Final Exam: Written Questions

W4111 – Introduction to Databases: 20231COMS4111W002

Ziggy Chen, xc2605

Guidance and Submission Instructions

Foundation

F1 – Benefits of DBMS

Before DBMS, organizations managed data by writing application programs that manipulated data in files. List 5 problems with the application-file approach and succinctly state of DBMS solve the problem.

----------Answer----------
Problems with the application-file approach

1. **Data Redundancy** (data is usually stored across multiple files and has duplicates, leading to inconsistency or large storage size requirement):
   DBMS solves the problem by centralising the data and allowing data to be shared between different files.
2. **Simultaneous Access Problem** (simultaneous access and manipulation to those files may cause data inconsistencies):
   DBMS has concurrency control features that guarantee data integrity and consistency.
3. **Data Access Difficulty** (complex codes are required to retrieve and manipulate data in files, leading to data access difficulty).
   DBMS offers a standardised query language (SQL) that simplifies data access and manipulation for all applications, making everything more accessible and consistent.
4. **Data Security** (those files can be accessed or manipulated easily, leading to data security concerns).
   DBMS offers various security features, including access control, encryption and auditing, to protect the database and make sure that only authorised users can manipulate the data.
5. **Data Isolation** (data is stored in different files and/or formats. This makes it challenging to combine and analyse the data for users):
   DBMS uses schema to well-define the data format, making it easier for users to integrate and/or retrieve data from various sources.

F2 – Types of Data

Briefly explain structured data, semi-structured data and unstructured data. Give an example of each type of data.

----------Answer----------

1. **Structured Data**: structured data should have a well-defined structure. That is, individual data of the same type contains the same attribute set AND fields. So it can be easily stored AND/OR searched. Examples can be a Google Sheet or an Excel file.
2. **Unstructured Data**: unstructured data does NOT show any structure. It contains data items that may have an entirely different format stored in their original formats. Examples can be a collection of images or a collection of videos.
3. **Semi-structured Data**: for semi-structured data, data items of the same type may have different attribute sets. Examples can be JSON or Extensible Markup Language (XML).

F3 – Physical Data Independence

What is physical data independence? What is a benefit?

Definition: Physical data independence means that the system can be modified at a physical level without leading to application programs being rewritten.
Benefit: We can optimise physical performance, upgrade storage or change the physical storage solutions without impacting the application layer, which minimises the maintenance cost and disruptions of a system and makes it easier for hardware upgrades.

F4 – Concepts

Explain the following concepts and give an example of each:
1. Data manipulation language (DML)
2. Data definition language (DDL)
3. Procedure DML
4. Declarative DML

----------Answer----------

Data manipulation language (DML): Users can use a data manipulation language (DML) to access or manipulate a database as organised by the corresponding data model. An example can be:

```
select instructor.name
from instructor
where instructor.dept_name = 'History';
```

Data definition language (DDL): Users define a database schema by a series of codes expressed by a data-definition language (DDL). It is also used to specify specific properties of the data model. An example can be:

```
create table department
    (dept_name      char (20),
     building       char (15),
     budget         numeric (12,2));
```

Procedure DML: This specific type of DML requires users to what data are needed and the approach to get the desired data. Examples can be FORTRAN, C or BASIC.

Declarative DML: This specific type of DML does not require users to specify how to get the data. Users only need to specify what data are needed. Examples can be SQL, PROLOG or LISP.

Reference: The DB book P13-16 and https://www.geeksforgeeks.org/difference-between-procedural-and-non-procedural-language/.

F5 – Modeling

Briefly explain the following concepts and the role for each concept in data modeling. Give an example of the benefit of each level:

1. Conceptual model
2. Logical model
3. Physical model

----------Answer----------

Conceptual model:

1. Explanation: it only contains entity names and relationships.
2. Role: it is a high-level overview of a database system for audiences with limited data modelling knowledge, such as stakeholders.
3. Benefit Example: the conceptual model can give the stakeholders a high-level intuition about the database system, making it easier for decision-making and efficient system development.

Logical model:

1. Explanation: it includes entity names, entity relationships, attributes, primary keys, and foreign keys.
2. Role: it is for clarification and more detailed relationships between database entities.
3. Benefit Example: the logical model guarantees integrity and consistency of database design. Therefore, the mapping processes from the conceptual model to the actual implementation becomes smoother.

Physical model:

1. Explanation: a physical model has primary keys, foreign keys, table names, column names, and column datatypes. It is for technical implementation purposes.
2. Role: the major audiences are database administrators and database developers. It has all the detailed implementation information, such as data type.
3. Benefit Example: The physical model allows for more efficient data storage, access and management, enabling performance optimisation or resource utilisation optimisation for an existing database system.

F6 – Application Architectures

Briefly explain:

1. Two-tier database application architecture
2. Three-tier application architecture

----------Answer----------

Two-tier: the application is located on the client machine and utilises the database system's features on the server machine by executing query language statements.

Three-tier: the client machine serves solely as a front-end interface, without making any direct database interactions. Web browsers and mobile apps are the most prevalent types of application clients.

Reference: The DB book P23.

F7 – Database Administrators (DBA)

List 5 tasks/functions that a DBA performs. Do DBAs typically use DDLs or DMLs?

----------Answer----------

Schema definition: they execute data definition statements in the DDL to create the original database schema.

Storage structure and access-method definition: parameters that are relevant to the physical organisation of the data and the indices to be created may be specified by them.

Schema and physical-organisation modification: they modify the schema and physical organisation to reflect the changing needs of the organisation, OR they alter the physical organisation to improve the performance of the database.

