AI VIET NAM
@aivietnam.edu.vn

SQL in Data Analysis

(ERD & Database Normalization)

Data & Code

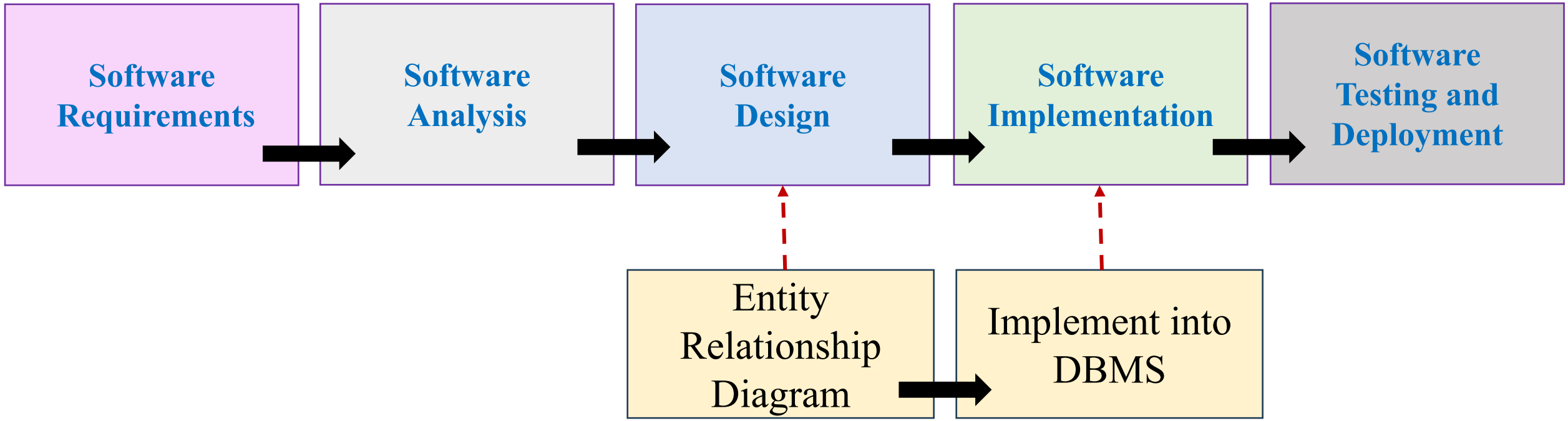
Vinh Dinh Nguyen
PhD in Computer Science

Outline

- Introduction to Entity Relationship Diagram
- Introduction to Database Normalization
- Advanced SQL Queries
- Kahoot Quiz
- Summary

ENTITY RELATIONSIP DIAGRAM

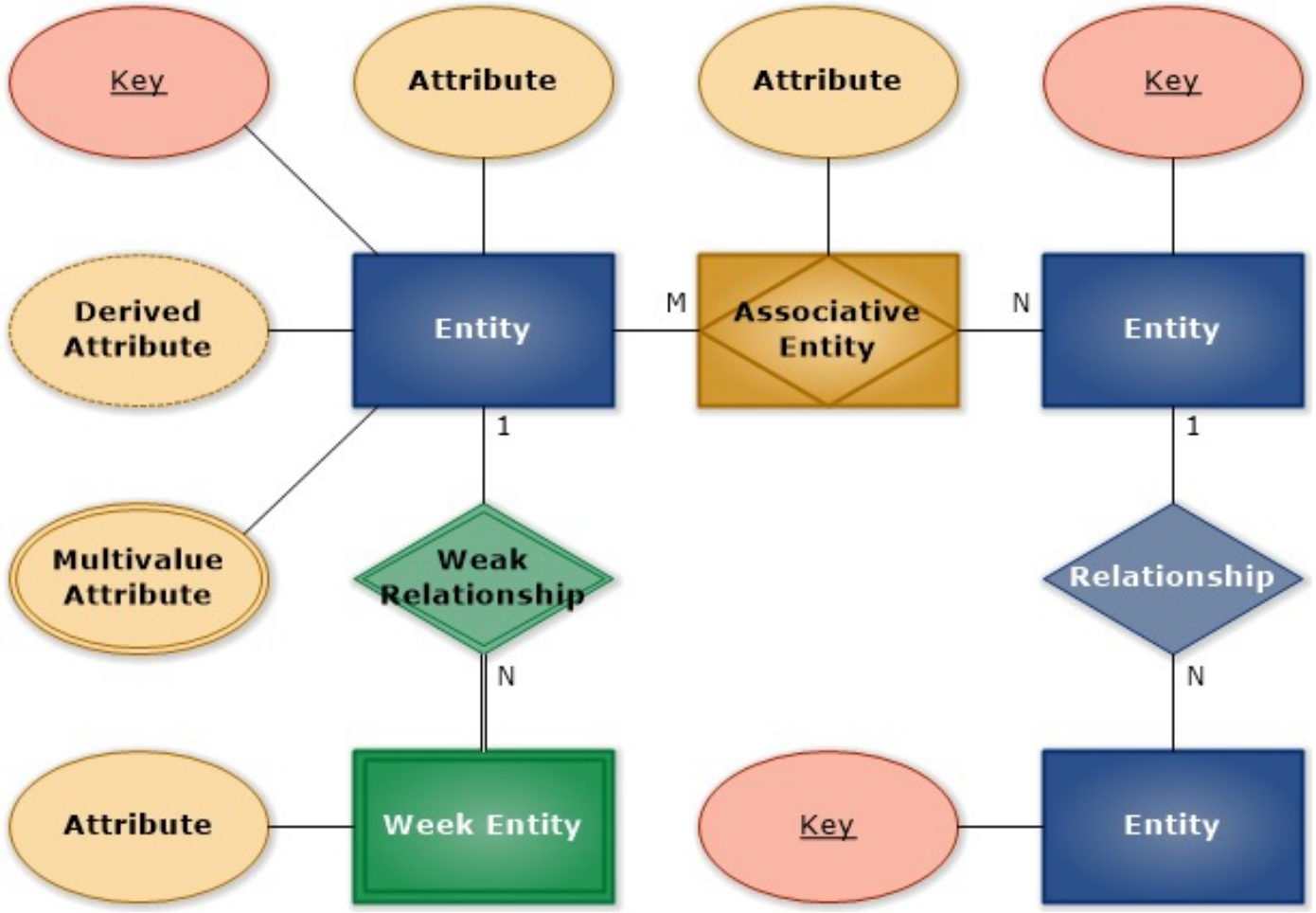
ER DIAGRAM



An Entity Relationship Diagram (ERD) is a visual representation of **different entities within a system and how they relate to each other**. It is a tool used to design and model relational databases, and shows the logical structure of the database

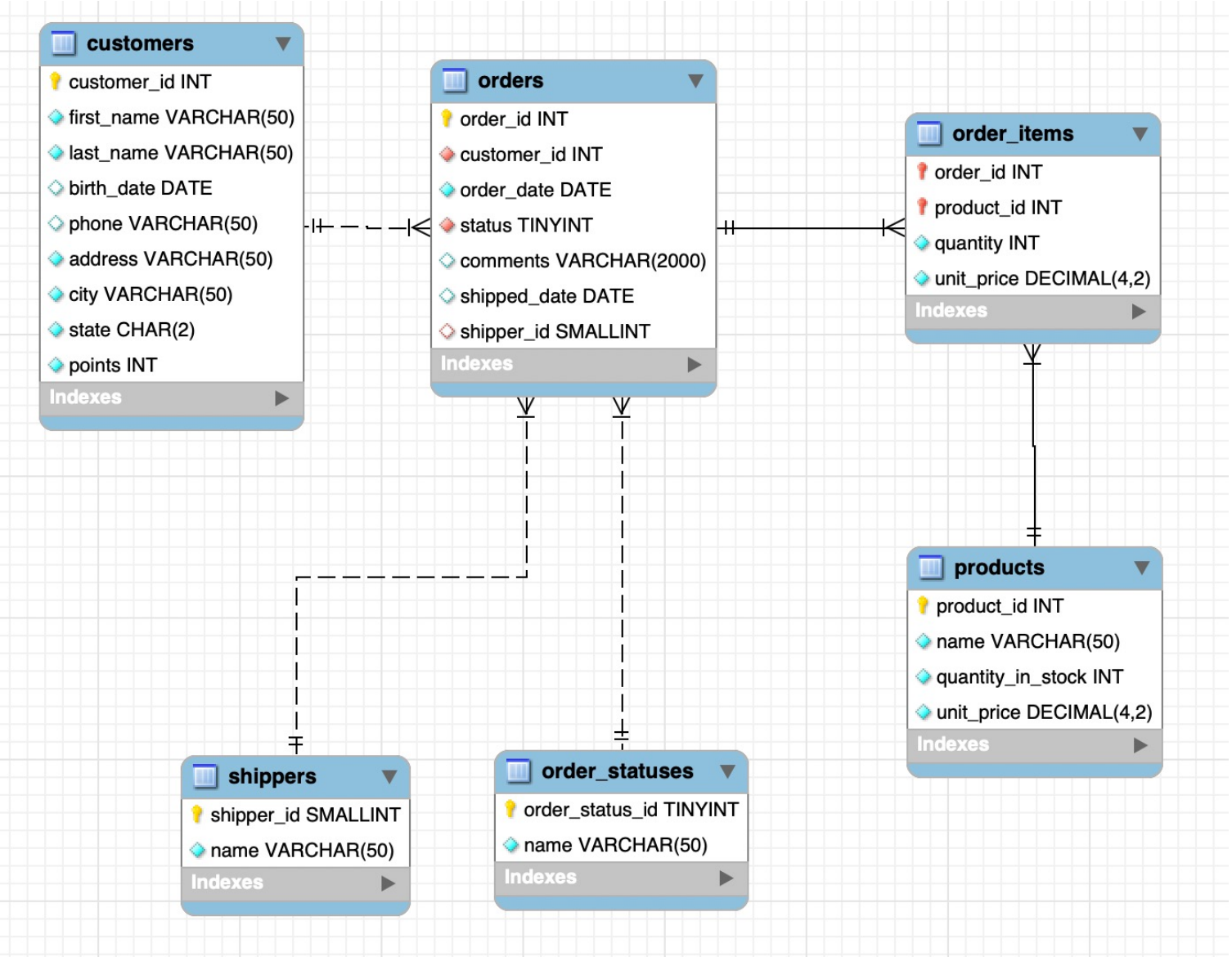
ENTITY RELATIONSHIP DIAGRAM

CHEN'S NOTATION



ENTITY RELATIONSHIP DIAGRAM

CROW'S FOOT NOTATION



Chen's Notation

Entities are represented in ER diagrams by a rectangle and named using singular nouns.

An entity like order item is a good example for this.

Weak Entity

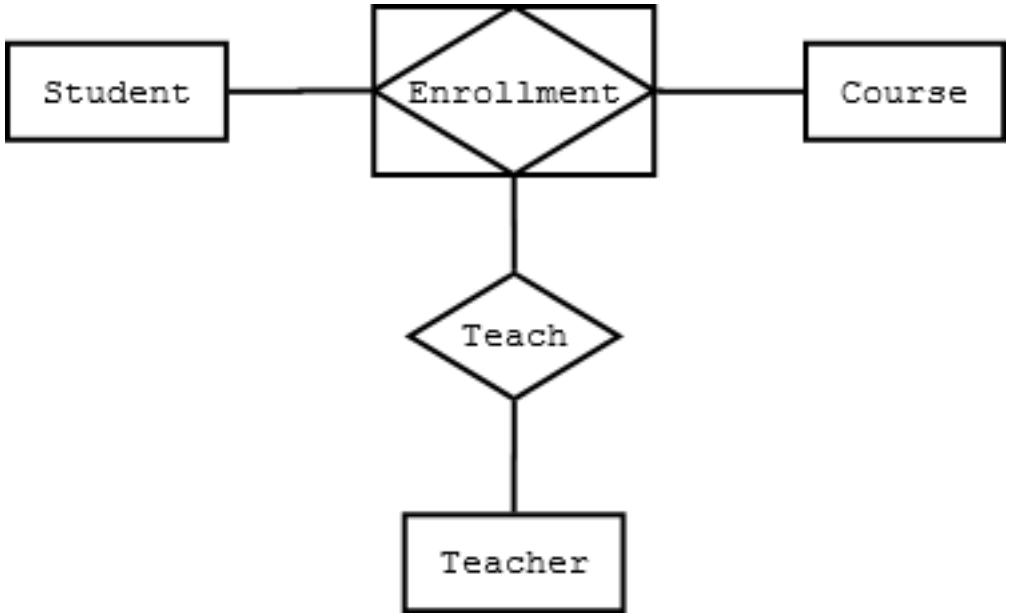
A weak entity is an entity that depends on the existence of another entity.
An entity that cannot be uniquely identified by its attributes alone.

An entity like order item is a good example for this. The order item will be meaningless without an order so it depends on the existence of the order.

Chen's notation

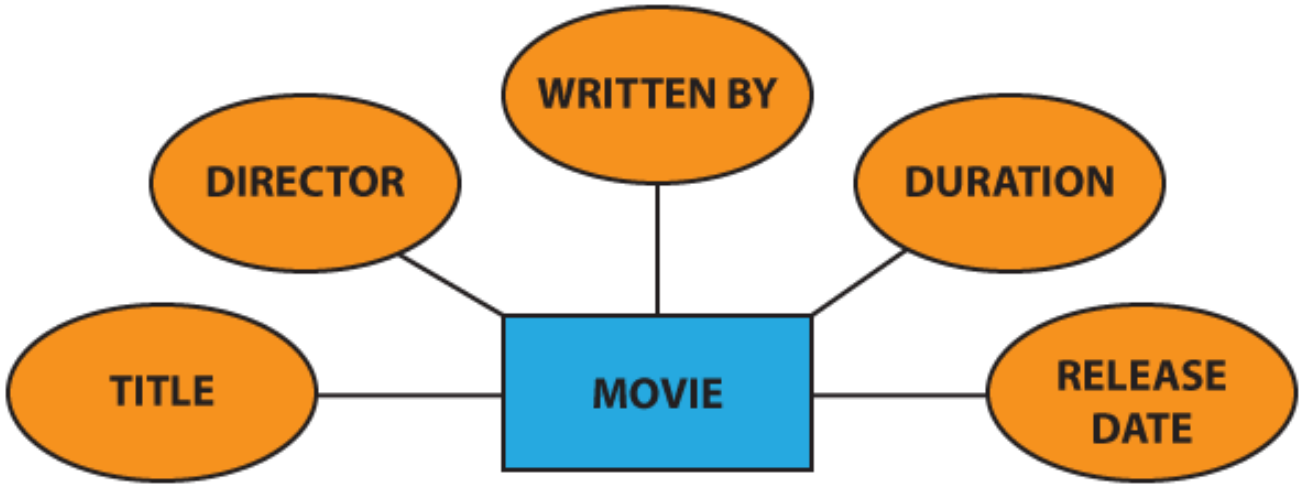
Associative Entity

An entity used in a many-to-many relationship (represents an extra table). All relationships for the associative entity should be many



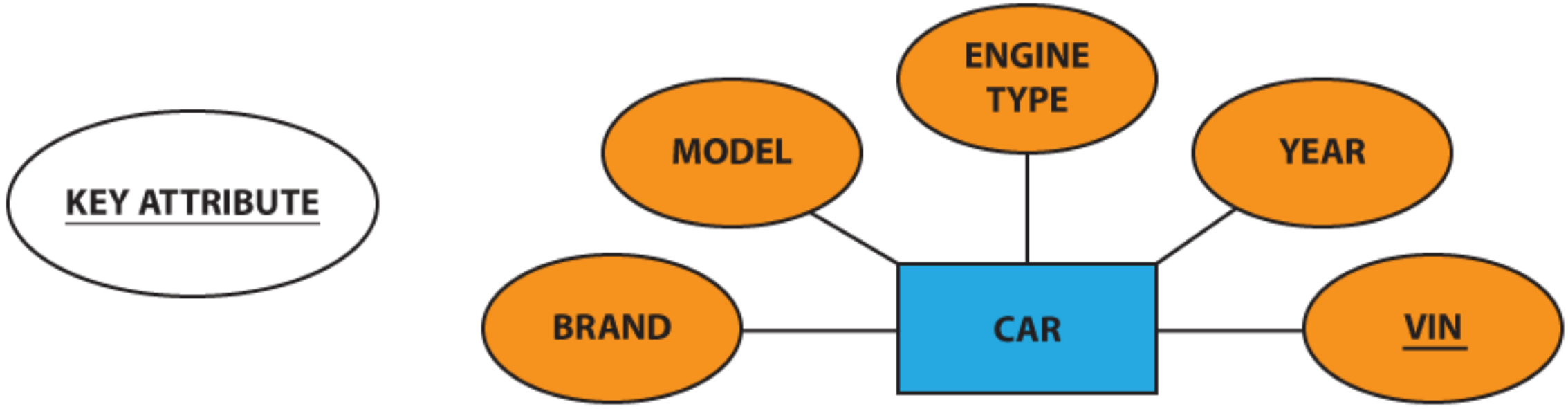
Attribute

In the Chen notation, each attribute is represented by an **oval** containing attribute's name:



