F24-W4111-03: Introduction to Databases: Homework 2, Part B

Nathan Gong

Submission Instructions

Note to TAs: The TAs will post submission instructions on Ed, CourseWorks and GradeScope.

Environment Setup

Check Packages and Connections

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")
```

Sample Database

For this homework, we need a more complex database and sample data to adequately write and test complex SQL.

We will use the Classic Models example database for this homework and subsequent homework assignments. The SQL script for creating the database/schema and loading the

data is in the same GitHub directory as this notebook. If you downloaded a zip file to start HW2B, the file is in the directory containing the notebook.

In [8]:
```
%ls -l *.sql
```

```
 Volume in drive C is OS
 Volume Serial Number is 940C-3CFD

 Directory of C:\Users\natha\W4111-Intro-to-Databases-Base\Homework-Assignments\HW2
\f24_w4111_hw2B


 Directory of C:\Users\natha\W4111-Intro-to-Databases-Base\Homework-Assignments\HW2
\f24_w4111_hw2B

10/10/2024  03:00 PM           195,764 mysqlsampledatabase.sql
               1 File(s)        195,764 bytes
               0 Dir(s)  804,639,961,088 bytes free
```

You install and create the database and install the code in the same way you set up and installed the sample database associated with the recommended textbook. You performed this task earlier in the semester. The steps for installing the database using the command line is also on the website for Classic Models.

The following cell will check if you have correctly installed the database.

In [9]:
```
%%sql

use classicmodels;

with one as (
    select
        *
    from
        orders join orderdetails using(orderNumber)
),
    two as (
        select
            *
        from
            customers join one using(customerNumber)
    ),
    three as (
        select
            *
        from
            two join products using(productCode)
    )
select
    customerNumber,
    customerName,
    orderNumber,
    status,
    quantityOrdered,
```

