F24-W4111: Homework 1A

Overview

- Check CourseWorks and Ed for submission instructions, deadlines, clarifications, etc.
- This homework is a set of written questions testing knowledge and review of concepts covered in:
  - The lectures, both the presentations and any information communicated during the lecture.
  - The slides that accompany the recommended textbook.
    - URL: https://db-book.com/slides-dir/index.html
    - You only need to reference the slides for chapters 1, 2, 3, 6, and 7.

Concepts

1. List purposes for/motivations for database systems compared to managing data by writing applications that process files.

Data Redundancy and Inconsistency: With file systems, data is often stored in multiple formats, leading to duplication and inconsistency. A database system centralizes the data and reduces redundancy.

Difficulty in Accessing Data: In a file system, writing new programs is necessary for every new task. Database systems provide a more flexible way to access and manipulate data without constant programming.

Data Isolation: File systems store data in multiple files and formats, making it hard to aggregate data. Database systems organize data into structured formats that are easier to query and relate.

Integrity Problems: Integrity constraints (e.g., account balance must be greater than zero) are typically embedded in application code in a file system, making it hard to enforce or modify them. In a database, these constraints can be explicitly defined and enforced by the system.

Atomicity of Updates: Failures during updates can leave file-based systems in inconsistent states. Database systems ensure that either all or none of a transaction's updates are completed.

Concurrent Access by Multiple Users: File systems struggle with concurrent access, often leading to data inconsistencies. Database systems are designed to handle concurrent access, ensuring data integrity.

Security Problems: File systems make it hard to manage different levels of access control for users. Database systems provide robust security mechanisms for managing user permissions at a granular level.

2. Database systems provide users with an abstraction view of the data.

a. Briefly explain the concept.

Database systems provide users with an abstraction layer, meaning that users can interact with the data without needing to know the complexities of how the data is stored or how the operations are performed internally. This abstraction is typically achieved through data models (e.g. elational models) and views that hide the implementation details. Users can query or update the database using high-level languages like SQL, and the database system handles tasks such as data storage, retrieval, optimization, and transaction management in the background.

For example, a user querying a table in a relational database does not need to know the physical storage location of the table or the specific algorithms used to retrieve the data. The system abstracts these complexities, allowing users to focus on what data they need rather than how it is obtained.

b. Why do writing applications that access files do not provide an abstraction?

Writing applications that access files directly do not offer this level of abstraction because they require programmers to handle all the details of data management manually. In such systems, the developer must manage:

1. File formats: Programmers need to know the specific file structure and ensure consistency across files.
2. Data access logic: Every task, such as reading, writing, or updating data, needs a custom-built function that interacts directly with the file system.
3. Concurrency and Integrity: Applications need to manage multiple users accessing files at the same time and ensure data integrity without built-in mechanisms like those in database systems.
4. Error handling: Any failure during an operation (e.g., partial writes) requires complex error-handling logic to maintain consistency.

Without abstraction, applications must explicitly address the "how" of every data operation, leading to a higher chance of errors, inefficiency, and complexity.

3. What are the 3 levels of data abstraction that a DBMS provides?

View Level, Logical level, and Physical level.

4. Explain the difference between database schema and database instance. What two concepts in an object-oriented language correspond to schema and instance?

Database Schema: A schema is the overall logical structure or blueprint of the database. It defines the organization of data and the relationships between tables, fields, and data types. The schema remains relatively unchanged over time, as it defines the structure rather than the data itself.

Database Instance:

An instance is the actual content of the database at a specific point in time. It refers to the data stored in the database according to the schema. The instance changes frequently as data is added, updated, or deleted.

In object-oriented programming, the concepts that correspond to schema and instance are:

Class (Schema): A class in an object-oriented language defines the structure of an object, including the attributes (fields) and methods (functions) that objects of this class will have. Like a schema, a class is a blueprint for creating objects but does not represent any specific data.

Object (Instance): An object is an instance of a class, meaning it is a specific entity created based on the class's definition. The object holds actual data (values in its attributes), just like a database instance holds the actual data according to the schema.

5. Briefly describe the concepts of data definition language and data manipulation language. What are the two types of data manipulation language?

Data definition language: DDL is a set of commands used to define the database schema. It allows users to specify the structure of the data in a database, including creating (Create Table), altering(Alter Table), and deleting (Drop Table) database objects like tables, indexes, and constraints. When DDL commands are executed, they result in changes to the metadata (data about data) stored in the database.

Data manipulation language: DML consists of commands used to retrieve, insert, modify, and delete data within the database. While DDL defines the structure, DML is focused on working with the data itself. It allows users and applications to manipulate the data according to the schema defined by DDL. Eg: select, insert, delete, update

Two types of DML:
Procedural DML -- require a user to specify what data are needed and how to get those data.
Declarative DML -- require a user to specify what data are needed **without specifying how to get those data.** E.g: SQL query to get the data

6. Briefly explain two-tier and three-tier database application architectures. Is a full-stack web application a two-tier architecture or a three-tier architecture?