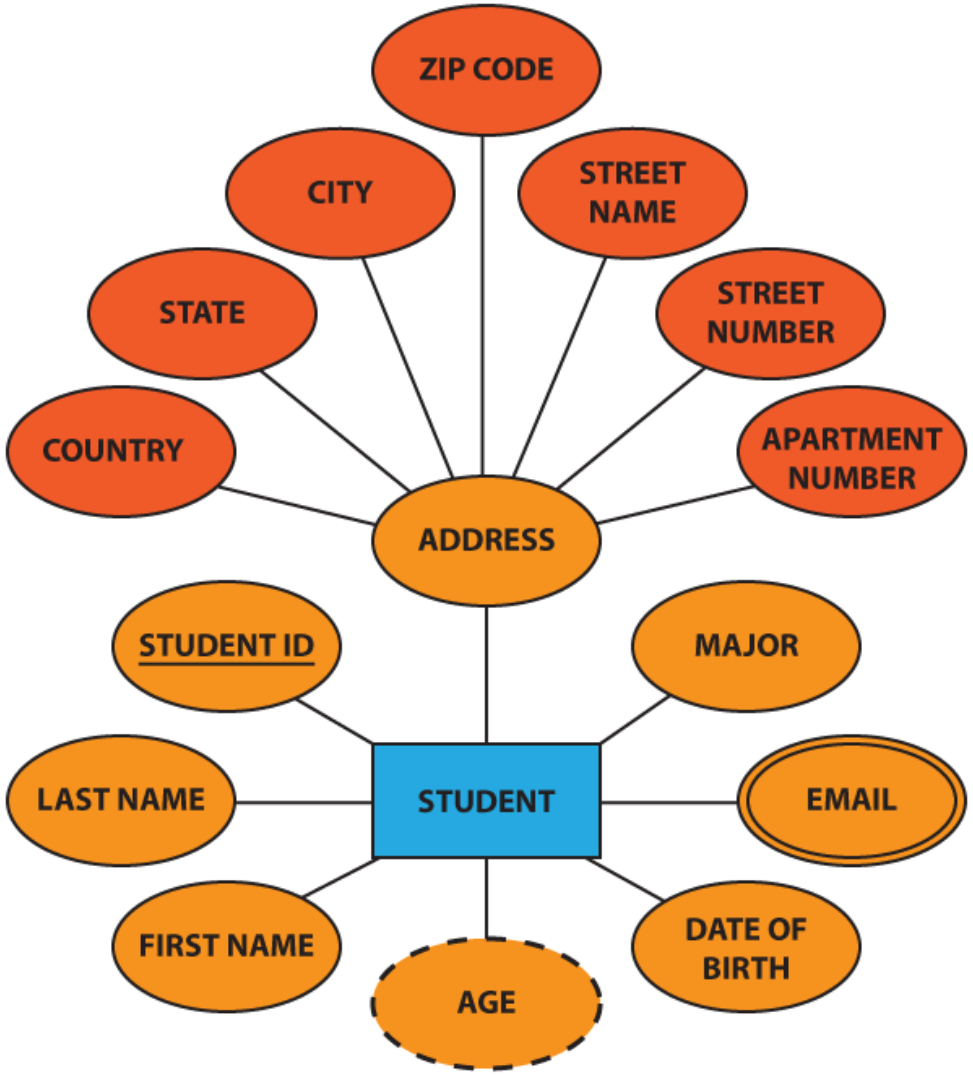
Key Attribute

key attribute – an attribute that uniquely identifies a particular entity. The name of a key attribute is underscored:

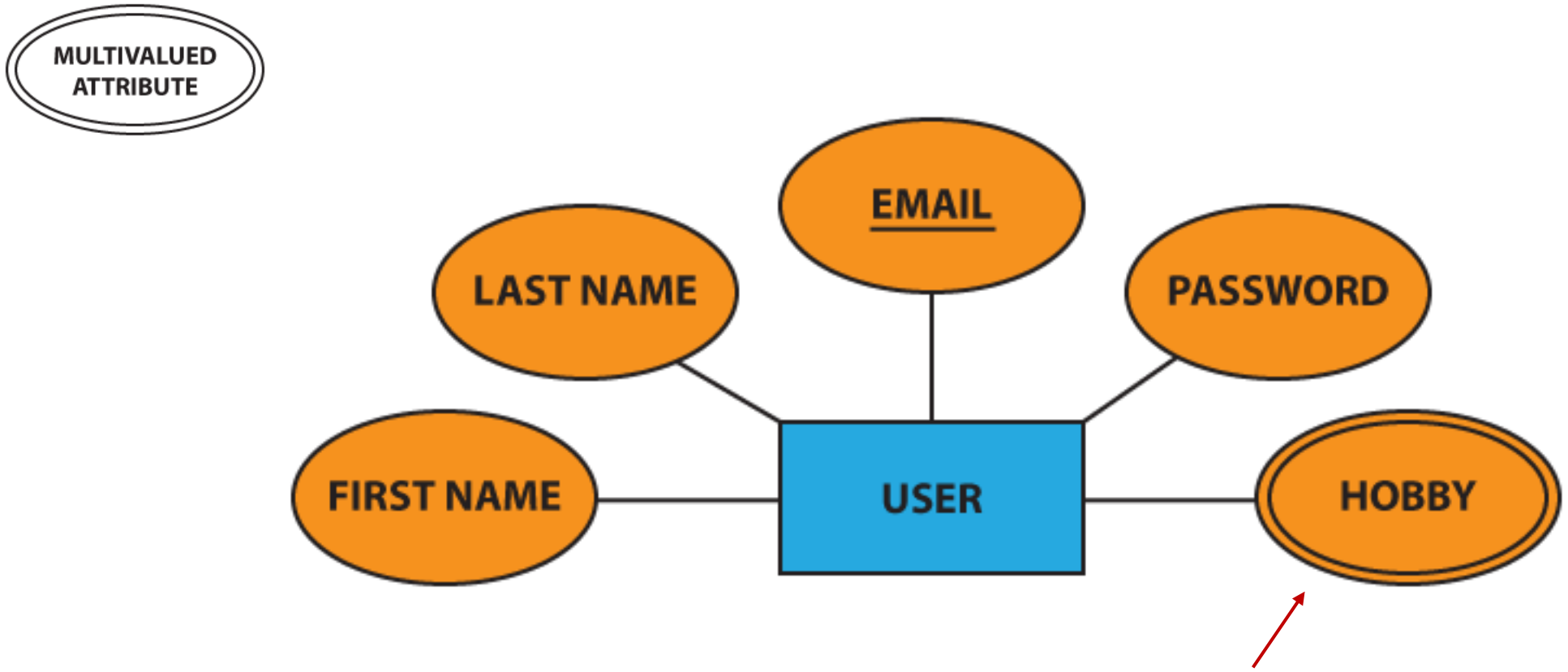


Composite Attributes

Some attributes can be further subdivided into smaller parts. For example, the attribute “address” can be subdivided into street name, street number, apartment number, city, state, zip code, and country.

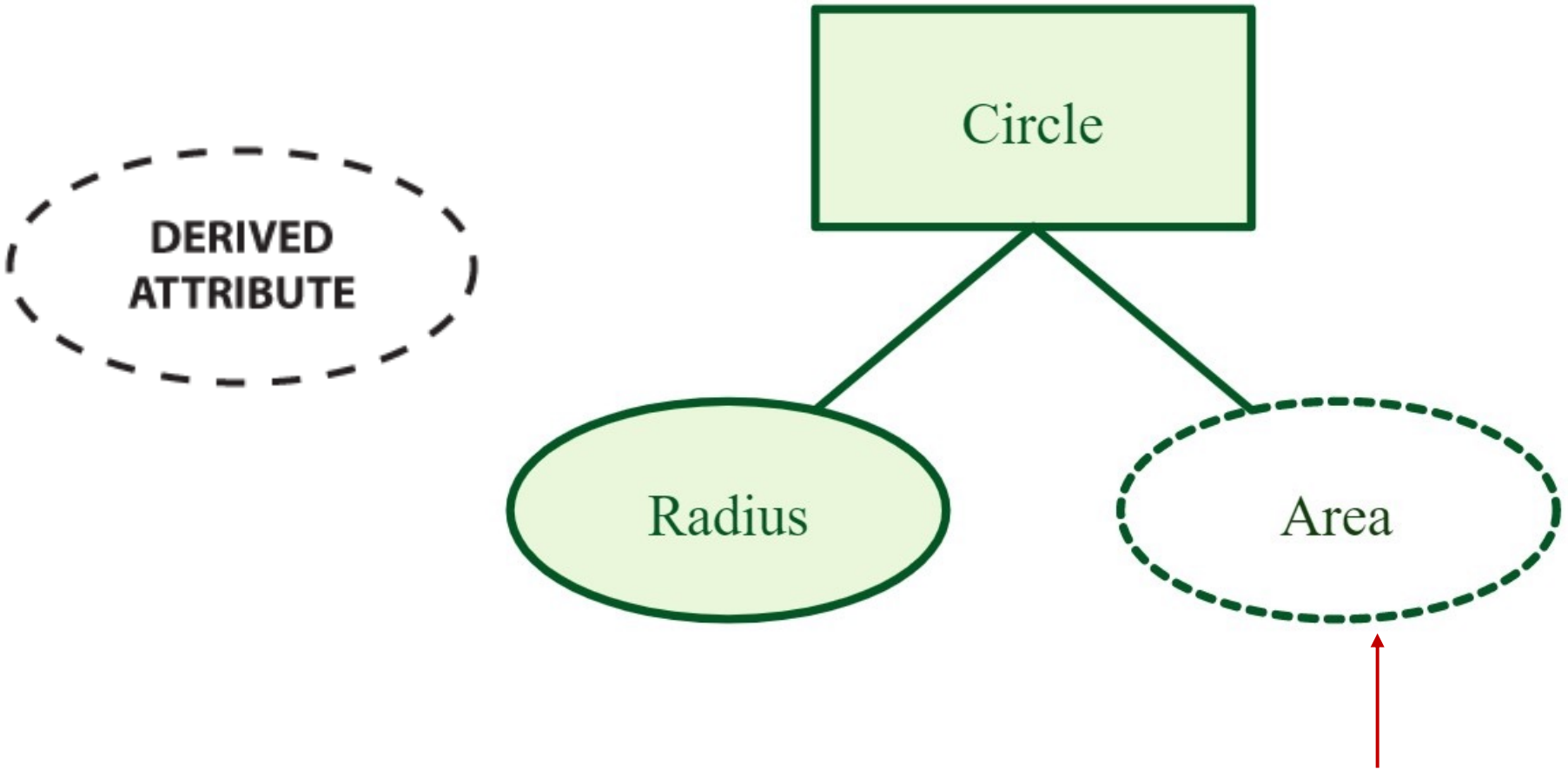


Multivalued Attribute



If an attribute can have more than one value it is called a multi valued attribute

Derived Attribute



An attribute based on another attribute. This is found rarely in ER diagrams. For example, for a circle, the area can be derived from the radius.

Relationship

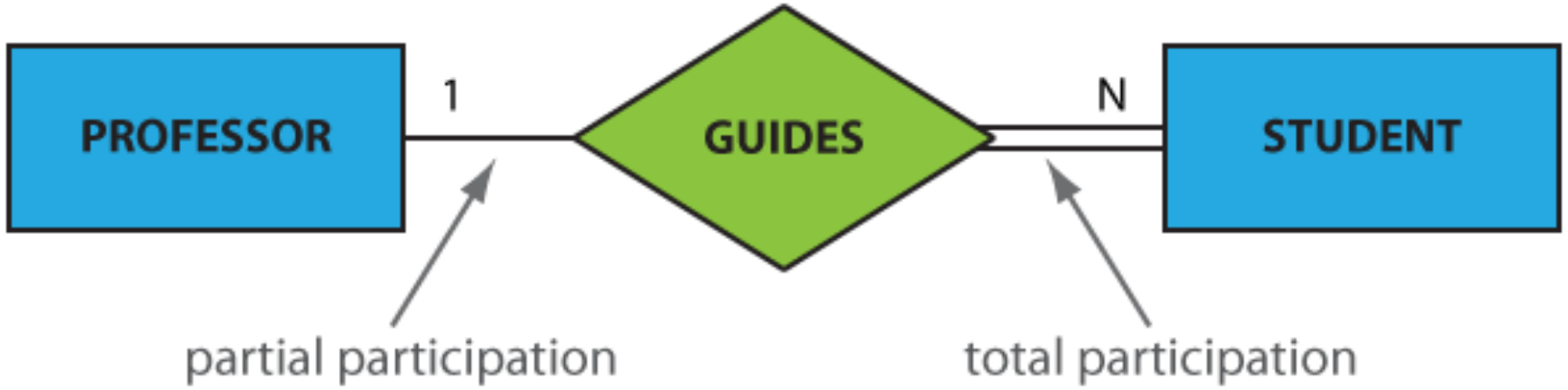
one-to-one (1:1)

one-to-many (1:N)

many-to-one (N:1)

many-to-many (M:N)

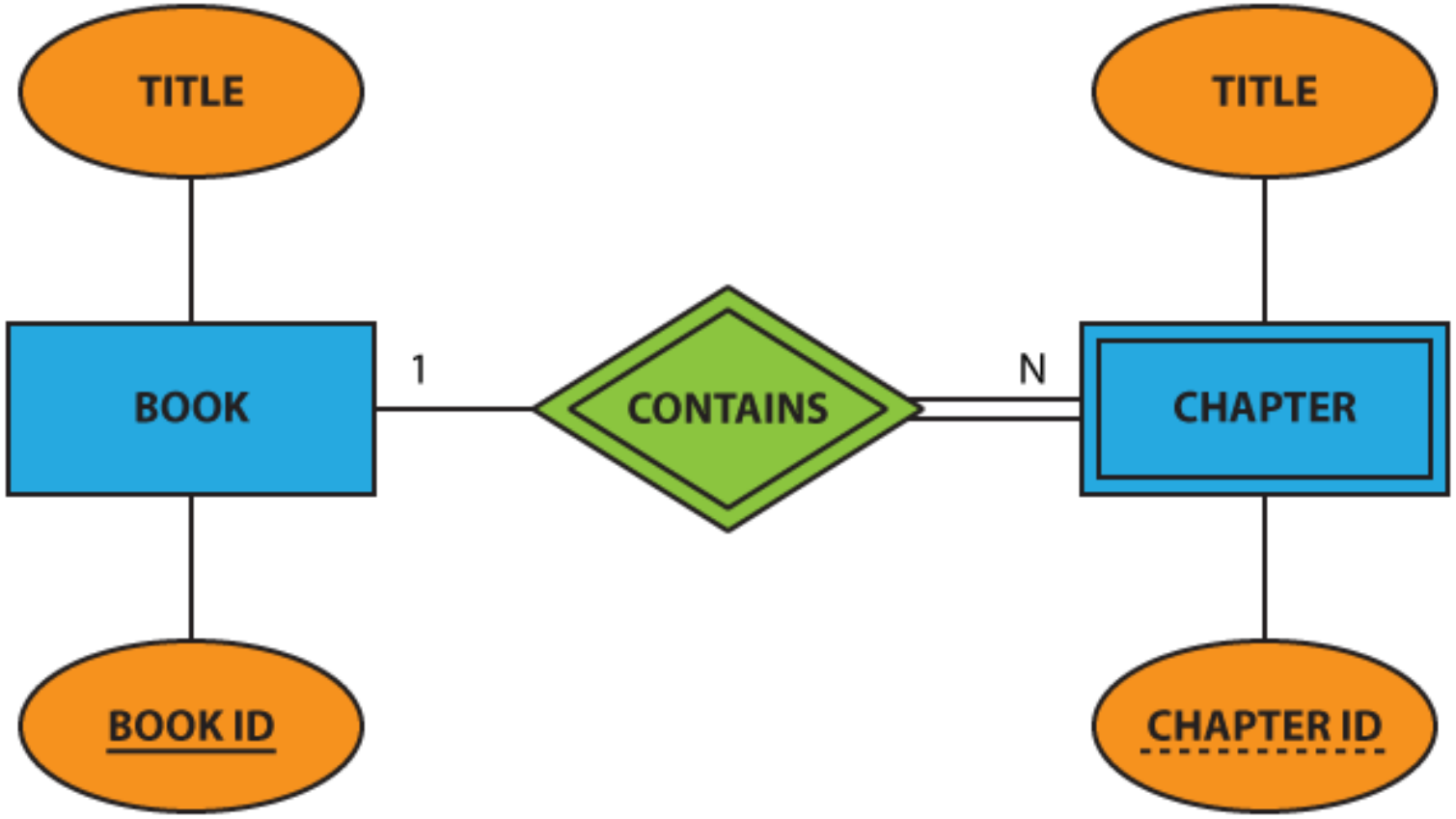
Participation Constraints



•**Total participation** means that every entity in the set is involved in the relationship

•**Partial participation** means that not all entities in the set are involved in the relationship

Participation Constraints

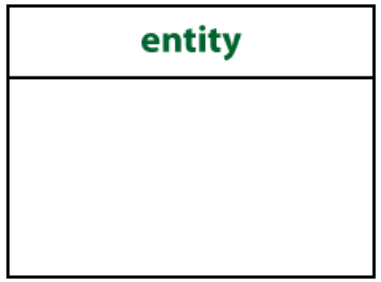


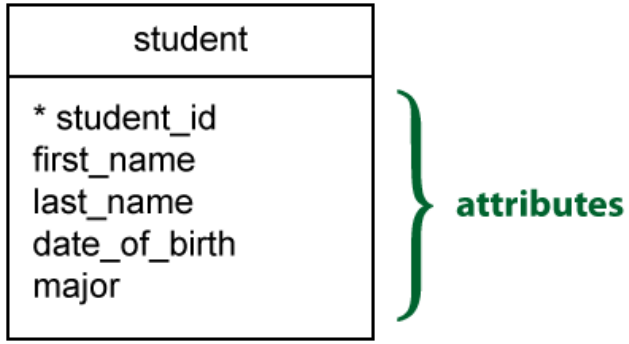
•**Total participation** means that every entity in the set is involved in the relationship

