EDO UNIVERSITY IYAMHO, NIGERIA.
DEPARTMENT OF COMPUTER SCIENCE

COURSE CODE: CSC 222
DATABASE DESIGN AND MANAGEMENT SYTEMS (3 Units)

Instructor:

Uddin, Osemengbe O.

- Office: Faculty of Science OA9
- Office Hours: Monday/Tuesday 11pm - 2pm
- Email: uddin.osemengbe@edouniversity.edu.ng

Course Logistics

Lectures

- Tuesday: 8am – 10am

Continuous Assessment (CA)

- 30% of the grade

Exams

- 70% of the grade

NOTE: No make-up or substitute continuous assessment (CA) or exams

COURSE CODE: CSC 222
DATABASE DESIGN AND MANAGEMENT SYSTEMS

Introduction to DBMS (SQL, MySQl) Transaction management, concurrency control, Object Oriented Databases, client/server systems, data warehouse, databases in electronic commerce, web database development and database administration. Single table database Relational Database System of related tables Minimum redundancy, Referential integrity, Database keys, The ACID model (guarantee of successful transactions): –Atomicity („all or nothing" rule), Consistency Enforcing referential integrity, Indexing field values, Principles for building a database, Queries, SQL – Structured Query Language. Logical operators, Normalization, and Decomposition. Database design and management with SQL Server

DATABASE MANAGEMENT SYSTEM (DBMS)

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data.

A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible.

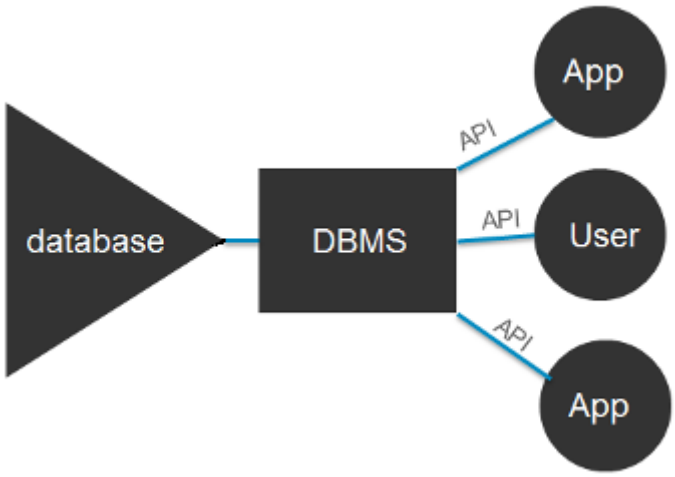
The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified -- and the database schema, which defines the database's logical structure. These three foundational elements help provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery. Many database management systems are also responsible for automated rollbacks, restarts and recovery as well as the logging and auditing of activity.

The DBMS is perhaps most useful for providing a centralized view of data that can be accessed by multiple users, from multiple locations, in a controlled manner. A DBMS can limit what data the end user sees, as well as how that end user can view the data, providing many views of a single database schema. End users and software programs are free from having to understand where the data is physically located or on what type of storage media it resides because the DBMS handles all requests.

The DBMS can offer both logical and physical data independence. That means it can protect users and applications from needing to know where data is stored or having to be concerned about changes to the physical structure of data (storage and hardware). As long as programs use the application programming interface (API) for the database that is provided by the

DBMS, developers won't have to modify programs just because changes have been made to the database.

With relational DBMSs (RDBMSs), this API is SQL, a standard programming language for defining, protecting and accessing data in a RDBMS.



Popular Types of DBMS

Popular database models and their management systems include:

Relational Database Management System (RDMS) - adaptable to most use cases, but RDBMS Tier-1 product**s can be quite expensive.**

NoSQL DBMS - well-suited for loosely defined data structures that may evolve over time.

In-memory Database Management System (IMDBMS) - provides faster response times and better performance.

Columnar Database Management System (CDBMS) - well-suited for data warehouses that have a large number of similar data items.

Cloud-based Data Management System - the cloud service provider is responsible for providing and maintaining the DBMS.

Advantages of a DBMS

Using a DBMS to store and manage data comes with advantages, but also overhead. One of the biggest advantages of using a DBMS is that it lets end users and application programmers access and use the same data while managing data integrity. Data is better protected and maintained when it can be shared using a DBMS instead of creating new iterations of the same data stored in new files for every new application. The DBMS provides a central store of data that can be accessed by multiple users in a controlled manner.

