DATABASE REPORT

17001172

Contents

Introduction – History of databases

Before the time computers were invented, the way in which data was stored was very limited. Before computers were designed or databases were a thing, individuals had to store information in some format, the most common way for the average person to store information was on paper which may have been in the format of a table. The first machine that read data storage was a stack of 'punched cards' in 1980, these cards were also known as IBM cards or Hollerith cards. During this period the inventor Herman Hollerith saw the potential of data to be stored digitally/ on a medium and therefore be read by a machine rather than the user. He designed this technology for the US Census, the technology was able to read data considering the number of punched holes in the stiff card, the holes were positioned in order to represent data or commands. The cards passed through a reader in which would break an electrical circuit (one per card-row) but a hole would briefly reconnect it again, reading a zero or one at each point along the card (Fair, 2017).

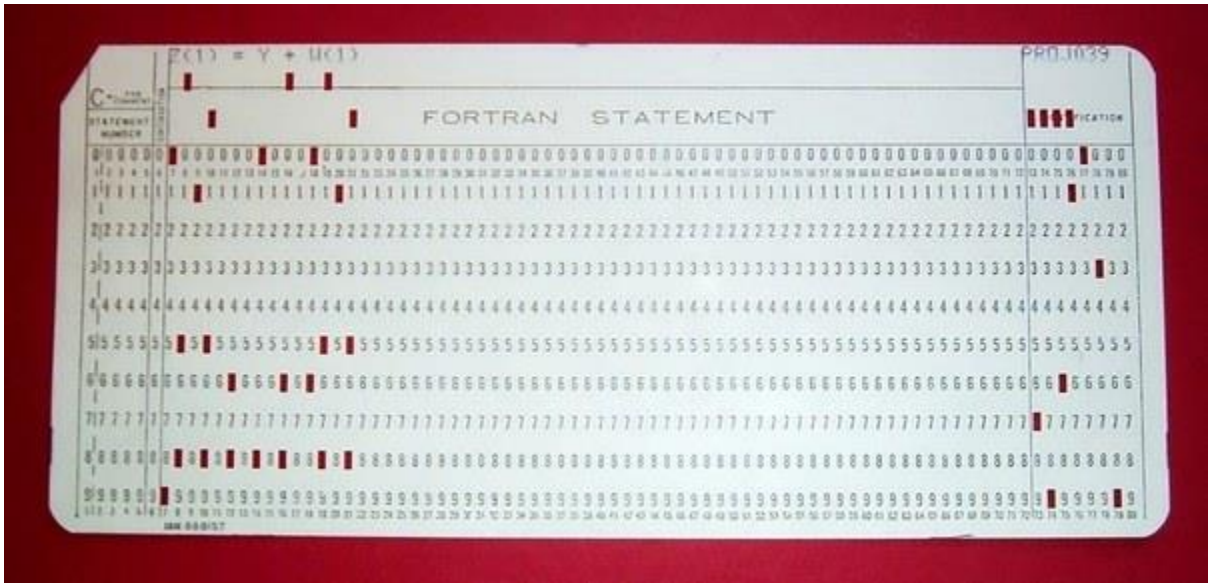


Figure 1 – What the cards looked like, (Fair 2007)

This was a very time consuming process where errors could easily take place -to solve this was the introduction of a machine which punched cards. Hollerith's enterprise then established the computer company IBM which was known to control the data processing market for most of the 20th century. These 80-column punch cards was the means of inputting data in 1970s. From this development computer scientists started to develop the idea of databases.

A database management system (DBMS) is said to be a 'class of software rivalling word processors and spreadsheets' (Gorman, 2015). An actual database itself is an organised group of data, this could be in any form such

as an address book, diary, word document on a computer or notes on your phone. These databases are required in order to have a quick access of information, for example in an address book an individual will have where a person lives, including postcode, potentially their telephone number with their name in order to see whose house it is. These tend to be done in alphabetical order which allows the address book owner to go to a particular letter to find the details quickly. There has been 5 database models which were established in order to keep data presented in a structured way, in particular for big businesses, organisations such as schools or youth clubs and shops. The first one was the flat file.