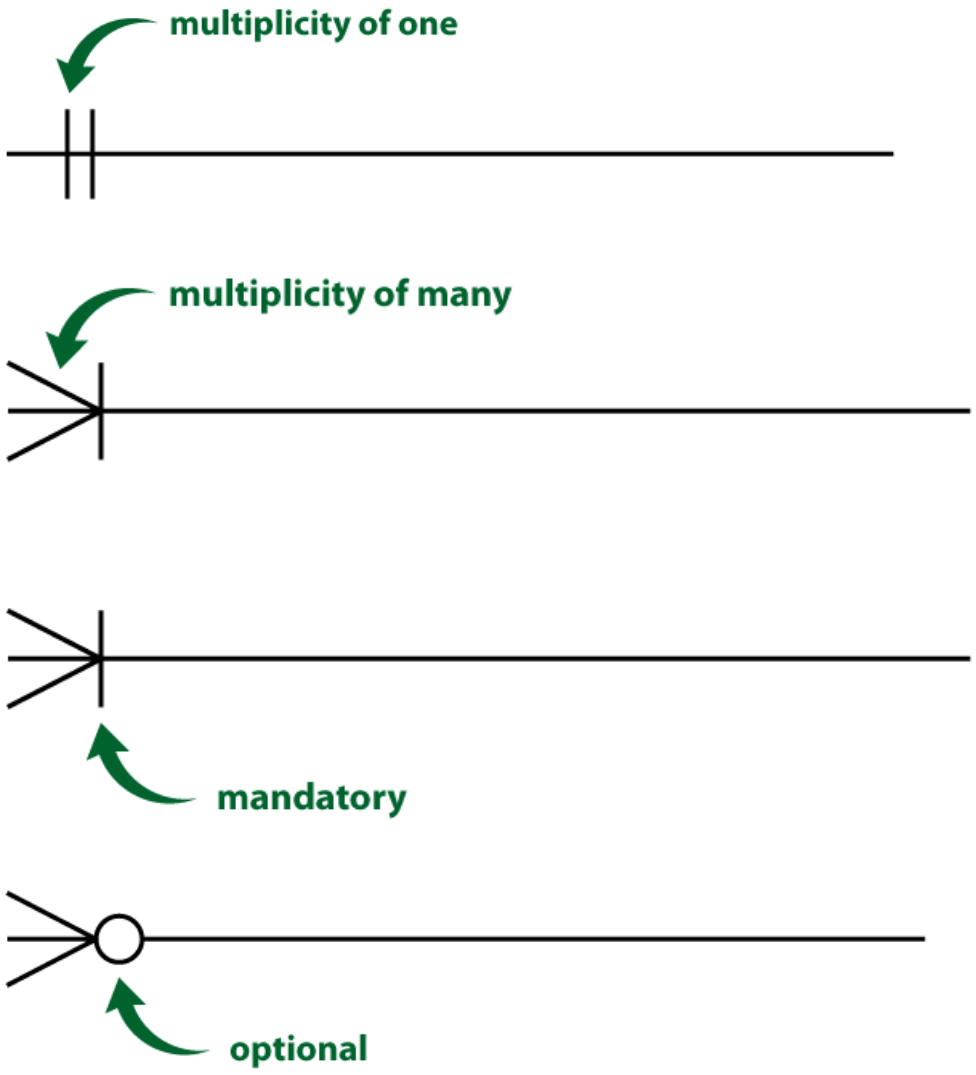
•**Partial participation** means that not all entities in the set are involved in the relationship

Crow's Foot Notation

Entities

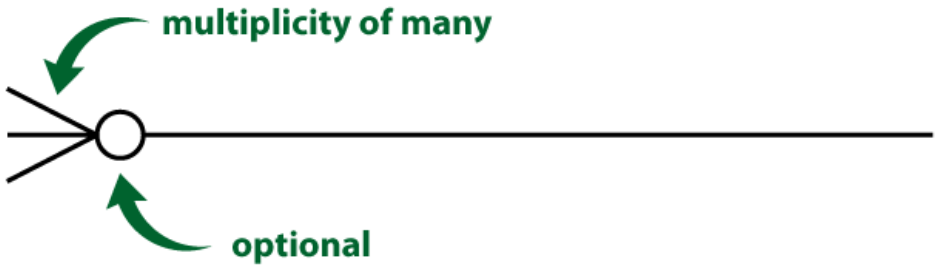


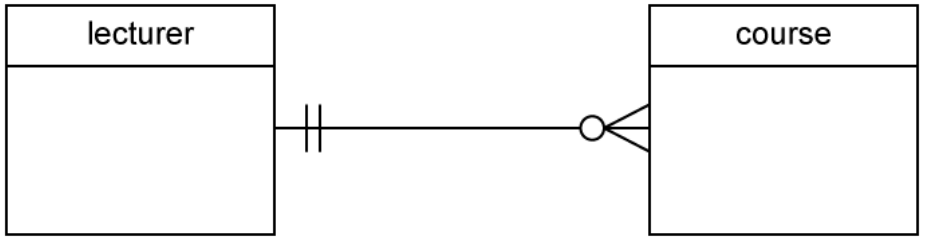




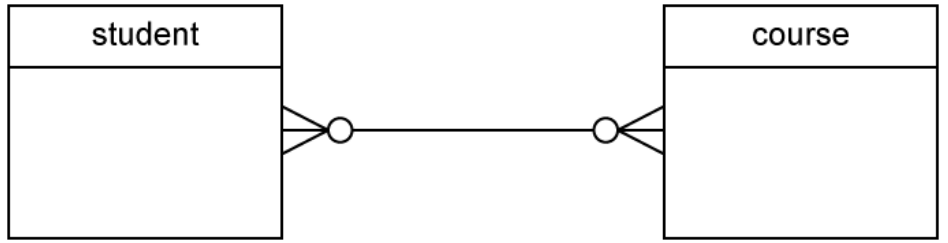
Crow's Foot Notation

zero or many





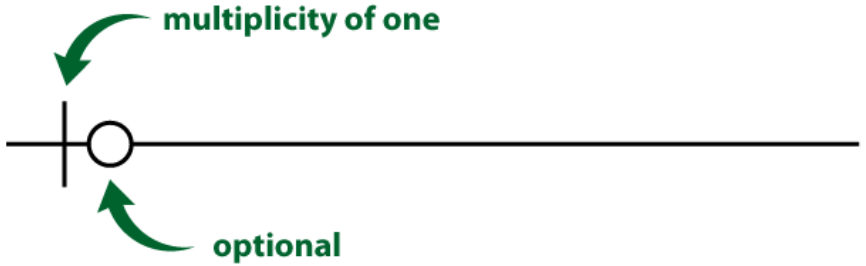
one or many

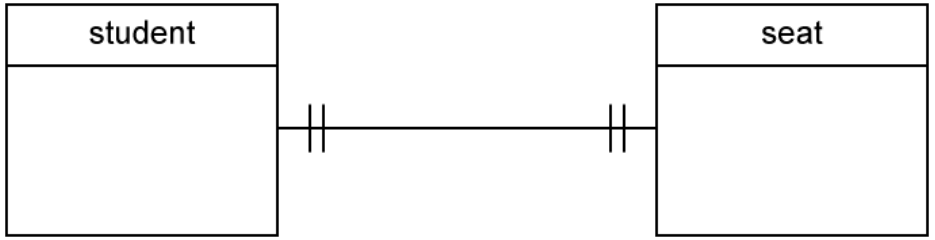


Crow's Foot Notation

one and only one

zero or one





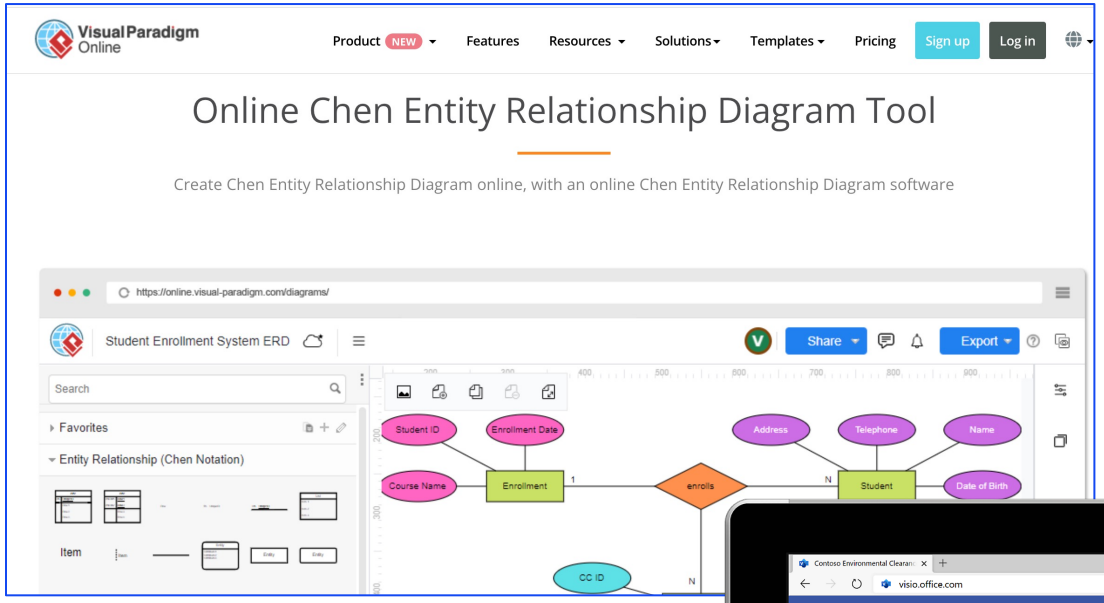
How to Draw ER Diagram

1.Identify all the entities in the system. An entity should appear only once in a particular diagram. Create rectangles for all entities and name them properly.

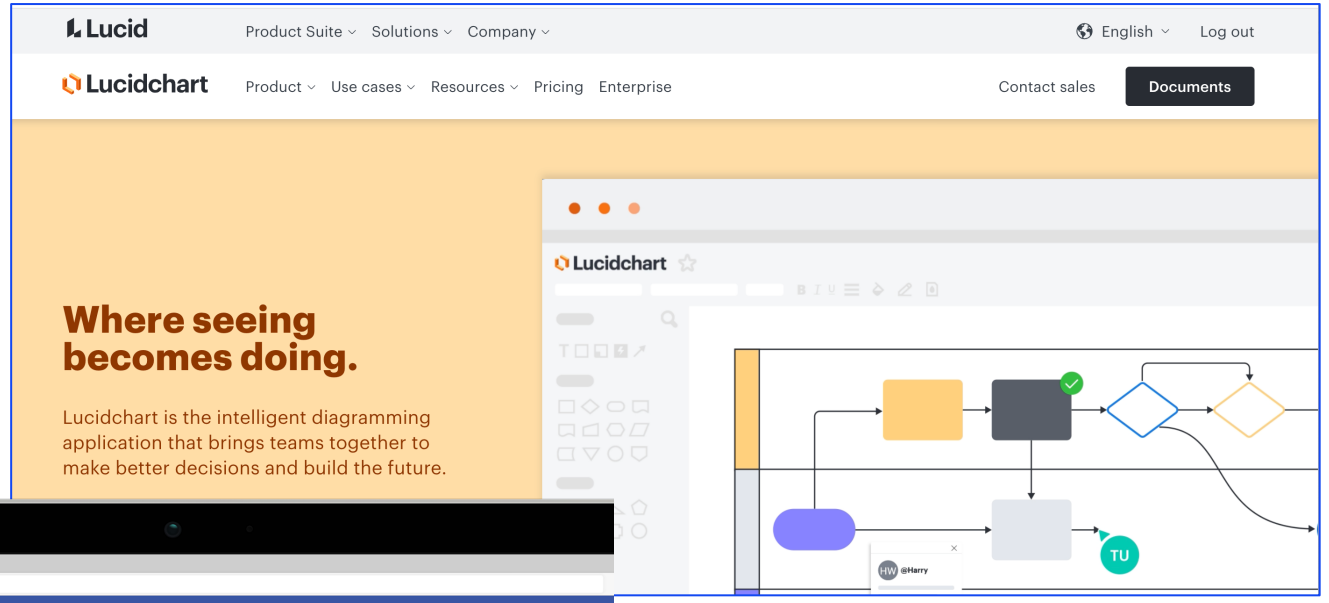
2. Identify relationships between entities. Connect them using a line and add a diamond in the middle describing the relationship.

3. Add attributes for entities. Give meaningful attribute names so they can be understood easily.

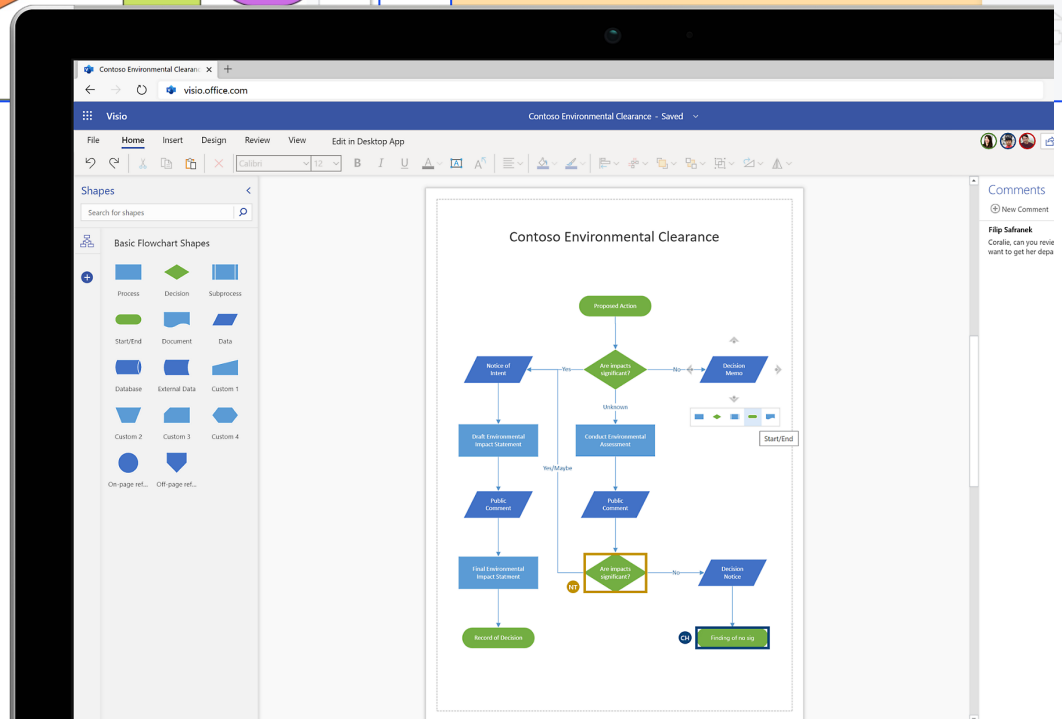
Tools



Visual Paradigm



Lucidchart



Microsoft Visio

ER DIAGRAM

Company Data Requirements

- The company want to store it product. Each product has a name, quantity in stock, and unit price.
- The company sell their products to customers. Each customer has a name, birth day, phone, address, city, state, and loyalty point.
- Each order placed by the customer need to have order date, status, comments, shipped date, and info of the shipper for that order. Also the company need to keep track of the quantity and unit price of the products sold in that order.

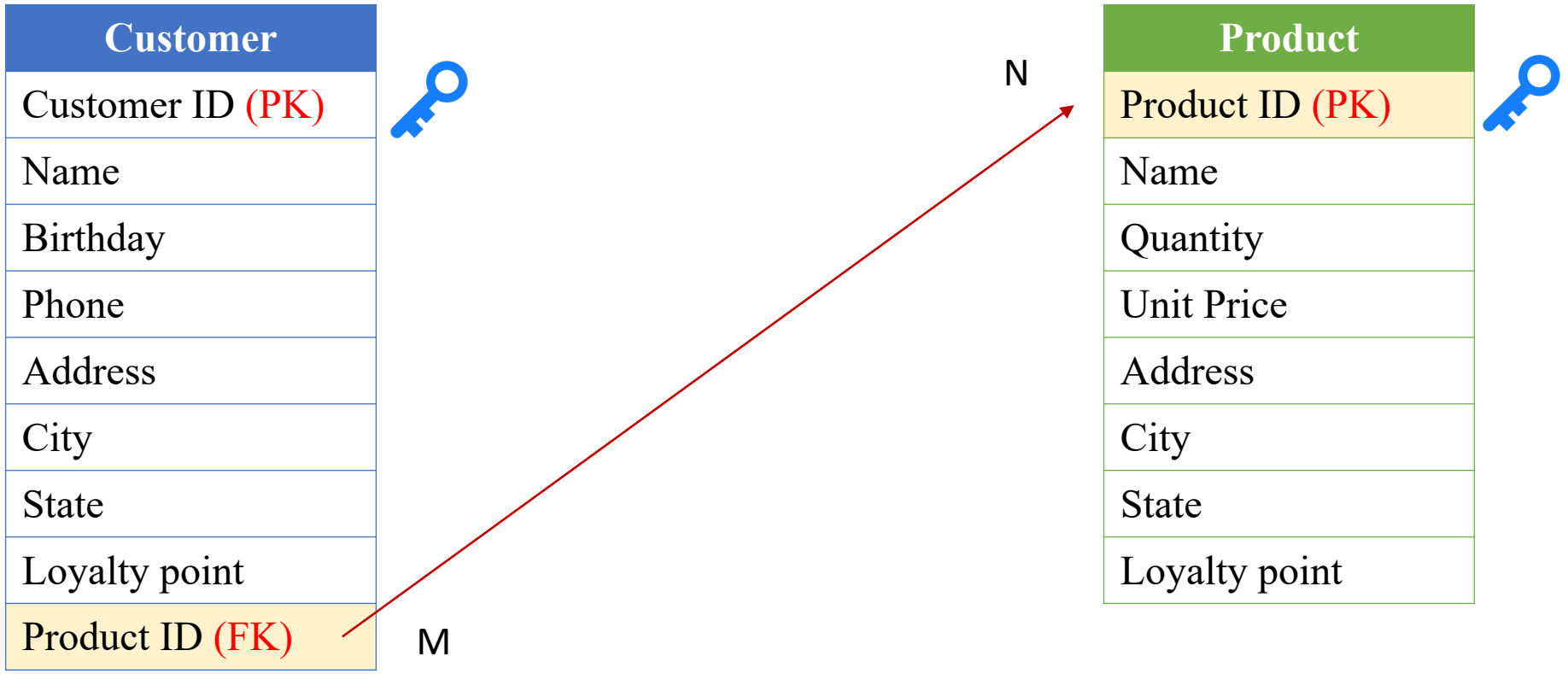
ER DIAGRAM

Customer
Customer ID (PK)
Name
Birthday
Phone
Address
City
State
Loyalty point

Product
Product ID (PK)
Name
Quantity
Unit Price
Address
City
State
Loyalty point

What is the relationship between Customer and Product?

ER DIAGRAM



What's the problem?

