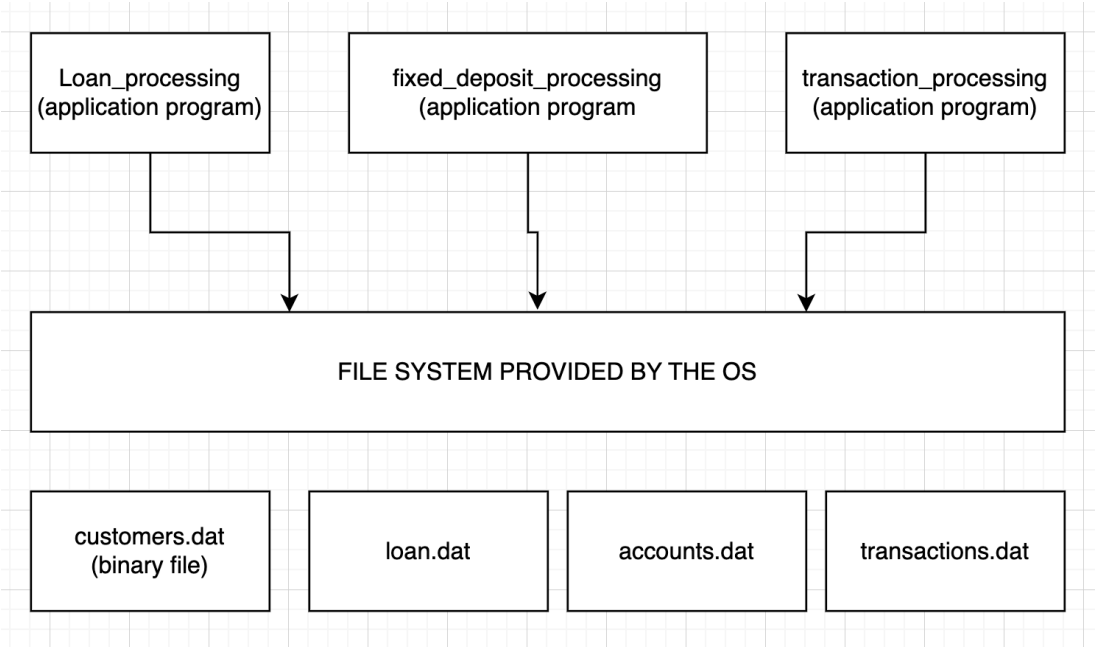
RDBMS Concepts

DB and DBMS

- Data
    - Meaningful facts, text, graphics, audio, video, and more complex values
    - A piece of information
- Database
    - An organized collection of logically related data
    - A collection of entities
    - Represents some aspect of REAL world
        - AKA miniworld
        - Example:
            - HR MANAGEMENT
            - ORDER PROCESSING SYSTEM
            - PAYROLL SYSTEM
            - ECOMMERCE APP
    - Some amount of interaction with events in the real world
    - Changes to the miniworld are reflected in the database
- Information
    - Processed data
    - Users are interested in information and not data
    - Applications use/access the data from some storage media, and provide very useful information
- Metadata
    - Data that describes other data
    - Example:
        - data → employee_id = 1234
        - metadata → type -> INTEGER, min -> 0, max-> 99999

Traditional (legacy) method of data storage

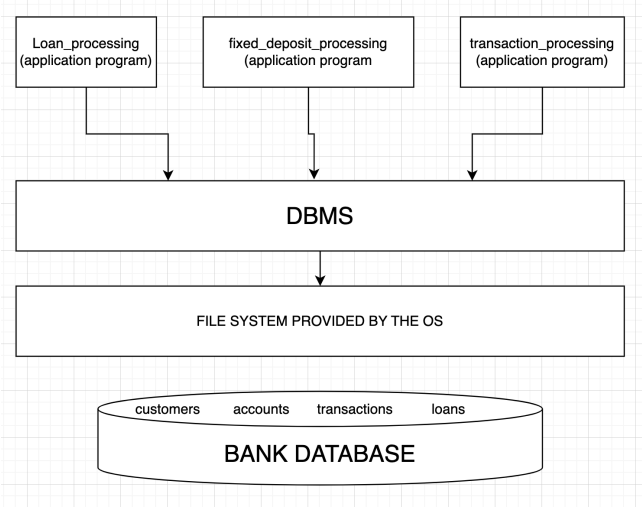


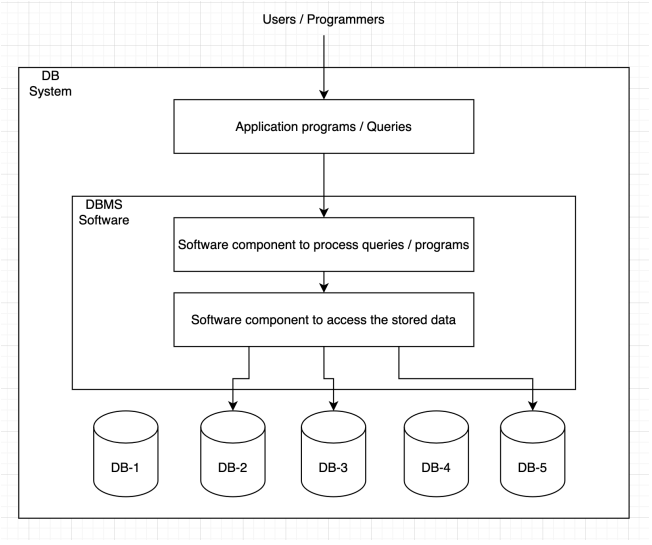
Limitations / downside of this approach:

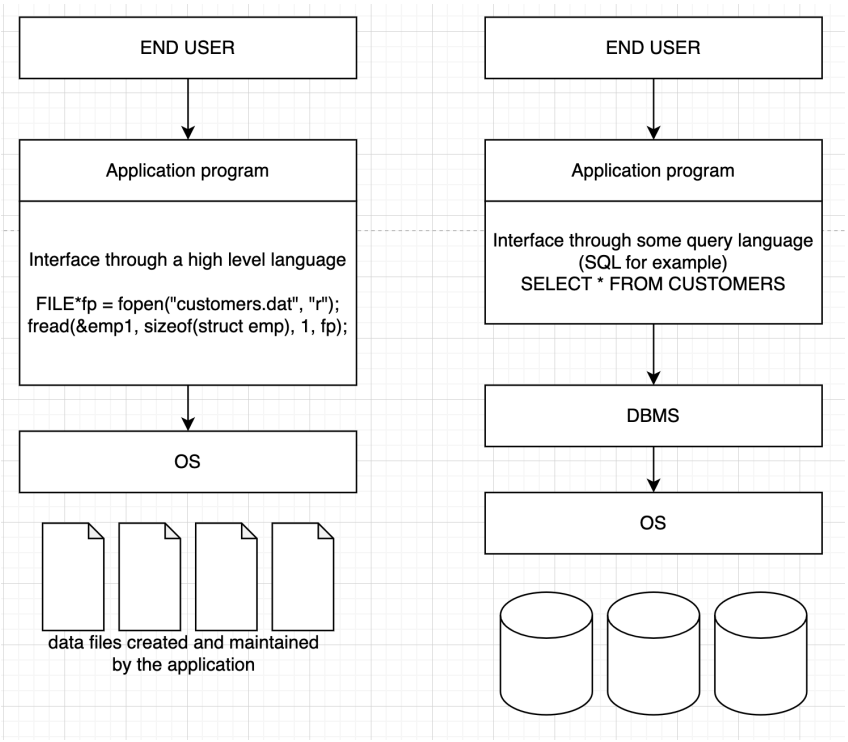
- Data dependence
- Data redundancy (duplication of data)
- In a networked environment, limited data sharing ability
- Development time is lengthy
- Excessive program maintenance

What is the alternative to this? - DBMS

- Database Management System
- Controlling redundancy
- Sharing of data with other applications
- Restrict unauthorized access
- Provide multiple interfaces
- Enforcing of constraints
- Provide facility for backup and recovery of data
- Enforce standards
- Reduce application development time
- Data should be available all the time, and must be up-to-date
- Provide a common language for communication (across different implementations of DBMS)

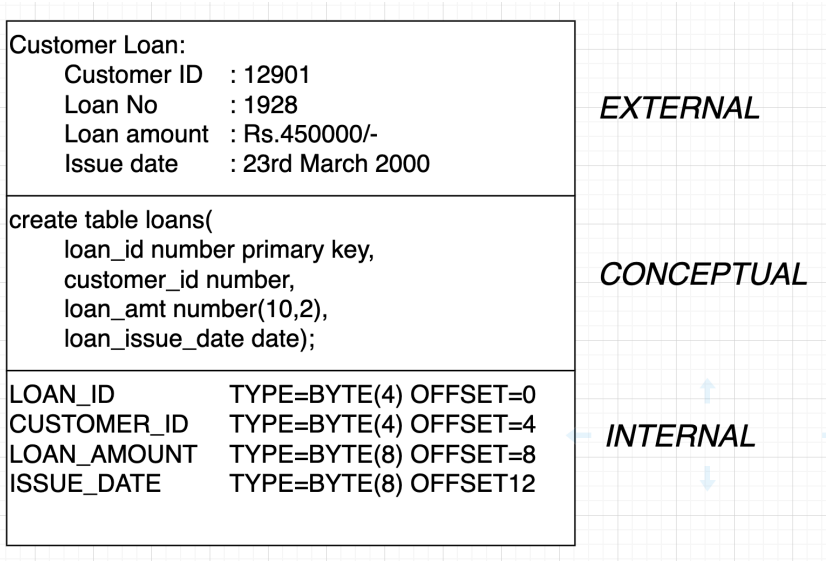






Different views of a data

1. External view (individual users)
2. Conceptual view
3. Internal / Storage view



Data model

- A Collection of Concepts
- Describe the structure of the database
- Describe the way “People” perceive (picturise) data

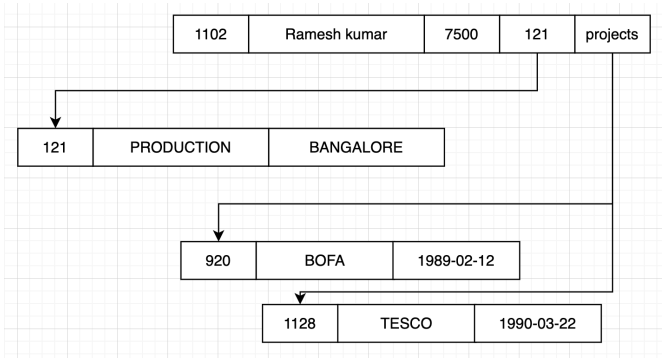
Different data model implementations:

1. Hierarchical (record based) data model (legacy)
2. Network data model (legacy)
3. Relational data model

```
struct employee {
       int id;
       char name[50];
       float salary;
       struct department dept;
       struct project* projects;
};

struct department {
       int id;
       char name[50];
       char location[50];
};

struct project {
       int id;
       char name[50];
       date start_date;
};
```



Few drawbacks of hierarchical model

- Knowledge of the structure and its physical representation is required by the developer
- Does not support for modifying the structure
- Time consuming - processing is done one record at a time