Granting of authorisation for data access: They manage access levels of the database. That is, they regulate which part of the database can be accessed by various users.

Routine maintenance: They should be responsible for routine maintenance such as backup management, available disk space check and performance/user monitoring.

DBAs may use both DDLs and DMLs, *but DBAs typically use DDLs much more frequently than DMLs.* DDLs are used for creating, altering or deleting database structures (**these are their main tasks**), while DMLs are used for data manipulation tasks like insertion, retrieval or modification.

Reference: P24-25 of the DB book.

Relational Model

R1 – Domain

Explain the importance of atomicity of domains. *float* is a type. An example of a *domain* might be a person's *weight*. The type for *weight* might be a float, but give an example of how *float* is not the *domain.*

----------Answer----------
Importance: it ensures all domains represent a single, indivisible value, which maintains data consistency and integrity. This simplifies data manipulation and querying.

We know that a person usually has a weight between 5.0 kg to 200.0 kg, so we can use the float type to represent a person's weight, and the range of 5.0-200.0 is the domain of a person's weight. But it is not very possible that a person can have a weight of 800.0 kg, which is also of float type. So, float is not the domain (not all values of float are within the range), say it is also possible can a float can be negative but this is not realistic for a person's weight.

R2 – Keys

Briefly define and explain the following concepts:

1. Superkey
2. Candidate key
3. Primary key
4. Foreign key

----------Answer----------

Here we will use examples for the explanation. Please refers to P43-45 of the DB book.

Super key: a super key is a set containing one or more attributes that allow us to identify a unique tuple in the relation. For example, in the instructor table (Figure 2.4), the set of ID AND name is enough to identify a unique instructor in the relation.

Candidate key: minimal super keys are called candidate keys. That is, no smaller subsets of a candidate key can be super key. From the previous example, we may use ID as a candidate key since this attribute is enough to identify a unique instructor, and NO smaller subset can do the same job.

Primary key: a primary key is a candidate key chosen by the database designer to identify unique tuples within the relation. Again, for the same instructor table, we see that in Figure 2.8, ID is underlined, and it is the chosen primary key by the designer to identify unique instructors.

Foreign key: a foreign key is a key in a table referencing the primary key in another table so that they are linked together. See Figure 2.8 for the student table, dept_name is a foreign key in the student relation referencing to the department relation.

Reference: The DB book P43-45.

R3 – Operators

The slides associated with the recommended text book list six basic relational operators

- select: $\boldsymbol{\sigma}$
- project: $\boldsymbol{\pi}$
- union: U
- set difference: –
- Cartesian product: x
- rename: ρ

Surprisingly, the list does not include *join:* $\bowtie$. This is because join it is possible to derive join from a relational expression using more basic operators.

Briefly explain how to derive join from basic operators.

What is the importance of the relational algebra being *closed under the operators* for the derivation?

----------Answer----------
Derivation: Now let us assume we have two relations, R and S, where attribute A exists in both relations.

The operation R ⋈ S, where R.A = S.A can be re-written as the following:

$\boldsymbol{\pi}$(desired attributes)$\boldsymbol{\sigma}$(R.A = S.A) (R X S).

The "R X S" produces all combinations of R and S, and "$\boldsymbol{\sigma}$(R.A = S.A)" selects those results with R.A = S.A. Then, the projection, "$\boldsymbol{\pi}$(desired attributes)", removes any undesired attributes if necessary.

Importance: if we apply relational algebra to some relations, it never returns something that is out of the scope, i.e., it never produces things other than valid relations that can be used as input for the next relational algebra operation. Therefore, join is equivalent to consecutive relational algebra operations.

R4 – Equivalent Queries

Briefly explain the concept of equivalent queries. Later lectures explained an important use of the concept. What is that use?

----------Answer----------
Explanation: when two or more different query expressions produce the same results under a given database, they are called equivalent queries.

Later in the lectures (lecture 10), equivalent queries were introduced to optimise query complications. See the figure from the lecture slide:

A relational algebra expression may have many equivalent expressions

- E.g., $\sigma_{salary<75000}(\prod_{salary}(instructor))$ is equivalent to $\prod_{salary}(\sigma_{salary<75000}(instructor))$

They produce the same results but they may have different computational complexity when we specify an evaluation-plan

SQL

S1 – Foundation

Codd’s Rule 0 states

Rule 0: The *foundation rule*:

> For any system that is advertised as, or claimed to be, a relational database management system, that system must be able to manage databases entirely through its relational capabilities.

Briefly explain and give examples of how the rule applies to:

1. Metadata
2. Security

----------Answer----------

Metadata: metadata of the database management system, including table schemas, column data types and constraints, should be accessible via relational queries.

Security: security features, including user permissions and access controls, should be able to be managed by the relational database's capabilities, such as SQL statements.

S2 – NULL

Codd’s Rule 3 states

Rule 3: *Systematic treatment of null values*:

> 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.

Briefly explain the importance of the rule for:

1. Using different database schemas defined by multiple people.
2. SQL aggregation (group by) queries.

----------Answer----------

Using different database schemas defined by multiple people: The rule ensures consistent handling of missing and/or incompatible data over schemas defined by different people, maintaining data integrity and simplifying data integration or comparison between those schemas.

SQL aggregation (group by) queries: the rule ensures aggregation queries can handle missing values in a systematic way by treating NULLs consistently. Therefore, aggregation queries can group data with missing or inappropriate information.