```
    priceEach,
    orderLineNumber,
    productName,
    productVendor
from
    three
where
    customerNumber = 114
order by
    customerNumber, orderNumber, orderLineNumber;
```

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

Out[9]:

customerNumber	customerName	orderNumber	status	quantityOrdered	priceEach	ord
114	Australian Collectors, Co.	10120	Shipped	35	110.45	
114	Australian Collectors, Co.	10120	Shipped	46	158.80	
114	Australian Collectors, Co.	10120	Shipped	29	118.94	
114	Australian Collectors, Co.	10120	Shipped	46	57.54	
114	Australian Collectors, Co.	10120	Shipped	34	72.36	
114	Australian Collectors, Co.	10120	Shipped	22	94.90	
114	Australian Collectors, Co.	10120	Shipped	24	106.79	
114	Australian Collectors, Co.	10120	Shipped	29	82.79	
114	Australian Collectors, Co.	10120	Shipped	29	71.73	
114	Australian Collectors, Co.	10120	Shipped	39	93.01	
114	Australian Collectors, Co.	10120	Shipped	29	68.79	
114	Australian Collectors, Co.	10120	Shipped	49	41.46	
114	Australian Collectors, Co.	10120	Shipped	47	91.34	
114	Australian Collectors, Co.	10120	Shipped	43	72.00	
114	Australian Collectors, Co.	10120	Shipped	24	81.77	
114	Australian Collectors, Co.	10125	Shipped	32	89.38	

customerNumber	customerName	orderNumber	status	quantityOrdered	priceEach	ord
114	Australian Collectors, Co.	10125	Shipped	34	138.38	
114	Australian Collectors, Co.	10223	Shipped	37	80.39	
114	Australian Collectors, Co.	10223	Shipped	32	104.81	
114	Australian Collectors, Co.	10223	Shipped	49	189.79	
114	Australian Collectors, Co.	10223	Shipped	47	110.61	
114	Australian Collectors, Co.	10223	Shipped	28	58.75	
114	Australian Collectors, Co.	10223	Shipped	38	60.94	
114	Australian Collectors, Co.	10223	Shipped	21	90.90	
114	Australian Collectors, Co.	10223	Shipped	29	113.90	
114	Australian Collectors, Co.	10223	Shipped	47	67.58	
114	Australian Collectors, Co.	10223	Shipped	23	68.10	
114	Australian Collectors, Co.	10223	Shipped	34	87.54	
114	Australian Collectors, Co.	10223	Shipped	20	66.73	
114	Australian Collectors, Co.	10223	Shipped	41	41.02	
114	Australian Collectors, Co.	10223	Shipped	25	84.03	

customerNumber	customerName	orderNumber	status	quantityOrdered	priceEach	ord
114	Australian Collectors, Co.	10223	Shipped	26	79.20	
114	Australian Collectors, Co.	10342	Shipped	55	63.14	
114	Australian Collectors, Co.	10342	Shipped	40	118.89	
114	Australian Collectors, Co.	10342	Shipped	22	115.22	
114	Australian Collectors, Co.	10342	Shipped	30	167.65	
114	Australian Collectors, Co.	10342	Shipped	25	76.39	
114	Australian Collectors, Co.	10342	Shipped	42	112.34	
114	Australian Collectors, Co.	10342	Shipped	55	136.70	
114	Australian Collectors, Co.	10342	Shipped	26	57.82	
114	Australian Collectors, Co.	10342	Shipped	39	30.59	
114	Australian Collectors, Co.	10342	Shipped	48	60.01	
114	Australian Collectors, Co.	10342	Shipped	38	124.99	
114	Australian Collectors, Co.	10347	Shipped	30	188.58	
114	Australian Collectors, Co.	10347	Shipped	27	132.97	
114	Australian Collectors, Co.	10347	Shipped	29	132.57	
114	Australian Collectors, Co.	10347	Shipped	45	115.03	
114	Australian Collectors, Co.	10347	Shipped	42	113.17	
114	Australian Collectors, Co.	10347	Shipped	21	136.69	

customerNumber	customerName	orderNumber	status	quantityOrdered	priceEach	ord
114	Australian Collectors, Co.	10347	Shipped	21	46.36	
114	Australian Collectors, Co.	10347	Shipped	50	51.05	
114	Australian Collectors, Co.	10347	Shipped	48	84.09	
114	Australian Collectors, Co.	10347	Shipped	34	60.59	
114	Australian Collectors, Co.	10347	Shipped	45	95.30	
114	Australian Collectors, Co.	10347	Shipped	26	84.33	

Entity Relationship Modeling

Create Entity Relationship Model

The ability to take a high-level description of a required database and produce an ER-diagram is one of the most fundamental skills needed for using databases. Consider the following scenario for creating a data model for the Harry Potter series of books.

The data model has the following entity types:

1. `Character` has the properties:
   - `last_name`
   - `first_name`
   - `description`
2. `Book` has the properties:
   - `title`
   - `volume_number` is the ordinal number of the book in the overall series. For example, Harry Potter and the Goblet of Fire has `volume_number` 4.
   - `publication_year`
3. `Chapter` has the properties:
   - `volume_number`
   - `chapter_number`

- `chapter_title`
- `chapter_summary`

4. `Event` has the following properties:
   - `event_type` is a value from a list of possible event types. The data model should support adding new allowed values for `event_type` without requiring changing the data model/schema.
   - `event_description`

This is where things get tricky. People think of an `Event` having a form like:

1. Event: Harry Receives His Hogwarts Letter

- **Type:** Milestone
- **Description:** After multiple failed attempts to deliver a letter to Harry, Hagrid personally delivers his acceptance letter to Hogwarts.
- **Participants:** Harry Potter, Hagrid, Dursley Family
- **Book:** *Harry Potter and the Philosopher's Stone*
- **Chapter:** Chapter 4: "The Keeper of the Keys"

2. Event: Sorting Hat Ceremony

- **Type:** Ceremony
- **Description:** Harry and his classmates are sorted into their respective houses. Harry is placed in Gryffindor despite the Sorting Hat considering Slytherin.
- **Participants:** Harry Potter, Ron Weasley, Hermione Granger, Draco Malfoy, Sorting Hat
- **Book:** *Harry Potter and the Philosopher's Stone*
- **Chapter:** Chapter 7: "The Sorting Hat"

3. Event: Battle of the Department of Mysteries

- **Type:** Battle
- **Description:** Harry and his friends fight Death Eaters at the Ministry of Magic. Sirius Black is killed by Bellatrix Lestrange.
- **Participants:** Harry Potter, Hermione Granger, Ron Weasley, Neville Longbottom, Luna Lovegood, Ginny Weasley, Death Eaters, Order of the Phoenix
- **Book:** *Harry Potter and the Order of the Phoenix*
- **Chapter:** Chapter 35: "Beyond the Veil"

4. Event: Dumbledore's Death

- **Type:** Tragedy
- **Description:** Severus Snape kills Albus Dumbledore at the top of the Astronomy Tower, as part of a prearranged plan.
- **Participants:** Albus Dumbledore, Severus Snape, Draco Malfoy, Death Eaters
- **Book:** *Harry Potter and the Half-Blood Prince*
- **Chapter:** Chapter 27: "The Lightning-Struck Tower"

5. Event: Harry's Final Duel with Voldemort

- **Type:** Battle
- **Description:** Harry confronts Voldemort in the Great Hall and finally defeats him, bringing an end to the Dark Lord's reign.
- **Participants:** Harry Potter, Lord Voldemort
- **Book:** *Harry Potter and the Deathly Hallows*
- **Chapter:** Chapter 36: "The Flaw in the Plan"

You must define and diagram an ER model using Crow's Foot notation that enables representing the information in a relational/SQL database. This should include defining:

- Primary keys
- Foreign keys
- Relationships necessary to represent and store the data.

To accomplish the task, you *may* need to create additional entity types, attributes, etc.

In the cell below, please document any assumptions/design decisions you make and include the ER diagram.

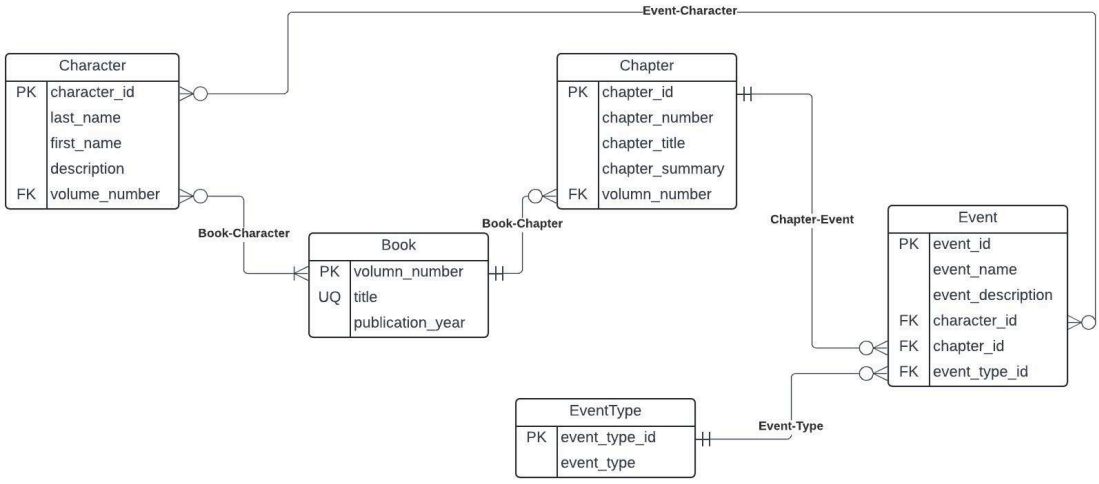
Answer:

Assumptions/design decisions:

- `Character` has a unique identifier `character_id` which acts as its primary key
- `Character` has a foreign key `volume_number` which refers to `Book.volume_number`, and a `Character` may belong to 1 or many `Book`s
- A `Character` may belong to 0 or many `Event`s
- `Book` has a unique identifier `volume_number` which acts as its primary key. `title` is also a unique key (normally book titles are not guaranteed to be unique, but all the books in the Harry Potter series have unique titles)
- A `Book` may have 0 or many `Character`s
- A `Book` may have 0 or many `Chapter`s
- `Chapter` has a unique identifier `chapter_id` which acts as its primary key
- `Chapter` has a foreign key `volume_number` which refers to `Book.volume_number`, and a `Chapter` may belong to only 1 `Book`
- A `Chapter` may have 0 or many `Event`s
- `Event` has a unique identifier `event_id` which acts as its primary key
- `Event` has a foreign key `character_id` which refers to `Character.character`, and an `Event` may have 0 or many `Character`s
- `Event` has a foreign key `chapter_id` which refers to `Chapter.chapter_id`, and an `Event` may belong to only 1 `Chapter`
- The `EventType` entity type allows for adding new allowed values for types of `Event`s without requiring changing the data model.

- `EventType` has a unique identifier `event_type_id` which acts as its primary key
- `Event` has a foreign key `event_type_id` which refers to `EventType.event_type_id`, and an `Event` may have only 1 `EventType`
- The `Event` form can be generated from its attributes and from its direct and indirect relationships with `EventType`, `Character`, `Book`, and `Chapter`

ER Diagram

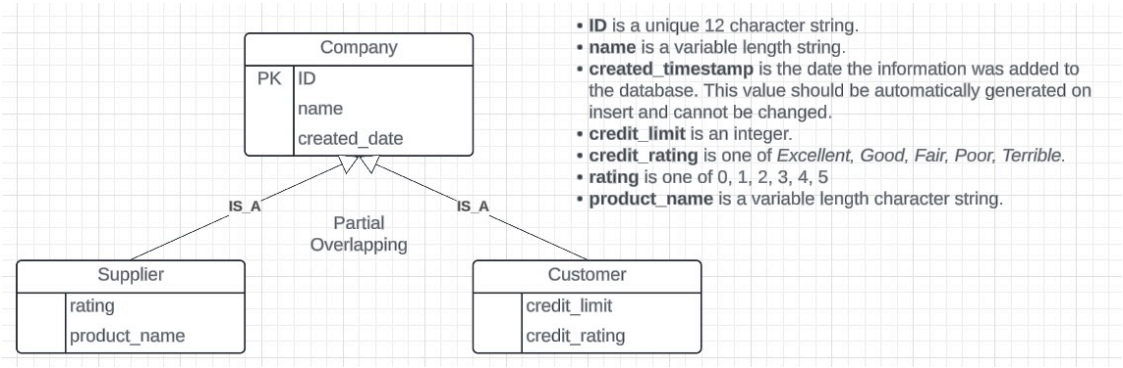


Implement ER Diagram

The diagram below is a simple ER Model using specialization to model companies with which a corporation may do business. You must create a database with the name `w4111_hw2_<uni>`, replacing `<uni>` with your UNI.

Your implementation must use the *3 Table Solution* to modeling specialization.

You need to create the necessary primary keys, foreign keys, views, etc. A query on the table/view `Company` must include a column indicating whether the `Company` is a `Supplier` or a `Customer.`



ER to SQL Data Model

In [10]:
```
%%sql

