F2 – Types of Data

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

- Structured data is organized in searchable and analyzable format. They have relational key and can easily mapped into pre-designed fields. EXA, inventory lists of manufactures.
- Semi-structured data is partially structured. It does not form a schema or predefined data model, but easier to store than unstructured data. EXA, JSON files.
- Unstructured data has no identifiable structure or format. Hard to analyze and not store in a database. EXA, image or video.

F3 – Physical Data Independence

What is physical data independence? What is a benefit?

- Physical data independence is a property of DBMS. It can allow change the schema at one level of DBMS without affecting the schema at the next or higher level. One of benefits is that it allows us to modify the physical schema without causing application program to be rewritten. Thus, reduce the cost and complexity of maintaining and updating.

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

- DML manipulates data in a database. EXA, delete.
- DDL manages database structure. EXA, create table.
- Procedure DML define functionality of a database object. It executes when certain events occur. EXA, triggers.
- Declarative DML define properties of a database object. It can ensure accuracy, consistency, and integrity. EXA, primary key constraint.

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

- Conceptual model is an abstract representation of a problem domain. It provides a high-level overview of data requirements and relationships. EXA, when we design the payment recording system of a hospital. It can show the complexity of problem domain and further assisting in decision-making and speeches.
- Logical model stands out from any physical implementations, but define the specific data structure and constraints. EXA, it can help to keep an ongoing inventory of resources and to monitor the project metrics for success.
- Physical model is design in specific database management system, including tables, indexes, etc. The way to storage and retrieve data is specified. EXA, baseball related schema in midterm has a lot of rows, where we use indexes achieving efficient storage and retrieving.

F6 – Application Architectures

Briefly explain:

1. Two-tier database application architecture

- Three-tier architecture Two-tier architecture: client application communicates directly with the database server. Client application responsible for user interface and application logic. Database server responsible for storing and managing data.
- Three-tier architecture: presentation, application, data three different layers. Presentation responsible for user interface. Application layer process requests from clients. Data layer responsible in storing and managing data.

F7 – Database Administrators (DBA)

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

- Database design and implementation. Design for specific data requirement and organization needs.
- Performance tunning and monitoring. Adjust to efficiency and index.
- Security management. Protect sensitive data.
- Backup and recovery. Prevent data loss.
- User management. Manage access permissions.

Use DDL for creating and modify table, index, views, etc. DML for insert, updating, deleting. Both are used depending specific needs at the time.

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.*

- Atomicity of domain restrict the attribute in a database can only contain a single, indivisible value. It cannot be divided into smaller components further. Thus, it ensures data integrity and avoid data anomalies. For example, a person weight can be weight in pounds and weight in kilograms.

R2 – Keys

Briefly define and explain the following concepts:

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

- Super key is a set of one or more attributes in a relation can uniquely identify each row in that relation. It may include additional attributes beyond the minimum requirement for uniqueness.
- Candidate key can be candidate key or a composite key. It cannot be further reduced while still ensuring uniqueness. It smallest and unique.
- Primary key is a special type of candidate key and chosen as the main reference for other table when there is a foreign key relationship in between. It is also unique identification of each row.
- Foreign key is a column or a set of columns in a table that refers to the primary key of another table. It relates two tables and reference data between 2 tables in integrity constraints.

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.

- $\boldsymbol{\pi}$(R.A, R.B, S.C)($\boldsymbol{\sigma}$(R.B = S.B)(R x S))
- In the above, we perform join of R and S on the attribute B. RxS creates all possible combinations of R and S, then select only when R.B matches with S.B. Next, only take information from R and S and remove the duplicates.

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

- Closed ensures any valid relational expression can be expressed as combination of these operators. A limit set of operators can derive new relations without outside introducing. Thus, completeness.
- It provides convenience for us to query language in uniform and systematic way using a small set of operators.

R4 – Equivalent Queries

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

- Equivalent queries are queries that produce same result set for a database.
- Use of equivalent queries is optimize query in using less computational power and faster.

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

- Briefly explain: RDBMS should handle all including structure, data storage, retrieval using relational model and sql language.
- Metadata refers to objects like tables, columns, and relationships in the database. Properties like schema definition, data types, constraints, and index have a standardized way to create, modify, and query using sql.
- Security is protecting data from unauthorized access and modification. Similarly, system need to operate in a standardized way to grant privileges to users and enforce security policies by sql.

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.
    - When there are multiple editors, this rule can help to maintain consistency and avoid inconsistent recognition toward meaning of null values as missing information.
    - When we perform group by function, null values in columns are considered equal, and they are put into a single group. Because it will directly affect the query(avg,count) output, it's important.

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

- Any changes in the course number or department number will case integrity problem, because the old key no longer refer to the correct course. Change in the structure is not allowed.
- It's not efficient index if we search for specific course or department. Singe, indivisible keys will save time and resources.

S4 – JOINs

Briefly explain the following concepts:

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

- Natural join: match tuples with same value for all matched columns, and only keep one copy of replicated columns.

- Equi-join: return tuples from tables when values in the matched columns are equal.

- Theta join: use comparison operator other than equality to match tuples from tables.

- Left join: retrieve all records from the left table and the matching records from the right table.

- Right join: retrieve all records from the right table and the matching records from the left table.

- Outer join: return all records when there is a match in left table or right table.

S5 – Natural Join

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

- When two tables have columns with same name, but different data types. Thus, natural join require same column datatypes, only one same-named column, joined rows have same name and logical relation. Without satisfying the requirements above, the natural join will produce errors or unexpected outcomes.

S6 – Views

List three benefits/use cases for defining views.

- Due to security concerns, view can be personalized collection of virtual relations.
- It's possible to support a large number of views on the top of actual relations, because it's not precomputed and stored. Take place of with clause.
- Database system stored view as definition. Therefore, content of view is recomputed when we query and not out of date.

S7 – Materialized View

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

- Define: the view has already been evaluated and stored.
- Advantage: speed up query processing and cheaper.
- Disadvantage: they must keep up-to-date when data used in the view definition changes.

S8 – View Updates

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

- Avg. It depends not only on the old average and the tuple being inserted or deleted, but also the number of tuples. Once the individual tuple is not available, one-to-one correspondence characteristics make it impossible to update.
- Join involving multiple tables. When a tuple is inserted, we need to check if it’s present in other tables, where may need changes to underlying tables.

S9 – Primary/Unique

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

- Primary key serves as a primary identifier for the table, cannot allow null value. A unique constraint can allow null.

S10 – Cascade

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

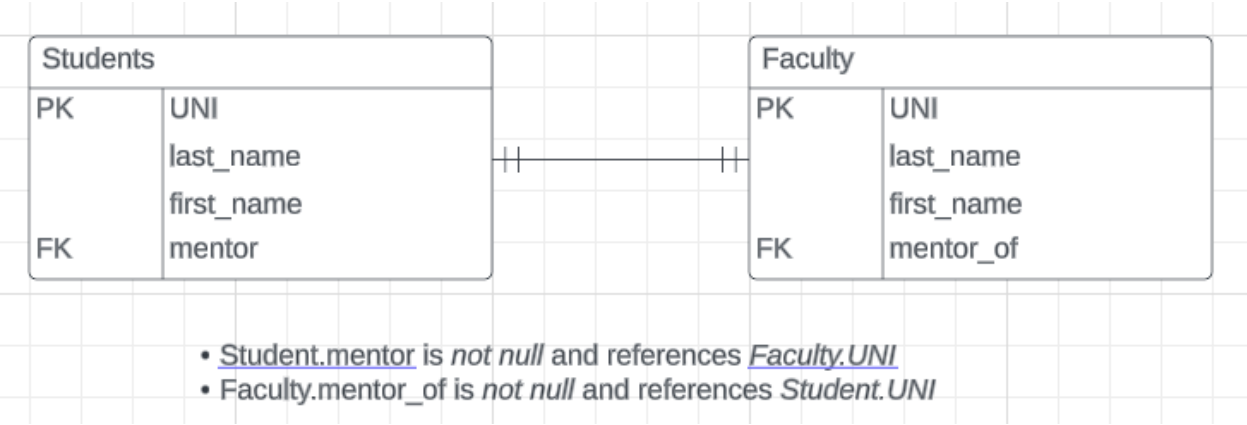
1. *orders.customerNumber → customers.customerNumber*
2. *orderdetails.orderNumber → orders.orderNumber*

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

- When related tuples in the parent table is updated or deleted, automatic update or delete will happened to child tables.
- *orderdetails.orderNumber → orders.orderNumber.* When orderNumber is deleted from orders, then orderNumber in orderdetails are deleted correspondingly.

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?

- Into our model, when we want to insert a new student Alice with faculty Bell, under the constrain, we need to add Bell in Faulty table at first with Bell's uni, then add Alice into Student table with Alice uni. Without constrain, we can add Alice into Student table at first, then use uni of Alice refer back to Faulty table as foreign constrain, then add-in Bell's other information. The logical of second is easier to follow without always reminder order rule in the brain.

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?

- Trigger.
- Create a stored procedure to perform the check and then call it within insert or update.

S13 – Asset

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

- Check enforces condition of a single row. Assert can cross multiple rows or tables.

S14 – Types, Domains

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

- User defined types define by users to create new data type by combining existing data types.
- User defined domain is a constraint toward a data type.
- Benefits: improved data consistency, data integrity, allow more complex operations to be performed on the data.

S15 – SQL Injection

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

- Poorly written web application does not properly validate user input before using it in sql statement. Then, attacker can insert malicious sql code into sql statement through inputs from web applications.

S16 – Functions, Procedures, Triggers

Briefly lists two differences between:

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

- A function returns a value. Procedures may not return a value.
- Functions used to compute value in calculation. Procedure has series of operations or tasks.

- Functions and Triggers

- Functions are invoked by call in query or statement. Triggers response automatically.
- A function returns a value. Trigger not returns a value.

- Triggers and Procedures

  - Triggers are invoked automatically. Procedures are invoked by calling.
  - Triggers actions is performed based on changes depending on what happened to data. Procedures perform changes on data(operation).

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.

- Associative entity.
- The many-to-many relationship allows one instructor to associated with multiple students. It can also prevent data duplication and inconsistencies.
- It's easier to add in new attributes related with advising relationship. For example, there are class in advising circle. Based on that class, we need to add course number, time, location, etc to the relationship.

E2 – Types of Relationships

Briefly explain the following concepts:

- Binary and Non-Binary Relationships
- Relationship Cardinality
  - Binary is relationship between two entities. Non-binary will evolve three or more entities.
  - Number of entities can involve in a relationship between two or more entities. There are one-to-one, one-to-many, many-to-many.

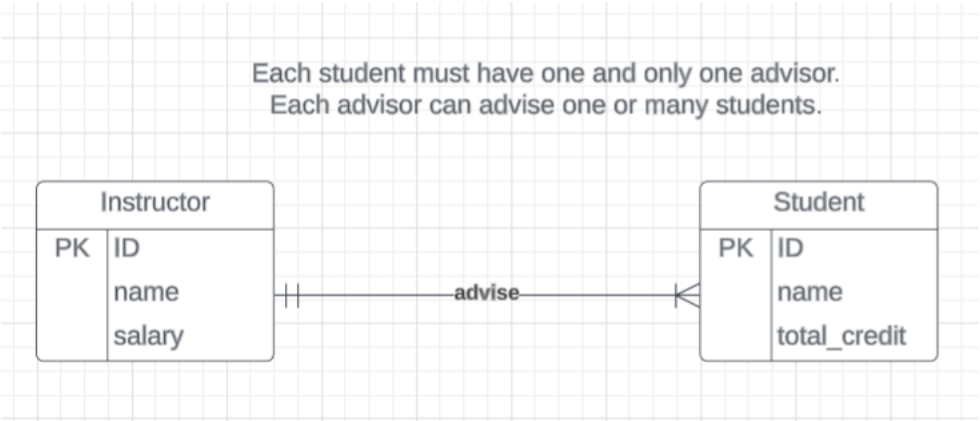
E3 – Participation

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?

- Foreign key in student table refers to advisor cannot be null. We use 'NOT NULL'.



E4 – Weak Entity

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

- Weak entity set is one whose existence is dependent on another entity set.
- EXA orderdetails and order, where orderdetails are dependent on order.

E5 – Specialization

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

incomplete/complete
disjoint/overlapping

- entity in parent class represented in at least one subclass is complete. Incomplete specialization allows some entities in the parent class not belong to any sub-class.
- Disjoint subclasses have no common entities. Overlapping subclasses have entities belong to more than one subclass (may overlap).

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?

- Duplicate data need more storage space and slow down searching and manipulating data.
- Redundant data have inconsistency problem. When data is not up-to-date, the query results will influence the further decision accuracy.

N2 – Decomposition

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

- Lossless decomposition original information can reconstructed after decomposition. Splitting a relation into smaller, then link them back to the original relation.

N3 – Functional Dependency

Briefly explain the following concepts:

1. Functional Dependency
2. Closure of Functional Dependencies

- Functional dependency is a relationship when value of one attribute uniquely determines the value of another attribute.
- Closure of functional dependency repeatedly applying a set of inference rules to determine all possible functional dependencies from a given set.

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?

- ID and dept_name is candidate key. Building and budget are dependent on dept_name, but not ID, where shows partial functional dependency. Thus, it's not in BCNF.

N5 – Third Normal Form

Briefly explain the difference between BCNF and 3rd Normal Form.

- BCNF requires every determinant in relation is a candidate key. It removes all redundancy and functional dependencies. It's not always possible for all relations.
- 3NF requires 2NF and all non-prime attributes are dependent only on the candidate key. All redundancy is eliminated by transitive dependencies. It's can achieve for all relations.

N6 – Armstrong's Axioms

Briefly list Armstrong's Axioms for Functional Dependencies.

- Reflexivity rule. If alpha is a set of attributes and beta belong to alpha, then alpha->beta.
- Augmentation rule. If alpha->beta holds and gamma is a set of attributes, then gamma*alpha->gamma*beta holds.
- Transitivity rule. If alpha->beta holds and beta->gamma holds, then alpha-> gamma holds.

Big Data

B1 – MapReduce

Briefly define the following concepts in MapReduce:

1. Map
2. Reduce
3. Shuffle

- Map apply to each input record in parallel. It takes a set of input key-value pairs, then perform computation on each pair producing a set of key-value pairs.
- Reduce apply to each intermediate key in parallel across multiple nodes. It takes the key-value pair from map function to produce final output.
- Shuffle ensure all values for a given key are sending to the same reducer node. It occurs between the map and reduce phase. Map produces key-value pair, then shuffle sends out values to same reducer node.

B2 – Algebraic Operation

Modern big data processing systems introduce the concepts of:

1. Directed acyclic graphs.
2. Algebraic operations.

Briefly the concepts.

- DAG is a graph data structure. Node represents computational task. Edge represents dependencies between tasks.
- Algebraic operations are based on map, reduce, filter, etc operations on algebraic principles. It's efficient at transform and process large dataset in parallel.

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.

- Filter out death before 2008. ('dod')
- Sort data by birth data. ('dob')
- Groupby people if there is actor in primaryprofessions column.

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

B3 – Concepts

Briefly explain the following concepts:

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

- Data warehouse gather data from multiple sources at a single site, under a unified schema. Today, it can also collect and store data from non-relational source, where unification is not possible.
- Data lake is a repository where data can be stored in multiple formats, including structured records and unstructured file format.
- ETL is process of extracting data from one or more sources, then transforming it into a format that suitable for analysis. In the end, loading to the data warehouse or data lake.

Database Management System Implementation

D1 – Storage Types

Briefly explain and list some differences between:

- RAM
- Solid State Drives
- Hard Drives

- RAM is a volatile memory devise that used to store data temporarily for fast access by CPU. RAM is fastest among three with less storage capacity.
- SSD is non-volatile storage and store permanently. Use flash memory to store without moving parts. Faster and more expensive than HDD. Less prone to physical damage. Consume less power and less heat.
- HDD is non-volatile storage and store permanently. Use spinning disk with r/w heads to store. It's slower and less expensive than SSD. Larger storage capacities, suitable for archival and bulk storage.

D2 – Addressing

Briefly explain the concepts of:

- Logical block addressing
- Cylinder-Head-Sector addressing

- LBA r/w data without worrying physical location of data in the disk. Each sector of disk has unique number. This number is the key for accessing.
- CHS address place of data on a hard drive with actual move of disk to locate the data. Cylinder, head, and sector are three values help to specify the location of data.

D3 – Elevator Algorithm

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

- Elevator algorithm moves disk arm in one direction and serving all the requests in that direction until there are no more requests in that direction. It moves up and down. It stop at each floor to process the request.
- Advantage: It reduce the access time by minimizing the time to seek data and reducing the time spend back and forth.

D4 – Fixed Length Records versus Variable Length Records

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

- Fixed length records have fixed size and structure. It's easier to locate(indexing) and access while resulting in efficient storage and retrieve.
- Variable length records have different size and structure. It's more efficient in allocating space. Flexibility in different length. Easier to update.

D5 – BLOBs

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

- Google Cloud, Microsoft Azure Blob Storage. They can store image, audio in higher accessibility and durability.

D6 – File Organization

Give scenarios where:

- Sequential record organization is better than heap file organization.
- High volume of sequential access to the data (batch processing sys). Efficient range queries (all records within a specific date range). Limited number of insertions, updates, or deletion aim to lower the cost.
- Multi-table clustering is better than sequential record organization.
- Complex relationship needs to join two or more table frequently. High volume of queries involve multiple table. Optimize the storage of data by grouping.
- You would use table partitioning.
- When large amount of data is stored, it's hard to manage and querying efficiently. Queries involve a specific time period or geographical region. Scenarios where need to provide data availability and reduce the impact of failover mechanism for individual partitions.

D7 – Buffer Replacement Algorithm

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

- When query exhibit temporal locality and accessing the same data repeatedly in short brusts, MRU is better.

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.

- Projection like filtering by putting range to column. Aggregation like sum, count, and average. Data is faster retrieving only certain columns and completing the query handling from column orientation.
- When values in columns are repeated, it's easier to compress, which gives better compression ratios. Thus, storage is more efficient.

D8 – Index Types

Briefly explain the following concepts:

- Clustering index
- Dense index
- Sparse index

- Sort data based on one or more columns. Then, define the stored order of data on disk.
- Entry to every record in a table. Used in Small dataset and fast access to individual records are advantages.
- Only some of the records in a table has index entry. It's usually used in large table.

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

- No. Only one is allowed.

Must a sparse index be a clustering index?

- Not necessarily.

D9 – Hash versus B+ Tree

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

- Hash indices efficient support equality queries on search key. B+ tree requires more disk access for point lookups and equality queries.
- CONS: hash indices do not support range queries.
- CONS: deletion and insert need to locate the bucket on the search key at first. When bucket does not have enough space, a bucket overflow occurs. The overflow buckets need to linked from bucket.

D10 – Degree

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

- Tree degree = (block size-header size)/(Pointer size+key size)
- The larger block size, the larger degree of tree.
- The larger key size, the smaller degree of tree.

D11 – Covering Index

What is a covering index and what is the benefit?

- Covering index covers all columns needed from a specific table, removing the need to access the physical table for advanced operation.
- Because table access is replaced with index look up, reduced logical and physical reads boosting the performance.

D12 – Number of Indexes

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

- Require more disk space to store index structure.
- Slow down operation (insertion, updates, deletion), because the need of update all index for consistency.

D13 – Buffering and Logging

Briefly explain:

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

- Force policy would force-output all modified blocks to disk when they commit. No-force policy allows a transaction to commit even if has modified some blocks that have not yet written to disk.
- Steal policy allows system to write modified blocks to disk even if transactions related to modification have not all committed. No-steal policy means that block modified by a transaction is still active and not written to disk.
- Redo/Undo logging can undo or redo in case of failure. When changes are not immediately written to disk in policies, we still need redo and undo. Force policy we need undo. No-force need both. Steal need redo. No-steal need undo.

D14 – Access Path

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

- It's process of selecting the most efficient way to access the tuples of relation.
- Optimizer perform access path selection by determine order at first. The, relations for each query block are processed.

What is the "most selective index?"

- Index that finds the smallest number of rows.

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.

- When c is the common column for table L and R, creating a clustered index on column c of both tables. Then, the query efficiency of both tables will largely improve.

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

- Nested-loop join

D16 – JOIN Algorithms

Briefly explain the following concepts:

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

- Nested-loop join compares each row from one table with each row from another table. Use when one table small and another large.
- Block nested-loop join reads a block of data from one table and compare with all rows in another table until all the blocks of data have been read.
- Indexed nested-loop join like nested-loop join, but using index on the join columns to speed up the process. After read from one, use index to find matching row in another.
- Merge-join sorts both tables on the join column and then merge together.
- Hash-join creates a hash table for one of tables and then scanning through other table to find matching rows.

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?

- Where followed by a condition

D18 – Optimization Techniques

Briefly explain the following concepts relative to query optimization:

- Operator selection
- Equivalent queries
- Operator selection select the best operator to use for each operator in each operation plan. Then, optimizing by selecting best.
- Equivalent query produces the same result by different operations. Then, the most efficient plan can be chosen.

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';
```

- SELECT * FROM customers JOIN orders USING(customerNumber) WHERE country = 'France' and status = 'Shipped' ORDER BY customerNumber

D20 – Locking

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

- 2PL ensures that transactions are executed in an atomic manner, which ensures serialization. Lock before accessing and release lock after transaction complete.
- S2PL holds all locks until the end of transaction. Shared locks cannot be released before complete. It ensures serializable manner.
- PROS: S2PL can read queries where data not changed and locks can be released for the next transaction to obtain data in stronger ensuring serializability.

D22 – Phantom

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

- When a transaction reads the same set of rows twice but getting different results, it means there is another transaction inserts or deletes rows that affecting first transaction.

D25 – Serializable

Briefly explain serializability and conflict serializability.

- Serializability ensures concurrent transactions will produce the same result, even they produce one after the other.
- Conflict serializability orders any conflicting operation in the same way as some serial execution.

D26 – CAP

Briefly explain the CAP theorem.

- CAP states it's impossible for a distributed system to simultaneously provide guarantees of consistency, availability, partition toleration. It must decide whether to do one of following: cancel the operation by reducing availability, but ensure consistency. Or proceed operation providing availability, but risk inconsistency.

D27 – Consistency

Briefly explain *eventual consistency.*

- It is a trade-off between consistency and availability. If no new updates are made to a given data item, eventually all accesses to that item will return the last updated value, resulting in high availability.

D28 – Sharing

Briefly explain database sharding and its benefits.

- Database sharding partition a single database into multiple smaller databased called shards. Each shard contains a subset of the data and is stored in separate server.
- PROS: scalability distributes the load across multiple servers and allows for horizontal scaling.
- PROS: in case of hardware failure or other issue, failover to other shards can be allowed.
- PROS: more flexibility in terms of data storage and retrieval.

D29 – Scaling

Briefly explain:

- Scale up versus scale out.
- Shared disk/data versus shared nothing/sharding.
- Scale up adds more resources like CPU, memory, or storage to a single server to improve its performance. Vertical scaling.
- Scale out adding more servers to a system to improve its performance. Horizontal scaling.
- Shared disk allows multiple servers to access the same disk or data store.
- Shared nothing partition data across multiple servers, where each server responsible for a subset of the data.