AI VIET NAM
@aivietnam.edu.vn

SQL in Data Analysis

(ERD & Database Normalization)

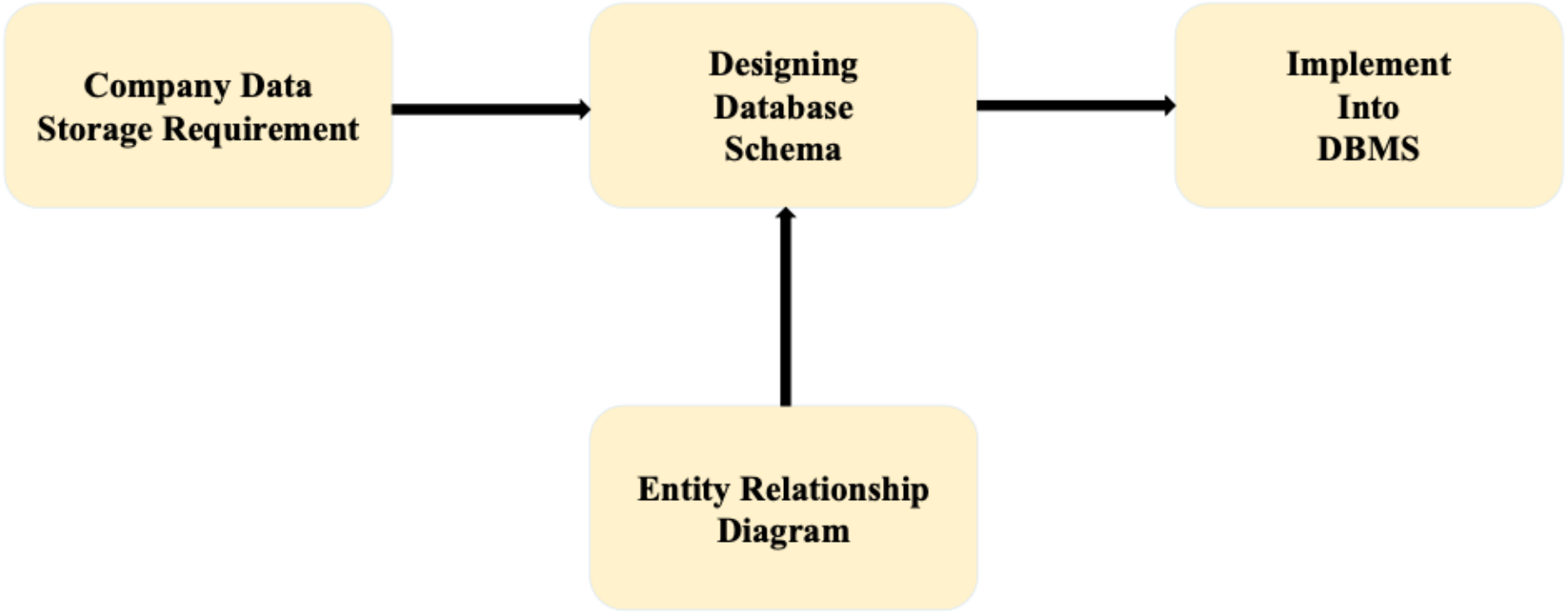
Vinh Dinh Nguyen
PhD in Computer Science

Outline

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

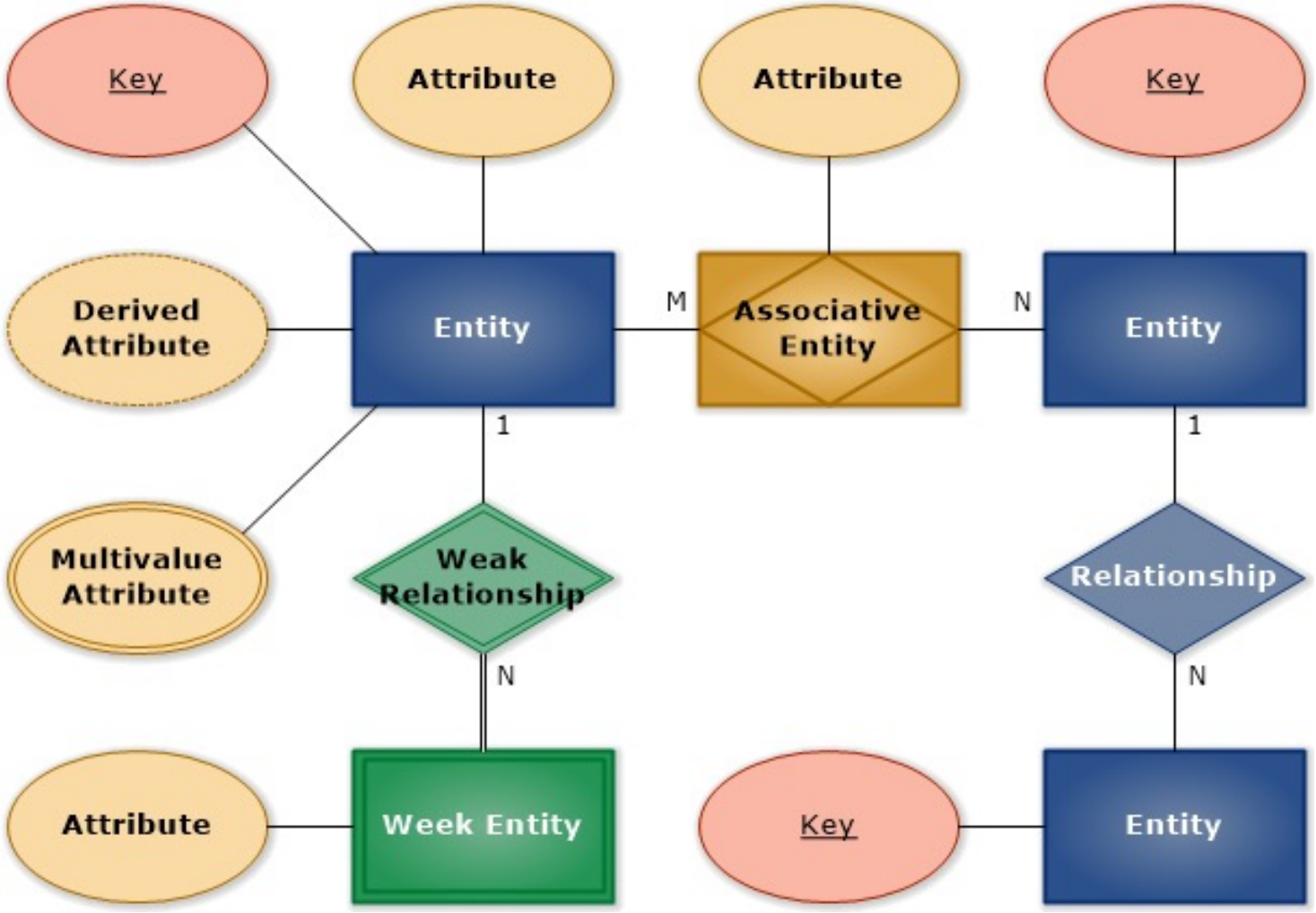
ENTITY RELATIONSIP DIAGRAM

ER DIAGRAM



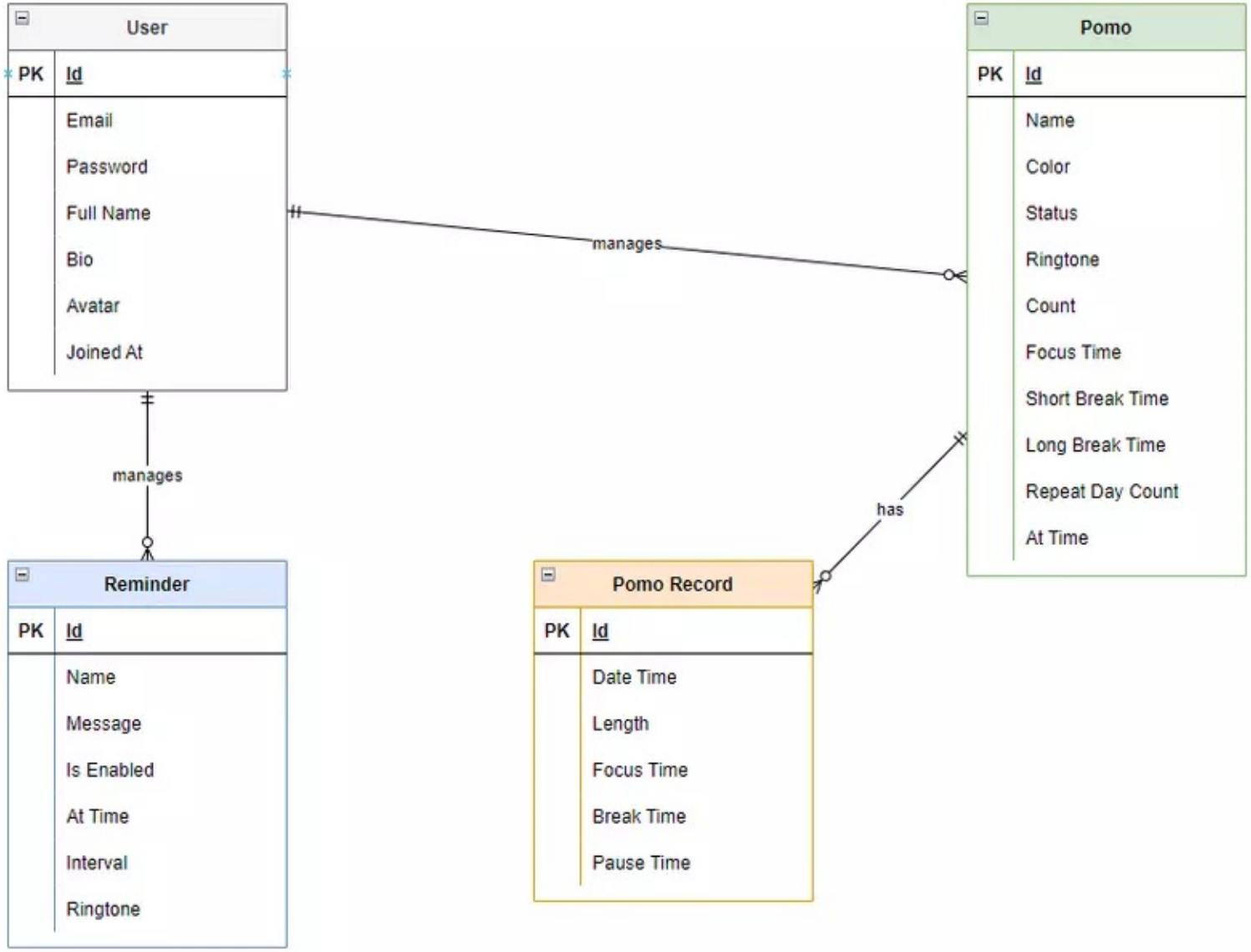
An Entity Relationship Diagram (ERD) is a visual representation of **different entities within a system and how they relate to each other**. It is a tool used to design and model relational databases, and shows the logical structure of the database

Chen Notation



AI VIET NAM
@aivietnam.edu.vn

Crow's Foot Notation



Crow's foot notation

Entity

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.

Chen's notation

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.

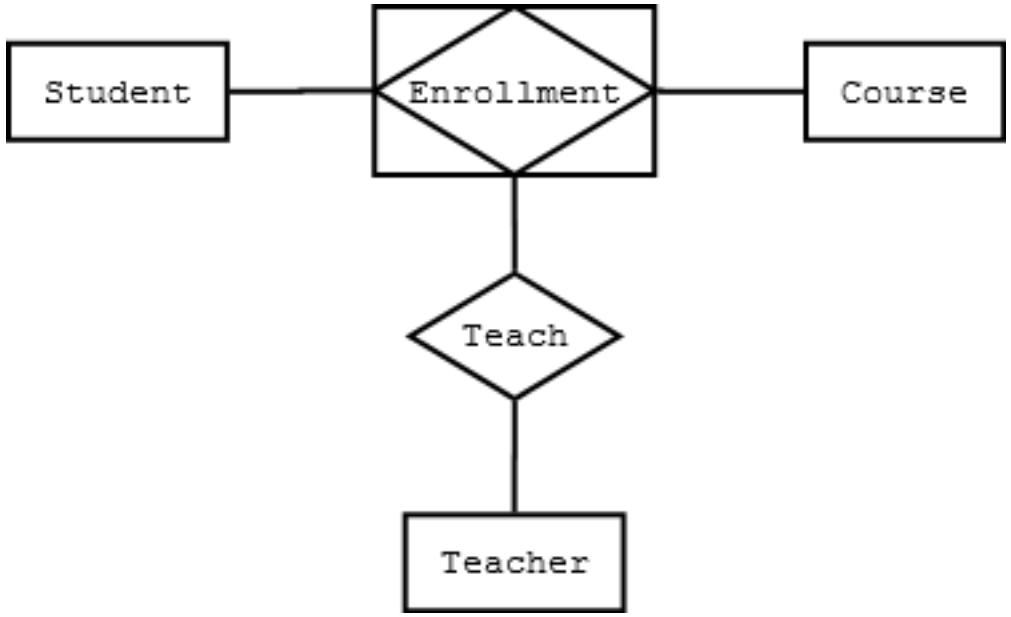
An entity like order item is a good example for this. The order item will be meaningless without an order so it depends on the existence of the order.

Chen's notation

It uses a foreign key combined with its attributed to form the primary key.

