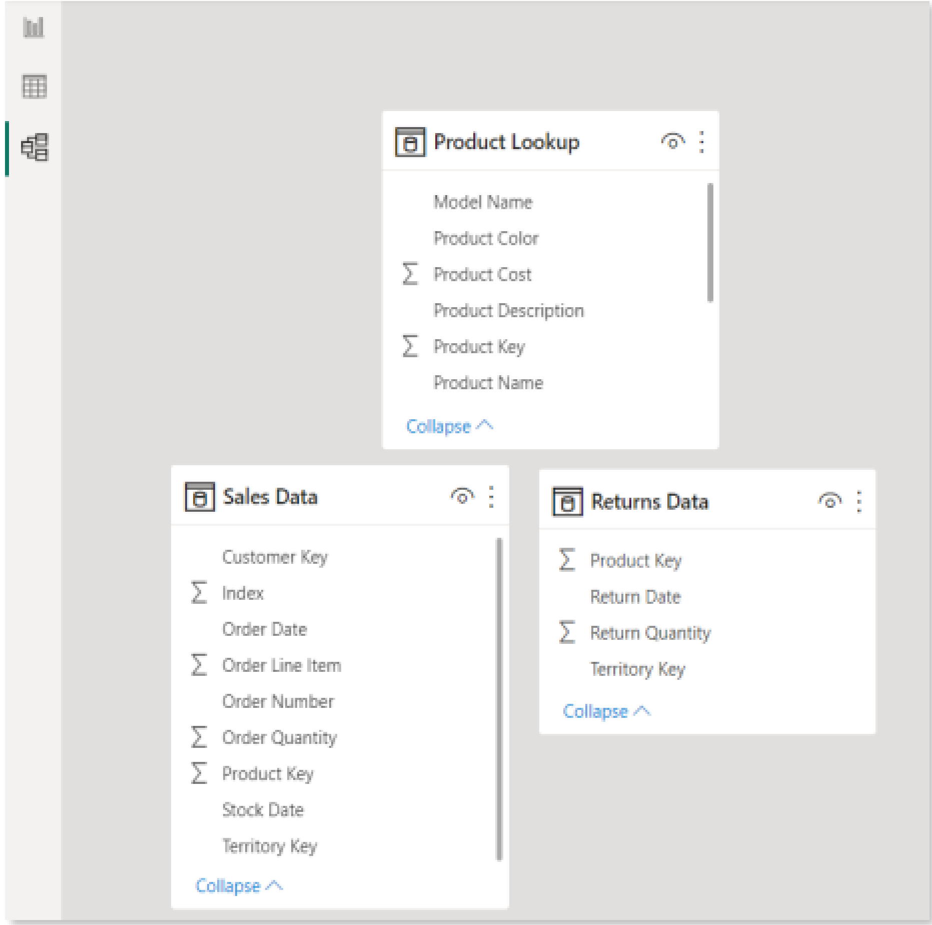
DATABASES

Lecture 2. Data model. Normalization

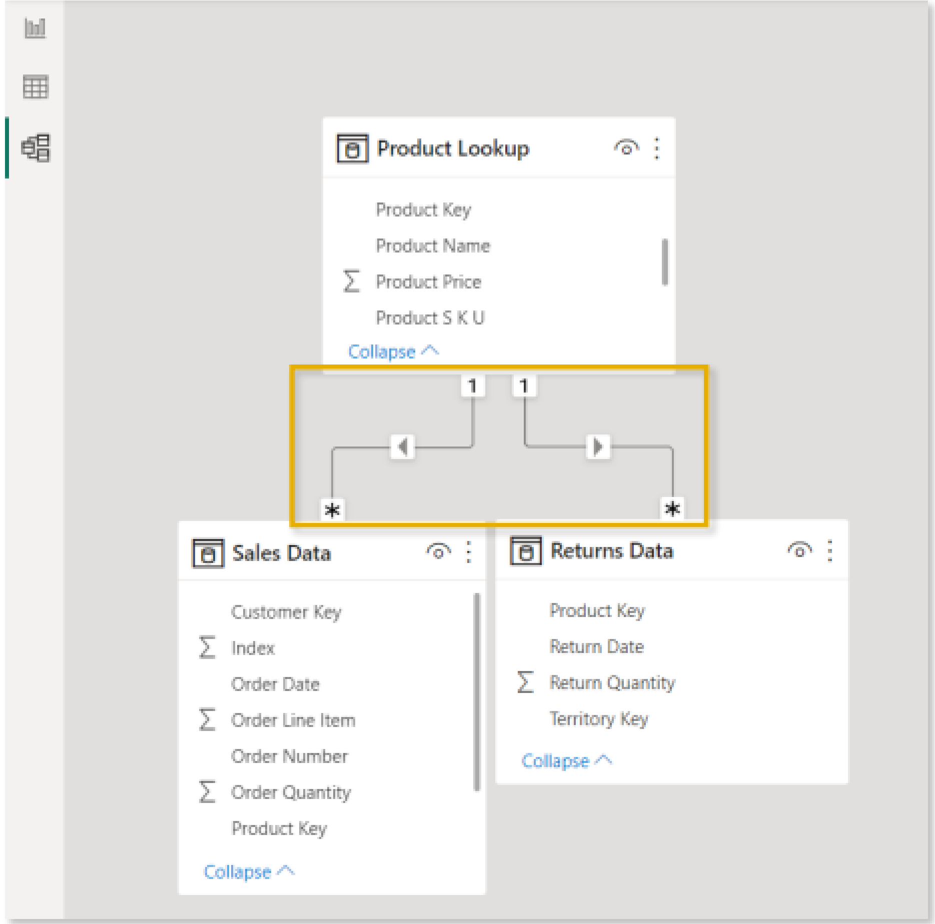
WHAT IS A DATA MODEL?



This **IS NOT** a data model

- This is a collection of independent tables, which share no connections or relationships

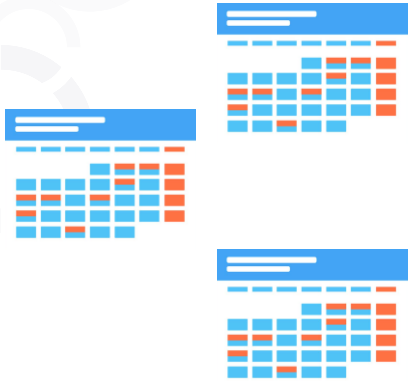
WHAT IS A DATA MODEL?

