, policy\_date)$$

Answer

```
CREATE TABLE policy (
    policy_type varchar(32),
    policy_no varchar(32),
    policy_date varchar(32),
    PRIMARY KEY (policy_type, policy_no)
);
```

R2

Use the RelaX calculator (https://dbis-uibk.github.io/relax/calc/gist/4f7866c17624ca9dfa85ed2482078be8/relax-silberschatz-english.txt/0) with the textbook's sample data for this question.

Answer Format: Your answer to the relational algebra query should contain three secions:

1. A Markdown cell with the relational algebra statement.
2. An image capture of the query exection tree.
3. An image capture of the result table.

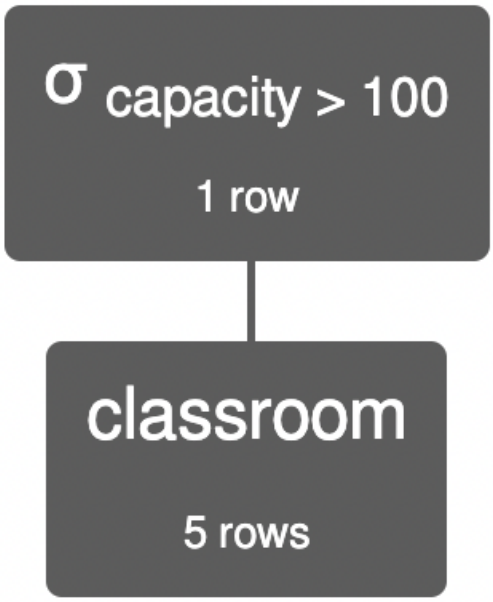
For example, a query returning all classrooms with `capacity > 100` would have the following cells:

Relational Algebra Statement

`σ capacity > 100 (classroom)`

Query Execution Tree

This must show the execution tree and relational algebra statement.



$$\sigma_{\text{capacity} > 100}(\text{classroom})$$

Execution time: 1 ms

Query Result

This must show the relational algebra statement and the result table.

$$\sigma_{\text{capacity} > 100}(\text{classroom})$$

Execution time: 1 ms

classroom.building	classroom.room_number	classroom.capacity
'Packard'	101	500

The Question

In the sample data,

- The relation `advisor` represents the advisor-student relationship between a `student` and an `instructor`.

- Write a relational algebra expression that produces the following information:
  - The `ID` and `name` of `student`, and the `ID` and `name` on `instructor`
  - For students and instructors in the CS department.
  - The information should be `null` if the instructor does not advise a student and vice-versa.
- To help, you are trying to produce the following information.
- **Note:**
  - You **may not** use full outer join.
  - You will have to use the column rename operation for project.

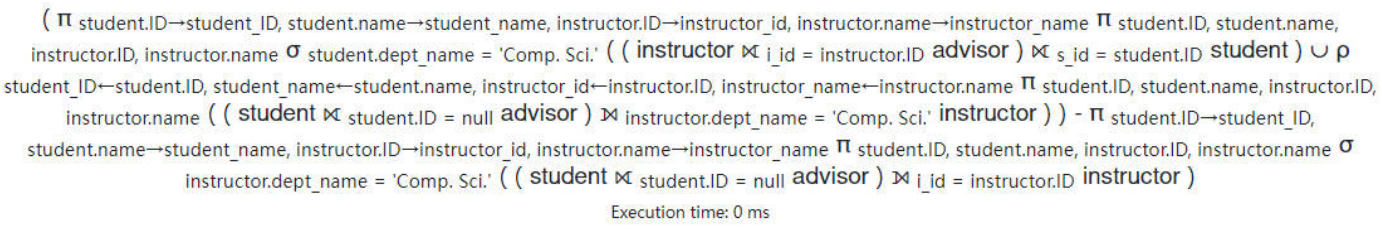
student_id	student_name	instructor_id	instructor_name
12345	'Shankar'	10101	'Srinivasan'
128	'Zhang'	45565	'Katz'
76543	'Brown'	45565	'Katz'
null	null	83821	'Brandt'
54321	'Williams'	null	null