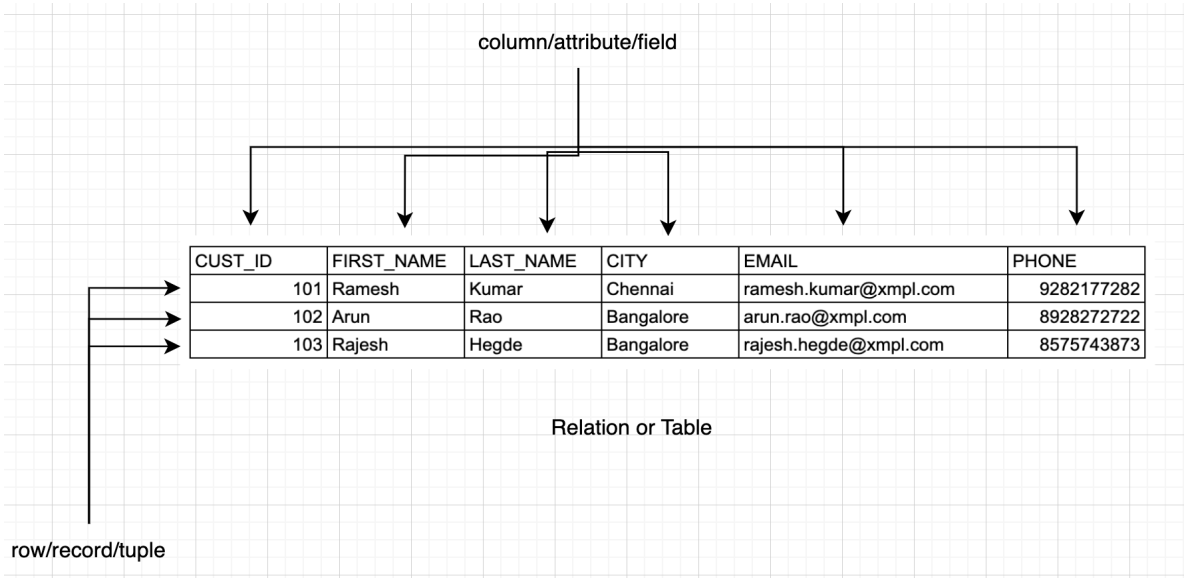
Network model (Self study)

Record based Relational data model

Terminologies:

1. Row (record or tuple)
2. Column (attribute or field)
3. Table (Relation)

External view of a Relation / Table:



Some key pointers:

- A database ia collection of related tables (of the miniworld)
    - For example, in a HR management system, the relations (tables):
        - EMPLOYEES
        - DEPARTMENTS
        - HOLIDAYS
        - EMPLOYEE_LEAVES
        - JOBS
    - In a order processing system:
        - PRODUCTS
        - SUPPLIERS

        - CUSTOMERS
        - CUSTOMER_ORDERS
- Every relation (table) has a unique name in a database
- Every attribute in a relation has a unique name
- The value of an attribute is atomic
- Every row in a relation is also unique
- Sometimes a value of an attribute may be unknown or have no value - this is represented using a special value called NULL
    - It is neither zero nor blank nor empty string
- Keys
    - Candidate key - set of one or more columns that may uniquely identify a row in a given table
        - email_id, phone, emp_no, passport_no, aadhar_no - all of these may uniquely identify an employee in a table full of employees
    - Primary key - Designated column among the candidate keys. The value of this may never be NULL nor duplicate
    - Composite primary key - When a primary key is a combination of more than one column
    - Foreign key - It is a regular column in one table but the same column exists in another table as a primary key. For example, DEPT_NO is the primary key in the DEPARTMENTS table, but may exist in the EMPLOYEES table (indicating the department number of the employee) as a FOREIGN KEY
- Integrity Constraints
    - Domain (field level) integrity
        - Allowable values for an attribute
        - Data type, NULL or NOT NULL, CHECK
    - Entity Integrity
        - No primary key value may be null (unique as well)
    - Referential Integrity
        - When FOREIGN KEY is applied on a column
        - Cascade UPDATE or cascade DELETE rules

ER (Entity Relationship) Model

Constitutes:

- Entity
    - Source of information
    - Person, Customer, Employee, Product, …
    - Nouns
- Attribute
    - Data of an entity
    - For the entity Customer
        - name, address, city, state, country, email, phone, …

- Relationships
    - Relationship between two or more entities
    - CUSTOMER ***places order*** for PRODUCTS
    - STUDENT ***registers*** for a COURSE
    - EMPLOYEE ***applies*** for a LEAVE
    - DOCTOR ***prescribes*** MEDICINE for a PATIENT

Simple Vs Composite attributes

- Simple attribute
    - Can not be divided into more simpler attributes
    - For example, *age* or *first_name* of a person
- Composite attribute
    - Can be split into more attributes
    - For example, *address* of a CUSTOMER can be split into HOUSE_NAME_NO, STREET, CITY, STATE, COUNTRY, PINCODE
    - *date_of_birth* of an EMPLOYEE can be split into DAY, MONTH, YEAR

Single valued Vs Multi-Valued attributes

- Single valued:
    - Can take a single value for each entity instance
    - For example, for an employee, the *salary* attribute may take only one value
- Multi-valued:
    - Can take multiple values (usually comma separated values)
    - For example, the *skill_set* attribute for an EMPLOYEE can be
        - “java, c, c++, c#”
        - “python, php, mysql, html”
        - “html, javascript, css, angular, react”