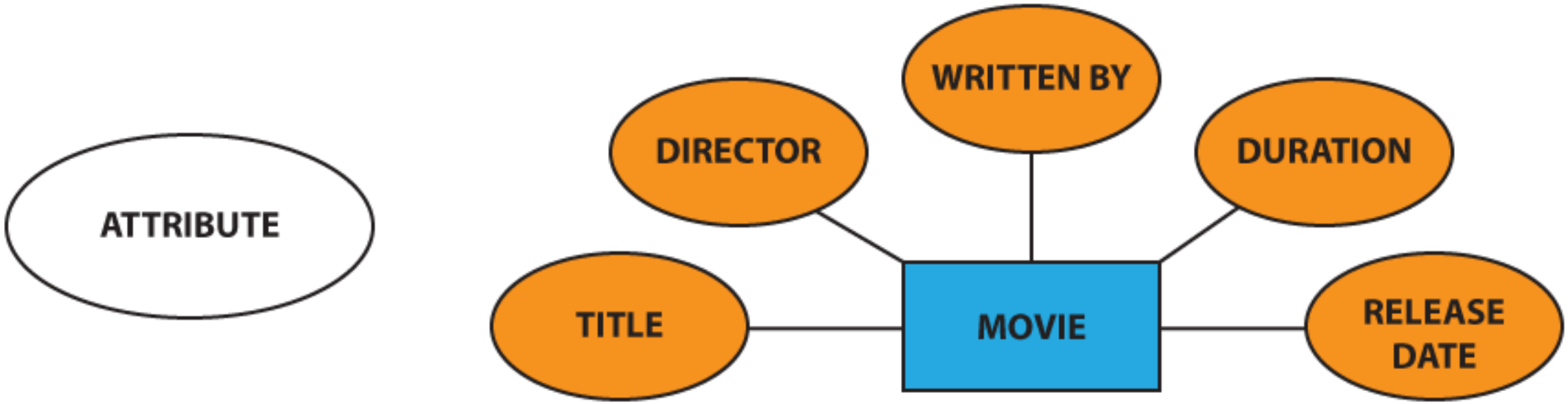
Associative Entity

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



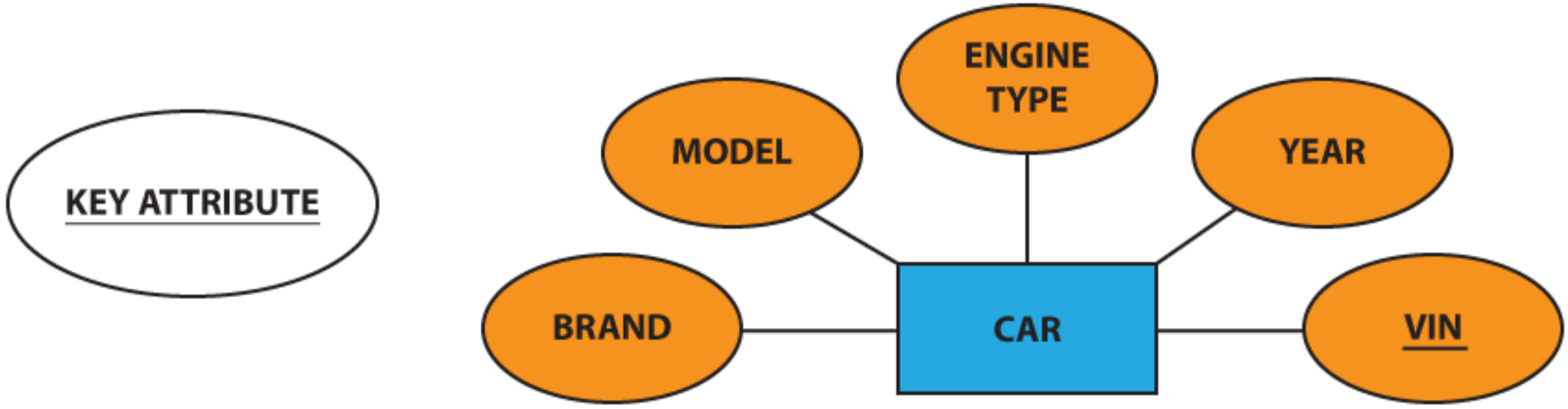
Attribute

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



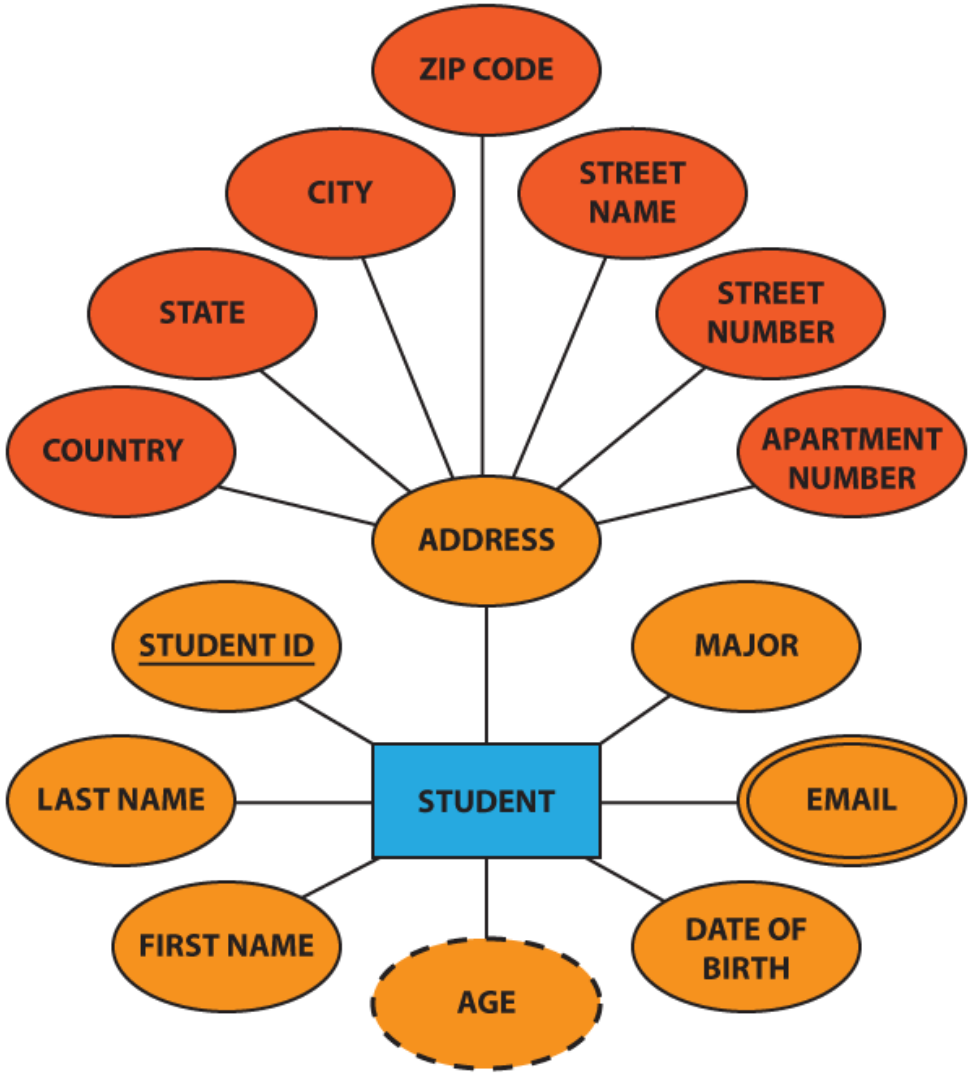
Key Attribute

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

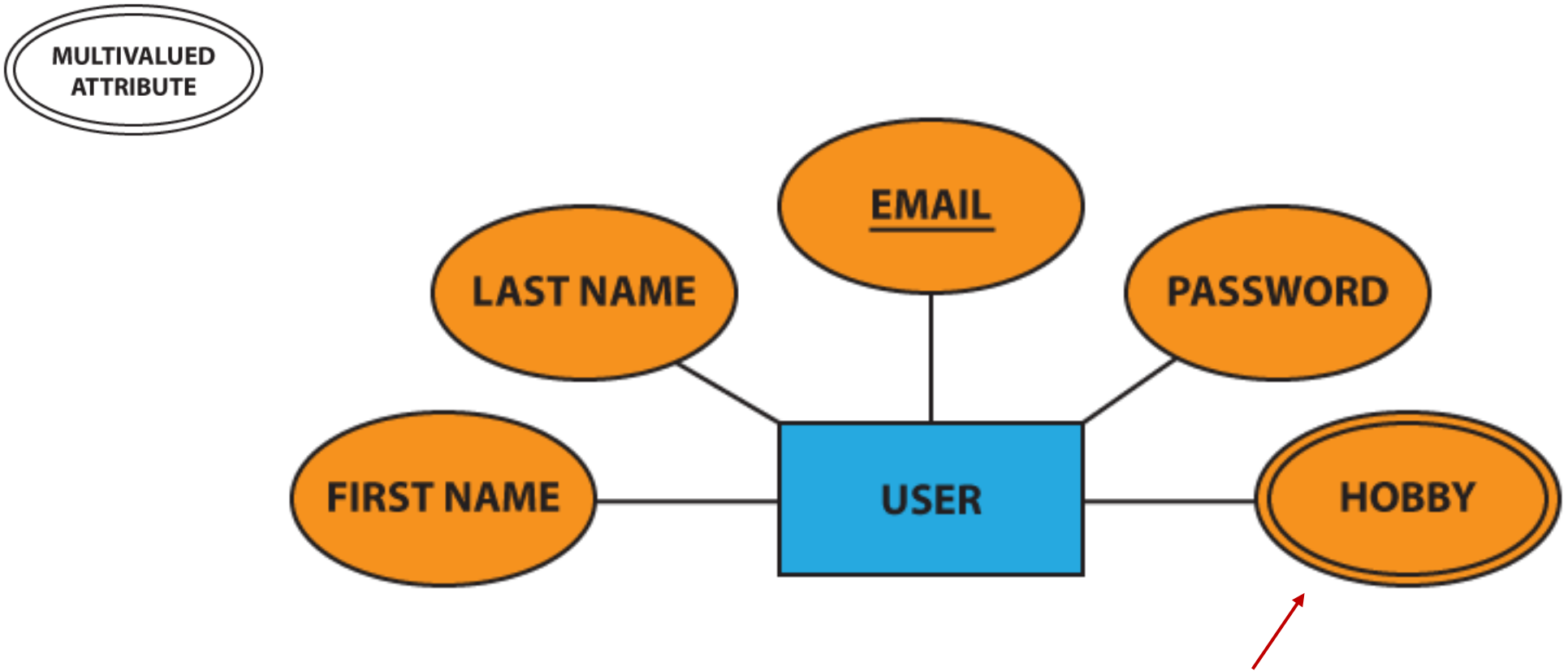


Composite Attributes

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.

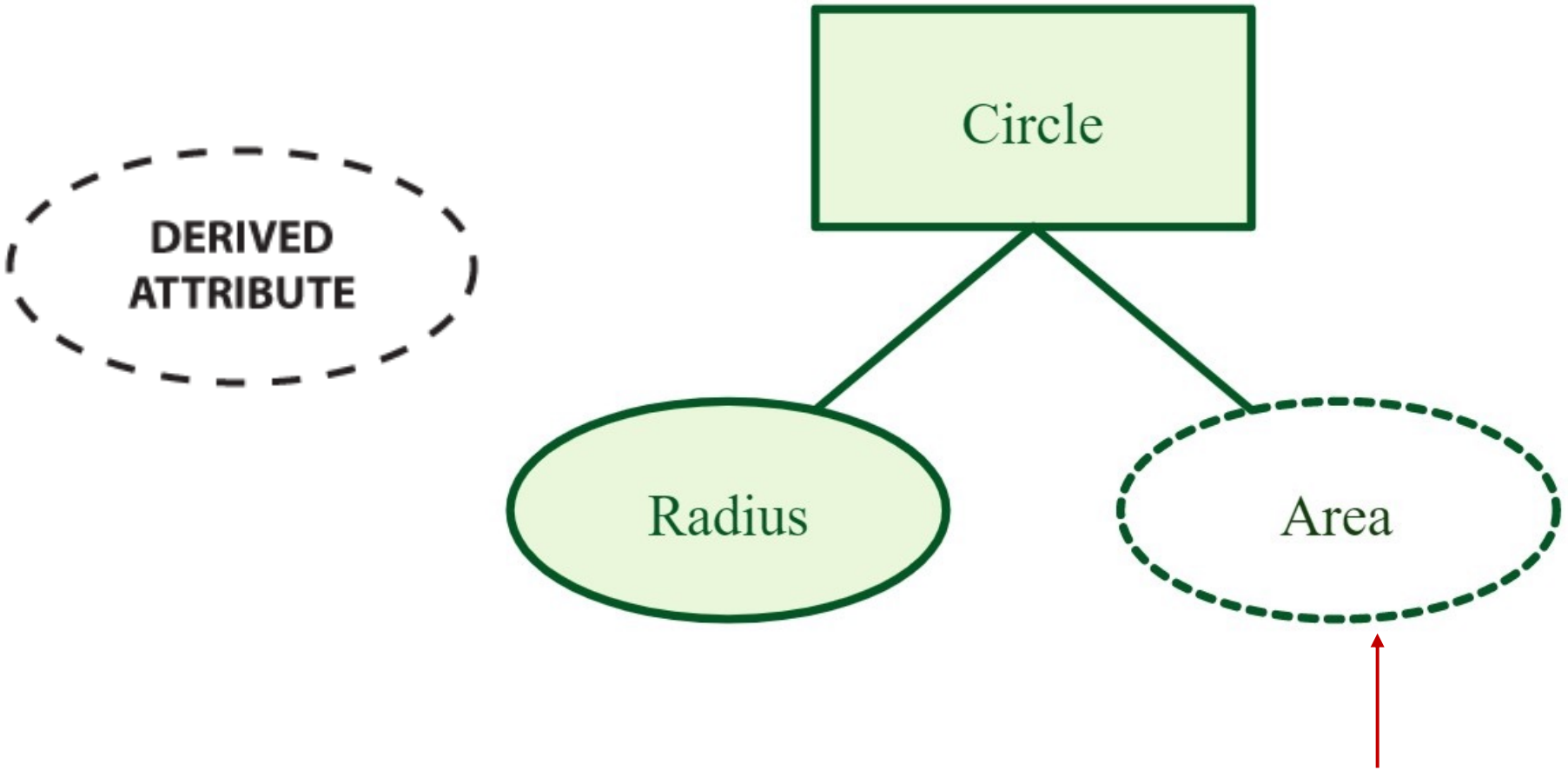


Multivalued Attribute



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)

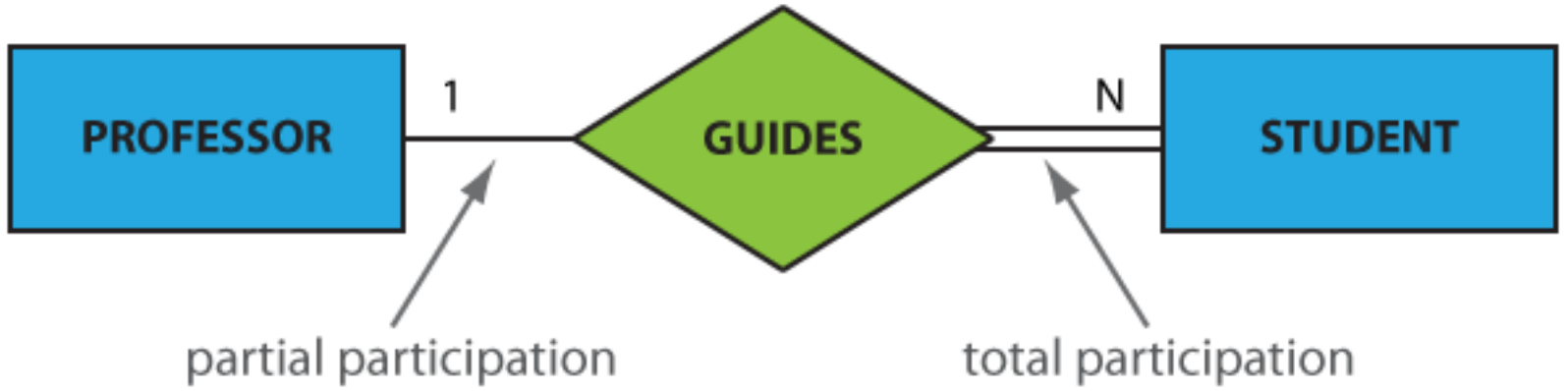
many-to-one (N:1)

many-to-many (M:N)

Participation Constraints

•**Total participation** means that every entity in the set is involved in the relationship, e.g., each student must be guided by a professor (there are no students who are not guided by any professor). In the Chen notation, this kind of relation is depicted as a double line.

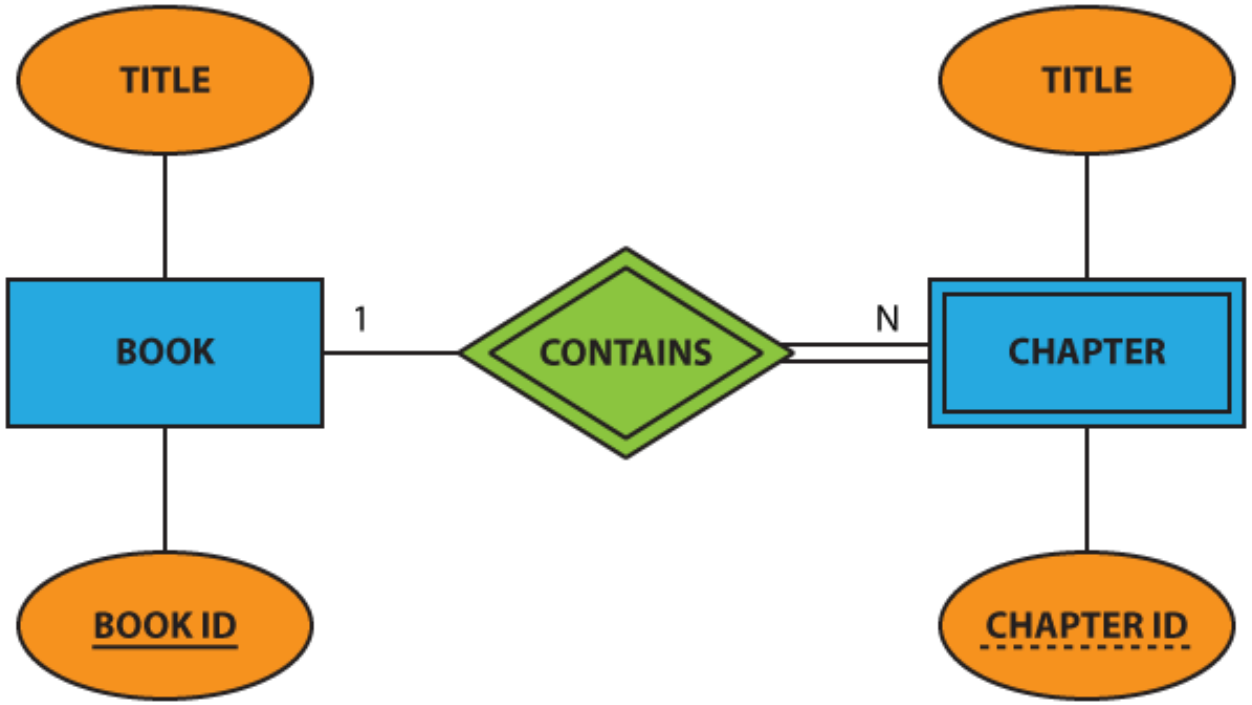
•**Partial participation** means that not all entities in the set are involved in the relationship, e.g., not every professor guides a student (there are professors who don't). In the Chen notation, a partial participation is represented by a single line.



Participation Constraints

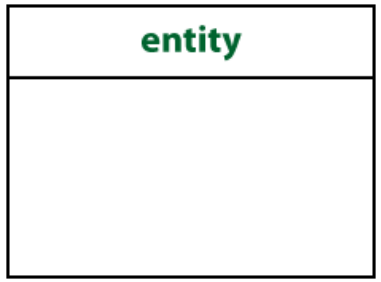
•**Total participation** means that every entity in the set is involved in the relationship, e.g., each student must be guided by a professor (there are no students who are not guided by any professor). In the Chen notation, this kind of relation is depicted as a double line.

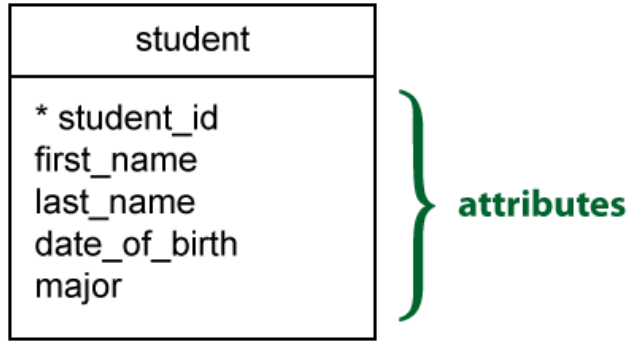
•**Partial participation** means that not all entities in the set are involved in the relationship, e.g., not every professor guides a student (there are professors who don't). In the Chen notation, a partial participation is represented by a single line.