/* Your create and alter table statements. */

CREATE DATABASE IF NOT EXISTS w4111_hw2_ng2695;
USE w4111_hw2_ng2695;

CREATE TABLE IF NOT EXISTS Company_t (
    ID CHAR(12),
    name VARCHAR(64),
    created_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    type CHAR(1),
    PRIMARY KEY (ID)
);

CREATE TRIGGER before_company_t_update
BEFORE UPDATE on Company_t
FOR EACH ROW
BEGIN
    IF New.created_timestamp <> OLD.created_timestamp THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: Cannot change the value of the created_timestamp
    END IF;
END;

CREATE TABLE IF NOT EXISTS Supplier_t (
    ID CHAR(12),
    rating TINYINT,
    product_name VARCHAR(64),
    CHECK (rating IN (0, 1, 2, 3, 4, 5)),
    PRIMARY KEY (ID),
    FOREIGN KEY (ID) REFERENCES Company_t(ID)
);

CREATE TABLE IF NOT EXISTS Customer_t (
    ID CHAR(12),
    credit_limit INT,
    credit_rating ENUM('Excellent', 'Good', 'Fair', 'Poor', 'Terrible') NOT NULL,
    PRIMARY KEY (ID),
    FOREIGN KEY (ID) REFERENCES Company_t(ID)
);

DROP VIEW IF EXISTS Company;
CREATE VIEW Company AS
SELECT
    c.ID AS ID,
    c.name AS name,
    c.created_timestamp AS created_timestamp,
    c.type AS type,
    s.rating AS rating,
    s.product_name AS product_name,
    c2.credit_limit AS credit_limit,
    c2.credit_rating AS credit_rating
FROM
    Company_t AS c
```

```
LEFT JOIN
    Supplier_t AS s ON c.ID = s.ID
LEFT JOIN
    Customer_t AS c2 ON c.ID = c2.ID;

DROP VIEW IF EXISTS Supplier;
CREATE VIEW Supplier AS
SELECT
    c.ID AS ID,
    c.name AS name,
    c.created_timestamp AS created_timestamp,
    s.rating AS rating,
    s.product_name AS product_name
FROM
    Company_t AS c
LEFT JOIN
    Supplier_t AS s ON c.ID = s.ID;

DROP VIEW IF EXISTS Customer;
CREATE VIEW Customer AS
SELECT
    c.ID AS ID,
    c.name AS name,
    c.created_timestamp AS created_timestamp,
    c2.credit_limit AS credit_limit,
    c2.credit_rating AS credit_rating
FROM
    Company_t AS c
LEFT JOIN
    Customer_t AS c2 ON c.ID = c2.ID;
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
0 rows affected.
0 rows affected.
(pymysql.err.OperationalError) (1359, 'Trigger already exists')
[SQL: CREATE TRIGGER before_company_t_update
BEFORE UPDATE on Company_t
FOR EACH ROW
BEGIN
    IF New.created_timestamp <> OLD.created_timestamp THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: Cannot change the value of the created_timestamp
field';
    END IF;
END;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

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

Write a relational algebra expression that produces a relation of the form

`(student_id, student_name, student_dept, advisor_id, advisor_name, advisor_dept)`

All students must have a row in the result, event if the student does not have an advisor. All instructors must appear in the result even if the instructor does not advise an students.

