Eric Zorn

10/8/17

ICT 4300

Professor/Dr. Lakahni

**Module 5 ICT 4300 Homework Questions**

1. **Question 1 (What are the pros and cons of using different types of database management system?) relational, object, and object relational**
   1. There are many pros and cons of using a database management system, no matter which type it may be. A relational database is a collection of data items organized as a set of formally-described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables. An object-oriented database management system or otherwise known as OODBMS, is a DMBS that supports the modelling and creation of data as object, similar to the languages like Java and C. This DBMS allows the support for classes of objects. OODBMS is known to still be in testing and in its infancy stage. The third at last type of DBMS is the object-relational database management system or otherwise known as ORDBMS. This is the idea that OODBs can be implemented on top of a relational database and is more common in available products that we know today. As mentioned in the notes, the relational databases were perfect when developers were using the procedural programming paradigm. When OOP came into the picture, it was hard to visualize the relation between an object in a program and the way that data is persisted about that object in the relational database. Data that is stored in relational databases is a poor representation of the real-world entities that were located in the different classes or objects. When the developers normalize the database, there tends to be relationships between the object and the database tables that do not go together. This would split data from an object into multiple tables which can become very confusing. Using relational databases make it impossible to represent the data from an object in a relational database. When using an OOP database, the object in the code and the database is complex with multiple different data types and uses one system-wide ID for identification of an object. The ID cannot be changed during the lifetime of the object. The ID is also independent of the any of the values or attributes found in the object table. The ID is the primary key on the other hand when you are talking about a relational database. You are able to store an object in a relational database. This would require mapping the class to one or more tables distributed over relations in a relational database. The negatives to using a fully relational database is that the database relies on a key that can repeat itself and the databases have to be normalized every time that there is a new field being entered into the database. This will even occur with an object relational database. Using an object-oriented database seems like the most logical choice because you can add different classes that are comprised of classes, variables, and methods that are unique to that table. Every chance that you have to add a new table is not difficult because it will be comprised with its own key. Also, like OOP, you can use inheritance for database tables as well.
2. **Why do you think object databases never took off?**
   1. I believe that object databases and object relational databases didn't take off for multiple reasons. First of all, OOP languages did not take off for quite a while when the technological boom occurred. Originally, as mentioned before, there was the use of procedural programming languages. Only within a somewhat recent time period, languages like Java, C, and many others became popular. The benefit to these languages is that they are all object-oriented. OOP languages are allowed to use their own private and public methods, as well as using inheritance for easier programming. Now that these methodologies are becoming more useful and the industry standard, it has made the idea of creating databases to be the same. You could relate relational data to procedural programming languages, while OOP languages are the same concepts as OOP database creation languages. I do not think that object oriented databases are gone or will never become popular. In fact, I believe that they will most likely be popular as the progression of the languages grow and the idea behind “classical” programming becomes more popularized. Building software in components and using inheritance shortens development time, while also making the understanding of the software flow much easier.
3. **What do you think is the future of relational databases, object databases, and object-relational databases?**
   1. I find that the idea behind either relational and object-relational databases are going to eventually die out and do not show potential for the future. The idea behind object databases has been around for a while and has not yet taken off. Stemming off of what I have mentioned in the second question, I believe that object oriented programming languages has just become popularized. I believe that only now will we start to see object databases popular. They will become popularized as the languages become more widely used like Java and C. This in fact has become the standard. Even languages like JavaScript that used to be unique, now are using the OOP method to produce the proper and most efficient programming and building software in components.