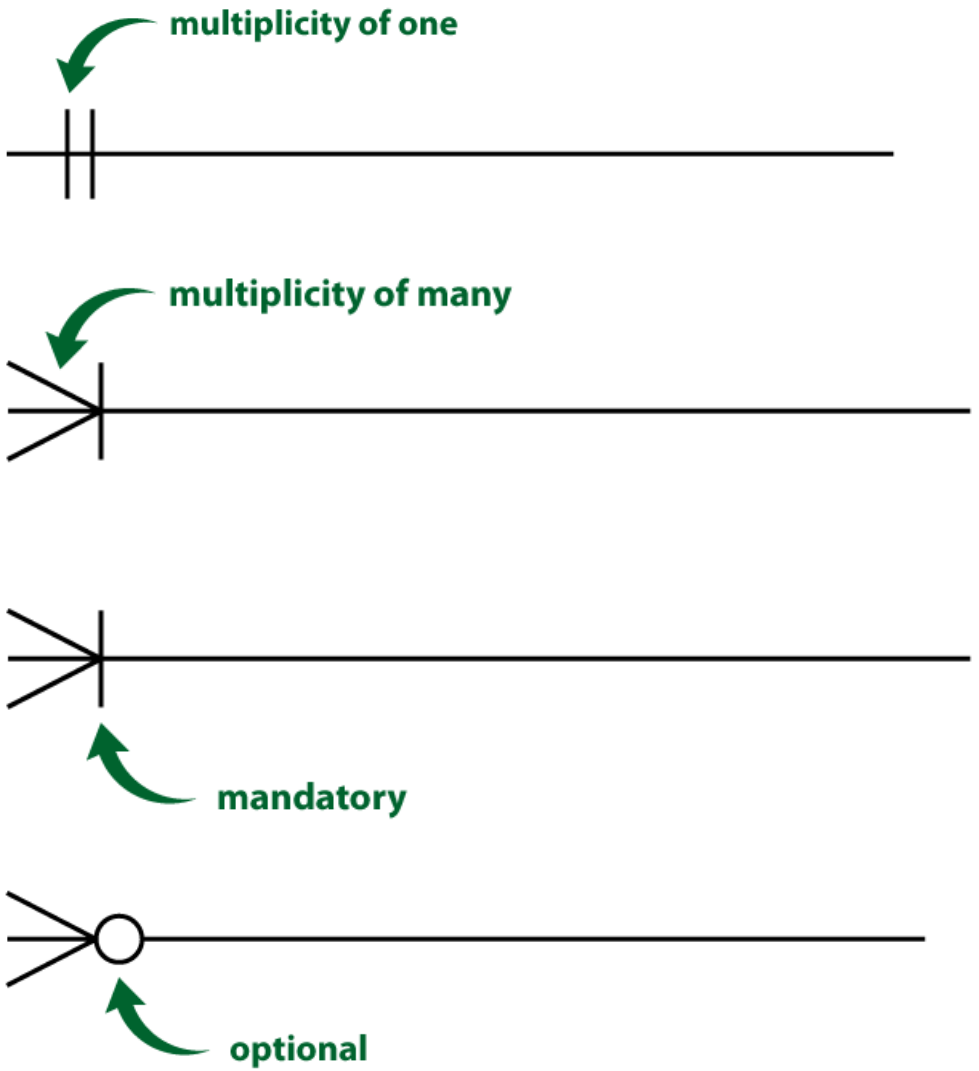


Crow's Foot Notation

Entities

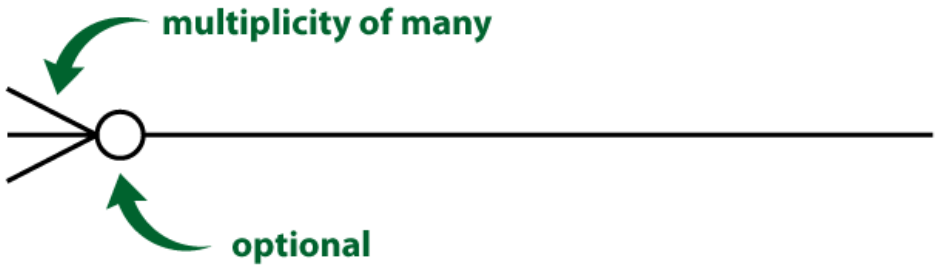


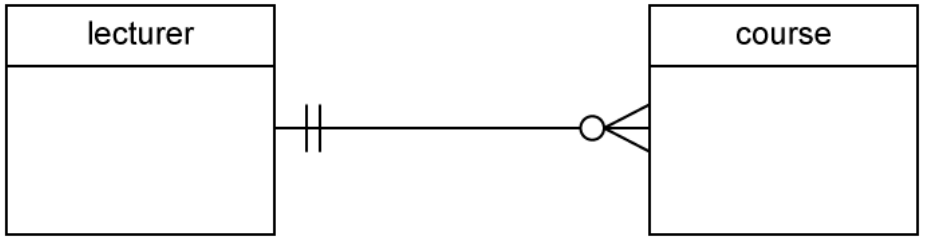




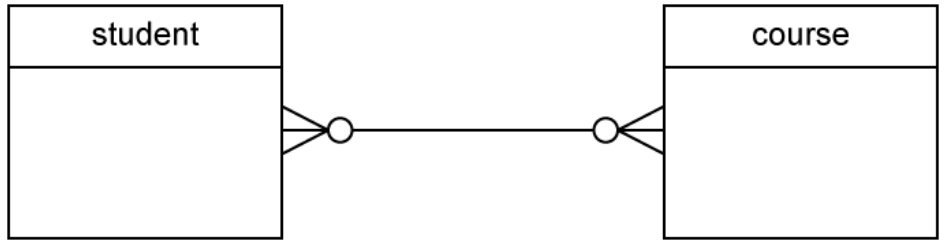
Crow's Foot Notation

zero or many



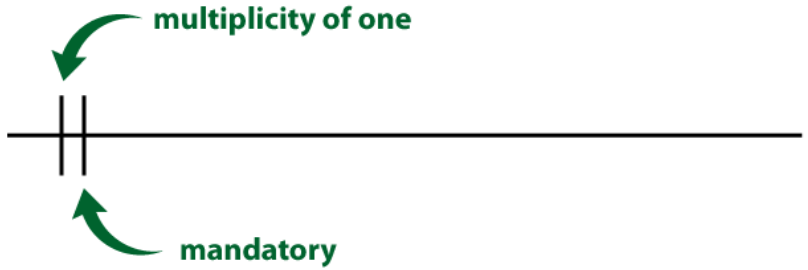


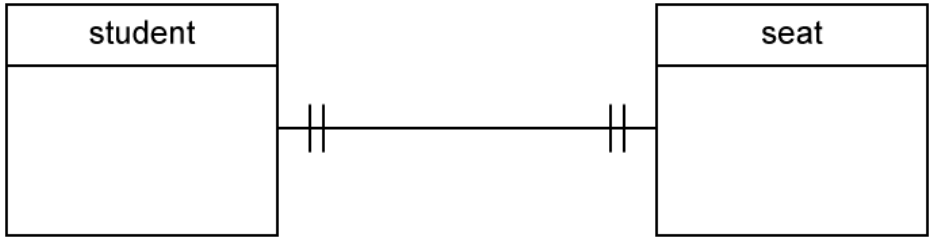
one or many



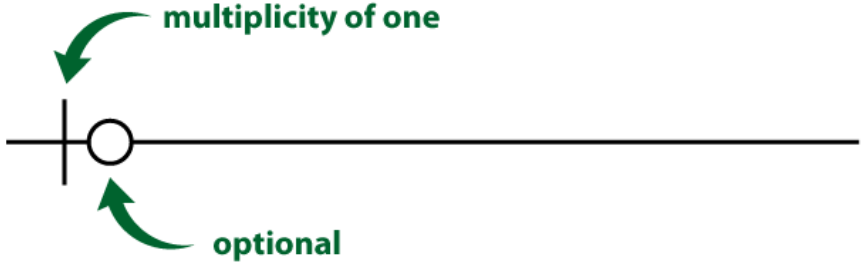
Crow's Foot Notation

one and only one





zero or one



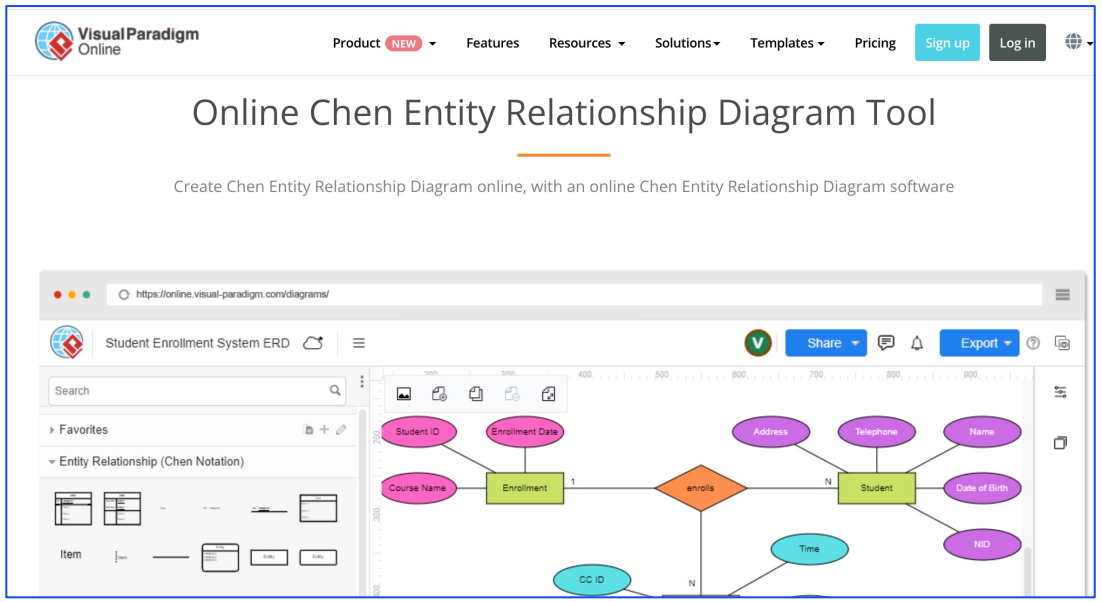
How to Draw ER Diagram

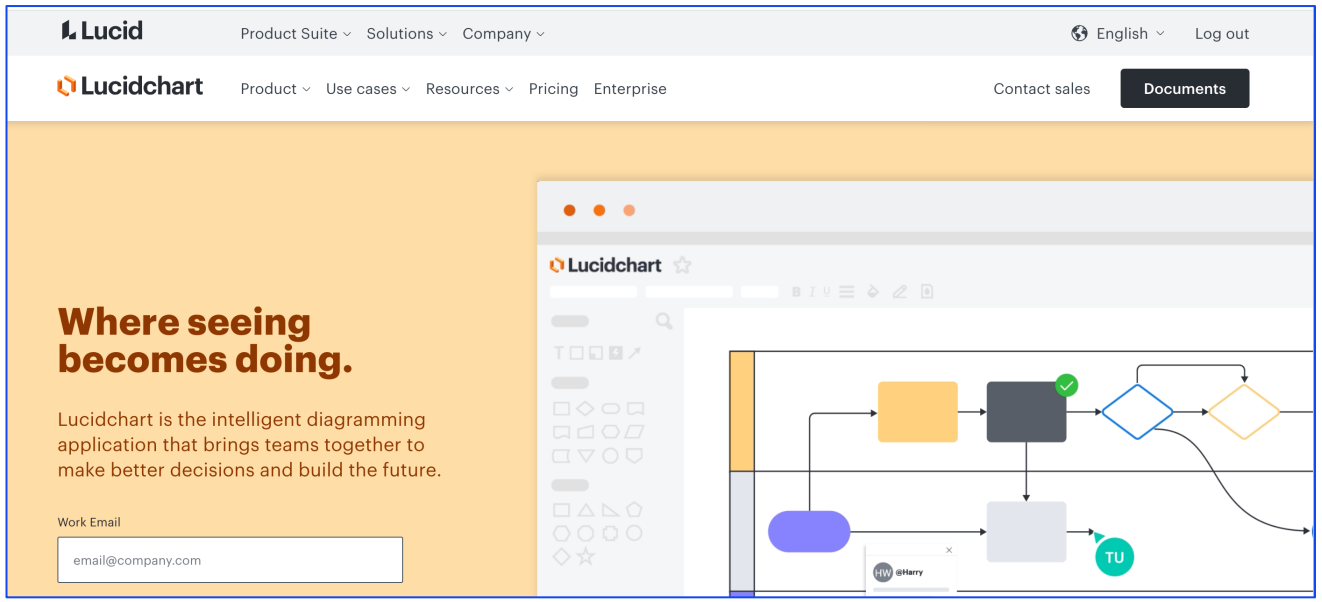
1.**Identify all the entities** in the system. An entity should appear only once in a particular diagram. Create rectangles for all entities and name them properly.

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 Online
Product NEW
Features
Resources
Solutions
Templates
Pricing
Sign up
Log in
Online Chen Entity Relationship Diagram Tool
Create Chen Entity Relationship Diagram online, with an online Chen Entity Relationship Diagram software
https://online.visual-paradigm.com/diagrams/
Student Enrollment System ERD
Share
Export
Search
Favorites
Entity Relationship (Chen Notation)
Item
Student ID
Enrollment Date
Course Name
Enrollment
enrolls
Student
Address
Telephone
Name
Date of Birth
NID
Time
CC ID

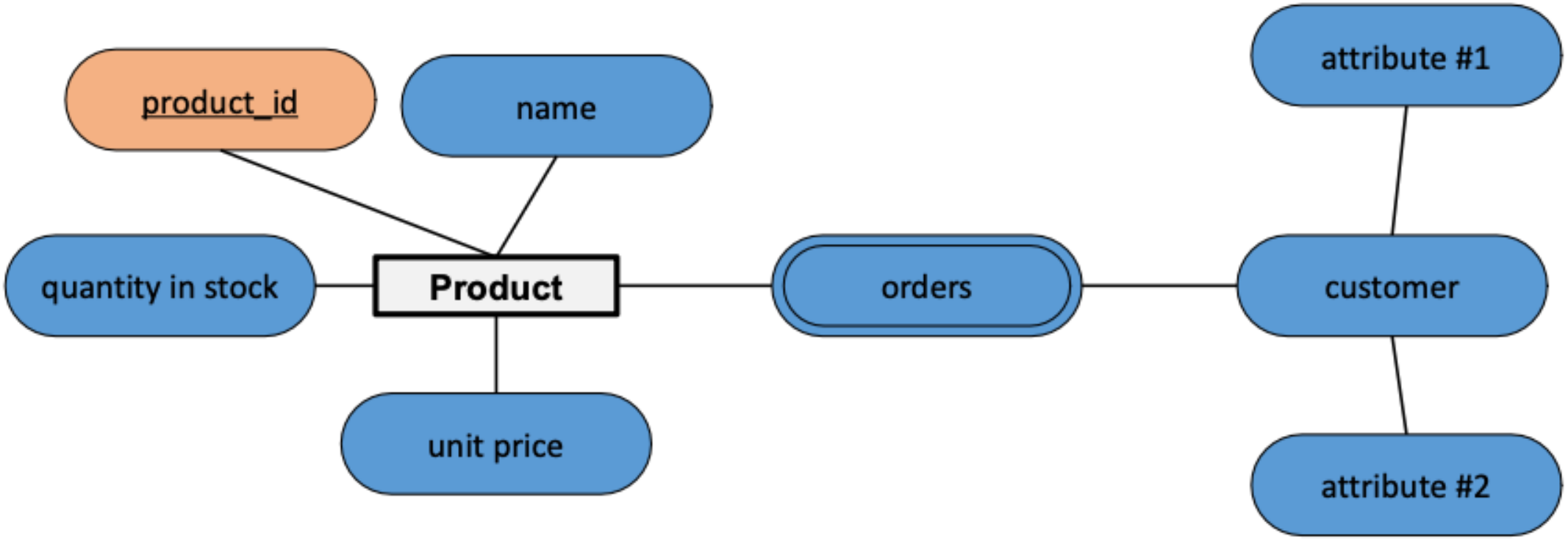

Lucid
Product Suite
Solutions
Company
English
Log out
Lucidchart
Product
Use cases
Resources
Pricing
Enterprise
Contact sales
Documents
Where seeing becomes doing.
Lucidchart is the intelligent diagramming application that brings teams together to make better decisions and build the future.
Work Email
email@company.com
Lucidchart
TU

ER DIAGRAM

Company Data Requirements

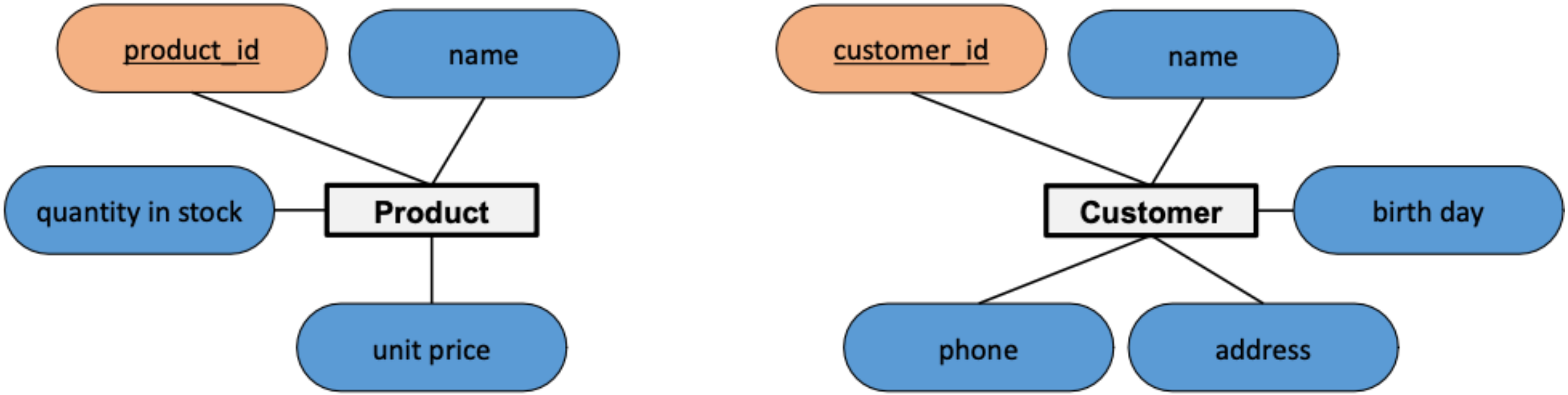
- The company want to store it product. Each product has a name, quantity in stock, and unit price.
- The company sell their products to customers. Each customer has a name, birth day, phone, address, city, state, and loyalty point.
- Each order placed by the customer need to have order date, status, 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



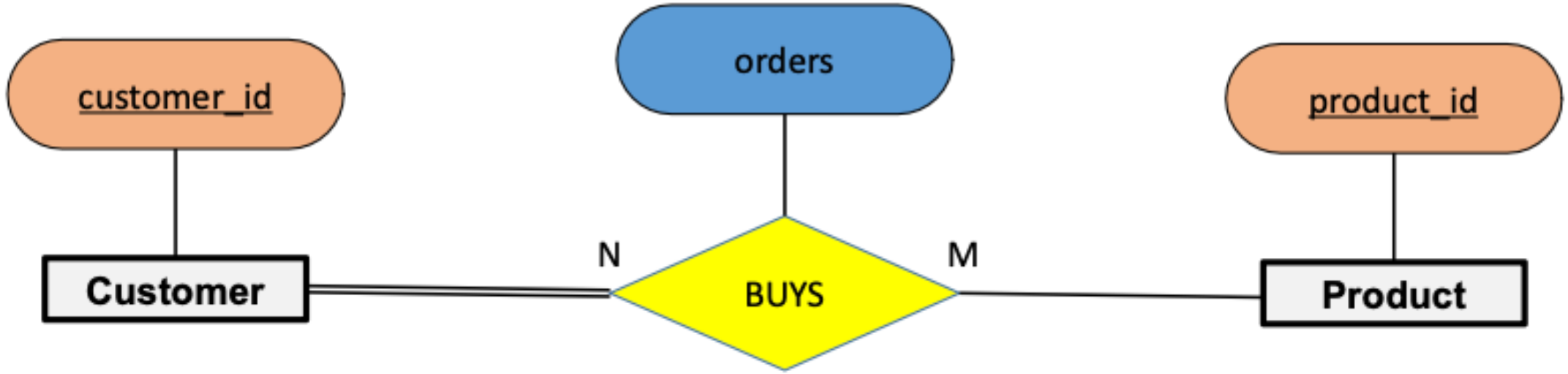
Multi-valued attribute is a bad design.
The customer is not depend on the product.

ER DIAGRAM



The customer now become it own entity

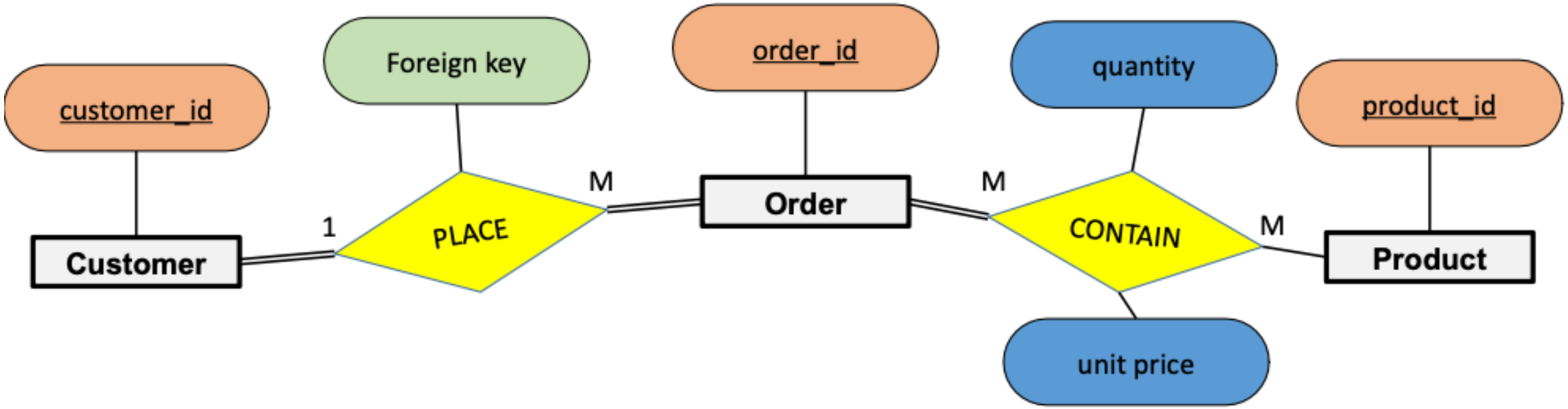
ER DIAGRAM



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

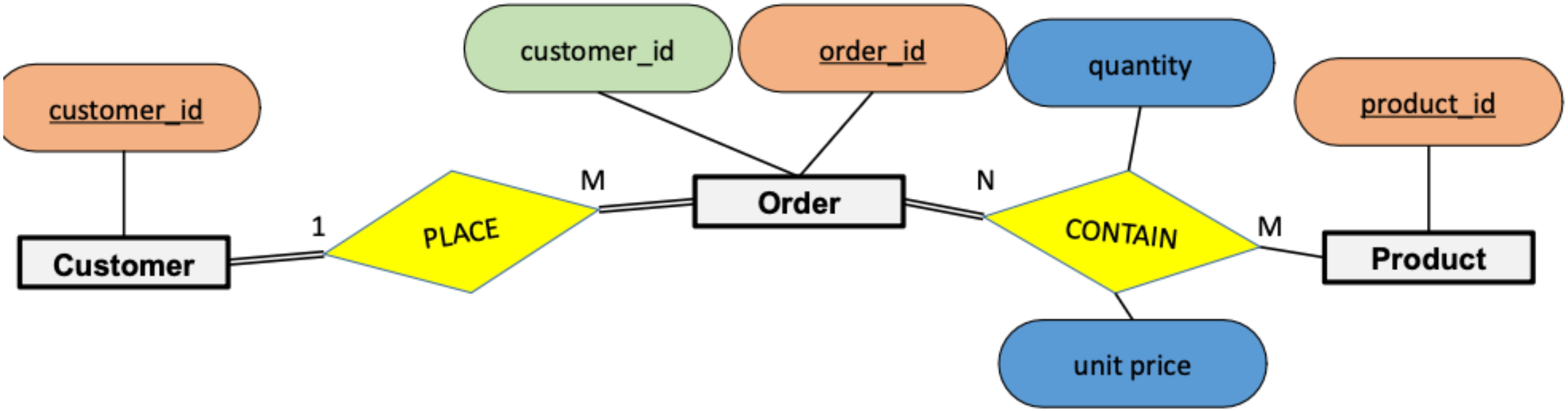


Which Entity will hold the foreign key attribute?