Follow the format you used for the previous homework, i.e. paste the text of your expression in the markdown cell below and include the execution result for your expression. You only need to include the first page of results if there is more than one page of results.

Text copy of your relational algebra expression.

π student_id←student.ID, student_name←student.name, student_dept←student.dept_name, advisor_id←advisor.i_id, advisor_name←instructor.name, advisor_dept←instructor.dept_name (instructor ⟕ instructor.ID=advisor.i_id (student ⋈ student.ID=advisor.s_id advisor))

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

student_id	student_name	student_dept	advisor_id	advisor_name	advisor_dept
12345	'Shankar'	'Comp. Sci.'	10101	'Srinivasan'	'Comp. Sci.'
null	*null*	*null*	*null*	'Wu'	'Finance'
null	*null*	*null*	*null*	'Mozart'	'Music'
44553	'Peltier'	'Physics'	22222	'Einstein'	'Physics'
45678	'Levy'	'Physics'	22222	'Einstein'	'Physics'
null	*null*	*null*	*null*	'El Said'	'History'
null	*null*	*null*	*null*	'Gold'	'Physics'
128	'Zhang'	'Comp. Sci.'	45565	'Katz'	'Comp. Sci.'
76543	'Brown'	'Comp. Sci.'	45565	'Katz'	'Comp. Sci.'
null	*null*	*null*	*null*	'Califieri'	'History'

SQL

Use the Classic Models database for these questions.

Write a SQL statement that produces a table of the form

`(country, productCode, productName, no_of_customers, number_of_orders)`

Where

- `country` is the `country` from the `customers` table.
- `product_code` and `product_name` has the obvious meaning.
- `no_of_customers` is the distinct number of customers that ordered the product.
- `number_of_orders` is the number of orders that contained an `orderdetails` with the `product_code.`

The result should show 0 for `number_of_customers` and `number_of_orders` when there have been no orders for a product by a customer in the country.

In [11]:
```
%%sql

USE classicmodels;
```

```sql
SELECT
    c.country AS country,
    p.productCode AS productCode,
    p.productName AS productName,
    COUNT(DISTINCT c.customerNumber) AS no_of_customers,
    COUNT(od.orderNumber) AS number_of_orders
FROM
    products p
LEFT JOIN
    orderdetails od ON p.productCode = od.productCode
LEFT JOIN
    orders o ON od.orderNumber = o.orderNumber
LEFT JOIN
    customers c ON o.customerNumber = c.customerNumber
GROUP BY
    c.country, p.productCode, p.productName
ORDER BY
    c.country, p.productCode
LIMIT 20;
```

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

Out[11]:

country	productCode	productName	no_of_customers	number_of_orders
None	S18_3233	1985 Toyota Supra	0	0
Australia	S10_1678	1969 Harley Davidson Ultimate Chopper	2	2
Australia	S10_1949	1952 Alpine Renault 1300	4	4
Australia	S10_2016	1996 Moto Guzzi 1100i	3	4
Australia	S10_4698	2003 Harley-Davidson Eagle Drag Bike	2	3
Australia	S10_4962	1962 LanciaA Delta 16V	3	4
Australia	S12_1099	1968 Ford Mustang	1	1
Australia	S12_1666	1958 Setra Bus	4	5
Australia	S12_2823	2002 Suzuki XREO	1	1
Australia	S12_3990	1970 Plymouth Hemi Cuda	1	1
Australia	S18_1097	1940 Ford Pickup Truck	4	5
Australia	S18_1129	1993 Mazda RX-7	2	2
Australia	S18_1342	1937 Lincoln Berline	4	4
Australia	S18_1367	1936 Mercedes-Benz 500K Special Roadster	4	4
Australia	S18_1589	1965 Aston Martin DB5	1	1
Australia	S18_1749	1917 Grand Touring Sedan	2	2
Australia	S18_1889	1948 Porsche 356-A Roadster	1	1
Australia	S18_1984	1995 Honda Civic	2	2
Australia	S18_2248	1911 Ford Town Car	2	2
Australia	S18_2319	1964 Mercedes Tour Bus	1	1

In [ ]: