Reference card

Database relationships

Database relationships

Relational Database Management Systems (RDBMSs) and **Entity-Relationship Data Models** are closely related and work in conjunction to facilitate efficient and structured data management.

Relational Database Management Systems

RDBMSs are **software systems** for managing relational databases, utilising relational models for **structured data organisation and manipulation**.

Entity-Relationship Data Models

Entity-Relationship Data Models **represent database structure** by capturing entities, attributes, and relationships, providing a **visual representation of data organisation**.

Types of relationships

Relationships are the **established associations between two or more tables**. These relationships differ based on how certain columns in these tables relate to one another.

There are different types of relationships that can exist between tables in a database:

One-to-one (1:1)

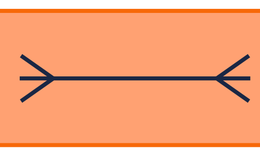
An entity in one table relates to a **maximum of one other entity** in another table.

One-to-many (1:M)

One entity in a table relates to **multiple entities** in another table and vice versa.

Many-to-many (M:M)

Entities in one table, either **individually or in groups**, relate to **individuals or groups of entities in another table**.

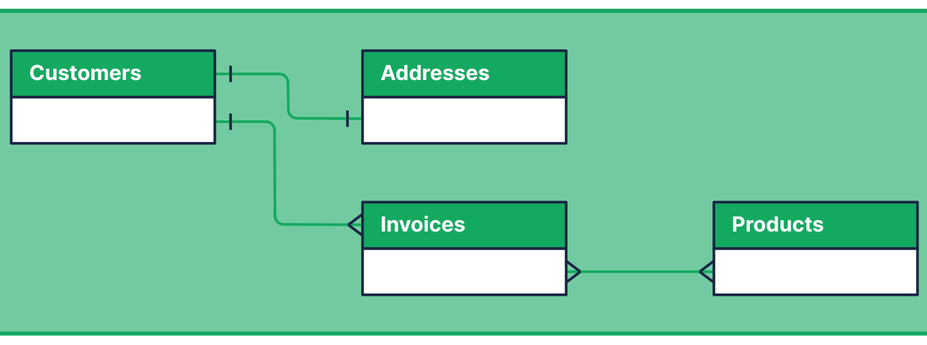


Entity-Relationship Diagrams

An Entity-Relationship Diagram, or ERD for short, is a **graphical representation of the relationships that exist** among the entities in a database typically drawn at up to three levels of abstraction.

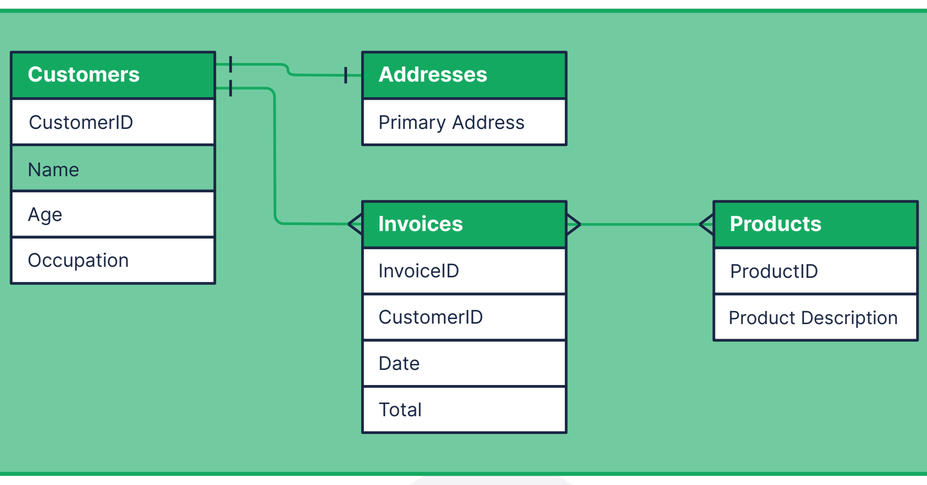
Conceptual data model

Includes **entities** and **cardinalities** only.



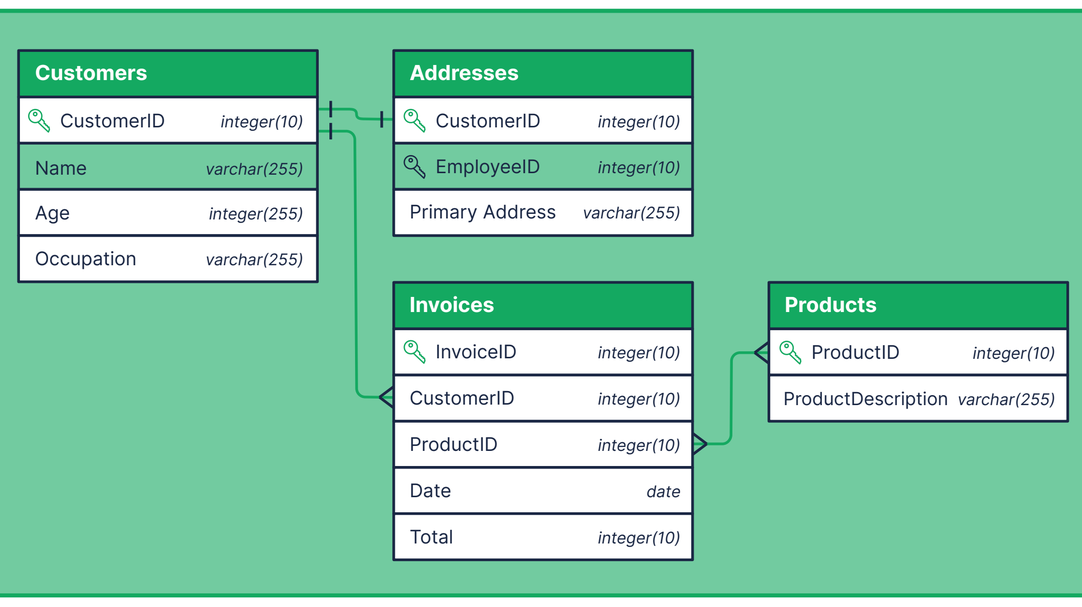
Logical data model

Includes **entities**, **cardinalities**, and **attributes**.



Physical data model

Includes **entities**, **cardinalities**, **attributes**, **keys**, and **datatypes**.



Database keys

Database keys **determine how tables are connected** or on which specific columns they are connected.

Primary keys

A primary key is a column that **uniquely identifies each row** in a table.

Create a primary key constraint:

```
CREATE TABLE table_name (
        Column1 datatype(size) PRIMARY KEY,
        Column2 datatype(size),
        Column3 datatype(size)
);
```

Primary keys need to adhere to the following rules / specifications:

- No duplicates.
- No null values.
- Can not be deleted unless first deleting the referencing table or removing the constraint.

Composite keys

A composite key is a **key that consists of two or more columns** that together uniquely identify an entity occurrence and make up the primary key.

Create a primary key constraint based on composite keys:

```
CREATE TABLE table_name (
        Column1 datatype(size),
        Column2 datatype(size),
        Column3 datatype(size)
        PRIMARY KEY (Column1,Column2)
);
```

Foreign keys

A foreign key is the **column or set of columns which corresponds to the primary key** in another table.

Create a primary key constraint:

```
CREATE TABLE table_name (
        Column1 datatype(size),
        Column2 datatype(size),
        Column3 datatype(size)
        FOREIGN KEY (Column1) REFERENCES
            table_name (column_name)
);
```

Foreign keys have the following characteristics:

- Null values are allowed in a foreign key.
- Foreign key values can be duplicated.
- There can be more than a single foreign key in a table.

EXPLORE AI ACADEMY