Flat File Model

What is it

According to Burleson (1999) prior to commercial database systems used today databases were 'nothing more than a loosely coupled collection of flat files'. Flat files is a type of a database where data is stored in one single table, it is much different from the other models which contain multiple tables. This initially gives the model its name, as the tables are not linked to other files making it impossible to link data items and cerate relationships. It stores data in a plain text file such as Microsoft Word, Access or any other spreadsheet application, and each line of data is known to hold just one record of data. To retrieve a record from the database it required the user to read the file in a linear manner, this caused problems as it was very time consuming and the user may have had a very lengthy file. Burleson (1999) continues to explain that the 'best media file to store a flat file database on was a magnetic type due to their natures both being linear'. In order to separate the different parts of data this is done by using delimiters. Delimiters are simply punctuation marks such as commas, full stops, semi colons and colons, they were seen to speed up this process of reading the database. Although a flat file database cannot contain multiple tables the database itself it may be used as a table in a relational database. Users may use flat file databases to store a list of phone numbers of their family members in, which would include their name and furthermore their number. However, with this model the data may be inserted twice causing redundancy. In the case of phone numbers there may be indexes used to identify each new person, there are two indexes used with flat file databases. These can be clustered where the database is then told to store close values close to another on the disk allowing quick retrieval of data. A non-clustered index has pointers to the physical rows, there may be many pointers on the different rows, which makes it confusing for the user.

Patient Id	Name	D.o.B	Gender	Phone	Doctor Id	Doctor	Room
134	Jeff	4-Jul-1993	Male	7876453	01	Dr Hyde	03
178	David	8-Feb-1987	Male	8635467	02	Dr Jekyll	06
198	Lisa	18-Dec-1979	Female	7498735	01	Dr Hyde	03
210	Frank	29-Apr-1983	Male	7943521	01	Dr Hyde	03
258	Rachel	8-Feb-1987	Female	8367242	02	Dr Jekyll	06

Figure 2- this is an example of a basic flat file table

(ictlounge.com)

To ensure a database is working effectively it must be designed logically, with poor design it can result in a poor working system and the information held in the database can be full of errors and mistakes. Two of the biggest problems within any type of database model is redundant data and anomalies. Sumathi et al (2010), explains that redundant data is the storage of the same information more than one and it could be removed from the system without any loss of information. Redundancy can bring about many problems with your RDBMS operations as when data is inserted into the database as new data but already exists (redundancy data) it must be duplicated. A massive problem faced years ago with this was when the data is updated for example if it was a student surname, all redundancy data had to be at the same time updated to reflect consistency and accuracy of information. They go on to state that this excess data can lead to anomalies which are insertion, deletion and updating. Anomalies occur during the development or changes to a DBMS. An anomaly is described as an 'erroneous change to data' and more specifically a single record. An inserting anomaly occurs whenever a record is added to a detailed table with no related record to fit that table an example of this is adding a student to a school without them being enrolled on a course.

Adding a record to a flat file database it required the whole database to be read to ensure there was no duplication of data, however with large databases this was a massive problem as individuals missed certain records. This lead to anomalies. If data needed to be updated in a flat file database, this was to be done manually and where there was a duplication of data this had to be done multiple times to ensure the data was up to date and accurate. If a family members number was added twice but the number changed the number would have to be updated on both rows. It can be easy to spot anomalies on small tables, however with large files it can be very difficult and therefore this model of database is not as popular due to the errors it comes with.

In order to find a number within the database it also requires the user to read through the whole database, starting from the beginning reading each record one at a time until the anticipated number was found.

Who developed it and when-

From the punched cards was the development of flat-file databases, once computers were developed and technology had grew rapidly. Databases were then created by the IBM Company where the data could be read, stored and sent to others who had the appropriate technology.

During this development it was apparent that flat files also had rules where the data must comply with particular standards – every column is restricted to a specific data type. The first row within a flat file is known as the field name, this makes it easier to recognise what data each field deals with and furthermore each row is a tuple where the attributes are stored about the entity.

Main applications this model used for-

The applications used for flat file databases can include Microsoft Excel, Microsoft Access, and FileMaker.

Is the model still used today-

Flat file databases do not tend to be used as much today because of the features the other models have to offer, however, for basic data storage such as team players names and ages from a football club this type of model may be used. The reason for this is due to its basic structure it is clear to see the members of the club and what age group they fall into.

