Data Modeling

Data Boot Camp

Lesson 9.3

Class Objectives

By the end of today's class, you will be able to:

- Apply data modeling techniques to database design.
- Normalize data.
- Identify data relationships.
- Create visual representations of a database through ERDs.

Data Normalization

Data normalization is the process of restructuring data to a set of defined “normal forms”, which eliminates data redundancy and inconsistencies.

Data Normalization

Three most common forms:

01 First normal form (1NF)

02 Second normal form (2NF)

03 Third normal form (3NF)

First Normal Form (1NF)

- Each field contains a single value.
- Each row is unique.

First normal form (1NF)

Each field in a table row should contain a single value.

Raw Data

VIN	Services Performed	Customer Name	Model
3D7LX39C86G106066	Oil Change	Marcia Jackson	Prius
1FAFP33Z24W147740	Oil Change, Alignment	Patricia Smith	Equinox
KNDJD736875757954	Oil Change, Brake Replacement	Mikhail Ivanov	CRV
3N1AB7AP7EL611028	Transmission Rebuild	Lucas Gonzalez	Tahoe

Normalization

First Normal Form
(Each row is unique)

VIN	Services Performed	Customer Name	Salutation
3D7LX39C86G106066	Oil Change	Marcia Jackson	Prius
1FAFP33Z24W147740	Oil Change	Patricia Smith	Equinox
1FAFP33Z24W147740	Alignment	Patricia Smith	Equinox
KNDJD736875757954	Oil Change	Mikhail Ivanov	CRV
KNDJD736875757954	Brake Replacement	Mikhail Ivanov	CRV
3N1AB7AP7EL611028	Transmission Rebuild	Lucas Gonzalez	Tahoe

Second Normal Form (2NF)

- Must be in 1NF
- Must have one unique identifier for each row (Primary Key)
- All columns are dependent of the Primary Key

Second Normal Form (2NF)

Adds a Primary Key, and all columns are directly dependent on that key. To transform the data below, we'll need separate tables for Vehicle, Customer, and Service Performed.

VIN	Services Performed	Customer Name	Model	Make
3D7LX39C86G106066	Oil Change	Marcia Jackson	Prius	Toyota
1FAFP33Z24W147740	Oil Change	Patricia Smith	Escape	Ford
1FAFP33Z24W147740	Alignment	Patricia Smith	Escape	Ford
KNDJD736875757954	Oil Change	Mikhail Ivanov	CRV	Honda
KNDJD736875757954	Brake Replacement	Mikhail Ivanov	CRV	Honda
3N1AB7AP7EL611028	Transmission Rebuild	Lucas Gonzalez	Tahoe	Chevy

2NF Normalization

Second Normal Form (2NF)

Customer

ID	First	Last
1	Marcia	Jackson
2	Patricia	Smith
3	Mikhail	Ivanov
4	Lucas	Gonzalez

This is the same data in 2NF; note that yellow columns are Primary Keys and blue columns are Foreign Keys which reference the Primary Keys from other tables.

Vehicle

VIN	Customer ID	Model	Make
3D7LX39C86G106066	1	Prius	Toyota
1FAFP33Z24W147740	2	Equinox	Chevy
KNDJD736875757954	3	CRV	Honda
3N1AB7AP7EL611028	4	Tahoe	Chevy

Services

ID	Vehicle	Service
1	3D7LX39C86G106066	Oil Change
2	1FAFP33Z24W147740	Oil Change
3	1FAFP33Z24W147740	Alignment
4	KNDJD736875757954	Oil Change
5	KNDJD736875757954	Brake Replacement
6	3N1AB7AP7EL611028	Transmission Rebuild

1NF -> 2NF

Will generally require to make a new table.

Third Normal Form (3NF)

- Must be in 2NF
- Each non-primary key column depends only on the primary key
- There is no transitive dependency between columns

Transitive dependency is the reliance of a column's value on another column through a third column.

Transitive Dependency

Transitive	If A $\Longrightarrow$ B and B $\Longrightarrow$ C then A $\Longrightarrow$ C
Dependence	• One value relies on another. • City relies on ZIP code; age depends on birthday.

Example

- Consider the **Vehicle** table.
- **VIN** $\Longrightarrow$ **Model**
- **Model** $\Longrightarrow$ **Make**
- Therefore **VIN** $\Longrightarrow$ **Make**

Vehicle

VIN	Customer ID	Model	Make
3D7LX39C86G106066	1	Prius	Toyota
1FAFP33Z24W147740	2	Equinox	Chevy
KNDJD736875757954	3	CRV	Honda
3N1AB7AP7EL611028	4	Tahoe	Chevy

- **Make** is dependent of **Model**, and **Make** is not dependent on **VIN**
- We will automatically know the car **Make** once we know the car **Model**
- There is a Transitive dependency of **Make** with **Model**

→ **Make** is redundant and can be removed

Third Normal Form (3NF) — "Simplify the Relationships"

To break the same data into 3NF, we'd use the tables below.