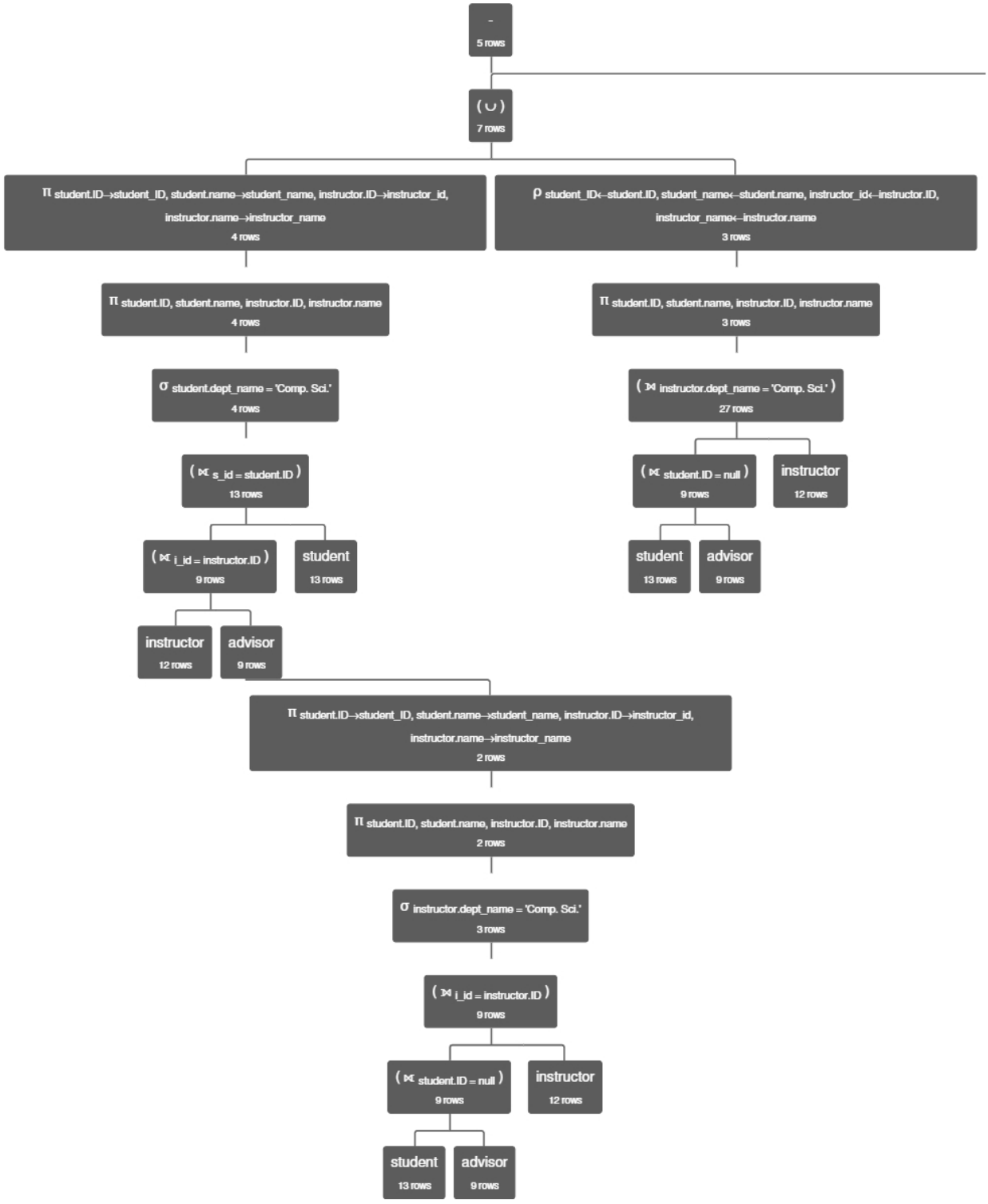
Answer

Query

```
( π student_ID←student.ID, student_name←student.name, instructor_id←instructor.ID, instructor_name←instructor.name π student.ID, student.name, instructor.ID, instructor.name σ student.dept_name = 'Comp. Sci.' ( ( instructor ⋈ i_id = instructor.ID advisor ) ⋈ s_id = student.ID student ) ∪ ρ student_ID←student.ID, student_name←student.name, instructor_id←instructor.ID, instructor_name←instructor.name π student.ID, student.name, instructor.ID, instructor.name ( ( student ⋈ student.ID = null advisor ) ⋈ instructor.dept_name = 'Comp. Sci.' instructor ) ) - π student_ID←student.ID, student_name←student.name, instructor_id←instructor.ID, instructor_name←instructor.name π student.ID, student.name, instructor.ID, instructor.name σ instructor.dept_name = 'Comp. Sci.' ( ( student ⋈ student.ID = null advisor ) ⋈ i_id = instructor.ID instructor )
```

Query Tree





Query Result

(π student.ID→student_ID, student.name→student_name, instructor.ID→instructor_id, instructor.name→instructor_name π student.ID, student.name, instructor.ID, instructor.name σ student.dept_name = 'Comp. Sci.' ((instructor ⋉ i_id = instructor.ID advisor) ⋉ s_id = student.ID student) ∪ ρ student_ID←student.ID, student_name←student.name, instructor_id←instructor.ID, instructor_name←instructor.name π student.ID, student.name, instructor.ID, instructor.name ((student ⋉ student.ID = null advisor) ⋈ instructor.dept_name = 'Comp. Sci.' instructor)) - π student.ID→student_ID, student.name→student_name, instructor.ID→instructor_id, instructor.name→instructor_name π student.ID, student.name, instructor.ID, instructor.name σ instructor.dept_name = 'Comp. Sci.' ((student ⋉ student.ID = null advisor) ⋈ i_id = instructor.ID instructor)

Execution time: 0 ms

student_ID	student_name	instructor_id	instructor_name
128	'Zhang'	45565	'Katz'
