ECSE/ECIE/ECCE 2206 DATABASE PROGRAMMING (ORACLE DB) NOTES

INTRODUCTION DATABASES

PROGRAMMING LANGUAGE GENERATIONS

First Generation Language

1GL or first-generation language was (and still is) *machine language* or the level of instructions and data that the processor is actually given to work on (which in conventional computers is a string of 0s and 1s).

Second Generation Language

2GL or second-generation language is assembler (sometimes called "assembly") language. A typical 2GL instruction looks like this:

ADD 12,8

An assembler converts the assembler language statements into machine language.

Third Generation Language

3GL or third-generation language is a "high-level" programming language, such as PL/I, C, or Java. Java language statements look like this:

```
    public boolean handleEvent (Event evt) {
switch (evt.id)  {
case Event.ACTION_EVENT:  {
if ("Try me" .equald(evt.arg)) {
```

A compiler converts the statements of a specific high-level programming language into machine language. (In the case of Java, the output is called bytecode, which is converted into appropriate machine language by a Java virtual machine that runs as part of an operating system platform.) A 3GL language requires a considerable amount of programming knowledge.

Fourth Generation Language

4GL or fourth-generation language is designed to be closer to natural language than a 3GL language. Languages for accessing databases are often described as 4GLs. A 4GL language statement might look like this:

EXTRACT ALL CUSTOMERS WHERE "PREVIOUS PURCHASES" TOTAL MORE THAN $1000

Fifth Generation Language

5GL or fifth-generation language is programming that uses a visual or graphical development interface

to create source language that is usually compiled with a 3GL or 4GL language compiler. Microsoft, Borland, IBM, and other companies make 5GL visual programming products for developing applications in Java, for example. Visual programming allows you to easily envision object-oriented programming class hierarchies and drag icons to assemble program components.

Introduction to 4 GLs Programming

Fourth-generation languages attempt to make communicating with computers as much like the processes of thinking and talking to other people as possible. The problem is that the computer still only understands zeros and ones, so a compiler and interpreter must still convert the source code into the machine code that the computer can understand. Fourth-generation languages typically consist of English-like words and phrases. When they are implemented on microcomputers, some of these languages include graphic devices such as icons and onscreen push buttons for use during programming and when running the resulting application.

Many fourth-generation languages use Structured Query Language (SQL) as the basis for operations. SQL was developed at IBM to develop information stored in relational databases. Eventually, it was adopted by the American National Standards Institute (ANSI) and later by the International Standards Organization (ISO) as a means of managing structured, factual data. Many database companies offer an SQL-type database because purchasers of such databases seek to optimize their investments by buying open databases, i.e., those offering the greatest compatibility with other systems. This means that the information systems are relatively independent of vendor, operating system, and computer platform.

Examples of 4GL

Examples of fourth-generation languages include PROLOG, an **artificial intelligence** language that applies rules to data to arrive at solutions; and OCCAM and PARLOG, both parallel-processing languages. Newer languages may combine SQL and other high-level languages. IBM's Sonnet is being modified to use sound rather than visual images as a computer interface.

What is a database (DB)?

i. Is a collection of data that exists over a long period of time, often many years'
ii. managed through a database management system

What is a database management system (DBMS)?

1. or simply `database system'
2. `a powerful tool for creating and managing [and manipulating] large amounts of data [(several gigabytes; 10^9 bytes)] *efficiently* and allowing it to *persist* over long periods of time, *safely*
3. focus on secondary, rather than main, memory
4. powerful, but simple, programming interface

Why do we need database systems?

1. data management problem
2. think of analogies from your personal information space

DBMS vs. `just a file system'

i. DBMS's evolved from file systems
ii. file systems also store large amounts of data over a long period of time in secondary memory

iii. however, file systems
    i. can lack efficient access
    ii. have no direct support for queries
    iii. limit organization to directory creation and hierarchical organization
    iv. have no sophisticated support for concurrency
    v. do not ensure durability

ACID Properties in Databases

A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain **A**tomicity, **C**onsistency, **I**solation, and **D**urability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

- **Atomicity** − This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
    - should not be able to execute half of an operation
    - either all or none of the effects of a transaction are made permanent
- **Consistency** − The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
    - there should be no surprises in the world, e.g., gpa > 4.0, balance < 0, cats should never have more than 1 tail!
    - the effect of concurrent transactions is equivalent to some serial execution
    - use constraints, triggers, active DB elements (context-free)
- **Durability** − The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.
    - if power goes out, nothing bad should happen
    - once accepted, the effects of a transaction are permanent (until, of course, changed by another transaction)
    - use logs
- **Isolation** − In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.
    - concurrency control

- transactions should not be able to observe the partial effects of other transactions
- use locks (whole relations or individual tuples?)

Database uses

Databases for Businesses

iv. The business world depends on databases 24 hours a day/seven days a week. Inventory, order processing, payroll, accounting, shipping and transportation routing are often tracked within a main database that keeps the company functioning.

Databases for Educational Institutions

i. From elementary schools to colleges, educational institutions have used databases to keep track of students, grades, transfers, transcripts and other student data. There are even specialized database packages geared toward schools and colleges.

Databases for Non-Profit Organizations

1. Like businesses and educational institutions, non-profit organizations must have a system to keep track of information. Many charities and other non-profit groups use a database for keeping track of donations, volunteers, hours served in the community, clients helped and other information related to the organization.

Databases for Household and Family Management

1. The database also has a home in household and family management for many individuals and families. Many individuals/families use a database to keep track of family birthdays, bills and expenses within a home; addresses of friends and relatives; movie/DVD collections; and other lists.

Databases In Use Everyday

1. Each time you make a purchase and the sales clerk asks for a your address or IDNO, your information is kept and stored in a customer database. These collections of data are used to send mailings of special offers, discounts and other deals.

When a database holds details about people, it's likely to include their first name, surname and their date of birth. In addition to this, specialist *information* is stored depending on the database's intended use.

- The police have details of all known criminals in a database, eg crimes they've comitted.
- Schools use a database to store details about their pupils, eg how many days they've been off school sick.
- A hospital will store details of all its patients in a database, eg a history of their health issues.
- The Government uses a database to store records of people's income tax payments.
- A database is used to keep track of all the drivers in central London who have (or haven't) paid the Congestion Charge.

DATABASE

Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information.

Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks.

A **database management system** stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

Characteristics

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics −

- **Real-world entity** − A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.

- **Relation-based tables** − DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.

- **Isolation of data and application** − A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.

- **Less redundancy** − DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.

- **Consistency** − Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.

- **Query Language** − DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.

- **ACID Properties** − DBMS follows the concepts of **A**tomicity, **C**onsistency, **I**solation, and **D**urability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.

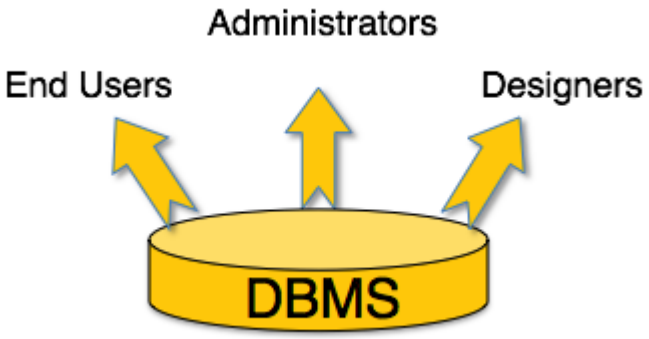
- **Multiuser and Concurrent Access** − DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.

- **Multiple views** − DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.

- **Security** − Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

Users

A typical DBMS has users with different rights and permissions who use it for different purposes. Some users retrieve data and some back it up. The users of a DBMS can be broadly categorized as follows −



- **Administrators** − Administrators maintain the DBMS and are responsible for administrating the database. They are responsible to look after its usage and by whom it should be used. They create access profiles for users and apply limitations to maintain isolation and force security. Administrators also look after DBMS resources like system license, required tools, and other software and hardware related maintenance.

- **Designers** − Designers are the group of people who actually work on the designing part of the database. They keep a close watch on what data should be kept and in what format. They identify and design the whole set of entities, relations, constraints, and views.

- **End Users** − End users are those who actually reap the benefits of having a DBMS. End users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.

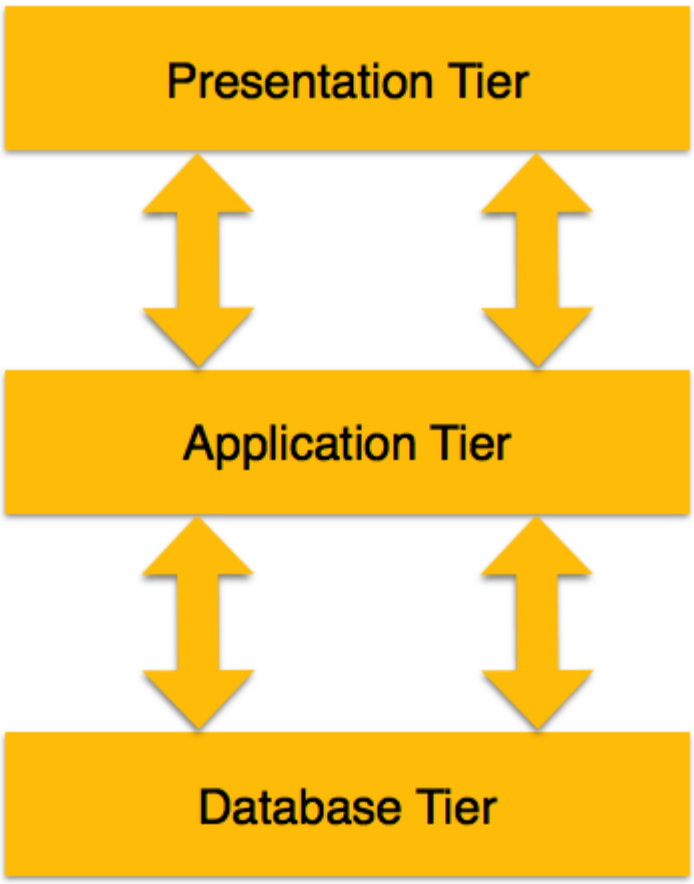
The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent **n** modules, which can be independently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



- **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.

- **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.

- **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.

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

Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

The very first data model could be flat data-models, where all the data used are to be kept in the same plane. Earlier data models were not so scientific, hence they were prone to introduce lots of duplication and update anomalies.

Database - Advantages & Disadvantages

Advantages

ii. Reduced data redundancy
iii. Reduced updating errors and increased consistency
iv. Greater data integrity and independence from applications programs
v. Improved data access to users through use of host and query languages
vi. Improved data security
vii. Reduced data entry, storage, and retrieval costs
viii. Facilitated development of new applications program

Disadvantages

iii. Database systems are complex, difficult, and time-consuming to design
iv. Substantial hardware and software start-up costs
v. Damage to database affects virtually all applications programs
vi. Extensive conversion costs in moving form a file-based system to a database system
vii. Initial training required for all programmers and users

DATABASE DESIGN

Database design is the organisation of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, she can begin to fit the data to the database model.

Database design involves classifying data and identifying interrelationships. This theoretical representation of the data is called an *ontology*. The ontology is the theory behind the database's design.

1.Determining data to be stored

In a majority of cases, a person who is doing the design of a database is a person with expertise in the area of database design, rather than expertise in the domain from which the data to be stored is drawn e.g. financial information, biological information etc. Therefore, the data to be stored in the database must be determined in cooperation with a person who does have expertise in that domain, and who is aware of what data must be stored within the system.

This process is one which is generally considered part of requirements analysis, and requires skill on the part of the database designer to elicit the needed information from those with the domain knowledge. This is because those with the necessary domain knowledge frequently cannot express clearly what their system requirements for the database are as they are unaccustomed to thinking in terms of the discrete data elements which must be stored. Data to be stored can be determined by Requirement Specification.

