1. What is the difference between controlled and uncontrolled redundancy?

If there are multiple locations where the same data is stored, that data is redundant. This redundancy can either be controlled by the DBMS or uncontrolled. If the redundancy is controlled, it means that the DBMS makes sure to keep these multiple copies of the data in sync; if there needs to be a change or update to this data, the DBMS updates all of the copies of the redundant data. In an uncontrolled system, the DBMS does not update all of the copies, and the data can get out of sync.

1. Give 2 or more examples of systems in which it may make sense to use traditional file processing instead of a database approach. Explain why you chose these systems.

One example of a system that should not use a database is a video game, storing saved games locally. This data will include a lot of information about the state of the game when the user saved. However, this data is not relational in nature, and there is not a practical use case for trying to normalize this data out of these saved states into tables to make it query-able. Each individual save is independent from all the other saved states. On top of this, there is likely only ever going to be one user.

A second example is a student saving their assignments for the classes on their local machine. Again, in this scenario, it would be possible to create a database system giving attributes to things like dates or the class each assignment is from, but it would not provide a benefit for the user. There are not scenarios where the user would want to write queries to find information about these files, nor are the relationships between the files very useful outside of purely keeping the files organized. Again, this is also a one user scenario.

1. This chapter has described several major advantages of a database system. What are two disadvantages?

Database setup can require a lot of capital investment. While there are many free options, licensing full featured enterprise software can be prohibitively expensive. You may also have to be paying one or more DBAs and engineers to manage the DB, depending on its size. That’s not to say alternatives to a database would not require capital, but cost analysis would have to be performed.

A second disadvantage is complexity. To have a useful database, the entities and relationships must be engineered to suit the user’s needs, ensuring fast and efficient data access. For some businesses, this can be a monumental, time consuming task; designing a data base schema is not an easy thing. Not only does the enterprise have to design the schema, but there has to be time spent on estimating how much hardware is going to be required to store the data, how the network will need to be set up for the users to access the data, and much, much more (and then, again, the capital to purchase the hardware).

1. List four to six major steps that you would take in setting up a database for a particular enterprise.
   1. First, the business needs must to be evaluated. There are several key questions that should be answered. Who is going to be using the database? What kind of data do they need to store/access/maintain? How much data needs to be stored, and how fast does this data grow? How long does the data need to be kept? This step builds a specification of the user use cases and requirements.
   2. Given the data that is being used, the particular database type and software must be selected. Depending on the type of data, one might implement an XML database or graph/object database instead of a relational database. Additionally, the actual database software should be selected, taking into account features required and cost. For example, an enterprise might study the cost and features of Oracle, DB2, and MySQL, to find that Oracle DB suits the needs the best.
   3. With all this information, hardware requirements should be determined, and hardware acquired (this is the physical design of the database). Three primary concerns might include how much processing power we might need to serve our users, how much storage we need for our data and backup, and network design to make sure users can query this data quickly and efficiently, avoiding bottlenecks.
   4. The database must be also be designed at the logical level. The data is compartmentalized into distinct but related entities, in order to reduce unneeded redundancy and optimize performance. This should be done with regard to the spec created earlier, focusing on how the users are going to be querying the data, making sure the design meets their needs. One might employ the entity relationship model to plan the schema.
   5. Finally, the database must also be installed, initialized, and populated. Necessary data pipelines must be set up, for example as auto-populating data into the database from XML files in a directory. Additionally, user accounts and privileges must be set up, so the enterprise users can actually use the database once data is loaded. Any database applications that users might be using (for those who don’t use the query language directly) should also be set up at this stage.
2. List at least 3 different entities that a university would maintain, beyond those listed in the chapter.
   1. A university will have an entity that has information about registered clubs and their membership. These could be student leadership clubs, outing clubs, or any other clubs officially registered with the University.
   2. Universities will maintain an entity for room reservations, for both internal and external events. These events might be job fairs, businesses renting out large university venues, or things of that nature. The university will need to maintain a record of when these spaces are reserved.
   3. Universities will need an entity or entities regarding meal plans, if there are food services. Many on-campus students will be paying for one of many different meal plans, and the university will need to have information about which meal plan a given student has, as well as track usage.
3. Explain the concept of physical data independence and its importance in database systems.