12345	'Shankar'	10101	'Srinivasan'
54321	'Williams'	*null*	*null*
76543	'Brown'	45565	'Katz'
null	*null*	83821	'Brandt'

Entity Relationship Model and Implementation

Explanation

For this problem,

- There is a written description of a data model.
- You must draw (using Lucidchart) a Crow's Foot notation ER diagram for the *logical model* implementing the written description. Note that not all concepts in the data model description can be modeled in the ER diagram.
- You must then write SQL DDL statements and execute the statements to create tables and constraints realizing the written data model description.

Written Description

There are the following entity types:

- `employee`:
  - `employee_id` is a 4 digit number that may begin with 0, e.g. `0201`. An employee must have a unique `employee_id`.
  - `last_name` is a string with maximum length 64. An employee must have a last name.
  - `first_name` is a string with maximum length 64. An employee must have a first name.
  - `employee_type` must be one of the following values, `regular`, `manager`, `executive`.
  - `employee_email` may be unknown, but if known it must be unique.

- `project`:
  - `project_code` is a two character code that must contain two uppercase English letters (A, B, ..., Z) and is unique.
  - `project_name` is a text string of maximum length 32.

- `project_team` is an associative entity of the form (none of the values may be NULL):
  - `project_code`
  - `sponsor_id` is `employee_id` of an employee who is an `executive`.
  - `manager_id` is the `employee_id` of an employee who is a `manager`.
  - `employee_id` is the `employee_id` of an employee working on the project.

- Constraints on `project_team`:
  - `project_code` is unique in the table.

- An `employee_id` can appear at most three times.
- The combination of `(sponsor_id, manager_id)` can appear at most once.