S3 – Atomic Domains

The Columbia University directory of courses uses 20231COMS4111W002 for this section's "key." This is not atomic and is composed of:

- Year: 2023
- Semester: 1
- Department: COMS
- Course number: 4111
- Faculty code: W

Explain why having the non-atomic key creates problems for:

1. Integrity constraints.
2. Indexes

----------Answer----------

Integrity Constraints: Such a non-atomic key makes it very complicated to maintain data integrity by setting constraints. Individual components of those values must be checked independently, which may lead to consistency.

Indexes: More resources are required to manage multi-element keys. Therefore, non-atomic keys create inefficiencies in database indexing.

S4 – JOINs

Briefly explain the following concepts:

- Natural join
- Equi-join
- Theta join
- Left join
- Right join
- Outer join

----------Answer----------
Natural join merges two tables based on columns with the same name and data type/domain. They match them and remove any duplication.
Equi-join merges two tables by matching the same value of a specific column or specific columns specified by the user.
Theta join merges two tables in a manner that is similar to Equi-join, but it allows not only same value conditions, but it also supports comparison operators (>, <, =, etc.) for one or more columns, which is called a theta condition.
Left join returns all the rows from the left table and the matching rows from the right table.
Right join returns all the rows from the right table and the matching rows from the left table.
Outer join returns all the rows from both tables, even though some are unmatched or have null values.

Reference: https://www.guru99.com/joins-sql-left-right.html#5 (https://www.guru99.com/joins-sql-left-right.html#5)

S5 – Natural Join

Briefly explain how using the natural join might produce an incorrect answer.

----------Answer----------
Assume we have two relations as follows

Person(ID, last_name, first name)
Phone(ID, device_name, phone_number, person_ID)

Natural join uses the column with the same name to merge two tables. In this case, it uses the column ID. However, they do not refer to the same thing. The ID in the Person relation is the person_ID in the Phone. So natural join will produce an incorrect answer. Furthermore, if those two IDs have different types, the natural join will not work at all.

S6 – Views

List three benefits/use cases for defining views.

----------Answer----------
Simplification: views make it easier for naive users to do complicated queries. This is because view converts those complicated query statements into views so that users without comprehensive knowledge of SQL can access the results of such queries via pre-defined views.
Privacy: We often do not want users to get access to the entire logical model (e.g. salary column of a people table can be confidential). A view can prevent users from accessing that

information by not including them in the view.
Safety: using views may protect the application from schema changes. Definitions of views and schema are isolated. Therefore, changing the schema definition will not change views. Also, views can be set to read-only so that users may not modify the underlying relation of the dataset by using views.

Reference: Lecture and lecture slides.

S7 – Materialized View

What is a *materialized view?* List one advantage and disadvantage of a materialized view.

----------Answer----------
Definition: Materialised views are stored in certain database systems and can be refreshed to ensure that they remain up-to-date if the actual relations used in the view definition change.

Advantage: Materialised views can simplify queries since there is no need to execute complex queries repeatedly.

Disadvantage: Materialised views require additional storage space. It may also lower system performance during refresh operations.

Reference: The DB book P140

S8 – View Updates

Explain two scenarios in view definition for which it is not possible to update the underlying tables?

----------Answer----------
The view is not simple: If the view does not satisfy any of the following: (a) its “from” clause has only one relation; (b) its “select” clause solely consists of attribute names of the relation and does not include any expressions, aggregates, or distinct specifications; (c) its attributes that are not specified in the “select” clause can be assigned a null value, that is, they do not have a “not null” constraint or are not a part of the primary key; (d) There is no “group by” or “having” clause present in the query. Then the view cannot be updated. Therefore, the underlying tables cannot be updated in those four scenarios.

Reference: The DB book P141-142

S9 – Primary/Unique

What is the main difference between a primary key constraint and a unique constraint?

----------Answer----------
A primary key constraint uniquely identifies each row in a table and **cannot contain null values**. In contrast, a unique constraint ensures that the values in one or more columns are unique, **but those values can be null**. Also, a table can have only one primary key constraint but may have more than one unique constraint.

S10 – Cascade

The *Classic Models* databases have several foreign key constraints. Two examples are:

1. *orders.customerNumber → customers.customerNumber*
2. *orderdetails.orderNumber → orders.orderNumber*

Briefly explain the concept of *cascading actions* relative to foreign keys. For which of the two examples above might cascading make sense?

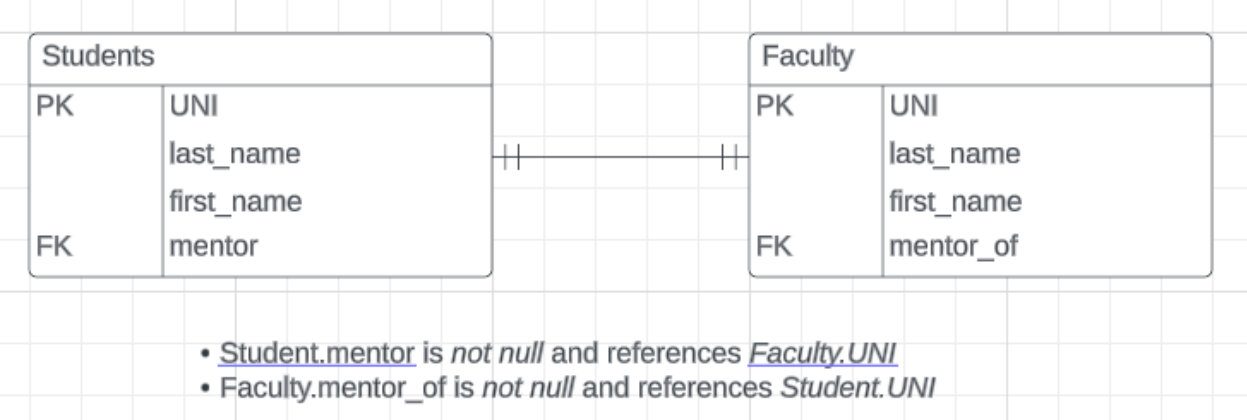
----------Answer----------
Explanation: cascading actions in foreign keys refer to automatic updates or deletions in the child tables when changing information of the referenced table, ensuring data integrity and consistency between relations.

The cascading actions may work for the second example, "orderdetails.orderNumber → orders.orderNumber". Say, if a customer cancels an order, then the 'orders' table will be modified. Cascading actions will then modify information in the 'orderdetails' table to ensure data integrity and consistency.

But we typically do not delete a customer, and it does not make sense that when customers are leaving, we should not cancel their existing orders.

S11 – Foreign Keys and Transactions

Consider the logical data model below.



Some DBMS support deferring enforcing foreign key constraints until transaction commit. How would that capability help with the above model?

----------Answer----------
When someone has the above model, inserting data can be suffering without deferring enforcing foreign key constraints until the transaction commits. Say a student must have a mentor, and a mentor must have a student, so how to insert the first pair when the database has no data? If you insert a student, there is no mentor to be referenced and vice versa. One may successfully insert data into the database by deferring enforcing foreign key constraints because the constraints will not be applied for intermediate statements.

S12 – Complex Check Constraints

Some databases do not support complex check constraints. Consider the following constraint:

check (time_slot_id in (select time_slot_id from time_slot))

Assume the DBMS does not support subqueries in check constraints. What database capability would you use to implement equivalent functionality?

----------Answer----------
If subqueries in check constraints are not supported, we may use a trigger to implement equivalent functionality. In this example, we may use a trigger to checks whether the 'time_slot_id' value exists in the 'time_slot' table before allowing any insert or update operation. The trigger is of the format:

CREATE TRIGGER check_time_slot_id
BEFORE INSERT OR UPDATE ON some_table
FOR EACH ROW
(some code that checks whether the 'time_slot_id' value exists in the 'time_slot' table)
END;

Reference: The DB book P152-153

S13 – Asset

What is the difference between an *Assert* constraint and a *Check* constraint?

----------Answer----------
Difference 1: *Check* constraints can only include one relation, whereas *Assert* constraints are database-level constraints, meaning they may include multiple relations. (*Check* constraints are defined at the relation level, whereas *Assert* constraints are defined at the database schema level.)
Difference 2: *Check* constraints only maintains a single relation's integrity, whereas *Assert* constraint maintains the integrity of the entire database.
Difference 3: Most DBMSs support *check* constraints, whereas not all DBMs support *Assert* constraints, as they are not part of the SQL standard.

Reference: The DB book P152-153

S14 – Types, Domains

Some relational DBMS support *user defined types* and *user defined domains.* Briefly explain the concepts and benefits.

----------Answer----------
User-defined types and *user-defined domains* are that allows users to create customised data types based on existing base types. For example,

create type Pounds as numeric(12,2) final;

is a *user-defined type*, whereas

create domain DDollars as numeric(12,2) not null;

is a *user-defined domain*. They can be used later in the definition of the attributes. The benefit is that they **enhance flexibility, and maintainability in database design and simplify the code**. Say, if we want the Pounds type only contains one decimal, we may modify the Pounds type, not in all definitions. Also, we use Pounds type to ensure that we never write the “numeric(12,2)” again and to prevent any typos. *User-defined types* and *user-defined domains* also **make it easier to ensure the format of the data is of consistent form.**

S15 – SQL Injection

Poorly written web applications can suffer from SQL Injection (Attacks). Briefly explain the concept.

----------Answer----------
Concept: in many web applications, we can set a user input as a variable and insert this variable into a prepared statement to make a query. But poorly written web applications can be easily attacked by malicious hackers by using the technique namely SQL Injection. Instead of putting desired inputs, they put well-designed input to steal information or damage the database.

Example:
Desired input: a person's name

Prepared statement: "select * from instructor where name = '" + name + "'".

The hackers, instead of putting a name into it, they put X' or 'Y' = 'Y. Then the query becomes

select * from instructor where name = 'X' or 'Y' = 'Y',

which will return all information ('Y' = 'Y' is always TRUE). This may lead to credential leaks.

Reference: The DB book P189-190.

S16 – Functions, Procedures, Triggers

Briefly lists two differences between:

- Functions and Procedures
- Functions and Triggers
- Triggers and Procedures

----------Answer----------
Functions and Procedures: functions cannot modify data and always return values, whereas procedures can modify data and sometimes return values.

Functions and Triggers: functions cannot modify data, always return values and can be called via SQL statements, whereas triggers can modify data, but they cannot return any values and are invoked by certain actions (update, insert, etc.) only.

Triggers and Procedures: triggers cannot return any values and are invoked by certain actions (update, insert, etc.) only, whereas procedures sometimes return values and can be called via SQL statements.