Physical data independence describes the fact that physical data is independent to logical data. That is, information about *how* data is stored, and *where* data is stored, is completely independent to the logical structure of the data. This abstraction is part of what makes databases so appealing; the end user does not need to know anything about the hardware behind the database in order to access the data. The logical design of the database can be done without regard to the actual physical storage structure. Additionally, if there are any changes to the hardware, this does not negatively impact the user in any way—these changes are completely transparent!

1. What are the main functions of a database administrator?

Overall, the DBA has general control of the whole database system. As part of this role, there are several important functions the DBA must perform. First, the DBA is responsible for the schema definition. When new tables/data need to be added to the database, the DBA is responsible for this logical implementation (done using DDL). As part of this logical design, the DBA also defines how the logical data is stored, and how it is to be accessed. Thirdly, the DBA is in charge of authorization, making sure that the database users can only access data that they are permitted to access. Last but not least, the DBA is in charge of maintenance. The DBA will back up the database as needed, and make sure the database has all the storage it needs to operate. Also in the maintenance category, the DBA should make sure that very expensive jobs are not bogging down the system for other users. This could include killing these jobs, and/or helping the user craft a better optimized query (if it is a query the users might be using often). This said, if there is not an efficient way to access the data, the DBA might even consider a redesign of some relations, in order to make an inefficient use case more efficient.

1. Explain the difference between two-tier and three-tier architectures. Why would you use one over the other?

In a two-tiered architecture, a client interacts directly with the database. This contrasts with a three-tiered architecture, where a client interacts with an application server, which then in turn interacts with the database. In general, two-tiered systems are simpler to both implement and maintain, as there are few points of failure. Three-tiered systems are more complex, but also come along with some advantages. Because a three-tiered system is distributed by its very nature, they are often easier to scale. Additionally, the application middleware allows additional security (and features) to be implemented in order to protect and control data. For these reasons, three-tiered systems are often implemented in large web based applications, and other deployments where the user base is spread out. For smaller and more centralized applications (ex. A small local business), two-tier architectures are likely more desirable due to their simplicity.

1. Why are many organizations using object-oriented databases instead of relational ones?

Many organizations are using object-oriented databases due to the databases ability to implement more advanced, richer data types, where the normal tabular relationship doesn’t describe the data efficiently. That is, typically these databases are implemented instead of relational databases due to the nature of the data that the organizations are trying to model. For example, an object-oriented graph database might significantly better model things such as circuit schematics or navigational maps, both of which are cases where traditionally this data would be modeled as a graph rather than a table.

1. Define the following terms:
   1. Atomicity

Atomicity generally means all or nothing. Better yet, it means something is not reducible. That is, if there are several sub-steps that need to be completed for a given single logical procedure, every sub-step must be completed. If any sub-operations fail or cannot be completed, none of the steps should be completed. The overall operation is said to be indivisible.

* 1. Concurrency control

Concurrency control refers to a DBMS managing multiple simultaneous users, making sure the DB stays in a consistent state. If multiple users are attempting to read/write/edit the same DB tuples, the DBMS is responsible for making sure that no corruption occurs, and the user’s concurrent actions do not conflict. This is commonly achieved by delaying one user’s requests until the other user has completed their operation.

* 1. DBMS

DBMS stands for **D**ata**b**ase **M**anagement **S**ystem. The DBMS has many functions, for both end users as well DBAs. For end users, the DBMS is responsible for managing user interaction and authorization, data storage and retrieval (including insertion and deletion), as well as concurrency controls. When a DB is queried, the DBMS is responsible for figuring out how to execute this query efficiently and get the data back to the user. On the DBA side, the DBMS handles and maintains metadata, aids database backup, enforces constraints and maintains consistency. Essentially, a DBMS is the full-stack of software that runs and manages the entire database application, from DBA up to the end user.

* 1. Metadata

Metadata is data about data. That is, metadata contains information about the structure and setup of the database. Metadata includes information such as table and column names, index information, user information, and even procedures. This metadata is a key part of the database system, used to describe data structure to advanced users, and to aid the DBMS in making queries/transactions.

* 1. Schema

Database schema is the logical structure of a database. A database schema describes table names, column names, data types, indexes, and more. A schema is often used as an interchangeably with “logical design”. The schema itself does not include actual tuples of data but is rather a template or skeleton for the data.