EPPS 6354: Information Management

Assignments

Assignment 1

Answer 1,2,6 and one of the remaining questions.

1. **Name and describe three applications you have used that employed a database system to store and access persistent data. (e.g. airlines, online trade, banking, university system).**

   Online Banking: Most banking systems use a relational database to manage customer information, account transactions, and balances. They support functions like money transfers, loan management, and customer service, ensuring data is securely stored and accessible.
   E-commerce Platforms (Amazon): E-commerce platforms utilize databases to store data about products, user accounts, and purchase histories. This enables functions such as product browsing, cart management, order tracking, inventory management, and personalized recommendations.
   Airline Reservation Systems: Airlines use complex database systems to handle flight schedules, bookings, passenger details, and crew management. These systems need to be extremely efficient to handle the high volume of transactions and concurrent users.

2. **Propose three applications in domain projects (e.g. criminology, economics, brain science, etc.) Be sure you include:**
   a. **Purpose**
   b. **Functions**
   c. **Simple interface design**

   Criminology - Crime Mapping System
   Purpose: To assist law enforcement agencies in tracking crime patterns and hotspots.
   Functions: Allows users to input and query crime data, generate statistical reports, and view crime data on a map.
   Simple Interface Design: A map-based interface with filters for date, type of crime, and outcomes. Sidebar for statistical reports and data entry forms.
   Economics - Economic Data Analysis Tool
   Purpose: To analyze economic trends and forecast future economic conditions.
   Functions: Users can input economic indicators, generate trend graphs, and perform comparative analysis between different regions or time periods.
   Simple Interface Design: Dashboard with graphs and sliders for time range, economic indicators, and regions for analysis.
   Brain Science - Cognitive Decline Tracking Application
   Purpose: To monitor and analyze cognitive health over time for early detection of neurological disorders.

Functions: Allows users to input cognitive test results, track progress over time, and receive alerts on significant changes.
Simple Interface Design: User profile pages, test result input forms, and a timeline view of cognitive scores.

3. **If data can be retrieved efficiently and effectively, why data mining is needed?**

   Efficient and effective data retrieval refers to the ability to quickly access requested data from a database, which is well-managed and organized. However, data retrieval alone is often insufficient for several reasons, leading to the necessity of data mining:
   Discovering Patterns and Trends: Data mining goes beyond simple retrieval to analyze large datasets and identify hidden patterns, correlations, and trends that are not apparent through standard queries. For instance, data mining can reveal purchasing patterns that help businesses develop targeted marketing strategies.
   Predictive Analysis: While data retrieval can tell you what has happened, data mining applies machine learning algorithms and statistical methods to predict what might happen in the future. This is crucial in domains like finance for predicting stock trends, in healthcare for predicting disease outbreaks, or in retail for forecasting consumer behavior.
   Automating Decision-Making: Data mining automates the extraction of actionable insights from large datasets, supporting decision-making processes. For example, it can automate credit scoring for loan approval processes by analyzing applicant data against historical data.
   Handling Unstructured Data: A lot of valuable data today is unstructured (e.g., text, images, videos). Data mining includes techniques to analyze and derive meaningful information from unstructured data, which traditional database systems are not equipped to handle directly.
   Enhancing Data Quality: Data mining can also improve data quality by detecting anomalies and inconsistencies, which can then be corrected. This enhances the reliability of the data for all users and applications.
   In essence, while efficient data retrieval is about accessing the right data quickly, data mining is about making sense of this data and generating valuable insights that influence strategic decisions and predictions. This broader scope and deeper analysis are why data mining is essential even when data retrieval systems are effective.

4. **Why NoSQL systems emerged in the 2000s? Briefly contrast their features with traditional database systems.**
   (Left based on option given in instructions)
5. **What are the things current database system cannot do?**
   (Left based on option given in instructions)
6. **Describe at least three tables that might be used to store information in a social-network/social media system such as Twitter or Reddit.**

   Users Table: Stores information about each user, such as user ID, username, email address, password hash, and date of account creation.
   Posts Table: Contains details about posts made by users, including post ID, content, timestamp, user ID of the poster, and potentially a reference to parent posts (for replies).

Relationships Table: Manages follower/following relationships between users, with columns for follower ID and followed ID, each linking back to the Users table.

