Advantages of a relational database Management System

The following are advantages of RDBMS:

1. Data is only stored once.

In the previous example, the city data was gathered into one table so now there is only one record per city. The advantages of this are

- No multiple record changes needed
- More efficient storage
- Simple to delete or modify details.
- All records in other tables having a link to that entry will show the change.

2. Complex queries can be carried out.

A language called SQL has been developed to allow programmers to 'Insert', 'Update', 'Delete', 'Create', 'Drop' table records. These actions are further refined by a 'Where' clause. For example

```
SELECT * FROM Customer WHERE ID = 2
```

This SQL statement will extract record number 2 from the Customer table. Far more complicated queries can be written that can extract data from many tables at once.

3. Better security.

By splitting data into tables, certain tables can be made confidential. When a person logs on with their username and password, the system can then limit access only to those tables whose records they are authorized to view. For example, a receptionist would be able to view employee location and contact details but not their salary. A salesman may see his team's sales performance but not competing teams.

4. Cater for future requirements.

By having data held in separate tables, it is simple to add records that are not yet needed but may be in the future. For example, the city table could be expanded to include every city and town in the country, even though no other records are using them all as yet. A flat file database cannot do this.

5. Ease of use:

The revision of any information as tables consisting of rows and columns is much easier to understand.

6. Flexibility:

Different tables from which information has to be linked and extracted can be easily manipulated by operators such as project and join to give information in the form in which it is desired.

7. Precision:

The usage of relational algebra and relational calculus in the manipulation of he relations between the tables ensures that there is no ambiguity, which may otherwise arise in establishing the linkages in a complicated network type database.

Summary - advantages of a relational database over flat file

- Avoids data duplication
- Avoids inconsistent records
- Easier to change data
- Easier to change data format
- Data can be added and removed easily
- Easier to maintain security.

Disadvantages of the Relational Database Management System

The following are some dis-advantages of RDBMS.

1. Performance:

A major constraint and therefore disadvantage in the use of relational database system is machine performance. If the number of tables between which relationships to be established are large and the tables themselves effect the performance in responding to the SQL queries.

2. Physical Storage Consumption:

With an interactive system, for example an operation like join would depend upon the physical storage also. It is, therefore common in relational databases to tune the databases and in such a case the physical data layout would be chosen so as to give good performance in the most frequently run operations. It therefore would naturally result in the fact that the lays frequently run operations would tend to become even more shared.

3. Slow extraction of meaning from data:

if the data is naturally organized in a hierarchical manner and stored as such, the hierarchical approach may give quick meaning for that data.

4. Data Complexity:

Data in an RDBMS resides in multiple tables, which are linked to each other through shared key values. An RDBMS does not force database designers to impose a coherent table structure; inexperienced programmers may design systems that create unnecessary complexity or limit the future development of the database through poorly chosen data types. The flexibility of an RDBMS presents a double-edged sword. Experienced designers work magic, but inexperienced designers wreak havoc on a company's data.

5. Broken Keys and Records:

Relational databases require shared keys to link information spread across several tables. For example, a customer table may include client demographics, with a unique index number identifying the record within the table. A sales table may identify the customer only by that index number. If the data types linking the keys are different, the database cannot link the records without additional rework by the report developer. Likewise, if a table lacks a unique key, the database may return inaccurate results. If the application accessing a database isn't coded to lock records during an edit, users could inadvertently corrupt data, leading to broken records.

6. Developer Expertise:

As the complexity of a relational database increases, the skill set required by the RDBMS administrator, various users and report developers also increases. A mission-critical database may require expertise that exceeds the budget of a small business; furthermore, if the developers did not uniformly engage in best practice design, a subsequent developer may not understand hidden intricacies that could lead to broken queries or inaccurate reports. This risk increases if database and application development is performed by different people.

7. Hardware Performance:

Complex queries require sophisticated processing power. Although most desktop computers can manage the databases of the size and complexity often encountered in a small business setting, a database with external data sources or very complex data structures may require more powerful servers to return results within an acceptable response time.

RDBMS Application

Relational data systems are proven market leaders in persistence solutions. They are mature and reliable choice for a general purpose solution. Moreover, RDBMS is supported by a huge number of applications and components and a large amount of human resources experienced in the technology.

However, RDBMS has some valuable advantages, which can't be underestimated:

- native Object Oriented language support (no object-relational impedance mismatch);
- native complex object structures and hierarchies support;
- easy refactoring and product evolution;
- zero-administration;
- Continuous regression testing.

Some of the most obvious usecases and advantages of RDBMS are listed below.

Object Oriented Development

The fact that RDBMS technology is the perfect match for using in OO development environment is obvious from the very name. Though RDBMS has a wide support in tools and methods of using in OO environment there will always be an overhead of the translation from the object to relational world, commonly known as object-relational impedance mismatch.

Embedded Applications

Embedded device applications need a small-footprint zero-administration database, which makes a perfect match with an RDBMS. In addition, they usually need a quick response-time, which can be better achieved by a native OO technology, when no object-relational conversion is needed.

Complex Object Structures And Hierarchies

In applications having to deal with deep object structures (tree or graph), RDBMS offers a much easier native way to store them. Trees or graphs cannot be easily translated to RDBMS structure, so that the application has to deal with often complex conversion algorithms, which are difficult to maintain and refactor. On the other hand, RDBMS can store these structures transparently without any additional coding.

The same is true for difficult inter-objects relationships. In RDBMS they are realized with foreign keys. In some cases, you will need to fetch object by object until you will reach the final relation. In RDBMS all you need is to specify an activation depth or use transparent activation to reach all the related objects through the top-object fields.

From the said above you can conclude that the applications using objects with collection members (one-to-many relationships) will also benefit from RDBMS technology.

Refactoring And Schema Evolution

Most applications evolve as the time goes. It means that their class and data structure (schema) can change. In applications using relational databases schema evolution is quite work-consuming process: the schema must be changed, the class structure must be changed and the query collection must be changed too. In RDBMS applications, only class structure is a subject of change and the process can be highly automated by using modern refactoring techniques available from IDEs.

Agile Development

RDBMS makes it possible to implement agile development process in your team:

- continuous refactoring;
- agile modeling;
- continuous regression testing;
- configuration management;
- developer "sandboxes".

The use of RDBMS technology complicates the adoption of these techniques due to the technical impedance mismatch, the cultural impedance mismatch, and the current lack of tool support.