# Normalization

# Trivial Functional Dependency

- **Trivial** – If a functional dependency (FD) $X \rightarrow Y$ holds, where Y is a subset of X, then it is called a trivial FD. Trivial FDs always hold.
- **Non-trivial** – If an FD $X \rightarrow Y$ holds, where Y is not a subset of X, then it is called a non-trivial FD.
- **Completely non-trivial** – If an FD $X \rightarrow Y$ holds, where x intersect $Y = \Phi$, it is said to be a completely non-trivial FD.

# Full Dependency

- If the determinant of the dependency is either a candidate key or a super key then such dependency is defined as full dependency.
- Note: (1) There is no condition on dependent.
  (2) The amount of redundancy caused by each and every full dependency is zero.

# Partial Dependency

- If a non-prime attribute of the relation is getting derived by only a part of the composite candidate key then such dependency is defined as partial dependency.
- Note:(1) While identifying the partial dependencies we shall verify that the dependent must be a non-prime attribute and the determinant must be a part of candidate key.
- (2) In order to have partial dependency in a relation, it should have at least one composite candidate key. (3) Partial dependency will cause redundancy and hence it will get eliminated by normalization technique.

# Transitive Dependency

- If a non-prime attribute of the relation is getting derived by either another non-prime attribute or the combination of part of the candidate key along with a non-prime attribute then such dependency would be defined as transitive dependency.
- Note: (1) While identifying the transitive dependency make sure that the dependent is a non-prime attribute and the determinant could be either a non-prime attribute or the combination of part of the candidate key along with non-prime attribute.
  (2) Transitive dependency will cause redundancy in the relation and hence it will get eliminated by normalization technique.

Example:
For relation R(A,B,C,D,E)
if AD is a candidate key then
Prime attributes = {A,D}
Non prime attributes = {B,C,E}

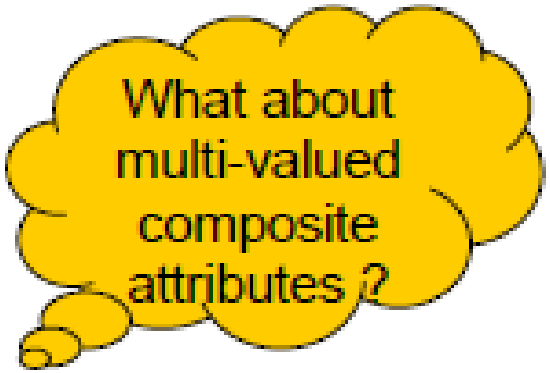
- Prime attributes = parts of candidate key of given relational table.
- Non-prime attributes = not a part of candidate key.

# 1NF

- 1NF: The relation should have no non-atomic values.

$R_{non1NF}$

| ID | Name | LivesIn |
|---|---|---|
| 100 | Pettersson | {Stockholm, Linköping} |
| 101 | Andersson | {Linköping} |
| 102 | Svensson | {Ystad, Hjo, Berlin} |





$R1_{1NF}$

| ID | Name |
|---|---|
| 100 | Pettersson |
| 101 | Andersson |
| 102 | Svensson |

$R2_{1NF}$

| ID | LivesIn |
|---|---|
| 100 | Stockholm |
| 100 | Linköping |
| 101 | Linköping |
| 102 | Ystad |
| 102 | Hjo |
| 102 | Berlin |

- 2NF: *no* ***nonprime attribute should be functionally*** dependent on a ***part of a candidate key (= partial*** dependency).
- 2NF: Every **nonprime attribute is fully functionally** dependent on every candidate key.
- A table is said to be in 2NF if both the following conditions hold:

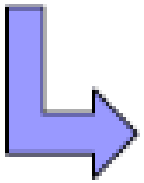
1. Table is in 1NF (First normal form)
2. No non-prime attribute is dependent on the proper subset of any candidate key of table.

# 2NF

- 2NF: *no* ***nonprime*** *attribute* should be functionally dependent on a ***part*** of a candidate key (= partial dependency).

$R_{non2NF}$

| *EmpID* | *Dept* | Work% | EmpName |
|---|---|---|---|
| *100* | *Dev* | 50 | Baker |
| *100* | *Support* | 50 | Baker |
| *200* | *Dev* | 80 | Miller |



Normalization

$R1_{2NF}$

| *EmpID* | EmpName |
|---|---|
| *100* | Baker |
| *200* | Miller |

$R2_{2NF}$

| *EmpID* | *Dept* | Work% |
|---|---|---|
| *100* | *Dev* | 50 |
| *100* | *Support* | 50 |
| *200* | *Dev* | 80 |

| Teacher_id | Subject | Teacher_name |
|---|---|---|
| 1 | Maths | A |
| 1 | Physics | A |
| 2 | Biology | B |
| 3 | Physics | C |
| 3 | Chemistry | C |

**Candidate Keys**:
{teacher_id, subject}
**Non prime attribute**:
teacher_name

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher_name is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table".

