1.
Identify and summarize five issues to be addressed in database administration. [12]

Chapter 16 discusses the following database administration issues:

Database Installation, Creation, and Configuration: These activities range from simplicity on smaller products to complexity for more sophisticated products such as Oracle and DB2, where the user has several options to consider. Configuration typically involves issues such as location of the database and its related files, communication issues for server and client (for client-server databases) in a multi-user environment, physical structure of the database, logical structure of the database, and access rights of users of the database.

Database Security: This typically involves access of the database, its resources, and data. This includes management of user profiles, user accounts, system privileges, and database object privileges.

Database Management : This typically involves any combination of the following activities: reorganizing existing database tables and indexes, deleting unnecessary indexes or moving other objects backup and recovery of database objects, making alterations to the database itself, making alterations to database components (tablespaces, datafiles, tables, procedures, etc.), creating additional database objects (tablespaces, datafiles, tables, users, indexes, procedures, etc.), training users, and database performance tuning.

Database Backup and Recovery: In general, database backup and recovery refers to the various strategies and procedures involved in protecting a database against data loss, and reconstructing the data should that loss occur. The reconstructing of data is achieved through media recovery, which refers to the various operations involved in restoring, rolling forward, and rolling back a backup of database files.

Database Tuning: This involves changing the physical and/or logical structure of the database to ensure optimum performance.

Database Removal: For sophisticated products such as Oracle and DB2, database removal must follow a carefully planned procedure to avoid damaging or loosing subsequent access to the database.

2.

2.1 Give the definition of a distributed database system. [03]

According to section 17.1, a distributive database system consists of a collection of sites, connected via a communication network in which:

- Each site is a database system in its own right.
- The sites work together (*if necessary*) so that a user at any given site can access data at any other site as if the data resides of the host (*user's*) site.

2.2 What is the difference between database fragmentation and database replication. Cite a situation that would warrant each. [04]

Database fragmentation involves the partitioning of a relation into fragments that are stored at different sites (this may be done for one or more relations). As an example, an **Employee** table may be fragmented to store employee information at different locations – a partition for each location.

Database replication is the act of duplicating one or more relations at multiple sites for the purpose of autonomy and/or improved access. As an example, consider constructing an integrated software system for a college or university. The software engineering team may decide to implement subsystems such as Financial Management and Academic Management as independent but interoperable units. This would necessitate duplicating critical tables (such as **Department**, **Student**, **Course**, etc.) in both subsystems.

2.3 State and clarify four of Date's twelve rules for distributed databases. [12]

See section 17.3

2.4 Discuss two challenges to distributed databases, and alternate approaches for addressing each challenge. [08]

According to section 17.4, there are five well known challenges to distributed database systems. These are:

- Query Optimization
- Catalog Management
- Update Propagation
- Concurrency Control
- Transaction Management

See section 17.4 for more details.

3.

3.1 Describe two challenges to OO database management systems. [06]

Figure 18.2 of lecture 18 is repeated here. It lists five challenges to OO database management systems.

Figure 18.2: Challenges to Object-Oriented Database Management Systems

1. OT as proposed by its ardent proponents, encourages generalization and the building of inheritance hierarchies where objects consist of other objects (as in subtype, component and aggregation relationships. If pursued to their logical conclusion without pragmatic deviations (as discussed in lectures 3 and 5), the system could break down into a convoluted mess. Another related problem is that hierarchies, as we know them, do not lend themselves to representation of M:M relationships. Consider, for instance, an M:M relationship between **Academic Program** and **Course**. How do we model and implement this in the OO paradigm? Do we assume that academic programs contain courses or vice versa? History has shown that hierarchical based systems (such as CODASYL) are unsuited for complex distributed databases. For this reason, purely OO databases have been criticized as "CODASYL warmed over."