Two-tier architecture -- the application resides at the client machine, where it invokes database system functionality at the server machine. It involves:

1. Client Tier (Presentation Layer): The client machine, which directly interacts with the user, runs the application. It contains the user interface and some business logic.
2. Server Tier (Database Layer): The database resides on the server, and the application directly communicates with the database system. The client sends queries to the database, which processes the requests and returns the data.

Three-tier architecture -- the client machine acts as a front end and does not contain any direct database calls. It involves:

1. Client Tier (Presentation Layer): This tier interacts with the user and handles input and output. Typically, this is a web browser or mobile application.
2. Application Tier (Logic Layer): This middle layer contains the business logic and acts as an intermediary between the client and the database. It processes client requests, runs business rules, and sends queries to the database. This is often a web server or application

server.

3. Database Tier (Data Layer): This tier stores the data and handles queries. It is typically a database management system running on a separate server.

A full-stack web application is generally considered a three-tier architecture because it typically involves Front-End (Client Tier), Back-End (Application Tier), and Database (Data Tier): The database system that stores and retrieves the data.

7. What are the four types of database users based on skill level and for what they use the database. Which type of user defines schema and defines what information users can access?
   1. Naive Users (End Users):
      These users interact with the database through application interfaces or forms without writing queries or interacting directly with the database. They perform simple tasks like data entry, viewing reports, or running pre-built queries.
   2. Application Programmers:
      These are skilled professionals who write application programs that access the database. They use APIs or embedded SQL to interact with the database, providing an interface or functionality for naive users to perform tasks.
   3. Sophisticated Users:
      These users interact directly with the database system using query languages (like SQL) to perform complex queries, analytics, and data manipulation. They typically do not write application programs but may use tools like data analysis software to extract insights from the database.
   4. Database Administrators (DBAs):
      DBAs are the type of users who define the schema and control what information users can access. They are responsible for creating and managing the database structure, enforcing security measures, setting access permissions, and ensuring the overall efficiency and integrity of the database.

Database Administrators play a crucial role in defining both the logical structure of the database (schema) and controlling user access to different parts of the database

Relational Model

Understanding Data

The following is data in the SSOL format for our course COMS W4111 but with made up values for Student Name, Student UNI, Student PID, Email, School, Level, Affiliation, Points.

student_name	uni	student_pid	email	school	level	affiliation	points
Ferguson, Donald F	dff9	C001234567	dff9@.columbia.edu	CC	U03	CCCOMS	3
Baggins, Frodo	fb001	C001234889	frodo.baggins@shire.gov	EN	U03	ENCOMS	3
Romanoff, Natasha	nr2103	C009999999	nr2103@barnder.com	BC	U03	BCCOMS	3
Prince, Diana	dp2121	C007171717	wonderwoman@avengers.org	BC	U03	BCCOMS	3
Wayne, Bruce	bw9101	C008121212	batman@justice-league.org	GS	U04	GSCOMA	3
Potter, Harry James	hjp1001	C009898989	hjp1001@columbia.edu	BU	P04	BUEXMS	3

Answer the following questions based on your general knowledge of Columbia University and examples given in class.

1. Which attributes are not from an atomic domain?
   Student_name
2. Which attributes are likely to be candidate keys?
   Uni and student_pid
3. Which attributes are likely to be foreign keys into another relation?
   School, level, affiliation are likely to be foreign key to connect with tables store school information, student year information, and student affiliation information.
4. Which attribute is clearly a surrogate key?
   There is no clearly surrogate key in this table because all columns have underlying meaning.

An Algebra

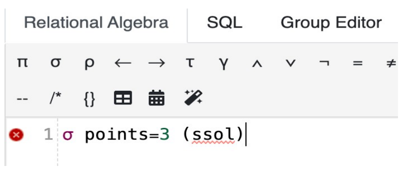
“The result of a relational-algebra operation is a relation and therefore relational-algebra operations can be composed together into a relational-algebra expression.”

Assume that the name of the relation above is SSOL. Give two relational algebra expressions that are examples of composition that when applied to the relation above produce the following relation.

student_name	uni	school
Romanoff, Natasha	nr2103	BC
Prince, Diana	dp2121	BC

You can use the relax calculator to type your expressions, and then copy/paste the expressions into your answers. The syntax check will indicate errors but you are not executing the expressions.