Stored Vs Derived attribute

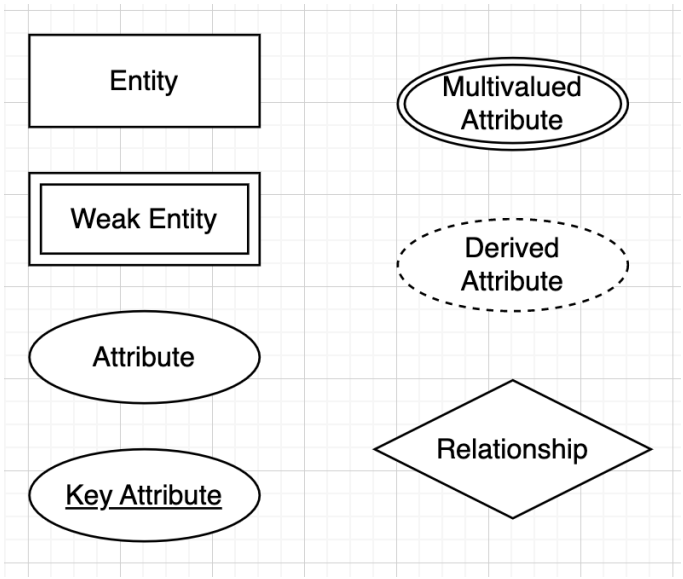
- Stored attributes:
    - Usually a fixed value, stored in the table permanently
    - For example, *first_name* of a CUSTOMER
- Derived attributes:
    - Calculated periodically and stored, based on a value of another attribute
    - For example, *age* of a PERSON is calculated based on the *date_of_birth*
    - *order_total* for an ORDER placed by a CUSTOMER is a calculated field (based on the products, quantities, discounts, etc.