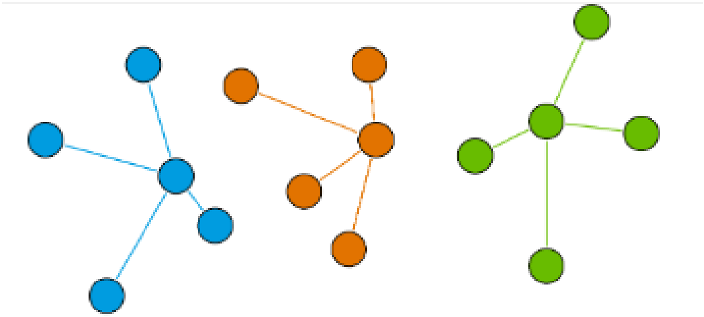


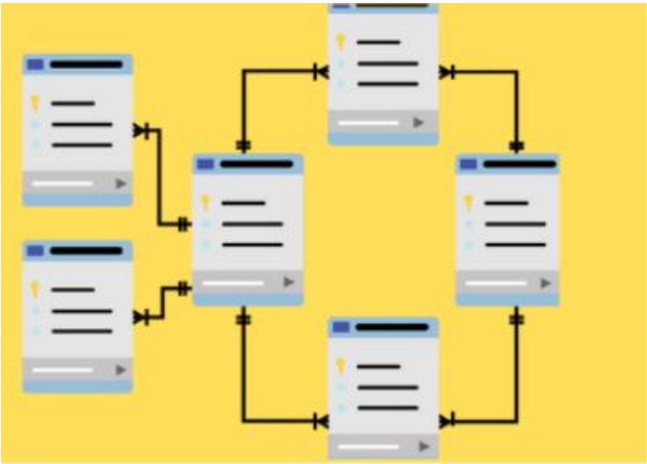
This **IS** a data model

- The tables are connected via relationships, based on a common field (Product Key)

Data model







Data stored in tables

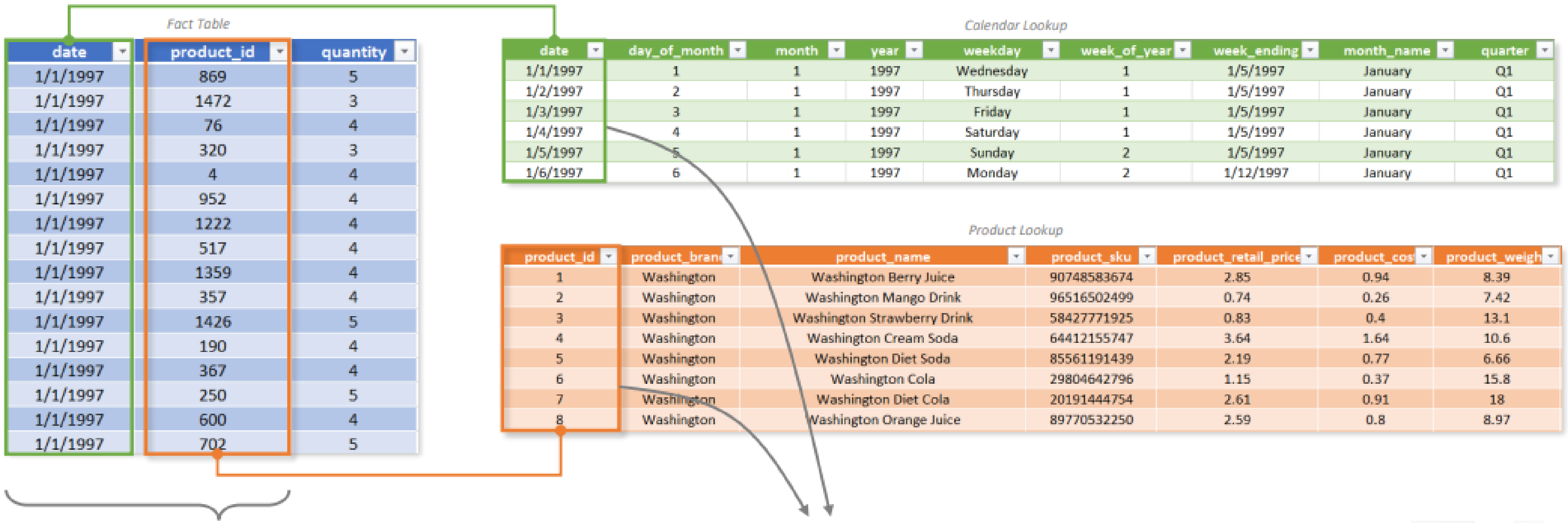
Data need to be logically connected

Data model is organized table structure

The data model is an organized structure of tables.

PRIMARY KEY AND FOREIGN KEY

Primary Key and Foreign Key – need in tables to create relations



Fact Table

date	product_id	quantity
1/1/1997	869	5
1/1/1997	1472	3
1/1/1997	76	4
1/1/1997	320	3
1/1/1997	4	4
1/1/1997	952	4
1/1/1997	1222	4
1/1/1997	517	4
1/1/1997	1359	4
1/1/1997	357	4
1/1/1997	1426	5
1/1/1997	190	4
1/1/1997	367	4
1/1/1997	250	5
1/1/1997	600	4
1/1/1997	702	5

Calendar Lookup

date	day_of_month	month	year	weekday	week_of_year	week_ending	month_name	quarter
1/1/1997	1	1	1997	Wednesday	1	1/5/1997	January	Q1
1/2/1997	2	1	1997	Thursday	1	1/5/1997	January	Q1
1/3/1997	3	1	1997	Friday	1	1/5/1997	January	Q1
1/4/1997	4	1	1997	Saturday	1	1/5/1997	January	Q1
1/5/1997	5	1	1997	Sunday	2	1/5/1997	January	Q1
1/6/1997	6	1	1997	Monday	2	1/12/1997	January	Q1

Product Lookup

product_id	product_bran	product_name	product_sku	product_retail_price	product_cos	product_weigh
1	Washington	Washington Berry Juice	90748583674	2.85	0.94	8.39
2	Washington	Washington Mango Drink	96516502499	0.74	0.26	7.42
3	Washington	Washington Strawberry Drink	58427771925	0.83	0.4	13.1
4	Washington	Washington Cream Soda	64412155747	3.64	1.64	10.6
5	Washington	Washington Diet Soda	85561191439	2.19	0.77	6.66
6	Washington	Washington Cola	29804642796	1.15	0.37	15.8
7	Washington	Washington Diet Cola	20191444754	2.61	0.91	18
8	Washington	Washington Orange Juice	89770532250	2.59	0.8	8.97

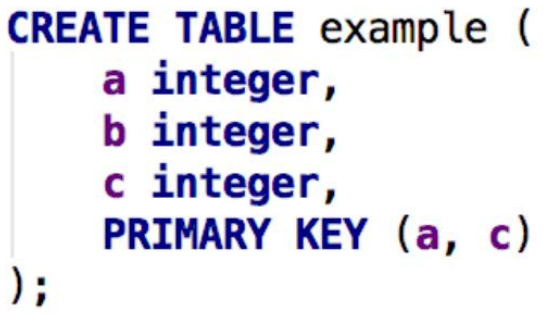
These are **foreign keys (FK)**
They contain multiple instances of each value, and relate to **primary keys** in dimension tables

These are **primary keys (PK)**
They uniquely identify each row of the table, and relate to **foreign keys** in fact tables

PRIMARY KEY AND FOREIGN KEY

Primary Key and Foreign Key – need in tables to create relations

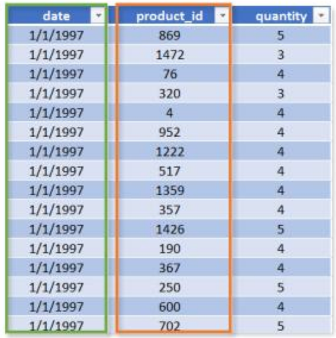
- A primary key is used to uniquely identify each row in a table.
- A primary key can consist of one or more columns on a table. When multiple columns are used as a primary key, they are called a composite key.



```
CREATE TABLE example (
    a integer,
    b integer,
    c integer,
    PRIMARY KEY (a, c)
);
```

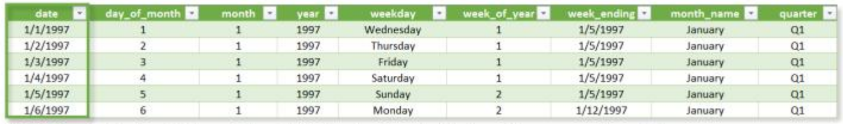
For example: IIN, email address, vehicle identification number, passport number.

FOREIGN KEY



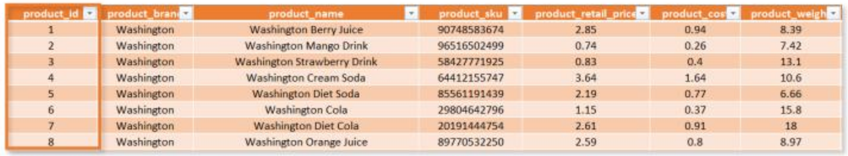
date	product_id	quantity
1/1/1997	869	5
1/1/1997	1472	3
1/1/1997	76	4
1/1/1997	320	3
1/1/1997	4	4
1/1/1997	952	4
1/1/1997	1222	4
1/1/1997	517	4
1/1/1997	1359	4
1/1/1997	357	4
1/1/1997	1426	5
1/1/1997	190	4
1/1/1997	367	4
1/1/1997	250	5
1/1/1997	600	4
1/1/1997	702	5

*This **Fact** table contains **quantity** values, along with **date** and **product_id** fields*



date	day_of_month	month	year	weekday	week_of_year	week_ending	month_name	quarter
1/1/1997	1	1	1997	Wednesday	1	1/5/1997	January	Q1
1/2/1997	2	1	1997	Thursday	1	1/5/1997	January	Q1
1/3/1997	3	1	1997	Friday	1	1/5/1997	January	Q1
1/4/1997	4	1	1997	Saturday	1	1/5/1997	January	Q1
1/5/1997	5	1	1997	Sunday	2	1/5/1997	January	Q1
1/6/1997	6	1	1997	Monday	2	1/12/1997	January	Q1

*This **Calendar Lookup** table contains attributes about each **date** (month, year, quarter, etc.)*

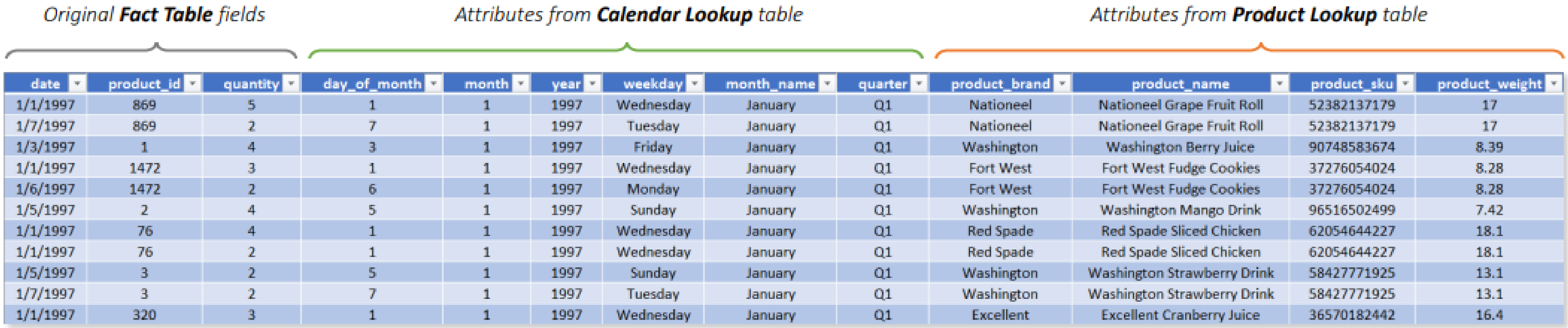


product_id	product_bran	product_name	product_sku	product_retail_price	product_cos	product_weigh
1	Washington	Washington Berry Juice	90748583674	2.85	0.94	8.39
2	Washington	Washington Mango Drink	96516502499	0.74	0.26	7.42
3	Washington	Washington Strawberry Drink	58427771925	0.83	0.4	13.1
4	Washington	Washington Cream Soda	64412155747	3.64	1.64	10.6
5	Washington	Washington Diet Soda	85561191439	2.19	0.77	6.66
6	Washington	Washington Cola	29804642796	1.15	0.37	15.8
7	Washington	Washington Diet Cola	20191444754	2.61	0.91	18
8	Washington	Washington Orange Juice	89770532250	2.59	0.8	8.97

*This **Product Lookup** table contains attributes about each **product_id** (brand, SKU, price, etc.)*

- A foreign key constraint specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table.
- We say this maintains the referential integrity between two related tables.
- May NOT be UNIQUE, may be repeated in tables
- A table can have more than one foreign key constraint.
- This is used to implement many-to-many relationships between tables.

RELATIONSHIPS VS. MERGED TABLES



*Original **Fact Table** fields*

*Attributes from **Calendar Lookup** table*

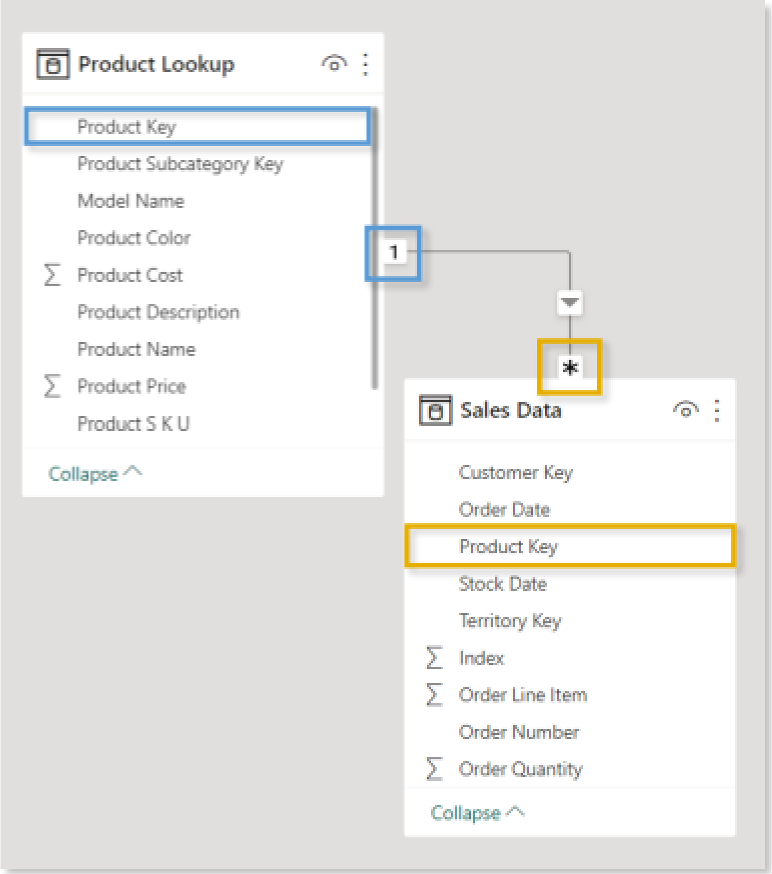
*Attributes from **Product Lookup** table*