ER DIAGRAM

Customer ID	Name	Birday	Phone	Product ID
001	Vinh	2005	010111222	001
001	Vinh	2005	010111222	002
001	Vinh	2005	010111222	003

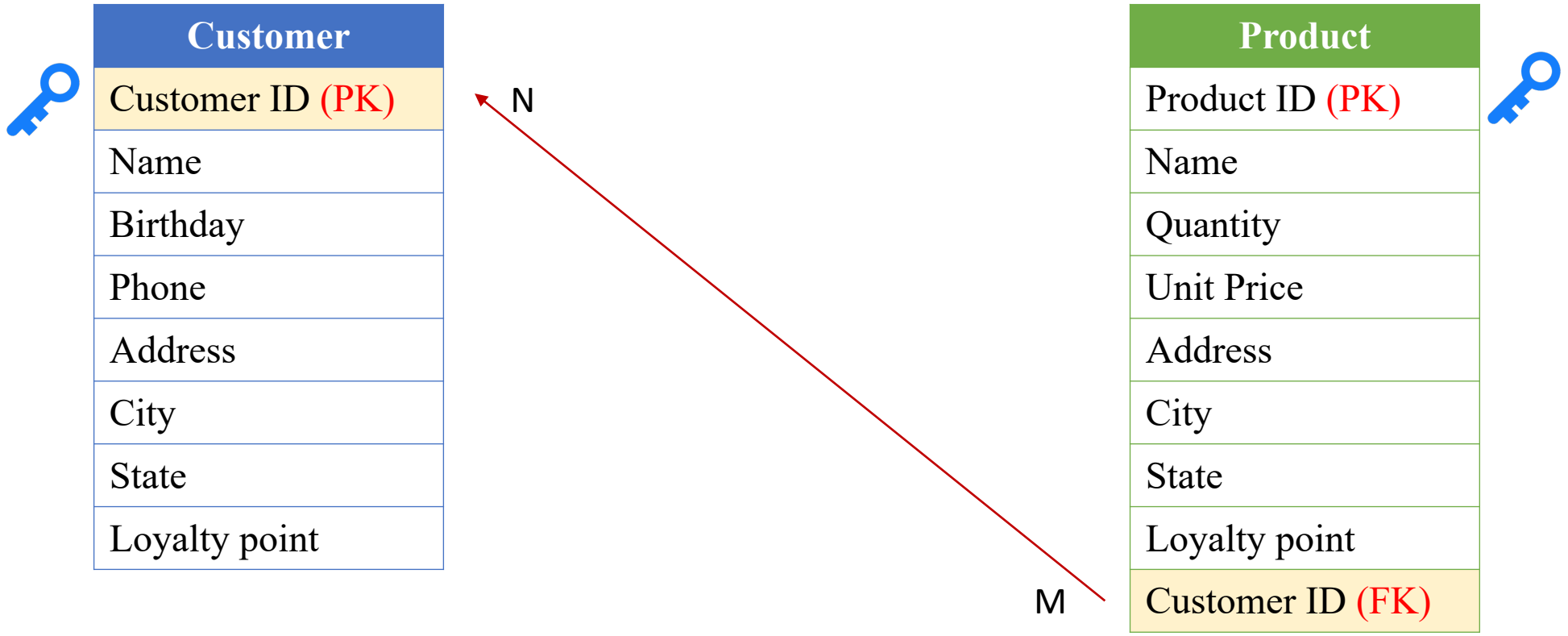
Foreign key

Product ID	Name	Quantity	Price
001	Smart Phone	20	200000
002	TV	10	1000000
003	Watch	15	500000

Primary Key

What's the problem?

ER DIAGRAM



What's the problem?

ER DIAGRAM

Customer ID	Name	Birday	Phone
001	Vinh	2005	0978465787
002	An	2006	0978465786
003	Loc	2007	0978465788

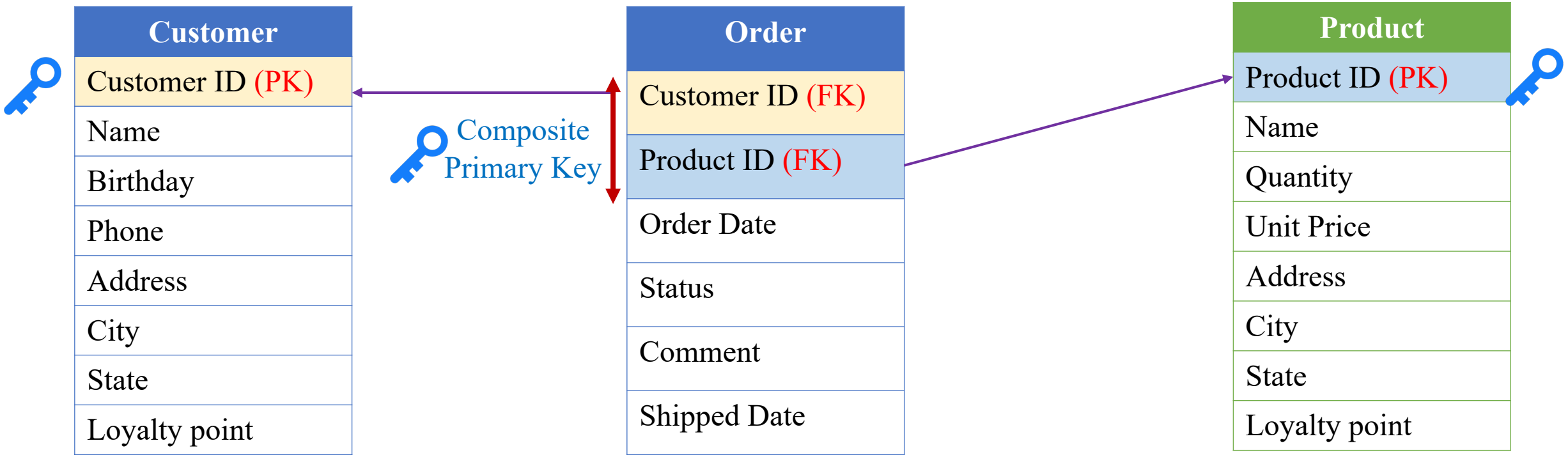
Primary key

Product ID	Name	Quantity	Price	Customer ID
001	Smart Phone	20	200000	001
001	Smart Phone	20	200000	002
001	Smart Phone	20	200000	003

Foreign key

What's the problem?

ER DIAGRAM


Customer
Customer ID (PK)
Name
Birthday
Phone
Address
City
State
Loyalty point
Composite Primary Key
Order
Customer ID (FK)
Product ID (FK)
Order Date
Status
Comment
Shipped Date
Product
Product ID (PK)
Name
Quantity
Unit Price
Address
City
State
Loyalty point

ER DIAGRAM

Customer ID	Name	Birday	Phone
001	Vinh	2005	0978465787
002	An	2006	0978465786
003	Loc	2007	0978465788

Primary key

Product ID	Name	Quantity	Price
001	Smart Phone	20	200000
002	TV	10	1000000
003	Watch	15	500000

Primary key

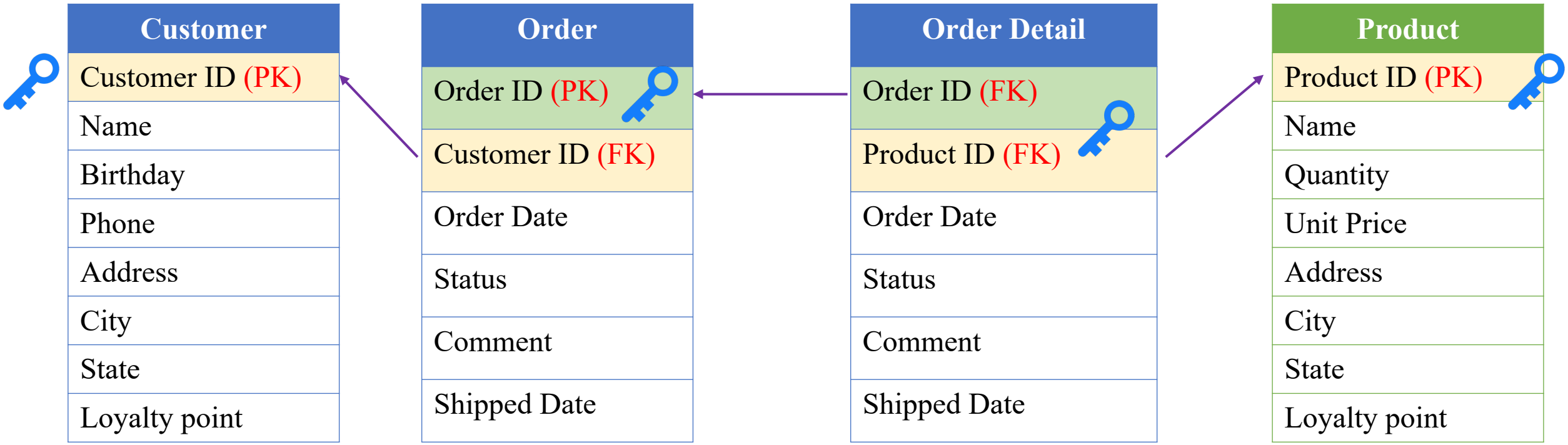
Customer ID	Product ID	Order Date	Status	Comment	…
001	001	12-06-2025	Thanh Toán	Giao giờ hành chính	
001	002	12-06-2025	Thanh Toán	Giao giờ hành chính	
002	003	10-06-2025	Đang chuyển hàng	Giao ngày lễ	

Foreign key Foreign key

Composite key

What's the problem?

ER DIAGRAM


Customer
Customer ID (PK)
Name
Birthday
Phone
Address
City
State
Loyalty point
Order
Order ID (PK)
Customer ID (FK)
Order Date
Status
Comment
Shipped Date
Order Detail
Order ID (FK)
Product ID (FK)
Order Date
Status
Comment
Shipped Date
Product
Product ID (PK)
Name
Quantity
Unit Price
Address
City
State
Loyalty point

ER DIAGRAM

Customer ID	Name	Birday	Phone
001	Vinh	2005	0978465787
002	An	2006	0978465786
003	Loc	2007	0978465788

Primary key

Product ID	Name	Quantity	Price
001	Smart Phone	20	200000
002	TV	10	1000000
003	Watch	15	500000

Primary key

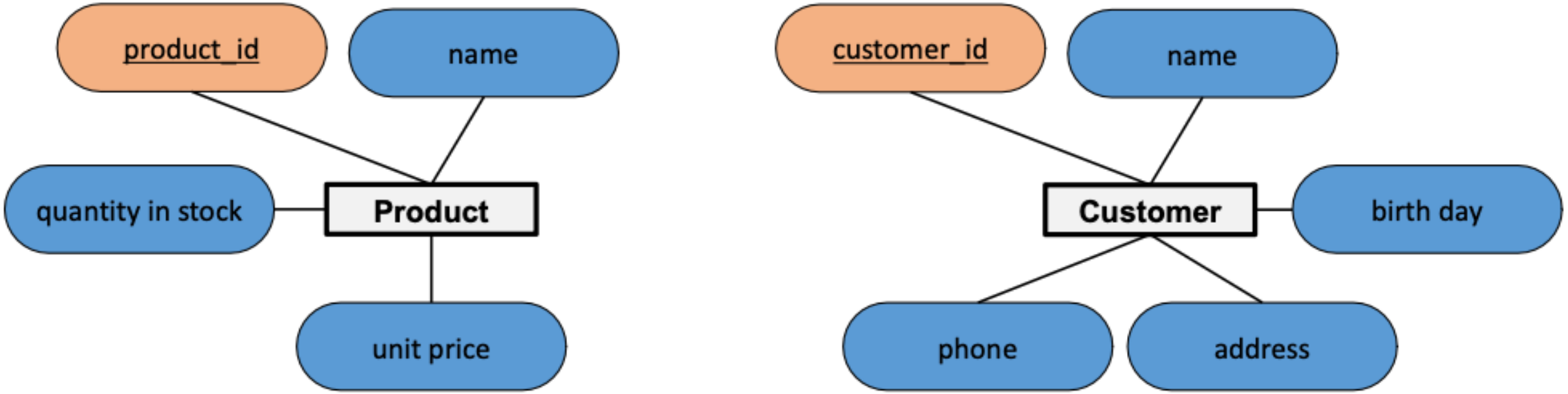
Order ID	Customer ID	Status	Comment
001	001	Thanh Toán	Giao giờ hành chính
002	002	Chưa Thanh Toán	Giao cuối tuần
003	003	Đang chuyển hàng	Giao ngày lễ

Primary key

Composite key

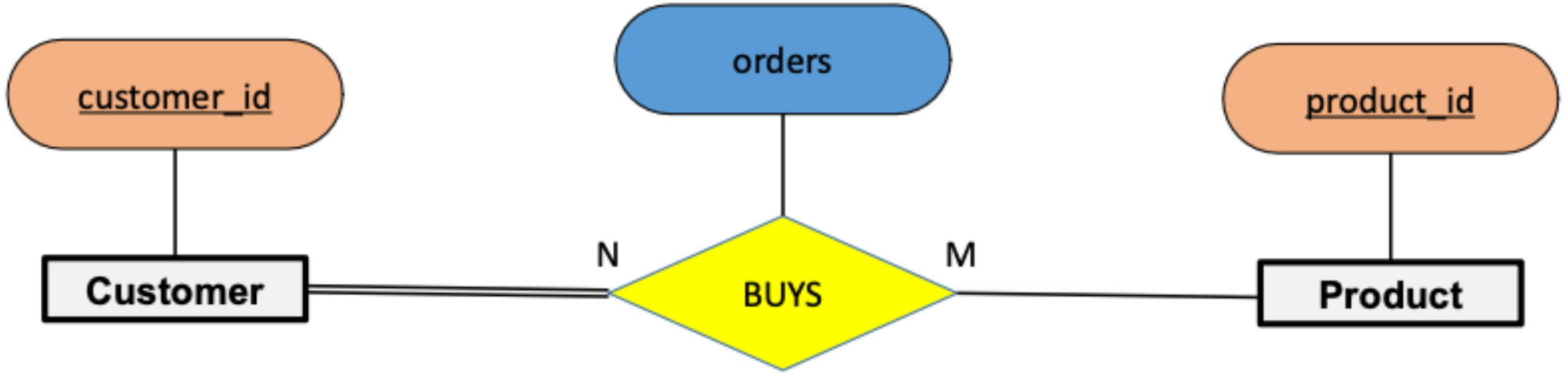
Order ID	Product ID	Quantity	Order Date
001	001	5	12-06-2025
001	002	6	13-06-2025
002	003	4	14-06-2025

ER DIAGRAM



The customer now become it own entity

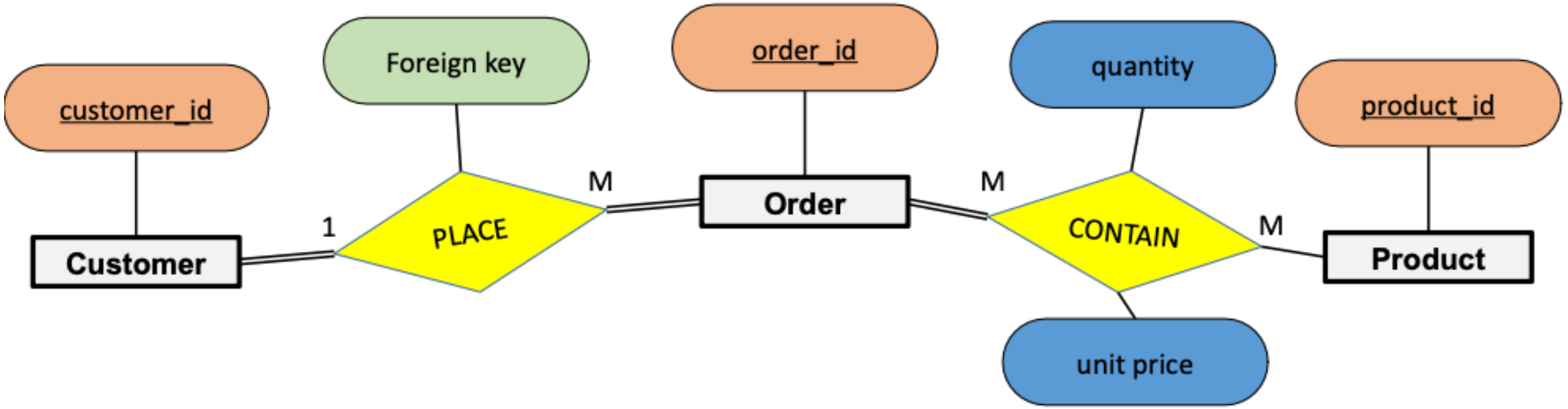
ER DIAGRAM



Which Entity will hold the orders attribute?

In a many-to-many relationship which ever entity hold the attribute will create Multi-valued attribute.

ER DIAGRAM

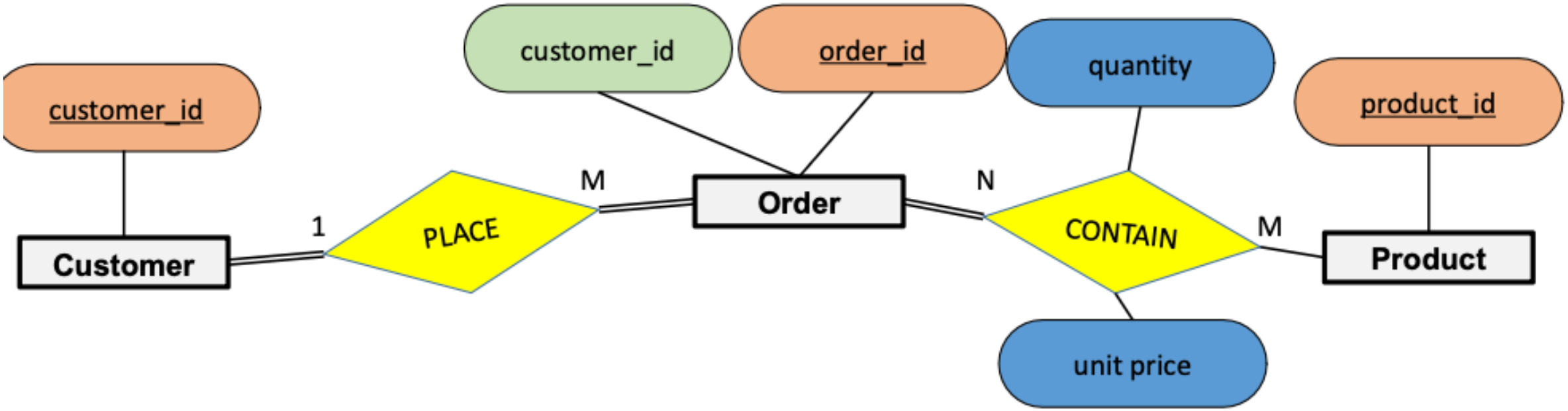


Which Entity will hold the foreign key attribute?

In one-to-many relationship the 'many' entity will hold the foreign key so as not to create Multi-valued attribute.