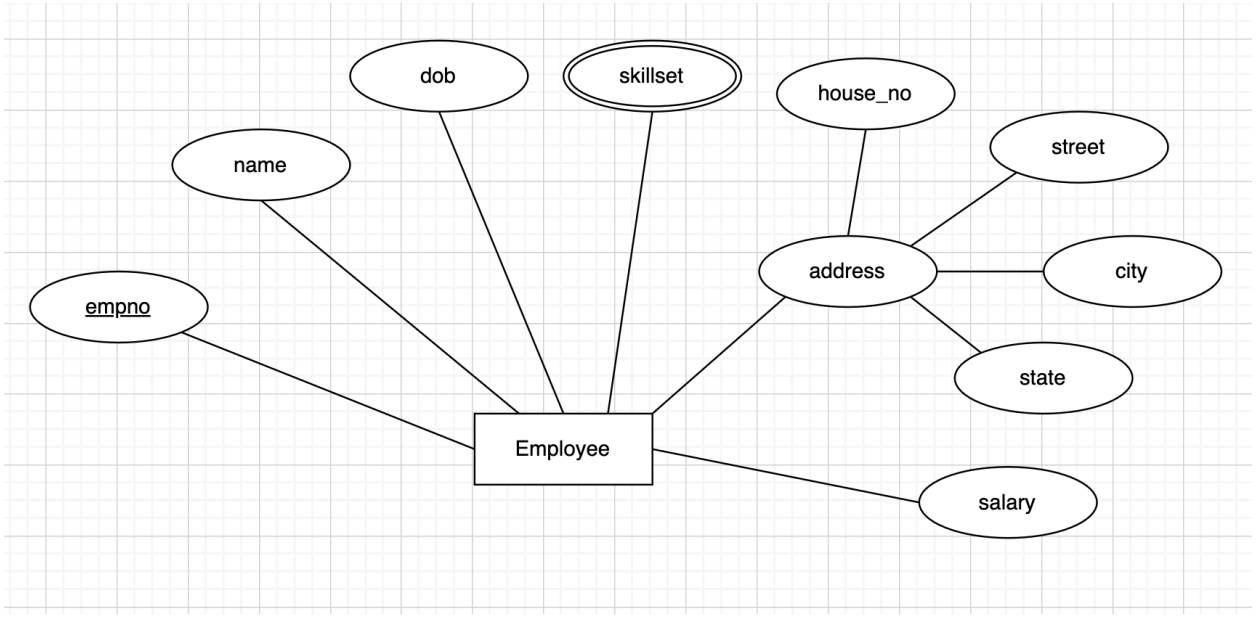
Regular Vs Weak entities

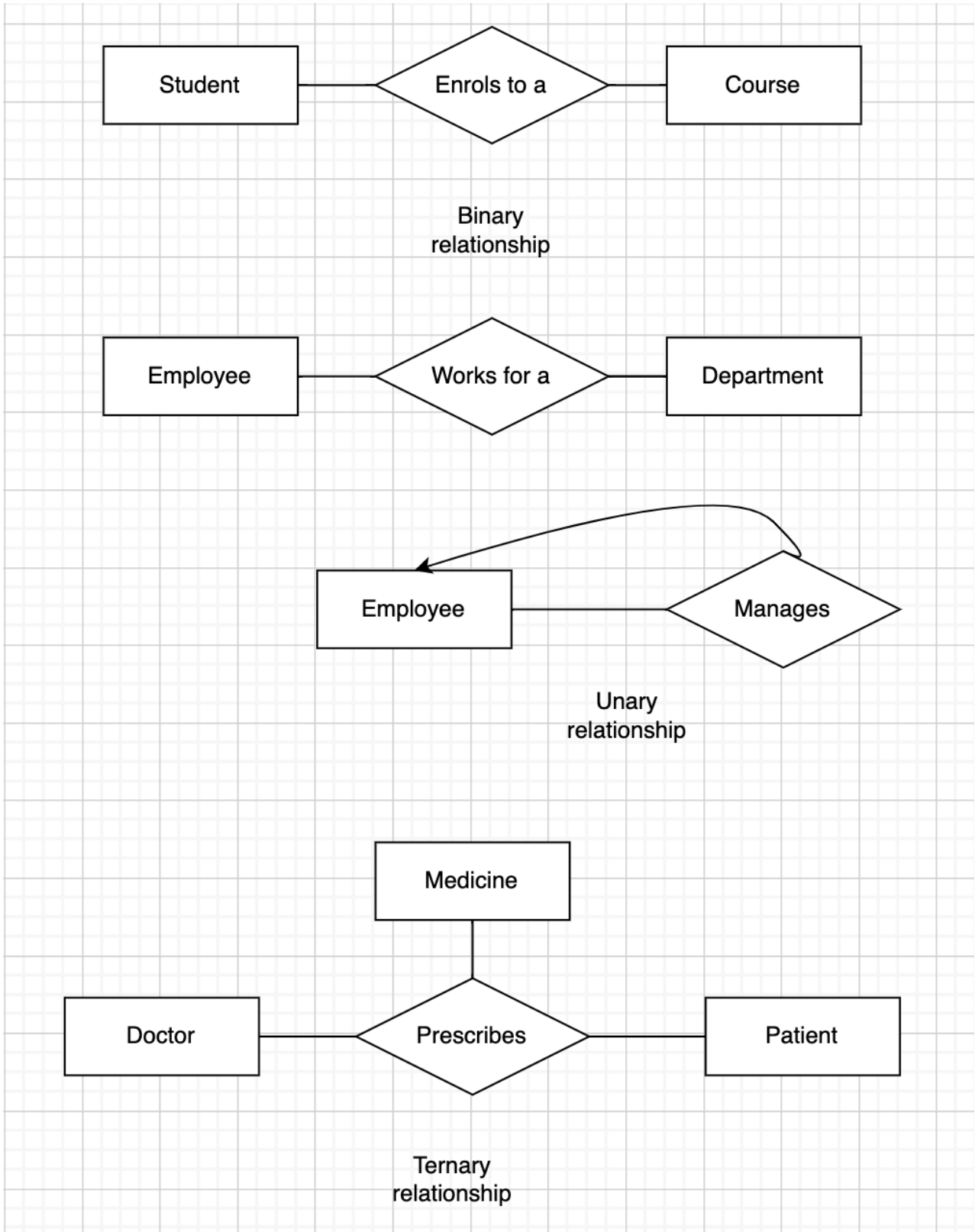
- Regular entity
    - Entity that has its own PRIMARY KEY attribute
    - For example, EMPLOYEE, PRODUCT, CUSTOMER, etc
- Weak entity:
    - An entity that does not have own PRIMARY KEY, but uses the PRIMARY KEY of another entity
    - For example, *NOMINEE* of an insurance policy holder
    - Very rarely found

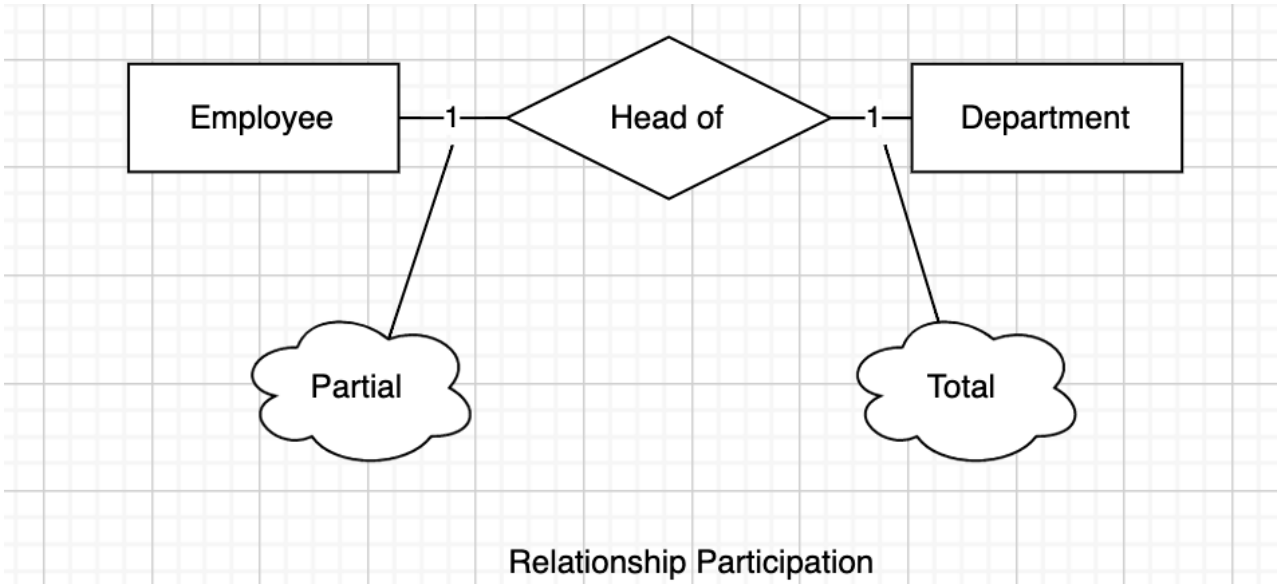
Relationships

- A relationship between two entity types defines the set of all associations between those two entities
- An instance of the relationship between these two entities, is called *a relationship instance*
- Degree:
    - Unary → Only one entity is participating in the relationship
        - *Employee → Manager_of_employee*
    - Binary → Two entities are participating in the relationship
        - *Employee → works_in → Department*
    - Ternary → Three entities are participating in the relationship
        - *Doctor → prescribes → Medicine for Patient*
        - *Customer → purchases → Product* delivered by *Courier*
- Cardinality:
    - One-To-One
        - EMPLOYEE → LAPTOP
        - HOD → DEPARTMENT
    - One-To-Many
        - DEPARTMENT → EMPLOYEE
    - Many-To-One
        - EMPLOYEE → DEPARTMENT
    - Many-To-Many
        - EMPLOYEE → PROJECT
        - one-to-many from both sides
- Relationship participation
    - Total
        - Every entity instance must be connected through the relationship to another instance of the participating entity types
        - Every employees *belong to* department/s
    - Partial
        - All entity instances need not participate in the relationship
        - Employee *heads a* department
            - Department → total participation (all departments have HOD)
            - Employee → partial participation (all employees cannot be HOD)