For example,

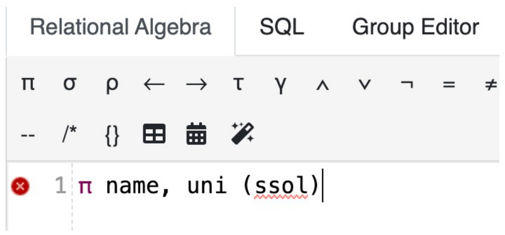


```
σ points=3 (ssol)
```

shows how to type an expression that is a select which matches all rows in the original relation.

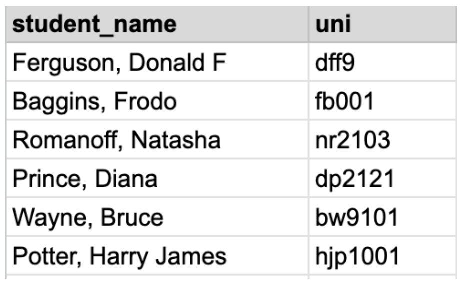
π student_name, uni, school (σ school = 'BC' (SSOL))

And,



```
π name, uni (ssol)
```

shows how to type an expression that produces



student_name	uni
Ferguson, Donald F	dff9
Baggins, Frodo	fb001
Romanoff, Natasha	nr2103
Prince, Diana	dp2121
Wayne, Bruce	bw9101
Potter, Harry James	hjp1001

π student_name, uni, school (σ uni = 'nr2103' ∨ uni = 'dp2121' (SSOL))

Schema

Translate this relation schema definition into an SQL create table statement. This question uses the notation from class slides.

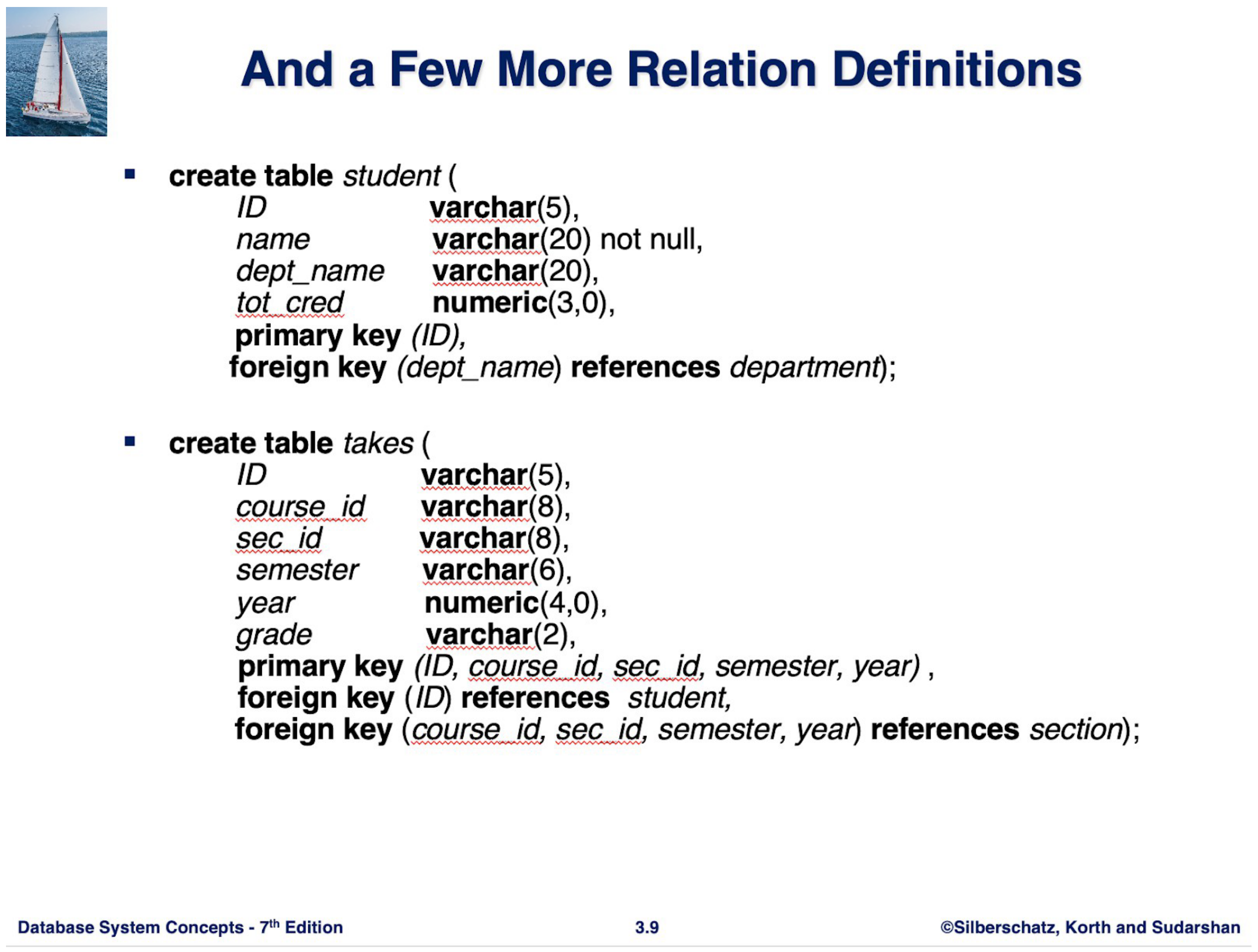
instructor = (<u>UNI</u>, last_name, first_name)

```
CREATE TABLE instructor (
   UNI CHAR(10) PRIMARY KEY,
   last_name VARCHAR(255),
   first_name VARCHAR(255)
);
```