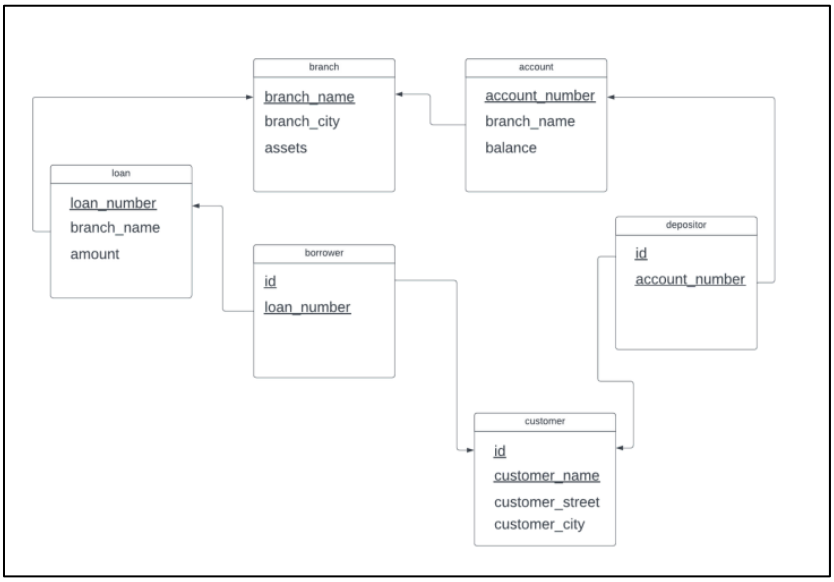
Assignment 2

1. **What are the differences between *relation schema, relation and instance*? Give an example using the university database to illustrate.**

   Relation schema defines the structure of a relation, including its name and the attributes it contains. It is essentially the blueprint for a table in a database, specifying the columns and their data types. Student(ID, Name, Major)
   In the context of a database, a relation is often used interchangeably with the term "table." It is a set of tuples that share the same attributes according to the relation schema. Example, a table named 'Student'.
   An instance of a database is a snapshot at a particular moment in time, containing the actual data entries in the database tables. It represents the state of the physical database. Example from university database: At a particular moment, this table might contain:
   (001, Alice, Computer Science)
   (002, Bob, Mathematics)
   (003, Carol, History)

2. **Draw a schema diagram for the following bank database:**

   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*)



3. **Consider the above bank database. Assume that branch names (branch_name) and customer names (customer_name) uniquely identify branches and customers, but loans and accounts can be associated with more than one customer. What are the appropriate primary keys? (Underline each in diagram)**

Based on the schema and given constraints:
Customers: Primary Key – id, customer_name
Depositor: account_number
Borrower: loan_number
Loan: loan_number
Branch: branch_name

4. **Given your choice of primary keys, identify appropriate foreign keys.**
   Account_number
   Loan_number

Assignment 3

1. **Open the Online SQL interpreter (https://www.db-book.com/db7/university-lab-dir/sqljs.html)**
   Task completed.

2. **Write SQL codes to get a list of:**
   a. **Student IDs (hint: from the *takes* relation)**
      SELECT DISTINCT ID
      FROM takes;
   b. **Instructors**
      SELECT *
      FROM instructor;
   c. **Departments**
      Select *
      FROM department;

3. **Write in SQL codes to do following codes:**
   a. **Find the ID and name of each student who has taken at least one Comp. Sci. course; make sure there are no duplicate names in the result.**
      SELECT DISTINCT s.ID, s.name
      FROM student s
      JOIN takes t ON s.ID = t.ID
      JOIN course c ON t.course_id = c.course_id
      WHERE c.dept_name = 'Comp. Sci.';

ID	name
00128	Zhang
12345	Shankar
45678	Levy
54321	Williams
76543	Brown
98765	Bourikas

   b. **Add grades to the list**
      SELECT DISTINCT s.ID, s.name, t.grade
      FROM student s
      JOIN takes t ON s.ID = t.ID
      JOIN course c ON t.course_id = c.course_id
      WHERE c.dept_name = 'Comp. Sci.';

ID	name	grade
00128	Zhang	A
00128	Zhang	A-
12345	Shankar	C
12345	Shankar	A
45678	Levy	F
45678	Levy	B+
45678	Levy	B
54321	Williams	A-
54321	Williams	B+
76543	Brown	A
98765	Bourikas	C-
98765	Bourikas	B

c. **Find the ID and name of each student who has not taken any course offered before 2017.**

```
SELECT DISTINCT s.ID, s.name
FROM student s
WHERE NOT EXISTS (
        SELECT 1
        FROM takes t
        JOIN section sec ON t.course_id = sec.course_id
        WHERE s.ID = t.ID AND sec.year < 2017
);
```