2.Determining data relationships

Once a database designer is aware of the data which is to be stored within the database, they must then determine where dependency is within the data. Sometimes when data is changed you can be changing other data that is not visible. For example, in a list of names and addresses, assuming a situation where multiple people can have the same address, but one person cannot have more than one address, the address is dependent upon the name. When provided a name and the list the address can be uniquely determined; however, the inverse does not hold - when given an address and the list, a name cannot be uniquely determined because multiple people can reside at an address. Because an address is determined by a name, an address is considered dependent on a name.

(NOTE: A common misconception is that the relational model is so called because of the stating of relationships between data elements therein. This is not true. The relational model is so named because it is based upon the mathematical structures known as relations.)

3. Logically structuring data

Once the relationships and dependencies amongst the various pieces of information have been determined, it is possible to arrange the data into a logical structure which can then be mapped into the storage objects supported by the database management system. In the case of relational databases the storage objects are tables which store data in rows and columns. In an Object database the storage objects correspond directly to the objects used by the Object-oriented programming language used to write the applications that will manage and access the data. The relationships may be defined as attributes of the object classes involved or as methods that operate on the object classes.

The way this mapping is generally performed is such that each set of related data which depends upon a single object, whether real or abstract, is placed in a table. Relationships between these dependent objects is then stored as links between the various objects.

Each table may represent an implementation of either a logical object or a relationship joining one or more instances of one or more logical objects. Relationships between tables may then be stored as links connecting child tables with parents. Since complex logical relationships are themselves tables they will probably have links to more than one parent.

4. ER diagram (entity-relationship model)

Database designs also include ER (entity-relationship model) diagrams. An ER diagram is a diagram that helps to design databases in an efficient way.

Attributes in ER diagrams are usually modeled as an oval with the name of the attribute, linked to the entity or relationship that contains the attribute.

Entity-Relationship Model

Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model

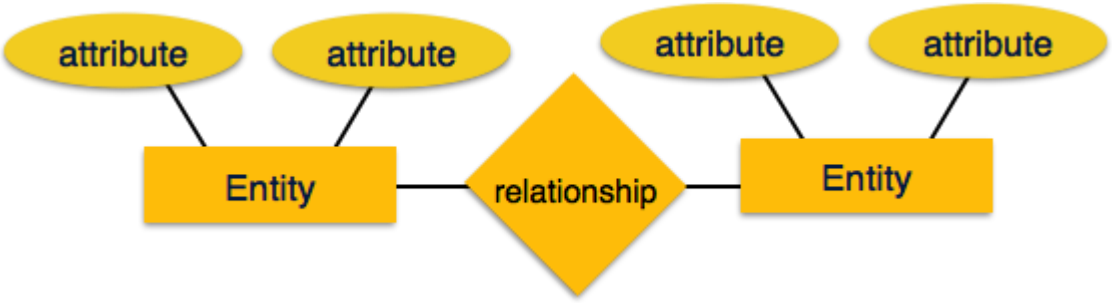
creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of a database.

ER Model is based on −

- **Entities** and their *attributes.*
- **Relationships** among entities.

These concepts are explained below.



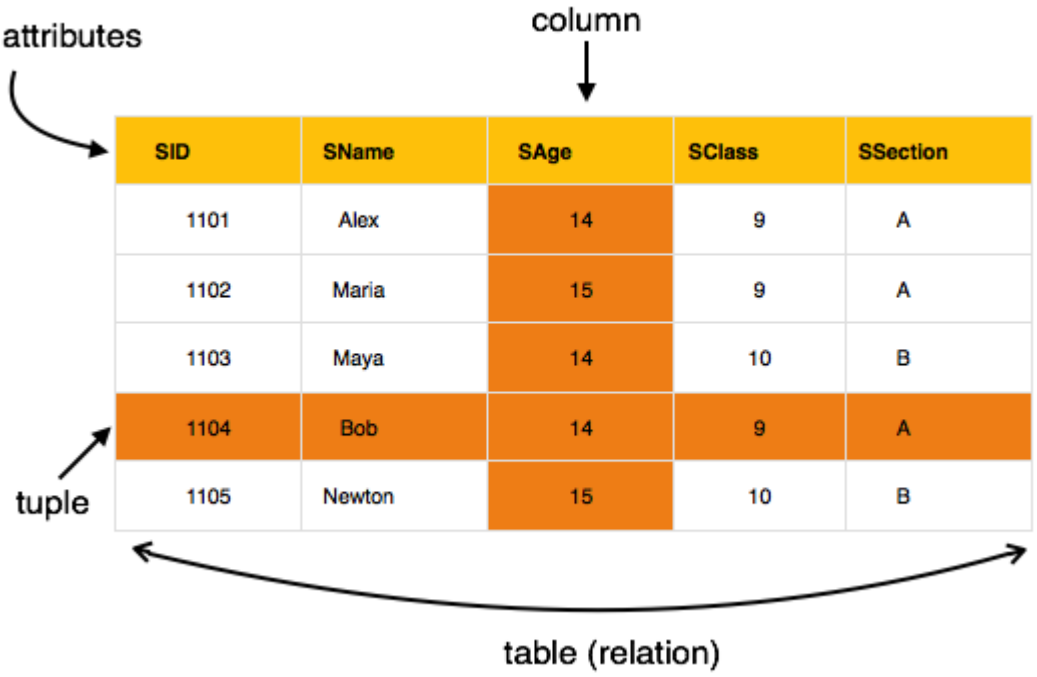
- **Entity** − An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class,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

Relational Model

The most popular data model in DBMS is the Relational Model. It is more scientific a model than others. This model is based on first-order predicate logic and defines a table as an **n-ary relation**.



