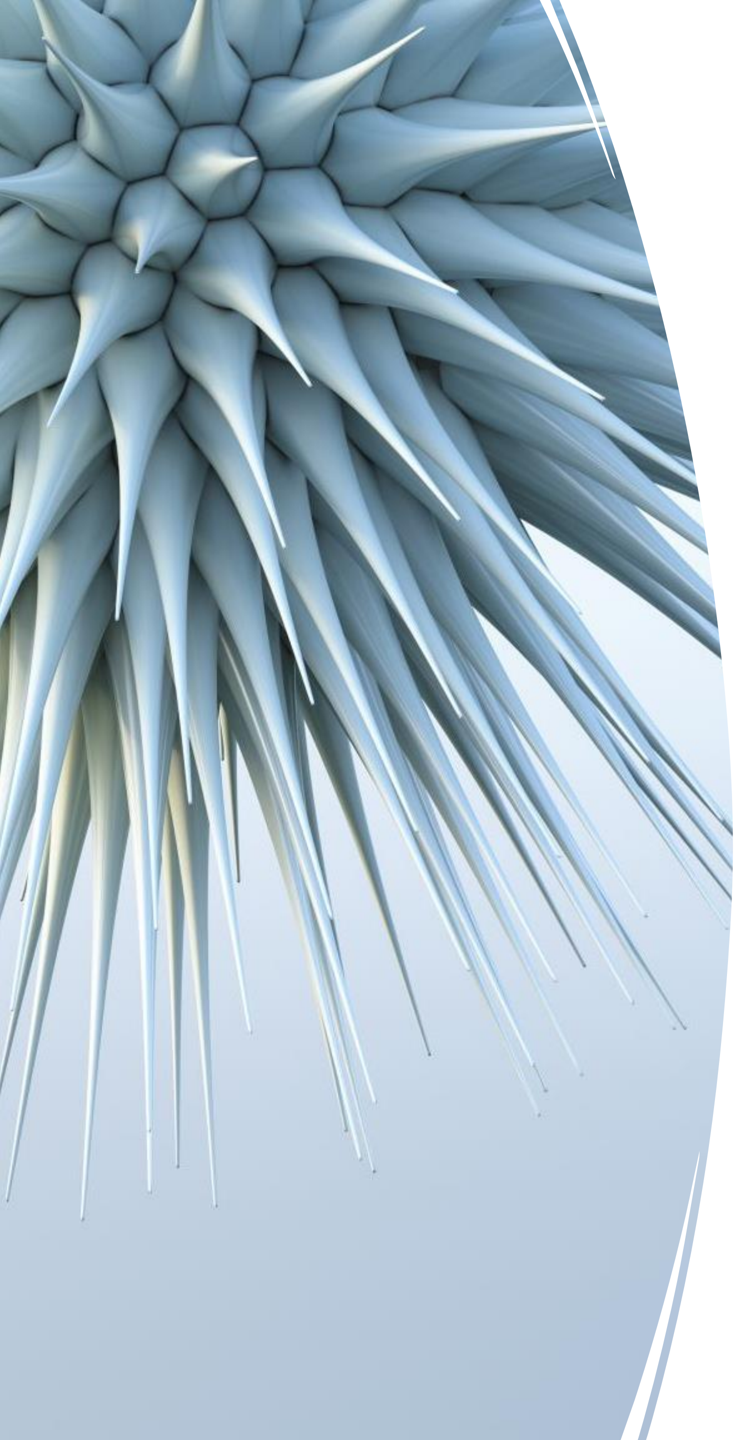


Normalization

EID	EN	DID
1	A	1
2	B	2
3	C	3
4	A	3
5	E	1

Foreign key

DID	DNAME	ADDRESS
1	CSE	BLOCK A
2	ECE	BLOCK B
3	ME	BLOCK C

EID	EN	DID
1	A	1
2	B	2
3	C	3
4	A	3
5	E	1

Foreign key

DID	DNAME	ADDRESS
1	CSE	BLOCK A
2	ECE	BLOCK B
3	ME	BLOCK C

EID	EN	DID	DID	DNAME	ADDRESS
1	A	1	1	CSE	BLOCK A
2	B	2	2	ECE	BLOCK B
3	C	3	3	ME	BLOCK C
4	A	3	3	ME	BLOCK C
5	E	1	1	CSE	BLOCK A

EID	EN	DID
1	A	1
2	B	2
3	C	3
4	A	3
5	E	1

Foreign key

DID	DNAME	ADDRESS
1	CSE	BLOCK A
2	ECE	BLOCK B
3	ME	BLOCK C

EID	EN	DID	DNAME	ADDRESS
1	A	1	CSE	BLOCK A
2	B	2	ECE	BLOCK B
3	C	3	ME	BLOCK C
4	A	3	ME	BLOCK C
5	E	1	CSE	BLOCK A
6	E	1	CSE	BLOCK A
7	E	1	CSE	BLOCK A
8	E	1	CSE	BLOCK A

Redundancy

EID	EN	DID
1	A	1
2	B	2
3	C	3
4	A	3
5	E	1

Foreign key

DID	DNAME	ADDRESS
1	CSE	BLOCK A
2	ECE	BLOCK B
3	ME	BLOCK C

EID	EN	DID	DNAME	ADDRESS
1	A	1	CSE	BLOCK A
2	B	2	ECE	BLOCK B
3	C	3	ME	BLOCK C
4	A	3	ME	BLOCK C
5	E	1	CSE	BLOCK A
6	m	1	CSE	BLOCK A
7	n	1	CSE	BLOCK A
8	n	1	CSE	BLOCK A
Null	null	4	AE	BLOCK F

INSERTION ANAMOLY

Redundancy

EID	EN	DID
1	A	1
2	B	2
3	C	3
4	A	3
5	E	1

Foreign key

DID	DNAME	ADDRESS
1	CSE	BLOCK A
2	ECE	BLOCK B
3	ME	BLOCK C

DELETION ANAMOLY

EID	EN	DID	DNAME	ADDRESS
1	A	1	CSE	BLOCK A
~~2~~	~~B~~	~~2~~	~~ECE~~	~~BLOCK B~~
3	C	3	ME	BLOCK C
4	A	3	ME	BLOCK C
5	E	1	CSE	BLOCK A
6	m	1	CSE	BLOCK A
7	n	1	CSE	BLOCK A
8	n	1	CSE	BLOCK A
9	o	1	ECE	BLOCK A

Redundancy

EID	EN	DID
1	A	1
2	B	2
3	C	3
4	A	3
5	E	1

Foreign key

DID	DNAME	ADDRESS
1	CSE	BLOCK A
2	ECE	BLOCK B
3	ME	BLOCK C

UPDATION ANAMOLY

EID	EN	DID	DNAME	ADDRESS
1	A	1	CS	BLOCK A
~~2~~	~~B~~	~~2~~	~~ECE~~	~~BLOCK B~~
3	C	3	ME	BLOCK C
4	A	3	ME	BLOCK C
5	E	1	CS	BLOCK A
6	m	1	CS	BLOCK A
7	n	1	CS	BLOCK A
8	n	1	CSE	BLOCK A
9	o	1	ECE	BLOCK A

Redundancy

Normalization

- **Normalization** is the process of organizing the data in the database.
- **Normalization** is used to minimize the redundancy from a relation or set of relations.
- It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.

Is this a good database design?

sid	sname	address	cid	cname	grade
124	Britney	USA	206	Database	A++
204	Victoria	Essex	203	Java	C
124	Britney	USA	201	S/Eng I	A+
206	Emma	London	206	Database	B-
124	Britney	USA	202	Semantics	B+

Insertion Anomaly

sid	sname	address	cid	cname	grade
124	Britney	USA	206	Database	A++
204	Victoria	Essex	203	Java	C
124	Britney	USA	201	S/Eng I	A+
206	Emma	London	206	Database	B-
124	Britney	USA	202	Semantics	B+

- Tried to insert data in a record that makes null value in a prime attribute.

