Databases and Data Models

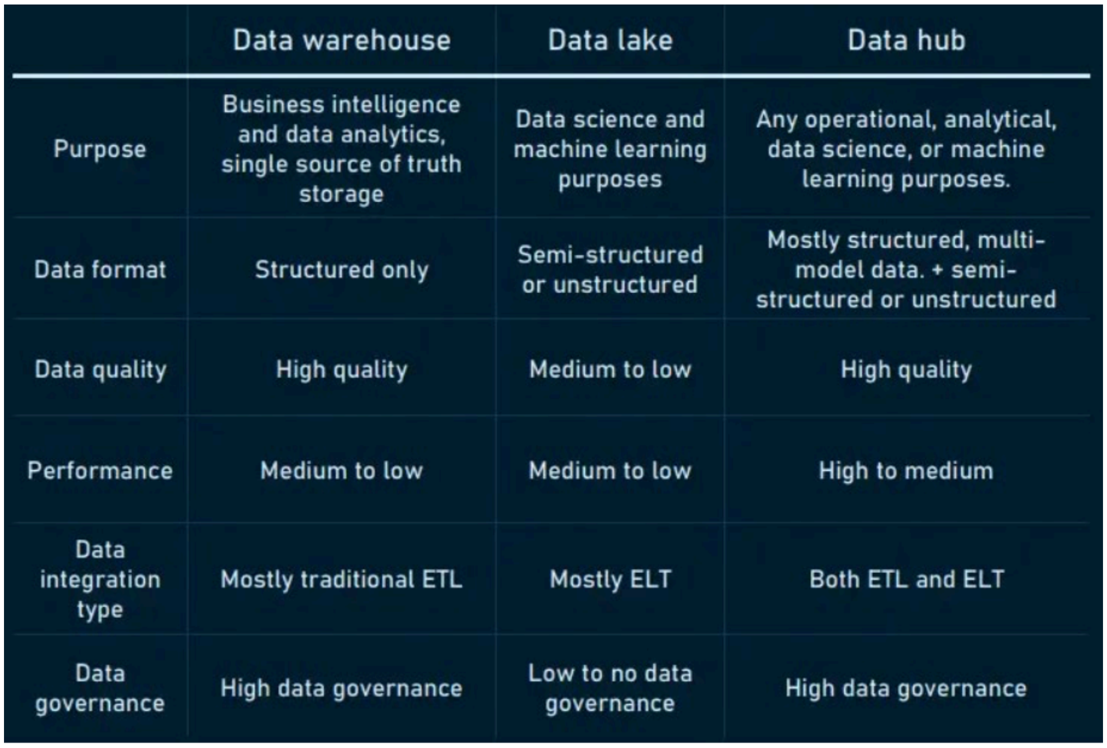
Database: A shared collection of logically related data(and a description of this data), designed to meet the information needs of an organization.

Database Management System(DBMS): A software system that enables users to define, create, maintain, and control access to the database.

Data Lake	Data Warehouse
A central location that holds a large amount of data in its native, raw format.	Integrates data and information collected from various sources into one comprehensive database. The components are used to analyze customers and support business intelligence activities.
data is unstructured, semi-structured, and structured(minimally processed)	highly structured data that is cleaned, pre-processed, and refined.
contain unconventional data such as log and sensor data	contain historic and relational data
contains vast amounts of data	contains less data and data must be processed
can be used for a wide variety of applications such as machine learning, streaming analytics, and AI	used for specific reasons such as business intelligence.

Data Hub: a centralized system for data storage, definition, and delivery.

- It is a hybrid of a data lake and a data warehouse.
- It has a central repository for applications to dump data.
- It functions as a management point, allowing organizations to see how data flows across the enterprise.



	Data warehouse	Data lake	Data hub
Purpose	Business intelligence and data analytics, single source of truth storage	Data science and machine learning purposes	Any operational, analytical, data science, or machine learning purposes.
Data format	Structured only	Semi-structured or unstructured	Mostly structured, multi-model data. + semi-structured or unstructured
Data quality	High quality	Medium to low	High quality
Performance	Medium to low	Medium to low	High to medium
Data integration type	Mostly traditional ETL	Mostly ELT	Both ETL and ELT
Data governance	High data governance	Low to no data governance	High data governance

Data Storage Options

- traditional databases
    - structured and organized, managing and analyzing data is easier
- data warehouses
    - designed to store large amounts of historical data from various sources, making it easier to analyse trends and patterns.
- data lakes
    - a single repository for processing raw data for the purpose of analytics, visualization, AI uses cases and machine learning to create value
- data hubs
    - designed to exchange or share data

Data Model

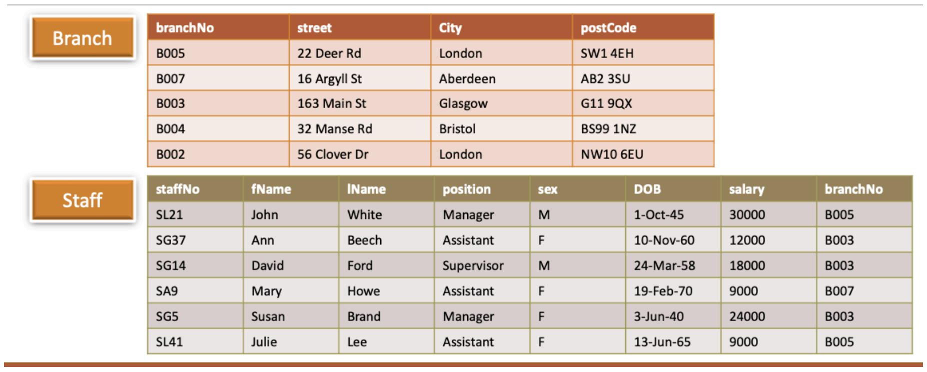
Integrated collection of concepts for describing data, relationships between data, and constraints on the data in an organization.