The main highlights of this model are −

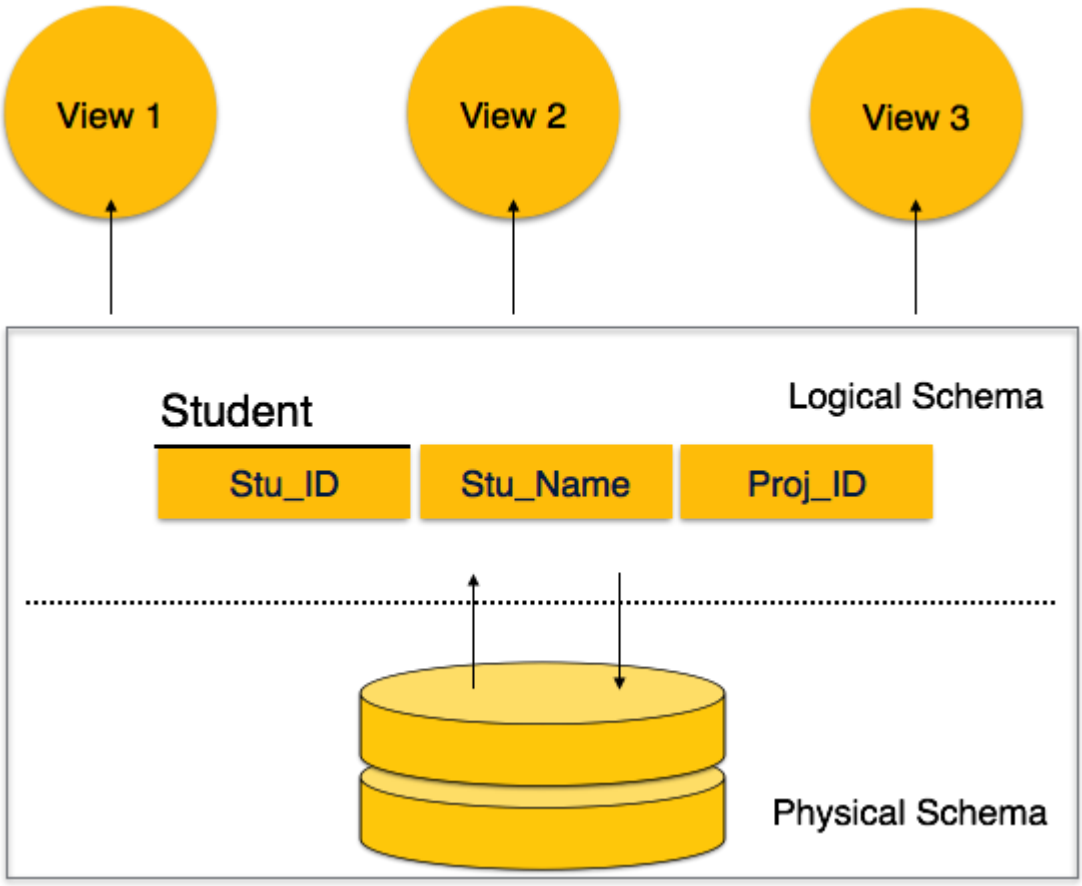
- Data is stored in tables called **relations**.
- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from a same domain.

Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

database and make it useful.



A database schema can be divided broadly into two categories −

- **Physical Database Schema** − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- **Logical Database Schema** − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Database Instance

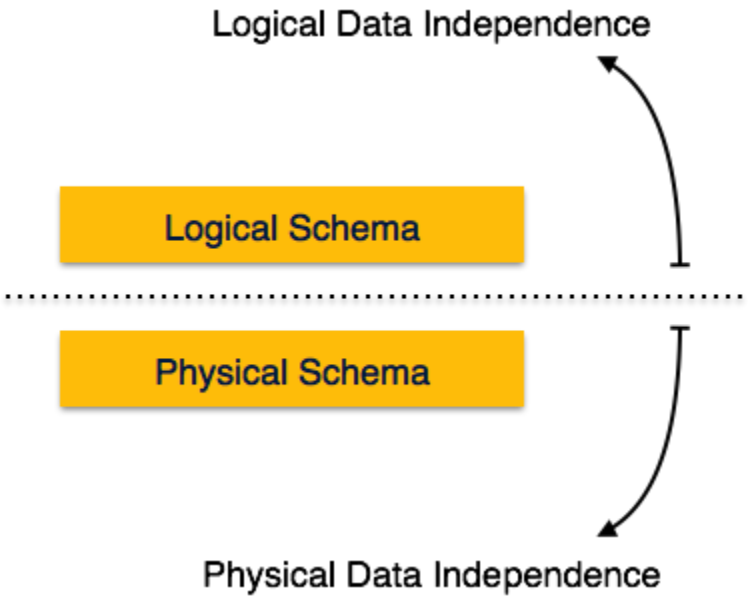
It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

If a database system is not multi-layered, then it becomes difficult to make any changes in the database system. Database systems are designed in multi-layers as we learnt earlier.

Data Independence

A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.



Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

Physical Data Independence

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

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

Constraints

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called **Relational Integrity Constraints**. There are three main integrity constraints −

- Key constraints
- Domain constraints
- Referential integrity constraints

Key Constraints

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called **key** for that relation. If there are more than one such minimal subsets, these are called ***candidate keys***.

Key constraints force that −

- in a relation with a key attribute, no two tuples can have identical values for key attributes.
- a key attribute can not have NULL values.

Key constraints are also referred to as Entity Constraints.