To make the table complies with 2NF we can break it in two tables.

# teacher_details table:

| Teacher_id | Teacher_name |
|---|---|
| 1 | A |
| 2 | B |
| 3 | C |

# teacher_subject table:

| Teacher_id | Subject |
|---|---|
| 1 | Maths |
| 1 | Physics |
| 2 | Biology |
| 3 | Physics |
| 3 | Chemistry |

- 3NF: 2NF + no **nonprime attribute is transitively** dependent on any candidate key.
- 3NF: **2NF + no nonprime attribute should be** functionally dependent on a set of nonprime attributes.
- A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

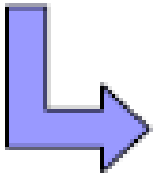
1. X is a **super key** of table
2. Y is a prime attribute of table

# 3NF

- 3NF: **2NF +** no **nonprime** attribute should be functionally dependent on a set of nonprime attributes

$R_{non3NF}$

| *ID* | Name | Zip | City |
|---|---|---|---|
| *100* | Andersson | 58214 | Linköping |
| *101* | Björk | 10223 | Stockholm |
| *102* | Carlsson | 58214 | Linköping |



Normalization

$R1_{3NF}$

| *ID* | Name | Zip |
|---|---|---|
| *100* | Andersson | 58214 |
| *101* | Björk | 10223 |
| *102* | Carlsson | 58214 |

$R2_{3NF}$

| *Zip* | City |
|---|---|
| *58214* | Linköping |
| *10223* | Stockholm |

| emp_id | emp_name | emp_zip | emp_state | emp_city | emp_district |
|---|---|---|---|---|---|
| 1001 | John | 282005 | UP | Agra | Dayal Bagh |
| 1002 | Ajeet | 222008 | TN | Chennai | M-City |
| 1006 | Lora | 282007 | TN | Chennai | Urrapakkam |
| 1101 | Lilly | 292008 | UK | Pauri | Bhagwan |
| 1201 | Steve | 222999 | MP | Gwalior | Ratan |

**Super keys**: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…

**Candidate Keys**: {emp_id}

**Non-prime attributes**: all attributes except emp_id are non-prime as they are not part of any candidate keys.

Here, emp_state, emp_city & emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency.

# employee table:

| emp_id | emp_name | emp_zip |
|---|---|---|
| 1001 | John | 282005 |
| 1002 | Ajeet | 222008 |
| 1006 | Lora | 282007 |
| 1101 | Lilly | 292008 |
| 1201 | Steve | 222999 |

# employee_add table:

| emp_zip | emp_state | emp_city | emp_district |
|---|---|---|---|
| 282005 | UP | Agra | Dayal Bagh |
| 222008 | TN | Chennai | M-City |
| 282007 | TN | Chennai | Urrapakkam |
| 292008 | UK | Pauri | Bhagwan |
| 22999 | MP | Gwalior | Ratan |

# Boyce-Codd Normal Form

- A table R is in BCNF if for every non-trivial FD $A \rightarrow B$, A is a super key.
- BCNF: **Every determinant is a super key** (in practice: every determinant is a candidate key)
- BCNF = decompose if $X \rightarrow A$ is such that X is not a super key and A is a prime attribute.
- Example: Given R(A,B,C,D) and $AB \rightarrow CD$, $C \rightarrow B$. Then R is in 3NF but not in BCNF
- C is a determinant but not a super key (tuples are not uniquely identified in R)

| emp_id | emp_nationality | emp_dept | dept_type | dept_no_of_emp |
|---|---|---|---|---|
| 1001 | Austrian | Production and planning | D001 | 200 |
| 1001 | Austrian | stores | D001 | 250 |
| 1002 | American | design and technical support | D134 | 100 |
| 1002 | American | Purchasing department | D134 | 600 |

emp_id->emp_nationality

emp_dept -> {dept_type, dept_no_of_emp}

**Candidate key**: {emp_id, emp_dept}

The table is not in BCNF as neither emp_id nor emp_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables

emp_nationality table:

| emp_id | emp_nationality |
|---|---|
| 1001 | Austrian |
| 1002 | American |

emp_dept table:

| emp_dept | dept_type | dept_no_of_emp |
|---|---|---|
| Production and planning | D001 | 200 |
| stores | D001 | 250 |
| design and technical support | D134 | 100 |
| Purchasing department | D134 | 600 |

emp_dept_mapping table:

| emp_id | emp_dept |
|---|---|
| 1001 | Production and planning |
| 1001 | stores |
| 1002 | design and technical support |
| 1002 | Purchasing department |