However, if you were to insert the address of particular members this may cause problems with data security. Chickowski, 2010 states that the popular method of creating and exchanging flat files could leave 'sensitive data vulnerable'. With the use of flat file databases still popular in cases like discussed above it is known that the information the database presents is not secure in that 'many enterprises have sensitive data stored in flat files and floating around the network'. The reason for this data to be unsecure is because the file is unencrypted and therefore can be read by anyone who opens the file. They also state that with other databases such as relational there is 'at least some level of security built in', in these there is authentication because these are core elements of the platform.

Advantages-

The most common pro on using a flat file database as a management system is that they are easy to understand and implementation can be done without much skills. With the use of only one table this keeps all the data together and therefore the user can see where the next set of data for example for a person would be entered.

Disadvantages-

In flat file databases it can be difficult to see what data is truly stored in the database, making it easy to forget what was stored in it. Meaning that there is more data to read through so the computer may appear to be slower when accessing and searching through the data. This can be very inconvenient especially at times where the user needs information about the object presented quickly. If this is the case, the model of the database will slow down quick findings of data as it will all appear in one big table therefore the more data in the table the harder it will be to find information. Information may also be repeated or parts of repeated data may need updated which would breach the Data Protection Act 1998 as the data is not seen as up-to-date. The size of the file is much larger due to the unnecessary repeated data.

Recommendations-

This model carries more cons than pros, due to the age it was developed. It wouldn't be recommended for big organisations such as schools due to the high number of students making it harder to find information.

Hierarchical model

What it is-

The hierarchical data model is also known as the upside down model, this is because the data is structured to a tree-like manner. The data within this model is stored in individual records which are then linked together to give it the tree structure. A record is simply a collection of fields. Powell, (2006) states that these tables take on a 'child-parent relationship, each child table has a single parent table whereas a parent table can have multiple child tables'. Similarly like the real world, the child tables are completely dependent on those parent tables. Nodes in this model represents data records. This model uses relationships which are represented as a link or line in a hierarchical model diagram. It is important to note that this model is particularly special in that it enables its users and designers to distinguish the structure logically rather than physically. The top record that begins the hierarchy database is known as the root and in the case of the child-parent relationship represents the parent. Below that is then the children and below them is their children and so on. This is shown in figure 2 below.

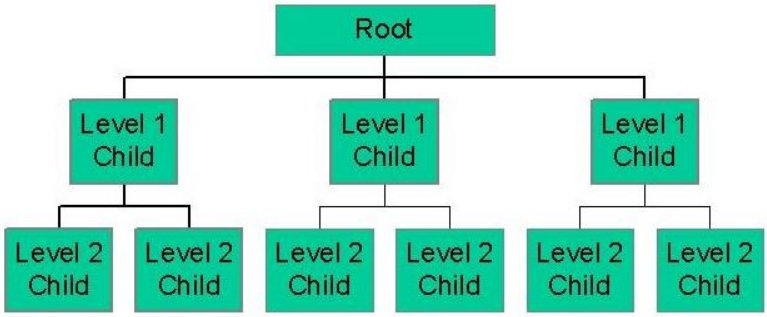


Figure 2 – Hierarchical Database Model

(DBMS Internals, n.d.)

The first set of children is known as level one, level two is the second set and this continues for the length of the database. The hierarchical path shows the order in which the data goes through the system, after the root on the first level the data is read from the left to right so child B then C.

Relationships emerge in this model, there can be 3 typical relationships between tables, in order to create relationships within the database you must have a relational database management system (RDBMD), these are one to one, one to many and many to many. However, in Crow's foot notation there are a few more. A relationship is a link between two relational database tables such as the table discussed above, where the foreign key in one table is the primary key in another.

In a one-to-one relationship this is where one record links to just one other record in another table. An example produced by FmHelp.Filemarker.com is shown below in figure 3.