Vehicle

VIN	Customer	Model	Make
3D7LX39C86G106066	1	Prius	Toyota
1FAFP33Z24W147740	2	Equinox	Chevy
KNDJD736875757954	3	CRV	Honda
3N1AB7AP7EL611028	4	Tahoe	Chevy

Vehicle

VIN	Customer ID	Model
3D7LX39C86G106066	1	1
1FAFP33Z24W147740	2	2
KNDJD736875757954	3	3
3N1AB7AP7EL611028	4	4

Make

ID	Make
TOY	Toyota
CHE	Chevy
HON	Honda

Model

ID	Model	Make
1	Prius	TOY
2	Equinox	CHE
3	CRV	HON
4	Tahoe	CHE

Activity: Pet Normalizer

In this activity, you will practice data normalization skills by using the provided data.

Suggested Time:

15 minutes

Activity: Pet Normalizer

Instructions	In pgAdmin, create a new database called `pets_db`.
	Use Excel to get the data into 1NF.
	Using the normalized CSV, create the following tables with continued normalized practices: ● a table for owners that takes an ID and the owner's name. ● a table for pet names that takes two IDs, the pet's name, and the pet's type.
	Using the CSV file as guide, insert the data into respective tables.
Hint	Be sure that each table has a unique primary key.
Bonus	Create a service table that displays the different types of services that are offered.
	Create a `pet_names_updated` table that takes an ID that will connect to the services table.
	Join all three tables.

Time’s Up! Let’s Review.

Foreign Keys

Foreign Keys

Foreign Keys reference the primary key of another table.

Can have a different name. It does not have to be unique.

	Primary Key	
	A	B
1	family_id	family
2	1	Simpson
3	2	Jones

	Primary Key	Foreign Key	
	A	B	C
1	child_id	family_id	children
2	11	1	Bart
3	22	1	Lisa
4	33	1	Maggie

Activity: Foreign Keys

In this activity, you will create and populate two new tables with foreign keys that reference existing data.

Suggested Time:

15 minutes

Activity: Foreign Keys

Instructions	Create a `customer` table with a customer first name and customer last name.
	Create a `customer_email` table with a foreign key that references a field in the original customer table.
	Populate the `customer_email` table with emails.
	Create a `customer_phone` table with a foreign key that references a field in the original customer table.
	Populate the `customer_phone` table with phone numbers.
	Test foreign keys by writing a query to insert data in the `customer_phone` table that does not have a reference ID in the `customer` table.
	Join all three tables.
Hint	Think about how you can select certain columns in a table. Use those columns as a reference to insert data into a table.
	Make sure all tables have primary keys that increment with each new row of data.

Time's Up! Let's Review.

Data Relationships

Data Relationships

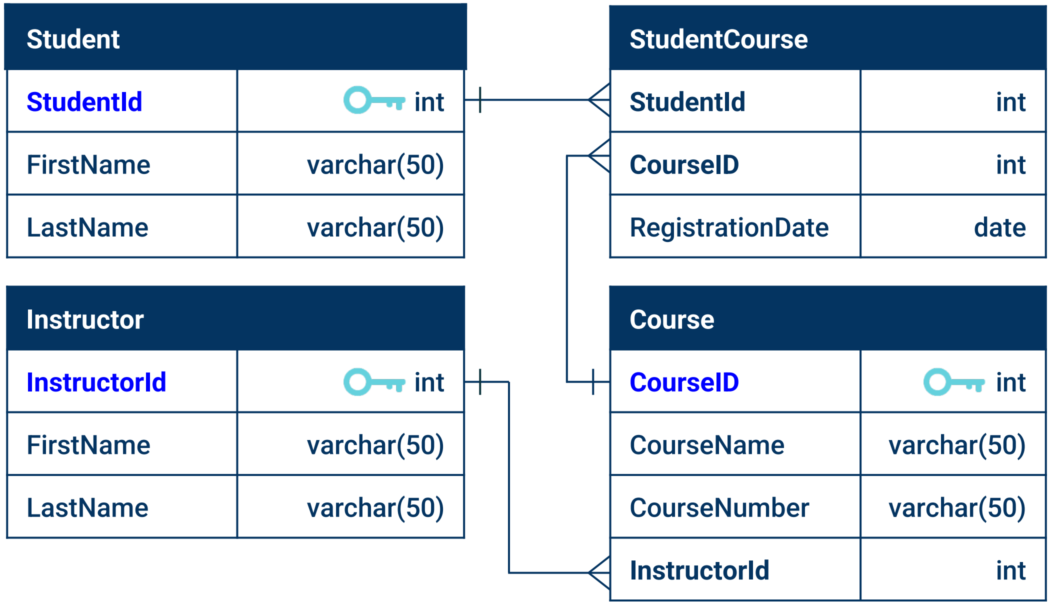
Relationships Link Tables/Entities.