date	product_id	quantity	day_of_month	month	year	weekday	month_name	quarter	product_brand	product_name	product_sku	product_weight
1/1/1997	869	5	1	1	1997	Wednesday	January	Q1	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/7/1997	869	2	7	1	1997	Tuesday	January	Q1	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/3/1997	1	4	3	1	1997	Friday	January	Q1	Washington	Washington Berry Juice	90748583674	8.39
1/1/1997	1472	3	1	1	1997	Wednesday	January	Q1	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/6/1997	1472	2	6	1	1997	Monday	January	Q1	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/5/1997	2	4	5	1	1997	Sunday	January	Q1	Washington	Washington Mango Drink	96516502499	7.42
1/1/1997	76	4	1	1	1997	Wednesday	January	Q1	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/1/1997	76	2	1	1	1997	Wednesday	January	Q1	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/5/1997	3	2	5	1	1997	Sunday	January	Q1	Washington	Washington Strawberry Drink	58427771925	13.1
1/7/1997	3	2	7	1	1997	Tuesday	January	Q1	Washington	Washington Strawberry Drink	58427771925	13.1
1/1/1997	320	3	1	1	1997	Wednesday	January	Q1	Excellent	Excellent Cranberry Juice	36570182442	16.4

You can, but it’s extremely inefficient!

- Merging tables creates **redundancy** and often requires **significantly more memory and processing power** to analyze compared to a relational model with multiple small tables

RELATIONSHIP CARDINALITY



Cardinality refers to the uniqueness of values in a column.

- Ideally, all relationships in the data model should follow a **one-to-many** cardinality: **one** instance of each primary key, and **many** instances of each foreign key

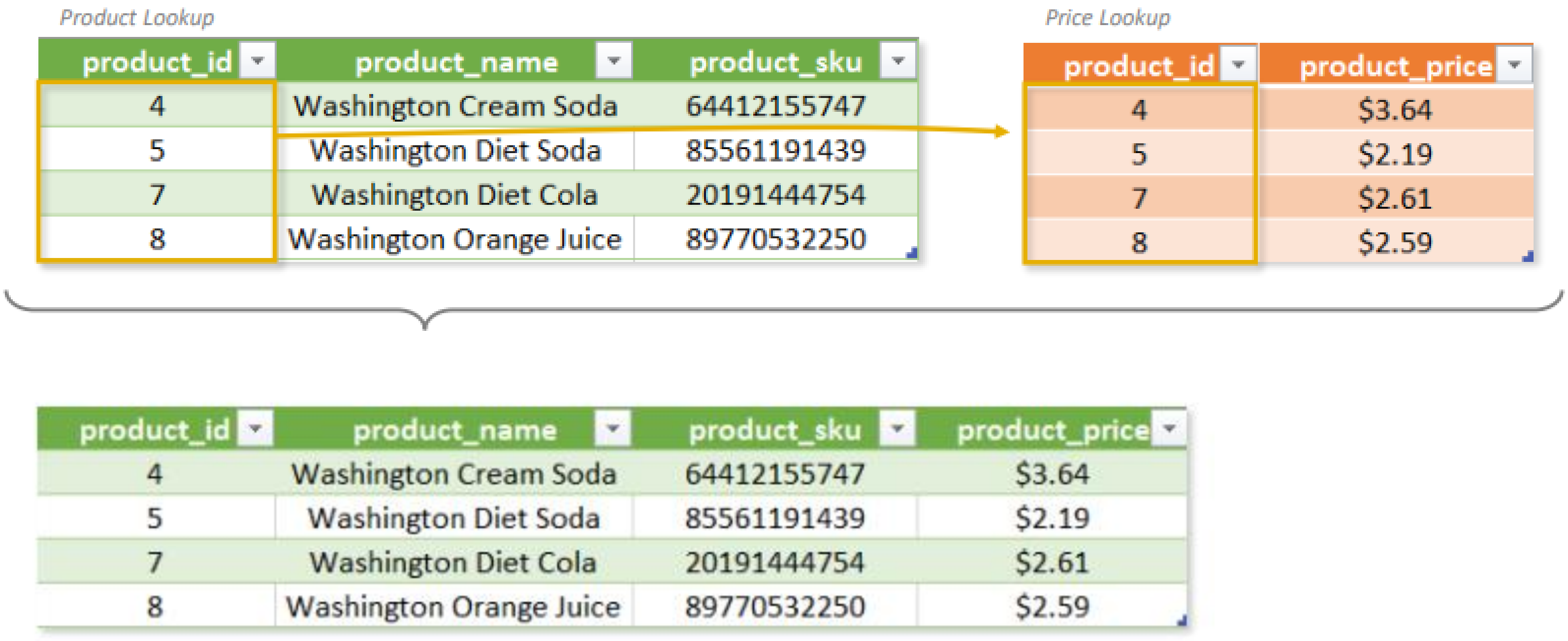
In this example there is only ONE instance of each Product Key in the Product table (noted by a "1"), since each row contains attributes of a single product (name, description, price, etc.)

*There are MANY instances of each Product Key in the Sales table (noted by an asterisk *), since there are multiple sales for each product*

ONE-TO-ONE CARDINALITY

ONE TO ONE - In a one-to-one relationship, one record in a table is associated with one and only one record in another table.

- Connecting the two tables above using product_id creates a one-to-one relationship, since each product ID only appears once in each table
- This isn't necessarily a "bad" relationship, but you can simplify the model by merging the tables into a single, valid dimension table



Product Lookup

product_id	product_name	product_sku
4	Washington Cream Soda	64412155747
5	Washington Diet Soda	85561191439
7	Washington Diet Cola	20191444754
8	Washington Orange Juice	89770532250

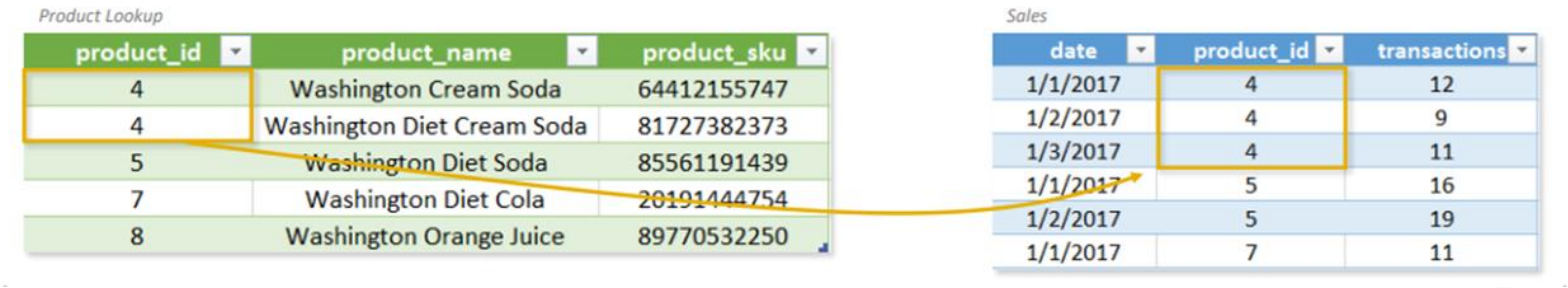
Price Lookup

product_id	product_price
4	$3.64
5	$2.19
7	$2.61
8	$2.59

product_id	product_name	product_sku	product_price
4	Washington Cream Soda	64412155747	$3.64
5	Washington Diet Soda	85561191439	$2.19
7	Washington Diet Cola	20191444754	$2.61
8	Washington Orange Juice	89770532250	$2.59

MANY-TO-MANY CARDINALITY

MANY TO MANY - a many-to-many relationship occurs when multiple records in a table are associated with multiple records in another table. Relational database systems usually don't allow you to implement a direct many-to-many relationship between two tables. We can solve this problem using “cross-reference table” (“join table”)



Product Lookup

product_id	product_name	product_sku
4	Washington Cream Soda	64412155747
4	Washington Diet Cream Soda	81727382373
5	Washington Diet Soda	85561191439
7	Washington Diet Cola	20191444754
8	Washington Orange Juice	89770532250

Sales

date	product_id	transactions
1/1/2017	4	12
1/2/2017	4	9
1/3/2017	4	11
1/1/2017	5	16
1/2/2017	5	19
1/1/2017	7	11

- If we try to connect the tables above using **product_id**, we’ll get a **many-to-many** relationship warning since there are multiple instances of product_id in both tables
- Even if we force this relationship, how would we know which product was actually sold on each date – **Cream Soda** or **Diet Cream Soda**?

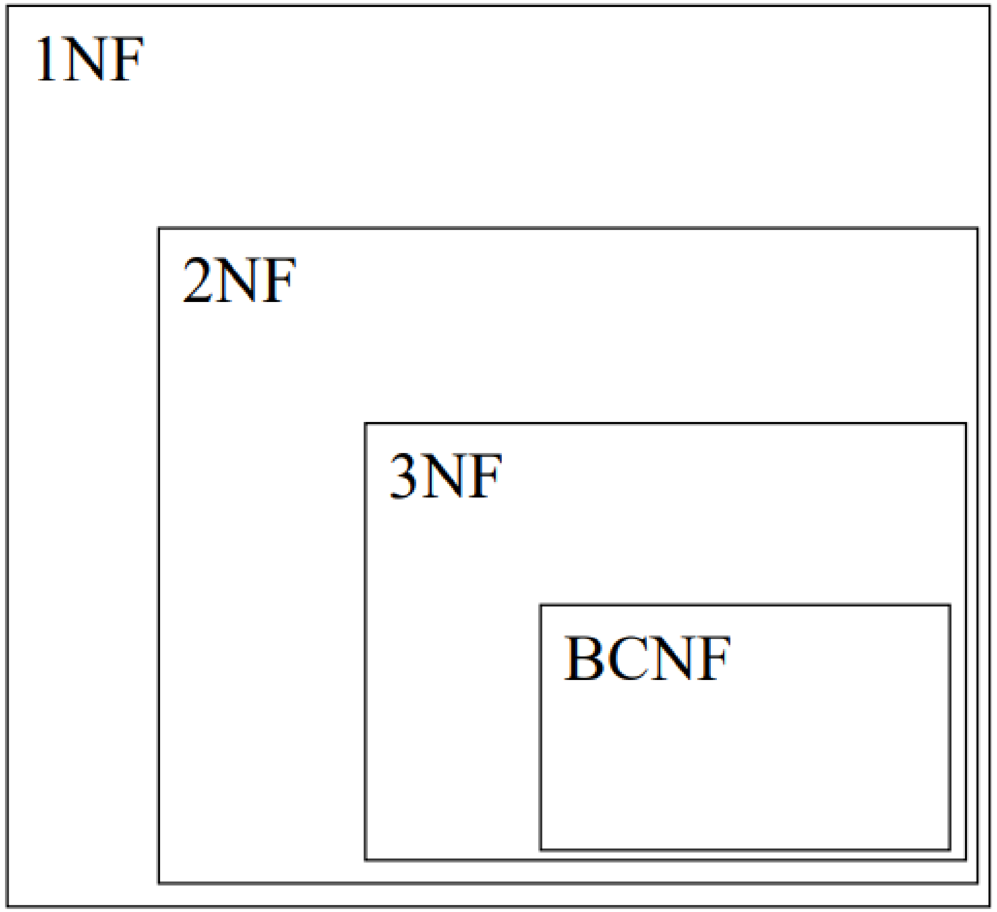
NORMALIZATION

Normalization is a database design technique that organizes tables in a manner that reduces redundancy and dependency of data. Normalization divides larger tables into smaller tables and links them using relationships. The purpose of Normalization is to eliminate redundant (useless) data and ensure data is stored logically.

BENEFITS OF NORMALIZATION

- improved data integrity (no redundant data)
- smaller databases
- better performance
- fewer indexes
- less management
- prevents data modification anomalies

NORMALIZATION



There is a sequence to normal forms:

1NF is considered the weakest,

2NF is stronger than 1NF,

3NF is stronger than 2NF, and

BCNF is considered the strongest

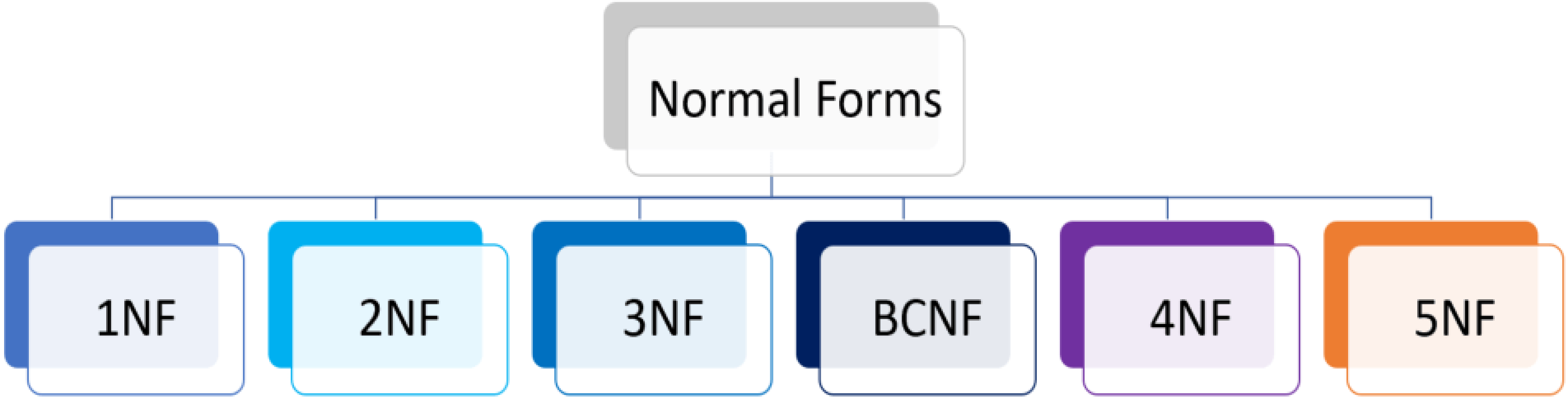
Also,

any relation that is in BCNF, is in 3NF;

any relation in 3NF is in 2NF; and

any relation in 2NF is in 1NF.

TYPES OF NORMALIZATION



Normal Forms

1NF

2NF

3NF

BCNF

4NF

5NF

First Normal Form

We say a relation is in 1NF if all values stored in the relation are single-valued and atomic. 1NF places restrictions on the structure of relations. Values must be simple.

The following is not in 1NF:

Name	Birth_date	Certifications	Certification 1	Certification 2	Certification 3
John	1990/02/12	PBI, MCSA, MTA	PBI	MCSA	MTA
Kelly	1988/10/01	Excel	Excel	NULL	NULL
Bryan	1991/01/15	PBI, Excel	PBI	Excel	NULL
Rob	1986/08/07	PBI	PBI	NULL	NULL

To obtain 1NF relations we must, without loss of information, eliminate repeating groups of data or repeating columns + primary key.

First Normal Form

We say a relation is in 1NF if all values stored in the relation are single-valued and atomic.
1NF places restrictions on the structure of relations. Values must be simple.

Learner_ID	Name	Birth_date	Certification_ID	Certification_name	Vendor_name	Date_acquired
1	John	1990/02/12	1	PBI	Microsoft	2017/01/08
1	John	1990/02/12	2	MCSA	Microsoft	2020/04/16
1	John	1990/02/12	3	MTA	Microsoft	2021/05/12
2	Kelly	1988/10/01	4	Excel	Microsoft	2018/06/03
3	Bryan	1991/01/15	1	PBI	Microsoft	2019/12/11
3	Bryan	1991/01/15	4	Excel	Microsoft	2020/04/07
4	Rob	1986/08/07	1	PBI	Microsoft	2022/01/15

An outer join between will produce the information we saw before

Second Normal Form

Note: If PK is set as a single then you don't need to do anything, the table is already in 2NF

- A relation will be in 2NF if it is in 1NF.
- All non-key attributes are fully functional dependent on the primary key.
- Create separate tables for sets of values that apply to multiple records.
- Relate these tables with a foreign key.

Learner_ID	Name	Birth_date
1	John	1990/02/12
2	Kelly	1988/10/01
3	Bryan	1991/01/15
4	Rob	1986/08/07

Certification _ID	Certification _name	Vendor_ name
1	PBI	Microsoft
2	MCSA	Microsoft
3	MTA	Microsoft
4	Excel	Microsoft

Second Normal Form

Learner_ID	Certification_ID	Date_acquired
1	1	2017/01/08
1	2	2020/04/16
1	3	2021/05/12
2	4	2018/06/03
3	1	2019/12/11
3	4	2020/04/07
4	1	2022/01/15

Third Normal Form

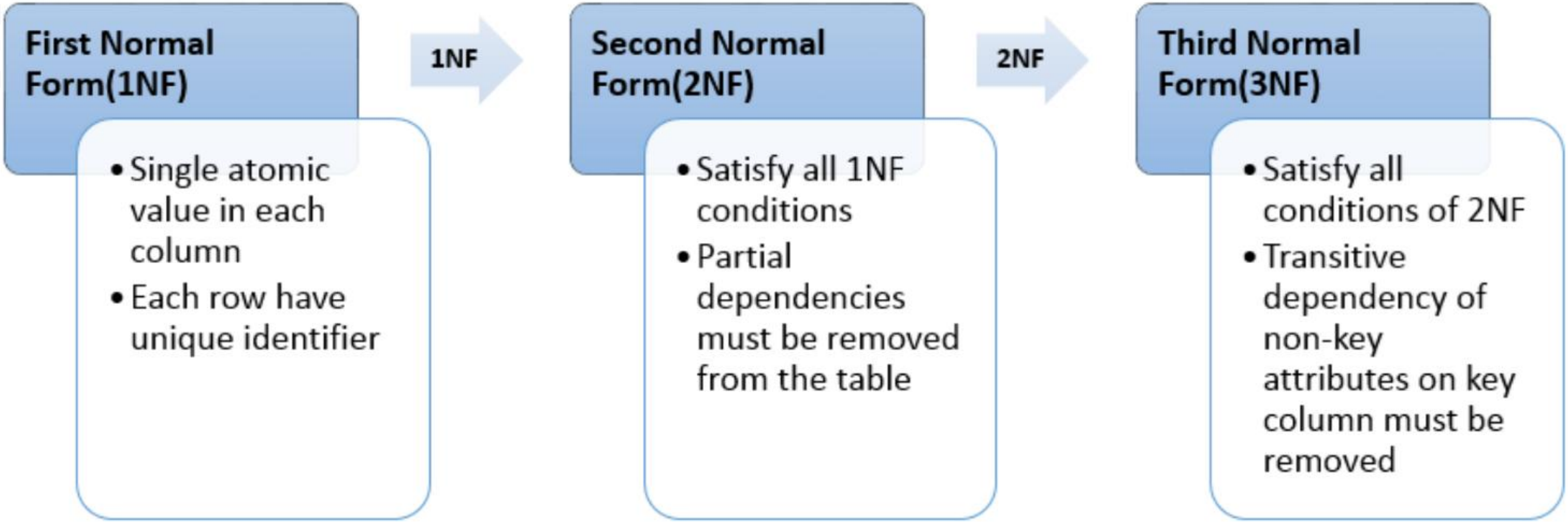
- 3NF satisfy all conditions of 2NF
- A relation in 3NF will not have any transitive dependencies of non-key attributes on key column must be removed.

Learner_ID	Name	Birth_date
1	John	1990/02/12
2	Kelly	1988/10/01
3	Bryan	1991/01/15
4	Rob	1986/08/07

Certification_ID	Certification_name	Vendor_name
1	PBI	Microsoft
2	MCSA	Microsoft
3	MTA	Microsoft
4	Excel	Microsoft

Certification_ID	Certification_name	Vendor_ID
1	PBI	1
2	MCSA	1
3	MTA	1
4	Excel	1

Vendor_ID	Vendor_name
1	Microsoft
2	Amazon



First Normal Form(1NF)

- Single atomic value in each column
- Each row have unique identifier

1NF

Second Normal Form(2NF)

- Satisfy all 1NF conditions
- Partial dependencies must be removed from the table

2NF

Third Normal Form(3NF)

- Satisfy all conditions of 2NF
- Transitive dependency of non-key attributes on key column must be removed

Third Normal Form

Learner_ID	Name	Birth_date
1	John	1990/02/12
2	Kelly	1988/10/01
3	Bryan	1991/01/15
4	Rob	1986/08/07

Learner_ID	Certification_ID	Date_acquired
1	1	2017/01/08
1	2	2020/04/16
1	3	2021/05/12
2	4	2018/06/03
3	1	2019/12/11
3	4	2020/04/07
4	1	2022/01/15

Certification_ID	Certification_name	Vendor_ID
1	PBI	1
2	MCSA	1
3	MTA	1
4	Excel	1

Vendor_ID	Vendor_name
1	Microsoft
2	Amazon

Boyce Codd normal form (BCNF)

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF.

A table complies with **BCNF** if it is in 3NF and for every **functional dependency** X->Y, X should be the super key of the table.

Fourth Normal Form (4NF)

The **Fourth Normal Form (4NF)** builds on BCNF by addressing a specific problem: **multi-valued dependencies**.

A table is in 4NF if:

- It already satisfies all the conditions of BCNF.
- It contains no multi-valued dependencies.