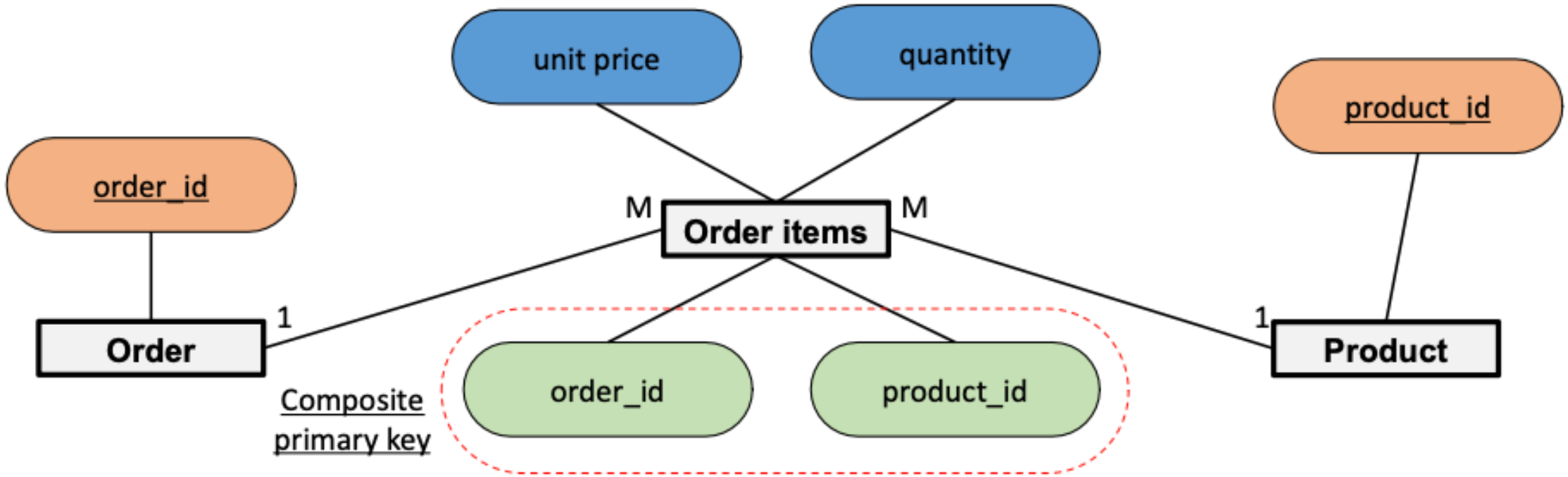
ER DIAGRAM

Foreign key(s) – Is an attribute used to refer a relation to a different entity



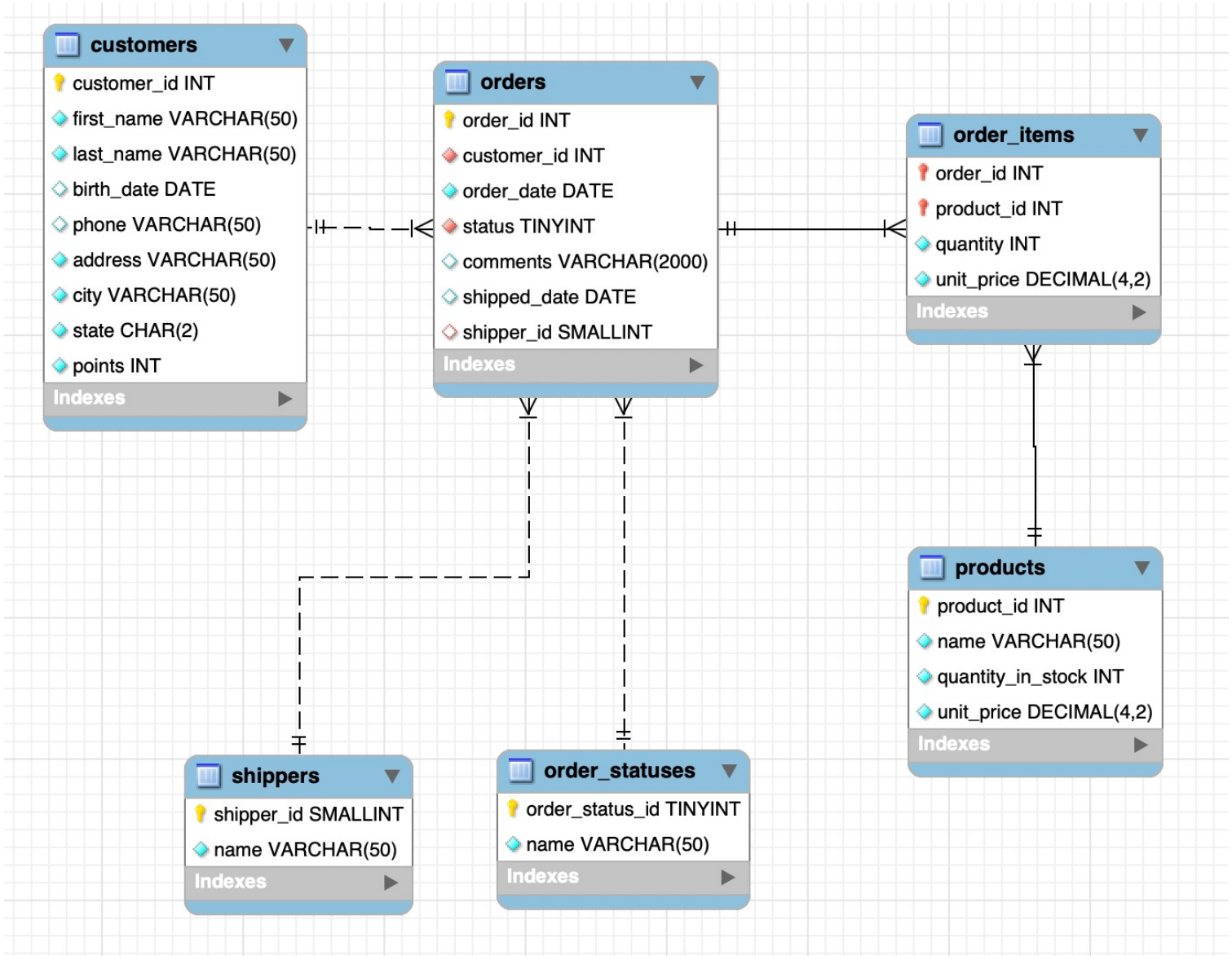
We have the same problem with quantity and unit price attributes that come from a many-to-many relationship.

ER DIAGRAM



We can resolve many-to-many relationship by using composite primary key

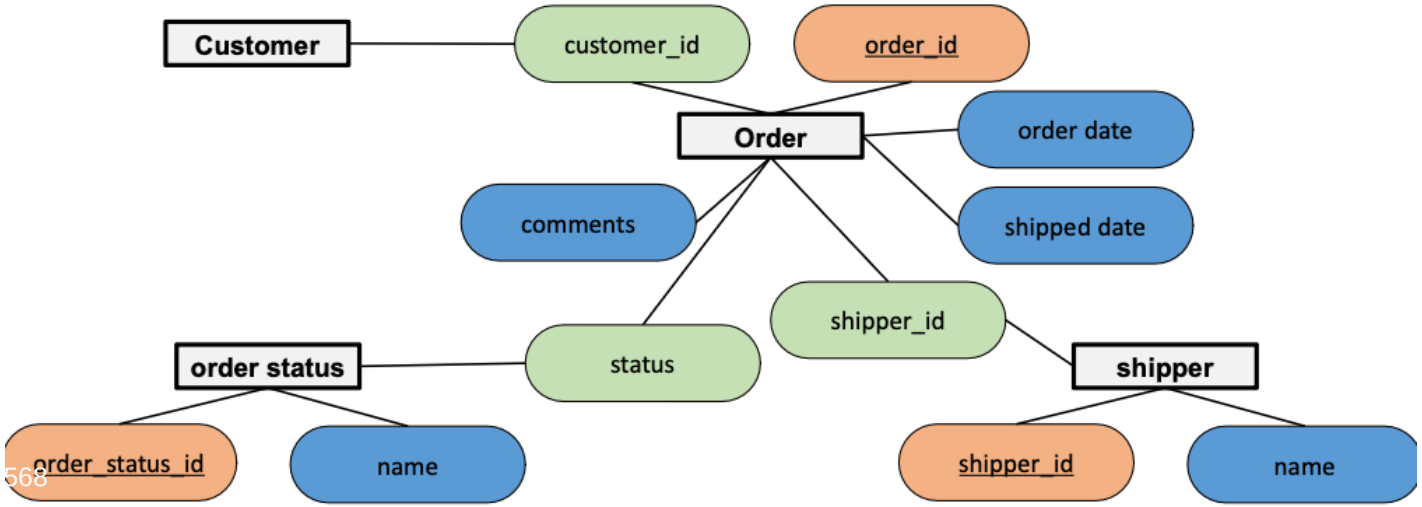
Crow's Foot ERD



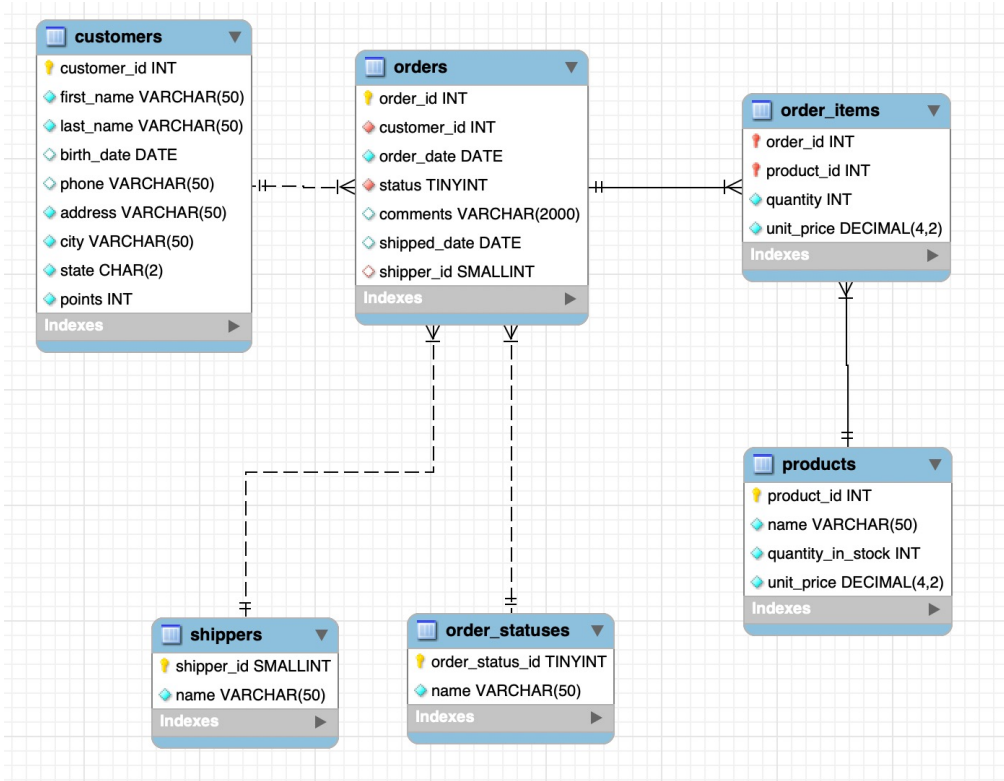
IMPLEMENT

ERD

IMPLEMENT ER DIAGRAM



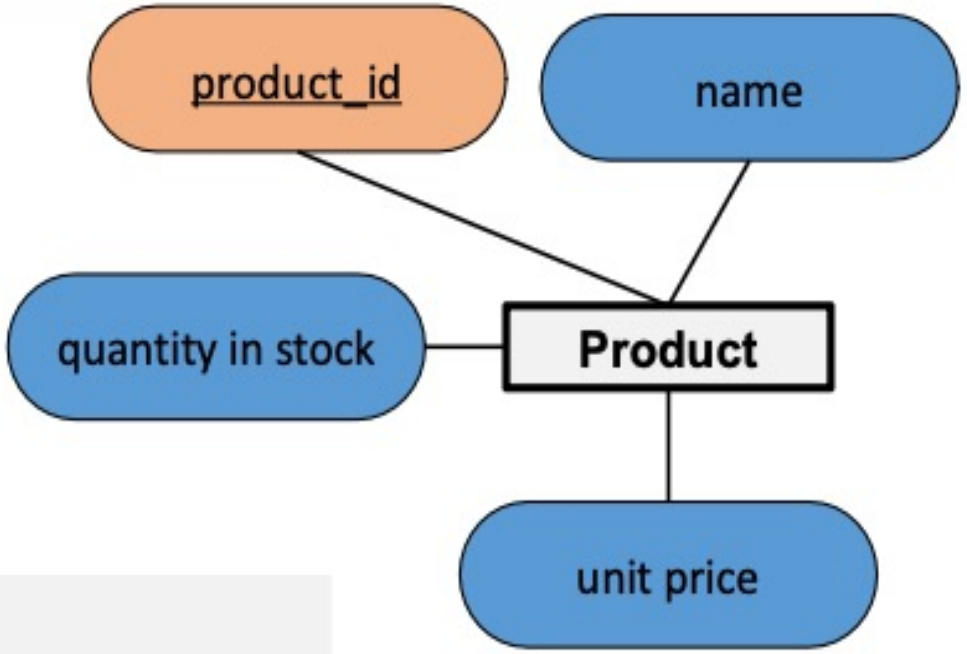
Thứ tự tạo table trong DBMS?



IMPLEMENT ER DIAGRAM

Create our database schema

```
DROP DATABASE IF EXISTS sql_store;
CREATE DATABASE sql_store;
USE sql_store;
```



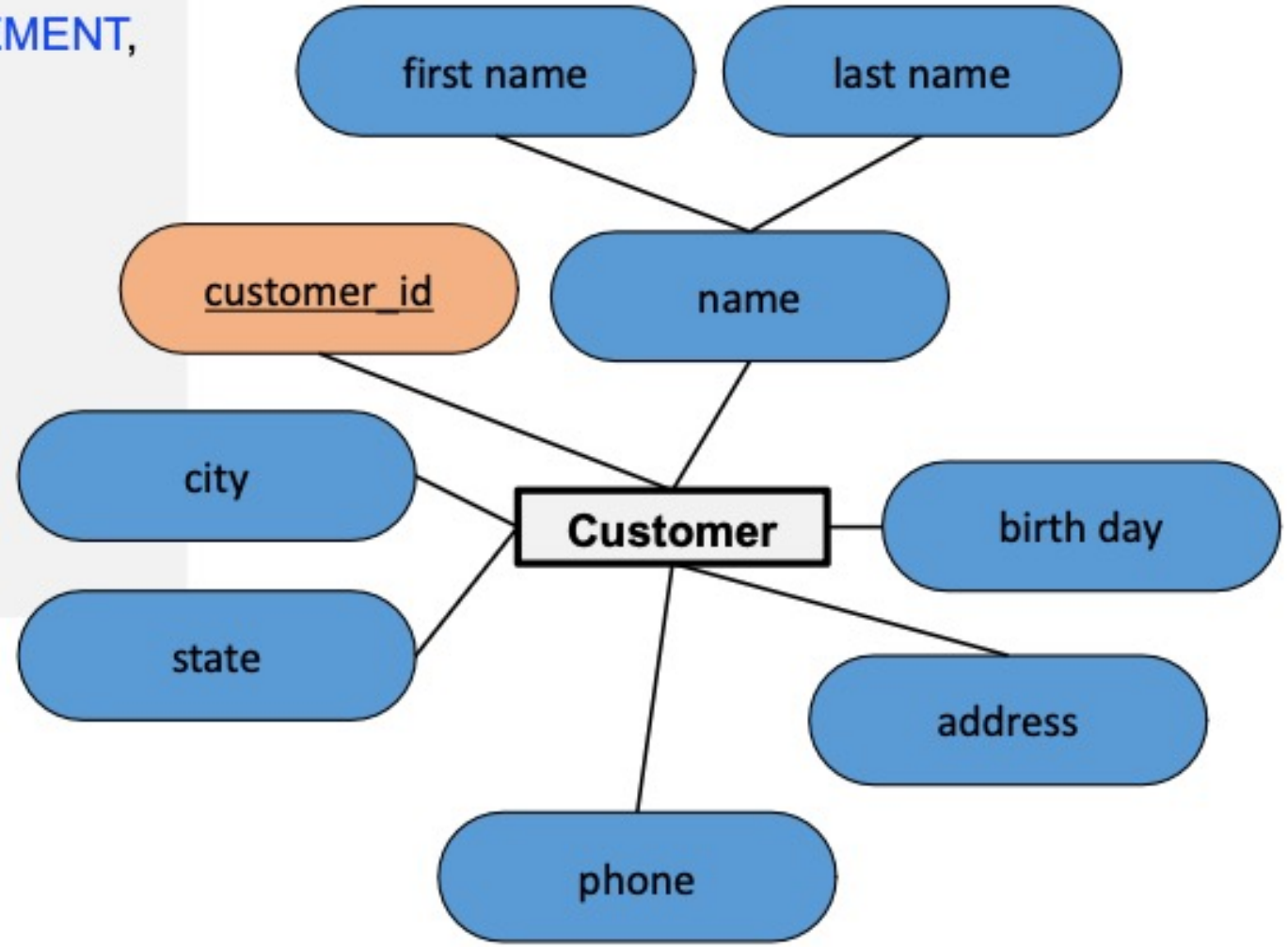
Create our Products table

```
CREATE TABLE products (
    product_id INT PRIMARY KEY NOT NULL  AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    quantity_in_stock INT NOT NULL,
    unit_price DECIMAL(4,2) NOT NULL
)
```