Types of relationships:

01 One-to-One

02 One-to-Many

03 Many-to-Many



One-to-One Relationship. "-"

Each item in one column is linked to only one other item from the other column.

ID	Name	Social Security
1	Homer	111111111
2	Marge	222222222
3	Lisa	333333333
4	Bart	444444444
5	Maggie	555555555

Here, each person in the Simpsons family can have only one social security number.

Each social security number can be assigned only to one person.

One-to-Many Relationship “-<“

This example has two tables. The first table lists only addresses.

The second table lists each person's Social Security number and address. As before, one Social Security number is unique to one individual.

ID	Address	ID	Name	Social Security	AddressID
11	742 Evergreen Terrace	1	Homer	111111111	11
12	221B Baker Street	2	Marge	222222222	11
		3	Lisa	333333333	11
		4	Bart	444444444	11
		5	Maggie	555555555	11
		6	Sherlock	112233445	12
		7	Watson	223344556	12

One-to-Many Relationship “-<“

- Each address can be associated with multiple people.
- Each person has an address.
- The two tables, joined, would look like this.

ID	Address	ID	Name	Social Security	AddressID
11	742 Evergreen Terrace	1	Homer	111111111	11
12	221B Baker Street	2	Marge	222222222	11
		3	Lisa	333333333	11
		4	Bart	444444444	11
		5	Maggie	555555555	11
		6	Sherlock	112233445	12
		7	Watson	223344556	12

Many-to-Many Relationship “>-<“

- Each child can have more than one parent.
- Each parent can have more than one child.

Child_ID	Child
1	Bart
2	Lisa
3	Maggie

Parent_ID	Parent
11	Homer
12	Marge

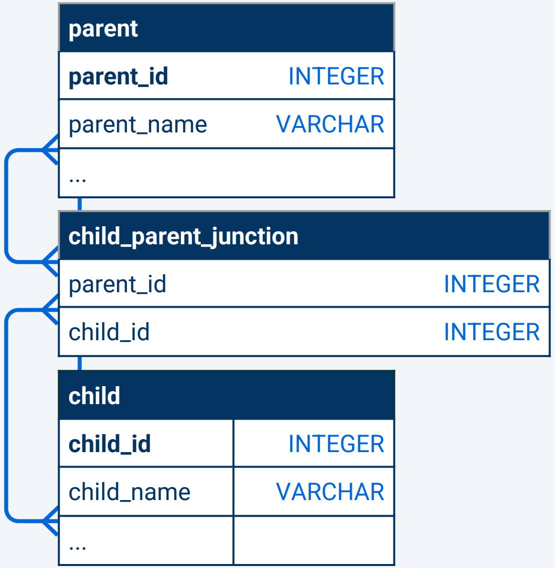
Many-to-Many Relationship “>-<“

- Each child can have more than one parent.
- Each parent can have more than one child.
- The two tables are joined in a **junction table**.

Child_ID	Child	Parent_ID	Parent
1	Bart	11	Homer
1	Bart	12	Marge
2	Lisa	11	Homer
2	Lisa	12	Marge
3	Maggie	11	Homer
3	Maggie	12	Marge

Junction Table

The junction table contains many parent_id’s and many child_id's.



	parent_id integer	child_id integer
1	11	1
2	11	2
3	11	3
4	12	1
5	12	2
6	12	3

Join child and parent table to junction table ↓

	parent_name character varying (255)	child_name character varying (255)
1	Homer	Bart
2	Homer	Lisa
3	Homer	Maggie
4	Marge	Bart
5	Marge	Lisa
6	Marge	Maggie

Activity: Data Relationships

In this activity, you will create table schemata for students and available courses, and then create a junction table to display all courses taken by students.

Suggested Time:

15 minutes

Activity: Data Relationships

Instructions You are the database consultant at a new university. Your job is to design a database model for the registrar. The database will keep track of information on students, courses offered by the university, and the courses each student has taken.	Create a `students` table that keeps track of the following: ● Unique ID number of each student ● Last and first names of each student Create a `courses` table that keeps track of the following: ● Unique ID number of each course ● Name of each course Create a `student_courses_junction` that keeps track of the following: ● All courses that have been taken by each student ● Term in which a course was taken by a student (spring or fall) Which data model is appropriate here: one to one, one to many, or many to many?
Bonus	Make sure all tables have primary keys that increment with each new row of data.

