# Data Modeling

Data Boot Camp

Lesson 9.4

# Class Objectives

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

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

## Data Normalization

**Data normalization** is the process of restructuring data to a set of defined "normal forms."

The process of data normalization **eliminates data redundancy and inconsistencies** .

# Data Normalization

- Process of restructure data to a set of “normal forms.”
- Reduce and eliminate data redundancy and inconsistencies
- Three most common forms:
    - First normal form (1NF)
    - Second normal form (2NF)
    - Third normal form (3NF)
- There are even more levels!

# First Normal Form (1NF)

- Each field in a table row should contain a single value
- Each row is unique
  - Rows can have a fields that repeat
  - But whole rows do not fully match

Raw Data

| family | children |
|---|---|
| Smiths | Chris, Abby, Susy |
| Jones | Steve, Mary, Dillion |

Normalization

First Normal Form

| family | children |
|---|---|
| Smiths | Abby |
| Smiths | Chris |
| Smiths | Susy |
| Jones | Dillon |
| Jones | Mary |
| Jones | Steve |

# Second Normal Form (2NF)

- Be in First Normal Form
- Single Column Primary Key
  - Primary Key
  - Identifies the table and row uniquely
- Generally there could be a need to create a new table

Data in 1NF

| family | children |
|---|---|
| Smiths | Abby |
| Smiths | Susy |
| Jones | Mary |
| Smiths | Chris |
| Jones | Dillion |
| Jones | Mary |

2NF Normalization

Family Table

| family_id | family |
|---|---|
| 1 | Smiths |
| 2 | Jones |

Child Table

| family_id | children |
|---|---|
| 1 | Chris |
| 1 | Abby |
| 1 | Susy |
| 2 | Steve |
| 2 | Mary |
| 2 | Dillion |

# Each Normal Form Builds on the Previous

## 01 First Normal Form

- Each field in a table row should contain a single value
- Each row is unique

## 02 Second Normal Form

- Each field in a table row should contain a single value
- Each row is unique
- Single Column Primary Key

## 03 Third Normal Form

- Each field in a table row should contain a single value
- Each row is unique
- Single Column Primary Key
- No transitive dependent columns

# Converting Our Example to Second Normal Form

- Our example is already in 1NF
- To convert to 2NF, we need to add a primary key
- We will add `store_id` column as our primary key
- Since `store_name` is functionally dependent to `store_id`, `store_address` is transitively dependent on `store_id`!

| owner_name | store_name | store_address |
|---|---|---|
| Marshall | Soups and Stuff | 123, Fake St. |
| Susan | Sink Emporium | 44, New Drive |
| Susan | Tasty Burgers | 99, Old Lane |

2NF Normalization

| store_id | owner_name | store_name | store_address |
|---|---|---|---|
| 1 | Marshall | Soups and Stuff | 123, Fake St. |
| 2 | Susan | Sink Emporium | 44, New Drive |
| 3 | Susan | Tasty Burgers | 99, Old Lane |

## Second Normal Form (2NF) to Third Normal Form (3NF)

Split any transitive dependent columns into new tables

We need to split `store_address` from `store_id`

| store_id | owner_name | store_name | store_address |
|---|---|---|---|
| 1 | Marshall | Soups and Stuff | 123, Fake St. |
| 2 | Susan | Sink Emporium | 44, New Drive |
| 3 | Susan | Tasty Burgers | 99, Old Lane |

3NF Normalization

| store_id | owner_name | store_name |
|---|---|---|
| 1 | Marshall | Soups and Stuff |
| 2 | Susan | Sink Emporium |
| 3 | Susan | Tasty Burgers |

| store_name | store_address |
|---|---|
| Soups and Stuff | 123, Fake St. |
| Sink Emporium | 44, New Drive |
| Tasty Burgers | 99, Old Lane |

## Foreign Keys

## Foreign Keys

Foreign Keys reference the primary key of another table.

- Can have a different name
- Do not need to be unique

Primary Key

| family_id | family |
|---|---|
| 1 | Smiths |
| 2 | Jones |

Primary Key

Foreign Key

| child_id | family_id | children |
|---|---|---|
| 11 | 1 | Chris |
| 22 | 1 | Abby |
| 33 | 1 | Susy |
| 44 | 2 | Steve |
| 55 | 2 | Mary |
| 66 | 2 | Dillion |

## Data Relationships

## Data Relationships

01

One-to-One

02

One-to-Many

03

Many-to-Many

## One-to-One Relationship

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

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

| 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 |

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

# Many-to-Many Relationship

| ID | Child | | ID | Parent |
|---|---|---|---|---|
| 1 | Bart | | 11 | Homer |
| 2 | Lisa | | 12 | Marge |
| 3 | Maggie | | | |

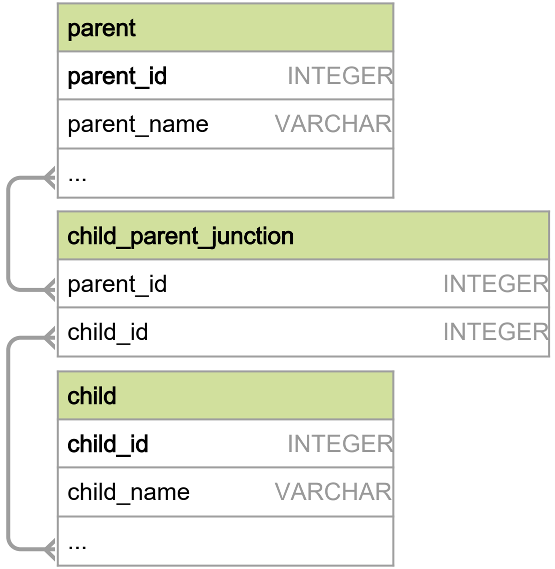
- Each child can have more than one parent.
- Each parent can have more than one child.

# Many-to-Many Relationship

| ChildID | Child | ParentID | 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 |

- 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.

# Junction Table



The Junction table contains many parent_ids and many child_ids

| | 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

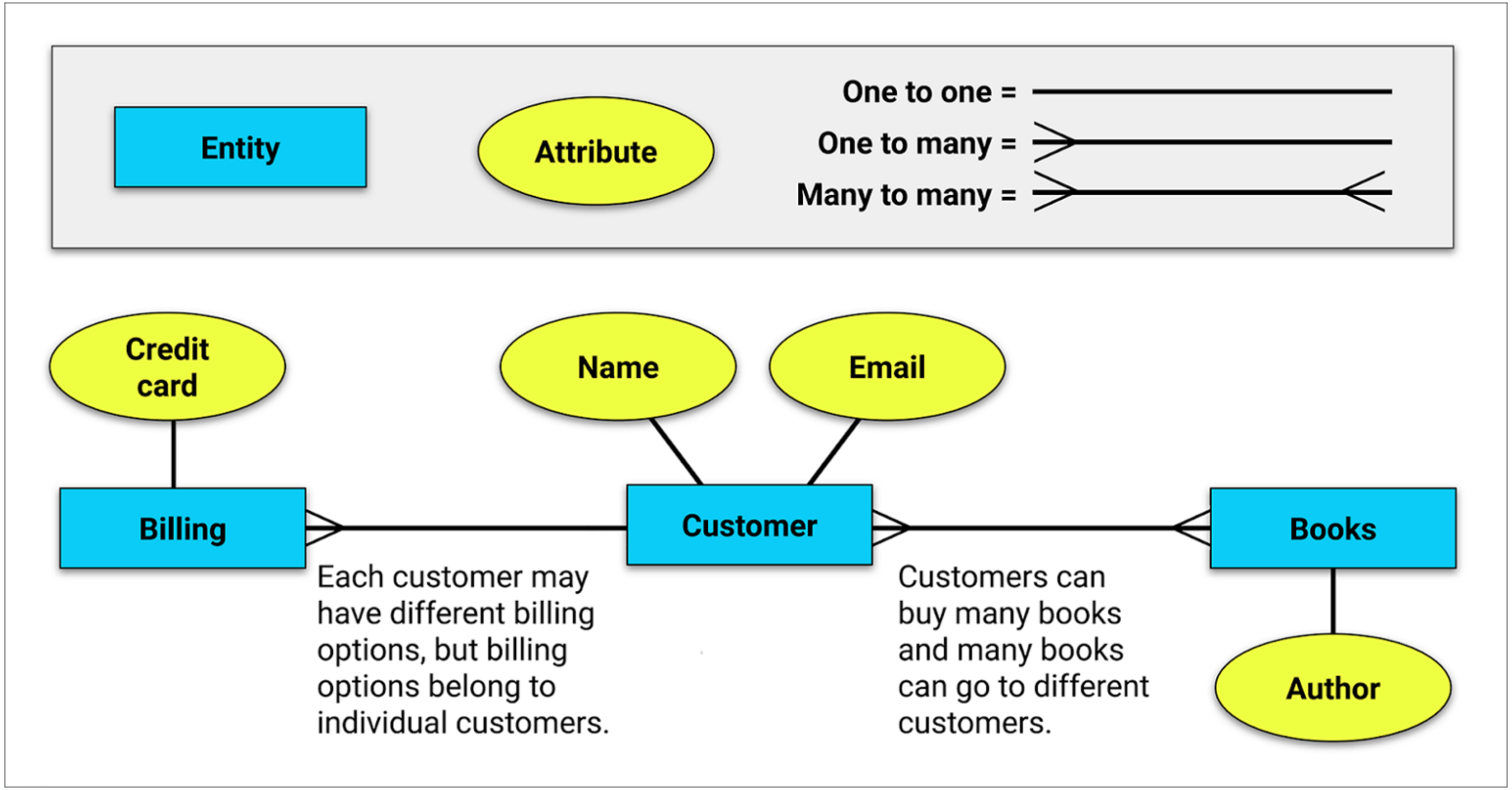
| | parent_name character varying (255) | child_name character varying (255) |
|---|---|---|
| 1 | Homer | Bart |
| 2 | Homer | Lisa |
| 3 | Homer | Maggie |
| 4 | Marge | Bart |
| 5 | Marge | Lisa |
| 6 | Marge | Maggie |

## Entity Relationship Diagrams

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

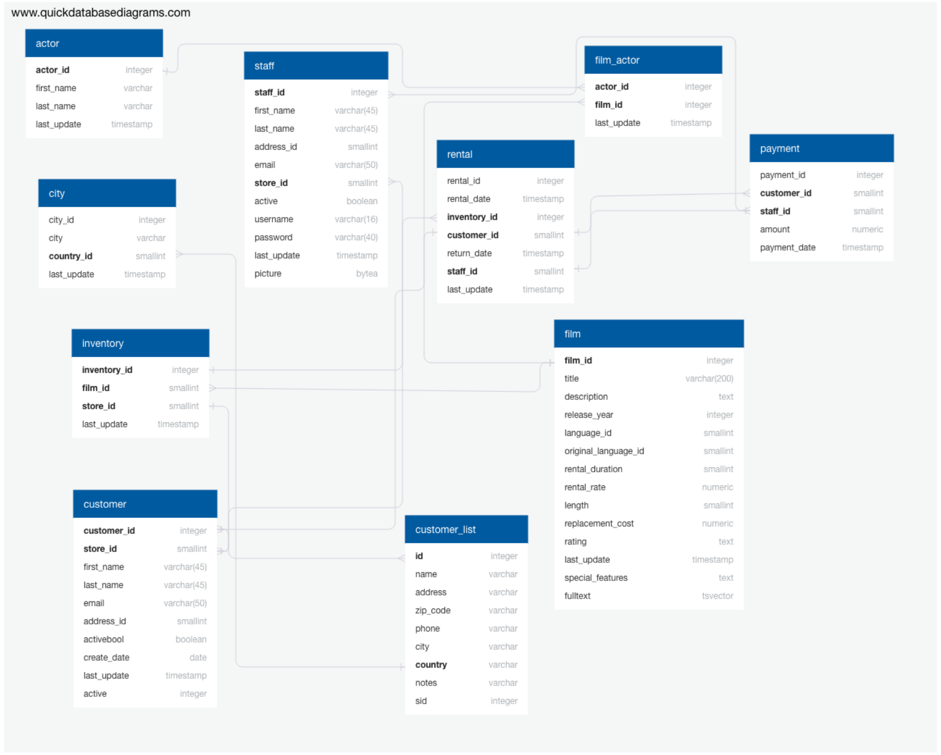
# ERDs

ERD use the following notation to create the models.


Entity
Attribute
One to one =
One to many =
Many to many =
Credit card
Name
Email
Billing
Customer
Books
Author
Each customer may have different billing options, but billing options belong to individual customers.
Customers can buy many books and many books can go to different customers.

## ERDs

Entities, their data types, and relationships are all illustrated in the diagram.



## ERDs

There are three models used when creating diagrams:

- **Conceptual**: basic information containing table and column names.
- **Logical**: slightly more complex than conceptual models with IDs and data types defined.
- **Physical**: the blueprint of the database, reflecting physical relationships between entities.