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

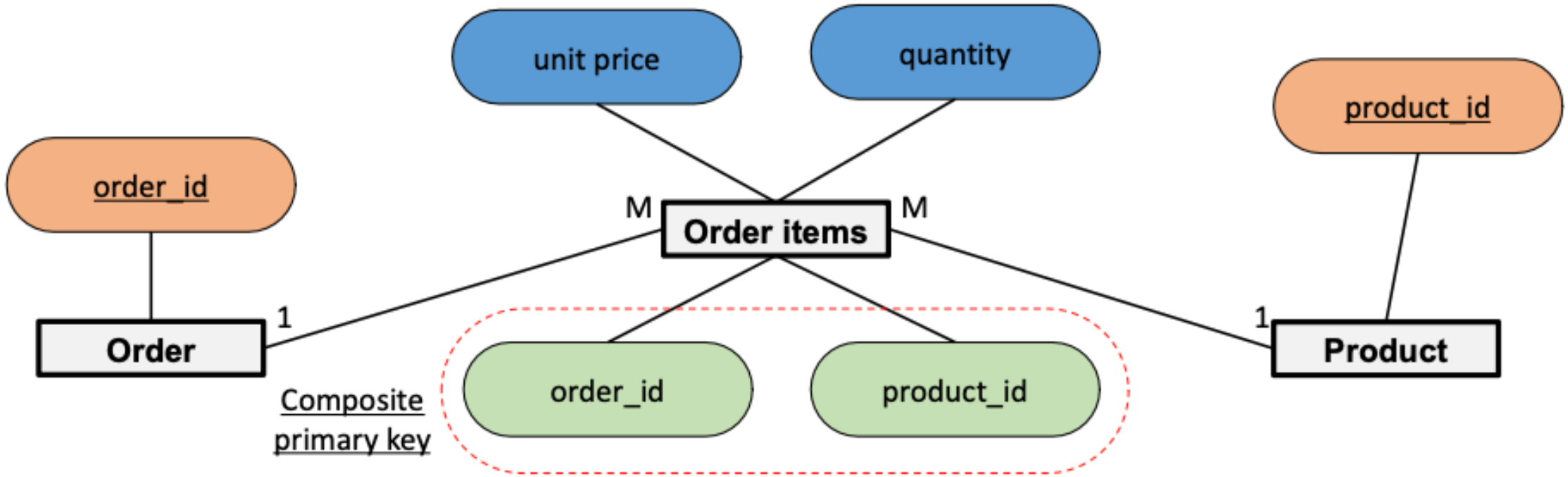
ER DIAGRAM

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



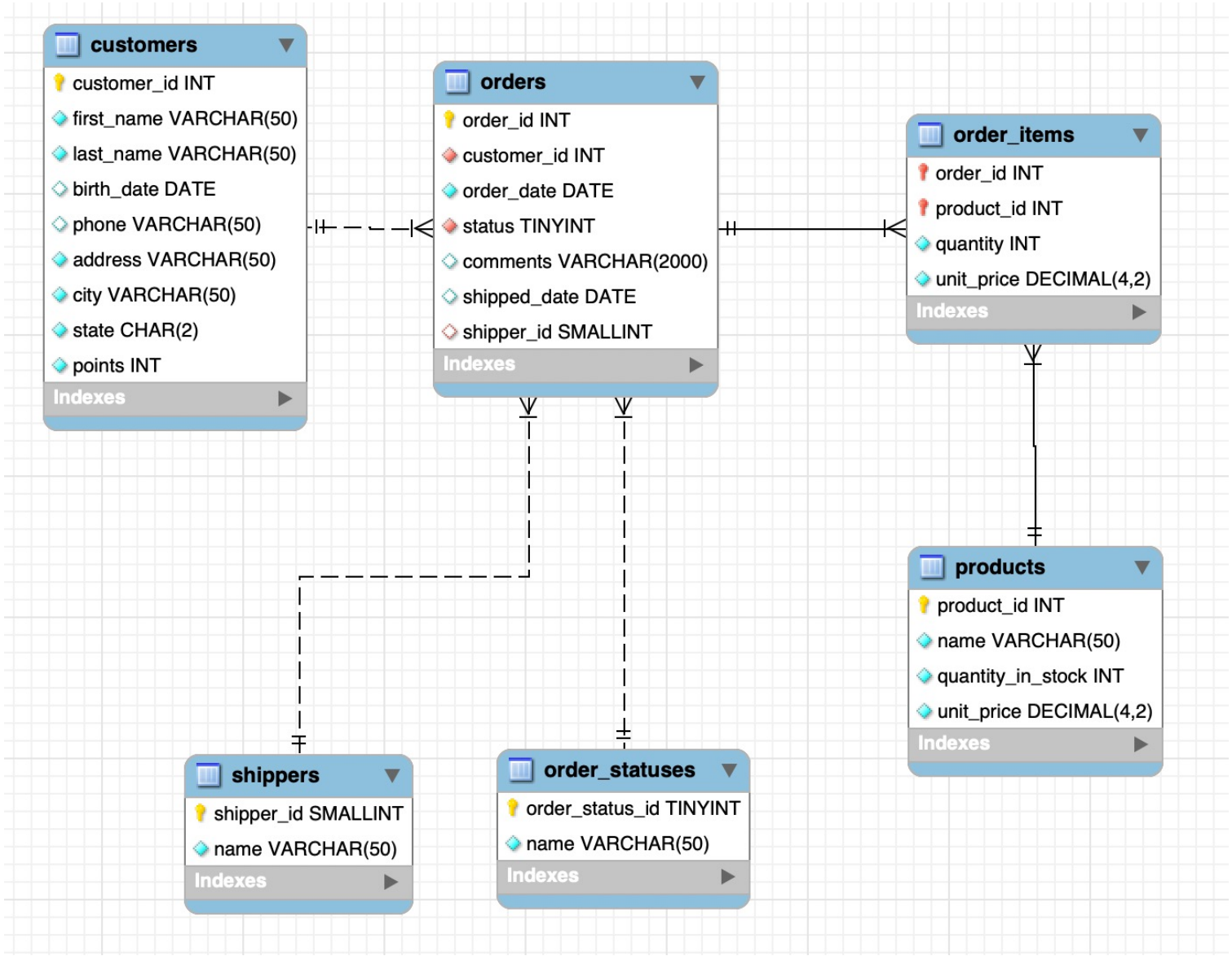
We have the same problem with quantity and unit price attributes that come from a many-to-many relationship.

ER DIAGRAM



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

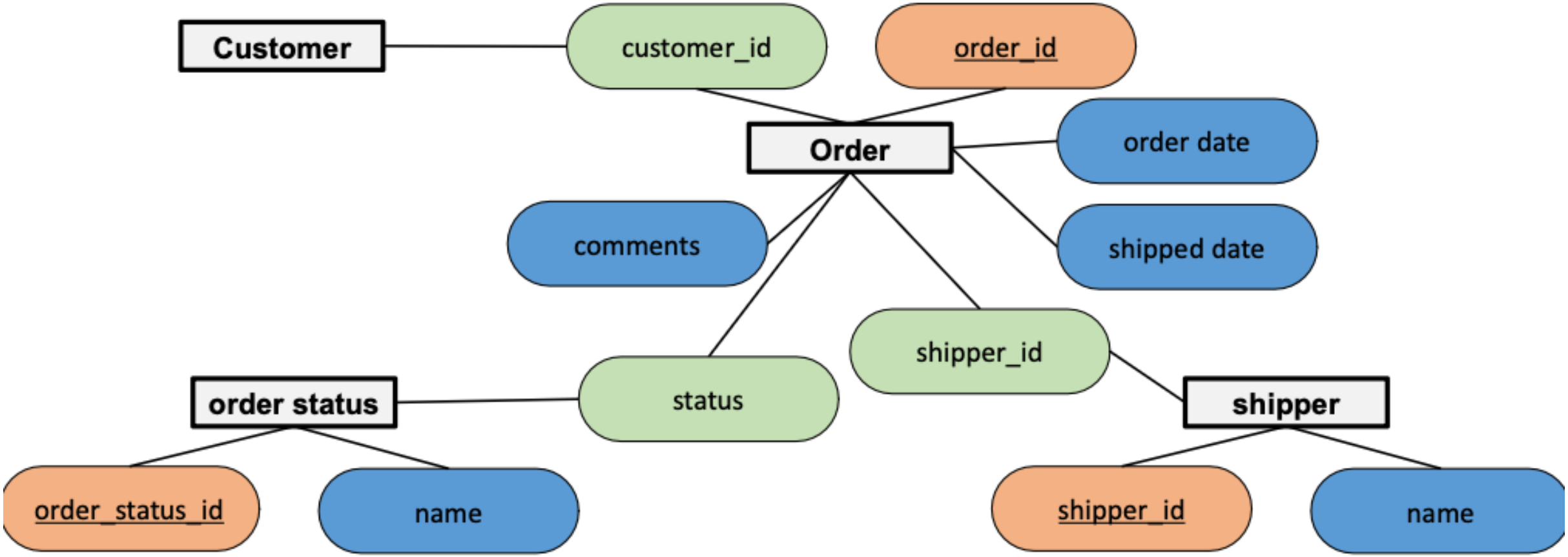
Crow's Foot ERD



IMPLEMENT

ERD

IMPLEMENT ER DIAGRAM

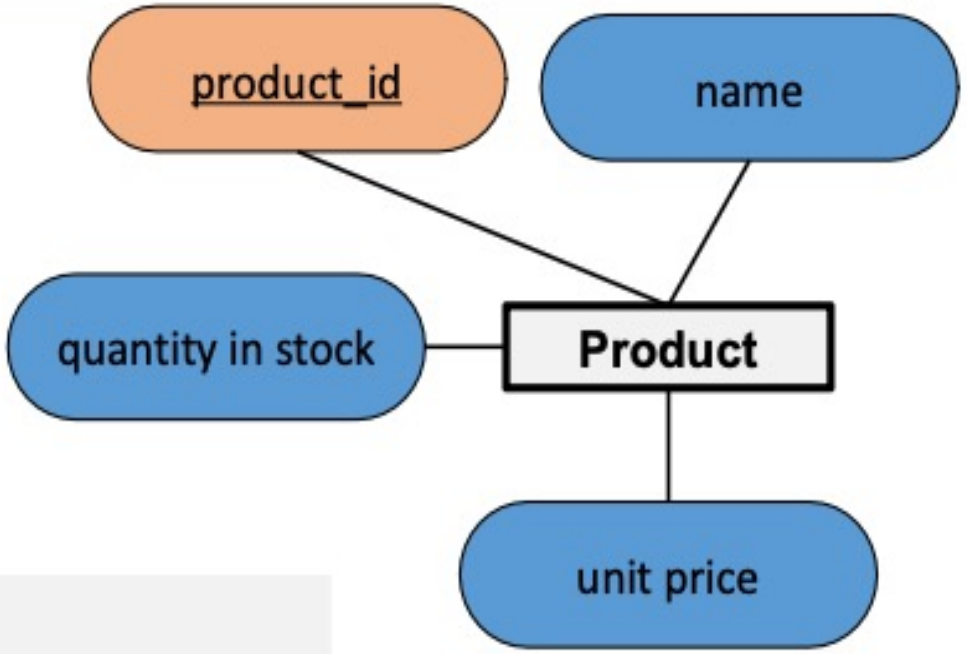


Again the order status and shipper entity don't have a foreign key so they should be created first then the order entity

IMPLEMENT ER DIAGRAM

Create our database schema

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



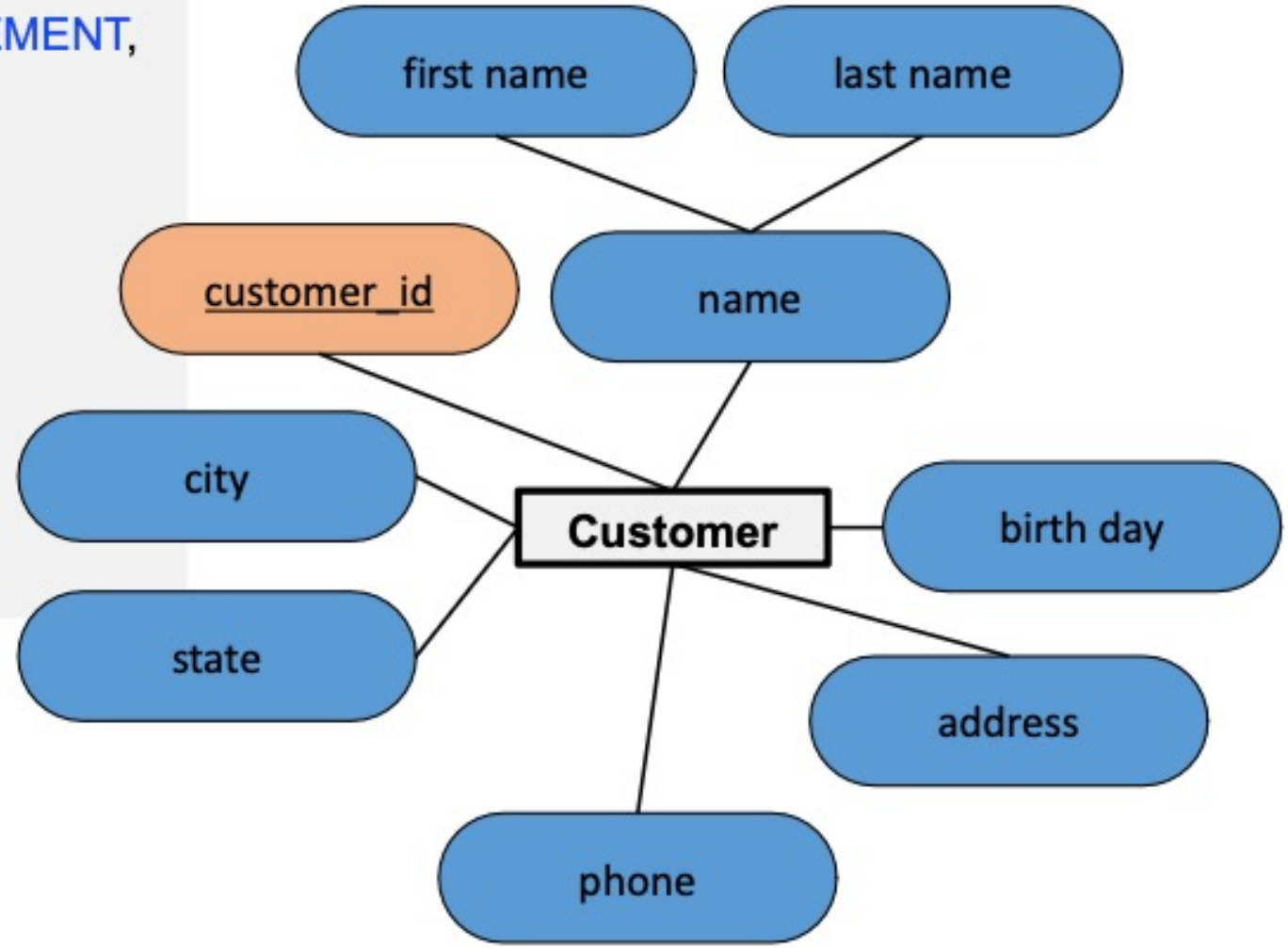
Create our Products table

```
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
)
```