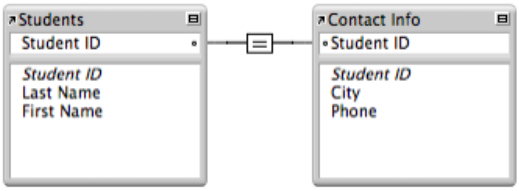
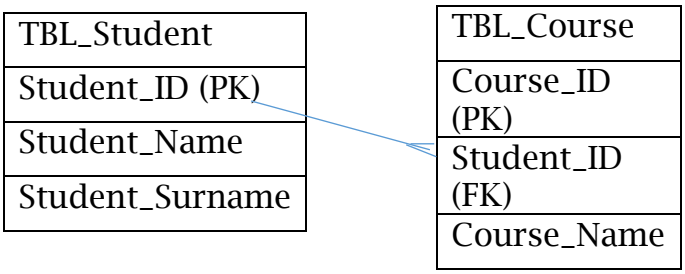


Figure 3 – one to one relationship

This figure shows a one to one relationship between two tables name students and contact information, the link between both the tables is the student ID. One student can only have one contact information and furthermore one contact information record can only relate to one of the students.

The one-to-many relationship means that one record in a table can be linked with one or more records in another table, for example each course can have many students however each student can only be enrolled in one course. As shown below.



In the diagram above there is a link between the two tables, in the student table the Student ID is the primary key and in the course table the primary key is Course ID, these ID's will be unique for each new record inserted into the table. There is a foreign key field in the course ID table which is student ID this allows many students to be enrolled on one course. This returns the students on that particular course. It is important to remember in this type of relationship that a student can only be enrolled in one course, whereas with the many-to-many relationship a student can be enrolled in many.

The many-to-many relationship occurs when more than one record in a table is linked with more than one record in another table. This relationship could exist between course and students where students are enrolled in many courses and the course has many students on it. However this relationship is difficult to implement as it tends to get very complicated and muddled as each record would contain the same ID's making it difficult to refer to the records. To prevent this two one to many relationships are used and is done by introducing a third table which is known as the junction table, this table must include the primary keys required to make it a many to many relationship. This table can be used to store attributes between the two tables needing linked.

There is another relationship which can be used in a relational database which is many to one, in this relationship there are multiple records which relate to one record which is in another table, just like the other relationships. This type of relationship can be understood as the look up table relationship. There is only one difference between this relationship and the one to one and that is that there is no primary key used as a link.

When was it developed-

Ricardo, (1990) highlights that this model is known as the 'basis of the oldest database management systems' which was discovered after attempts to organise data for the US space programme. The hierarchical database was developed by IBM (a large computer company) in 1960 which also was the introduction of Database Management Systems known as DBMS. The potential of this creation was founded by IBM who noticed that management information systems (MIS) were becoming very popular, this was a massive change in the terms of data storage. The opening commercial database that

came accessible was in 1964 and was called the Integrated Data Store (IDS). Foote (2017) explains this was developed by a man named Charles Bachman, with General Electric supporting his research. This model was seen as a 'flexible way of representing objects' and the one to one relationships between tables. IBM continued to use and discover this model for their Information Management System (IMS). Until the other models developed which proved to be more appropriate for sophisticated types of data management.

Main applications this model was used for-

Codex.cs.(n.d) states that the hierarchical model was used as IBMs by the 'North American Aviation for the Apollo moon-landing program'. They created their own database system in order to keep data on the moon landing – however there was a massive problem with data redundancy, and therefore GUAM (Generalised Update Access Method) was created. Coronel et al (2013) explains this was 'based on the recognition that the many smaller parts would come together as components of still larger components' until they finally came together in the final unit. In order for this to work it maintained the hierarchy structure. To create a hierarchical DBMS applications such as Microsoft Access, Excel and SQL server.

Is the model still used today-

This type of database system can still be used today, whether individuals know they are using this model or not they have stored their information in one file (the parent or root) and this file is then comprised or linked to other files. hierarchical models tend to be most useful when the primary focus is assembling information on a concrete hierarchy an example of this is different departments in a business which has already got a stable hierarchical framework for example – one department is higher than the other and incorporates the correct relationships and from each department is the employees and their details.

In this model there are commands which help easy access to the data for example, the 'get' command locates a recording the database which is particularly useful if there is a lot of records on DBMS, unlike with the flat file database it required the user to read through the whole database in order to find a record. There is also a insert or delete command which allows one single record to be deleted or a new record to be entered, this is also used for updating a record.