Central storage and management of data within the DBMS provides:

- Data abstraction and independence
- Data security
- A locking mechanism for concurrent access
- An efficient handler to balance the needs of multiple applications using the same data
- The ability to swiftly recover from crashes and errors, including restartability and recoverability
- Robust data integrity capabilities
- Logging and auditing of activity
- Simple access using a standard application programming interface (API)
- Uniform administration procedures for data

Another advantage of a DBMS is that it can be used to impose a logical, structured organization on the data. A DBMS delivers economy of scale for processing large amounts of data because it is optimized for such operations.

A DBMS can also provide many views of a single database schema. A view defines what data the user sees and how that user sees the data. The DBMS provides a level of abstraction between the conceptual schema that defines the logical structure of the database and the physical schema that describes the files, indexes and other physical mechanisms used by the database. When a DBMS is used, systems can be modified much more easily when business requirements change. New categories of data can be added to the database without disrupting the existing system and applications can be insulated from how data is structured and stored.

Of course, a DBMS must perform additional work to provide these advantages, thereby bringing with it the overhead. A DBMS will use more memory and CPU than a simple file storage system. And, of course, different types of DBMSes will require different types and levels of system resources.

DBMS - Architecture

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed, or replaced.

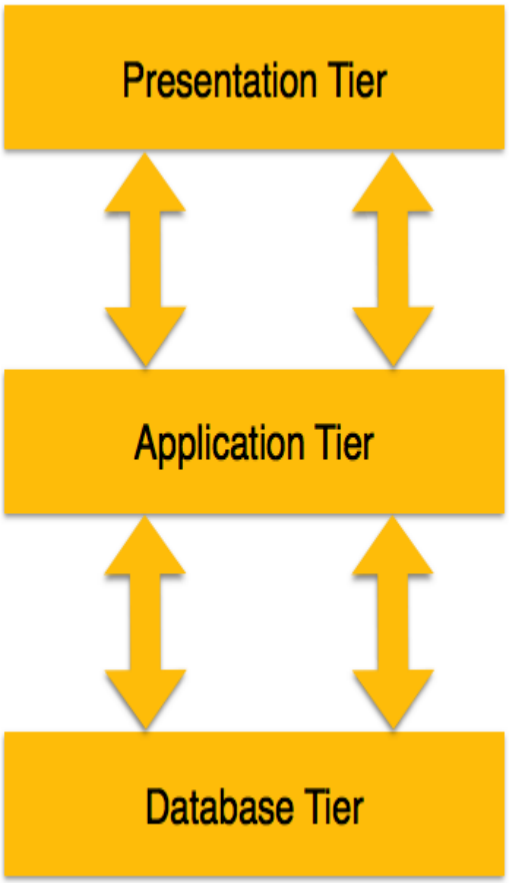
In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the

DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

3-tier Architecture

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.



User (Presentation) Tier – End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

Application (Middle) Tier – At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

Database (Data) Tier – At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

NOTE:

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

Database Model

A database model shows the logical structure of a database, including the relationships and constraints that determine how data can be stored and accessed. Individual database models are designed based on the rules and concepts of whichever broader data model the designers adopt. Most data models can be represented by an accompanying database diagram.

Types of Database Models

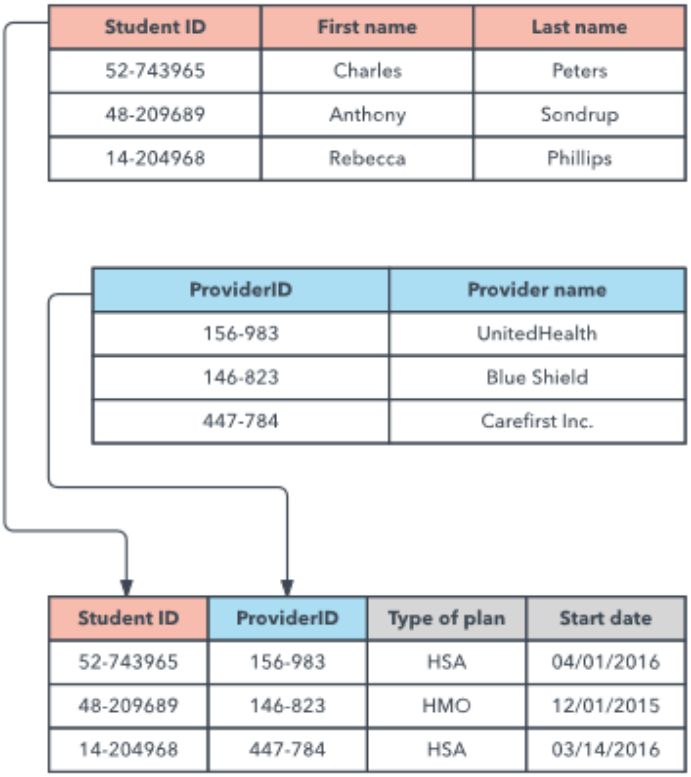
There are many kinds of data models. Some of the most common ones include:

- Hierarchical database model
- Relational model
- Network model
- Object-oriented database model
- Entity-relationship model
- Document model
- Entity-attribute-value model
- Object-relational model

Relational Model

This is most common model. The relational model sorts data into tables, also known as relations, each of which consists of columns and rows. Each column lists an attribute of the entity in question, such as price, zip code, or birth date. Together, the attributes in a relation are called a domain. A particular attribute or combination of attributes is chosen as a primary key that can be referred to in other tables, when it's called a foreign key. Each row, also called a tuple, includes data about a specific instance of the entity in question, such as a particular employee.

The model also accounts for the types of relationships between those tables, including one-to-one, one-to-many, and many-to-many relationships. Here's an example:

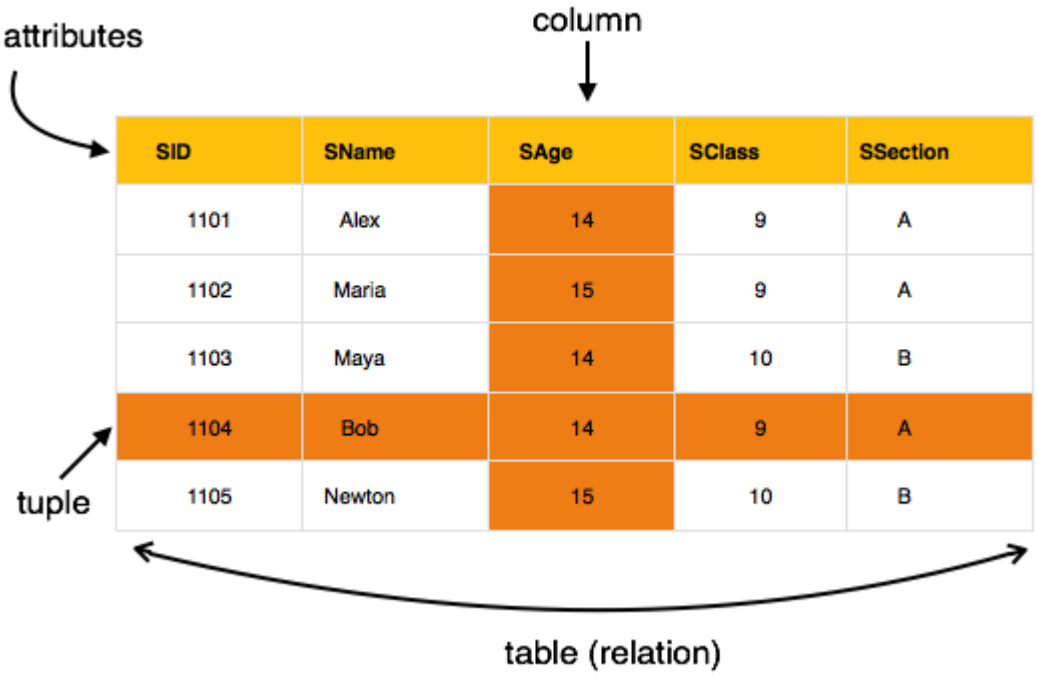


Student ID	First name	Last name
52-743965	Charles	Peters
48-209689	Anthony	Sondrup
14-204968	Rebecca	Phillips

ProviderID	Provider name
156-983	UnitedHealth
146-823	Blue Shield
447-784	Carefirst Inc.

Student ID	ProviderID	Type of plan	Start date
52-743965	156-983	HSA	04/01/2016
48-209689	146-823	HMO	12/01/2015
14-204968	447-784	HSA	03/14/2016

Another example is:



SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

The main highlights of this model are −

1. Data is stored in tables called relations.
2. Relations can be normalized.
3. In normalized relations, values saved are atomic values.

4. Each row in a relation contains a unique value.
5. Each column in a relation contains values from a same domain.

Relational databases are typically written in Structured Query Language (SQL). The model was introduced by E.F. Codd in 1970.

