**What is an ER Diagram?**

**ER Diagram** = **Entity-Relationship Diagram**  
It is a **visual representation** of the structure of a database.

* **Entities** (things we store information about)
* **Attributes** (properties/details of entities)
* **Relationships** (how entities are connected)

🔵 **Entity** → Real-world object (Customer, Order, Product)  
🟡 **Attribute** → Details (Customer Name, Order Date, Product Price)  
🔷 **Relationship** → Action between entities (Customer *places* Order)

**How to Draw an ER Diagram from Business Information or Documents**

**Step-by-Step:**

**1. Read the Business Document Carefully**

* Find **nouns** → Entities.  
  (e.g., *Customer*, *Order*, *Product*)
* Find **adjectives** → Attributes.  
  (e.g., *Customer Name*, *Order Date*, *Product Price*)
* Find **verbs** → Relationships.  
  (e.g., *places*, *buys*, *sells*)

**2. Identify Entities**

* Anything important to store in database.
* Example from Business Info:
  + Entities: Customer, Order, Product

**3. Identify Attributes**

* Each entity has attributes.
* Example:
  + Customer → CustomerID, Name, Email
  + Product → ProductID, Name, Price

**4. Identify Relationships**

* How entities are related.
* Example:
  + Customer **places** Order
  + Order **includes** Product

**5. Determine Cardinality**

* One-to-One (1:1)
* One-to-Many (1:M)
* Many-to-Many (M:N)

Example:

* One Customer → Many Orders (1:M)
* One Order → Many Products (M:N)

**6. Draw the Diagram**

* Draw **rectangles** for entities.
* Draw **ellipses** for attributes and connect them.
* Draw **diamonds** for relationships between entities.
* Add lines showing cardinality (1:1, 1:M, M:N).

**🖍️ Simple Example**

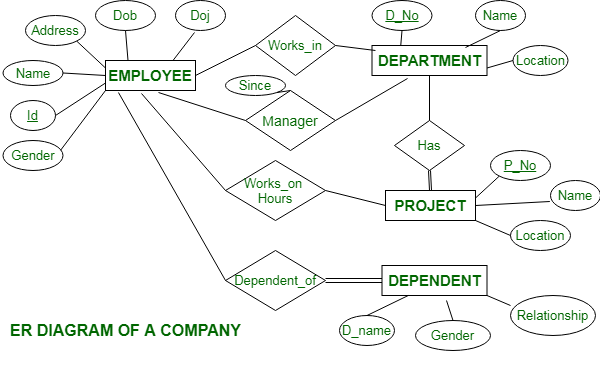
**Business Info:**

"A customer can place many orders. Each order can include multiple products."

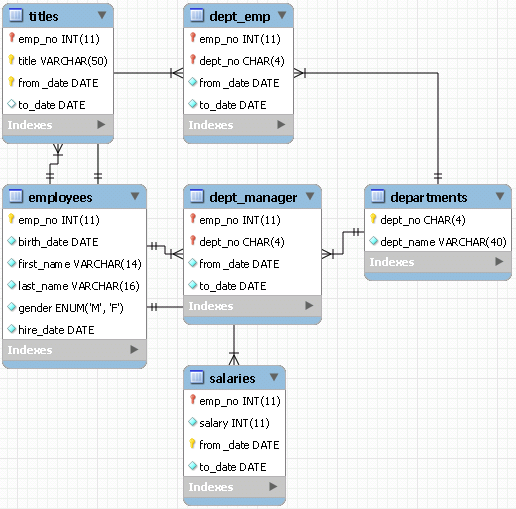
**ER Diagram Plan:**

| **Entity** | **Attributes** | |
| --- | --- | --- |
| Customer | CustomerID (PK), Name, Email | |
| Order | OrderID (PK), OrderDate | |
| Product | ProductID (PK), ProductName, Price | |
| **Relationship** | | **Details** | |
| Customer *places* Order | | 1 Customer → Many Orders | |
| Order *includes* Product | | Many Orders → Many Products (M:N) | |
|  | |  | |

**Sample ER diagram:**



**Schema diagram of a Company :**



**What is Data Modeling Technique?**

Data Modeling is a Process of designing how data should be organized, stored, connected, and accessed in a database. Invented by ralph kimball

Ralph Kimball introduced the data warehouse/business intelligence industry to dimensional modeling in 1996 with his seminal book, The Data Warehouse Toolkit. Since then, the Kimball Group has extended the portfolio of best practices.