Denormalization

- Adds redundant data to improve query performance by reducing the number of joins required
- Simplifies data access by storing all data in one place
- Can result in data inconsistency if not properly managed
- Increases storage requirements due to the duplicated data
- Simplifies queries by reducing the number of joins required, which can result in faster query execution

Database design

- Designing a database means reflecting the **domain (subject area)** in data form
- A database is always a **model of the real-world domain**

Poor design can lead to:

- Invalid or inconsistent data
- Loss of information (missing relationships or entities)

Stages of database design

- **Requirements Analysis** – study the domain and collect requirements.
- **Logical Data Modeling** – build a logical model of the domain (entities, attributes, relationships).
- **Physical Design & Normalization** – implement the model in a database system and apply normalization

Analyzing the requirements of the subject area

1. USE CASE compilation
2. Analytical process involving stakeholders (owners and domain experts)
3. Conceptual database schema

Logical modeling

- Continues requirements analysis
- Refines the conceptual model into a detailed logical model
- Defines:
  - Primary and foreign keys
  - Data types for attributes
  - Logical constraints
- Normalization usually applied up to 3NF (higher forms are rare in practice)

Physical design and normalization

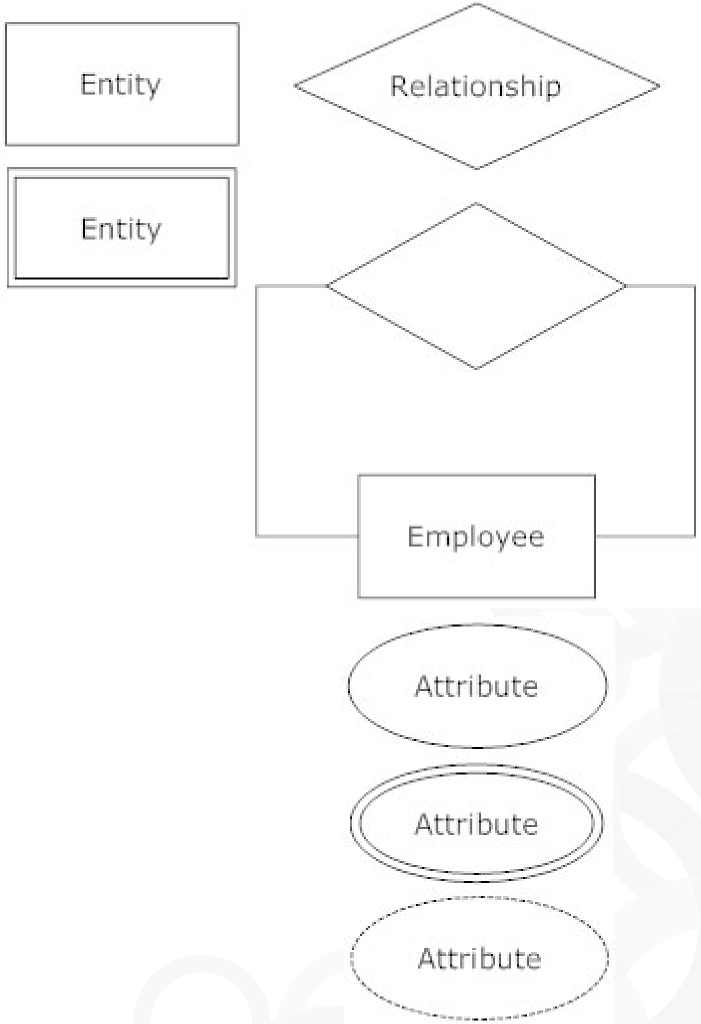
- Choosing a specific DBMS
- Defining DBMS-specific data types
- Creating indexes for fast data access
- Defining views (for abstraction and convenience)
- Setting access restrictions and security rules
- Agreeing on naming conventions in advance

ER Diagrams

Entity-Relationship Diagrams, or ERDs, are a key tool in database design.
They show entities, attributes, and the relationships between them. ERDs represent the logical structure of the database before implementation. They are especially useful for communication, because both technical and non-technical people can understand them..

Common ERD Symbols

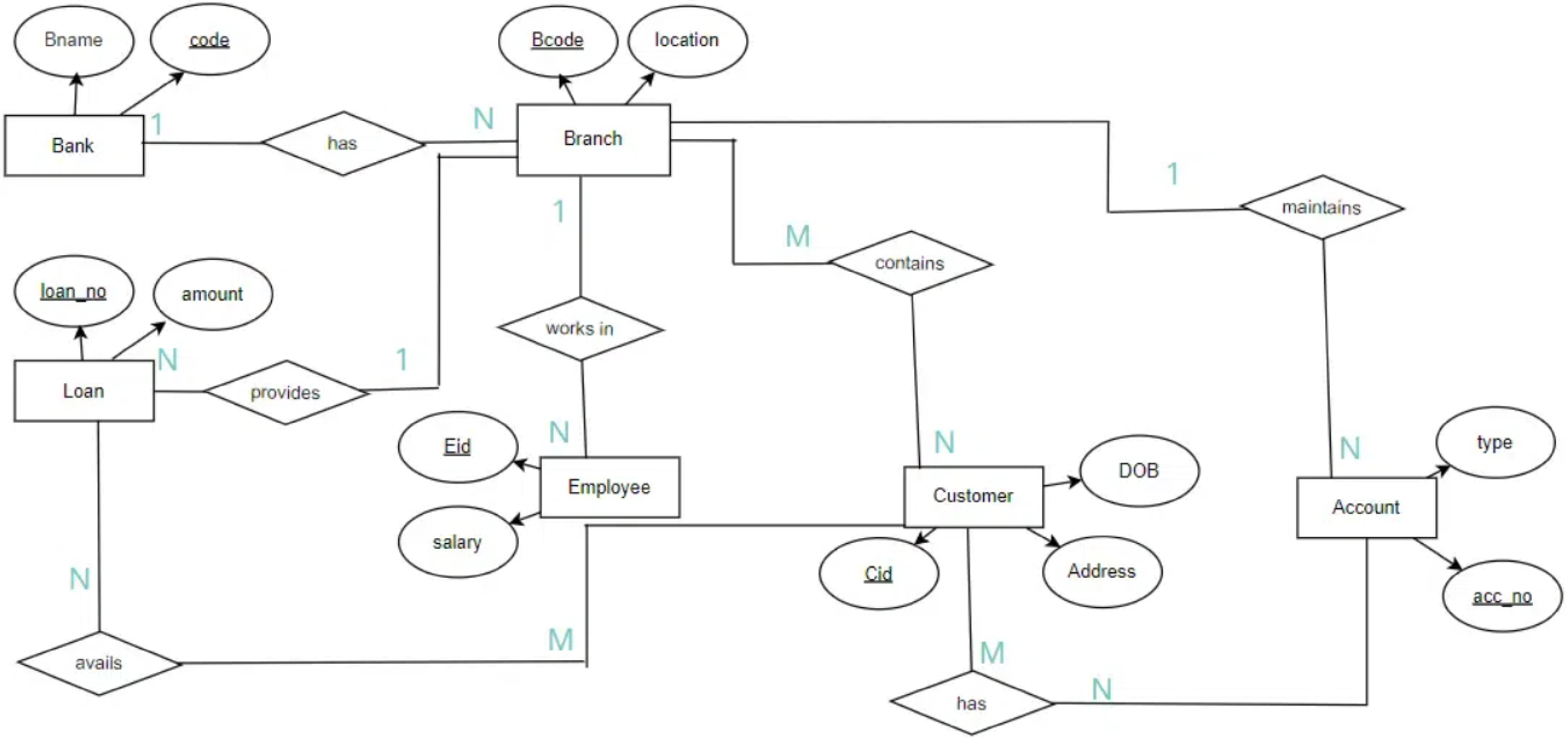
An ER diagram has three main components: entities, relationships, and attributes connected by lines.
Entities, which are represented by rectangles. An entity is an object or concept about which you want to store information.
Relationships, which are represented by diamond shapes, show how two entities share information in the database.
Attributes, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity.
Connecting lines, solid lines that connect attributes and show the relationships of entities in the diagram.
Cardinality specifies the numerical attribute of the relationship between entities. It can be one-to-one, many-to-one, or many-to-many.