Types of data model:

- Physical data models
- Object-Based data models
    - entity-relationship, semantic, functional, object-oriented
- Record-Based data models
    - relational data model, network data model, hierarchical data model

Relational Data Model

- based on the concept of mathematical relations where data and relationships are represented as tables, each of which has a number of columns with a unique name.



Branch

branchNo	street	City	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

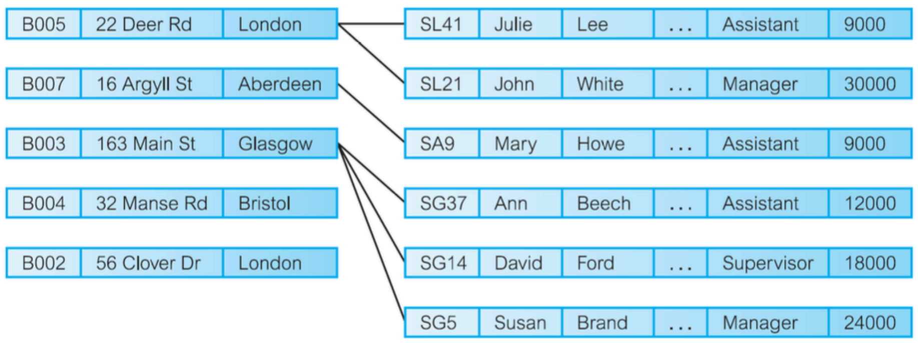
Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

-

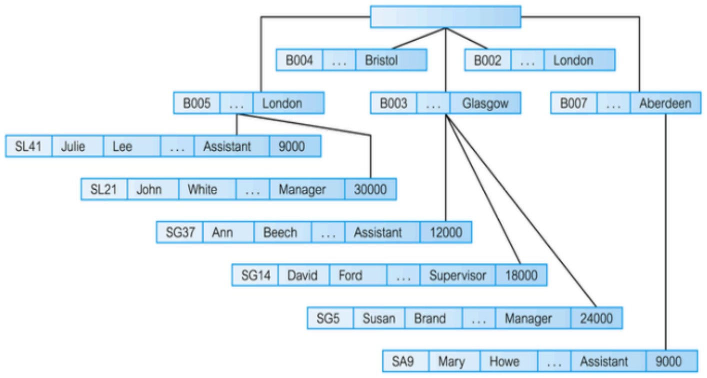
Network Data Model

- data is represented as a collection of records and relationships are represented by sets.
- the records are organized as generalized graph structures with records appearing as nodes and sets as edges


B005 22 Deer Rd London
B007 16 Argyll St Aberdeen
B003 163 Main St Glasgow
B004 32 Manse Rd Bristol
B002 56 Clover Dr London
SL41 Julie Lee ... Assistant 9000
SL21 John White ... Manager 30000
SA9 Mary Howe ... Assistant 9000
SG37 Ann Beech ... Assistant 12000
SG14 David Ford ... Supervisor 18000
SG5 Susan Brand ... Manager 24000

Hierarchical Data Model

- restricted type of network model
- data is also represented as collections of records and relationships are represented by sets.
- each node only has one parent
- it can be represented as a tree graph, records are nodes and sets are edges.



B004 ... Bristol
B002 ... London
B005 ... London
B003 ... Glasgow
B007 ... Aberdeen
SL41 Julie Lee ... Assistant 9000
SL21 John White ... Manager 30000
SG37 Ann Beech ... Assistant 12000
SG14 David Ford ... Supervisor 18000
SG5 Susan Brand ... Manager 24000
SA9 Mary Howe ... Assistant 9000

Relational Database

A collection of normalized relations with distinct relation names. Created by E.F. Codd.

Terminologies

- Relation(table/file): a two-dimensional table with columns and rows.
- Attribute(column/field): named column of a relation
- Domain: set of allowable values for one or more attributes
- Degree: the number of attributes in a relation.
- Tuple(row/record): a row of a relation.
- Cardinality: the number of tuples on a relation

Properties of relation

- relation name: distinct from other relation names in the relational schema
- each cell contains exactly one atomic value
- each attribute has a distinct name
- values of an attribute are all from the same domain
- each tuple is distinct, there are no duplicates tuples.
- order of attributes has no significance
- order of tuples has no significance.

Relational Keys

- Superkey
  - an attribute or set of attributes that uniquely identifies a tuple within a relation.
- Candidate Key (a subset of superkey)
  - superkey such that no proper subset is a superkey within the relation
  - In each tuple of relation, values of this key uniquely identify the tuple (uniqueness)
  - no proper subset of superkey has the uniqueness property (irreducibility)
- Primary Key (subset of candidate key)
  - candidate key selected to identify tuples uniquely within relation.
- Alternate Keys (subset of candidate key)
  - candidate keys that are not selected to be the primary key.
- Foreign Key
  - attribute or set of attributes within one relation that matches the candidate key of some relation.