It helps in:

* Understanding the business structure
* Creating efficient databases
* Making sure the data is accurate and easy to retrieve

**In short:**  
Data Modeling is like making a blueprint before building a database or application!

**📋 Why is Data Modeling Important?**

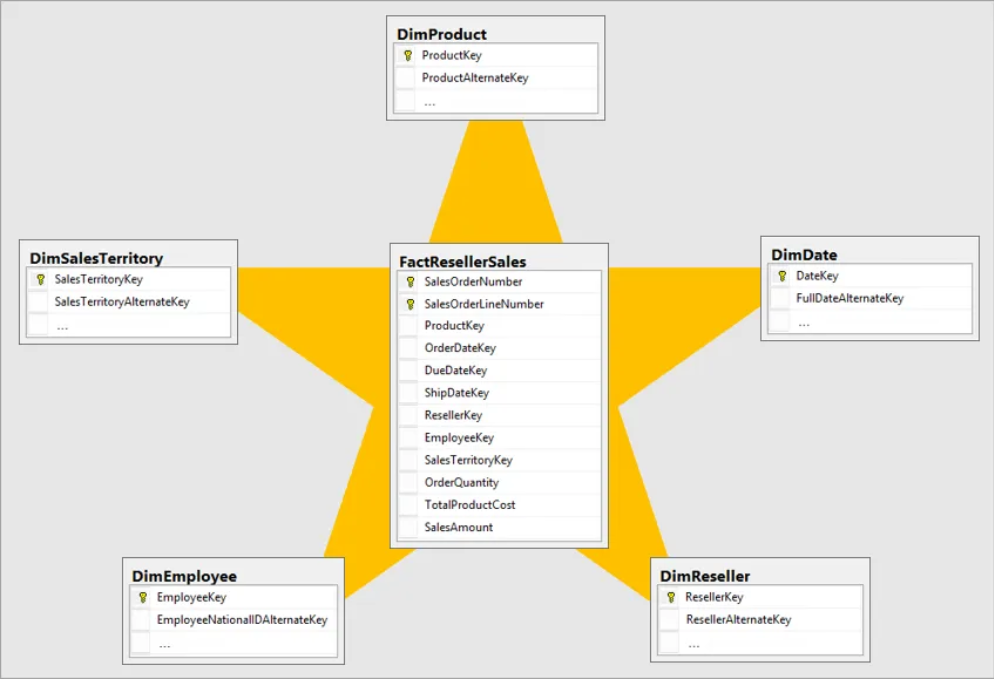
| Purpose | Benefit |
| --- | --- |
| Structure data properly | Avoids confusion later |
| Defines relationships clearly | Improves database quality |
| Reduces errors | Saves time and cost |
| Helps communication | Easier to explain database design to others |

**Different types of data modelling:**

* 1. **Star Schema**
  2. **Snowflake schema**

**✨ STAR SCHEMA**

| **Point** | **Details** |
| --- | --- |
| Shape | Looks like a star ⭐ (central fact table with surrounding dimension tables) |
| Main Idea | Fact Table in center, directly connected to Dimension Tables |
| Fact Table | Stores measurable data (e.g., Sales Amount, Quantity) |
| Dimension Tables | Store descriptive information (e.g., Customer Name, Product Name, Date, Store Location) |
| Joins | Simple joins (Fact ↔ Dimension) |

****

**Example:**

* **Fact Table: Sales  
  (SalesID, ProductID, CustomerID, StoreID, DateID, Quantity, TotalAmount)**
* **Dimension Tables:**
  + **Customer (CustomerID, Name, Gender, City)**
  + **Product (ProductID, ProductName, Category)**
  + **Store (StoreID, StoreName, Location)**
  + **Date (DateID, FullDate, Month, Year)**