Time's Up! Let's Review.

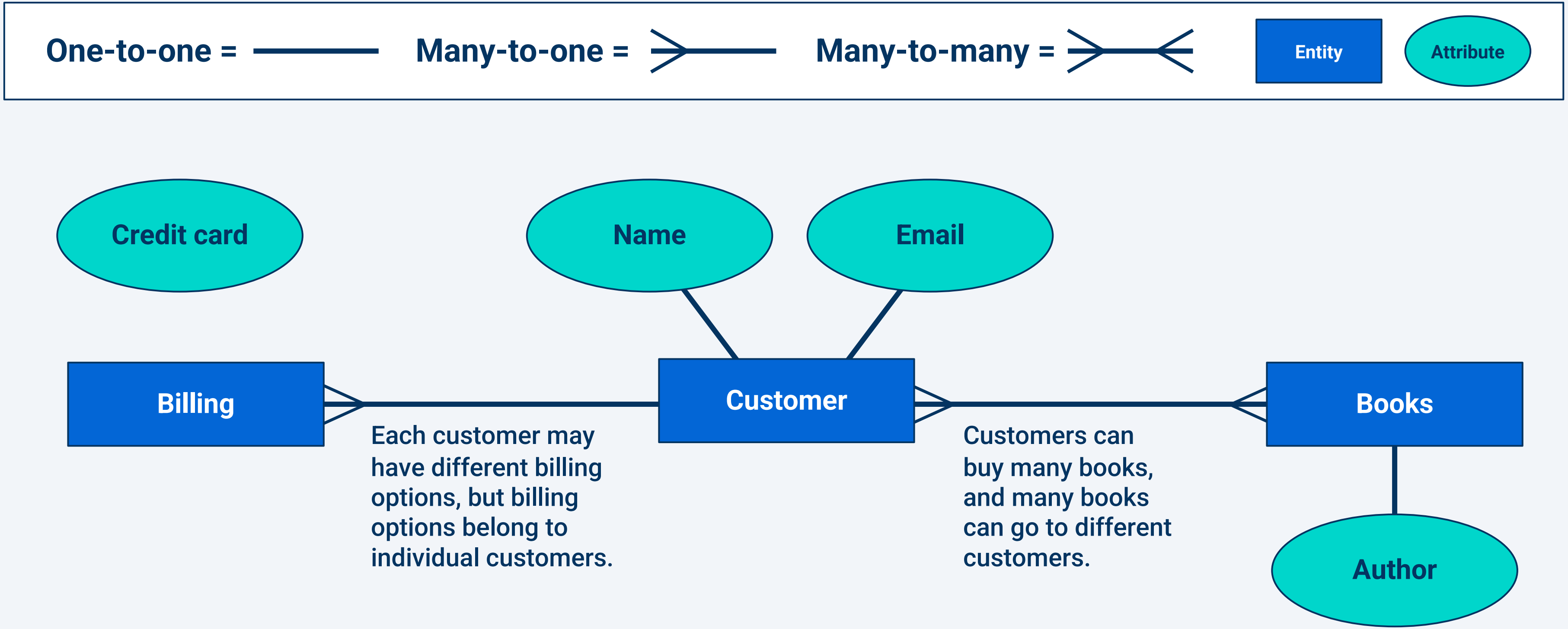
Back at 8:10 pm

Entity Relationship Diagrams

An **entity relationship diagram**, or ERD, is a visual representation of entity relationships within a database.

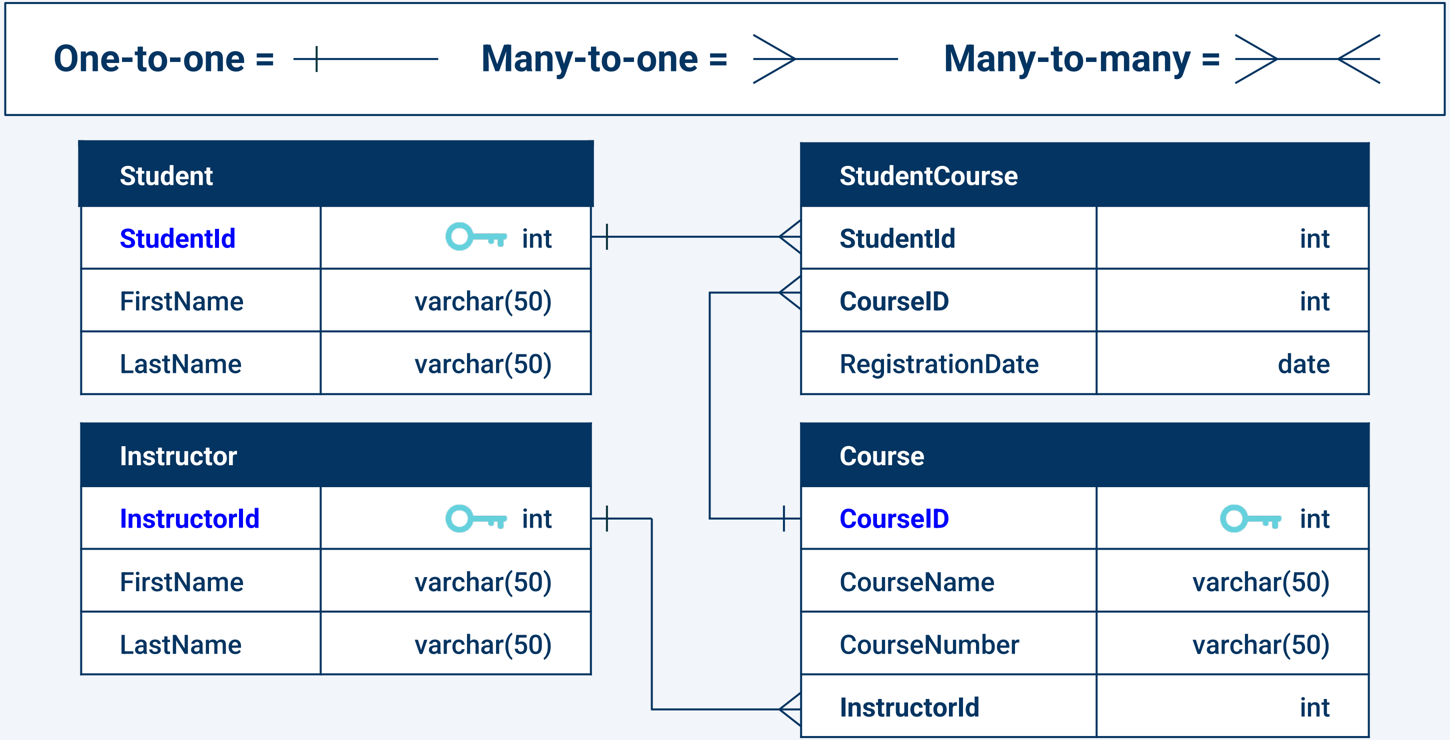
Entity Relationship Diagram

ERD uses the following notations to create the relationships.



Entity Relationship Diagram

A typical ERD design.



Entity Relationship Diagram

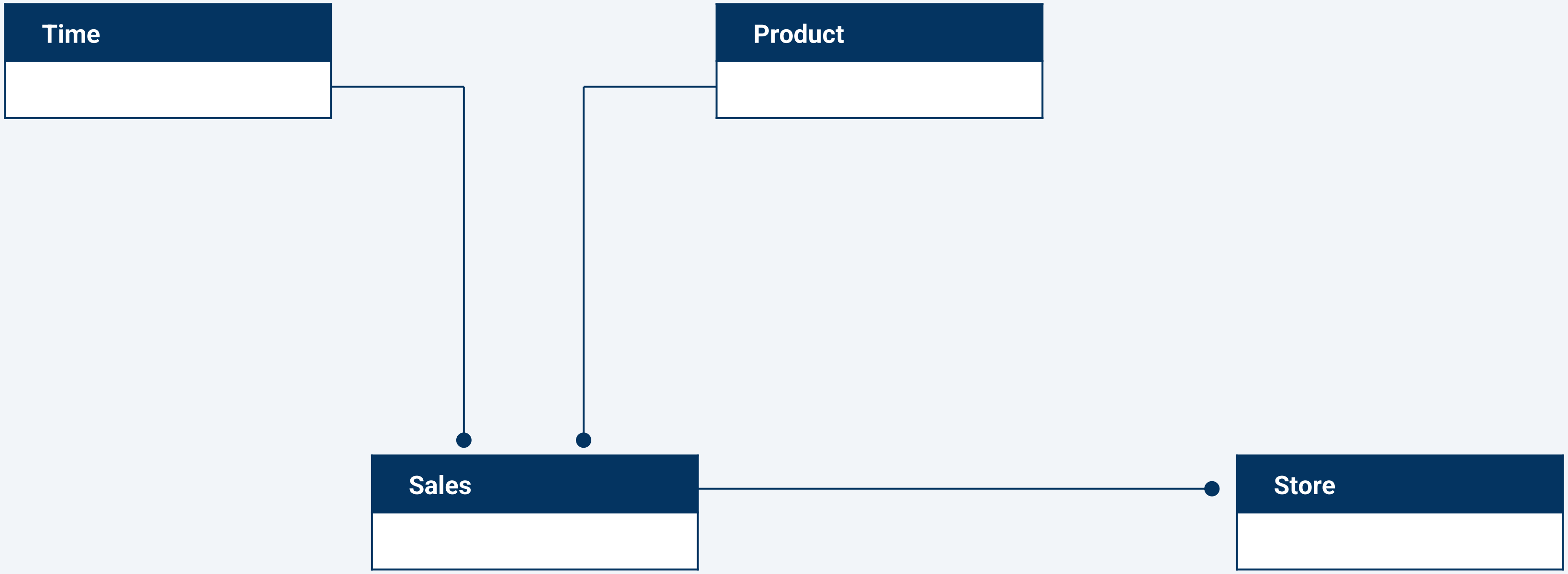
Three Types of ERDs or Data Models

01 Conceptual Model Design

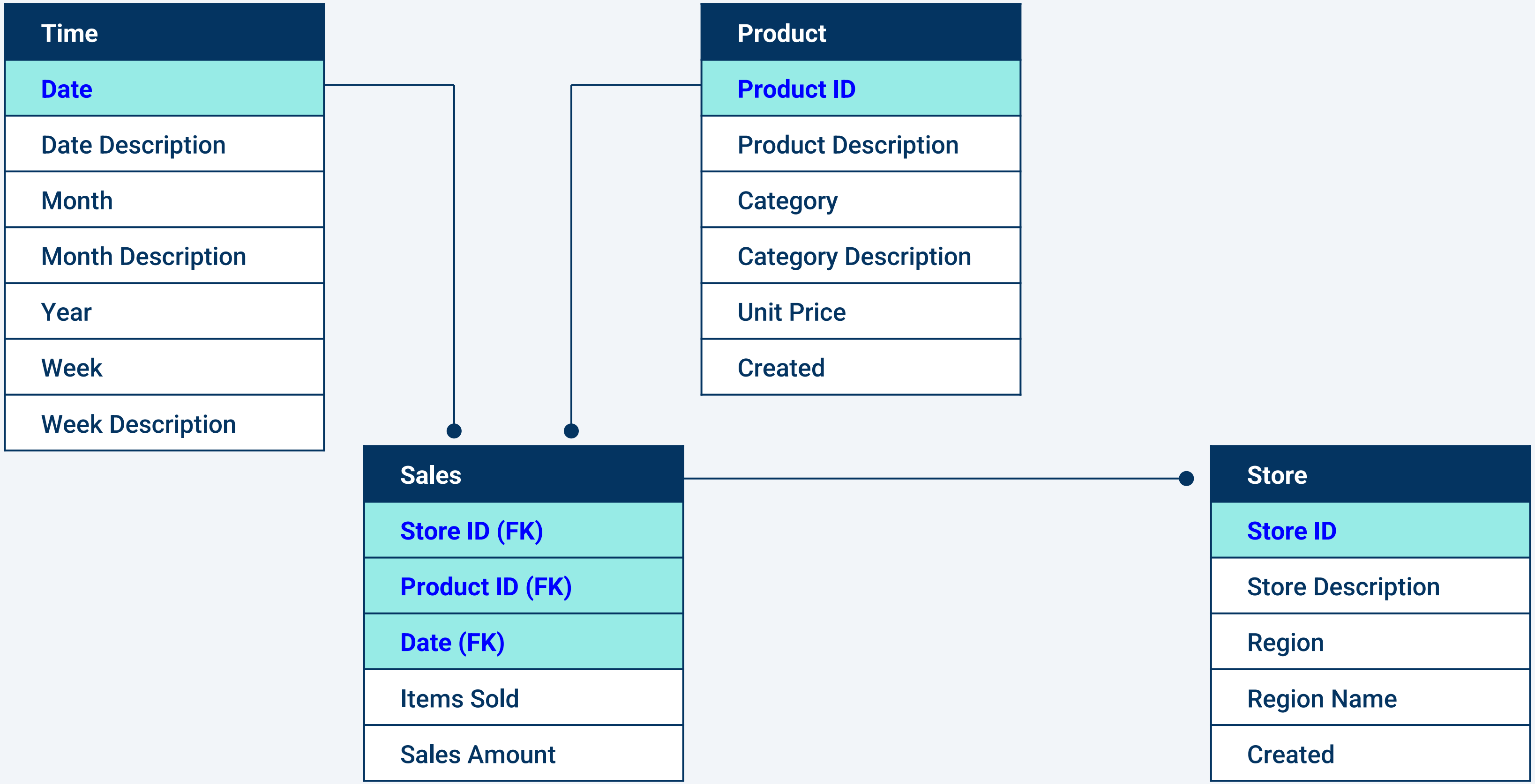
02 Logical Model Design

03 Physical Model Design

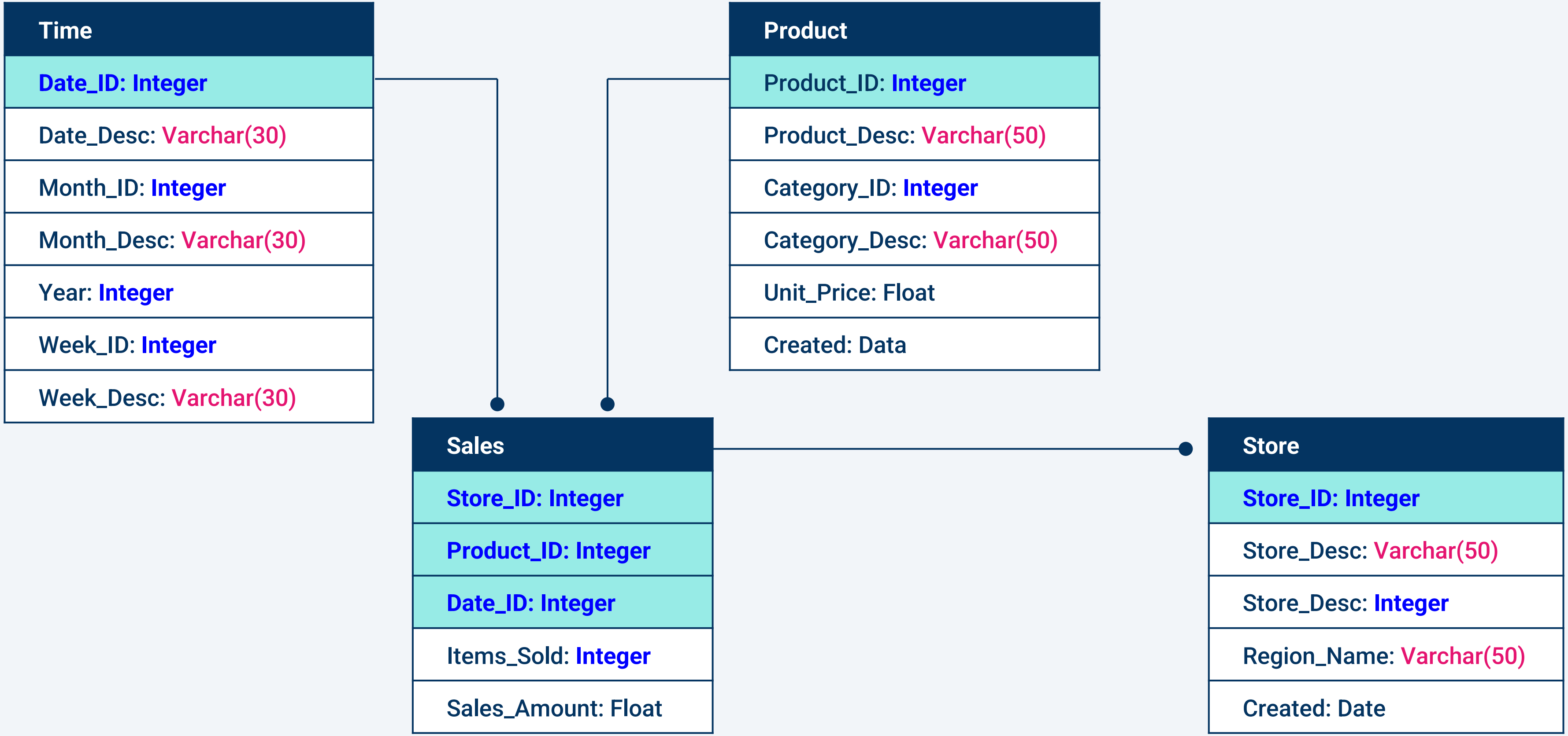
Conceptual Model Design



Logical Model Design



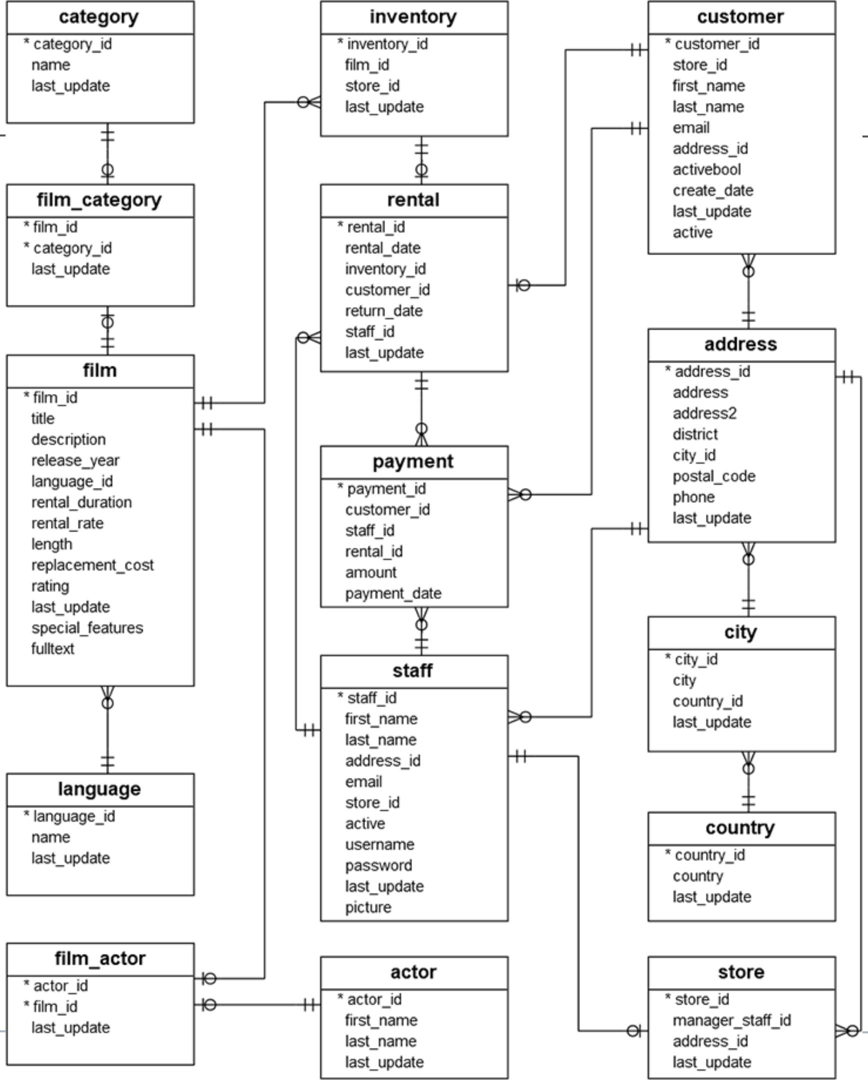
Physical Model Design



Entity Relationship Diagram

An ERD illustrates:

- entities
- data types
- relationships.



Partner Activity: Designing an ERD, Part 1

In this activity, you will create a conceptual ERD for a gym owner.

Suggested Time:

10 minutes

Partner Activity: Designing an ERD, Part 1

Instructions You are meeting with a gym owner who wants to organize his data in a database. Create a conceptual ERD for the gym owner.	Determine the entities that will be present in the database, along with their attributes. Be sure to include the following: trainers, members, and gym as well as one more entity that you think is necessary.
	Create a diagram using the **Quick Database Diagrams** tool.