This model provides a lot of security to the data in the system, with the use of passwords to be able to view some records. Those who are at the root of the model have more control and power to the database than those at the very bottom, an example is a shop the owner will be at the root and then the manager and then employees so the employees will not be able to view the

data that the managers can. In other words, the level of data each person on the system will be limited.

Advantages-

The most important advantage with the hierarchical model is all the data in the system is relevant to the topic/object, all tables must have a link together in the parent-child manner. Therefore this allows the end user to determine whether the data is relevant because if no links could be made then it would not be necessary. DBMS which have a lot of data to be entered tend to use this model as it is more capable to deal with large amounts of data and therefore uses one to many relationships. The hierarchical model has been tried and tested by many businesses before and there are many applications and outlines available for new businesses to base their system on.

Disadvantages-

Although this model is easy to use and simple in that the structure is explained like as a tree it proves difficulty in the features it provides, with only being able to produce one to many relationships whereas models discussed later can do much more. This model therefore is not versatile which means it is unable to adapt to different functions or activities. Coronel et al (2013) states that there are not many real world situations which adhere to the one to many relationship.

Recommendations-

This model is very difficult to navigate therefore it is time consuming to find certain parts of data which are not already queried. The performance of this model on a system can vary with how much data is on it, the more data the slower the processing of data.

Network model

what it is-

The network model is known to have similar features as the model just discussed, where the system is referred to as a collection of records in one-to-many relationships. The structure of this model is also somewhat like the hierarchical model where there is a structure which consists of branches or nodes. However, this model allows more than one parent, whereas the model before only allowed a child to have one parent, Yannakoudakis et al (1988) explains that prevents any redundancy introduced by the hierarchic model by ‘defining multiple incoming pointers to each record occurrence’. These nodes represent records and the links represent relationships within the database and these become pointers. A relationship in a network model is also referred to as a ‘set’, each set is then made up of at least two record types. An example of this by Rob et. al (2013) is an owner record (equivalent

to parent record in hierarchical model) and a member record (equivalent to the child). Below is a diagram which shows how this works.

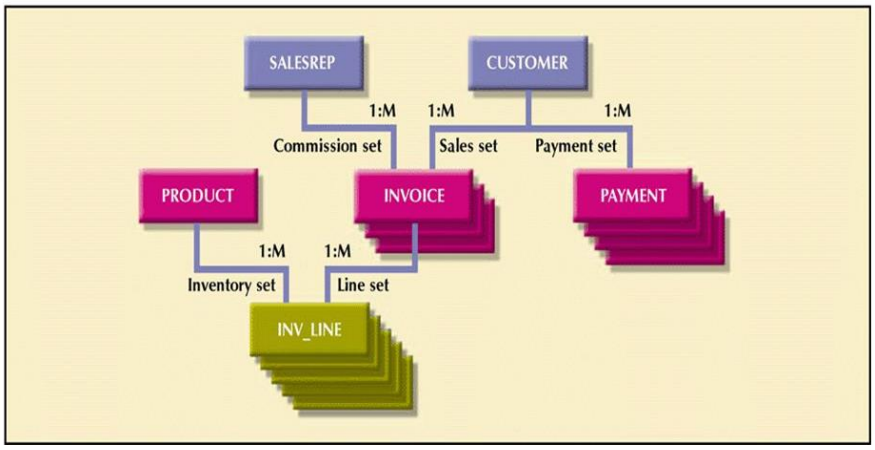


Figure 4 network database model

(Smartsheet.org, n.d.)

Notice how the product, invoice and payment records are all members of both salesrep and customer. Sometimes a network model cannot support the level of many-to-many relationships and therefore needs broke down to two separate one-to-one relationships. This model looks like a web structure due to the links which cross over each other. The unlimited number of links with this model allows it to be more flexible than the others.

When was it developed-

This model was originally developed by a man named Charles Bachman, this model was developed into a standard specification in 1969 by the CODASYL which is the Conference on Data Systems Languages. In 1971 came the second publication of this model which became the foundation of most implementations, this was seen as an improvement of the already existing hierarchical database model. Bachman had seen many faults and flaws in the hierarchical model and decided to create a model which was somewhat similar however which was more flexible and had less defaults. The data within this model is easily accessed as there are more links established and one piece of information will bring up another.

