Question Bank and with Solution

On

Introduction of DBMS, Relational Model, ERD and SQL

Q1. Who is a DBA? What are the responsibilities of a DBA?

Ans: A database administrator (short form DBA) is a person responsible for the design, implementation, maintenance and repair of an organization's database. They are also known by the titles Database Coordinator or Database Programmer, and is closely related to the Database Analyst, Database Modeller, Programmer Analyst, and Systems Manager.

The role includes the development and design of database strategies, monitoring and improving database performance and capacity, and planning for future expansion requirements. They may also plan, co-ordinate and implement security measures to safeguard the database.

Q2: What is a data model? List the types of data model used.

Ans: A database model is the theoretical foundation of a database and fundamentally determines in which manner data can be stored, organized, and manipulated in a database system. It thereby defines the infrastructure offered by a particular database system. The most popular example of a database model is the relational model. types of data model used

- Hierarchical model
- Network model
- Relational model
- Entity-relationship
- Object-relational model
- Object model

Q3: What is data base management system?

Ans

- A database management system (DBMS) is a software package with computer programs that control the creation, maintenance, and the use of a database.
- It allows organizations to conveniently develop databases for various applications by database administrators (DBAs) and other specialists.
- A database is an integrated collection of data records, files, and other database objects.
- A DBMS allows different user application programs to concurrently access the same database. DBMSs may use a variety of database models, such as the relational model or object model, to conveniently describe and support applications.
- It typically supports query languages, which are in fact high-level programming languages, dedicated database languages that considerably simplify writing database application programs.
- Database languages also simplify the database organization as well as retrieving and presenting information from it.
- A DBMS provides facilities for controlling data access, enforcing data integrity, managing concurrency control, recovering the database after failures and restoring it from backup files, as well as maintaining database security.

Q4: What are the disadvantages of file processing system?

Ans: The disadvantages of file processing systems are

a) Data redundancy and inconsistency
b) Difficulty in accessing data
c) Data isolation
d) Integrity problems
e) Atomicity problems
f) Concurrent access anomalies

Q5: What are the advantages of using a DBMS?

Ans: The advantages of using a DBMS are

a) Controlling redundancy
b) Restricting unauthorized access
c) Providing multiple user interfaces

d) Enforcing integrity constraints.
e) Providing back up and recovery

Q 6. Give the levels of data abstraction.

a) Physical level
b) Logical level
c) View level
Explain all the three.

Q 7. Define instance and schema.

Instance: Collection of data stored in the data base at a particular moment is called an Instance of the database.

Schema: The overall design of the data base is called the data base schema.

Q8. Define the terms of Data base schemas.

Ans 1) Physical schema

2) logical schema.

Physical schema: The physical schema describes the database design at the physical level, which is the lowest level of abstraction describing how the data are actually stored.
Logical schema: The logical schema describes the database design at the logical level, which describes what data are stored in the database and what relationship exists among the data.

Q9. What is conceptual schema?

The schemas at the view level are called subschema's that describe different views of the database.

Q10. Define data model.

A data model is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints.

Q11. What is storage manager?

A storage manager is a program module that provides the interface between the low level data stored in a database and the application programs and queries submitted to the system.

Q12.What are the components of storage manager?

The storage manager components include

a) Authorization and integrity manager
b) Transaction manager
c) File manager
d) Buffer manager

Q13.What is the purpose of storage manager?

The storage manager is responsible for the following

a) Interaction with the file manager

b) Translation of DML commands in to low level file system commands
c) Storing, retrieving and updating data in the database

Q14.List the data structures implemented by the storage manager. .
The storage manager implements the following data structure
a) Data files
b) Data dictionary
c) Indices

Q15.What is a data dictionary?
A data dictionary is a data structure which stores meta data about the structure of the database ie. The schema of the database.

Q16.What is an entity relationship model?
The entity relationship model is a collection of basic objects called entities and relationship among those objects. An entity is a thing or object in the real world that is distinguishable from other objects.

Q17.What are attributes? Give examples.
An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set.
Example: possible attributes of customer entity are customer name, customer id, Customer Street, customer city.

Q18.What is relationship? Give examples
A relationship is an association among several entities.

Example: A depositor relationship associates a customer with each account that he/she has.

Q19. Define the terms i) Entity set ii) Relationship set
Entity set: The set of all entities of the same type is termed as an entity set.
Relationship set : The set of all relationships of the same type is termed as a relationship set.

Q20.What are the two types of participation constraint.
 Total: The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R.
 Partial: if only some entities in E participate in relationships in R, the participation of entity set E in relationship R is said to be partial.

Q21. Define the terms i) DDL ii) DML
DDL: Data base schema is specified by a set of definitions expressed by a special language called a data definition language.
DML: A data manipulation language is a language that enables users to access or manipulate
data as organized by the appropriate data model

Q22.Define single valued and multi valued attributes.
Single valued attributes: attributes with a single value for a particular entity are called single valued attributes.
Multi valued attributes: Attributes with a set of value for a particular entity are called multivalued attributes.

Q23.What are stored and derived attributes?
Stored attributes: The attributes stored in a data base are called stored attributes.
Derived attributes: The attributes that are derived from the stored attributes are called derived attributes.

Q24. What are composite attributes?
Composite attributes can be divided in to sub parts.

Q25. Define null values.
In some cases a particular entity may not have an applicable value for an attribute or if we do not know the value of an attribute for a particular entity. In these cases null value is used.

Q26. Define the terms i) Entity type ii) Entity set
Entity type: An entity type defines a collection of entities that have the same attributes.
Entity set: The set of all entities of the same type is termed as an entity set.

Q27. What is meant by the degree of relationship set?
The degree of relationship type is the number of participating entity types.

Q28. Define the terms.
i) Key attribute
ii) Value set
Key attribute : An entity type usually has an attribute whose values are distinct from each individual entity in the collection. Such an attribute is called a key attribute.
Value set: Each simple attribute of an entity type is associated with a value set that specifies the set of values that may be assigned to that attribute for each individual entity.
Q 29. What does the cardinality ratio specify?
Mapping cardinalities or cardinality ratios express the number of entities to which another
entity can be associated. Mapping cardinalities must be one of the following:
• One to one
• One to many
• Many to one
• Many to many
Q30. Define weak and strong entity sets.
Weak entity set: entity set that do not have key attribute of their own are called weak entity sets. Strong entity set: Entity set that has a primary key is termed a strong entity set.

Q31. List six major steps that you would take in setting up a database for a enterprise.
Answer: Six major steps in setting up a database for an enterprise are:
• Define the high-level requirements of the enterprise (this step generates a document known as the system requirements specification.)
• Define a model containing all appropriate types of data and data relationships.
• Define the integrity constraints on the data.
• Define the physical level.

• For each known problem to be solved on a regular basis (e.g., tasks to be carried out by clerks or Web users) define a user interface to carry out the task, and write the necessary application programs to implement the user interface.
• Create/initialize the database.

Q 32. Definitions

The **relational data model** is based on a collection of tables. The user of the database system may query these tables, insert new tuples, delete tuples, and update (modify) tuples. There are several languages for expressing these operations.

• The **schema** of a relation refers to its logical design, while an **instance** of the relation refers to its contents at a point in time. The schema of a database and an instance of a database are similarly defined. The schema of a relation includes its attributes, and optionally the types of the attributes and constraints on the relation such as primary and foreign key constraints.

• A **superkey** of a relation is a set of one or more attributes whose values are guaranteed to identify tuples in the relation uniquely. A candidate key is a minimal superkey, that is, a set of attributes that forms a superkey, but none of whose subsets is a superkey. One of the candidate keys of a relation is chosen as its **primary key**.

• A **foreign key** is a set of attributes in a referencing relation, such that for each tuple in the referencing relation, the values of the foreign key attributes are guaranteed to occur as the primary key value of a tuple in the referenced relation.

• A **schema diagram** is a pictorial depiction of the schema of a database that shows the relations in the database, their attributes, and primary keys and foreign keys.

• The **relational query languages** define a set of operations that operate on tables, and output tables as their results. These operations can be combined to get expressions that express desired queries.

- The **relational algebra** provides a set of operations that take one or more relations as input and return a relation as an output. Practical query languages such as SQL are based on the relational algebra but add a number of useful syntactic features.

Q 33. Difference between Data structure and DBMS.

Database is the collection of various in a formatted manner, like a table, which helps in storing large amount of information having common attributes or simply common headings . . . For eg : The attendance register of a class can be considered as a database which the information about all the students are stored in table format under the attributes like name , roll number , absent or present . . .

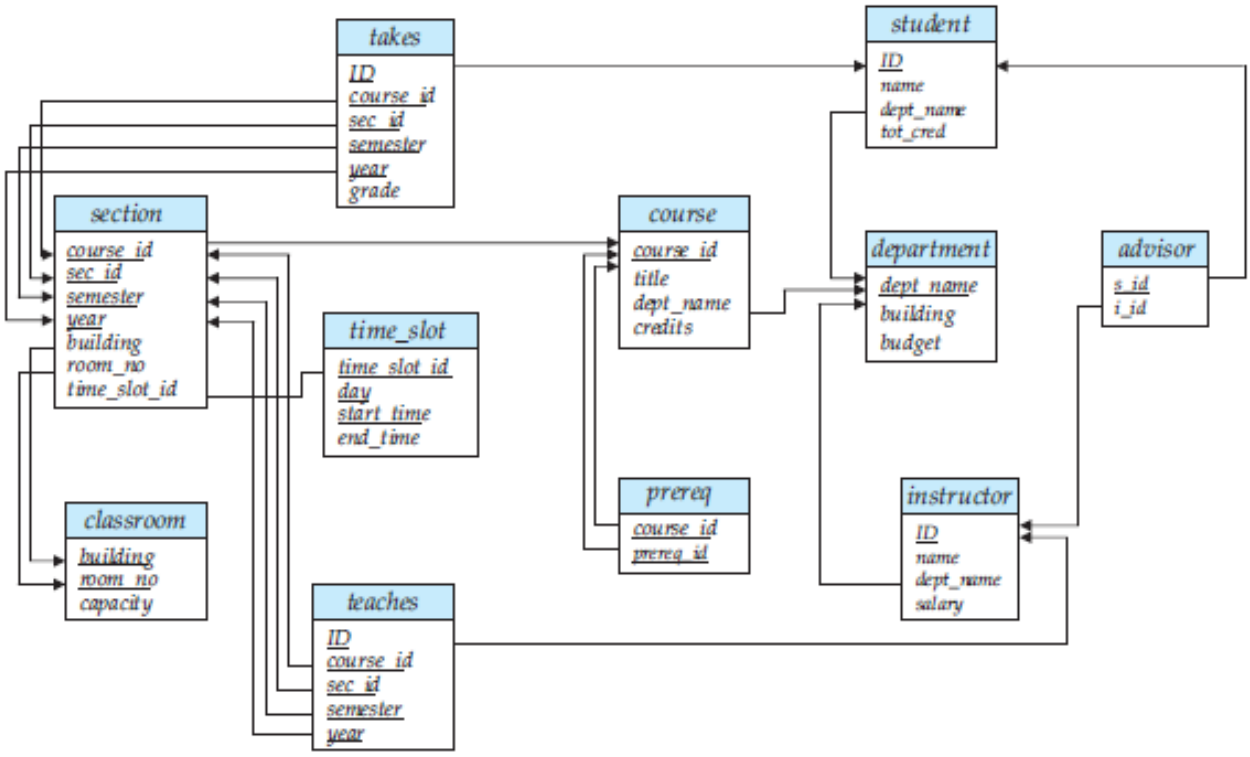
Data Structure is the way of arranging information in a database so that it becomes easy to access information whenever needed . Example: Consider an attendance register of a class which is not proper. i.e , the pages are not arranged according to a proper sequence . If you arrange it in a sequence (either date-wise or in any other sequence) , you form a data structure which has a particular sequence of accessing the details of the register.

Q 34. Kindly review the terms.

- Table
- Relation
- Tuple
- Attribute
- Domain
- Atomic domain
- Null value
- Database schema
- Database instance
- Relation schema
- Relation instance
- Keys
- Superkey
- Candidate key
- Primary key
- Foreign key
- Referencing relation
- Referenced relation
- Referential integrity constraint
- Schema diagram
- Query language
- Procedural language
- Nonprocedural language
- Operations on relations
- Selection of tuples
- Selection of attributes

- Natural join
- Cartesian product
- Set operations
- Relational algebra

Consider the following University Schemas and give the answers for below questions.



1. Find the titles of courses in the Comp. Sci. department that have 3 credits.

select *title*
from *course*
where *dept name* = 'Comp. Sci.'
and *credits* = 3

2. Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.

This query can be answered in several different ways. One way is as follows.
select distinct *student.ID*
from (*student* **join** *takes* **using**(*ID*))
join (*instructor* **join** *teaches* **using**(*ID*))
using(*course id*, *sec id*, *semester*, *year*)
where *instructor.name* = 'Einstein'

3. Find the highest salary of any instructor.
   select max(*salary*)
   from *instructor*

4. Find all instructors earning the highest salary (there may be more than one with the same salary).

select *ID*, *name*
from *instructor*
where *salary* = (**select max**(*salary*) **from** *instructor*)

5. Find the enrolment of each section that was offered in Autumn 2009.

One way of writing the query is as follows.

select *course id*, *sec id*, **count**(*ID*)
from *section* **natural join** *takes*
where *semester* = 'Autumn'
and *year* = 2009
group by *course id*, *sec id*

Note that if a section does not have any students taking it, it would not appear in the result. One way of ensuring such a section appears with a count of 0 is to replace **natural join** by the **natural left outer join** operation, Another way is to use a subquery in the **select** clause, as follows.

select *course id*, *sec id*,
(**select count**(*ID*)
from *takes*
where *takes.year* = *section.year*
and *takes.semester* = *section.semester*
and *takes.course id* = *section.course id*
and *takes.section id* = *section.section id*)
from *section*
where *semester* = 'Autumn'
and *year* = 2009
Note that if the result of the subquery is empty, the aggregate function **count** returns a value of 0.

6. **Find the maximum enrollment, across all sections, in Autumn 2009.**

One way of writing this query is as follows:
select max(*enrollment*)
from (**select count**(*ID*) **as** *enrollment*
from *section* **natural join** *takes*
where *semester* = 'Autumn'
and *year* = 2009
group by *course id*, *sec id*)

As an alternative to using a nested subquery in the **from** clause, it is possible to use a **with** clause, as illustrated in the answer to the next part of this question. A subtle issue in the above query is that if no section had any enrolment, the answer would be empty, not 0. We can use the alternative using a subquery, from the previous part of this question, to ensure the count is 0 in this case.

7. Find the sections that had the maximum enrollment in Autumn 2009.

The following answer uses a **with** clause to create a temporary view, simplifying the query.

with *sec enrollment* **as** (
select *course id*, *sec id*, **count**(*ID*) **as** *enrollment*
from *section* **natural join** *takes*
where *semester* = 'Autumn'
and *year* = 2009
group by *course id*, *sec id*)
select *course id*, *sec id*
from *sec enrollment*
where *enrollment* = (**select max**(*enrollment*) **from** *sec enrollment*)

It is also possible to write the query without the **with** clause, but the subquery to find enrollment would get repeated twice in the query.

Consider the Bank Database and write the queries as given below.

branch(branch name, branch city, assets)
customer (customer name, customer street, customer city)
loan (loan number, branch name, amount)
borrower (customer name, loan number)
account (account number, branch name, balance)
depositor (customer name, account number)

8. Find all customers of the bank who have an account but not a loan.

(**select** *customer name*
from *depositor*)
except
(**select** *customer name*
from *borrower*)

The above selects could optionally have **distinct** specified, without changing the result of the query.

9. Find the names of all customers who live on the same street and in the same city as "Smith".

One way of writing the query is as follows.

select *F.customer name*
from *customer F* **join** *customer S* **using**(*customer street*, *customer city*)
where *S.customer name* = 'Smith'

The join condition could alternatively be specified in the **where** clause, instead of using bf join .. using.

10. Find the names of all branches with customers who have an account in the bank and who live in "Harrison".

select distinct *branch name*
from *account* **natural join** *depositor* **natural join** *customer*
where *customer city* = 'Harrison'

As usual, the natural join operation could be replaced by specifying join conditions in the **where** clause.

Consider the Employee Database as given below and write the queries as given below.

employee (*employee name*, *street*, *city*)
works (*employee name*, *company name*, *salary*)
company (*company name*, *city*)
manages (*employee name*, *manager name*)

11. Find the names and cities of residence of all employees who work for First Bank Corporation.

select *e.employee name*, *city*
from *employee e*, *works w*
where *w.company name* = 'First Bank Corporation' **and**
w.employee name = *e.employee name*

12. Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000.

If people may work for several companies, the following solution will only list those who earn more than $10,000 per annum from "First Bank Corporation" alone.

select *
from *employee*
where *employee name* **in**
(**select** *employee name*
from *works*
where *company name* = 'First Bank Corporation' **and** *salary* > *10000*)

As in the solution to the previous query, we can use a join to solve this one also.

13. Find all employees in the database who do not work for First Bank Corporation.

The following solution assumes that all people work for exactly one company.

select *employee name*
from *works*
where *company name* 6= 'First Bank Corporation'

If one allows people to appear in the database (e.g. in *employee*) but not appear in *works*, or if people may have jobs with more than one company, the solution is slightly more complicated.

select *employee name*
from *employee*
where *employee name* **not in**
(**select** *employee name*
from *works*
where *company name* = 'First Bank Corporation')

14. Find all employees in the database who earn more than each employee of Small Bank Corporation.

The following solution assumes that all people work for atmost one company.

select *employee name*
from *works*
where *salary* > **all**
(**select** *salary*
from *works*
where *company name* = 'Small Bank Corporation')

If people may work for several companies and we wish to consider the *total* earnings of each person, the problem is more complex. It can be solved by using a nested subquery, but we illustrate below how to solve it using the **with** clause.

with *emp total salary* **as**
(**select** *employee name*, **sum**(*salary*) **as** *total salary*
from *works*
group by *employee name*)
select *employee name*
from *emp total salary*
where *total salary* > **all**
(select *total salary*
from *emp total salary, works*
where *works.company name* = 'Small Bank Corporation' **and**
emp total salary.employee name = *works.employee name*)