How to Draw an Entity Relation Diagram (ERD)

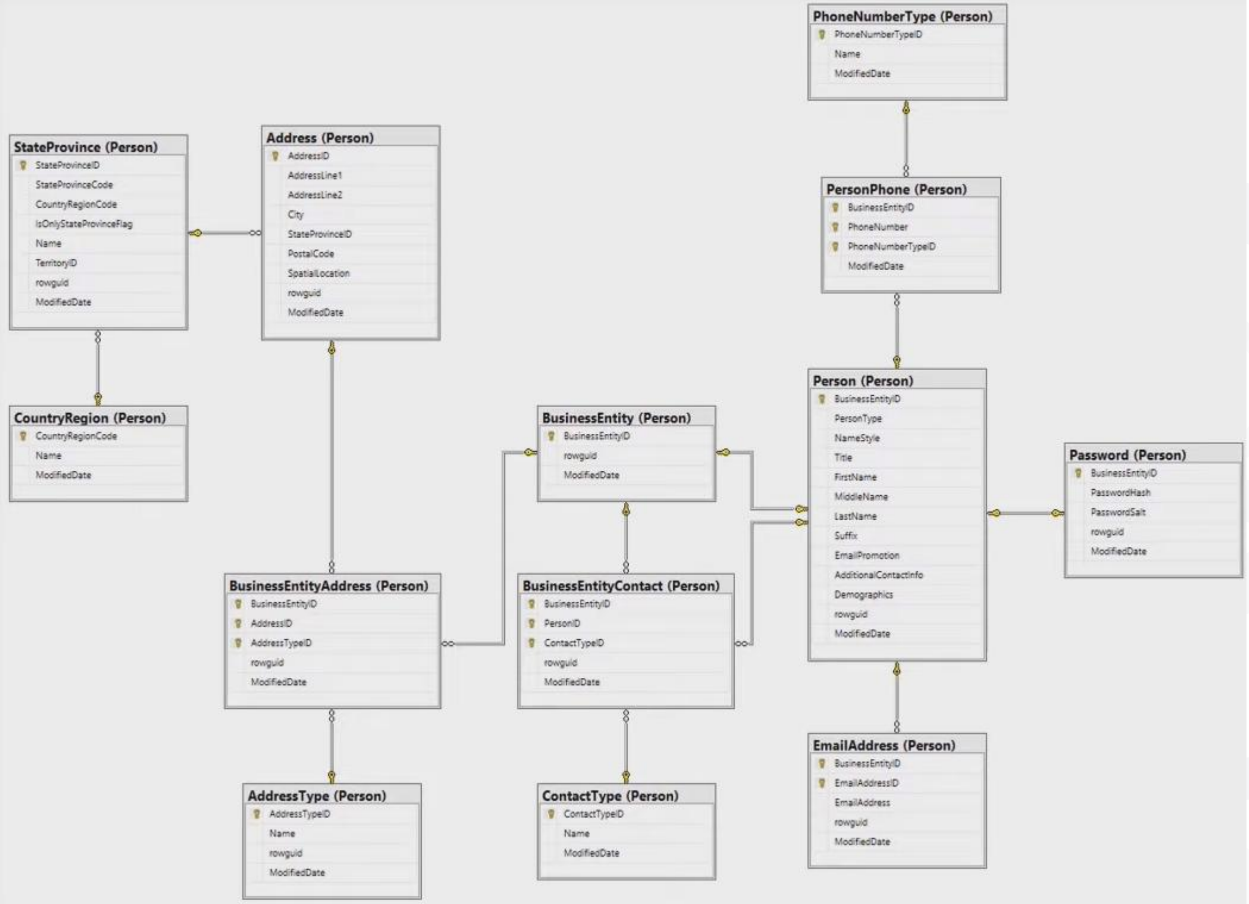
1. Identifying Entities Determine the main objects you want to represent in the database.
2. Defining Attributes Identify the properties(attributes) of properties of each entity.
3. Specifying Relationships Create relationships between entities to specify how entities interact with each other.
4. Drawing Entities Draw entities as rectangle and write the name.
5. Adding Attributes. To add attributes of an entity write attributes inside the rectangle or connect them with lines.
6. Connecting Entities Draw lines between the related entities to represent their connection.
7. Specifying Cardinality Indicate the minimum and maximum number of relationship instances associated with an entity using notations like crow's foot.
8. Organizing ER Diagram Organize all entities and relationships in a clean way for better readability and understanding.

How to Draw an Entity Relation Diagram (ERD)



ER(Entity relationship) diagrams

1. MySQL Workbench
2. ORACLE SQL developer
3. Data Modeler
4. pgModeler
5. SQL power Architect



PhoneNumberType (Person)
PhoneNumberTypeID
Name
ModifiedDate

StateProvince (Person)
StateProvinceID
StateProvinceCode
CountryRegionCode
IsOnlyStateProvinceFlag
Name
TerritoryID
rowguid
ModifiedDate

Address (Person)
AddressID
AddressLine1
AddressLine2
City
StateProvinceID
PostalCode
SpatialLocation
rowguid
ModifiedDate

PersonPhone (Person)
BusinessEntityID
PhoneNumber
PhoneNumberTypeID
ModifiedDate

CountryRegion (Person)
CountryRegionCode
Name
ModifiedDate

BusinessEntity (Person)
BusinessEntityID
rowguid
ModifiedDate

Person (Person)
BusinessEntityID
PersonType
NameStyle
Title
FirstName
MiddleName
LastName
Suffix
EmailPromotion
AdditionalContactInfo
Demographics
rowguid
ModifiedDate

Password (Person)
BusinessEntityID
PasswordHash
PasswordSalt
rowguid
ModifiedDate

BusinessEntityAddress (Person)
BusinessEntityID
AddressID
AddressTypeID
rowguid
ModifiedDate

BusinessEntityContact (Person)
BusinessEntityID
PersonID
ContactTypeID
rowguid
ModifiedDate

EmailAddress (Person)
BusinessEntityID
EmailAddressID
EmailAddress
rowguid
ModifiedDate

AddressType (Person)
AddressTypeID
Name
rowguid
ModifiedDate

ContactType (Person)
ContactTypeID
Name
ModifiedDate

Basic DB design tips

- Tables represent **real-world objects, events, or abstractions** from the domain
- Fields (columns) represent **attributes** of these objects
- Rows represent **instances** of the objects
- Data must be **valid** (no impossible values, e.g., Dogs with 97 legs)
- Data must be **consistent** (fields that depend on each other cannot contradict each other)

Bad practices

1. Ignoring normalization leads to data redundancy
2. No naming standards on the project.
3. One table for different types of data
4. Relevance of data representation (domain may change).