Reference: Reference: https://www.youtube.com/watch?v=VpJyzd5BY3Y

Entity – Relationship Modeling

E1 – Implementing Relationships

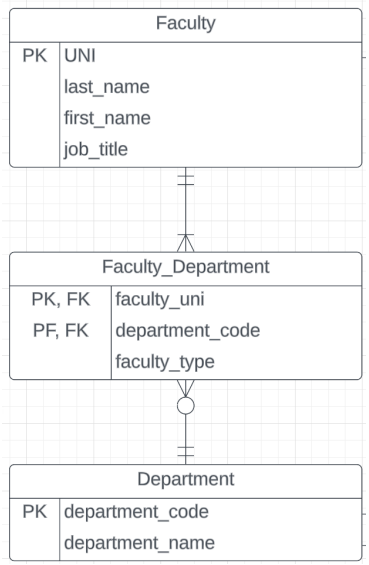
The book's entity-relationship modeling notation explicitly represents relationships. For example,

Crow's Foot notation, which we used in class examples, does not support relationships. What type of entity did we use instead? Give two examples/reasons that require using the entity type.

----------Answer----------

We can use **associative entities** to represent relationships.

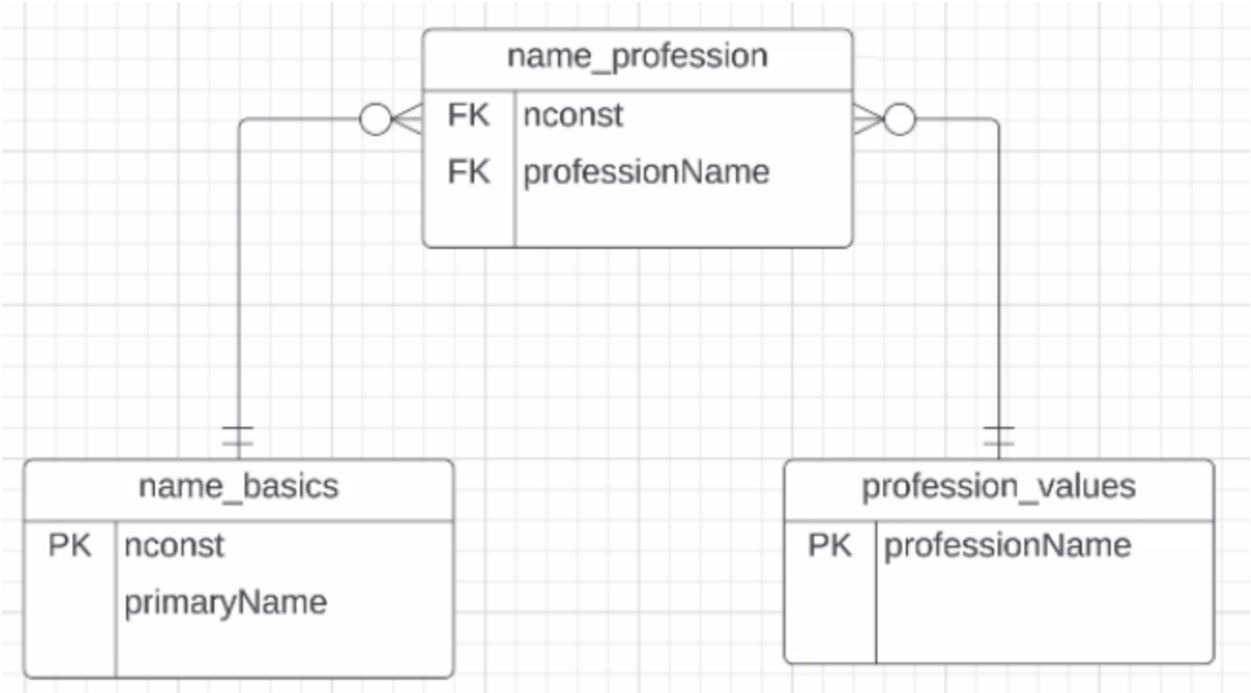
Example 1
Here we use an associative entity, Faculty_Department, to break down the many-to-many relationship and add an extra attribute faculty type to record more information which belongs to the relationship.



Reason: simplify complicated many-to-many relationships and add extra attributes.

Example 2

We create a name_profession associative entity in the IMDB database to eliminate the original non-atomic column, primaryProfessions.



Reason: eliminate non-atomic, multi-value columns.

Reference: HW 2 Part II (Non-programming) and Final Exam

E2 – Types of Relationships

Briefly explain the following concepts:

- Binary and Non-Binary Relationships
- Relationship Cardinality

----------Answer----------

Binary relationships involve two entity sets, whereas **non-binary relationships** involve more entity sets.

Relationship cardinality refers to the possible number of entities that can be related to each other through a relationship set.

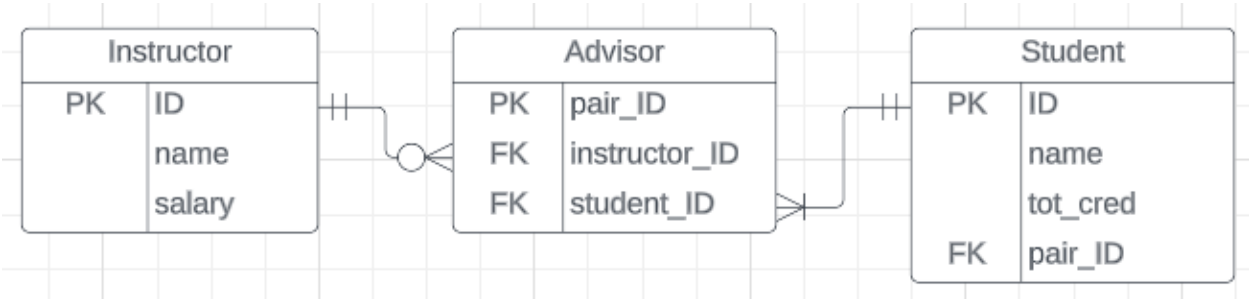
E3 – Participation

An important concept in ER modeling is *relationship participation.*

- **Total participation** (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set

Use Lucidchart to draw an equivalent diagram in Crow's Foot notation. What capability of SQL database definition would you use to enforce total participation?

----------Answer----------



We need to set the two FKs in the Advisor table to NOT NULL and use two (Advisor and Student) referenced tables to enforce total participation. However, this can lead to difficulties in inserting data. So, we must defer enforcing foreign key constraints until the transaction is committed. For example, in PostgreSQL, this can be done by DEFERRABLE INITIALLY DEFERRED.

E4 – Weak Entity

Briefly explain the concept of a *weak entity.* Give an example from the Classic Models database.

----------Answer----------
Explanation: a weak entity set is an entity set that must be dependent on another entity set, which is called identifying entity set. Weak entities use a discriminator attribute combined with the primary key of the related entity, to uniquely identify a weak entity. This can minimise redundancy in relationships.

Example: in the Classic Models database, the "OrderDetails" entity is a weak entity. It stores individual items in an order, like quantity and price, but can't be uniquely identified by these attributes. Instead, it relies on the "Orders" entity's primary key, "OrderNumber," and uses a composite key combining its partial key "ProductCode" with "OrderNumber" to uniquely identify ordered items.

E5 – Specialization

Briefly explain the following concepts relative to implementing inheritance/specialization in an SQL schema.

- incomplete/complete
- disjoint/overlapping

----------Answer----------
Incomplete/Complete:

In an incomplete specialisation, some parent class instances possess unique attributes, while others maintain only shared attributes. In a complete specialisation, each instance of the parent class exhibits at least one distinct attribute not common to the parent class.

Disjoint/Overlapping:

An object exclusively belongs to a single specialised subclass in a disjoint specialisation. Conversely, in an overlapping specialisation, an object may simultaneously be a member of multiple specialised subclasses.

Reference: Lecture 6 slide

Normalization

N1 – Duplicate/Redundant Data

A primary reason for schema normalization is to eliminate duplicate/redundant data. What are two problems that redundant/duplicate data can cause?

----------Answer----------
Need more space: Storing redundant data consumes extra storage space, which lowers the performance of the database.

Update inconsistency: if data is duplicated across multiple rows and/or tables, updating one piece of information may require changes in many places, which leads to the risk that users might not update all of them, leading to data inconsistency.

Reference: The DB book P304-305

N2 – Decomposition

Briefly explain the concept of *lossless decomposition* in normalization.

----------Answer----------
Say we have a relation, R, and we use A and B to form a decomposition of R. If replacing R with A and B will not lead to any information loss, then we say this decomposition is a lossless one. The goal of such decomposition is to eliminate redundancy and improve data consistency without any information loss.

Reference: The DB book P307

N3 – Functional Dependency

Briefly explain the following concepts:

1. Functional Dependency
2. Closure of Functional Dependencies

----------Answer----------
Functional Dependency: for relation R, if for every allowable instance r of R, we have that when t1 and t2 belong to r, t1[X] = t2[X] implies t1[Y] = t2[Y], then X->Y (X and Y can be sets of attributes). That is, for any two tuples in r, if the values of X agree, then the values of Y must also agree.

Closure of Functional Dependencies: If we have a set, F, which is a set of functional dependencies, then the closure of F, denoted by F+, is the set of all functional dependencies that can be logically implied by the elements in F. For example, F has A->B and B->C. Then we can logically imply A-> C (transitivity rule). Thus, F+ is A->B, B->C, A->C.

Reference: The DB book P310, 321.

N4 – BCNF

Consider the sample university database that comes with the recommended textbook.

Consider a hypothetical relation:

in_dep (ID, name, salary, dept_name, building, budget)

Why is the relation not in BCNF?

----------Answer----------
A relation R is in BCNF if that for all functional dependencies in F+ of form A->B, where A and B belong to R, at least one of the following holds: (a) A->B is a trivial functional dependency, i.e., B belongs to A. (b) A is a superkey for R.

In the hypothetical relation, we have dept_name -> building, budget, but dept_name is not a super key of in_dept and this functional dependency is not trivial (B does not belong to A). Therefore, the hypothetical relation is not in BCNF.

Reference: The DB book P314

N5 – Third Normal Form

Briefly explain the difference between BCNF and 3rd Normal Form.
----------Answer----------
The main difference is that the 3rd Normal Form is a relaxation of BCNF. That is, if a relation is in BCNF, it is in the 3rd Normal Form. However, if a relation is in the 3rd Normal Form, it might not be in BCNF. So BCNF is stronger than the 3rd Normal Form.

This is because, in the same setting as mentioned in N4, in addition to (a) and (b), if (c) each attribute X in B - A is contained in a candidate key of R holds, then we say R is in 3rd Normal Form (each attribute can be in different candidate key sets).

Reference: The DB book P 317

N6 – Armstrong's Axioms

Briefly list Armstrong's Axioms for Functional Dependencies.

----------Answer----------
Assume A, B and C are sets of attributes. We have:

Reflexivity Rule: If B is a subset of A, then A -> B holds.
Augmentation Rule: If A -> B holds and C is an attribute set, then CA -> CB holds.
Transitivity Rule: If A -> B and B -> C hold, then A -> C holds.

Reference: The DB book P321

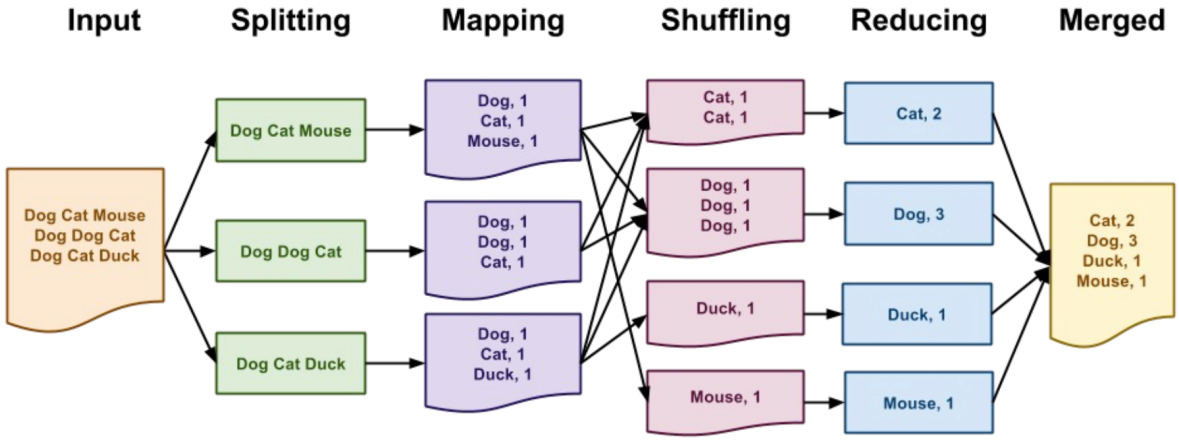
Big Data

B1 – MapReduce

Briefly define the following concepts in MapReduce:

1. Map
2. Reduce
3. Shuffle

----------Answer----------
Here we use the example in Lecture 12



Map: processes input data using a map function into key-value pairs (e.g. (Dog, 1)) for later grouping usage.
Shuffle: performs data exchange and data sorting to bring all the values of a key together for later reducing usage.

Reduce: uses a reduce function to aggregate (in this example, we count the number of each word) the prepared key-value pairs with the same key, so that the data size is minimised.

Reference: The DB book P484-488.

B2 – Algebraic Operation

Modern big data processing systems introduce the concepts of:

1. Directed acyclic graphs.
2. Algebraic operations.

Briefly the concepts.

Prof. Ferguson suggested in lectures some hypothetical algebraic operators for IMDB namebasics. An excerpt of the data is below. Suggest a couple of hypothetical operators to transform the data.

nconst	name	dob	dod	primaryProfessions	knownFor
nm0000001	Fred Astaire	1899	1987	soundtrack,actor,miscellaneous	tt0053137,tt0050419,tt0045537,tt0072308
nm0000002	Lauren Bacall	1924	2014	actress,soundtrack	tt0117057,tt0071877,tt0038355,tt0037382
nm0000003	Brigitte Bardot	1934		actress,soundtrack,music_department	tt0056404,tt0049189,tt0057345,tt0054452
nm0000004	John Belushi	1949	1982	actor,soundtrack,writer	tt0077975,tt0078723,tt0072562,tt0080455
nm0000005	Ingmar Bergman	1918	2007	writer,director,actor	tt0050976,tt0060827,tt0083922,tt0050986
nm0000006	Ingrid Bergman	1915	1982	actress,soundtrack,producer	tt0036855,tt0038787,tt0034583,tt0038109
nm0000007	Humphrey Bogart	1899	1957	actor,soundtrack,producer	tt0042593,tt0037382,tt0034583,tt0043265
nm0000008	Marlon Brando	1924	2004	actor,soundtrack,director	tt0070849,tt0078788,tt0068646,tt0047296
nm0000009	Richard Burton	1925	1984	actor,soundtrack,producer	tt0061184,tt0059749,tt0057877,tt0087803
nm0000010	James Cagney	1899	1986	actor,soundtrack,director	tt0029870,tt0042041,tt0035575,tt0031867

----------Answer----------

The concepts:

Directed Acyclic Graphs: a data structure representing operators with directed edges, showing dependencies between tasks without cycles. It is like a workflow.
Algebraic Operations: user-customised mathematical operations applied to datasets for efficient data manipulation and processing.

Possible Hypothetical Operators: (a) split name operator: an operator takes into the name and separates it into first name, middle name, last name, initials, etc. (b) split primary profession operator: an operator takes into primaryProfessions and separates into the following photo (this is from the lecture) to eliminate this non-atomic column. This operator support support parallel computing.

nm0000001	Fred Astaire	1899	1987
nm0000002	Lauren Bacall	1924	2014
nm0000003	Brigitte Bardot	1934	
nm0000004	John Belushi	1949	1982
nm0000005	Ingmar Bergman	1918	2007
nm0000006	Ingrid Bergman	1915	1982
nm0000007	Humphrey Bogart	1899	1957
nm0000008	Marlon Brando	1924	2004
nm0000009	Richard Burton	1925	1984
nm0000010	James Cagney	1899	1986

nconst	profession_id
nm0000001	1
nm0000001	2
nm0000001	3
nm0000002	3
nm0000002	4
nm0000003	3
nm0000003	4
nm0000003	5
nm0000004	1
nm0000004	3

Profession	Profession_id
actor	1
miscellaneous	2
soundtrack	3
actress	4
music_department	5
writer	6
director	7
producer	8
make_up_department	9
composer	10
assistant_director	11

(c) split knownFor operator: Similar to that in (b) for the knownFor, it eliminates the non-atomic column, knownFor, in a parallel way.

Reference: Lecture 12 slide.

B3 – Concepts

Briefly explain the following concepts:

- Data Warehouse
- Data Lake
- Extract-Transform-Load

----------Answer----------
Data Warehouse: a single-site storage facility that compiles information from different sources and organises it under a unified schema. This simplifies querying and enables historical trends study. It transfers query workload for decision support away from transaction processing systems.
Data Lake: a storage system that houses raw, unstructured, and semi-structured data from various sources, arranged and utilised as required (highly agile). This is designed for low-cost storage with less mature security than a data warehouse. A data lake is mainly for data scientists.
Extract-Transform-Load: the process of getting data into a data warehouse is called Extract-Transform-Load.

Reference: The DB book P524, Lecture 12 slide.

Database Management System Implementation

D1 – Storage Types

Briefly explain and list some differences between:

- RAM
- Solid State Drives
- Hard Drives

----------Answer----------

1. RAM is for temporary data storage, whereas SSDs and HDDs are non-for long-term storage.
2. RAM has a lower storage capacity compared to SSDs and HDDs. HDDs provide even more space compared with SSDs.
3. RAM is the fastest, SSDs are fast, and HDDs are the slowest.
4. RAM uses integrated circuits, SSDs use NAND flash memory, and HDDs use magnetic disks with read/write heads.

D2 – Addressing

Briefly explain the concepts of:

- Logical block addressing
- Cylinder-Head-Sector addressing

D3 – Elevator Algorithm

What is the elevator algorithm for disk arm scheduling and what is its benefit?

----------Answer----------

Elevator algorithm: a commonly used above algorithm that controls the arm to stop at each track and processes the services requests for it, then moves the arm forward. This can increase the number of accesses that can be processed.

Reference: The DB book P578.

D4 – Fixed Length Records versus Variable Length Records

Briefly define and list benefits of fixed length records and variable length records.

D5 – BLOBs

Most application scenarios no longer use database BLOBs. What technology do applications typically use in place of BLOBs?

D6 – File Organization

Give scenarios where:

- Sequential record organization is better than heap file organization.
- Multi-table clustering is better than sequential record organization.
- You would use table partitioning.

D7 – Buffer Replacement Algorithm

For which type of query is most-recently-used a much better replacement algorithm than least-recently-used?

D8 – Row Oriented versus Column Oriented

Explain why column oriented storage may be beneficial for scenarios in which:

1. Tables are large.
2. The only query operations are projection and aggregation.

D8 – Index Types

Briefly explain the following concepts:

- Clustering index
- Dense index
- Sparse index

Can there be more than one clustering index on a table?

Must a sparse index be a clustering index?

D9 – Hash versus B+ Tree

What is the primary benefit of a hash index relative to a B+ tree index? What are two disadvantages?

D10 – Degree

Explain the relationship between key size, block size and B+ tree degree.

D11 – Covering Index

What is a covering index and what is the benefit?

D12 – Number of Indexes

What are two disadvantages of adding many indexes to a table?

D13 – Buffering and Logging

Briefly explain:

- Force/No-Force policy
- Steal/No-steal policy
- The relationship between the policies and redo/undo logging.

D14 – Access Path

Briefly explain the role of access path selection in query processing/optimization.

What is the “most selective index?”

D15 – JOIN Optimization

Consider two tables L and R. Neither table is ordered and there are no indexes.

Consider the query *select * from L join R using(c).*

If the tables were large, give a scenario for creating an index for optimization and the type of index.

What optimization might the query processor make if L was much, much smaller than R?

D16 – JOIN Algorithms

Briefly explain the following concepts:

- Nested-loop join
- Block nested-loop join
- Indexed nested-loop join
- Merge-join
- Hash-join

D17 – Optimization

Consider the following query on a very large table *people.*

select last_name, first_name from people

What single word/modification added to the query might motivate creating a has index for optimization?

----------Answer----------
SELECT **DISTINCT** last_name, first_name FROM people;
The DISTINCT word motivates creating a hash index for optimisation.

D18 – Optimization Techniques

Briefly explain the following concepts relative to query optimization:

- Operator selection
- Equivalent queries

----------Answer----------
Operator Selection: we want the most efficient method for a specific operation in a query, so we choose the fastest operation according to the dataset size and its intrinsic structure.

Equivalent Queries: different query formulations that produce the same result. We want the fastest query according to the dataset size and its intrinsic structure.

D19 – Equivalent Query Selection

Assume the tables in Classic Models were very, very large.

What is an equivalent query that a query optimizer might use in place of

```
SELECT
	*
FROM
	customers JOIN orders USING(customerNumber)
WHERE
	country = ‘France’ and status = ‘Shipped’;
```

----------Answer----------

```
SELECT
	*
FROM
	(SELECT * FROM customers WHERE country = 'France') AS filtered_customers
JOIN
	(SELECT * FROM orders WHERE status = 'Shipped') AS filtered_orders
USING (customerNumber);
```

This query per-filters the database.

D20 – Locking

Briefly explain 2 Phase Locking and Strict 2 Phase Locking. What is the benefit of Strict 2 Phase Locking?

D22 – Phantom

What is a “phantom” relative to database transactions/query processing?

D25 – Serializable

Briefly explain serializability and conflict serializability.

D26 – CAP

Briefly explain the CAP theorem.

D27 – Consistency

Briefly explain *eventual consistency.*

D28 – Sharing

Briefly explain database sharding and its benefits.

D29 – Scaling

Briefly explain:

- Scale up versus scale out.
- Shared disk/data versus shared nothing/sharding.