Is it still used today-

This model uses SQL, (EDUCBA, n.d.) states that using tis language allows ‘manipulation of data that can be used to gain valuable insights and learnings’. In today’s world this model is less used and the reason for this is because searching for data within the model is ‘cumbersome because the system has to traverse the entire data set’.

Applications used for the model-

This model can be used on a range of applications, just like the other models Microsoft Access, Microsoft Server SQL, FileMaker, InterSystems Cache and Navicat.

Advantages-

The main advantage this model has over the others are that more data can be incorporated into the system whilst having a link to the other sets. It is most popular with those systems which include more complex data as the relationships help the user to visually see what is being presented. The flexibility this model offers allows navigation to be easier and therefore the search of information.

Disadvantages-

This particular model requires some training and knowledge about the structure of the data in order for it to be efficient, it tends to be difficult to know which tables to link together and certain bits of data entered can change the whole database. If the user is using many to many relationships in this model this can be very tricky to achieve and splitting it into two one-to-one relationships which causes confusion as you're working with double the relationships.

Recommendations-

Due to the difficulty of implementing this database and the technical structure it would not be recommended for a big organisation as working with multiple one to one relationships in order to create many to many it can be very confusing. Doing this can also cause you to loose vital data about a certain student, or miss out on an important step which will result in an unorganised DBMS.

Relational model

What it is-

The relational data is somewhat similar to the previous models already discussed, however it is more advanced in what it does whilst being simple. For a database to be relational it must hold relationships between the tables. Relational databases use flat file tables to store its information, these tables are known as relations which gives the model its name – furthermore these tables or relations are made up of tuples. Ricardo, (1990) explains these tuples as a 'row held within the table' and within these rows are entities or attributes. An entity can be a single thing or object which data is stored about. For example entities of a person would be DOB, name and surname. The information about these entities is then stored in attributes which Ricardo (1990) discusses as characteristics or properties of the entities. Tables are used in this model as a more organised way to hold information about the content and objects to be presented in the database. The columns

within the tables are known as domain and this is where the attributes are stored.

An example of a table like this would be

TBL_Student

Student_ID	Name	Surname
1	Jordan	Walker
2	Emma	Bell

As you can see the name of the table is 'Student' whilst the entities are Student Number, Name and Surname. The attributes are then held in the rows below e.g. 1, Jordan and Walker.

Coronel et al, 2013 go on to state that in a table each primary key must be 'unique to ensure that it will uniquely each row, when this occurs the table is said to exhibit entity integrity'. To ensure this stays like this there cannot be a null value placed as the primary key, nulls can sometimes be used as attributes of the primary key however if used incorrectly can cause many problems within the database. A null value is seen to have 3 different causes, these are an 'unknown attribute value, a known but missing attribute value and a 'not applicable' condition'. Tables then share common attributes which allow us to link the tables together, in particular using the primary key.

Coming from this is the foreign key, this is somewhat like the primary key in that it provides a link between the two tables. Techopedia.com,(n.d.) explains that the foreign key 'acts like a cross-reference between tables because it references the primary key of another table, thereby establishing a link between them'. Just like discussed in the hierarchical model, there can be links made between tables to create relationships in the database.

When was it developed-

As talked about before the development of databases was done by the company IBM, however a particular English computer scientist developed this particular model, he saw a slightly different way to handle data using databases. E.F.Codd in 1970 seen this particular model as a way forward in the evolution of technology for both the users and designers because of its simplicity feature. (Coronel et. al 2013) further defines the reason of this model being described as simple was because the information is held explicitly which makes it easily understood by those users with little or no knowledge. Codd first approached the topic of this database model in his research paper 'A Relational Model Of Data For Large Shared Data Banks', here he discussed what makes a relational database and the definition of it. For a database to be of the type relational it is deemed that it must include all of Codd's 12 rules any emergence of commercial systems were based on

the relational model. In 1974 and 1977 there were two relational database prototype systems which were created at UBC and System R at IBM San Jose.