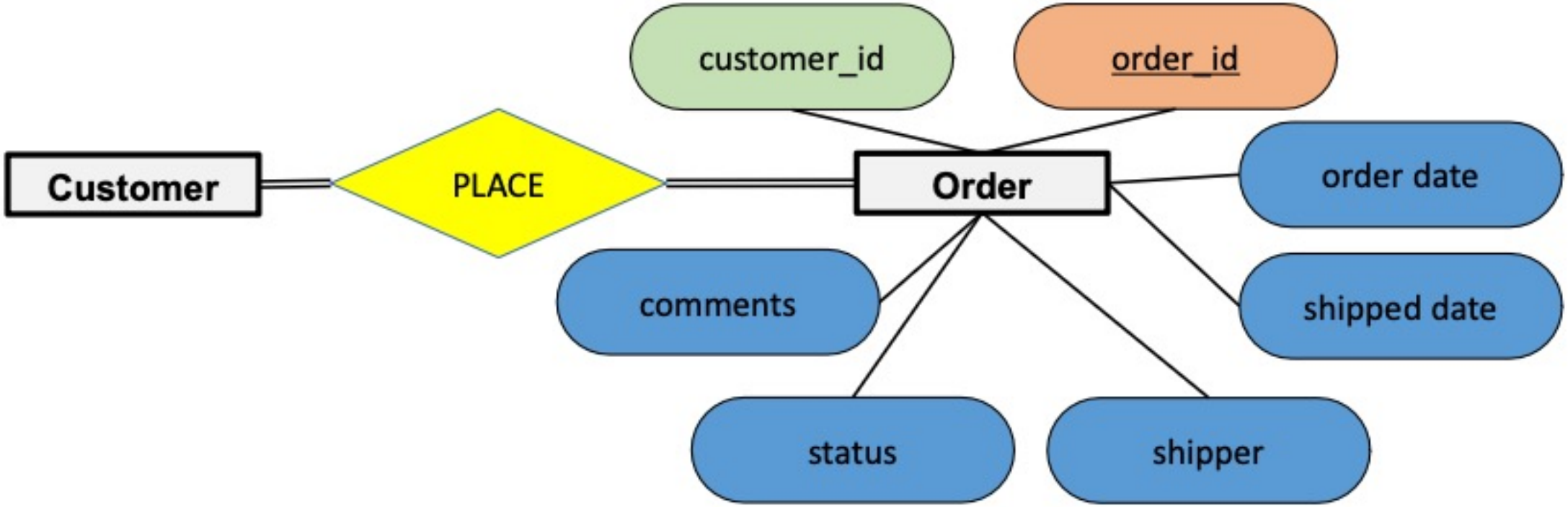
IMPLEMENT ER DIAGRAM

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)
);
```



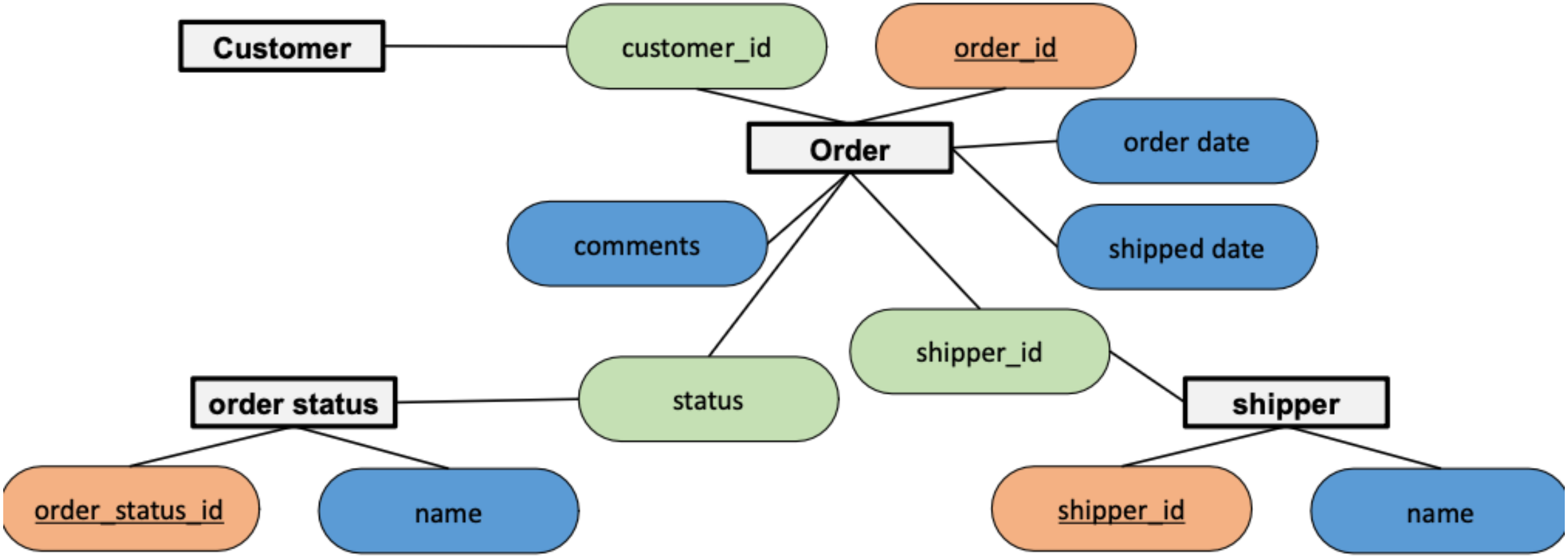
IMPLEMENT ER DIAGRAM



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



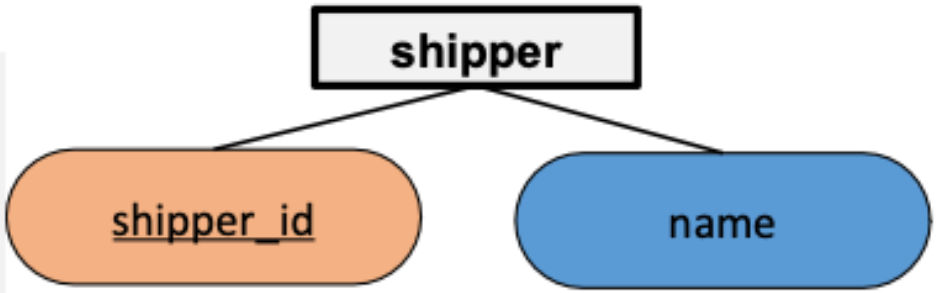
Again the order status and shipper entity don't have a foreign key so they should be created first then the order entity

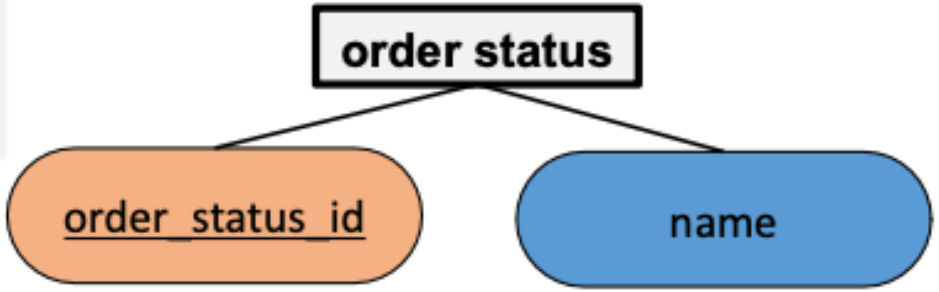
IMPLEMENT ER DIAGRAM

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)
);
```



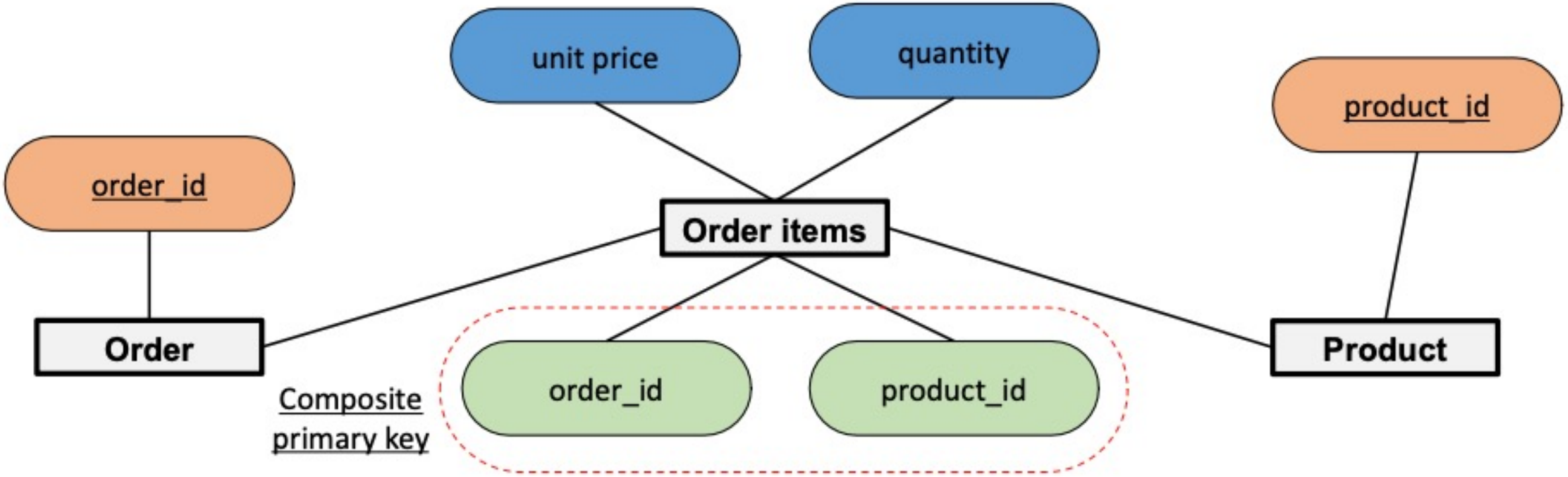


IMPLEMENT ER DIAGRAM

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
);
```

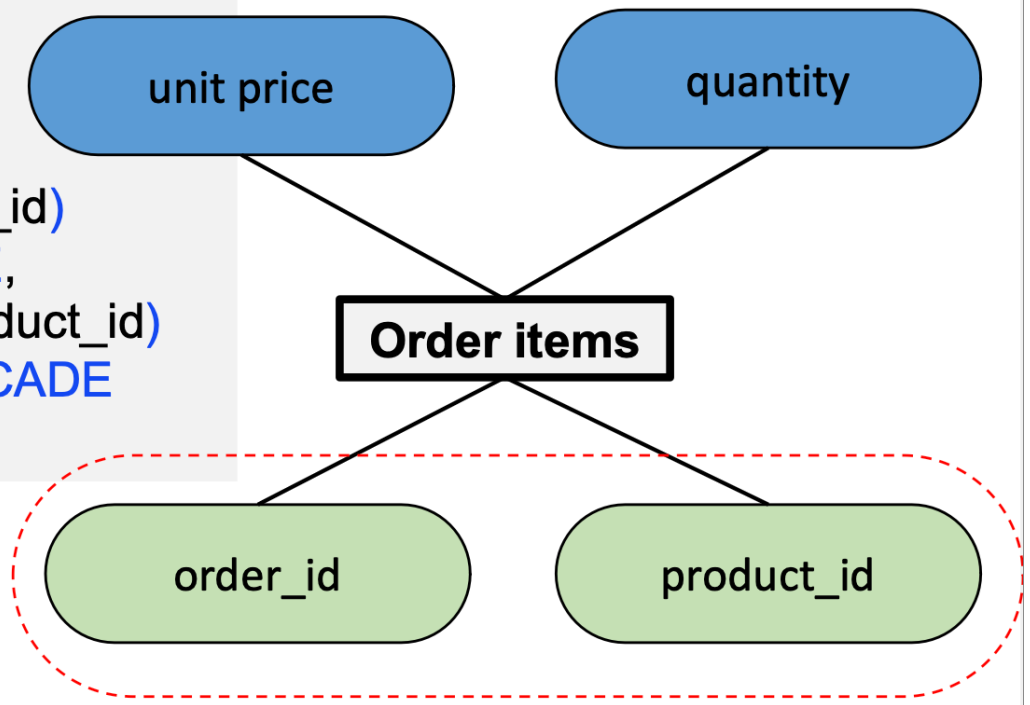
IMPLEMENT ER DIAGRAM



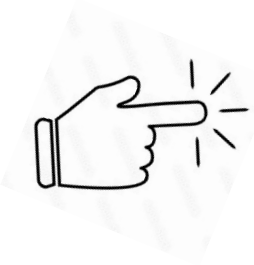
IMPLEMENT ER DIAGRAM

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
);
```



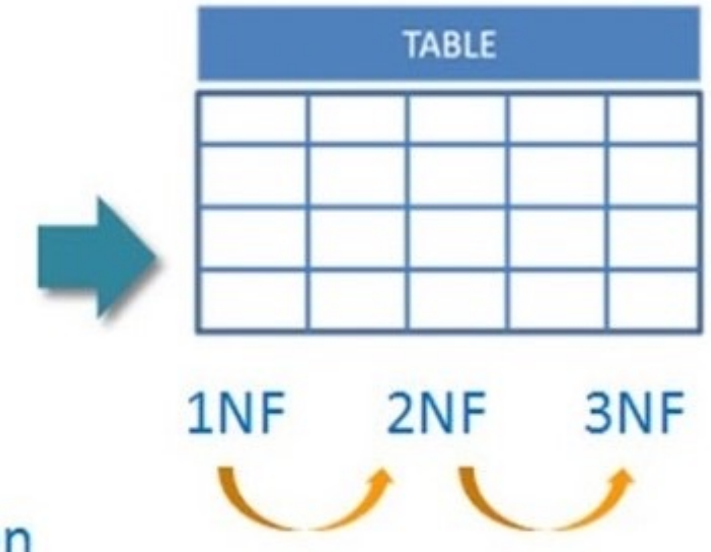
Outline

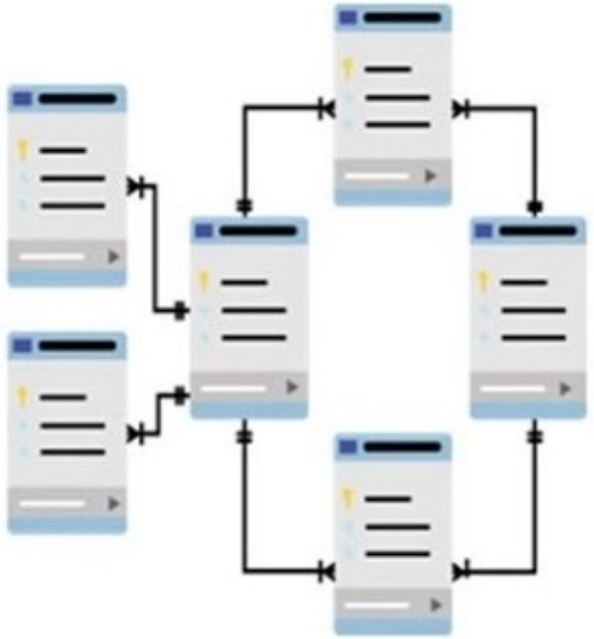


- Introduction to Entity Relationship Diagram
- Introduction to Database Normalization
- Advanced SQL Queries
- Kahoot Quiz
- Summary

Database Normalization

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

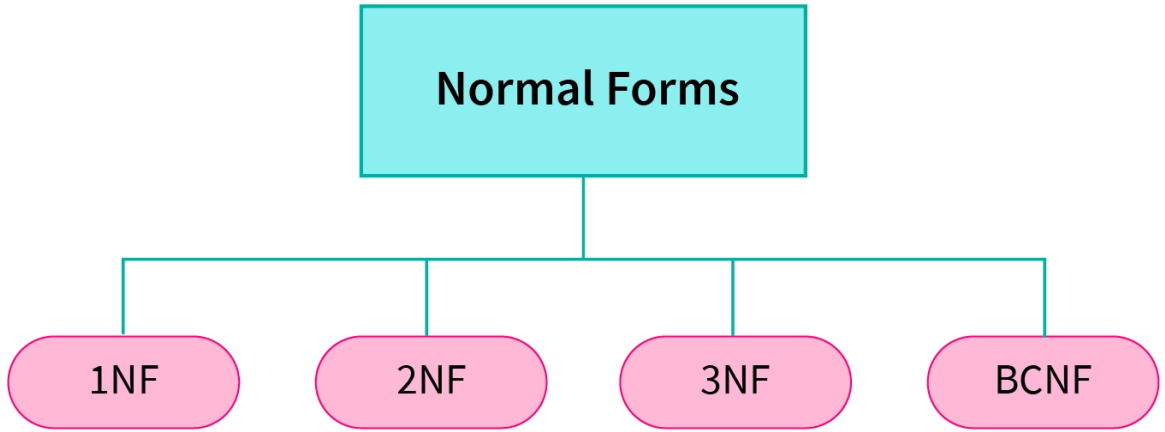
If a student drops out or graduates and their record is deleted, you might unintentionally lose information about the courses they were enrolled in if those courses are only stored in the same table.

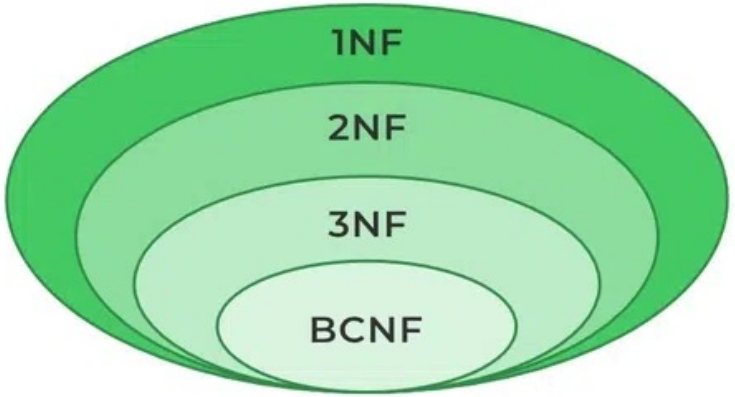
For example, if Alice (StudentID 1) is deleted:

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 candidate 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, LHS is the super key.

"LHS" and "RHS" refer to "Left-Hand Side" and "Right-Hand Side," respectively

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

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

Every cell should only have one single value.
Every record should be unique.

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.

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

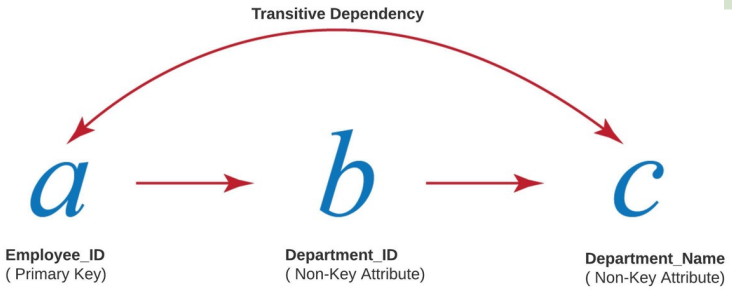
Student_Project

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

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

Student

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

Student_Location

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

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 Code	Project ID	Project Leader
101	P03	Andy
101	P01	Peter
102	P04	Marry
103	P02	Smith

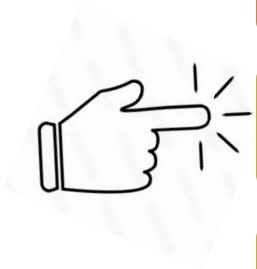
Student_Project

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

Project_Leader

Project Leader	Project ID
Andy	P03
Peter	P01
Marry	P04
Smith	P02

Outline



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

SQL Queries

INNER JOIN

AI VIET NAM
@aivietnam.edu.vn

INNER JOIN

order_id	customer_id	order_date	status	comments	shipped_date	shipper_id
1	6	2019-01-30	1	NULL	NULL	NULL
2	7	2023-05-30	2	NULL	2018-08-03	4
3	8	2017-12-01	1	NULL	NULL	NULL
4	2	2017-01-22	1	NULL	NULL	NULL
5	5	2017-08-25	2		2017-08-26	3
6	10	2018-11-18	1	Aliquam erat volutpat. In congue.	NULL	NULL
7	2	2018-09-22	2	NULL	2018-09-23	4
8	5	2018-06-08	1	Mauris enim leo, rhoncus sed, vestibulum sit am…	NULL	NULL
9	10	2017-07-05	2	Nulla mollis molestie lorem. Quisque ut erat.	2017-07-06	1
10	6	2018-04-22	2	NULL	2018-04-23	2

How can we query across multible tables? For example: How can we retrieve which customers placing the order #5?

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	MA	947
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	CO	457
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	TX	3073
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	TN	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	GA	1486
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796

INNER JOIN

1/ Get the first and last name of every customer placing orders

```
SELECT *
FROM orders
INNER JOIN customers ON  orders.customer_id = customers.customer_id;
```

order_id	customer_id	order_date	status	comments	shipped_date	shipper_id	customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	6	2019-01-30	1	NULL	NULL	NULL	6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	TX	3073
2	7	2023-05-30	2	NULL	2018-08-03	4	7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
3	8	2017-12-01	1	NULL	NULL	NULL	8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	TN	205
4	2	2017-01-22	1	NULL	NULL	NULL	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	MA	947
5	5	2017-08-25	2		2017-08-26	3	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	10	2018-11-18	1	Aliquam erat volutpat. In congue.	NULL	NULL	10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796
7	2	2018-09-22	2	NULL	2018-09-23	4	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	MA	947
8	5	2018-06-08	1	Mauris enim leo, rhoncus sed, vestibulum sit am...	NULL	NULL	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
9	10	2017-07-05	2	Nulla mollis molestie lorem. Quisque ut erat.	2017-07-06	1	10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796
10	6	2018-04-22	2	NULL	2018-04-23	2	6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	TX	3073

You can see that the customer_id from the customers table now match with the customer_id from the orders table.

Let refine our query.

INNER JOIN

1/ Get the first and last name of every customer placing orders

```
SELECT order_id, first_name, last_name
FROM orders
INNER JOIN customers ON  orders.customer_id = customers.customer_id;
```

order_id	first_name	last_name
4	Ines	Brushfield
7	Ines	Brushfield
5	Clemmie	Betchley
8	Clemmie	Betchley
1	Elka	Twiddell
10	Elka	Twiddell
2	Ilene	Dowson
3	Thacher	Naseby
6	Levy	Mynett
9	Levy	Mynett

We don’t want to keep repeating our table name every time. We can use Alias to keep our query short.

Let refine our query.

INNER JOIN

1/ Get the first and last name of every customer placing orders

```
SELECT order_id, first_name, last_name
FROM orders o
INNER JOIN customers c
      ON  o.customer_id = c.customer_id;
```

order_id	first_name	last_name
4	Ines	Brushfield
7	Ines	Brushfield
5	Clemmie	Betchley
8	Clemmie	Betchley
1	Elka	Twiddell
10	Elka	Twiddell
2	Ilene	Dowson
3	Thacher	Naseby
6	Levy	Mynett
9	Levy	Mynett

INNER JOIN

2/ Join the order_items table with the products table return the product_id, name, quantity, unit_price, and price = quantity * unit_price.

```
SELECT oi.product_id, name, quantity, oi.unit_price, quantity * oi.unit_price AS price
FROM orders_items oi
JOIN products p
      ON  oi.product_id = p.product_id;
```

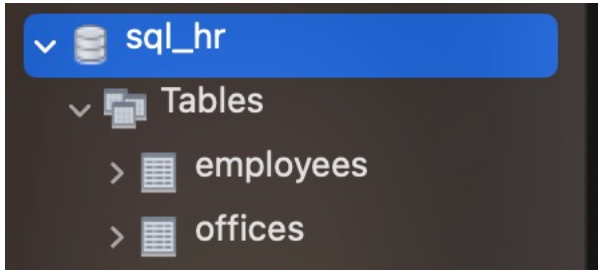
order_id	product_id	name	quantity	unit_price	price
2	1	Foam Dinner Plate	2	9.10	18.20
6	1	Foam Dinner Plate	4	8.65	34.60
10	1	Foam Dinner Plate	10	6.01	60.10
5	2	Pork - Bacon,back Peameal	3	9.89	29.67
6	2	Pork - Bacon,back Peameal	4	3.28	13.12
3	3	Lettuce - Romaine, Heart	10	9.12	91.20
4	3	Lettuce - Romaine, Heart	7	6.99	48.93
6	3	Lettuce - Romaine, Heart	4	7.46	29.84
7	3	Lettuce - Romaine, Heart	7	9.17	64.19
1	4	Brocolinni - Gaylan, Chinese	4	3.74	14.96
2	4	Brocolinni - Gaylan, Chinese	4	1.66	6.64
6	5	Sauce - Ranch Dressing	1	3.45	3.45
8	5	Sauce - Ranch Dressing	2	6.94	13.88
2	6	Petit Baguette	2	2.94	5.88
9	6	Petit Baguette	5	7.28	36.40
8	8	Island Oasis - Raspberry	2	8.59	17.18
10	9	Longan	9	4.28	38.52
4	10	Broom - Push	7	6.40	44.80

SQL Queries

SELF JOIN

SELF JOIN

Let take a look at the sql_hr schema, here is the data from the employees tables



employee_id	first_name	last_name	job_title	salary	reports_to	office_id
33391	D'arcy	Nortunen	Account Executive	62871	37270	1
37270	Yovonnda	Magrannell	Executive Secretary	63996	NULL	10
37851	Sayer	Matterson	Statistician III	98926	37270	1
40448	Mindy	Crissil	Staff Scientist	94860	37270	1
56274	Keriann	Alloisi	VP Marketing	110150	37270	1
63196	Alaster	Scutchin	Assistant Professor	32179	37270	2
67009	North	de Clerc	VP Product Management	114257	37270	2
67370	Elladine	Rising	Social Worker	96767	37270	2
68249	Nisse	Voysey	Financial Advisor	52832	37270	2
72540	Guthrey	Iacopetti	Office Assistant I	117690	37270	3
72913	Kass	Hefferan	Computer Systems Ana…	96401	37270	3
75900	Virge	Goodrum	Information Systems M…	54578	37270	3
76196	Mirilla	Janowski	Cost Accountant	119241	37270	3
80529	Lynde	Aronson	Junior Executive	77182	37270	4
80679	Mildrid	Sokale	Geologist II	67987	37270	4
84791	Hazel	Tarbert	General Manager	93760	37270	4
95213	Cole	Kesterton	Pharmacist	86119	37270	4
96513	Theresa	Binney	Food Chemist	47354	37270	5
98374	Estrellita	Daleman	Staff Accountant IV	70187	37270	5
115357	Ivy	Fearey	Structural Engineer	92710	37270	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL

This is the ID of the manager for that employee.

Now we can see that the manager ID is related back to the same table.

So how can we get the name of each employee and their manager?