2. The OO model does not encourage the introduction of keys to uniquely identify objects, since object identifiers (OIDs) are generated (as internal address) at the point of instantiation. Note however, that OIDs do not obliterate the need for user keys, since end users still need to have a way of identifying objects separate and apart from internal addresses. The problem is further compounded when we have intermediate transient objects, or what in relational terminology is referred to as logical (virtual) objects such as a natural join record. In the OO model, there is no way to negotiate these situations without duplicating data into separate storage variables which must be maintained by the running program, thus increasing system overheads.

   The relational model gracefully handles these situations by allowing the user to define keys and surrogates (note that surrogates are not identical to OIDs). Then thanks to data independence, you can define multiple logical views on physical data. It has been argued that encapsulation is to the OO model what data independence is to the relational model. However, when analyzed, encapsulation is not a perfect replacement for data independence.

3. The OO model does not promote data sharing, a fundamental tenet of the relational model. At best, we could have encapsulated objects sharing their data contents with other objects, but in a distributed environment database (or mere network with one database server and several users), this would significantly increase the system overheads.

4. The matters of class, instance and collection are particularly difficult to negotiate. In the OO paradigm as we know, a class is essentially a complex data type; an instance is a specific object which belongs to at least one class. Without intense programmatic intervention, it would be difficult to determine a collection of objects belonging to a given class.

   The relational model has no such problem. A relational table defines the type and contains the collection of related data (objects) all in the same place. And as you are aware, this data can be shared and used to construct any number of logical perspectives as required by external end users.

5. The perfect encapsulation of data structure and operation in a data model is still for the most part, an ideal. We have seen complete achievement of this objective in the programming domain, but seldom in the database domain.

3.2 Describe one hybrid approach that facilitates the peaceful coexistence of relational databases and object technology. [04]

Section 18.4 describes two hybrid approaches:

Hybrid Approach
In the Hybrid Approach I, a relational database is accessed by an OO user interface. This means that the application development is done via an OO Programming Language (OOPL), CASE tool, or RAD tool. There is no shortage of OOPL (some of them pure OOPLs, others are hybrid OOPLs). The more popular ones include C++, Java and Object Pascal.