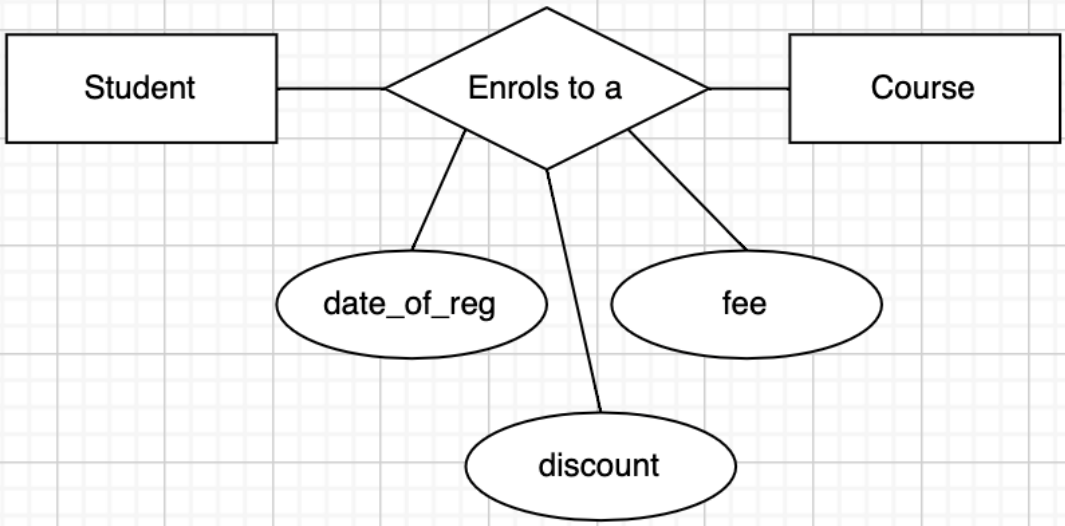

Entity
Multivalued Attribute
Weak Entity
Derived Attribute
Attribute
Relationship
Key Attribute


dob
skillset
house_no
name
street
address
city
empno
state
Employee
salary


Student
Enrols to a
Course
Binary relationship
Employee
Works for a
Department
Employee
Manages
Unary relationship
Medicine
Doctor
Prescribes
Patient
Ternary relationship



Relationship Participation



Attributes of a relationship

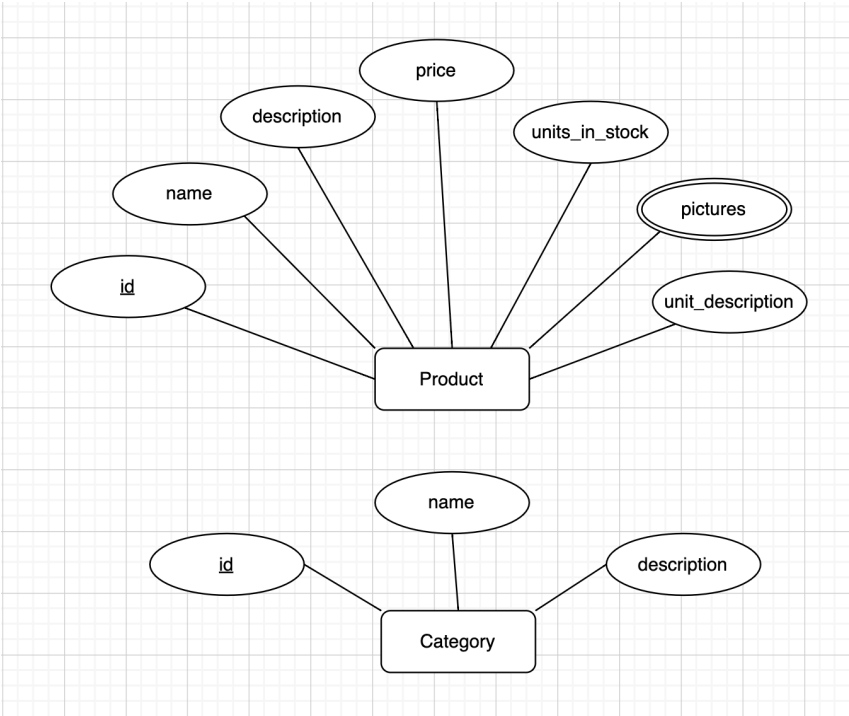
ER Diagrams

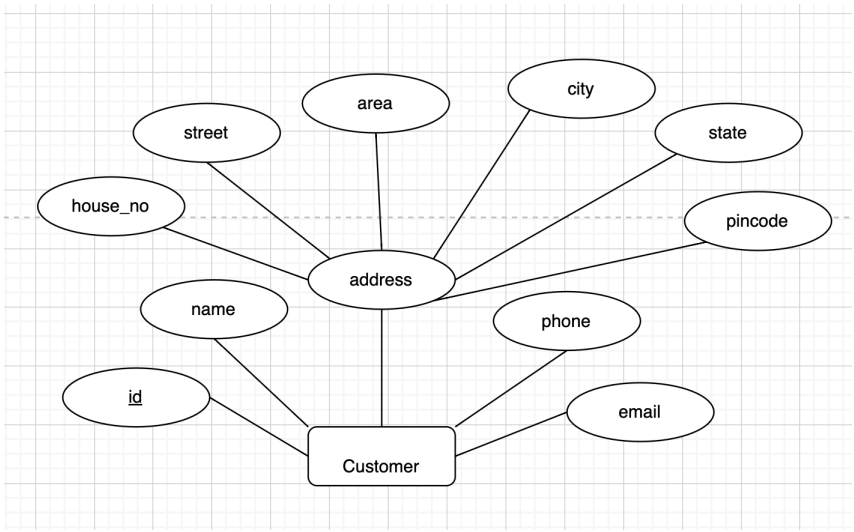
"Blu Yu" is a new brand of mobile phones and accessories. The company wants to set up a web and mobile application to showcase and sell their products. In this regard, you are to create the required database tables. There are several products in a few categories. The categories are:

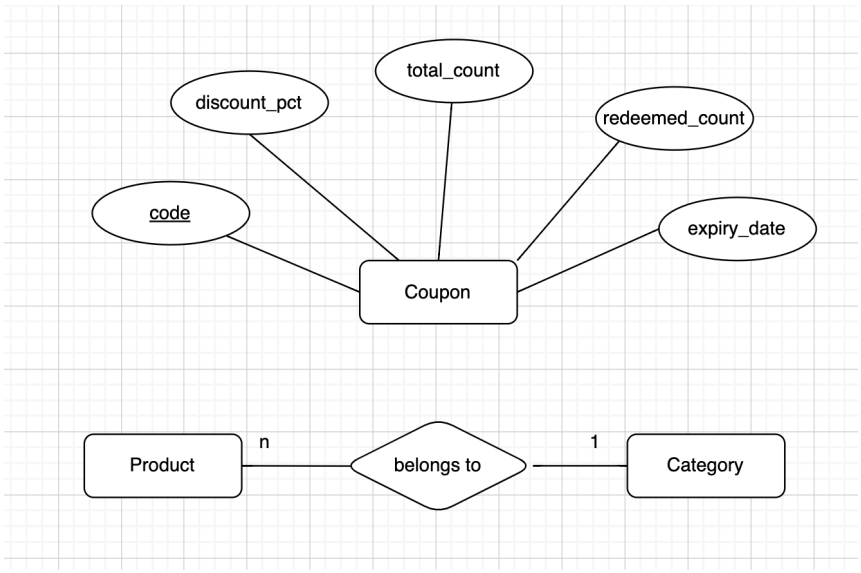
- Smartphones
- Wifi devices
- Bands
- Power banks
- Headphones
- VR Players
- Charging accessories
- Bluetooth speakers ... and more categories to come in the future. In each category, there may be one or more products. In order to showcase all these products, there may be one or more pictures (3 to 5) for each of the products.

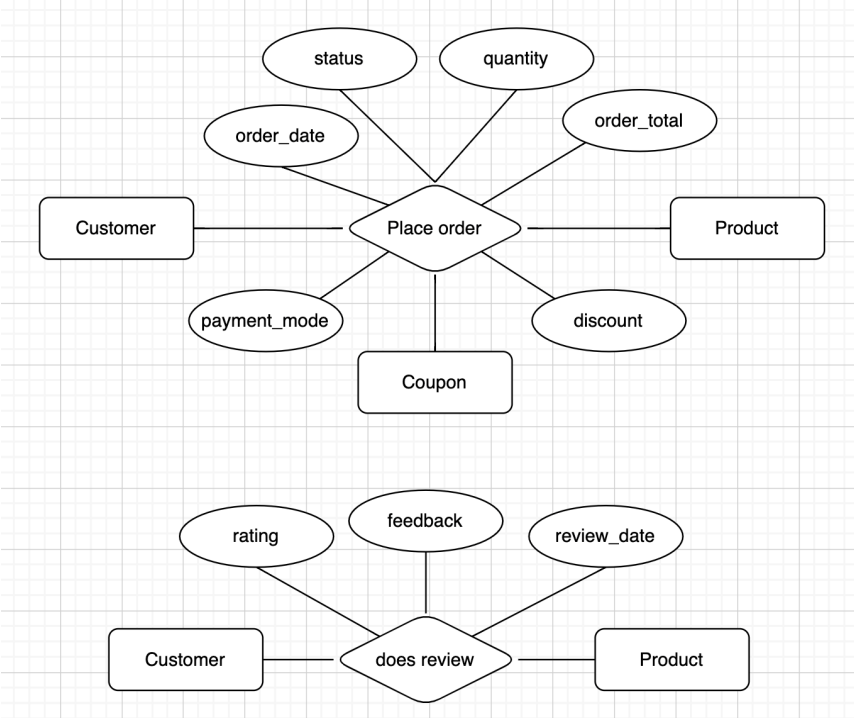
The application allows customers to browse, choose and buy products. A customer may place an order, which consists of one or more products of varying quantity. Occasionally, the company releases coupon codes, against which the customer may get good discounts. The coupon code is entered by the customer during the checkout process. The customers optionally re-visit the application to review the product they may have used and give rating and feedback on the same.

Identify the various entities, attributes and the relationships between entities from the above case study. Create proper ER diagrams representing the same.




area
city
street
state
house_no
pincode
address
name
phone
id
email
Customer


total_count
discount_pct
redeemed_count
code
expiry_date
Coupon
Product
n
belongs to
1
Category



Transforming ER Diagrams into Relations (Tables)

- Map regular entities to relations
- Composite attributes - use only their simple attributes
- Multi-valued attribute - (picture, for example in product) becomes a separate relation with a foreign key taken from the superior (product) entity
- Weak entity - becomes a separate relation with a foreign key taken from the superior entity
- Mapping relationships:
    - Binary 1:N
        - Primary key of the relation on “1” side of the relationship becomes a foreign key in the relation on “n” side of the relationship
    - Binary N:M
        - A new table is created representing the relationship
        - Contains two foreign keys - one from each side of the relationship
    - Binary 1:1 *(Self study)*
    - Unary *(Self study)*

Product (id, name, description, price, units_in_stock, units_description, *category_id*)
Picture (id, picture_url, *product_id*)
Category (id, name, description)
Customer (id, name, house_no, street, area, city, state, pincode, phone, email)
Coupon (code, discount_pct, total_count, redeemed_count, expires_on)
Customer_Order (id, *customer_id*, *product_id*, *coupon_code*, order_date, status, quantity, discount_pct, payment_mode)
Customer_Review (id, *customer_id*, *product_id*, review_date, rating, feedback)