- **Functional dependencies**:
  emp_id -> emp_nationality
  emp_dept -> {dept_type, dept_no_of_emp}
- **Candidate keys**:
  For first table: emp_id
  For second table: emp_dept
  For third table: {emp_id, emp_dept}
- This is now in BCNF as in both the functional dependencies left side part is a key.

- Boyce-Codd Normal Form (BCNF)
  3NF and all tables in the database should be only one primary key.

- Fourth Normal Form (4NF)
  Tables cannot have multi-valued dependencies on a Primary Key.

- Fifth Normal Form (5NF)
  A composite key shouldn't have any cyclic dependencies.

# 4NF

| S_ID | Subject | Activity |
|---|---|---|
| 1 | H | T |
| 1 | E | T |
| 1 | H | S |
| 1 | E | S |
| 2 | M | J |

| S_ID | Subject |
|---|---|
| 1 | H |
| 1 | E |
| 2 | M |

| S_ID | Activity |
|---|---|
| 1 | T |
| 1 | S |
| 2 | J |

# 5NF

| AGENT_COMPANY_PRODUCT | | |
|---|---|---|
| **Agent** | **Company** | **Product** |
| Suneet | ABC | Nut |
| Raj | ABC | Bolt |
| Raj | ABC | Nut |
| Suneet | CDE | Bolt |
| Suneet | ABC | Bolt |

| P1 | |
|---|---|
| **Agent** | **Company** |
| Suneet | ABC |
| Suneet | CDE |
| Raj | ABC |

| P2 | |
|---|---|
| **Agent** | **Product** |
| Suneet | Nut |
| Suneet | Bolt |
| Raj | Bolt |
| Raj | Nut |

| P3 | |
|---|---|
| **Company** | **Product** |
| ABC | Nut |
| ABC | Bolt |
| CDE | Bolt |

| P1 ⋈ P2 | | |
|---|---|---|
| **Agent** | **Company** | **Product** |
| Suneet | ABC | Nut |
| Suneet | ABC | Bolt |
| **Suneet** | **CDE** | **Nut** |
| Suneet | CDE | Bolt |
| Raj | ABC | Bolt |
| Raj | ABC | Nut |

| P3 | |
|---|---|
| **Company** | **Product** |
| ABC | Nut |
| ABC | Bolt |
| CDE | Bolt |

| P1 ⋈ P2 ⋈ P3 | | |
|---|---|---|
| **Agent** | **Company** | **Product** |
| Suneet | ABC | Nut |
| Suneet | ABC | Bolt |
| Suneet | CDE | Bolt |
| Raj | ABC | Nut |
| Raj | ABC | Bolt |

# Properties of decomposition

- Keep all attributes from the universal relation R.
- FD preservation: Preserve the identified functional dependencies.
- Lossless join decomposition: It must be possible to join the smaller tables to arrive at composite information without spurious tuples.

- An algorithm for decomposing a table into 3NF with both properties is as follows:

1. Find the minimal cover of F. Suppose it is E.
2. For each FD $A \rightarrow B$ in E, if b is a non-key attribute, have a table (A, B), where A is the primary key.
3. (optional but good) Merge tables that have identical primary key.
4. Have a table R-B, where B is the set of non-key attributes that appeared on the RHS of some FD in E.

Suppose R= (a, b, c, d, e, f)

The FD set F contains a,b → c,d,e and d → a.

The minimal cover of the functional dependencies is:

a,b-->c

a,b-->d

a,b-->e

d-->a

The candidate keys are

b,f,a

b,f,d

Thus c, e are non-key attributes.

The table is NOT in 2NF because there are partialdependencies .e.g., a.b → c

A FD-preserving decomposition to 3NF works as follows:

Using step 2, we will have two tables (a,b,c) and (a,b,e), where {a,b} is the primary key.

Using step 3, we merge the above table into (a,b,c,e).

Using step 4, we get the table (a,b,d,f).