Note:

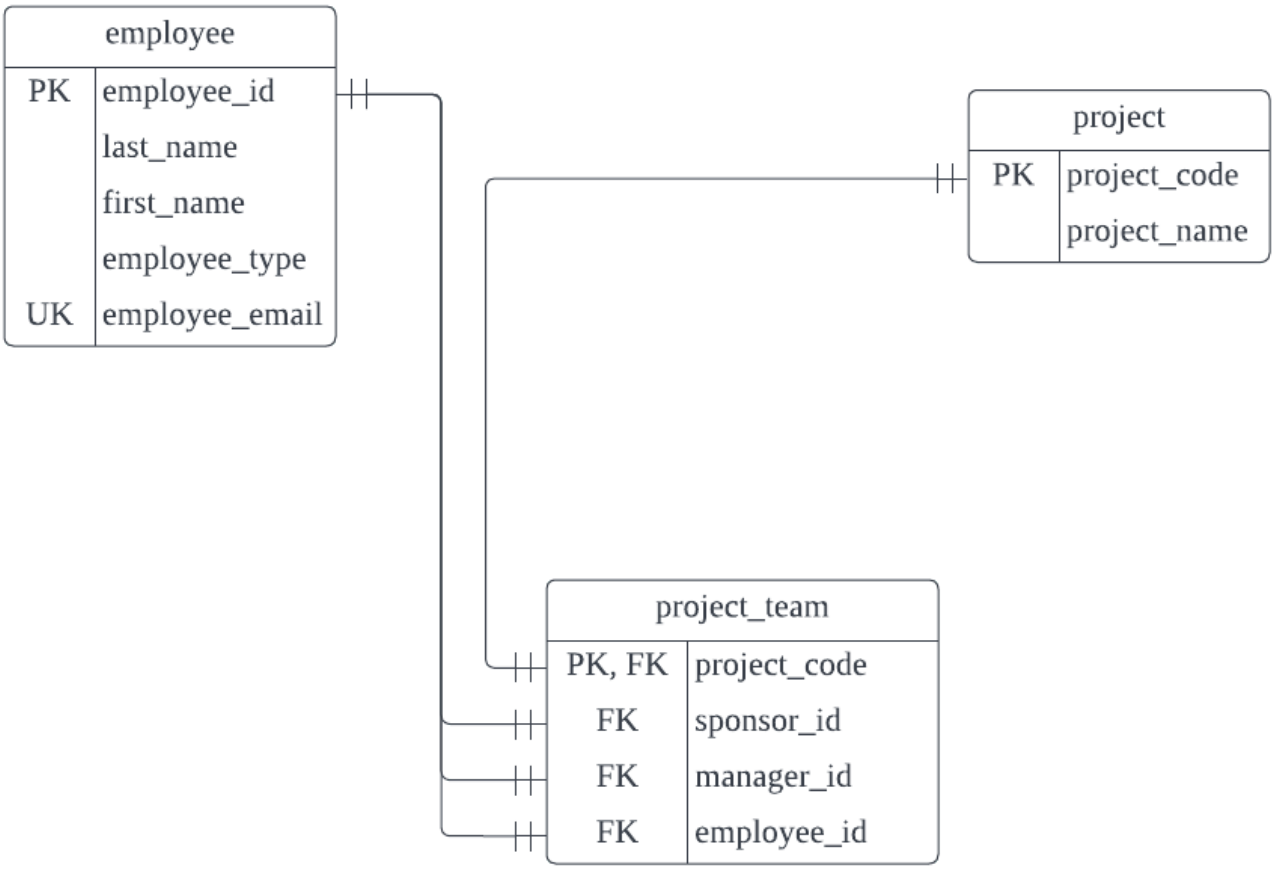
Being able to make sense out of a written description of a data model and producing a reasonably accurate diagram and DDL is an important skill. Most of the time, you will have to make assumptions or modify/extend constraints. The business statekholder/partner specifying the data model is not a database expert. There description may be incomplete or confused.

We are looking for your ability to apply what you have learned to a complex problem. If you have to make assumptions, note them. We will not deduct points for reasonable assumptions.

You may have to use check constraints, triggers, foreign keys, ... in your DDL.