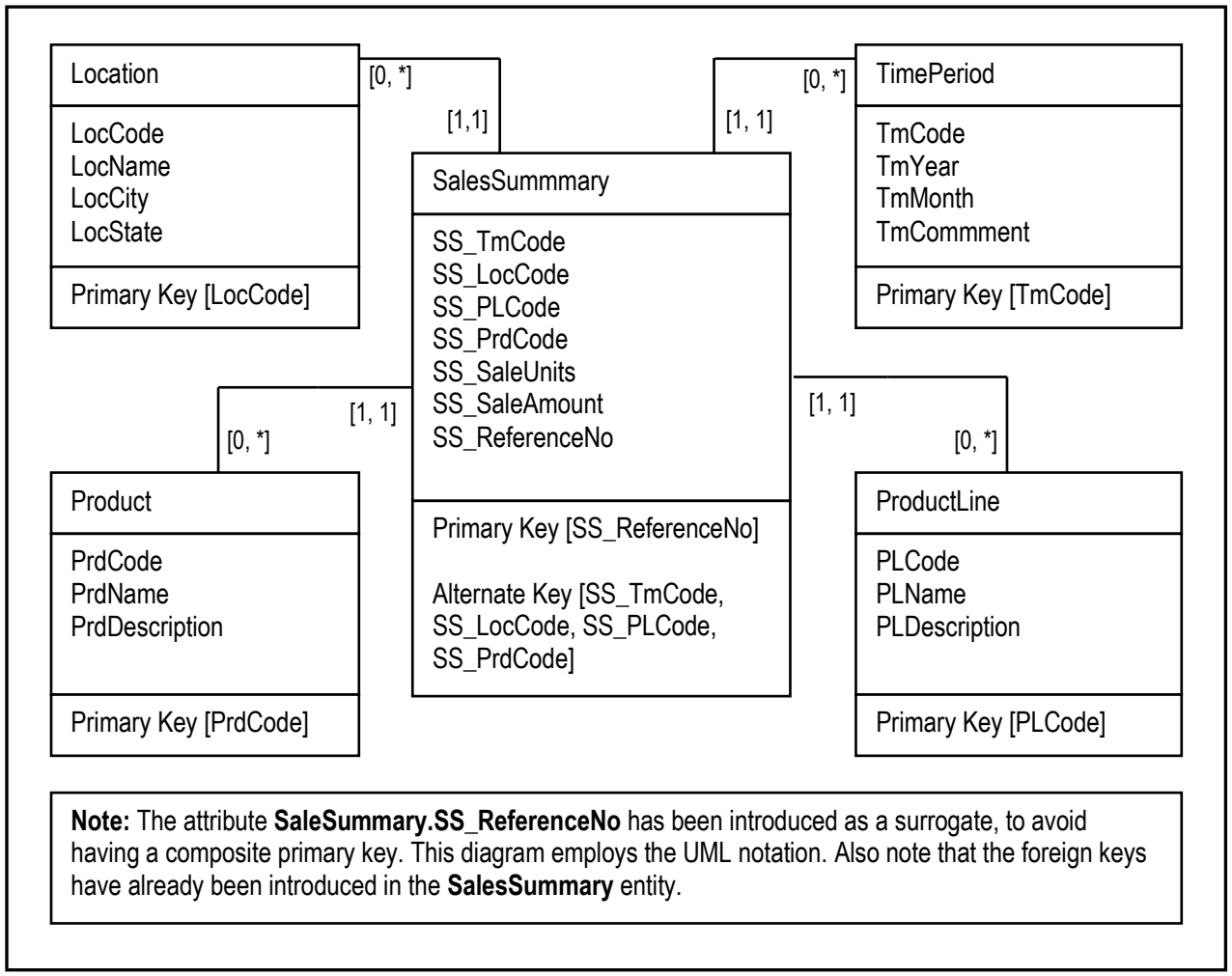
There are many object-oriented CASE (OO-CASE) tools and RAD tools that support this approach. They include (but are not confined to) Gupta Team Developer, Delphi, Oracle JDeveloper, WebSphere, etc. On the back-end, the leading relational DBMS suites are Oracle, DB2, MySQL, SQL Server, and PostgreSQL.

4.

Boch Toyota is a leading automobile distribution company with branches scattered across the US mainland. The company is desirous of keeping track of its sale of multiple product lines and products at various locations on a monthly and yearly basis. Sale is tracked in terms of number of units sold as well as the equivalent sale amount.

4.1 Construct a database conceptual schema that best represents this scenario, and illustrate it in an ERD. Your model may include additional attributes not mentioned in the case providing that you consider them essential. However, no additional entity should be introduced than those implied or mentioned in the case. [20]

What is required here is a star schema, as discussed in section 5.6 as well as section 19.3.3. A central fact table called **SalesSummary**, would store automobile sales, categorized related control tables **Location**, **ProductLine**, **Product**, and **TimePeriod**. Each control table forms a 1:M relationship with the fact table as illustrated in the ERD below (using UML notation).



4.2 This database schema appears in multiple operational databases operated by the company. However, it is desirable to preserve this schema for the entire organization over an extended time horizon, where information aggregated from the various operational databases will be stored. Based on what you have learned in this course, propose a relevant solution for this problem. [02]

The required solution is a data warehouse that aggregates data from the operational databases, and stores it over a longer time period.

4.3 Using an appropriate diagram propose a suitable architecture for your solution. Provide a brief clarification of the diagram. [12]

The preferred architecture is a data warehouse with a staging area. This staging area is used to extract the data from the various operational databases, and transform it to the schema used in the data warehouse. Finally, the information is loaded into the data warehouse. This is illustrated in the following diagram (taken from figure 19.5 of the text).