Domain Constraints

Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

Referential integrity Constraints

Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

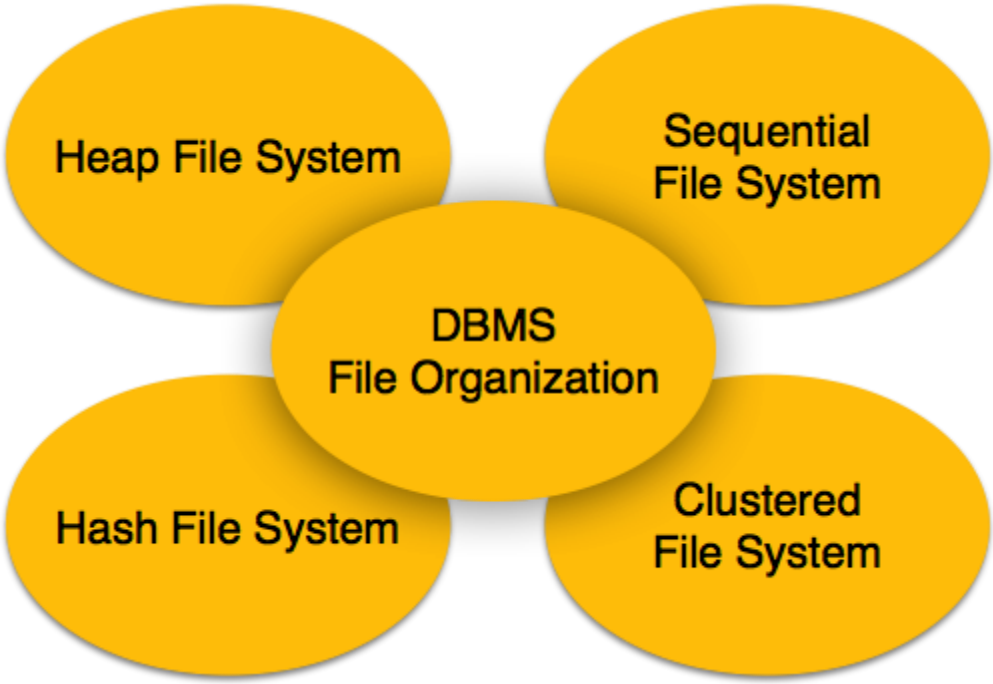
DBMS - File Structure

elative data and information is stored collectively in file formats. A file is a sequence of records stored in binary format. A disk drive is formatted into several blocks that can store records. File records are mapped onto those disk blocks.

File Organization

File Organization defines how file records are mapped onto disk blocks. We have four types of File Organization to organize file records –

Types of File Organization. File organization is a way of organizing the data or records in a file. It does not refer to how files are organized in folders, but how the contents of a file are added and accessed. There are several types of file organization, the most common of them are sequential, relative and indexed.



Heap File Organization

When a file is created using Heap File Organization, the Operating System allocates memory area to that file without any further accounting details. File records can be placed anywhere in that memory area. It is the responsibility of the software to manage the records. Heap File does not support any ordering, sequencing, or indexing on its own.

Sequential File Organization

Every file record contains a data field (attribute) to uniquely identify that record. In sequential file organization, records are placed in the file in some sequential order based on the unique key field or search key. Practically, it is not possible to store all the records sequentially in physical form.

Hash File Organization

Hash File Organization uses Hash function computation on some fields of the records. The output of the hash function determines the location of disk block where the records are to be placed.

Clustered File Organization

Clustered file organization is not considered good for large databases. In this mechanism, related records from one or more relations are kept in the same disk block, that is, the ordering of records is not based on primary key or search key.

File Operations

Operations on database files can be broadly classified into two categories −

- **Update Operations**

- **Retrieval Operations**

Update operations change the data values by insertion, deletion, or update. Retrieval operations, on the other hand, do not alter the data but retrieve them after optional conditional filtering. In both types of operations, selection plays a significant role. Other than creation and deletion of a file, there could be several operations, which can be done on files.

- **Open** − A file can be opened in one of the two modes, **read mode** or **write mode**. In read mode, the operating system does not allow anyone to alter data. In other words, data is read only. Files opened in read mode can be shared among several entities. Write mode allows data modification. Files opened in write mode can be read but cannot be shared.

- **Locate** − Every file has a file pointer, which tells the current position where the data is to be read or written. This pointer can be adjusted accordingly. Using find (seek) operation, it can be moved forward or backward.

- **Read** − By default, when files are opened in read mode, the file pointer points to the beginning of the file. There are options where the user can tell the operating system where to locate the file pointer at the time of opening a file. The very next data to the file pointer is read.

- **Write** − User can select to open a file in write mode, which enables them to edit its contents. It can be deletion, insertion, or modification. The file pointer can be located at the time of opening or can be dynamically changed if the operating system allows to do so.

- **Close** − This is the most important operation from the operating system's point of view. When a request to close a file is generated, the operating system

    - removes all the locks (if in shared mode),
    - saves the data (if altered) to the secondary storage media, and
    - releases all the buffers and file handlers associated with the file.

The organization of data inside a file plays a major role here. The process to locate the file pointer to a desired record inside a file various based on whether the records are arranged sequentially or clustered.

Other types of file organizations

- sequential,
- random,
- serial and
- indexed-sequential

1. **Sequential file organization**

- Records are stored and accessed in a particular order sorted using a key field.

- Retrieval requires searching sequentially through the entire file record by record to the end.
- Because the record in a file are sorted in a particular order, better file searching methods like the binary search technique can be used to reduce the time used for searching a file .
- Since the records are sorted, it is possible to know in which half of the file a particular record being searched is located, Hence this method repeatedly divides the set of records in the file into two halves and searches only the half on which the records is found.
- For example, of the file has records with key fields 20, 30, 40, 50, 60 and the computer is searching for a record with key field 50, it starts at 40 upwards in its search, ignoring the first half of the set.

Advantages of sequential file organization

- The sorting makes it easy to access records.
- The binary chop technique can be used to reduce record search time by as much as half the time taken.

Disadvantages of sequential file organization

- The sorting does not remove the need to access other records as the search looks for particular records.
- Sequential records cannot support modern technologies that require fast access to stored records.
- The requirement that all records be of the same size is sometimes difficult to enforce.

1. **Random or direct file organization**

- Records are stored randomly but accessed directly.
- To access a file stored randomly, a record key is used to determine where a record is stored on the storage media.
- **Magnetic** and **optical** disks allow data to be stored and accessed randomly.

Advantages of random file access

- Quick retrieval of records.
- The records can be of different sizes.

1. **Serial file organization**

- Records in a file are stored and accessed one after another.
- The records are not stored in any way on the storage medium this type of organization is mainly used on **magnetic tapes**.