Answer

Crow's Foot ER Diagram



Assume that each employee join one project

DDL Statements and Execution

In [14]:

```
%%sql

drop database if exists f22_midterm;

create database f22_midterm;
```

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

Out[14]:

```
[]
```

In [15]:

```
%%sql

use f22_midterm;

create table employee (
    employee_id     decimal(4,0)    UNSIGNED    ZEROFILL,
    last_name       varchar(64)     NOT NULL,
    first_name      varchar(64)     NOT NULL,
    employee_type   enum('regular', 'manager', 'executive') NOT NULL,
    employee_email  varchar(64),
    primary key (employee_id),
    unique (employee_email)
);

create trigger four_digit_begin_zero
    before insert on employee
    for each row
    begin
        if NEW.employee_id >= 1000 then
            signal sqlstate '40000';
        end if;
    end;

create table project (
    project_code varchar(2),
    project_name text(32),
    primary key (project_code),
    constraint upper_code check ( project_code = UPPER(project_code) )
);

create table project_team (
    project_code    varchar(2)      NOT NULL ,
    sponsor_id      decimal(4,0)    NOT NULL ,
    manager_id      decimal(4,0)    NOT NULL ,
    employee_id     decimal(4,0)    NOT NULL ,
    primary key (project_code),
    constraint project_team_ibfk_1
        foreign key (sponsor_id) references employee (employee_id),
    constraint project_team_ibfk_2
        foreign key (project_code) references project (project_code)
);

create trigger max_three_employee
    before insert on project_team
    for each row
    begin
        if count(employee_id) >= 3 then
            signal sqlstate '30000';
        end if;
    end;

create trigger max_one_spo_man
    before insert on project_team
    for each row
    begin
        if count(sponsor_id) >= 1
            AND
            count(manager_id) >= 1
        then
```

```
            signal sqlstate '10000';
        end if;
    end;

create trigger sponsor
    before insert on project_team
    for each row
    begin
        set @tmp = (
        SELECT
            employee.employee_type
        FROM
            employee
        WHERE
            employee.employee_id = NEW.sponsor_id
        LIMIT 1
        );
        if @tmp != 'executive' then
            signal sqlstate '12301';
        end if;
    end;

create trigger manager
    before insert on project_team
    for each row
    begin
        set @tmp = (
        SELECT
            employee.employee_type
        FROM
            employee
        WHERE
            employee.employee_id = NEW.manager_id
        LIMIT 1
        );
        if @tmp != 'manager' then
            signal sqlstate '12302';
        end if;
    end;

create trigger employee
    before insert on project_team
    for each row
    begin
        set @tmp = (
        SELECT
            employee.employee_type
        FROM
            employee
        WHERE
            employee.employee_id = NEW.employee_id
        LIMIT 1
        );
        if @tmp != 'employee' then
            signal sqlstate '12303';
        end if;
    end;
```

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

```
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[15]:

```
[]
```

SQL Queries

- You will use the Classic Models (https://www.mysqltutorial.org/mysql-sample-database.aspx) data for these questions.
- You loaded this database in a previous HW and tested that you have the database in the setup section.

S1

- Produce a table of the form: (*country*, *total*$_{c}$*ountry*$_{r}$*evenue*).
- Each entry in `orderdetails` produces revenue `quantityOrdered*priceEach`.
- The revenue an `order` produces is the sum of the revenue from the `orderdetails` in the `order`, *but only if the order's status is* `shipped`.
- An `order` has a `customer` and the `customer` is in a country. The `total_country_revenue` is the sum over all shipped orders for customers in a country.
- The result table should have `total_country_revenue` nicely formated, sorted descending and have only countries with `total_country_revenue` >= 200,000.
- **NOTE:** You should be able to produce the answer without my providing the correct query output. I was giggling diabolically like the Riddler from Batman when writing the question.
  Then something like the following happened.

- So the output is below. You must match the output.

In [16]:

```
%%sql

use classicmodels;

DROP TABLE if exists midterm_s1;

CREATE TABLE midterm_s1 AS
    SELECT
        customers.country as country,
        CONCAT('$', format(sum(orderdetails.quantityOrdered * orderdetails.priceEach), 2))
            as total_country_revenue
    FROM customers
        JOIN
            orders
            ON customers.customerNumber = orders.customerNumber