Entity Relationship Modeling

Reverse Engineering

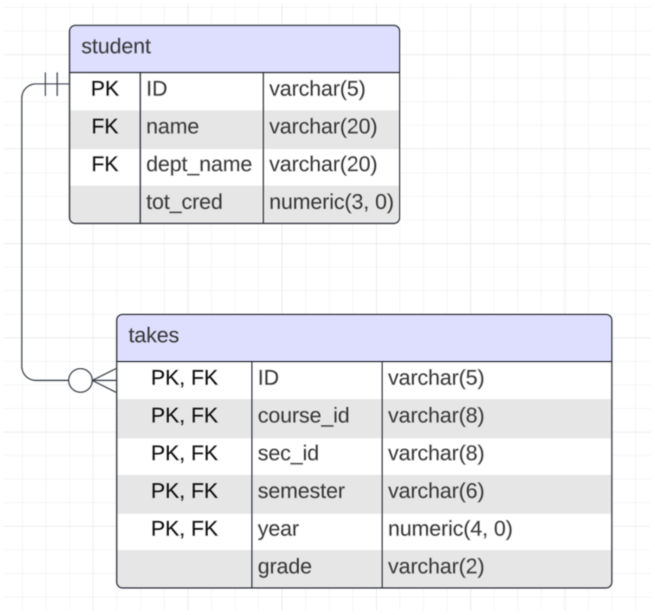
The following slide is from the slides associated with the recommended textbook.



And a Few More Relation Definitions

- **create table** *student* (
```
        ID              varchar(5),
        name            varchar(20) not null,
        dept_name       varchar(20),
        tot_cred        numeric(3,0),
        primary key (ID),
        foreign key (dept_name) references department);
```
- **create table** *takes* (
```
        ID              varchar(5),
        course_id       varchar(8),
        sec_id          varchar(8),
        semester        varchar(6),
        year            numeric(4,0),
        grade           varchar(2),
        primary key (ID, course_id, sec_id, semester, year) ,
        foreign key (ID) references  student,
        foreign key (course_id, sec_id, semester, year) references section);
```

Use LucidChart to draw an ER diagram in Crow's Foot Notation that is a reverse-engineered, logical model of the schema. You only need to do the entities *student* and *takes*. Do not worry about other referenced entities.



student		
PK	ID	varchar(5)
FK	name	varchar(20)
FK	dept_name	varchar(20)
	tot_cred	numeric(3, 0)

takes		
PK, FK	ID	varchar(5)
PK, FK	course_id	varchar(8)
PK, FK	sec_id	varchar(8)
PK, FK	semester	varchar(6)
PK, FK	year	numeric(4, 0)
	grade	varchar(2)

Pros and Cons

List some advantages and disadvantages of ER Modeling.

Advantages of ER Modeling:

1. Simplicity: The ER model is conceptually straightforward. Once the relationships between entities and attributes are understood, creating an ER diagram is easy.

2. Clear Visual Representation: It provides a diagrammatic view of the logical structure of the database, making it easier to grasp the relationships between entities.

3. Effective for Communication: The ER diagram serves as an effective tool for communication among database designers and stakeholders.

4. Strong Integration with Relational Models: The ER model can be easily converted into a relational model by transforming the entities and relationships into database tables.

5. Easy Conversion to Other Models: It allows for easy conversion to other database models, such as hierarchical or network data models.

Disadvantages of ER Modeling:

1. Limited Constraints and Specifications: The ER model may not fully capture all constraints or specifications needed for complex databases.

2. Loss of Information: Certain details or information may not be fully represented or can be hidden in the ER model.

3. Limited Representation of Relationships: Compared to other models like the relational model, the ER model provides a more limited representation of relationships between entities.

4. No Data Manipulation Representation: The ER model doesn't effectively represent data manipulation operations.

5. Best for High-Level Design: The ER model is mainly useful for high-level design and may not be as effective for detailed database design.

6. Lack of Standard Notation: There is no universal industry-standard notation for ER diagrams, leading to possible inconsistencies in interpretation.