**[Customer]**

**|**

**[Product]**

**|**

**[Store] ---> [Sales] <--- [Date]**

**✅ Easy to query  
✅ Fast performance  
✅ Simple design**

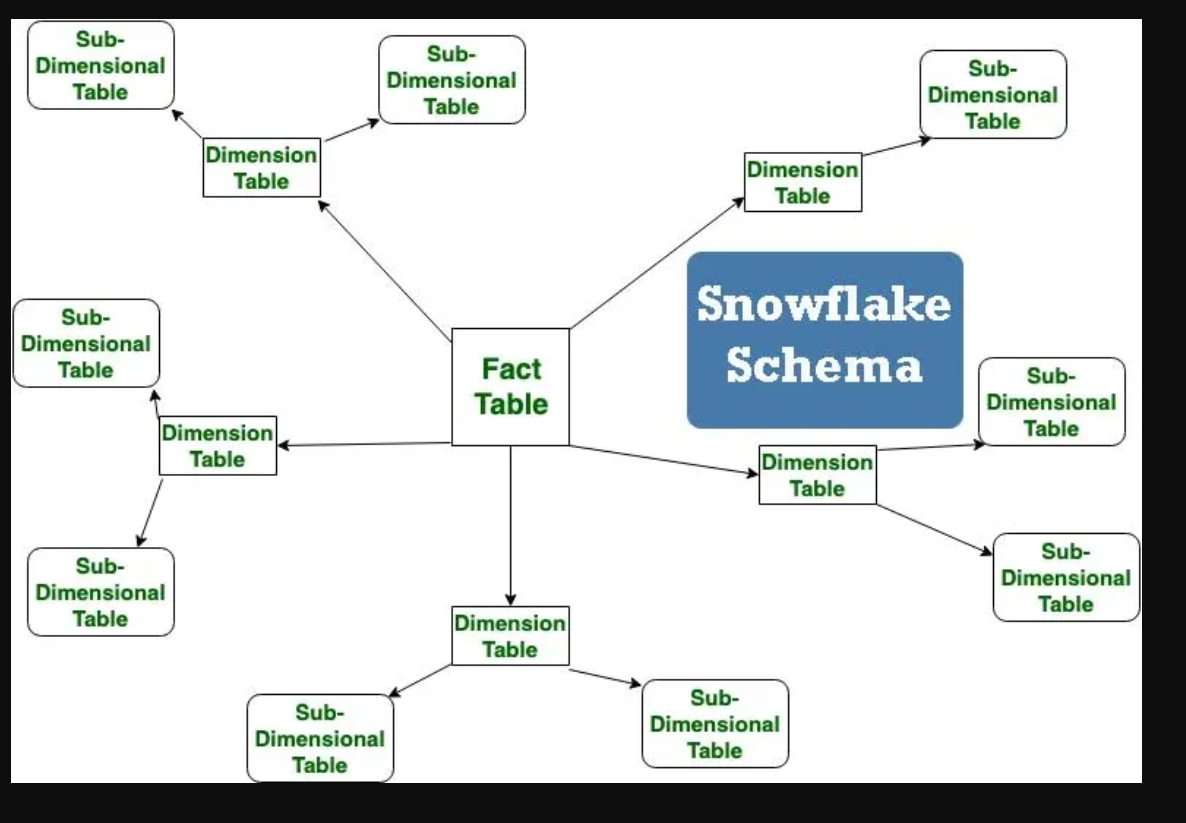
**🧊 SNOWFLAKE SCHEMA**

| **Point** | **Details** |
| --- | --- |
| Shape | Looks like a snowflake ❄️ (dimension tables are normalized — split into multiple tables) |
| Main Idea | Fact Table connected to normalized dimension tables (dimensions split into sub-tables) |
| Fact Table | Same (measurable data) |
| Dimension Tables | Are broken down into multiple related tables |
| Joins | More complex (Fact ↔ Dimension ↔ Sub-dimension) |

**📋 Example:**

Instead of one big Product table, we break it:

* Product Table (ProductID, ProductName, CategoryID)
* Category Table (CategoryID, CategoryName)



**Same for Customer, Store, etc.**

**[Customer]--->[City]**

**|**

**[Product]--->[Category]**

**|**

**[Store]--->[Region]**

**|**

**[Sales]**

**|**

**[Date]**

✅ Saves storage space  
✅ Avoids data redundancy  
❌ Slightly slower queries (because more joins)

**Key Differences**

| **Star Schema** | **Snowflake Schema** |
| --- | --- |
| Dimension tables are denormalized (all info in one table) | Dimension tables are normalized (split into sub-tables) |
| Fewer joins → Faster querying | More joins → Slower querying |
| Simple design | Complex design |
| Uses more storage | Saves storage |
| Best for simple reporting | Best when storage efficiency is critical |