Main applications this model used for –

Nolan et al., (2014)states that Codd stated in his research paper how the relational model 'solved some of the shortcomings of the current models and discussed some areas where a relational model needed advanced', this particular paper was seen as the first introduction to the relational model. Meanwhile the problems Codd had already highlighted the model has improved and evolved into the popular model it is today. The rules set out by Codd allow representation of real-world database systems. Databases were known to be a solution to address problems of decentralized file stores.

Is the model still used today-

Relational databases today tend to use SQL which is a domain specific language used relational, the standard application and programming interface for relational is SQL. Rockoff, (2011) book explains that 'SQL is the most widely used software tool for communicating with data residing in relational databases'. According to the ANSI (American National Standards Institute) it is the typical language for this model of database, it produces statements which allows the user to choose from for example, update data on a database. Systems that use SQL include Access, Ingres and Sybase, most RDBMS have more commands added into their system, the most common demands used within the SQL language is 'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop' which allows the user to do everything they need on the database. This language enables reports to be produced from the database and interactive queries.

The relational model is a very important one today, with the features already discussed, and redundancy of data tends to not occur here.

Disadvantages-

The most common disadvantage of the relational database is the cost of setting up the DBMS, as you tend to need special software to ensure the running of the management system goes smoothly and effectively. Usually for large organisations like a university or school the tutors may need to be taught how the system works, this training is also costly as is maintaining the system.

Advantages-

This model is unlike the others as it is structured where the data can be easily found and is categorized to allow users to find the data they are looking for quickly. This model allows queries and reports to be made on certain data, which makes it easier to view data on one particular student. The data is sorted into types which minimises human input error as letters

couldn't be inputted into the data field. SQL which is used for this model can be used on many applications which gives users a choice of software to use this model on, this doesn't restrain the user to purchase one certain interface. The RDBMS is much quicker in sorting and searching through the data both on web applications and desk.

Recommendations-

This model provides the highest security throughout the 5 models, this is particularly essential for organisations like a university or school to ensure the Data Protection Act 1998 is not breached. It allows sensitive tables or information on students to be hidden and students cannot view the same information that tutors can. Furthermore, tutors can't view the same level of data that the head of departments can. Ricardo, C. (1990) states that unlike the other models produced the relational model achieves the structural independence as well as data independence whereas the other models don't portray structural independence.

Object oriented model

What is it

The object oriented database model also known as OODBMS (Object-Oriented Database Management System) is simply a DBMS where the data is represented in the collection of objects. It is hard to exactly pin point what exactly is the OO model, Rob et all (2013) explains the reason of this as 'there is no standard OO data model'. However there is a list of characteristics which a model must contain to be deemed as OO.

These include,

- 'Support the representation of complex objects
- Must be extensible, by defining new data types as well as operations
- Must support encapsulation; the data and method's implementation must be hidden external entities
- Must display inheritance
- Must support object identity'

The way data is structured in this particular model is through the objects which represent real word entities, and these have attributes and methods. The attributes can reference more than one object and with the encapsulation this is hidden. The use of classes are used in this model where similar objects are grouped together, these are then organised in the style of hierarchy. The class is known to describe the category of the object.

When it was developed

This model's name 'object-oriented database system' first came about in 1985 by Orion Research Project at MCC by Won Kim who first developed the database model. The first database was released in 1991 by a man named

Rick Cattell who was part of the ODMG (Object Data Management Group) where they set specifications that would enable portable applications that stores objects in a DBMS. In order to use this database model the Object Query Language is used as opposed to the SQL kin the other models, this language was developed by the group however due to its complexity this language has never been used as much as the SQL.

Main applications for the model-

This is the 4th database model which was designed to meet new features and in order for this database to be used on newer applications such as CAD (Computer Aided Design), CASE (Computer-aided software engineering) and GIS (Geographical information systems).

Advantages-

Objected oriented databases are known to model real life examples, as the objects are organised into classes which helps the database to be easier maintained.

Disadvantages –

However this model is based on objects rather than the data entered into the DBMS and the processing of the information. This is because the object oriented model is seen to be not a technology as it is seen to be a problem solving approach to unstructured data instead of a way to view and lay out data. This model is much more difficult to understand compared to the relational model due to the size of the system.