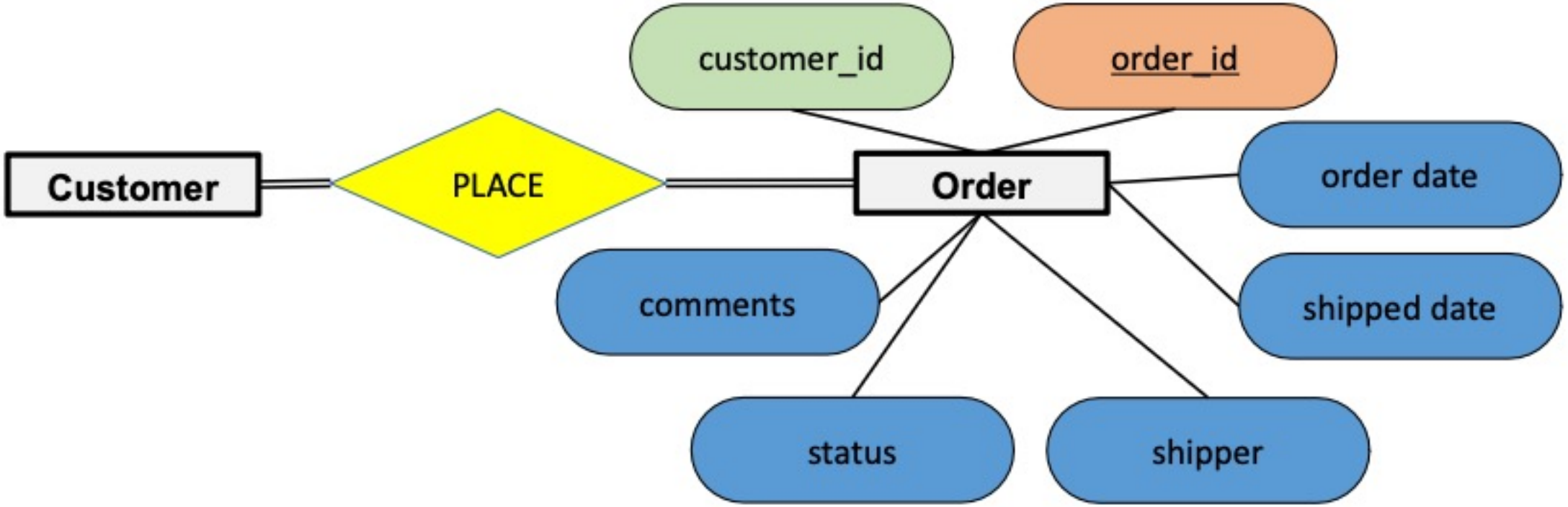
IMPLEMENT ER DIAGRAM

Create our Customers table

```
CREATE TABLE customers (
        customer_id INT NOT NULL AUTO_INCREMENT,
        first_name VARCHAR(50) NOT NULL,
        last_name VARCHAR(50) NOT NULL,
        birth_date DATE DEFAULT NULL,
        phone VARCHAR(50) DEFAULT NULL,
        address VARCHAR(50) NOT NULL,
        city VARCHAR(50) NOT NULL,
        state CHAR(2) NOT NULL,
        points INT NOT NULL DEFAULT '0',
        PRIMARY KEY (customer_id)
);
```



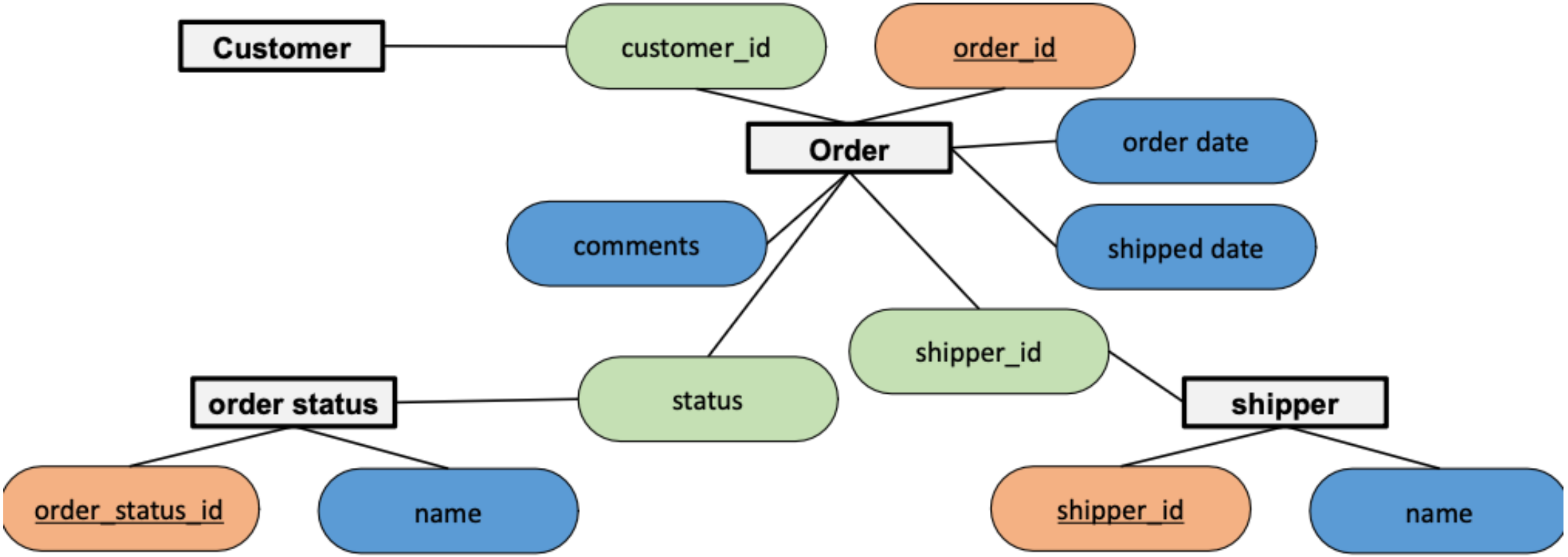
IMPLEMENT ER DIAGRAM



The order table relate to customer table which we already created

Althought the status and shipper attribute don't need to be it seperate entity, for the recreation of our practice database we will design it as an entity.

IMPLEMENT ER DIAGRAM



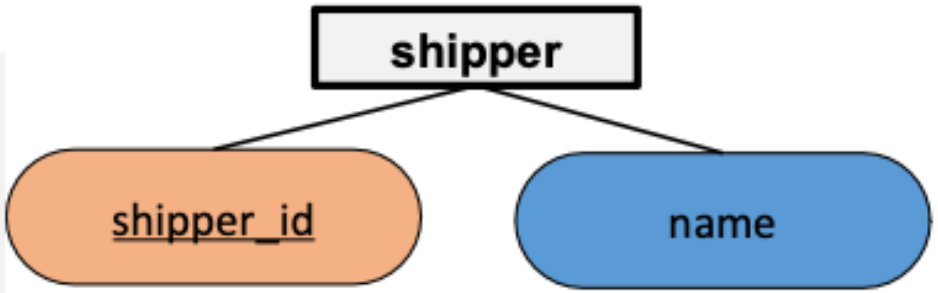
Again the order status and shipper entity don't have a foreign key so they should be created first then the order entity

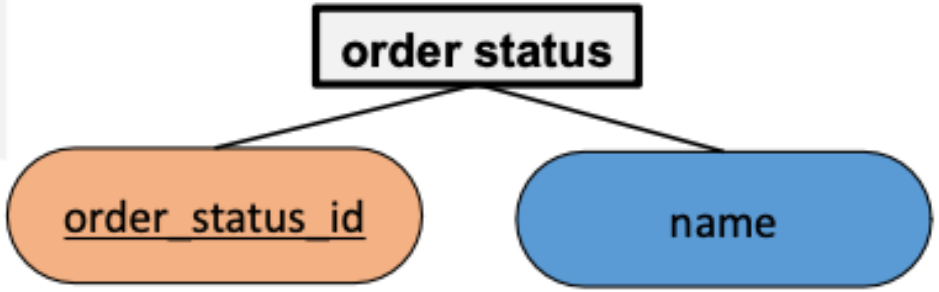
IMPLEMENT ER DIAGRAM

Create our Shipper and Order status table

```
CREATE TABLE shippers (
        shipper_id INT NOT NULL AUTO_INCREMENT,
        name VARCHAR(50) NOT NULL,
        PRIMARY KEY (shipper_id)
);

CREATE TABLE order_statuses (
        order_status_id INT NOT NULL,
        name VARCHAR(50) NOT NULL,
        PRIMARY KEY (order_status_id)
);
```





IMPLEMENT ER DIAGRAM

Create our Orders table

```
CREATE TABLE orders (
        order_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
        customer_id INT NOT NULL,
        order_date DATE NOT NULL,
        status INT NOT NULL DEFAULT '1',
        comments VARCHAR(2000) DEFAULT NULL,
        shipped_date DATE DEFAULT NULL,
        shipper_id INT DEFAULT NULL,
        CONSTRAINT fk_orders_customers FOREIGN KEY (customer_id)
                REFERENCES customers(customer_id) ON UPDATE CASCADE,
        CONSTRAINT fk_orders_order_statuses FOREIGN KEY (status)
                REFERENCES order_statuses(order_status_id) ON UPDATE CASCADE,
        CONSTRAINT fk_orders_shippers FOREIGN KEY (shipper_id)
                REFERENCES shippers (shipper_id) ON UPDATE CASCADE
);
```

```sql
CREATE TABLE Parent (
    id INT PRIMARY KEY,
    name VARCHAR(50)
);

CREATE TABLE Child (
    id INT PRIMARY KEY,
    parent_id INT,
    FOREIGN KEY (parent_id) REFERENCES Parent(id)
        ON UPDATE CASCADE
);
```

```sql
INSERT INTO Parent (id, name) VALUES (1, 'Parent A');
INSERT INTO Child (id, parent_id) VALUES (101, 1);
```

Parent_ID	Name
1	Parent A

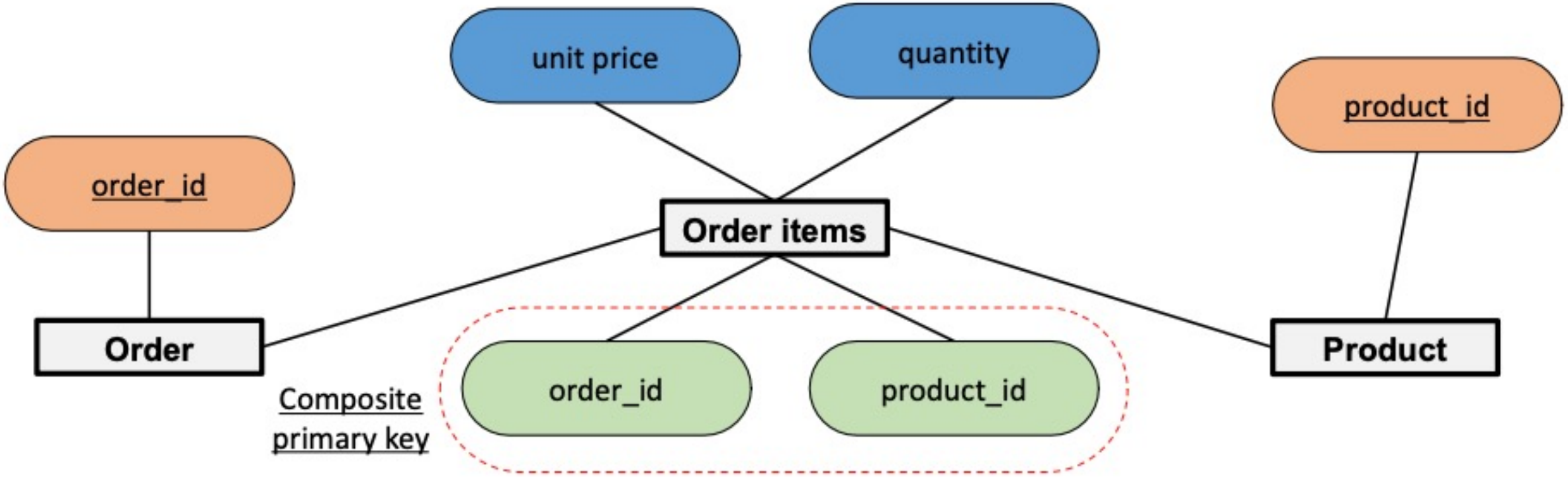
ID	Parent_ID
1	1

```sql
UPDATE Parent SET id = 2 WHERE id = 1;
```

Parent_ID	Name
2	Parent A

ID	Parent_ID
1	2

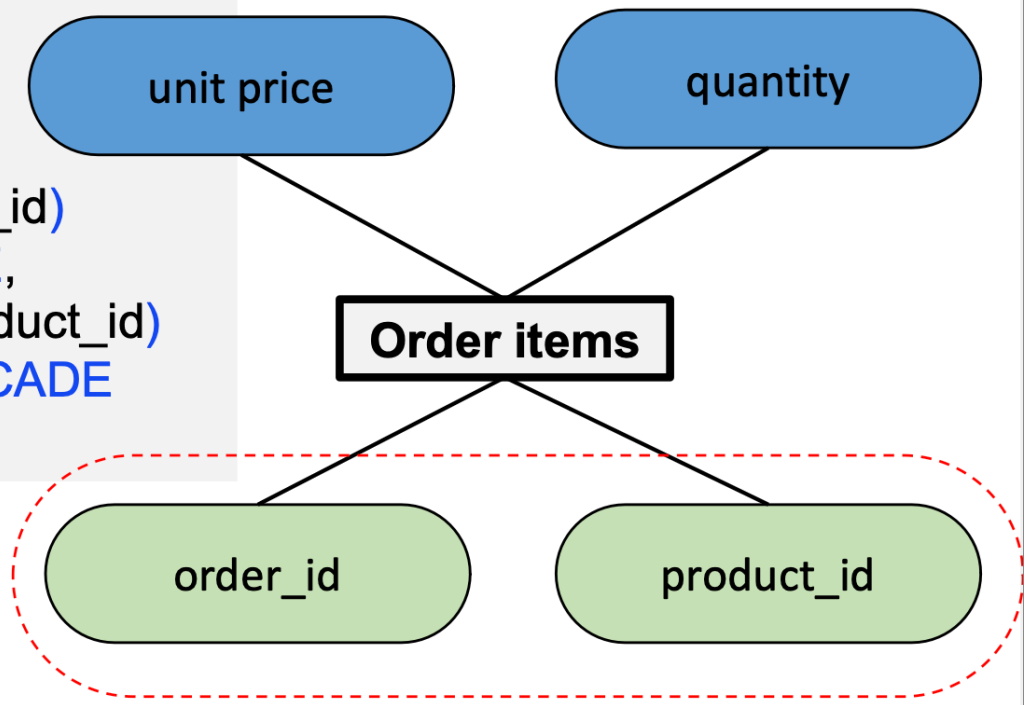
IMPLEMENT ER DIAGRAM



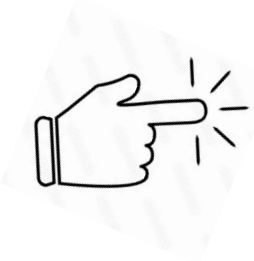
IMPLEMENT ER DIAGRAM

Create our Order items table

```
CREATE TABLE order_items (
        order_id INT NOT NULL AUTO_INCREMENT,
        product_id INT NOT NULL,
        quantity INT NOT NULL,
        unit_price DECIMAL(4,2) NOT NULL,
        PRIMARY KEY (order_id, product_id),
        CONSTRAINT fk_order_items_orders FOREIGN KEY (order_id)
            REFERENCES orders(order_id) ON UPDATE CASCADE,
        CONSTRAINT fk_order_items_products FOREIGN KEY (product_id)
            REFERENCES products (product_id) ON UPDATE CASCADE
);
```

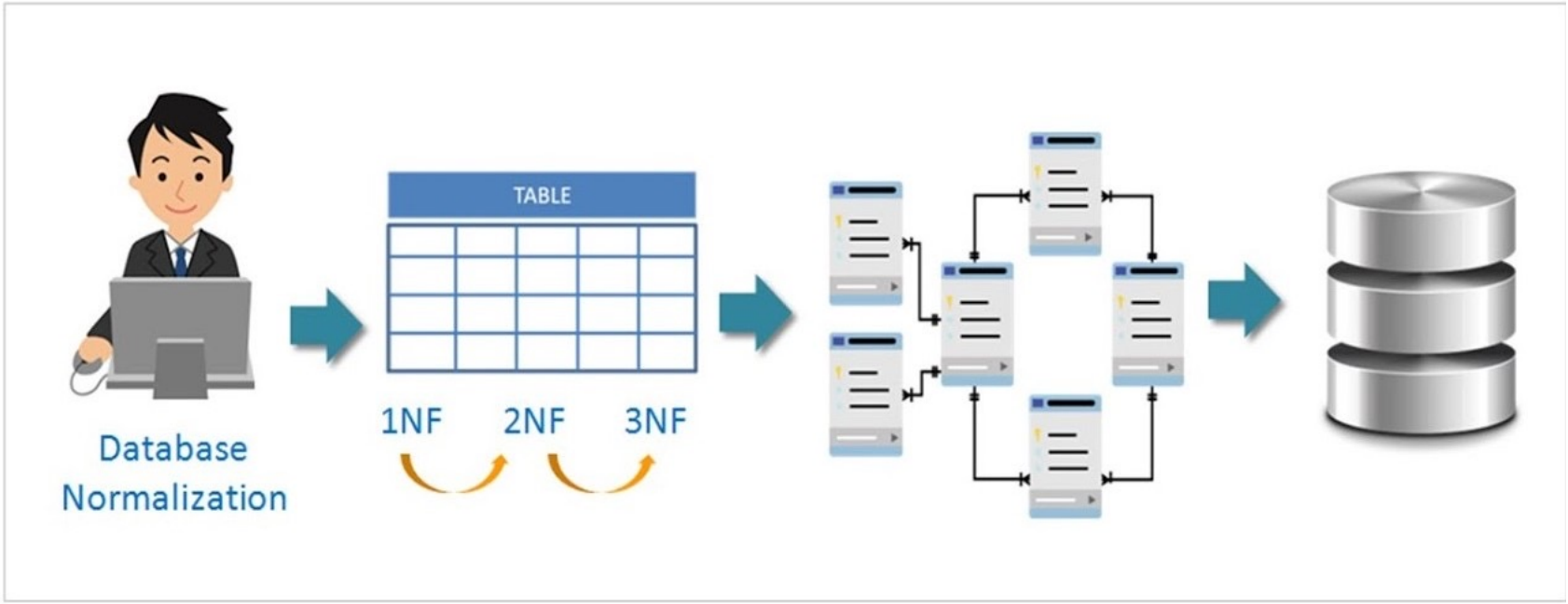


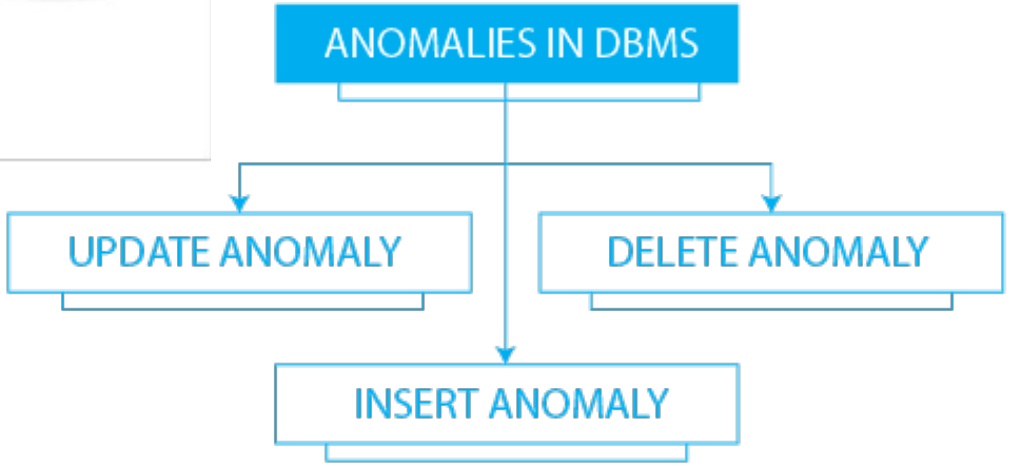
Outline



- Introduction to Entity Relationship Diagram
- Introduction to Database Normalization
- Advanced SQL Queries
- Kahoot Quiz
- Summary

Database Normalization





Why Database Normalization?