Advantages of serial file organization

- It is simple
- It is cheap

Disadvantages of serial file organization

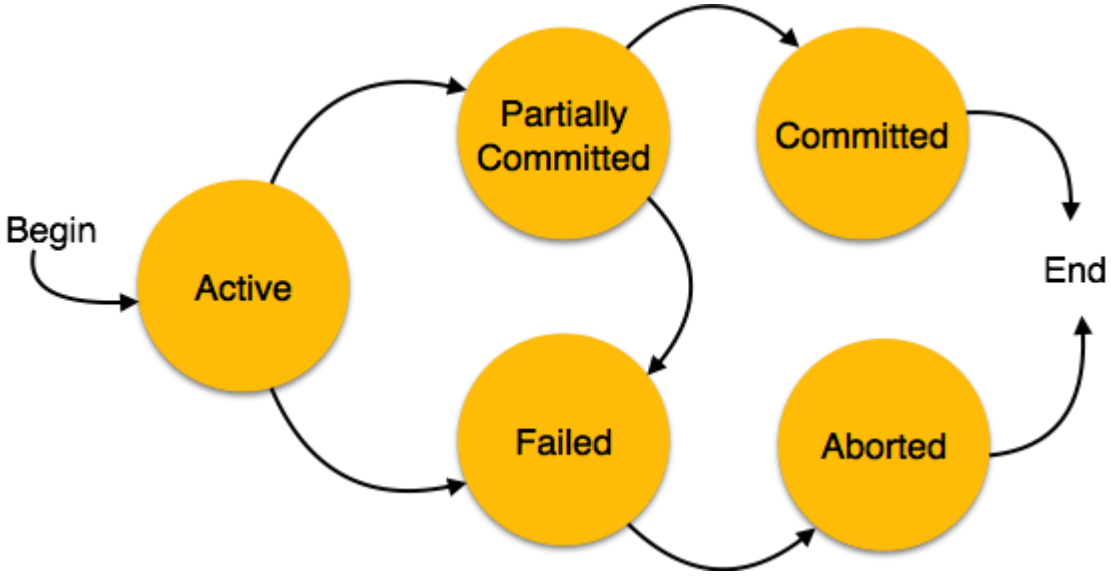
- It is cumbersome to access because you have to access all proceeding records before retrieving the one being searched.
- Wastage of space on medium in form of inter-record gap.
- It cannot support modern high speed requirements for quick record access.

1. **Indexed-sequential file organization method**

- Almost similar to sequential method only that, an index is used to enable the computer to locate individual records on the storage media. For example, on a **magnetic drum**, records are stored sequential on the tracks. However, each record is assigned an index that can be used to access it directly.

States of Transactions

A transaction in a database can be in one of the following states −



- **Active** − In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** − When a transaction executes its final operation, it is said to be in a partially committed state.
- **Failed** − A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- **Aborted** − If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts −
    - Re-start the transaction
    - Kill the transaction

- **Committed** − If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

DBMS - Concurrency Control

In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions. We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions. Concurrency control protocols can be broadly divided into two categories −

- Lock based protocols
- Time stamp based protocols

Lock-based Protocols

Database systems equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it. Locks are of two kinds −

- **Binary Locks** − A lock on a data item can be in two states; it is either locked or unlocked.
- **Shared/exclusive** − This type of locking mechanism differentiates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. Read locks are shared because no data value is being changed.

INTRODUCTION TO ORACLE DATABASE SQL COMMANDS

STRUCTURED QUERY LANGUAGE (SQL)

SQL can communicate with Oracle Database Server. It has the following advantages

- Effective
- Easy to learn and use
- Functionality complete (with SQL you can define, retrieve and manipulate data in tables)

SQL statements

1. Data Manipulation Language (DML)
    - Select
    - Insert
    - Update
    - Delete
    - Merge

2. Data Definition Language (DDL)
    - Create
    - Alter
    - Drop
    - Rename
    - Truncate
    - Comment

3. Data Control Language (DCL)
    - Grant
    - Revoke

4. Transaction Control Language (TCL)
    - Commit
    - Rollback
    - Savepoint

Working with Oracle Database

Oracle Database provides an organized mechanism for storing, managing, and retrieving information.Tables are the basic storage structure for holding business data.

Creating Tables

You create tables with the SQL CREATE TABLE statement. With Oracle Database , you have two options for creating tables.

- Use the graphical interface that generates the SQL statement
- Enter the CREATE TABLE statement in the SQL Workshop tool

When creating tables, you must provide:

- Table name
- Column name(s)
- Data types for each column

Guidelines for creating tables:

- Table and column naming rules

  - Must start with a letter, which is followed by a sequence of letters, numbers,_,#,0r $

  - Must be 1 to 30 characters long

  - Must not be an oracle server reserved word

- **Oracle data types**

  VARCHAR2(n): Variable length charter string up to n characters

  CHAR(n): Fixed length charter string of n characters

  NUMBER(n): Integer number of up to n digits

  NUMBER(precision, scale): Fixed-point decimal number. “precision” is the total number

  of digits; “scale” is the number of digits to the right of the decimal point. The decimal

  point is not counted.

  NUMBER: Floating-point decimal number

  DATE: DD-MON-YY (or YYYY) HH:MM:SS A.M. (or P.M.) form date-time.

  LONG: Variable-length character string up to 2 GB. Only one long-type column in a

  table. (1 byte ASCII char)

  NCHAR: LONG for international character sets (2-byte per character)

  CLOB: Single-byte ASCII character data up to 4 GB

  NCLOB: 2-byte CLOB

BLOB: Binary data (e.g., program, image, or sound) of up to 4 GB

BFILE: Reference to a binary file that is external to the database (OS file)

RAW(size) or LONG_RAW: raw binary data

ROWID: Unique row address in hexadecimal format

You can also set up constraints on your columns to control the data in them.

CREATING DATABASES BY USE OF SQL

5. Creating a database user by logging in a system user

SQL > connect system/tuk123; (or the password)

SQL > create user student2020
identified by tuk123;

SQL > grant create session to student2020

SQL > grant dba to student2020;

6. Connect as user student as transact

SQL > connect student2020/tuk123;

```
SQL > Create table employees
      (
      employee_id          number(7)  not null,
      first_name           varchar2(20),
      last_name            varchar2(20),
      cellphone            varchar2(12),
      email                varchar2(20),
      hire_date            date,
      job_id               varchar2(5),
      salary               number(12,2),
      manager_id           number(6),
      department_id        number(4)
      );
```

```
SQL > Create table jobs_grade
      (
      jobs_id                varchar2(5) not null,
      job_title              varchar2(20),
      min_salary              number(8),
      max_salary               number(8)
      );


SQL >  Create table department
      (
      department_id            number(4)  not null,
      department_name          varchar2(20),
      manager_id                number(6),
      location_id               number(4)
      );
```

Adding data into table employees

```
SQL >  insert into employees
values(1000,'Simon', 'Otieno','0722456789','otieno@yahoo.com','01-jan-90','5500',32000,
5000,10);

SQL >  insert into employees
values(1001,'Alice', 'Mwangi','0720766659','alice@yahoo.com','02-feb-80','5600',42000, 5000,
10);
```

Adding data into jobs_grade

```
SQL > insert into jobs_grade
values('5500',' Accountant',10000,100000);

SQL >  insert into  jobs_grade
values('5600',' Database specialist',30000,150000);
```

Adding data into department

```
SQL > insert into department
values(10,'Finance',5000,22);

SQL > insert into department
values(20,'Human Resources',5100,41);
```

DESCRIBE
List the table structure

examples
desc employees;
desc jobs_grade;
desc department;

WRITING SQL STATEMENTS

i. SQL statements are not case sensitive
ii. SQL can be entered on many lines
iii. Keywords cannot be split across lines
iv. Clauses are usually placed on separate lines for readability and ease of editing
v. Indents make it more readable
vi. Keywords may be entered in caps and all others in lowercase

SELECT – retrieving data using SQL select statement

This is used to view, query or report on data from tables. A select statement retrieves information from the database. With select statement you can use the following capabilities

3. *Projection- choose columns/fields from a table through a query.*
4. *Selection- choose rows in a table*
5. *Joining – bring together data that is stored in different tables by specifying the link between them.*

examples
select * from employees;
select * from jobs_grade;
select * from department;
select employee_id,first_name,last_name, email , job_id, salary from employees;
select jobs_id,job_title ,min_salary, max_salary from jobs_grade;
select department_id , department_name ,manager_id ,location_id from department;

Defining a Column/field as ALIAS (renaming fields with the select statement)

select employee_id "Employee ID",first_name "First Name" ,last_name "Last Name", email "Email" , job_id "Job ID",salary " Monthly Pay " from employees;

ARITHMETIC EXPRESSIONS

Operator	Description
+	Add
-	Subtract
*	Multiply

/ **Divide**

examples;

Select employee_id, first_name,last_name, salary, salary + 3000 from employees;

Select employee_id, first_name,last_name, salary, 12 salary + 700 from employees;*

Select employee_id, first_name,last_name, salary, 12 (salary + 700) from employees;*

Select employee_id, first_name,last_name, salary, 12 (salary - 2000) from employees;*

NB: Use of BODMAS applies in arithmetic expressions

DUPLICATE ROWS /RECORDS;

example

select distinct department_id from employees;

RESTRICTING AND SORTING DATA WITH SELECT STATEMENT

use of the where clause;

Comparison conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
Between … And	Between two values
IN (set)	Match any of the list
Like	Match a character pattern
IS Null	is a null value

examples

select employee_id "Employee ID",first_name "First Name" ,last_name "Last Name", email "Email" , job_id "Job ID",salary " Monthly Pay " from employees
where employee_id=1000;

select employee_id "Employee ID",first_name "First Name" ,last_name "Last Name", email "Email" , job_id "Job ID",salary " Monthly Pay " from employees
where salary > 10000;

select employee_id "Employee ID",first_name "First Name" ,last_name "Last Name", email "Email" , job_id "Job ID",salary " Monthly Pay " from employees
where salary in (10000,20000,30000);

select last_name, salary

from employees
where salary <= 10000
select last_name, salary
from employees
where salary between 3000 and 15000;

select employee_id,last_name, salary, manager_id
from employees
where manager_id in (100,101,201);

select employee_id, first_name,last_name
from employees
where first_name like 'S%';

select employee_id,last_name, salary, manager_id
from employees
where manager_id is null;

Logical Conditions

Operator	Meaning
AND	**Returns true is both are true**
OR	**Returns true if one is true**
NOT	**Return true if condition is false**

Examples
select employee_id,last_name
from employees
where salary >= 10000 and manager_id=5000;

select employee_id,last_name
from employees
where department_id not in (90,60,30);

USING THE ORDER BY CLAUSE
select last_name,job_id, department_id
from employees
order by hire_date desc;

select last_name,job_id, department_id
from employees
order by hire_date asc;

NB : ascending or descending;

UPDATE COMMAND

This is used to update data in given tables

examples
update employees
set salary= 50000
where employee_id=1001;

update employees;
set last_name='Opiyo';
where employee_id=1000;

update employees
set department_id=70
where employee_id=1001;

DELETE COMMAND
Used to delete or remove records from tables

examples
delete from employees
where employee_id=1000;

ROLLBACK
Undo some transactions; used with data manipulation language commands

example
rollback;

COMMIT
ensures that records are permanently saved. used with data manipulation language commands
example
commit;

CREATING A COPY OF A TABLE
examples

create table employees2
*as select * from employees;*

create table employees3
*as select * from employees;*

create table jobs_grade2
*as select * from jobs_grade;*

create table jobs_grade3
*as select * from jobs_grade;*
create table department2
*as select * from department;*

create table department3
*as select * from department;*

CREATING A TABLE BU USING A SUBQUERY

create table department10
as select employee_id,last_name,first_name,salary, salary(120/100) NewSalary, hire_date*
from employee where department_id=10;

TRUNCATE COMMAND

Remove all rows but leave the table structure intact;

example
truncate employees3;

DROP TABLE COMMAND

i. All data and structure is deleted
ix. Pending transactions are not committed
x. All indexes are dropped
xi. All constraints are dropped
xii. You cannot rollback drop table

examples
drop table jobs_grade3;

drop table department3;

drop table employees3;

USING SINGLE ROW FUNCTIONS TO CUSTOMIZE OUTPUT

Character functions

v. lower
vi. upper
vii. initcap
viii. concat
ix. substr
x. length
xi. trim

e.tc

Examples
select lower(last_name), upper(last_name) from employees;

select initcap(first_name) from employees;

Number functions

ii. Round – round value to specified decimal
iii. Trunc – Truncates value to specified decimal
iv. Mod – returns remainder of division

Examples
select round(45.92356,2) from dual;

select trunc(45.92356,2) from dual;

select mod(1600,300) from dual;

Working with dates

Select last_name , hire_date from employees
where hire_date < '01-feb-88';

select sysdate from dual;

select sysdate + 7 from dual;

select systdate – 7 from dual;

REPORTING GROUPED DATA

TYPE OF GROUPED FUNCTIONS

2. avg
3. count
4. max
5. min
6. stddev
7. sum
8. variance

etc

examples
select max(salary), min(salary), sum(salary),avg(salary) from employees;

select min(hire_date), max(hire_date) from employees;

select count(*) from employees;

select count(salary) from employees where department _id in (10,20,30,40);

select count(distinct department_id) from employees;

select department_id,avg(salary) from employees group by department_id;

select avg(salary) from employees group by department_id;

select department_id,job_id,sum(salary) from employees group by department_id,job_id;

HAVING CLAUSE

select department_id,max(salary) from employees group by
department_id
having max(salary) > 10000 ;

DISPLAYING DATA FROM MULTIPLE TABLES

This is achieved using

1. *cross joins*
2. *natural joins*
3. *using clause*
4. *full outer joins*
5. *arbitrary joins*

examples

select employees.employee_id, employees.last_name,department_id from employees join departments using (department_id)

INCLUDING CONSTRAINTS

2. *Enforce rules at table level*
3. *Prevent deletion of a table if there are dependencies*
4. *Following constraints are valid*
    - *Not null*
    - *Unique*
    - *Primary key*
    - *Foreign Key*
    - *Check*

example

create table employees5
(

employee_id number(6) constraint employees5_id_pk primary key,
first_name varchar2(20)
);

A. Not null constraint ensures that the column contains no null or empty values

B. unique key constraint – requires that every value must be unique

example

create table employees31
(
employee_id number(6),
last_name varchar2(20) not null,
email varchar2(20),
salary number(10,2),
hire_date date not null,
constraint emp_email_uk unique(email)
);

C. Primary key – Constraint creates a primary key for the table. Only one primary key can be created for each table.

D. Foreign key – or referential integrity constraint designates a column or combination of columns as a foreign key and establishes a relationship between a primary key or a unique key in the same table or different table

example

Creating foreign key constraint
Create table department50
(
Department_id number(5) constraint department_id_pk primary key,
Department_name vachar2(20)
)

create table employees51
(
employee_id number(6) constraint emp_id_pk primary key,
last_name varchar2(20) not null,
first_name varchar2(20),
salary number(10,2),
department_id number(4),
constraint emp_dept_fk foreign key (department_id) references department50(department_id)
)

NB: the table department must exist with primary key on department_id

E. Check constraint
create table employees35
(
employee_is number(6),
first_name varchar2(20),
last_name varchar2(20) not null,
email varchar2(20),
salary number(10,2) constraint emp_salary_min_check check (salary >0)
);
Integrity constraint error

When you have constraints in place on columns, an error is returned if you try to violate the constraint rule.

ALTER TABLE STATEMENTS

This is used to

viii. Add a new column to a table
ix. Modify an existing column
x. Define default value for a new column
xi. Drop a column from a table

example

alter table jobs_grade
add column effective_date date;

alter table employees
add constraint emp_id_pk primary key;

alter table jobs_grade1
drop column jobs_title;

alter table employees10
drop constraint emp_dept_fk;

1.Adding a new Column
alter table employees11
  add net_pay number(12,2);

2.Rename a column
alter table employees11
  rename column tax to deduction;

3.Modify a column

```
alter table employees11
  modify tax number(10,2);
```

4 **Drop a Column from a table**

```
alter table employees11
  drop column deduction;
```

DATA OBJECTS

OBJECT	DESCRIPTION
Table	Basic unit of storage
View	Logically represents subsets of data from one or more tables
Sequence	Generate numeric values
Index	Improves the performance of some queries
Synonyms	Gives alternative names to objects

What is a view ?

A view is a logical table based on a table or another view

Advantages of a view

1. To restrict data access
2. To make complex queries easy
3. To provide data independence
4. To present different views of the same data

Creating a View

create view empvu80
*as select * from employees;*

Modifying a view

can be done by create or replace view

Removing a view

views can be removed by drop command
e.g *drop view empvu80*

Sequences

A sequence is a database object that creates integer values. You can create sequences and use them to generate numbers

create sequence sequence1
increment by 1
start with 2
maxvalue 100

nocycle;

```
SQL> select SEQUENCE_NAME,min_value,max_value,increment_by,last_number from user_sequences;
```

Indexes
Indexes are database objects that you can create to improve on the performance of some queries

create index emp_last_name_idx
on employees(last_name)

Removing an Index
drop index index_name e.g *drop emp_last_name_idx*

Synonyms
Synonyms are database objects that enables you to call a table by another name

Examples

create synonym muchiri for employees;

DATA DICTIONARY
This is a collection of tables and views in oracle database created and maintained by oracle server and contains information about a database. It is an important tool for all users from end users to application developers and database administrators.

DATA DICTIONARY STRUCTURE
1. *USER – user's views*
2. *ALL – expanded user view*
3. *DBA – database admin view*
4. *V$ - performance related data*

NB :To use commands in this category you must be logged in as DBA or System user

examples
describe dictionary;

*select * from dictionary;*

desc user_tables;

select table_name from user_tables;

desc user_constraints;

*select * from user_constraints*

where table_name='employees';