- Let, a new course has been introduced but no student has been registered yet.
- So cid=205, cname=OS. But all the other values will be null.
- Here, sid is a primary key and can't be null.
- This type of anomaly is known as insertion anomaly.

Update Anomaly

sid	sname	address	cid	cname	grade
124	Britney	USA	206	Database	A++
204	Victoria	Essex	203	Java	C
124	Britney	USA	201	S/Eng I	A+
206	Emma	London	206	Database	B-
124	Britney	USA	202	Semantics	B+

- When we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.

- Lets assume, Britney updated her address from USA to London.
- If it is not updated in all 3 places, database will face an inconsistent state.

Deletion Anomaly

SID	SNAME	ADDRESS	CID	CNAME	GRADE
124	Britney	USA	206	Database	A++
204	Victoria	Essex	203	Java	C
124	Britney	USA	201	S/Eng I	A+
206	Emma	London	206	Database	B-
124	Britney	USA	202	Semantics	B+

- We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else. Or deletion make other unsaved data removal.

- Assume, university wants to discontinue Java course with cid=203.
- The whole tuple will be deleted.
- Then we will loose all the data of Victoria.
- This type of anomaly is known as deletion anomaly.

Is this a good design?

sid	sname	address	cid	cname	grade
124	Britney	USA	206	Database	A++
204	Victoria	Essex	203	Java	C
124	Britney	USA	201	S/Eng I	A+
206	Emma	London	206	Database	B-
124	Britney	USA	202	Semantics	B+

- So, the design is good only for read mode.
- In write mode, it will go to inconsistent state.
- We have to decompose the table for removing all the dependency.

Functional Dependency

- Functional dependency (FD) is a set of constraints between two attributes in a relation.
- Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.
- Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y. (or Y is determined by X)
- The left-hand side attributes determine the values of attributes on the right-hand side.

Functional Dependency

Sid → Sname (Sid determines Sname)

Sachin

Sachin

Is these two person same or different?

<u>1st Possibility:</u>

Sid	Sname
1	Sachin
1	Sachin

<u>2nd Possibility:</u>

Sid	Sname
1	Sachin
2	Sachin

X→Y

Here, X is determinant.

Here, Y is dependent.

Informally we can say, if we face any difficulty in dependent side, we have to look into determinant side to solve the confusion.

1st case: redundant data of same person

2nd case: different person

Rule out the functional dependency based on the tables??

EID->ENAME

ENAME->EID

EID	ENAME
1	A
2	B
3	B

Rule out the functional dependency based on the tables??

INCORRECT

EID->ENAME

ENAME->EID

EID	ENAME
1	A
2	B
3	B

Rule out the functional dependency based on the tables??

A->B

B->A

A	B
1	1
1	2
2	2

Rule out the functional dependency based on the tables??

A->B

B->A

A	B
1	1
1	2
2	2

Rule out the functional dependency based on the tables??

A->B

B->C

B->A

C->B

C->A

A->C

A	B	C
1	1	4
1	2	4
2	1	3
2	2	3
2	4	3

Rule out the functional dependency based on the tables??

A->B

B->C

B->A

C->B

C->A

A->C

A	B	C
1	1	4
1	2	4
2	1	3
2	2	3
2	4	3

Rule out the functional dependency based on the tables??

A->B

B->C

B->A

C->B

C->A

A->C

A	B	C
1	1	4
1	2	4
2	1	3
2	2	3
2	4	3

Armstrong's Axioms

Armstrong's Axioms are a set of rules, that when applied repeatedly, generates a closure of functional dependencies.

- **Reflexive rule** − If X is a set of attributes and Y is subset of X, then X holds a value of Y. $Y \subseteq X$ then $X \rightarrow Y$
- **Augmentation rule** − If a → b holds and y is attribute set, then ay → by also holds. That is adding attributes in dependencies, does not change the basic dependencies.
- **Transitivity rule** − Same as transitive rule in algebra, if a → b holds and b → c holds, then a → c also holds. a → b is called as a functionally that determines b.