Normalization is a database design technique that reduces data redundancy.
Normalization rules divides larger tables into smaller tables and links them using relationships

1. Insertion anomalies: This occurs when we are not able to insert data into a database because some attributes may be missing at the time of insertion.

Students Table

StudentID	StudentName	CourseID	CourseName	Professor
1	Alice	101	Math	Dr. Smith
2	Bob	102	Science	Dr. Jones

Redundant Data

If a new student, Carol, needs to enroll in the Math course, you would have to insert the course information again, even though it already exists.

StudentID	StudentName	CourseID	CourseName	Professor
3	Carol	101	Math	Dr. Smith

Partial Data Insertion

If a new course needs to be added to the database but no students are enrolled yet, it's difficult to insert this data. For example, if a History course taught by Dr. Brown is introduced:

StudentID	StudentName	CourseID	CourseName	Professor
		103	History	Dr. Brown

Null Value Issues

If a new student, Dave, is admitted but hasn't yet enrolled in any course, inserting his data might require null values for CourseID, CourseName, and Professor.

StudentID	StudentName	CourseID	CourseName	Professor
4	Dave	NULL	NULL	NULL

Why Database Normalization?

Normalization is a database design technique that reduces data redundancy.
Normalization rules divides larger tables into smaller tables and links them using relationships

2. Updation anomalies: This occurs when the same data items are repeated with the same values and are not linked to each other.

StudentID	StudentName	CourseID	CourseName	Professor
1	Alice	101	Math	Dr. Smith
2	Bob	102	Science	Dr. Jones
3	Carol	101	Math	Dr. Smith

Redundant Data Update

If Dr. Smith's name changes to Dr. Brown, this change must be made in multiple rows.

StudentID	StudentName	CourseID	CourseName	Professor
1	Alice	101	Math	Dr. Brown
2	Bob	102	Science	Dr. Jones
3	Carol	101	Math	Dr. Brown

Inconsistent Data

If only one instance of Dr. Smith's name is updated, the database will have inconsistent data.

StudentID	StudentName	CourseID	CourseName	Professor
1	Alice	101	Math	Dr. Brown
2	Bob	102	Science	Dr. Jones
3	Carol	101	Math	Dr. Smith

Why Database Normalization?

Normalization is a database design technique that reduces data redundancy.
Normalization rules divides larger tables into smaller tables and links them using relationships

3. Delete anomalies: This occurs when deleting one part of the data deletes the other necessary information from the database.

StudentID	StudentName	CourseID	CourseName	Professor
1	Alice	101	Math	Dr. Smith
2	Bob	102	Science	Dr. Jones
3	Carol	101	Math	Dr. Smith

Unintended Data Loss

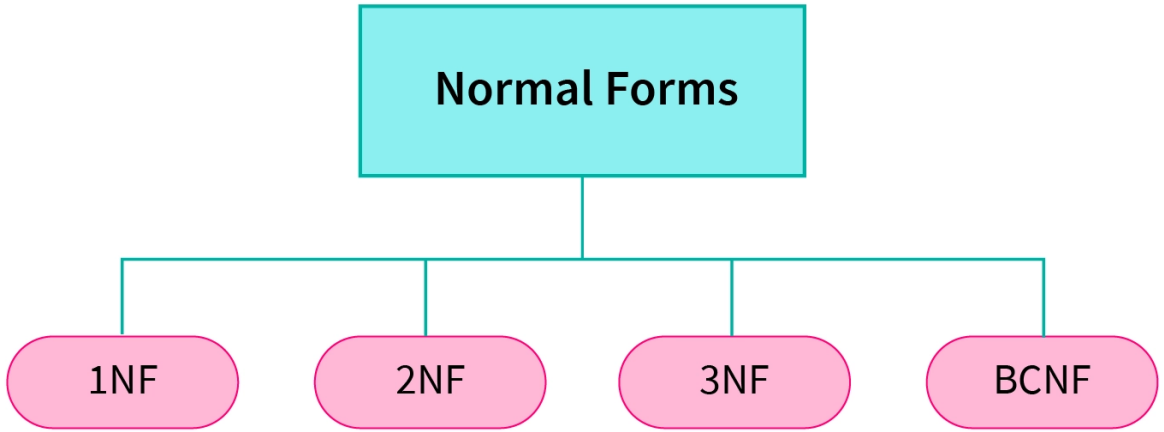
If a student drops out or graduates and their record is deleted, you might unintentionally lose information about the courses they were enrolled in if those courses are only stored in the same table.

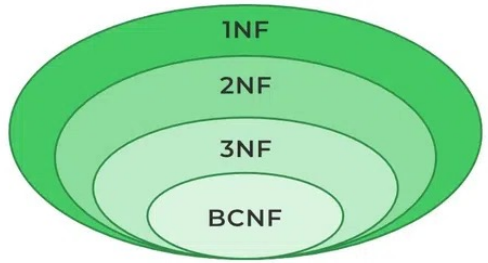
For example, if Alice (StudentID 1) is deleted:

StudentID	StudentName	CourseID	CourseName	Professor
2	Bob	102	Science	Dr. Jones
3	Carol	101	Math	Dr. Smith

Orphan Records: Deleting a record that is referenced by other records, leading to orphaned data.

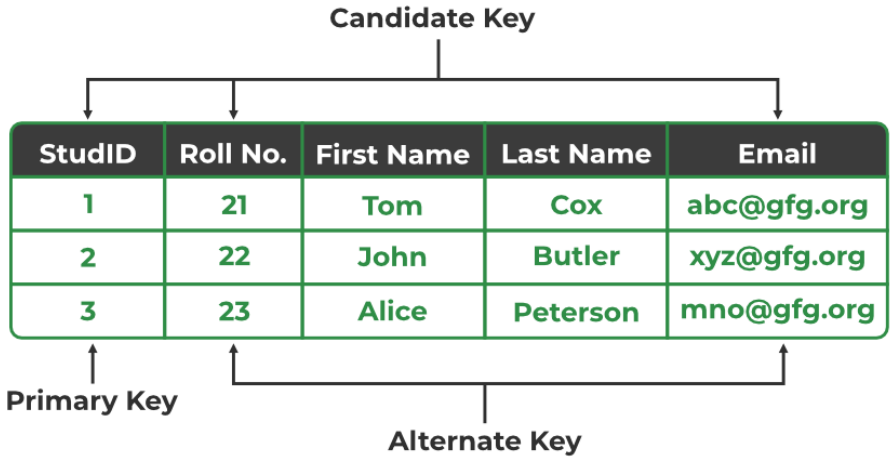
Normalization Form





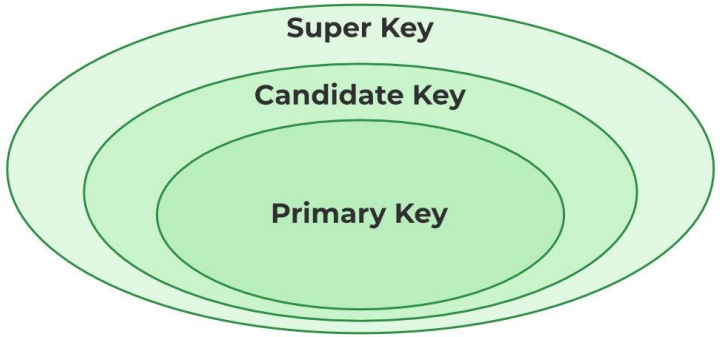
- **1NF:** A relation is in 1NF if all its attributes have an atomic value.
- **2NF:** A relation is in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
- **3NF:** A relation is in 3NF if it is in 2NF and there is no transitive dependency.
- **BCNF:** A relation is in BCNF if it is in 3NF and for every Functional Dependency say P->Q, P should be a super key.

Terminology

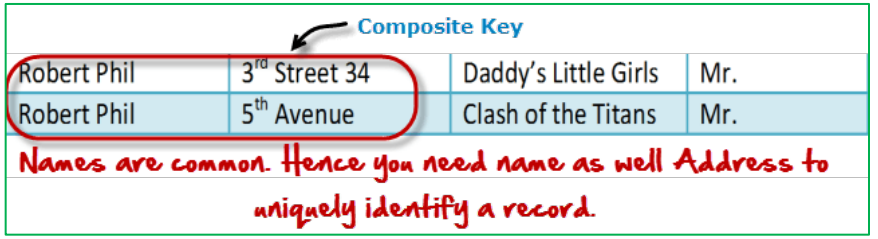


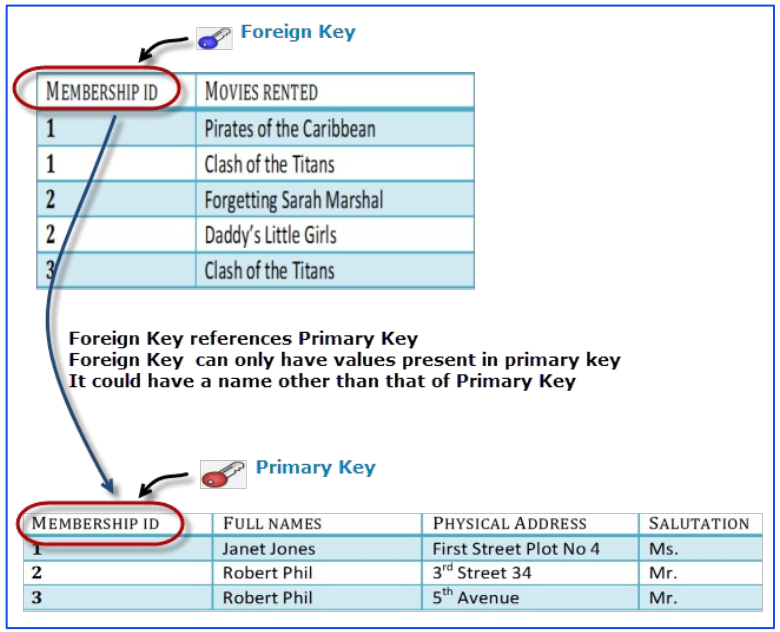
A super key is a group of single or multiple keys that identifies rows in a table. It supports NULL values.

For Example, StuID, Roll No., (StuID, Roll No.), (StuID, First Name),...



A primary is a single column value used to identify a database record uniquely.





Normalization Form

❖ **1NF:** A relation is in 1NF if all its attributes have an atomic value.

A relation R is said to be in 1 NF (First Normal) if and only if:
•All the attributes of R are **atomic**.
•It does not contain **any multi-valued** attributes.

Student Code	Student Name	Phone Number
101	Vinh	19001080, 19001081
101	Vinh	19001082
102	Hung	19001083
103	Lan	19001884

Mỗi ô chỉ chứa một giá trị đơn (không có mảng, danh sách,...)
Không có các cột lặp
Mỗi hàng là duy nhất

Student Code	Student Name	Phone Number
101	Vinh	19001080
101	Vinh	19001080
101	Vinh	19001082
102	Hung	19001083
103	Lan	19001884

Normalization Form

❖ **1NF:** A relation is in 1NF if all its attributes have an atomic value.

•All the attributes (columns) contain only atomic (indivisible) values.

•Each column contains values of a single type.

•Each record (row) is unique, meaning it can be identified by a primary key.

•There are no repeating groups or arrays in any row.

Student Code	Student Name	Phone Number
101	Vinh	19001080, 19001081
101	Vinh	19001082
102	Hung	19001083
103	Lan	19001884

Student Code	Student Name	Phone Number
101	Vinh	19001080
101	Vinh	19001080
101	Vinh	19001082
102	Hung	19001083
103	Lan	19001884

Normalization Form

❖ **2NF:** The normalization of 1NF relations to 2NF involves the elimination of partial dependencies.

1.The table must be in first normal form
2.It must not contain any partial dependency, i.e., all non-prime attributes are fully functionally dependent on the primary key.
3.All non-key attributes are fully functionally dependent on the entire primary key

Student Code	Student Name	Project ID	Project Name
101	Vinh	P03	Project 103
101	Vinh	P01	Project 101
102	Hung	P04	Project 104
103	Lan	P02	Project 102

Student

Student Code	Student Name
101	Vinh
101	Vinh
102	Hung
103	Lan

Project

Project ID	Project Name
P03	Project 103
P01	Project 101
P04	Project 104
P02	Project 102

Student_Project

Student Code	Project ID
101	P03
101	P01
102	P04
103	P02

the table has no partial dependencies (i.e., no non-prime attribute is dependent on a part of a candidate key)

Normalization Form

❖ **2NF:** The normalization of 1NF relations to 2NF involves the elimination of partial dependencies.

Mọi thuộc tính **không khóa** phải **phụ thuộc hoàn toàn vào khóa chính, không phụ thuộc một phần.**

Student Code	Student Name	Project ID	Project Name
101	Vinh	P03	Project 103
101	Vinh	P01	Project 101
102	Hung	P04	Project 104
103	Lan	P02	Project 102

Student

Student Code	Student Name
101	Vinh
101	Vinh
102	Hung
103	Lan

Project

Project ID	Project Name
P03	Project 103
P01	Project 101
P04	Project 104
P02	Project 102

Student_Project

Student Code	Project ID
101	P03
101	P01
102	P04
103	P02

the table has no partial dependencies (i.e., no non-prime attribute is dependent on a part of a candidate key)

Normalization Form

❖ **2NF:** The normalization of 1NF relations to 2NF involves the elimination of partial dependencies.

COURSES Table

STUD_NO	COURSE_NO	COURSE_FEE
1	C1	1000
2	C2	1500
1	C4	2000
4	C3	1000
4	C1	1000
2	C5	2000

2NF?

Normalization Form

❖ **2NF:** The normalization of 1NF relations to 2NF involves the elimination of partial dependencies.

STUDENT_COURSES Table

STUD_NO	COURSE_NO
1	C1
2	C2
1	C4
4	C3
4	C1
2	C5

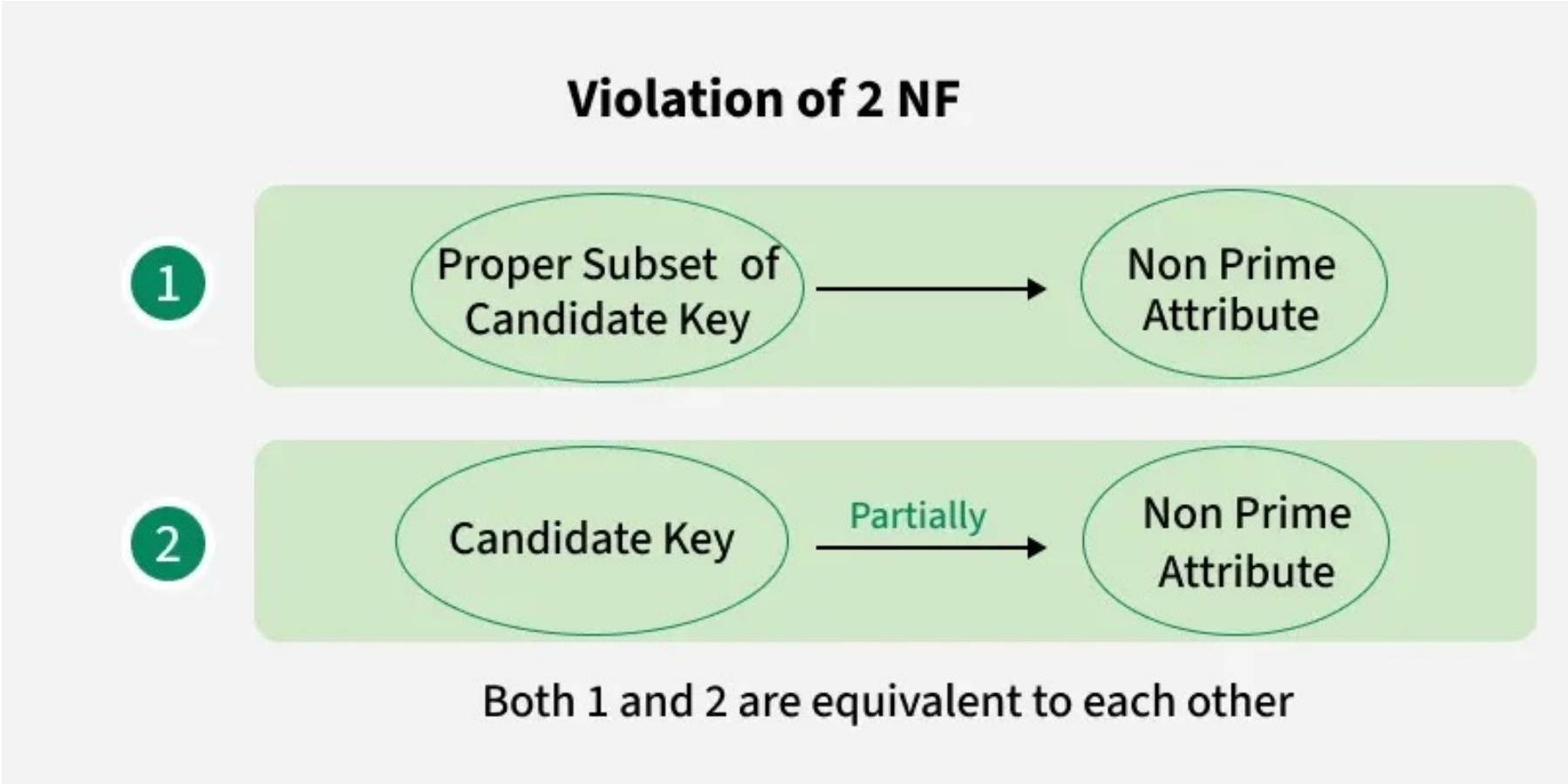
COURSES Table

COURSE_NO	COURSE_FEE
C1	1000
C2	1500
C4	2000
C3	1000
C5	2000

2NF?

