Normalization: process of efficiently organizing data in the DB.

↓

RELATIONS
(attributes grouped together)

↓

Accurate representation of data, relationships and constraints.

Goal:  - Eliminate redundant data in a DB.
       - Ensure data dependencies make sense.

Guidelines for ensuring that DBs are normalized → normal forms: 1NF, 2NF, 3NF, BCNF.

↓

Normalization: series of tests on a relation to determine whether it satisfies or violates the requirements of a normal form.

Note: meet practical business requirements.

↓

Normalization: A technique for producing a set of relations with desirable properties, given the data requirements of an enterprise.

Reason for normalization: to prevent possible corruption of DB stemming from update anomalies (insertion, deletion, modification).
FUNCTIONAL DEPENDENCIES

Normalization: a formal method that identifies relations based on their primary key and the **functional dependencies** among their attributes.

Constraint between attributes.

**Functional dependency**: Describes the relationship between attributes in a relation. If A and B are attributes of a relation R, B is functionally dependent on A (den. A → B), if
each value of A in R is associated with exactly one value of B in R.

B is functionally dependent on A

**Determinant:** attribute or set of attributes on the left hand side of the arrow.

**Identify the candidate key** for a relation: recognise the attribute (group of attributes) that uniquely identifies each row in a relation. All of the attributes that are not part of the primary key (non-primary key attributes) should be functionally dependent on the key.

---

**FUNCTIONAL DEPENDENCIES**

1. Staff_No $\rightarrow$ Position (Position is functionally dependent on Staff_No)
2. Staff number 04.21 $\rightarrow$ Manager
3. Position $\rightarrow$ Staff_No (Staff_No is not functionally dependent on Position)
4. Manager $\rightarrow$ Staff number 04.21
5. Manager $\rightarrow$ Staff number 933
PROCESS OF NORMALIZATION

Unnormalized form (UNF): A table that contains one or more repeating groups.

Repeating group: an attribute or group of attributes within a table that occurs with multiple values for a single occurrence of the nominated key attributes of that table.
First normal form (1NF): A relation in which the intersection of each row and column contains one and only one value.

UNF $\rightarrow$ 1NF: remove repeating groups:
- Entering appropriate data in the empty columns of rows.
- Placing repeating data along with a copy of the original key attribute in a separate relation. Identifying a primary key for each of the new relations.
**Full functional dependency**: If A and B are attributes of a relation, B is fully functionally dependent on A if B is functionally dependent on A, but not any proper subset of A.

A → B is partially dependent if there is some attribute that can be removed from A and the dependency still holds.
Ex.

- Staff_No, SName $\rightarrow$ Branch_No  partial
- Staff_No $\rightarrow$ Branch_No  full

**Second normal form (2NF):** A relation that is in 1NF and every non-primary key attribute is fully functionally dependent on the primary key.

Note: applies to relations with composite keys (primary key composed of two or more attributes). A relation with a single attribute primary key is in at least 2NF.

**1NF $\rightarrow$ 2NF:*** remove partial dependencies: the functionally dependent attributes are removed from the relation by placing them in a new relation along with a copy of their determinant.
Transitive dependency: A condition where A, B and C are attributes of a relation such that if $A \rightarrow B$ and $B \rightarrow C$, then $C$ is transitively dependent on $A$ via $B$ (provided that $A$ is not functionally dependent on $B$ or $C$).
Third normal form (3NF): A relation that is in 1NF and 2NF, and in which no non-primary key attribute is transitively dependent on the primary key.

2NF → 3NF: remove transitive dependencies: the translatively dependent attributes are removed from the relation by placing them in a new relation along with a copy of their determinant.