ID	name
00128	Zhang
12345	Shankar
19991	Brandt
23121	Chavez
44553	Peltier
45678	Levy
54321	Williams
55739	Sanchez
70557	Snow
76543	Brown
76653	Aoi
98765	Bourikas
98988	Tanaka

d. **For each department, find the maximum salary of instructors in that department. You may assume that every department has at least one instructor.**

```
SELECT d.dept_name, MAX (i.salary) as max_salary
FROM department d
JOIN instructor i ON d. dept_name = i.dept_name
GROUP BY d.dept_name
```

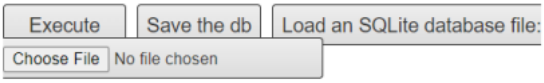
dept_name	max_salary
Biology	72000
Comp. Sci.	92000
Elec. Eng.	80000
Finance	90000
History	62000
Music	40000
Physics	95000

e. **Find the lowest, across all departments, of the per-department maximum salary computed by the preceding query.**

```
SELECT MIN(max_salary) as lowest_max_salary
FROM (
        SELECT Max (i.salary) as max-salary
        FROM instructor i
        GROUP BY i.dept_name
) as max_salaries;

( d.dept_name, MAX (i.salary) as max_salary
FROM department d
JOIN instructor i ON d. dept_name = i.dept_name
GROUP BY d.dept_name
```

f. **Add names to the list**



dept_name	name	max_salary
Biology	Crick	72000
Comp. Sci.	Brandt	92000
Elec. Eng.	Kim	80000
Finance	Wu	90000
History	Califieri	62000
Music	Mozart	40000
Physics	Einstein	95000

4. **Find an instructor (with name and ID) who has never given an A grade in any course she or he has taught. (Instructors who have never taught a course trivially satisfy this condition.)**

Enter SQL commands here

```
SELECT i.ID, i.name
FROM instructor i
WHERE NOT EXISTS (
    SELECT 1
    FROM teaches t
    JOIN takes tk ON t.course_id = tk.course_
    WHERE i.ID = t.ID AND tk.grade = 'A'
);
```

Execute | Save the db | Load an SQLite d

Choose File | No file chosen

ID	name
12121	Wu
15151	Mozart
22222	Einstein
32343	El Said
33456	Gold
45565	Katz
58583	Califieri
76543	Singh
98345	Kim

5. **Write SQL query to find the number of students in each section. The result columns should appear in the order "courseid, secid, year, semester, num". You do not need to output sections with 0 students.**
   Select course_id, sec_id, year, semester, count(ID) as num From takes Group by course_id, sec_id, year, semester;

Assignment 4

1. **Explain the difference between a weak and a strong entity set. Use an example other than the one in Chapter 6 to illustrate. (Consult Ch. 6, 6.5.3)**
   Strong Entity Set: This is an entity that can exist independently of other entities. It has a primary key that uniquely identifies its instances.
   Weak Entity Set: This entity cannot exist independently without a certain other entity, known as its owner. It does not have a primary key that can uniquely identify its instances without considering the primary key of the owner entity.
   Example: Imagine an online bookstore. A "Customer" entity is a strong entity because each customer can be uniquely identified by a customer ID, regardless of any other entity. In contrast, an "Order" entity might be considered a weak entity if it cannot be uniquely identified without associating it with a specific customer. The order might have an order ID, but it only makes sense in conjunction with the customer who placed it.

2. **Design an E-R diagram for keeping track of the scoring statistics of your favorite sports team. You should store the matches played, the scores in each match, the players in each match, and individual player scoring statistics for each match. Summary statistics should be modeled as derived attributes with an explanation as to how they are computed. (Consult: Practice Exercise solutions on textbook website)**
   a. **Draw the E-R diagram using draw.io. Read this website for instructions.**
   b. **Expand to all teams in the league (Hint: add team entity)**

   Entities:
   Team: Attributes include Team_ID, Name.
   Player: Attributes include Player_ID, Name, Team_ID (foreign key).
   Match: Attributes include Match_ID, Date, Location.
   Score: Attributes include Match_ID (foreign key), Player_ID (foreign key), Points_Scored.
   Relationships:
   Plays (between Team and Match): Attributes include Team_Score.
   Participates (between Player and Match): No additional attributes.
   Scores (between Player and Match): Attributes include Points.
   Derived Attributes:
   Player's average scores can be derived by dividing the total points scored by the number of matches played.
   Team's average score can be derived similarly.
   For expanding to all teams:
   Simply ensure that the Team entity is related to Matches through a Plays relationship. This allows tracking of all matches any team plays in the league.