Codd's 12 Rules

Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.

These rules can be applied on any database system that manages stored data using only its relational capabilities. This is a foundation rule, which acts as a base for all the other rules.

Rule 1: Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3: Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as data dictionary, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6: View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule

A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence
The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9: Logical Data Independence
The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

Rule 10: Integrity Independence
A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: Distribution Independence
The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

Rule 12: Non-Subversion Rule
If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

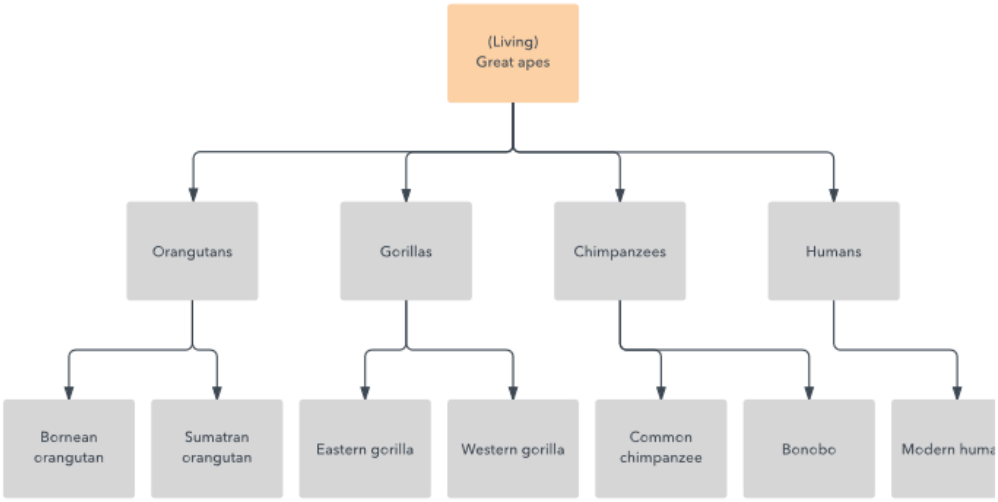
Assignment

a. Write a detailed note on Database Languages based on
    1. Relational Algebra
    2. Relational Calculus

b. List and explain the different database keys (Primary key, Super key, Candidate key, Alternate key, Composite key, Foreign key) and write on Database Integrity Rules

c. Write a detailed note on the different normalisation forms (Introduction, 1NF, 2NF, 3NF and Boyce–Codd Normal Form(BCNF))

Hierarchical Model
The hierarchical model organizes data into a tree-like structure, where each record has a single parent or root. Sibling records are sorted in a particular order. That order is used as the physical order for storing the database. This model is good for describing many real-world relationships.



This model was primarily used by IBM's Information Management Systems in the 60s and 70s, but they are rarely seen today due to certain operational inefficiencies.

Network Model
The network model builds on the hierarchical model by allowing many-to-many relationships between linked records, implying multiple parent records. Based on mathematical set theory, the model is constructed with sets of related records. Each set consists of one owner or parent record and one or more member or child records. A record can be a member or child in multiple sets, allowing this model to convey complex relationships. It was most popular in the 70s after it was formally defined by the Conference on Data Systems Languages (CODASYL).

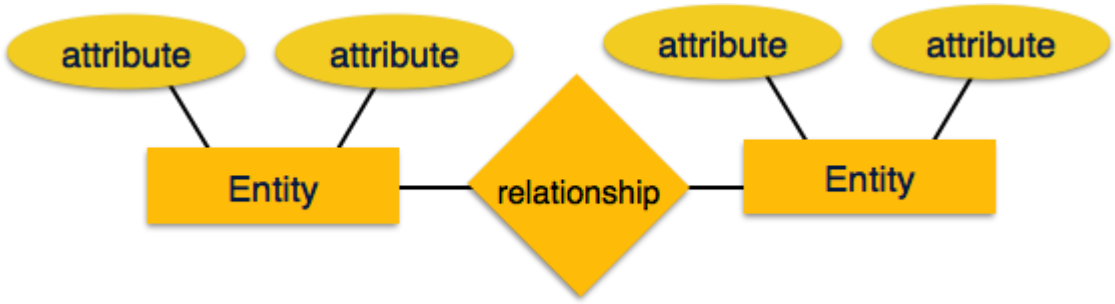
Object-Oriented Database Model
This model defines a database as a collection of objects, or reusable software elements, with associated features and methods. There are several kinds of object-oriented databases:
A **multimedia database** incorporates media, such as images, that could not be stored in a relational database.
A **hypertext database** allows any object to link to any other object. It's useful for organizing lots of disparate data, but it's not ideal for numerical analysis.
The object-oriented database model is the best known post-relational database model, since it incorporates tables, but isn't limited to tables. Such models are also known as hybrid database models.