	When you are satisfied with the conceptual diagram, update it to a logical ERD by including column data types and primary keys.
	Which data model is appropriate here: one to one, one to many, or many to many?
Hint	Check Slack for the **documentation** for more in-depth explanations of entity relationship diagrams.

Time’s Up! Let’s Review.

Partner Activity: Designing an ERD, Part 2

In this activity, you and and your partner will continue designing an entity relationship diagram for the gym by transitioning your logical ERD created in the previous activity to a physical ERD.

Suggested Time:

10 minutes

Partner Activity: Designing an ERD, Part 2

Hints

Foreign keys are added to each table represented by the FK acronym, followed by the relationship, e.g., OrderID INT FK >- Order.OrderID.

You will need to add foreign keys to your tables in order to map the data relationships.

Remember to document the relationships between entities using the correct symbols. Here are the allowed relationship types:

_	one TO one	0_	zero or one TO one
_<	one TO many	0_0	zero or one TO zero or one
>_	many TO one	_0<	one TO zero or many
>_<	many TO many	>0_	zero or many TO one
_0	one TO zero or one		

Partner Activity: Designing an ERD, Part 2

Hints

Keep in mind the following:

- Each member belongs to only one gym.
- Trainers work for only one gym, but a gym has many trainers.
- Each member must have a trainer, but each trainer may instruct multiple members.
- Each member has one credit card on file.

Once you have created tables in pgAdmin, you can check the table creation with the following syntax:

```sql
SELECT * FROM Members;
```

Time’s Up! Let’s Review.

Instructor Demonstration

Introduction to Unions

Activity: Unions

In this activity, you will practice unions by combining data from multiple tables without the use of joins.

Suggested Time:

15 minutes

Activity: Unions

Instructions	Using `UNION`, write a PostgreSQL statement to query the number of rows in tables `city` and `country`.
	Use `UNION` to display from the tables `customer` and `customer_list` the ID of all customers who live in the city of London. Determine whether both tables contain the same customers by using `UNION ALL`.
Hint	For the second problem, consider using subqueries.

Time’s Up! Let’s Review.