Database Systems Overview Homework

// Task 1- **What database models do you know?**

Database systems can be based on different data models or database models respectively. A data model is a collection of concepts and rules for the description of the structure of the database. Structure of the database means the data types, the constraints and the relationships for the description or storage of data respectively.

**The most often used data models are:**

* **Network Model and Hierarchical Model-** he network model and the hierarchical model are the predecessors of the relational model. They build upon individual data sets and are able to express hierarchical or network like structures of the real world.
* **Relational Model-** The relational model is the best known and in today’s DBMS most often implemented database model. It defines a database as a collection of tables (relations) which contain all data.   
  This module deals predominantly with the relational database model and the database systems based on it.
* **Object-oriented Model-** Object-oriented models define a database as a collection of objects with features and methods. A detailed discussion of object-oriented databases follows in an advanced module.
* **Object-relational Model - Object-** oriented models are very powerful but also quite complex. With the relatively new object-relational database model is the wide spread and simple relational database model extended by some basic object-oriented concepts. These allow us to work with the widely know relational database model but also have some advantages of the object-oriented model without its complexity.
* **Entity–relationship model (ER model )-** In [software engineering](http://en.wikipedia.org/wiki/Software_engineering), an entity–relationship model (ER model) is a [data model](http://en.wikipedia.org/wiki/Data_modeling) for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a [database](http://en.wikipedia.org/wiki/Database) such as a [relational database](http://en.wikipedia.org/wiki/Relational_database). The main components of ER models are [entities](http://en.wikipedia.org/wiki/Entities) (things) and the relationships that can exist among them, and databases.

// Task 2**-** **Which are the main functions performed by a Relational Database Management System (RDBMS)?**

* **Create**
* **Read**
* **Update**
* **Delete**

//Task 3**- Define what is "table" in database terms.**

* A table is a collection of related data held in a structured format within a database. It consists of fields (columns), and rows.
* In [relational databases](http://en.wikipedia.org/wiki/Relational_database) and [flat file databases](http://en.wikipedia.org/wiki/Flat_file_database), a table is a set of data elements (values) using a model of vertical [columns](http://en.wikipedia.org/wiki/Column_(database)) (which are identified by their name) and horizontal [rows](http://en.wikipedia.org/wiki/Row_(database)), the [cell](http://en.wikipedia.org/w/index.php?title=Cell_(database)&action=edit&redlink=1) being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows. Each row is identified by the values appearing in a particular column subset which has been identified as a [unique key](http://en.wikipedia.org/wiki/Unique_key) index.

// Task 4**- Explain the difference between a primary and a foreign key.**

|  |  |
| --- | --- |
| **Primary key** | **Foreign key** |
| Primary key uniquely identify a record in the table. | Foreign key is a field in the table that is primary key in another table. |
| Primary Key can't accept null values. | Foreign key can accept multiple null value. |
| By default, Primary key is clustered index and data in the database table is physically organized in the sequence of clustered index. | Foreign key do not automatically create an index, clustered or non-clustered. You can manually create an index on foreign key. |
| We can have only one Primary key in a table. | We can have more than one foreign key in a table. |

// Task 5**- Explain the different kinds of relationships between tables in relational databases.**

* **One-to-One Relationships-** In a one-to-one relationship, each row in one database table is linked to one and only one other row in another table. In a one-to-one relationship between Table A and Table B, each row in Table A is linked to another row in Table B. The number of rows in Table A must equal the number of rows in Table B.

## One-to-Many Relationships- In a [one-to-many relationship](http://www.databaseprimer.com/pages/relationship_1tox/), each row in the related to table can be related to many rows in the relating table. This effectively save storage as the related record does not need to be stored multiple times in the relating table.

## Many-to-Many Relationships- In a [many-to-many relationship](http://www.databaseprimer.com/pages/relationship_xtox/), one or more rows in a table can be related to 0, 1 or many rows in another table. A mapping table is required in order to implement such a relationship.

// Task 6**- When is a certain database schema normalized?   
What are the advantages of normalized databases?**

* Normalization is basically to design a database schema such that duplicate and redundant data is avoided. If some piece of data is duplicated several places in the database, there is the risk that it is updated in one place but not the other, leading to data corruption.
* The main advantage of normalisation is that it helps to reduce redundancy.

// Task 7**- What are database integrity constraints and when are they used?**

* Integrity constraints provide a way of ensuring that changes made to the database by authorized users do not result in a loss of data consistency.
* Integrity constraints are used to ensure accuracy and consistency of data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity. Many types of integrity constraints play a role in referential integrity (RI).

// Task 8**- Point out the pros and cons of using indexes in a database.**

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Speed up searching of values in a certain column or group of columns | Adding and deleting records in indexed tables is slower! |
| Indices can be built-in the table (clustered) or stored externally (non-clustered) | Create your own foreign key |
| Simple data model | Poor for complex data |

// Task 9**- What's the main purpose of the SQL language?**

* Manipulation of relational databases.

// Task 10- **What are transactions used for? Give an example**

* Database transaction is collection of SQL queries which forms a logical one task. For transaction to be completed successfully all SQL queries has to run successfully. Database transaction executes either all or none, so for example if your database transaction contains 4 SQL queries and one of them fails then change made by other 3 queries will be rolled back. This way your database always remain consistent whether transaction succeeded or failed.
* Suppose your account balance is 1000$ and you make a withdrawal request of 900$. At fourth step your balance is updated to 900$ and ATM machine stops working due to power outage. Once power comes back and you again tried to withdraw money you surprised by seeing your balance just 100$ instead of 1000$. This is not acceptable by any person in the world so we need transaction to perform such task.

// Task 11- **What is a NoSQL database?**

* Non-relational databases.

// Task 12- **Explain the classical non-relational data models**

A non-relational database is a database that does not incorporate the table/key model that relational database management systems (RDBMS) promote. These kinds of databases require data manipulation techniques and processes designed to provide solutions to big data problems that big companies face.

// Task 13- **Give few examples of NoSQL databases and their pros and cons.**

|  |  |
| --- | --- |
| **Redis** | |
| **Pros** | **Cons** |
| * Stores data in a variety of formats: list, array, sets and sorted sets | Super complex to configure -- requires consideration of data size to configure well |
| Pipelining!  Multiple commands at once | Master-slave architecture means if the master wipes out, and SENTINEL doesn't work, the system is sad |
| * Blocking reads -- will sit and wait until another process writes data to the cache | Lots o' server administration for monitoring and partitioning and balancing. |

|  |  |
| --- | --- |
| **CouchDB** | |
| **Pros** | **Cons** |
| Simplicity. You can store any JSON data, and each document can have any number of binary attachments. | Arbitrary queries are expensive. To do a query that you haven't created a view for, you need to create a temporary view. This can be solved to some extent by using Lucene. |
| Thanks to map/reduce, querying data is somewhat separated from the data itself. This means that you can index deeply within your data, and on whether or not something exists, and across types, without paying a significant penalty. You just need to write your view functions to handle them. | There's a bit of extra space overhead with CouchDB compared to most alternatives. |

|  |  |
| --- | --- |
| **Memcached** | |
| **Pros** | **Cons** |
| Low complexity | Doesn't do anything besides be an in-memory key/value store |
| Simple to configure | Caches sharded by client do not scale across AWS zones |
| Few command macros == simple to master | Unbalanced memcached clusters require a full system restart |