SELF JOIN

1/ Get the first and last name of every employees and their respective manager

```
USE sql_hr;

SELECT
        e.employee_id,
        e.first_name,
        e.last_name,
        m.employee_id,
        m.first_name,
        m.last_name
FROM employees e
JOIN employees m
        ON  e.reports_to = m.employees_id;
```

employee_id	first_name	last_name	manager_id	first_name	last_name
33391	D'arcy	Nortunen	37270	Yovonnda	Magrannell
37851	Sayer	Matterson	37270	Yovonnda	Magrannell
40448	Mindy	Crissil	37270	Yovonnda	Magrannell
56274	Keriann	Alloisi	37270	Yovonnda	Magrannell
63196	Alaster	Scutchin	37270	Yovonnda	Magrannell
67009	North	de Clerc	37270	Yovonnda	Magrannell
67370	Elladine	Rising	37270	Yovonnda	Magrannell
68249	Nisse	Voysey	37270	Yovonnda	Magrannell
72540	Guthrey	Iacopetti	37270	Yovonnda	Magrannell
72913	Kass	Hefferan	37270	Yovonnda	Magrannell
75900	Virge	Goodrum	37270	Yovonnda	Magrannell
76196	Mirilla	Janowski	37270	Yovonnda	Magrannell
80529	Lynde	Aronson	37270	Yovonnda	Magrannell
80679	Mildrid	Sokale	37270	Yovonnda	Magrannell
84791	Hazel	Tarbert	37270	Yovonnda	Magrannell
95213	Cole	Kesterton	37270	Yovonnda	Magrannell
96513	Theresa	Binney	37270	Yovonnda	Magrannell
98374	Estrellita	Daleman	37270	Yovonnda	Magrannell
115357	Ivy	Fearey	37270	Yovonnda	Magrannell

NOTE: When doing a self join we must make sure that we use different alias's for our tables

SQL Queries

JOINNING MUTIPLE TABLES

AI VIET NAM
@aivietnam.edu.vn

JOINNING MUTIPLE TABLES

Let go back to our sql_store schema and take a look at orders tables

order_id	customer_id	order_date	status	comments	shipped_date	shipper_id
1	6	2019-01-30	1	NULL	NULL	NULL
2	7	2023-05-30	2	NULL	2018-08-03	4
3	8	2017-12-01	1	NULL	NULL	NULL
4	2	2017-01-22	1	NULL	NULL	NULL
5	5	2017-08-25	2		2017-08-26	3
6	10	2018-11-18	1	Aliquam erat volutpat. In congue.	NULL	NULL
7	2	2018-09-22	2	NULL	2018-09-23	4
8	5	2018-06-08	1	Mauris enim leo, rhoncus sed, vestibulum sit amet, cursus id, turpis.	ULL	
9	10	2017-07-05	2	Nulla mollis molestie lorem. Quisque ut erat.	2017-07-06	1
10	6	2018-04-22	2	NULL	2018-04-23	2

This point to our customers table

This point to our orders_statuses table

How can we retrieve the infomation from three tables? For example: We want to get the order_id, order_date from the orders table, the first and last name of the customer placing the order, and the status of that order?

JOINNING MUTIPLE TABLES

1/ Retrieve the order_id, order_date, first_name, last_name, and status of each orders

```
USE sql_store;

SELECT
        o.order_id,
        o.order_date,
        c.last_name,
        c.first_name,
        s.name AS status,
FROM orders o
JOIN customers c
        ON  o.customer_id = c.customer_id
JOIN order_statuses s
        ON o.status = s.order_status_id;
```

order_id	order_date	last_name	first_name	status
8	2018-06-08	Betchley	Clemmie	Processed
6	2018-11-18	Mynett	Levy	Processed
4	2017-01-22	Brushfield	Ines	Processed
3	2017-12-01	Naseby	Thacher	Processed
1	2019-01-30	Twiddell	Elka	Processed
10	2018-04-22	Twiddell	Elka	Shipped
9	2017-07-05	Mynett	Levy	Shipped
7	2018-09-22	Brushfield	Ines	Shipped
5	2017-08-25	Betchley	Clemmie	Shipped
2	2023-05-30	Dowson	Ilene	Shipped

AI VIET NAM
@aivietnam.edu.vn

JOINNING MUTIPLE TABLES

2/ From the invoicing database, retrieve respective to payments table

- **payment_id, date, amount from payments table**
- **name, phone from clients table**
- **the invoice number, invoice_total, payment_total from invoices table**
- **and the payment method name from payment_methods table**

payment_id	date	amount	name	phone	number	invoice_total	payment_total	payment_meth...
1	2019-02-12	8.18	Topiclounge	971-888-9129	03-898-6735	175.32	8.18	Credit Card
2	2019-01-03	74.55	Vinte	315-252-7305	75-587-6626	157.78	74.55	Credit Card
3	2019-01-11	0.03	Yadel	415-144-6037	20-848-0181	126.15	0.03	Credit Card
4	2019-01-26	87.44	Topiclounge	971-888-9129	41-666-1035	135.01	87.44	Credit Card
5	2019-01-15	80.31	Yadel	415-144-6037	55-105-9605	167.29	80.31	Credit Card
6	2019-01-15	68.10	Yadel	415-144-6037	33-615-4694	126.38	68.10	Credit Card
7	2019-01-08	32.77	Topiclounge	971-888-9129	52-269-9803	180.17	42.77	Credit Card
8	2019-01-08	10.00	Topiclounge	971-888-9129	52-269-9803	180.17	42.77	Cash

```
USE sql_invoicing;
SELECT
        p.payment_id, p.date, p.amount,
        c.name, c.phone,
        i.number, i.invoice_total, i.payment_total,
        pm.name AS payment_method
FROM payments p
JOIN clients c ON p.client_id = c.client_id
JOIN invoices i ON p.invoice_id = i.invoice_id
JOIN payment_methods pm ON p.payment_method = pm.payment_method_id;
```

SQL Queries

COMPOUND JOIN CONDITION

COMPOUND JOIN CONDITION

Let go back to our sql_store schema and take a look at orders_items tables

order_id	product_id	quantity	unit_price
1	4	4	3.74
2	1	2	9.10
2	4	4	1.66
2	6	2	2.94
3	3	10	9.12
4	3	7	6.99
4	10	7	6.40
5	2	3	9.89
6	1	4	8.65
6	2	4	3.28
6	3	4	7.46
6	5	1	3.45
7	3	7	9.17
8	5	2	6.94
8	8	2	8.59
9	6	5	7.28
10	1	10	6.01
10	9	9	4.28

Both of these column are not unique

This is call composite primary key, that mean we need both column to uniquely identify a record.

note_id	order_Id	product_id	note
1	1	2	first note
2	1	2	second note

So how can we join a table with composite key?

COMPOUND JOIN CONDITION

1/ Retrieve the note for the respective order item

```
USE sql_store;

SELECT *
FROM order_items o
JOIN order_item_notes n
        ON o.order_id = n.order_id
        AND o.product_id = n.product_id;
```

order_id	product_id	quantity	unit_price	note_id	order_Id	product_id	note
2	6	2	2.94	1	2	6	first note
5	2	3	9.89	2	5	2	second note

SQL Queries

IMPLICIT JOIN SYNTAX

IMPLICIT JOIN SYNTAX

Let see an example JOIN statment.

```
SELECT *
FROM orders o
JOIN customers c
      ON o.customer_id = c. customer_id;
```

And it Implicit Join Syntax.

```
SELECT *
FROM orders o, customers c
WHERE o.customer_id = c. customer_id;
```

These two query are equivalent but it is recommended to use the explicit Join Syntax.

order_id	customer_id	order_date	status	comments	shipped_date	shipper_id	customer_id	first_name	last_name	birth_date	phone	address	city	s...	po...
1	6	2019-01-30	1	NULL	NULL	NULL	6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	TX	3073
2	7	2023-05-30	2	NULL	2018-08-03	4	7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
3	8	2017-12-01	1	NULL	NULL	NULL	8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	TN	205
4	2	2017-01-22	1	NULL	NULL	NULL	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	MA	947
5	5	2017-08-25	2		2017-08-26	3	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	10	2018-11-18	1	Aliquam er...	NULL	NULL	10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796
7	2	2018-09-22	2	NULL	2018-09-23	4	2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	MA	947
8	5	2018-06-08	1	Mauris eni...	NULL	NULL	5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
9	10	2017-07-05	2	Nulla molli...	2017-07-06	1	10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796
10	6	2018-04-22	2	NULL	2018-04-23	2	6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	TX	3073

IMPLICIT JOIN SYNTAX

The reason being, if we use the Implicit Syntax and forget to add the WHERE condition we can produce an CROSS JOIN.

```
SELECT c.first_name AS customer,
       p.name AS product
FROM orders o, customers c;
```

```
SELECT c.first_name AS customer,
       p.name AS product
FROM orders o
CROSS JOIN customers c;
```

customer	product
Babara	Longan
Babara	Island Oasis - Raspberry
Babara	Sweet Pea Sprouts
Babara	Petit Baguette
Babara	Sauce - Ranch Dressing
Babara	Brocolinni - Gaylan, Chinese
Babara	Lettuce - Romaine, Heart
Babara	Pork - Bacon,back Pearneal
Babara	Foam Dinner Plate
Ines	Broom - Push
Ines	Longan
Ines	Island Oasis - Raspberry
Ines	Sweet Pea Sprouts
Ines	Petit Baguette

CROSS JOIN will return every records from the left table combining with every records from the right table.

SQL Queries

OUTER JOIN

OUTER JOIN

Let see an example JOIN statement.

```
SELECT
        c.customer_id,
        c.first_name,
        o.order_id
FROM customers c
JOIN orders o
        ON c.customer_id = o. customer_id
ORDER BY c.customer_id;
```

customer_id	first_name	order_id
2	Ines	4
2	Ines	7
5	Clemmie	5
5	Clemmie	8
6	Elka	1
6	Elka	10
7	Ilene	2
8	Thacher	3
10	Levy	6
10	Levy	9

We can see that only the customer who have place an order is shown in our result.

What if we want to see all customer?

AI VIET NAM
@aivietnam.edu.vn

OUTER JOIN

```
SELECT
        c.customer_id,
        c.first_name,
        o.order_id
FROM customers c
LEFT JOIN orders o
        ON c.customer_id = o. customer_id
ORDER BY c.customer_id;
```

When using LEFT JOIN all records from the left table i.e. customers are returned whether the condition is true or not.

All customer id records will be returned

customer_id	first_name	order_id
1	Babara	NULL
2	Ines	4
2	Ines	7
3	Freddi	NULL
4	Ambur	NULL
5	Clemmie	5
5	Clemmie	8
6	Elka	1
6	Elka	10
7	Ilene	2
8	Thacher	3
9	Romola	NULL
10	Levy	6
10	Levy	9

OUTER JOIN

```
SELECT
        c.customer_id,
        c.first_name,
        o.order_id
FROM customers c
RIGHT JOIN orders o
        ON c.customer_id = o. customer_id
ORDER BY c.customer_id;
```

When using RIGHT JOIN all records from the right table i.e. orders are returned whether the condition is true or not.

customer_id	first_name	order_id
2	Ines	4
2	Ines	7
5	Clemmie	5
5	Clemmie	8
6	Elka	1
6	Elka	10
7	Ilene	2
8	Thacher	3
10	Levy	6
10	Levy	9

All order id records will be returned

OUTER JOIN

1/ Write a query to get all customer id, their first name, their order id, and shipper name for that order

```
SELECT
        c.customer_id,
        c.first_name,
        o.order_id,
        sh.name AS shipper
FROM customers c
LEFT JOIN orders o
        ON c.customer_id = o. customer_id
LEFT JOIN shippers sh
        ON o.shipper_id = sh.shipper_id
ORDER BY c.customer_id;
```

customer_id	first_name	order_id	shipper
1	Babara	NULL	NULL
2	Ines	4	NULL
2	Ines	7	Mraz, Renner and Nolan
3	Freddi	NULL	NULL
4	Ambur	NULL	NULL
5	Clemmie	5	Satterfield LLC
5	Clemmie	8	NULL
6	Elka	1	NULL
6	Elka	10	Schinner-Predovic
7	Ilene	2	Mraz, Renner and Nolan
8	Thacher	3	NULL
9	Romola	NULL	NULL
10	Levy	6	NULL
10	Levy	9	Hettinger LLC

SQL Queries

USING CLAUSE

AI VIET NAM
@aivietnam.edu.vn

USING CLAUSE

Let return to ours example JOIN statement.

```
SELECT
        c.customer_id,
        c.first_name,
        o.order_id
FROM customers c
JOIN orders o
        ON c.customer_id = o. customer_id
ORDER BY c.customer_id;
```

customer_id	first_name	order_id
2	Ines	4
2	Ines	7
5	Clemmie	5
5	Clemmie	8
6	Elka	1
6	Elka	10
7	Ilene	2
8	Thacher	3
10	Levy	6
10	Levy	9

If both column of the JOIN condition have the same name i.e. "customer_id" we can simplify the query.

```
SELECT
        c.customer_id,
        c.first_name,
        o.order_id
FROM customers c
JOIN orders o
        USING (customer_id)
ORDER BY c.customer_id;
```

USING CLAUSE

Let return to ours example JOIN statement.

```
SELECT
        c.customer_id,
        c.first_name,
        o.order_id
FROM customers c
JOIN orders o
        ON c.customer_id = o. customer_id
ORDER BY c.customer_id;
```

customer_id	first_name	order_id
2	Ines	4
2	Ines	7
5	Clemmie	5
5	Clemmie	8
6	Elka	1
6	Elka	10
7	Ilene	2
8	Thacher	3
10	Levy	6
10	Levy	9

If both colunm of the JOIN condition have the same name i.e. "customer_id" we can simplify the query.

```
SELECT
        c.customer_id,
        c.first_name,
        o.order_id
FROM customers c
JOIN orders o
        USING (customer_id)
ORDER BY c.customer_id;
```

USING CLAUSE

This is extremely useful when writting compound join conditions.

```
SELECT *
FROM order_items o
JOIN order_item_notes n
      ON o.order_id = n.order_id
      AND o.product_id = n.product_id;
```

```
SELECT *
FROM order_items o
JOIN order_item_notes n
      USING (order_id, product_id);
```

order_id	product_id	quantity	unit_price	note_id	order_id	product_id	note
2	6	2	2.94	1	2	6	first note
5	2	3	9.89	2	5	2	second note

SQL Queries

UNIONS

UNIONS

Let see the result of this query.

```
SELECT customer_id, first_name, points
FROM customers;
```

customer_id	first_name	points
1	Babara	2273
2	Ines	947
3	Freddi	2967
4	Ambur	457
5	Clemmie	3675
6	Elka	3073
7	Ilene	1672
8	Thacher	205
9	Romola	1486
10	Levy	796

What if we want to rank them by tier:
- **<2000 point: Bronze**
- **2000 – 3000 point: Silver**
- **>3000 point: Gold.**

customer_id	first_name	points	tier
4	Ambur	457	Bronze
1	Babara	2273	Silver
5	Clemmie	3675	Gold
6	Elka	3073	Gold
3	Freddi	2967	Silver
7	Ilene	1672	Bronze
2	Ines	947	Bronze
10	Levy	796	Bronze
9	Romola	1486	Bronze
8	Thacher	205	Bronze

AI VIET NAM
@aivietnam.edu.vn

UNIONS

Let query the result for Bronze tier.

```
SELECT customer_id, first_name, points, 'Bronze' AS tier
FROM customers
WHERE points < 2000;
```

customer_id	first_name	points	tier
2	Ines	947	Bronze
4	Ambur	457	Bronze
7	Ilene	1672	Bronze
8	Thacher	205	Bronze
9	Romola	1486	Bronze
10	Levy	796	Bronze

UNION

for Silver tier.

```
SELECT customer_id, first_name, points, 'Silver' AS tier
FROM customers
WHERE points BETWEEN 2000 AND 3000;
```

customer_id	first_name	points	tier
1	Babara	2273	Silver
3	Freddi	2967	Silver

UNION

for Gold tier.

```
SELECT customer_id, first_name, points, 'Gold' AS tier
FROM customers
WHERE points > 3000;
```

customer_id	first_name	points	tier
5	Clemmie	3675	Gold
6	Elka	3073	Gold

Summary