Integrity Constraints

- Null Value
    - value for an attribute that is currently unknown or not applicable for a tuple
    - deals with incomplete or exceptional data
    - represents the absence of a value and is not the same as zero or spaces which are values
- Entity Integrity
    - in a base relation (the named relation corresponding to an entity in a conceptual schema, whose tuples are physically stored in the database), no attribute of a primary key can be null.
- Referential Integrity
    - if a foreign key exists, the foreign key value must match a candidate key value of some tuple in its home relation or the foreign key must be wholly null.
- General Constraints
    - additional rules specified by users or database administrators that define or constrain some aspect of the enterprise.

Database System Development Lifecycle

Information Systems: resources that enable the collection, management, control, and dissemination of information throughout an organization. The database is a fundamental component of it and its development should be viewed from the perspective of the wider requirements of the organization.

Database Planning: planning how the stages of the lifecycle can be realized most efficiently and effectively.

System Definition: specifying the scope and boundaries of the database system, including the major user views, its users, and application areas.

Requirements Collection and Analysis: collection and analysis of the requirements for the new database system.

Database Design: conceptual, logical, and physical design of the database.

Application Design: designing user interface and application programs that use and process the database.

Implementation: creating the physical database definitions and the application programs.

Data Conversion and Loading: loading data from the old system to the new system and where possible, converting any existing applications to run on the new database.

Testing: The database system is tested for errors and validated against the requirements specified by the users.

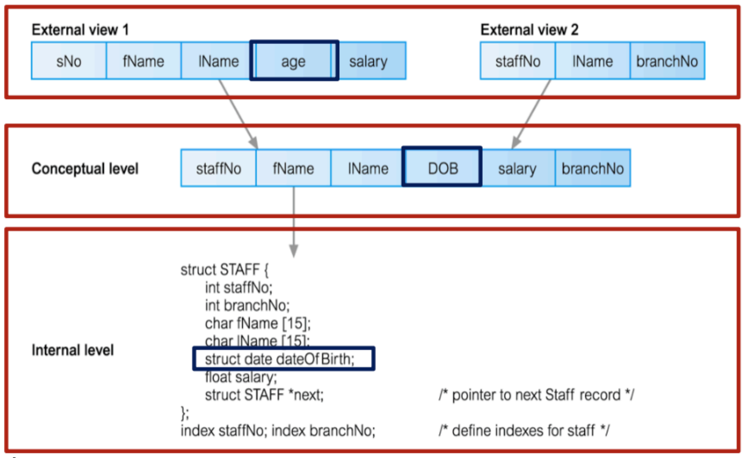
Operational Maintenance: The database system is fully implemented. The system is continuously monitored and maintained. when necessary, new requirements are incorporated into the database system through the preceding stages of the lifecycle.

Three Level Architecture

external level
- users view of the database
- describes the part of the database that is relevant to a particular user.

conceptual level
- community view of the database
- describes what data is stored in the database and relationships among the data.

internal level
- physical representation of the database on the computer
- describes how the data is stored in the database.



```
struct STAFF {
    int staffNo;
    int branchNo;
    char fName [15];
    char lName [15];
    struct date dateOfBirth;
    float salary;
    struct STAFF *next;            /* pointer to next Staff record */
};
index staffNo; index branchNo;     /* define indexes for staff */
```

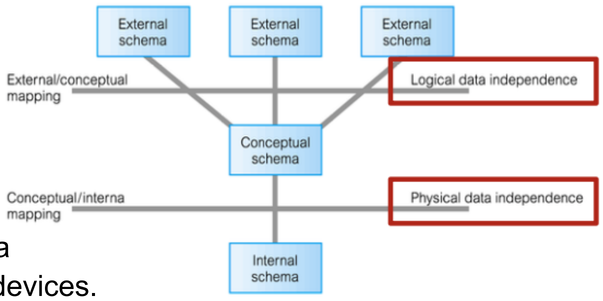
Importance of Maintaining Separation

levels should be independent of each other -> changes to one level do not affect another.

- all users should be able to access the same data
- a user view is immune to changes made in other views
- users should not need to know physical database storage details
- database administrators should be able to change database storage structures without affecting the users' views.
- internal structure should be unaffected by changes to physical aspects of storage.
- database administrators should be able to change the conceptual structure of the database without affecting all users.

Logical Data Independence

- immunity of external schemas to changes in conceptual schema such as the addition/removal of entities
- should not require changes to the external schema or rewrites of application programs,


External schema
External schema
External schema
External/conceptual mapping
Logical data independence
Conceptual schema
Conceptual/interna mapping
Physical data independence
Internal schema

Physical Data Independence

- immunity of conceptual schema to changes in the internal schema such as using different file organizations, and storage structures/devices.
- should not require a change to conceptual or external schemas.

Conceptual Database Design

To develop a conceptual data model that accurately reflects the data requirements of the enterprise, ensuring a solid foundation for further database design and implementation.

Entity Relation Modeling (ER Modeling)

Utilizing concepts such as Entity Types, Attributes, and Relationship Types to map out the database schema.

- Entity Type: a group of objects with the same properties, identified by an enterprise as having an independent existence.
- Attribute: property of an entity or a relationship type.
    - domain: set of allowable values for one or more attributes
    - simple attribute: attribute composed of a single component with an independent existence.
    - composite attribute: attribute composed of multiple components, each with an independent existence.
    - single-valued attribute: an attribute that holds a single value for each occurrence of an entity type.
    - multi-valued attribute: an attribute that holds multiple values for each occurrence of an entity type.
    - derived attribute: an attribute that represents a value that is derivable from the value of a related attribute or set of attributes, not necessarily in the same entity type.
- Relationship Type: Sets of meaningful associations among entity types.
    - relationship occurrence: uniquely identifiable association, which includes one occurrence from each participating entity type.
    - degree of relationship: number of participating entities in relationship
        - two: binary, three: ternary, four: quaternary
    - recursive relationship: relationship type where same entity type participates more than once in different roles. It may be given role names to indicate the purpose that each participating entity type plays in a relationship

- Strong Entity Type: not existence-dependent on some other entity type
- Weak Entity Type: existence-dependent on some other entity type

Multiplicity

- number/range of possible occurrences of an entity type
- may relate to a single occurrence of an associated entity type through a particular relationship
- represents policies established by the user or company
- Multiplicity for Complex Relationships: number of possible occurrences of an entity type in an n-array relationship when other n-1 values are fixed.

- Cardinality: describes the maximum number of possible relationship occurrences for an entity participating in each relationship type
  - one-to-one (1:1), one-to-many(1:*), many-to-many(*:*).
  - * can be replaced by N/M, e.g. (1;N) or (N:M)
- Participation: determines whether all or only some entity occurrences participate in a relationship.

Constructing an ER Model

1. identify entities
2. identify attributes and primary keys of entities
3. define relationships between entities
4. define cardinality and participation constraints in every relationship.

Logical Database Design

To translate the conceptual data model into a logical data model, then to validate this model to check that it is structurally correct using normalization and supports the required transactions, to create relations for the logical data model to represent the entities, relationships, and attributes that have been identified.

1. Derive Relations for Logical Data Model

Converting entities and relationships from the ER diagrams into tables and establishing relationships between them

Strong Entity: create a relation that includes all the simple attributes of that entity.

Weak Entity: create a relation that includes all the simple attributes of that entity.

- primary key is derived from each owner entity and so the identification of the primary key of the weak entity cannot be made until after all the relationships with the owner entities have been mapped.

Binary Relationship

- One-to-Many Relationship
  - “one side” is the parent entity and the “many side” is the child entity.
  - post a copy of the primary key attribute of the parent entity into the relation representing the child entity to act as a foreign key.
- One-to-One relationship
  - Mandatory participation on both sides
    - combine entities involved into one relation and choose one of the primary keys of the original entities to be the primary key of the new relation, while the other is used as an alternate key.
  - Optional participation on both sides
    - the choice of which entity's primary key to move is arbitrary. i.e. we can choose to post the primary key of A to B or vice versa.
  - Mandatory participation on one side
    - the primary key of the entity on the optional side is used as a foreign key in the entity on the mandatory participation side.
- Recursive relationships
    - same as a One-to-One relationship

- many-to-many relationship
    - create a relation to represent the relationship and include any attributes that are part of the relationship.
    - post a copy of the primary key attributes of the entities that participate in the relationship into a new relation, to act as foreign keys.
    - the foreign keys will form the primary key of the new relation, possibly in combination with some of the attributes of the relationship.
- complex relationship
    - create a new relation to represent the relationship and include any attributes that are part of the relationship, similar to many-to-many.
- multi-valued attributes
    - create a new relation to represent a multi-valued attribute and include the primary key of the entity in the new relation, to act as a foreign key.
    - unless the attribute itself is an alternate key, the primary key od the new relation is the combination of the attribute and the primary key of the entity.

2. Validate Relations using Normalisation
Applying normalization rules to minimize redundancy and avoid update anomalies.

3. Merge Logical Data Models into Global Model
merge logical data models into a single global logical data model that represents all user views of a database.

Functional Dependency

A relationship exists when one attribute uniquely determines another attribute. The attribute being determined is functionally dependent on the attribute determining it.
if A functionally determines B, it is denoted as A -> B.
- each value of A in R is associated with exactly one value of B in R.

Types of Functional Dependency (FD)
Full FD: Where all parts of a composite key are necessary to uniquely determine another attribute.
- if A and B are attributes of a relation, B is fully functionally dependent on A, if B is functionally dependent on A, but not on any proper subset of A. The subset of A cannot determine B.

Partial FD: When a subset of a composite key determines another attribute
- indicates that the schema is not in Second Normal Form (2NF)

Transitive FD: A, B, and C are attributes of a relation such that if B is functionally dependent on A, C is functionally dependent on B, then C is transitively dependent on A via B. (A is not functionally dependent on B or C)

Identifying Primary Key using FD
1. Identify all candidate keys, and identify the attribute that uniquely identifies each tuple in the relation. all attributes that are not part of a candidate key should be functionally dependent on the key.
2. The primary key is where all other attributes of the relation are functionally dependent on it.

Normalization

To produce a set of suitable relations supporting the data requirements of an enterprise.

Characteristics of a suitable set of relations

- minimal number of attributes necessary to support the data requirements of the enterprise.
- attributes with a close logical relationship are found in the same relation
- Minimal redundancy with each attribute represented only once with the important exception of attributes that form all or part of foreign keys.
- easier for the user to access and maintain data
- takes up minimal storage space on the computer

to have a set of suitable relations, we need

- Full Functional Dependency
    - one-to-one relationship between the attributes on the left-hand side(determinant) and those on the right-hand side of a functional dependency
    - holds all the time
    - determinant has the minimal number of attributes necessary to maintain the dependency with the dependent attributes.
    - remove partial dependency to prevent update anomalies.
- Remove transitive dependency to prevent update anomalies
    - its existence in a relation can potentially cause update anomalies in the insertion, deletion, and modification of records.
    - for example, deleting some part of the tuple can accidentally delete another part
- Minimise data redundancy anomalies to prevent update anomalies
    - updates to the data stored in the database are achieved with a minimal number of operations thus reducing the opportunities for data inconsistencies.
    - reduction in the file storage space required by the base relations thus minimizing costs.

Approach One

used as a bottom-up standalone database design technique to create a set of relations.

As normalization proceeds, the relations become progressively more restricted in format and less vulnerable to update anomalies.

Every binary relation(only 2 attributes) is always in Boyce-Codd Normal Form (BCNF). if a relation is BCNF, then it should be in 3NF, 2NF, 1NF.

Unnormalized form (UNF): a table that contains one or more repeating groups.

- to create an unnormalized table, transform the data from the information source into a table format with columns and rows.

First Normal Form (1NF): a relation in which the intersection of each row and column contains one and only one value.

- UNF to 1NF: identification and removal of repeating groups
    - nominate an attribute or a group of attributes to act as the key for the unnormalized table.
    - identify the repeating group in the unnormalized table which repeats for the key attributes.
    - remove the repeating group by
        - entering appropriate data into the empty columns or rows containing the repeating data
        - by placing the repeating data along with a copy of the original jey attributes into a separate relation.

Second Normal Form (2NF): a relation that is in 1NF and every non-primary key attribute is functionally dependent on the primary key.

- 1NF to 2NF: removal of partial dependencies
    - identify the primary key for the 1NF relation.
    - identify the functional dependencies in the relation
        - if there are partial dependencies, remove them, and place them into a new relation along with a copy of their determinant.

Third Normal Form (3NF): a relation that is in 1NF and 2NF and in which no non-primary key attribute is transitively dependent on the primary key.

- 2NF to 3NF: removal of transitive dependencies
    - identify the primary key in the 2NF relation
    - identify functional dependencies in the relation
        - if transitive dependencies exist, remove them and place them in a new relation along with a copy of their dominant

<u>Approach Two</u>
Used as a validation technique to check the structure of relations, which may have been created using a top-down approach such as ER modeling.

<u>Benefits of Normalisation</u>
Reduces Redundancy: Minimise duplicate data, ensuring that each piece of data is stored only once.

Eliminates Update Anomalies: Make the database more consistent by ensuring that updates, deletions, and insertions do not lead to data inconsistencies.

Improves Query Performance: In some cases, reducing the size of the tables and the dataset that queries need to navigate.

SQL (Structured Query Langauge)

Database language designed to use relations to transform inputs into required outputs.

- non-procedural (specify the information required)
- free-format

<u>SQL Statements</u>

- consists of reserved words and user-defined words
- case insensitive, except for literal character data
- an extended form of Backus-Naur form
    - upper case letters represent reserved words
    - lower case letters represent user-defined words
    - | represents choice among alternatives
    - {} indicate a required element
    - [] optional element
- literals
    - constants
    - non-numeric must be enclosed in single quotes ‘ ’
    - numeric cannot be enclosed in quotes

<u>Data Definition Language</u>

Used to create the database structure and the access mechanisms

<u>Primary Key</u>

- contain a unique, non-null value for each row
- only one per table
- ensure uniqueness by using UNIQUE()

<u>Foreign Key</u>

- column or set of columns that links each row in the child table containing the foreign key to the row of the parent table containing the matching primary key
- referential integrity: if FK contains a value, that value must refer to an existing row in the parent table
- FOREIGN KEY(___) REFERENCES _(parent)__

<u>CREATE SCHEMA</u>

- CREATE SCHEMA [Name | AUTHORIZATION CreatorId]

<u>DROP SCHEMA</u>

- DROP SCHEMA Name [RESTRICT | CASCADE]
- RESTRICT: schema must be empty or it fails
- CASCADE: drop all objects associated with schema, if one fails, the whole DROP SCHEMA fails

<u>CREATE TABLE</u>

- creates a table with one or more columns of the specified dataType
- NOT NULL: The system rejects any attempt to insert a null value
- DEFAULT: default value for the column
- PRIMARY KEY: always not null
- FOREIGN KEY: specifies the FK along with the referential action
- CREATE TABLE PropertyForRent (

```
        propertyNo Pnumber NOT NULL, ....
        rooms PRooms NOT NULL DEFAULT 4,
        rent PRent NOT NULL, DEFAULT 600,
        ownerNo OwnerNumber NOT NULL,
        staffNo StaffNumber
                Constraint StaffNotHandlingTooMuch ....
        branchNo BranchNumber NOT NULL,
        PRIMARY KEY (propertyNo),
        FOREIGN KEY (staffNo) REFERENCES Staff
        ON DELETE SET NULL ON UPDATE CASCADE ....);
```

- ALTER TABLE
    - add a new column, drop a column from table, add new table constraint, drop take constraint, set default for a column, drop default for column
    - DROP TABLE TableName [RESTRICT | CASCADE]
    - RESTRICT: if any other objects depend on their existence, SQL does not allow the request.
    - CASCADE: SQL drops all dependent objects.

Data Manipulation Language

Used to populate and query the tables.

INSERT

- insert data into a table
- INSERT INTO TableName [(columnList)]
  VALUES (dataValueList)
- columnList if omitted, SQL assumes all columns in the CREATE TABLE order
- any columns omitted must be declared as NULL when the table was created if DEFAULT was specified.
- dataValueList must match columnList
    - number of items are the same
    - direct correspondence in the position of the items
    - datatype must be compatible

INSERT … SELECT

- allows multiple rows to be copied from one or more tables to another.

UPDATE

- UPDATE TableName
  SET columnName 1 = dataValue1
          [, columnName 2 = dataValue2…..]
  [WHERE searchCondition]
- SET specifies names of one or more columns that are to be updated
- WHERE is optional
    - if specified, only rows that satisfy the condition are updated else all named columns are updated for all rows in the table.

DELETE

- DELETE FROM TableName
  [WHERE searchCondition]
- WHERE is optional
    - if specified, only rows that satisfy the condition are deleted else all rows from the table are deleted.

SELECT

- specify which columns are to appear in the output
- can only contain: column names, aggregate functions, constants, and expressions.
- FROM: specify the tables used
- WHERE: filter rows
    - = equals
    - !=/ <> not equal to
    - < less than
    - > greater than
    - <= less than or equal to
    - >= greater than or equal to
- AND, OR, NOT
    - evaluated left to right
    - expressions in brackets evaluated first (recommend to use parenthesis)
    - NOTs before ANDs and ORs, ANDs before ORs

- Range Search
    - BETWEEN and NOT BETWEEN
    - BETWEEN …. AND ……
- Set Membership
    - IN and NOT IN
- Pattern Matching Search Condition
    - %: zero or more characters
    - _: single character
    - LIKE ‘H%’: the first letter must be H, rest can be anything
    - LIKE ‘H_ _ _’: first-letter H, the word is four letters
    - LIKE ‘%e’: words end with e
    - LIKE ‘%glasgow%’: sequence of any length that contains glasgow
- NULL search condition
    - … IS NULL
    - selects those where the column value is NULL
- GROUP BY: forms groups of rows with the same column value
    - to get sub-totals
    - all column names in SELECT must appear in the GROUP BY clause unless the name is only used in an aggregate function.
    - if used together with WHERE, WHERE is applied first then groups are formed from the remaining rows satisfying predicate.
- HAVING: filter groups subjects to some condition
    - restrict groups that appear in the final result table
    - column names in the HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.
- ORDER BY: order of the output
    - if ordering by more than one column, order it from left to right
    - ASC: ascending order
    - DESC: descending order
- DISTINCT: to eliminate duplicates
- AS: to name column

Aggregates
- COUNT: number of values in the column
- SUM: sum of values in the column
- AVG: average of values in the column
- MIN: smallest value in the column
- MAX: largest value in the column
- DISTINCT may have effects with COUNT, SUM, and AVG but not with MIN, MAX
- if the select statement includes an aggregate function but no GROUP BY clause, then no item in the select can include any reference to a column unless the column is the argument to an aggregate function.
- COUNT, MIN, and MAX apply to numeric and non-numeric but SUM, and AVG only on numeric values
- except COUNT, all function eliminates nulls first and operate only on remaining non-null values.
- COUNT(*) counts all rows of a table, regardless of whether nulls or duplicate values occur.

Subqueries

- select statement embedded
- can be used in WHERE and HAVING clauses
    - ‘WHERE salary > AVG(salary)’ is wrong. use ‘salary > (SELECT AVG(salary) from ……)’
- may also appear in INSERT, UPDATE, and DELETE statements.
- ORDER BY may not be used
- must consist of a single column name or expression unless it uses EXISTS
- when a subquery is an operand in a comparison, it must be on the right-hand side
- subquery cannot be used as an operand in an expression.

ANY and ALL

- ALL: true only if it is satisfied by all values
- ANY/SOME: true if it is satisfied by any values
- can be used with subqueries that produce a single column of numbers.
- if the subquery is empty, ALL returns true, ANY returns false.

Multiple Tables

- can use subqueries provided result columns come from the same table.
- if result columns come from more than one table, must just join.
- to include more than one table in the FROM clause
- can use alias for a table named in the FROM clause.
- alias is separated from table name with a space
- alias can be used to qualify column names when there is ambiguity.

JOIN

- JOIN … ON …. = …..
- JOIN …. USING ……
- NATURAL JOIN

INNER JOIN

- combines data from two tables by forming pairs of related rows where the matching columns in each table have matching columns in each table have the same value.
- if one row of a table is unmatched, the row is omitted from the result table.

OUTER JOIN

- LEFT OUTER JOIN
    - includes rows of the first table unmatched rows with the second table
    - records from the left table and those matched from the right table
- RIGHT OUTER JOIN
    - includes rows of the second table that are unmatched with the first table.
    - records from the right table and those matched from the left table
- FULL OUTER JOIN (CROSS JOIN for MySQL)
    - returns all unmatched rows.

EXISTS and NOT EXISTS

- only for subqueries
- produce true/false
- true if there is at least one row in the result table
- false if the result table is empty
- SELECT …. FROM ….
  WHERE EXISTS (SELECT ……….)
- subquery can contain any number of columns as it only checks for existence

CASE (similar to if-else)

- goes through conditions and returns a value when the first condition is met.
- once a condition is true, it will stop reading and return the result.
- if no condition is met, return value in the ELSE clause
- if there is no ELSE CLAUSE, return NULL.

Data Visualization

Representation and presentation of data that exploits our visual perception abilities in order to amplify cognition.

Terminologies

- Representation: the way data is depicted
    - how to give form using visual variables to construct chart or graph types
    - circle maps, filled maps, scatterplots, slope graphs, bar charts, radial network diagrams, matrix charts, and more……
- Presentation: how data is integrated into the overall communicated work including colours, annotations, and interactive features.
    - make sure it is used unobtrusively
    - does not mislead by implying representation when it should not be.
    - use lightness of colour to depict the range of quantitative values.
- Visual Perception Abilities: how our eyes and brains process information most effectively
    - Preattentive reaction
    - Gestalt Laws of Similarity
    - Gestalt Laws of Proximity

Communicating Data

- make data attractive and easy to understand
- amplify cognition: maximizing how efficiently and effectively we are able to process the information into thoughts, insights, and knowledge.

Value

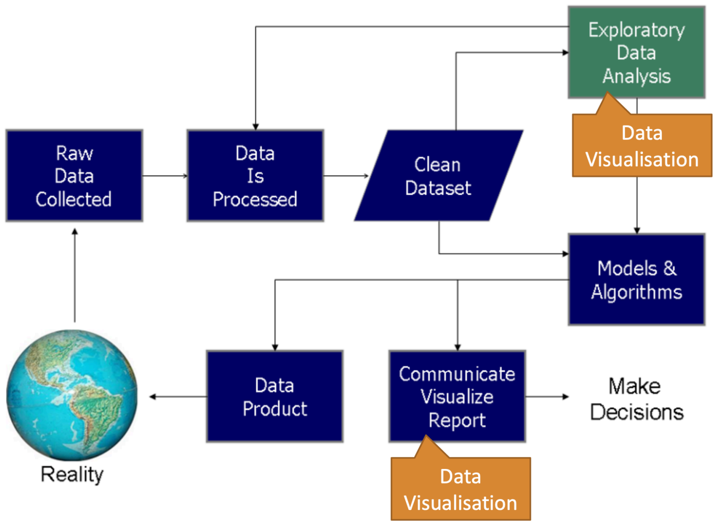
- to pull out key insights and for people to make decisions and act on that data.
- seek to portray data in ways that allow us to see it in a new light, to visually observe patterns, exceptions, and the possible stories that sit behind its raw state.
- well-designed visualization makes it more persuasive, and memorable and creates an emotional response.
- make a reader or users feel like they are more informed about a subject.

Process

- allows us to understand our data
- to see patterns, outliers, trends, and context
- therefore, make decisions
- take actions that are relevant to the business

Three Elements

- data: facts to be presented
- visuals: to see patterns, trends, and context
- narratives: using visualizations to tell a “data story”



Interactivity
- manipulating variables and parameters
- adjusting the view
- annotated details
- titles, introductions, user guides, labels, captions, narratives, legends, keys, units, data sources, attribution

data presentation and representation

Perceiving: attempt of viewers to decode the representations of the data and then convert them into perceived value

Interpreting: viewers' ability to form interpretations is influenced by their pre-existing knowledge about the portrayed subject matter and their capacity to utilize that knowledge to frame the implications of what has been read/viewed.

Comprehending: reasoning the consequence of perceiving and interpreting stages to arrive at a personal reflection of what all this means.

Functions

Exploratory: provide an interface to data to facilitate visual exploration
- more about visual analysis, aim to provide users with an interface to visually explore the data, to seek out personal discoveries, patterns, relationships, triggering, and iterating curiosities.
- example: scatterplot matrix
    - reveal correlations across a multivariate dataset, allowing the eye to scan the entire matrix and quickly identify variable pairings with strong or weak relationships.

Explanatory: convey an explanatory portrayal of data to a reader.
- conveying information that is based on a specific and focused narrative.
- requires a designer-driven, editorial approach to synthesize the requirements of the target audience with the key insights and the analytical dimensions that you wish to convey.
- example: Sankey diagram

Exhibition: use data as an exhibition of self-expression
- lack of structured narrative and absence of any visual analysis capability.
- it is to create an artifact, an aesthetic representation, or a technical demonstration

Visualization Tones

Pragmatic and Analytical
- delivers fast, efficient, and precise portrayals of data.
- such as bar charts, line charts, and dot plots.
- for a captive audience such as a corporate environment

Emotive and Abstract
- creating an aesthetic that portrays a general story or sense of pattern
- might not be able to pick out every data point or category.
- known reduction in the accuracy of value perception.

Good Data Visualisation
- accurate and not misrepresent data
- easy to understand
- relates to audience
- show only what is necessary

Design Principles

- use ink strategically: add elements only when helpful
- use good text design
    - use sans serif
    - use caps, bold, italics, and bigger fort size to create emphasis
    - keep fonts consistent
    - use a title for the takeaway message

- use good colour design
        - dark/bright colours read as “more”
        - grey to denote neutral or unimportant
        - choose an appropriate colour palette and keep them consistent.
        - red is perceived as negative
        - higher contrast between the objects, the more differently they are perceived.
    - use icons, photography, and interactivity

Data Storytelling

Combining data with visuals and narrative

Steps to build a data-driven story

- start by getting data right
- ensure all storylines are backed by data
- build trust in the narrative through data
- need all data points to tell the story
- never exaggerate or manipulate data

strategy

- define business critical question (level 1) that needs answering
- collect essential data
- define questions that address the business critical questions (level 2)
- for each question, produce data visualizations/insights to address the question
- define the main insight from the insights
- resolve with recommendation
- determine conclusion with next steps
- craft a coherent story based on insights
- present the story

NoSQL

presumption: Big data cannot be handled by the standard, RDBMS-based data systems

Relational Database

- maintains data integrity by storing data in tightly structured relations, with constraints and the ability to identify and find any piece of data uniquely across relations.
- robust and reliable use with 99.999% uptime
- manages concurrency through transactions
- still the most widely used

Changing Trends
- volume of data increasing too fast to be handled by one machine
- data is no longer structured, simple, and formatted
- application development is getting shorter
- applications are 24/7, globally distributed for millions of users to use concurrently with continuous streams of data.

NoSQL Database
- provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databses.
- also known as “Not only SQL” to emphasize that they may support SQL query languages.

ACID transactions in a Relational database

Atomicity: transactions are applied in an “all or nothing” manner, such that any commit made finishes the entire operation successfully or the database is rolled back to its state before the commit is initiated.

Consistency: data written to a database must be valid according to defined rules.

Isolation: ensures that transactions being executed in parallel would result in a final state identical to that which would be arrived at if the transactions were executed serially, the reads or writes performed on the database do not impact other reads and writes of separate transactions occurring on the same database.

Durability: once a transaction has been committed, it will stay committed so that changes made to the database will survive permanently.

BASE transactions in the NoSQL database

Basic Availability: each request is guaranteed a response -- successful or failed execution

Soft state: state of the system may change over time, without any input.

Eventual consistency: The database may be momentarily inconsistent but will be consistent eventually.

Characteristics of NoSQL
- Schemaless data representation: structure evolves over time, and can introduce new fields and nesting in the data structure.
- Development time: reduced because one does not have to deal with complex SQL queries.
- Speed: deliver in milliseconds rather than hundreds of milliseconds
- Plan ahead for scalability: The application can be elastic, and it can handle sudden spikes of load.

Document Stores
- MongoDB, CouchDB, RavenDB, Terrastore
- storage of a mass of documents, usually JSON/XML
- each document can have different, complex data structures
- no schema (therefore flexible)
- can query for any field in the data and retrieve part or all of the data based on the query.

Key : Value Stores
- Redis, Membase, Voldemort, MemcacheDB
- focuses on high performance, availability, and scalability by storing data in key : value pairs
- every value is associated with a unique key and retrieval must be fast.

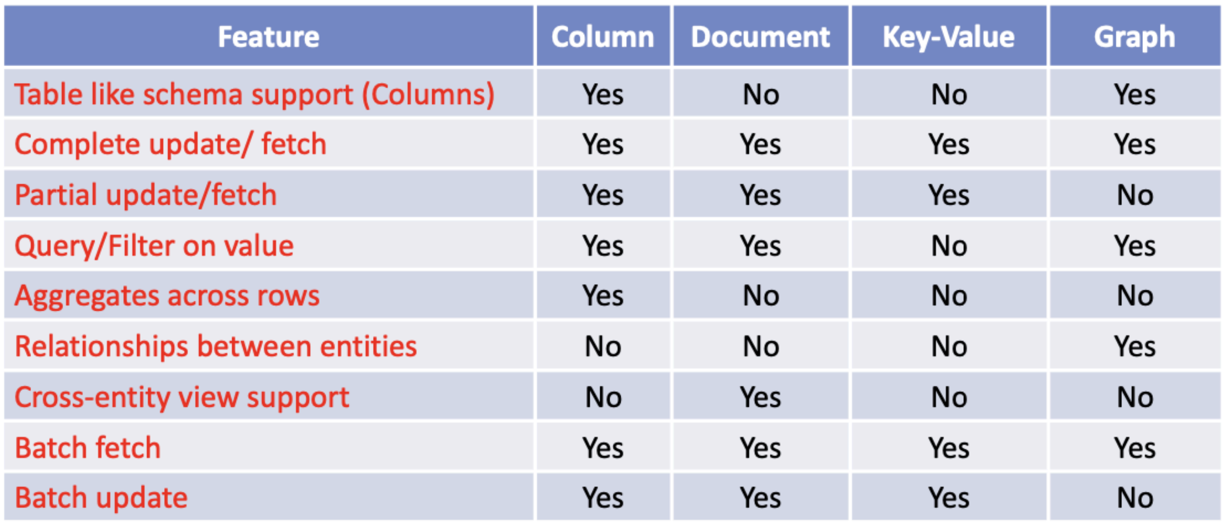
Column

- BigTable, Hadoop/HBase, Cassandra, SimpleDB, and Cloudera
- data are stored in rows but analyses involve looking at columns
- data items for each attribute are stored one after another on the disk.
- compress data for each attribute easily
- doing a row-based search is difficult, and frequent updates would be inefficient.
- reading of the entire column is fast.

Graph

- Neo4J, FlockDB, InfiniteGraph
- use graph structures with nodes and edges
- relationships are edges between nodes(data)
- each node has properties, labels, maybe a schema, and identifiers.
- query language enables searching and using the data in the graph form, in much more efficient ways than we can do with relational databases, such as using paths.

It has not been widely adopted yet because most businesses have legacy systems running on RDBMS. If they were to use OODBMS, they would still need to access RDBMS for certain functions. It is easier to manage one way of accessing data than two.



Feature	Column	Document	Key-Value	Graph
Table like schema support (Columns)	Yes	No	No	Yes
Complete update/ fetch	Yes	Yes	Yes	Yes
Partial update/fetch	Yes	Yes	Yes	No
Query/Filter on value	Yes	Yes	No	Yes
Aggregates across rows	Yes	No	No	No
Relationships between entities	No	No	No	Yes
Cross-entity view support	No	Yes	No	No
Batch fetch	Yes	Yes	Yes	Yes
Batch update	Yes	Yes	Yes	No