PRODUCTS

ID	NAME	DESCRIPTION	PRICE	UNITS_IN_STOCK	UNITS_DESCRIPTION	CATEGORY_ID
128	BY283 Power Bank	1000 MAH battery powered portable power bank	5600	12	1 power bank with type-C chargin	82
283	BYBT 202123 Bluetooth speaker	Stylish stereo enabled bluetooth speaker	899	90	Speaker with USB cable	37

PRODUCT_PICTURES

ID	PICTURE_URL	PRODUCT_ID
1	https://example.com/pics/asdf.jpeg	128
2	https://example.com/pics/sdf.jpeg	128
3	https://example.com/pics/rtyh.jpeg	128
4	https://example.com/pics/hjhdtf.jpeg	283
5	https://example.com/pics/wecvhgfh.jpeg	283

CATEGORIES

ID	NAME	DESCIPTION
12	Smart Watch	
82	Power banks	
37	Audio devices	
47	Smart phones	

CUSTOMERS

ID	NAME	HOUSE_NO	STREET	AREA	CITY	STATE	PINCODE
72831	Vinod Kumar	TF-1 Elegant Elite	1st cross 1st main	ISRO lyt	Bangalore	Karnataka	560078
73836	Ravish Khanna	874	7th main	JP NAGAR	MYSORE	Karnataka	

COUPONS

COUPON_COD	DISCOUNT_PCT	TOTAL_COUNT	REDEEDMED_COUNT	EXPIRES_ON
82JASGIWY383	25	100	2	30-Nov-2022
746HJDGD3763	50	100	78	31-Dec-2022

Customer_Review (id, customer_id, product_id, review_date, rating, feedback)

CUSTOMER_REVIEWS

ID	CUSTOMER_ID	PRODUCT_ID	REVIEW_DATE	RATING	FEEDBACK
45	72831	128	15-Nov-2022	8	WORTH THE MONEY
46	72831	283	15-Nov-2022	6	Small cracking noise in between
47	73836	283	16-Nov-2022	3	Don't buy this product

CUSTOMER_ORDERS

ID	CUSTOMER_ID	PRODUCT_ID	COUPON_CODE	ORDER_DATE	STATUS	QUANTITY	DISCOUNT_PC	PAYMENT_MODE
56	72831	***PROBLEM***	82JASGIWY383	12-Nov-2022	DELIVERED	***PROBLEM***		
57								
58								
59								
60								

SQL Commands

1. DDL - Data Definition Language
    a. CREATE
    b. ALTER
    c. DROP
2. DML - Data Manipulation Language
    a. INSERT
    b. UPDATE
    c. DELETE
3. TCL - Transaction Control Language
    a. COMMIT
    b. ROLLBACK
    c. SAVEPOINT
4. Queries
    a. SELECT

CREATE TABLE *TABLE_NAME* (
    COULMN_DEFINITION_1,
    …
    …
);

Column definition includes:

- Name of the column
- Data type of the column along with size and precision (if required)
    - NUMBER
    - CHAR
    - VARCHAR
    - DATE
    - LONG
    - BLOB
    - CLOB
    - Ref: https://docs.oracle.com/cd/A58617_01/server.804/a58241/ch5.htm for more details
- Constraints
    - NOT NULL
    - PRIMARY KEY
    - UNIQUE
    - CHECK
- Default value (if any)

INSERT INTO *TABLE_NAME* [*coulmn_list*] VALUES (*value_list*);

UPDATE *TABLE_NAME* SET *COL1=VAL1* [, *COL2=VAL2, …*] [WHERE *CONDITION*]

DELETE FROM *TABLE_NAME* [WHERE *CONDITION*]

SELECT [DISTINCT] *column_list* | *
FROM *table_references*
[WHERE *where_condition*]
[GROUP BY *column_list* [HAVING *having_condition*]]
[ORDER BY *column_list* [ASC|DESC]]

- SQL Commands are not case sensitive
- SQL Statements can be given in one or more lines (for clarity)
- Clauses are usually placed in separate lines
- Use tabs or spaces to indent the SQL command to enhance the readability.
- The keywords cannot be split into two lines. (SEL, ECT)

Operators in SQL Select statements:

- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulus)

NULL is not a value, so we cannot use the = operator to check the presence or absence of NULL. Use the “is null” or “is not null” instead.

Comparison operators:

- =
- >
- >=
- <
- <=
- != or <>
- BETWEEN / NOT BETWEEN
- IN / NOT IN
- LIKE / NOT LIKE
- IS NULL / IS NOT NULL

Logical operators:

- AND
- OR
- NOT

Sorting options:

- ORDER BY
    - ASC (default)
    - DESC
- Can specify multiple columns

Aggregate functions:

- Produce a single output for multiple rows of values in column
- MAX, MIN, AVG, SUM, COUNT
- Usually used with GROUP BY clause
- Can never be used in a WHERE clause
- Can be used in the HAVING clause to filter the result of grouping

DATABASE VIEWS

- A user's perception of data
- It is an object that can almost be used like a table, but does not contain any data
- A View contains a query (usually a complex one), that is fired when you execute a SELECT statement on the view.
    - CUSTOMER_ORDERS → a very complex query involving joining of 4 to 5 tables, group by and other clauses
    - SELECT * FROM CUSTOMER_ORDERS
- Can use all other SQL clauses like WHERE, GROUP BY etc
- Can also be joined with other views and/or tables

DATABASE INDEX

- It is a data structure that can help increase the performance of data retrieval
- Consider creating an index on a column of a table only if
    - the table has enormous amount of data
    - the column is frequently used in queries
- Primary key columns are automatically indexed
- Creating more indexes reduces the performance of DML statements, since the index data structure has to be rebuilt with new data (insert/update/delete)ed.