        JOIN
            orderdetails
            ON orders.orderNumber = orderdetails.orderNumber
    WHERE
        orders.status = 'shipped'
    GROUP BY
        customers.country
    ORDER BY
        sum(orderdetails.quantityOrdered * orderdetails.priceEach) DESC;

DELETE FROM
    midterm_s1
WHERE
    convert(replace(right(total_country_revenue, length(total_country_revenue)-1), ',', ''),
            decimal(16,2)) < 200000;

SELECT * FROM midterm_s1;
```

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

Out[16]:

country	total_country_revenue
USA	$3,032,204.26
France	$965,750.58
Spain	$947,470.01
Australia	$509,385.82
New Zealand	$416,114.03
UK	$391,503.90
Italy	$360,616.81
Finland	$295,149.35
Norway	$270,846.30

Singapore	$263,997.78
Canada	$205,911.86

Out[17]:

country	total_country_revenue
USA	$3,032,204.26
France	$965,750.58
Spain	$947,470.01
Australia	$509,385.82
New Zealand	$416,114.03
UK	$391,503.90
Italy	$360,616.81
Finland	$295,149.35
Norway	$270,846.30
Singapore	$263,997.78
Canada	$205,911.86

S2

Return the product information for products not ordered by any French customer (Customer's `country` is France).

I did not want to get hit by Batman again. So, here is a sample answer.

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

Out[18]:

productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP
S18_3233	1985 Toyota Supra	Classic Cars	1:18	Highway 66 Mini Classics	This model features soft rubber tires, working steering, rubber mud guards, authentic Ford logos, detailed undercarriage, opening doors and hood, removable split rear gate, full size spare mounted in bed, detailed interior with opening glove box	7733	57.01	107.57
S18_4027	1970 Triumph Spitfire	Classic Cars	1:18	Min Lin Diecast	Features include opening and closing doors. Color: White.	5545	91.92	143.62

In [17]:

```
%%sql

SELECT *
FROM
    products
WHERE
    products.productCode NOT IN (
        SELECT
            products.productCode as productCode
        FROM
            orders INNER JOIN
                customers
                ON orders.customerNumber = customers.customerNumber

            INNER JOIN
                orderdetails
                ON orderdetails.orderNumber = orders.orderNumber

            INNER JOIN
                products
                ON orderdetails.productCode = products.productCode
        WHERE
            customers.country = 'France'
        GROUP BY
            products.productCode
    );
```

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

Out[17]:

productCode	productName	productLine	productScale	productVendor	productDescription	qua
S18_3233	1985 Toyota Supra	Classic Cars	1:18	Highway 66 Mini Classics	This model features soft rubber tires, working steering, rubber mud guards, authentic Ford logos, detailed undercarriage, opening doors and hood, removable split rear gate, full size spare mounted in bed, detailed interior with opening glove box	
S18_4027	1970 Triumph Spitfire	Classic Cars	1:18	Min Lin Diecast	Features include opening and closing doors. Color: White.	

In [17]:
```
%%sql

SELECT *
FROM
    products
WHERE
    products.productCode NOT IN (
        SELECT
            products.productCode as productCode
        FROM
            orders INNER JOIN
                customers
                ON orders.customerNumber = customers.customerNumber

            INNER JOIN
                orderdetails
                ON orderdetails.orderNumber = orders.orderNumber

            INNER JOIN
                products
                ON orderdetails.productCode = products.productCode
        WHERE
            customers.country = 'France'
        GROUP BY
            products.productCode
    );
```

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

Out[17]:

productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP
S18_3233	1985 Toyota Supra	Classic Cars	1:18	Highway 66 Mini Classics	This model features soft rubber tires, working steering, rubber mud guards, authentic Ford logos, detailed undercarriage, opening doors and hood, removable split rear gate, full size spare mounted in bed, detailed interior with opening glove box	7733	57.01	107.57
S18_4027	1970 Triumph Spitfire	Classic Cars	1:18	Min Lin Diecast	Features include opening and closing doors. Color: White.	5545	91.92	143.62

In [ ]: