CMR INSTITUTE OF TECHNOLOGY 

MCA- Internal Assesment Test – I – Answer Key

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sub:** | **Database Management System** | **Sem: 2** |  | **Code:** | **16MCA23** |

|  |  |
| --- | --- |
| Part I-1(a) | . Explain in detail actors on the scene and workers behind the scene [10]  **Actors on the scene:**  1.**Database Administrators :** The DBA is responsible for authorizing access to the database,  for coordinating and monitoring its use, and for acquiring software and hardware resources  as needed. The DBA is accountable for problems such as breach of security or poor system  response time. In large organizations, the DBA is assisted by a staff that helps carry out these functions.  2 **Database Designers :** Database designers are responsible for identifying the data to be stored  in the database and for choosing appropriate structures to represent and store this data. These  tasks are mostly undertaken before the database is actually implemented and populated with  data. It is the responsibility of database designers to communicate with all prospective  database users, in order to understand their requirements, and to come up with a design that  meets these requirements.  3 **End Users :** End users are the people whose jobs require access to the database for querying,  updating, and generating reports; the database primarily exists for their use. There are several  categories of end users:   * Casual end users occasionally access the database, but they may need different information   each time. They use a sophisticated database query language to specify their requests and  are typically middle- or high-level managers or other occasional browsers.   * Naive or parametric end users make up a sizable portion of database end users. Their main   job function revolves around constantly querying and updating the database, using standard types of queries and updates—called canned transactions—that have been carefully programmed and tested.   * Sophisticated end users include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS so as to implement their applications to meet their complex requirements. * Stand-alone users maintain personal databases by using ready-made program packages]   that provide easy-to-use menu- or graphics-based interfaces. An example is the user of  a tax package that stores a variety of personal financial data for tax purposes.  4 **System Analysts and Application Programmers (Software Engineers) :** System analysts determine the requirements of end users, especially naive and parametric end users, and develop specifications for canned transactions that meet these requirements. Application programmers implement these specifications as programs; then they test, debug, document, and maintain these canned transactions. Such analysts and programmers (nowadays called software engineers) should  be familiar with the full range of capabilities provided by the DBMS to accomplish their tasks.  **Workers behind the scene :** |
|  |  |
|  | • **DBMS system designers and implementers** are persons who design and implement the DBMS modules and interfaces as a software package. A DBMS is a complex software system  that consists of many components or modules, including modules for implementing the catalog,  query language, interface processors, data access, concurrency control, recovery, and security.  The DBMS must interface with other system software, such as the operating system and compilers  for various programming languages.  • **Tool developers** include persons who design and implement tools—the software packages that facilitate database system design and use, and help improve performance. Tools are optional  packages that are often purchased separately. They include packages for database design,  performance monitoring, natural language or graphical interfaces, prototyping, simulation, and  test data generation. In many cases, independent software vendors develop and market these  tools.  • **Operators and maintenance personnel** are the system administration personnel who are  responsible for the actual running and maintenance of the hardware and software environment  for the database system. |

**2(a) Explain the characteristics of Database Approach. [7]**

* + 1. Self-describing nature of a database system.
    2. Insulation between programs and data, and data abstraction.
    3. Support of multiple views of the data.
       1. Sharing of data and multiuser transaction processing.

**2(b) Define database management system [3]**

A **database management system (DBMS)** is a collection of programs that enables users to create and maintain a database. The DBMS is hence a general-purpose software system that facilitates the processes of defining, constructing, and manipulating databases for various applications. Defining a database involves specifying the data types, structures, and constraints for the data to be stored in the database. Constructing the database is the process of storing the data itself on some storage medium that is controlled by the DBMS. Manipulating a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes in the mini world, and generating reports from the data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Part II-3  (a) | | **Define data independence? Explain the various types of data independence**. [6]  A major objective for three-level architecture is to provide data independence, which  means that upper levels are unaffected by changes in lower levels.   |  |  |  | | --- | --- | --- | | There are two kinds of data independence: | | **(2)** | | • | Logical data independence |  | | • | Physical data independence |  | | | |
| **Logical Data Independence:** | | **(2)** |

1. Logical data independence indicates that the conceptual schema can be changed without affecting the existing external schemas. The change would be absorbed by the mapping between the external and conceptual levels. Logical data independence also insulates application programs from operations such as combining two records into one or splitting an existing record into two or more records. This would require a. change in the external/conceptual mapping so as to leave the external view unchanged.

|  |  |
| --- | --- |
| **Physical Data Independence:** | **(2)** |

1. Physical data independence indicates that the physical storage structures or devices could be changed without affecting conceptual schema. The change would be absorbed by the mapping between the conceptual and internal levels. Physic 1data independence is achieved by the presence of the internal level of the database and the n, lPping or transformation from the conceptual level of the database to the internal level. Conceptual level to internal level mapping, therefore provides a means to go from the conceptual view (conceptual records) to the internal view and hence to the stored data in the database (physical records).

|  |  |
| --- | --- |
| **3(b) Explain DBMS interfaces** | **(4)** |

* + - Menu based
    - Form based
    - GUI
    - Natural Language Interface
    - Speech I/O
    - Interfaces Parametric users
    - Interface for DBA

**4(a) What are the advantages of using the DBMS approach [10]**

1.Controlling Redundancy

2.Restricting Unauthorized Access

3.Providing Persistent Storage for Program Objects and Data Structures.

4.Permitting Inference and Actions Using Rules

5 Providing Multiple User Interfaces

6 Representing Complex Relationships Among Data

7 Enforcing Integrity Constraints

8 Providing Backup and Recovery

|  |  |
| --- | --- |
| Part III-5(a) | Explain three – schema Architecture [6]   * **internal level**: has an internal/physical schema that describes the physical storage   structure of the database using a low-level data model)   * **conceptual level**: has a conceptual schema describing the (logical) structure of the   whole database for a community of users. It hides physical storage  details, concentrating upon describing entities, data types, relationships,  user operations, and constraints. Can be described using either high-level  or implementation data model.   * **external/view level**: includes a number of external schemas (or user views), each of   which describes part of the database that a particular category of users is  interested in, hiding rest of database. Can be described using either  high-level or implementation data model. (In practice, usually described  using same model as is the conceptual schema.) |
| (b) | Explain schema and instance with example [4]  The description of a database is called the database schema, which is specified during design and is not expected to change often. The actual data stored in the database probably changes often. The  data in the database at a particular time is called the state of the database, or a snapshot. Application requirements change occasionally, which is one of the reasons why software maintenance is important. On such occasions, a change to a database's schema may be called for. An example would be to add a Date\_of\_Birth field/attribute to the STUDENT table. Making changes to a database schema is known as schema evolution. |

6(a) Discuss the classification of Database Management system. [6]

* 1. Centralized database system: the DBMS and database are stored at a single site that is used by several other systems too
  2. Distributed database system: the actual database and the DBMS software are distributed from various sites that are connected by a computer network.
  3. Heterogeneous distributed database system: different sites might use different DBMS software, but there is additional common software to support data exchange between these sites.
  4. Homogeneous distributed database systems: use the same DBMS software at multiple sites.
  5. Multiuser database system: a database management system which supports multiple users concurrently.
  6. Object-oriented data model: a database management system in which information is represented in the form of objects as used in object-oriented programming.
  7. Single-user database system: a database management system which supports one user at a time.
  8. Traditional models: data models that preceded the relational model.

**6(b) Explain two-tier Client/Server Architecture for DBMS** [4]

The user interface programs and application programs can run on the client side. When DBMS access is required, the program establishes a connection to the DBMS (which is on the server side); once the connection is created, the client program can communicate with the DBMS. A standard called Open Database Connectivity

 (ODBC) provides an application programming interface (API), which allows client-side programs to call the DBMS, as long as both client and server machines have the necessary software installed.

 The different approach to two-tier client/server architecture was taken by some object-oriented DBMSs, where the software modules of the DBMS were divided between client and server in a more integrated way. For example, the server level may include the part of the DBMS software responsible for handling data storage on disk pages, local concurrency control and recovery, buffering and caching of disk pages, and other such functions. Meanwhile, the client level may handle the user interface; data dictionary functions; DBMS interactions with programming language compilers; global query optimization, concurrency control, and recovery across multiple servers; structuring of complex objects from the data in the buffers; and other such functions. In this approach, the client/server interaction is more tightly coupled and is done internally by the DBMS modules—some of which reside on the client and some on the server—rather than by the users/programmers. The exact division of functionality can vary from system to system. In such a client/server architecture, the server has been called a data server because it provides data in disk pages to the client. This data can then be structured into objects for the client programs by the client-side DBMS software.

 The architectures described here are called two-tier architectures because the soft-ware components are distributed over two systems: client and server. The advantages of this architecture are its simplicity and seamless compatibility with existing systems.

Part IV-7(a).Define the following terms with an example. [10]

1, Cardinality ratio 2, Relationship type 3, Weak entity set 4,Entity set

5, Participation

**Cardinality ratio :** In database design, the cardinality or fundamental principle of one data

table with respect to another is a critical aspect. The relationship of one to the other must be

precise and exact between each other in order to explain how each table links together.

**Relationship type :** A relationship is any association, linkage, or connection between the

entities of interest to the business; it is a two directional, significant association between two

entities, or between an entity and itself.

**Weak entity set** : An entity set that does not have a primary key is referred to as a weak entity set.

The existence of a weak entity set depends on the existence of a strong entity set; it must relate

to the strong set via a one-to-many relationship set.

**Entity set :** An entity is an object that exists and is distinguishable from other objects. For instance,

John Harris with S.I.N. 890-12-3456 is an entity, as he can be uniquely identified as one particular

person in the universe.

An entity may be concrete (a person or a book, for example) or abstract (like a holiday or a concept).

An entity set is a set of entities of the same type (e.g., all persons having an account at a bank).

**Participation :** A participation constraint defines the number of times an [object](http://osm7.cs.byu.edu/OSA/object.html) in an [object class](http://osm7.cs.byu.edu/OSA/objectClass.html)

can participate in a connected [relationship set.](http://osm7.cs.byu.edu/OSA/relationshipSet.html) Every connection of a relationship set must

have a participation constraint. However, participation constraints do not apply to relationships.

**8(a) Explain the database system environment** [6]

DBMS Component modules

Database System Utilities

Tools, application environments and communication facilities

Application development environments

**8(b) Explain the terms. 1, Composite attribute . 2. Key attribute of an Entity type** [4]

1.**A composite attribute**: If a set of attributes possesses this property, we can define a composite attribute that becomes a key attribute of the entity type. Notice that a composite key must be minimal; that is, all component attributes must be included in the composite attribute to have the uniqueness property.

2**. Key attribute** of an Entity type is an attribute whose values are distinct for each individual entity in the collection. Such an attribute is called a key attribute, and its values can be used to identify each entity uniquely. For example, the Name attribute is a key of the COMPANY entity type because no two companies are allowed to have the same name. For the PERSON entity type, a typical key attribute is SocialSecurityNumber .

**Part V-9(a) Explain the different types of attributes, which occur in ER model. [10]**

* simple/atomic vs. composite
* single-valued vs. multi-valued (or set-valued)
* stored vs. derived (Note from instructor: this seems like an implementation detail that ought not be considered at this (high) level of abstraction.)

A composite attribute is one that is composed of smaller parts. An atomic attribute is indivisible or indecomposable.

Example 1: A Birth Date attribute can be viewed as being composed of (sub-)attributes month, day, and year (each of which would probably be viewed as being atomic).

To describe the structure of a composite attribute, one can draw a tree (as in the aforementioned Figure 7.4). In case we are limited to using text, it is customary to write its name followed by a parenthesized list of its sub-attributes. For the examples mentioned above, we would write

BirthDate(Month, Day, Year)

Address(StreetAddr(StrNum, StrName, AptNum), City, State, Zip)

Single- vs. multi-valued attribute: Consider a PERSON entity. The person it represents has (one) SSN, (one) date of birth, (one, although composite) name, etc. But that person may have zero or more academic degrees, dependents, or (if the person is a male living in Utah) spouses! How can we model this via attributes Academic Degrees, Dependents, and Spouses? One way is to allow such attributes to bemulti-valued (perhaps set-valued is a better term), which is to say that we assign to them a (possibly empty) set of values rather than a single value.

To distinguish a multi-valued attribute from a single-valued one, it is customary to enclose the former within curly braces (which makes sense, as such an attribute has a value that is a set, and curly braces are traditionally used to denote sets). Using the PERSON example from above, we would depict its structure in text as

PERSON(SSN, Name, BirthDate(Month, Day, Year), { AcademicDegrees(School, Level, Year) }, { Dependents }, ...)

A more complicated example of a complex attribute is AddressPhone . Its structure is given by

{ AddressPhone( { Phone(AreaCode, Number) }, Address(StrAddr(StrNum, StrName, AptNum), City, State, Zip))}

**10 . Draw an ER schema diagram for the Company database which contains [10]**

**department** , employees and projects.

1, A department controls a number of projects.

2, One employee manages the department. An employee assigned to one department.

We keep track of the direct supervisor of each employee