3. **SQL exercise:**
   a. **Consider the query**
      select course_id, semester, year, sec_id, **avg** (tot_cred)
      from takes **natural join** student
      where year = 2017
      group by course_id, semester, year, sec_id

having count (ID) >= 2;

i. **Explain why appending natural join section in the from clause would not change the result. (Consult Ch. 4, 4.1.1)**
   Appending
   Appending a natural join section would not change the result because the section table contains the primary keys used in the group by clause. Since these are already unique in the takes table as used in the query, joining with section does not introduce any new rows or change the aggregation. Appending natural join section would not change the results, as the relations takes, and student already have shared columns with the same values.

ii. **Test the results using the Online SQL interpreter (https://www.db-book.com/db7/university-lab-dir/sqljs.html)**

b. **Write an SQL query using the university schema to find the ID of each student who has never taken a course at the university. Do this using no subqueries and no set operations (use an outer join). (Consult Ch. 4, 4.1.3)**

```
SELECT DISTINCT
    A.ID,
    NAME
FROM
    STUDENT A
LEFT OUTER JOIN
    TAKES B
    ON
    A.ID <> B.ID
```

Execute | Save the db | Load an SQLite database file:
Choose File | no file selected

ID	name
00128	Zhang
12345	Shankar
19991	Brandt
23121	Chavez
44553	Peltier
45678	Levy
54321	Williams
55739	Sanchez
70557	Snow
76543	Brown
76653	Aoi
98765	Bourikas
98988	Tanaka

c. **Consider the following database, write a query to find the ID of each employee with no manager. Note that an employee may simply have no manager listed or may have a null manager (use natural left outer join). (Consult Ch. 4, 4.1.3)**

---

employee (*ID*, *person_name*, *street*, *city*)
works (*ID*, *company_name*, *salary*)
company (*company_name*, *city*)
manages (*ID*, *manager_id*)

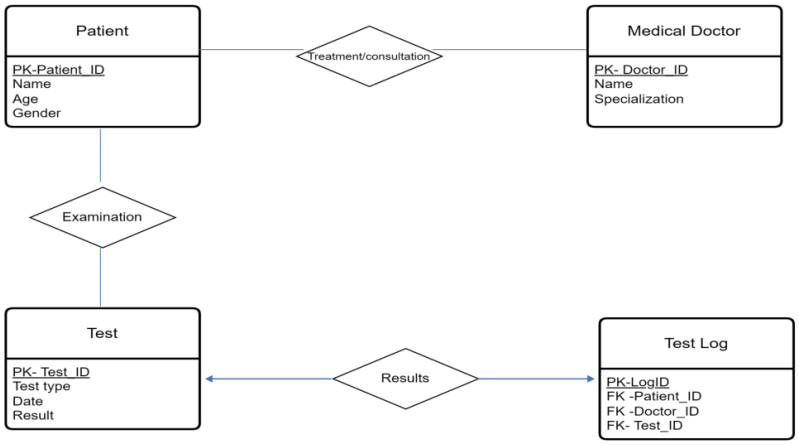
---

select e.id
from employee as e natural left outer join manages as m
where m.manager_id is null;

Assignment 5

1. **An E-R diagram can be viewed as a graph. What do the following mean in terms of the structure of an enterprise schema?**
    a. **The graph is disconnected.**
    In an E-R diagram, a disconnected graph means that there are two or more entity sets (or possibly sub-schemas) that have no direct or indirect relationships connecting them. This might imply that different parts of the enterprise are completely independent of each other, suggesting potential modular design or areas with no data integration.
    b. **The graph has a cycle.**
    A cycle in an E-R diagram indicates that there are circular dependencies where some entity sets are connected in a way that one can circle back to the original entity set via relationships. This can lead to complexities in implementation and may affect the integrity of navigational queries, potentially causing infinite loops or redundancy in data storage.

2. **Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted (Hint: use miro.com to draw the diagram with relationship sets).**



3. **We can convert any weak entity set to a strong entity set by simply adding appropriate attributes. Why, then, do we have weak entity sets?**
    Weak Entity Sets exist because they logically depend on other entities for their identity. They are used to model situations where the entity's existence is contingent upon another entity. Although technically any weak entity can be made strong by adding a unique identifier, doing so might not make sense logically or practically if those entities do not have a meaningful independent existence or if such an identifier would not naturally occur within the business context.