Normalization Form

❖ **2NF:** The normalization of 1NF relations to 2NF involves the elimination of partial dependencies.



Normalization Form

❖ **3NF:** The normalization of 2NF relations to 3NF involves the elimination of transitive dependencies.

A relation is in **Third Normal Form (3NF)** if it satisfies the following two conditions:

- It is in Second Normal Form (2NF): This means the table has no partial dependencies (i.e., no non-prime attribute is dependent on a part of a candidate key).
- There is no transitive dependency for non-prime attributes: In simpler terms, no non-key attribute should depend on another non-key attribute. Instead, all non-key attributes should depend directly on the primary key.

For example, if we have the following relationship between attributes:

•**A -> B** (A determines B)

•**B -> C** (B determines C)

This means that **A** indirectly determines **C** through **B**, creating a **transitive dependency**

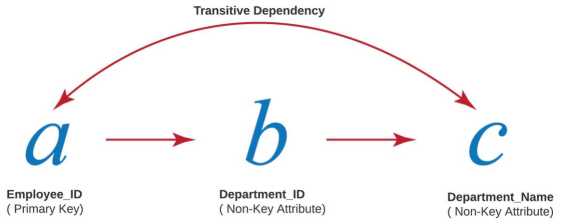
3NF eliminates these transitive dependencies to ensure that non-key attributes are directly dependent only on the primary key

Normalization Form

❖ **3NF:** The normalization of 2NF relations to 3NF involves the elimination of transitive dependencies.

A functional dependency X -> Z is said to be transitive if the following three functional dependencies hold:
•X -> Y & Y does not -> X & Y -> Z

Student Code	Student Name	Student Zipcode	Student City
1	Vinh	94000	CanTho
2	Vinh	84000	Tay Ninh
3	Andy	84000	Tây Ninh



Một phụ thuộc bắc cầu xảy ra khi một thuộc tính không khóa phụ thuộc vào một thuộc tính không khóa khác, thay vì phụ thuộc trực tiếp vào khóa chính.

Student_Location

Student Code	Student Name	Student Zipcode
1	Vinh	94000
2	Vinh	84000
3	Andy	84000

Student

Student Zipcode	Student City
94000	CanTho
84000	Tay Ninh
84000	Tây Ninh

Normalization Form

❖ **3NF:** The normalization of 2NF relations to 3NF involves the elimination of transitive dependencies.

CANDIDATE Table

CAND_NO	CAND_NAME	CAND_STATE	CAND_COUNTRY	CAND_AGE
1	Amayra	West Bengal	India	18
2	Rihaan	Haryana	India	17
3	Manya	Haryana	India	19

3NF?

Normalization Form

❖ **3NF:** The normalization of 2NF relations to 3NF involves the elimination of transitive dependencies.

CANDIDATE Table

CAND_NO	CAND_NAME	CAND_STATE	CAND_AGE
1	Amayra	West Bengal	18
2	Rihaan	Haryana	17
3	Manya	Haryana	19

STATE_COUNTRY Table

CAND_STATE	CAND_COUNTRY
West Bengal	India
Haryana	India

3NF?

Normalization Form

Student	Teacher	Subject
Shayan	R.Das	Database
Shayan	K.Raman	C
Tahira	R.Das	Database
Tahira	N.Gupta	C

1NF? 2NF? 3NF?

FD: { (student, Teacher) -> subject, (student, subject) -> Teacher, (Teacher) -> subject}

Normalization Form

Order ID	Customer ID	Customer Name	Customer City
1	C001	Alice	New York
2	C002	Bob	London
3	C001	Alice	New York
4	C003	Charlie	Paris

1NF? 2NF? 3NF?

Normalization Form

Course ID	Course Name	Department	Credits
CS101	Intro to CS	Computer Science	3
MA201	Calculus I	Mathematics	4
EN101	English Comp	English	3

1NF? 2NF? 3NF?

Normalization Form

❖ **BCNF (3.5NF):** Boyce-Codd Normal Form is an advanced version of 3NF as it contains additional constraints compared to 3NF.

1.The table must be in the third normal form
2.For every non-trivial functional dependency X -> Y, X is the superkey of the table. That means X cannot be a non-prime attribute if Y is a prime attribute.

Student ID	Course ID	Instructor ID	Instructor Name
101	CS101	I01	John Smith
102	CS101	I01	John Smith
103	MA201	I02	Alice Brown

Với các ràng buộc:
•(Student ID, Course ID) là khóa ứng cử.
•(Student ID, Instructor ID) là khóa ứng cử.

•**Instructor ID -> Course ID** (Mỗi giảng viên chỉ dạy một khóa học duy nhất).
•Instructor ID -> Instructor Name (Mỗi ID giảng viên có một tên duy nhất).

Trivial Functional Dependency

Một phụ thuộc hàm $X \rightarrow Y$ được gọi là **tầm thường** nếu tập hợp các thuộc tính Y là một tập con của tập hợp các thuộc tính X (tức là $Y \subseteq X$).

`MaNV` $\rightarrow$ `MaNV` : Đây là một phụ thuộc hàm tầm thường. Nếu bạn biết `MaNV`, bạn đương nhiên biết `MaNV`. ($\{MaNV\} \subseteq \{MaNV\}$)

(`MaNV` , `TenNV`) $\rightarrow$ `MaNV` : Đây cũng là một phụ thuộc hàm tầm thường. Nếu bạn biết cả `MaNV` và `TenNV`, bạn đương nhiên biết `MaNV`. ($\{MaNV\} \subseteq \{MaNV, TenNV\}$)

(`MaNV` , `TenNV`) $\rightarrow$ `TenNV` : Tương tự. ($\{TenNV\} \subseteq \{MaNV, TenNV\}$)

(`MaNV` , `TenNV`) $\rightarrow$ (`MaNV` , `TenNV`): Cũng là tầm thường. ($\{MaNV, TenNV\} \subseteq \{MaNV, TenNV\}$)

Non-Trivial Functional Dependency

Một phụ thuộc hàm $X \rightarrow Y$ được gọi là **không tầm thường** nếu tập hợp các thuộc tính Y **không phải** là tập con của tập hợp các thuộc tính X (tức là $Y \not\subseteq X$).

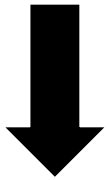
- `MaNV` $\rightarrow$ `TenNV`: Đây là một phụ thuộc hàm không tầm thường. `TenNV` không phải là tập con của `MaNV`. Điều này có nghĩa là nếu bạn biết mã nhân viên, bạn có thể xác định duy nhất tên nhân viên đó.
- `MaNV` $\rightarrow$ `PhongBan`: Không tầm thường. (`PhongBan` không phải là tập con của `MaNV`).
- `MaNV` $\rightarrow$ `Luong`: Không tầm thường. (`Luong` không phải là tập con của `MaNV`).
- `MaNV` $\rightarrow$ (`TenNV`, `PhongBan`): Không tầm thường. (`{TenNV, PhongBan}` không phải là tập con của `{MaNV}`).

Normalization Form

❖ **BCNF (3.5NF):** Boyce-Codd Normal Form is an advanced version of 3NF as it contains additional constraints compared to 3NF.

Rule 1: *The table should be in the 3rd Normal Form.*
Rule 2: *X should be a superkey for every functional dependency (FD) X–>Y in a given relation.*

Student ID	Course ID	Instructor ID	Instructor Name
101	CS101	I01	John Smith
102	CS101	I01	John Smith
103	MA201	I02	Alice Brown



Student ID	Course ID	Instructor ID
101	CS101	I01
102	CS101	I01
103	MA201	I02

Instructor ID	Instructor Name
I01	John Smith
I02	Alice Brown

Normalization Form

2NF: Remove partial dependency

3NF: Remove transitive dependency

4NF: Remove Multi-value Dependency

For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1.It should be in the **Boyce-Codd Normal Form**.

2.And, the table should not have any **Multi-valued Dependency**.

4NF Form

What is a Multi-valued Dependency

A table is said to have multi-valued dependency, if the following conditions are true,
1.For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.

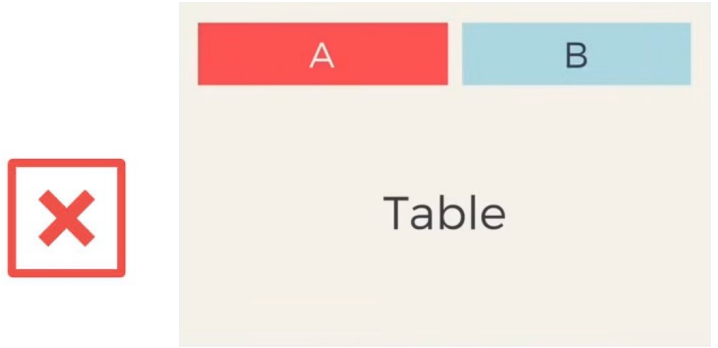
2.Also, a table should have at-least 3 columns for it to have a multi-valued dependency.

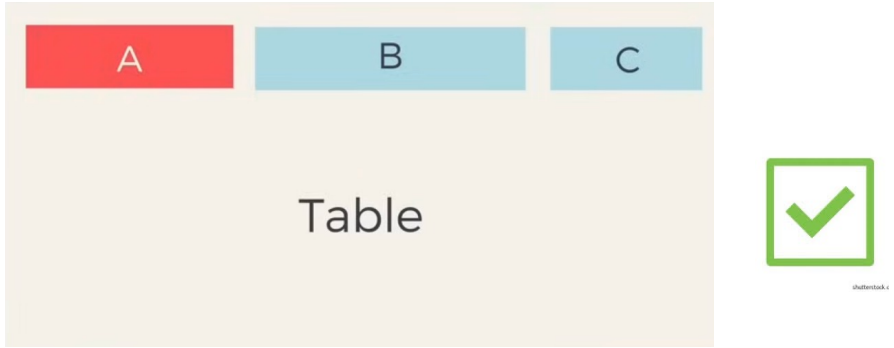
3.And, for a relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

4NF Form

What is a Multi-valued Dependency

Nó đã ở **BCNF và không có sự phụ thuộc đa trị (multivalued dependency) phi tầm thường nào mà bên trái không phải là khóa.**





4NF Form

Model_ID	Color	Assembly_Location
M1	Red	Factory_A
M1	Red	Factory_B
M1	Blue	Factory_A
M1	Blue	Factory_B
M2	Green	Factory_C
M2	Yellow	Factory_C

BCNF?

4NF Form

Model_ID	Color	Assembly_Location
M1	Red	Factory_A
M1	Red	Factory_B
M1	Blue	Factory_A
M1	Blue	Factory_B
M2	Green	Factory_C
M2	Yellow	Factory_C

Have any Multi-valued Dependency?

4NF Form

Model_ID	Color	Assembly_Location
M1	Red	Factory_A
M1	Red	Factory_B
M1	Blue	Factory_A
M1	Blue	Factory_B
M2	Green	Factory_C
M2	Yellow	Factory_C

Đây chính xác là định nghĩa của phụ thuộc đa trị không tầm thường:

- `Model_ID` ↠ `Color`
- `Model_ID` ↠ `Assembly_Location`

Not 4F

4NF Form

Model_ID	Color
M1	Red
M1	Red
M1	Blue
M1	Blue
M2	Green
M2	Yellow

Not 4F

Model_ID	Assembly_Location
M1	Factory_A
M1	Factory_B
M1	Factory_A
M1	Factory_B
M2	Factory_C
M2	Factory_C

5NF Form

A Relation is considered to be in 5NF, if and only if -

- It's in the 4NF.
- If we can further break down the table to remove redundancy and anomaly and then re-join the decomposed tables using candidate keys, we should not lose the original data or create a new recordset.
- In other words, combining two or more deconstructed tables should not result in the loss of records or the creation of new records.

5NF Form

A Relation is considered to be in 5NF, if and only if -

- Nó ở 4NF và mọi phép phân rã (decomposition) của quan hệ thành nhiều bảng con đều là phép phân rã có thể nối lại (lossless join), và không thể phân rã thêm mà vẫn giữ được tính không mất mát (lossless).

5NF Form

Dealer	Product	Supplier
D1	P1	S1
D1	P2	S1
D2	P1	S1
D2	P2	S2
D1	P1	S2
D2	P1	S2

4NF?

5NF Form

Dealer	Product
D1	P1
D1	P2
D2	P1
D2	P2
D1	P1
D2	P1

Product	Supplier
P1	S1
P2	S1
P1	S1
P2	S2
P1	S2
P1	S2

Dealer	Supplier
D1	S1
D1	S1
D2	S1
D2	S2
D1	S2
D2	S2

Case Study

StudentID	StudentName	Courses
1	Alice	{CS101, MA101}
2	Bob	{CS101, PH101, MA101}

Does it satisfy 1NF and Why?

Case Study

StudentID	StudentName	CourseCode
1	Alice	CS101
1	Alice	MA101
2	Bob	CS101
2	Bob	PH101
2	Bob	MA101

Does it satisfy 1NF, 2NF? Why?

Case Study

StudentID	StudentName
1	Alice
2	Bob

StudentID	Course Code	Course Name
1	CS101	Computer Science
1	MA101	Math
2	CS101	Computer Science
2	PH101	Physic
2	MA101	Math

Does it satisfy 1NF, 2NF? Why?

Case Study

StudentID	StudentName
1	Alice
2	Bob

Course Code	Course Name
CS101	Computer Science
MA101	Math
CS101	Computer Science
PH101	Physic
MA101	Math

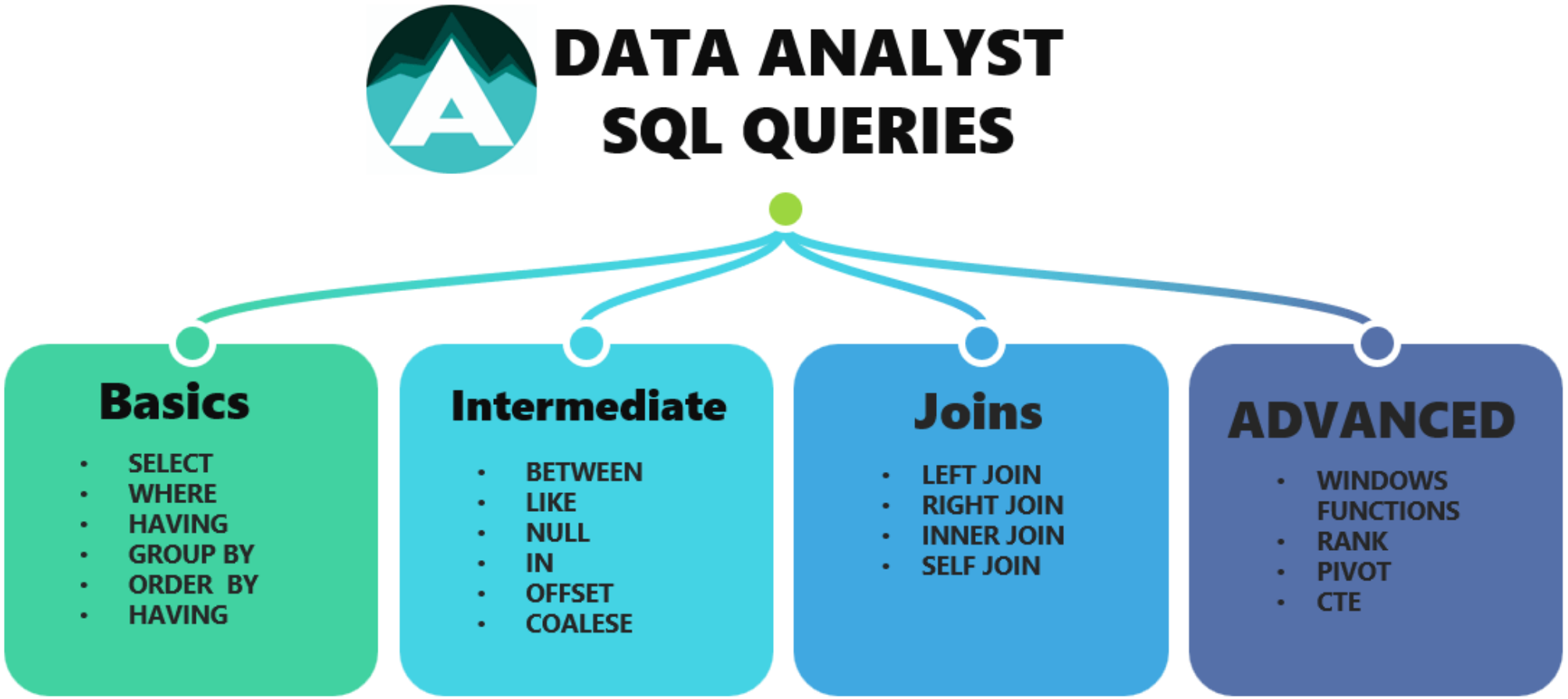
StudentID	Course Code
1	CS101
1	MA101
2	CS101
2	PH101
2	MA101

Does it satisfy 1NF, 2NF, 3NF ? Why?

Outline

- Introduction to Entity Relationship Diagram
- Introduction to Database Normalization
- Advanced SQL Queries
- Kahoot Quiz
- Summary

Summary


DATA ANALYST
SQL QUERIES
Basics
SELECT
WHERE
HAVING
GROUP BY
ORDER BY
HAVING
Intermediate
BETWEEN
LIKE
NULL
IN
OFFSET
COALESE
Joins
LEFT JOIN
RIGHT JOIN
INNER JOIN
SELF JOIN
ADVANCED
WINDOWS FUNCTIONS
RANK
PIVOT
CTE