Recommendations

This tends not to be used as much in today because of the language required to process the system, this approach is usually more used for website designs in order to put everything together in its classes. It is much harder to create an OODBMS compared to a relational system where the computer can run the system more quickly and therefore is more effective when looking for data quickly.

Reference list

Coronel, C., Morris, S. and Rob, P. (2013). *Database systems*. Boston, Mass.: Course Technology/Cengage Learning.

Nolan, G., Truxall, D., Sodd, R. and Cinar, O. (2014). *Android Best Practices*. 1st ed. Apress.

Ricardo, C. (1990). *Database systems*. New York: Macmillan.

Rockoff, L. (2011). *The language of SQL*. Boston, Mass.: Course Technology/Cengage Learning.

Techopedia.com. (n.d.). *What is a Foreign Key? - Definition from Techopedia*. [online] Available at: https://www.techopedia.com/definition/7272/foreign-key [Accessed 23 Oct. 2018].

EDUCBA. (n.d.). *4 Important Roles of Database Management System in Industry*. [online] Available at: https://www.educba.com/database-management-system/ [Accessed 24 Oct. 2018].

Rob, P., Coronel, C. and Crockett, K. (2013). *Database systems*. 8th ed. Andover: Cengage Learning.

Smartsheet. (n.d.). *All about Relational Database Models | Smartsheet*. [online] Available at: https://www.smartsheet.com/relational-database-modeling [Accessed 20 Oct. 2018].

Yannakoudakis, E. and Cheng, C. (1988). *Standard Relational and Network Database Languages*. London: Springer London.

Codex.cs.yale.edu. (n.d.). *Hierarchical Model, Appendix B*. [online] Available at: http://codex.cs.yale.edu/avi/db-book/db4/b.pdf [Accessed 15 Oct. 2018].

Coronel, C., Morris, S. and Rob, P. (2013). *Database systems*. Boston, Mass.: Course Technology/Cengage Learning, p.20.

DBMS Internals . . . (n.d.). *Hierarchical Data Model - DBMS Internals . . .*. [online] Available at: http://www.dbmsinternals.com/database-fundamentals/basic-architecture/hierarchical-data-model/ [Accessed 15 Oct. 2018].

Fmhelp.filemaker.com. (n.d.). *FileMaker Pro 17 Advanced Help*. [online] Available at: https://fmhelp.filemaker.com/help/17/fmp/en/index.html#page/FMP_Help/one-to-one-relationships.html [Accessed 15 Oct. 2018].

Foote, K. (2017). *A Brief History of Data Modeling - DATAVERSITY*. [online] DATAVERSITY. Available at: http://www.dataversity.net/brief-history-data-modeling/ [Accessed 16 Oct. 2018].

Powell, G. (2006). *Beginning Database Design*. 1st ed. John Wiley & Sons, p.8.

Ricardo, C. (1990). *Database systems*. New York: Macmillan, p.138.

Burleson, D. (1999). *Inside the database object model*. Boca Raton, Flor.: CRC Press, p.30.

Chickowski, E. (2010). *Flat-File Databases Often Overlooked In Security Schemes*. [online] Dark Reading. Available at: https://www.darkreading.com/risk/flat-file-databases-often-overlooked-in-security-schemes/d/d-id/1133363 [Accessed 9 Oct. 2018].

Driscoll, K. (2012). From Punched Cards to "Big Data": A Social History of Database Populism. *Futures of Communication, University of Southern California*, 1.

Fair, E. (2017). *How did punch cards work? What did they look like?*. [online] Quroa. Available at: https://www.quora.com/How-did-punch-cards-work-What-did-they-look-like [Accessed 8 Oct. 2018].

Sumathi, S. and Esakkirajan, S. (2010). *Fundamentals of relational database management systems*. Berlin: Springer, pp.228, 296.

Techopedia.com. (n.d.). *What is a Flat File Database? - Definition from Techopedia*. [online] Available at: https://www.techopedia.com/definition/7231/flat-file-database-database [Accessed 14 Oct. 2018].

Techterms.com. (n.d.). *Flat File Definition*. [online] Available at: https://techterms.com/definition/flatfile [Accessed 13 Oct. 2018].

Gorman, M. (2015). *Database Management Systems*. Saint Louis: Elsevier Science, p.1.