4. **SQL exercise:**
    a. **Consider the employee database**

employee (*ID*, *person_name*, *street*, *city*)
works (*ID*, *company_name*, *salary*)
company (*company_name*, *city*)
manages (*ID*, *manager_id*)

Where the primary keys are underlined. Give an expression in SQL for each of the following queries. (Hint: use from *employee* as e, works as w, company as c, manages as *m*)

i. **Find ID and name of each employee who lives in the same city as the location of the company for which the employee works.**

```
SELECT e.ID, e.person_name
FROM employee e
JOIN works w ON e.ID = w.ID
JOIN company c ON w.company_name = c.company_name
WHERE e.city = c,city;
```

ii. **Find ID and name of each employee who lives in the same city and on the same street as does her or his manager.**

```
SELECT
SELECT e.ID, e.person_name
FROM employee e
JOIN employee m ON e.manager_id = m.ID
WHERE e.city = m.city AND e.street = m.street;
```

iii. **Find ID and name of each employee who earns more than the average salary of all employees of her or his company.**

```
SELECT e.ID, e.person_name
FROM employee e
JOIN works w ON e.ID = w.ID
JOIN (
        SELECT company_name, AVG(salary) AS avg_salary
        FROM works
        GROUP BY company_name
) AS avg_salaries ON w.company_name = avg_salaries.company_name
WHERE w.salary > avg_salaries.avg_salary;
```

b. **Consider the following SQL query that seeks to find a list of titles of all courses taught in Spring 2017 along with the name of the instructor.**

```
select name, title
from instructor natural join teaches natural join section natural join course
where semester = 'Spring' and year = 2017
```

What is wrong with this query? (Hint: check book website)

Problem is that the use of natural join assumes that the common attribute names across the tables instructor, teaches, section, and course uniquely identify records, which might not be the case. For instance, name might be duplicated in

different sections. This could lead to erroneous pairings or duplications in the results. Specify explicit join conditions and select attributes to avoid ambiguity, ensuring that joins are made on the correct keys.

Assignment 6

1. **Look up websites containing the following data representations:**
   a. **Using JSON**
      Facebook's API employs JSON to facilitate data exchange between the web application and the server. When data is requested from the API, such as user profile information, the response is usually formatted as JSON strings. Facebook has developed the TAO data model (The Associations and Objects) to effectively represent the intricate connections among various entities, utilizing MySQL as the underlying database.

   b. **Using XML**
      SOAP web services, commonly used in financial institutions, rely on XML for data interchange. XML's structured format, which includes elements, attributes, and hierarchical relationships, is well-suited for the complex data interactions typical in banking applications. These services often adhere to detailed schemas that define the structure and rules of the XML data used. Systems built with technologies like Java EE or ASP.NET frequently implement these services, ensuring robust data management and integration capabilities.

   Analyze these websites in terms of structure and composition. Name the technology/methods used for creating the web database.
   For both data types, the underlying database technology could vary widely, but often involves relational databases such as MySQL, PostgreSQL, or non-relational databases like MongoDB, especially when JSON is heavily used.

2. **SQL exercise**
   a. **Express the following query in SQL using no subqueries and no set operations (Hint: left outer join)**
      select ID
      from student
      except
      select s id
      from advisor
      where i ID is not null
      SELECT s.ID
      FROM student
      LEFT JOIN advisor ON s.ID = advisor.s_id
      WHERE advisor.i_id IS NULL;

   b. **Using the university schema, write an SQL query to find the names and IDs of those instructors who teach every course taught in his or her department (i.e., every course that appears in the course relation with the instructor's department name). Order result by name.**
      SELECT I.name, I.ID

```
FROM instructor I
JOIN teaches T ON I.ID = T.instructor_id
JOIN course C ON T.course_id = C.course_id
GROUP BY I.name, I.ID, C.dept_name
HAVING COUNT(DISTINCT C.course_id) = (SELECT COUNT(DISTINCT course_id)
FROM course WHERE dept_name = I.dept_name)
ORDER BY I.name;
```

name ID
Brandt 83821
Crick 76766
Einstein 22222
El Said 32343
Katz 45565
Kim 98345
Mozart 15151
Srinivasan 10101
Wu 12121