Object-Relational Model
This model combines the two (Object and Relational) that make up its name. You may choose to describe a database with any one of these depending on several factors. The

biggest factor is whether the database management system you are using supports a particular model. Most database management systems are built with a particular data model in mind and require their users to adopt that model, although some do support multiple models.
One can also say that this hybrid database model combines the simplicity of the relational model with some of the advanced functionality of the object-oriented database model. In essence, it allows designers to incorporate objects into the familiar table structure. Languages and call interfaces include SQL3, vendor languages, ODBC, JDBC, and proprietary call interfaces that are extensions of the languages and interfaces used by the relational model.

Entity-Relationship Model
This model captures the relationships between real-world entities much like the network model, but it isn't as directly tied to the physical structure of the database. Instead, it's often used for designing a database conceptually. Here, the people, places, and things about which data points are stored are referred to as entities, each of which has certain attributes that together make up their domain. The cardinality, or relationships between entities, are mapped as well.

These concepts are explained below



Entity − An entity in an ER Model is a real-world entity having properties called attributes. Every attribute is defined by its set of values called domain. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.

Relationship − The logical association among entities is called relationship. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.

Mapping Cardinalities −

- one to one
- one to many
- many to one
- many to many

Other Database Models
A variety of other database models have been or are still used today.

Inverted file model
A database built with the inverted file structure is designed to facilitate fast full text searches. In this model, data content is indexed as a series of keys in a lookup table, with the values pointing to the location of the associated files. This structure can provide nearly instantaneous reporting in big data and analytics, for instance.
This model has been used by the ADABAS database management system of Software AG since 1970, and it is still supported today.

Flat model
The flat model is the earliest, simplest data model. It simply lists all the data in a single table, consisting of columns and rows. In order to access or manipulate the data, the computer has to read the entire flat file into memory, which makes this model inefficient for all but the smallest data sets.

Multidimensional model
This is a variation of the relational model designed to facilitate improved analytical processing. While the relational model is optimized for online transaction processing (OLTP), this model is designed for online analytical processing (OLAP).
Each cell in a dimensional database contains data about the dimensions tracked by the database. Visually, it's like a collection of cubes, rather than two-dimensional tables.

Semi-structured model
In this model, the structural data usually contained in the database schema is embedded with the data itself. Here the distinction between data and schema is vague at best. This model is useful for describing systems, such as certain Web-based data sources, which we treat as databases but cannot constrain with a schema. It's also useful for describing interactions between databases that don't adhere to the same schema.

Context model
This model can incorporate elements from other database models as needed. It cobbles together elements from object-oriented, semi-structured, and network models.

Associative model
This model divides all the data points based on whether they describe an entity or an association. In this model, an entity is anything that exists independently, whereas an association is something that only exists in relation to something else.
The associative model structures the data into two sets:

- A set of items, each with a unique identifier, a name, and a type
- A set of links, each with a unique identifier and the unique identifiers of a source, verb, and target. The stored fact has to do with the source, and each of the three identifiers may refer either to a link or an item.

Other, less common database models include:

- Semantic model, which includes information about how the stored data relates to the real world
- XML database, which allows data to be specified and even stored in XML format
- Named graph
- Triplestore

NoSQL database models
In addition to the object database model, other non-SQL models have emerged in contrast to the relational model:

The **graph database model**, which is even more flexible than a network model, allowing any node to connect with any other.

The **multivalue model**, which breaks from the relational model by allowing attributes to contain a list of data rather than a single data point.

The **document model**, which is designed for storing and managing documents or semi-structured data, rather than atomic data.

Databases on the Web
Most websites rely on some kind of database to organize and present data to users. Whenever someone uses the search functions on these sites, their search terms are converted into queries for a database server to process. Typically, middleware connects the web server with the database.
The broad presence of databases allows them to be used in almost any field, from online shopping to micro-targeting a voter segment as part of a political campaign. Various industries have developed their own norms for database design, from air transport to vehicle manufacturing.

Further Reading

Course Lecture Notes: http://www.edouniversity.edu.ng/oer/compsc/csc222.pdf