**FUNCTIONAL DEPENDENCIES**

Customer Relation
Customer_No → CName

Rental Relation
Customer_No, Property_No → RentStart, RentFinish

Property_Owner Relation
Property_No → PAddress, Rent, Owner_No, OName
Owner_No → OName TRANSITIVE

**PROPERTY FOR RENT RELATION**

<table>
<thead>
<tr>
<th>Property_No</th>
<th>PAddress</th>
<th>Rent</th>
<th>Owner_No</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG4</td>
<td>6 Lawrence Street, Glasgow</td>
<td>350</td>
<td>C040</td>
</tr>
<tr>
<td>PG16</td>
<td>5 Novar Drive, Glasgow</td>
<td>450</td>
<td>C093</td>
</tr>
<tr>
<td>PG36</td>
<td>2 Manor Road Glasgow</td>
<td>375</td>
<td>C093</td>
</tr>
</tbody>
</table>

Owner Relation

<table>
<thead>
<tr>
<th>Owner_No</th>
<th>OName</th>
</tr>
</thead>
<tbody>
<tr>
<td>C040</td>
<td>Tina Murphy</td>
</tr>
<tr>
<td>C093</td>
<td>Tony Shaw</td>
</tr>
</tbody>
</table>

The normalization process decomposes the original relation using a series of relation algebra projections. This result in a nonloss (lossless) decomposition, which is reversible using the natural join operation (primary key / foreign key mechanism).
Boyce-Codd Normal Form (BCNF): A relation is in BCNF if and only if every determinant is a candidate key.

Notes:

BCNF is a stronger form of 3NF

$$\text{BCNF} \Rightarrow \text{3NF}$$

$$\text{3NF} \nRightarrow \text{BCNF}$$
Violation of BCNF happen under specific conditions:
- A relation contains two (or more) composite candidate keys,
- which overlap and share at least one attribute in common.

**Transformation to BCNF:** remove violating functional dependencies by placing them in a new relation.

**EXAMPLE**

<table>
<thead>
<tr>
<th>Client_No</th>
<th>Interview_Date</th>
<th>Interview_Time</th>
<th>Staff_No</th>
<th>Room_No</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR76</td>
<td>13-May-95</td>
<td>10.30</td>
<td>SG5</td>
<td>G101</td>
</tr>
<tr>
<td>CR56</td>
<td>13-May-95</td>
<td>12.00</td>
<td>SG5</td>
<td>G101</td>
</tr>
<tr>
<td>CR74</td>
<td>13-May-95</td>
<td>12.00</td>
<td>SG37</td>
<td>G102</td>
</tr>
<tr>
<td>CR56</td>
<td>1-Jul-95</td>
<td>10.30</td>
<td>SG5</td>
<td>G102</td>
</tr>
</tbody>
</table>

The Client_Interview relation has the following functional dependencies:
- Client_No, Interview_Date → Interview_Time, Staff_No, Room_No
- Staff_No, Interview_Date, Interview_Time → Client_No
- Staff_No, Interview_Date → Room_No

**INTERVIEW RELATION**

<table>
<thead>
<tr>
<th>Client_No</th>
<th>Interview_Date</th>
<th>Interview_Time</th>
<th>Staff_No</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR76</td>
<td>13-May-95</td>
<td>10.30</td>
<td>SG5</td>
</tr>
<tr>
<td>CR56</td>
<td>13-May-95</td>
<td>12.00</td>
<td>SG5</td>
</tr>
<tr>
<td>CR74</td>
<td>13-May-95</td>
<td>12.00</td>
<td>SG37</td>
</tr>
<tr>
<td>CR56</td>
<td>1-Jul-95</td>
<td>10.30</td>
<td>SG5</td>
</tr>
</tbody>
</table>

**STAFF_ROOM RELATION**

<table>
<thead>
<tr>
<th>Staff_No</th>
<th>Interview_Date</th>
<th>Room_No</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG5</td>
<td>13-May-95</td>
<td>G101</td>
</tr>
<tr>
<td>SG37</td>
<td>13-May-95</td>
<td>G102</td>
</tr>
<tr>
<td>SG5</td>
<td>1-Jul-95</td>
<td>G102</td>
</tr>
</tbody>
</table>

Interview (Client_No, Interview_Date, Interview_Time, Staff_No)
Staff_Room (Staff_No, Interview_Date, Room_No)