FD's?

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

Consequences of Armstrong's axioms

- **Union**: If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow Y,Z$
- **Pseudo-transitivity**: If $X \rightarrow Y$ and $W,Y \rightarrow Z$ then $X,W \rightarrow Z$
- **Decomposition**: If $X \rightarrow Y$ and $Z \subseteq Y$ then $X \rightarrow Z$

Closure

- If F is a set of functional dependencies, then the closure of F, denoted as F^+, is the set of all functional dependencies logically implied by F.
- It will help you to find all the candidate key in a relation.

(A minimal superkey is candidate key. A **candidate key** is a set of attributes (or attribute) which uniquely identify the tuples in relation or table .)

<u>Example:</u>

Question: R(ABCD),   FD{A $\rightarrow$B, B $\rightarrow$C,C $\rightarrow$D}

<u>Solution:</u>

A^+=B (As A can directly determine B)

A^+=BC (As B can determine C, A can also determine C, transitive)

A^+=BCD (As C can determine D, A can also determine D, transitive)

A^+=BCDA (A can determine itself, reflexive), this is the final solution

Find Candidate Key

Example:

Question: R(ABCD), FD{A →B, B →C,C →D}

Solution:

A^+=BCDA (A is a candidate key, as it closure holds all the attributes of R).

Now let's find the closure of B,C and D.

B^+=BCD

C^+=CD

D^+=D

So A is the only candidate key.

Let find the closure for $(AB)^+$

$(AB)^+$ = ABCD

AB can't be a candidate key as it should be minimal. AB can be super key.

Closure and Candidate Key

<u>Example:</u>

Question: R(ABCD), FD{A →B, B →C,C →D, D →A}

<u>Solution:</u>

A^+=ABCD

B^+=BCDA

C^+=CDAB

D^+=DABC

Candidate key={A,B,C,D}

Closure and Candidate Key

Example:

Question: R(ABCDE), FD{A →B, B C→D,E→C, D →A}

Solution:

A^+=AB

B^+=B

C^+=C

D^+=ADB

E^+=EC

If you check all the dependents in the question, then you can understand that E can only come as a dependent (in right hand side) if we take different combinations of E to determine the candidate key.

$(AE)^+$=ABECD

$(BE)^+$=BECDA

$(CE)^+$=CE

$(DE)^+$=DEABC

Candidate key={AE,BE,DE}

- **GATE Question: In a schema with attributes A, B, C, D and E following set of functional dependencies are given**
  {A -> B, A -> C, CD -> E, B -> D, E -> A}
  Which of the following functional dependencies is NOT implied by the above set? (GATE IT 2005)
  A. CD -> AC
  B. BD -> CD
  C. BC -> CD
  D. AC -> BC

- **GATE Question: In a schema with attributes A, B, C, D and E following set of functional dependencies are given**
  {A -> B, A -> C, CD -> E, B -> D, E -> A}
  Which of the following functional dependencies is NOT implied by the above set? (GATE IT 2005)
  A. CD -> AC
  B. **BD -> CD**
  C. BC -> CD
  D. AC -> BC

- **GATE Question: Consider a relation scheme R = (A, B, C, D, E, H) on which the following functional dependencies hold: {A–>B, BC–> D, E–>C, D–>A}. What are the candidate keys of R? [GATE 2005]**
  (a) AE, BE
  (b) AE, BE, DE
  (c) AEH, BEH, BCH
  (d) AEH, BEH, DEH

- **GATE Question: Consider a relation scheme R = (A, B, C, D, E, H) on which the following functional dependencies hold: {A–>B, BC–> D, E–>C, D–>A}. What are the candidate keys of R? [GATE 2005]**
  (a) AE, BE
  (b) AE, BE, DE
  (c) AEH, BEH, BCH
  (d) **AEH, BEH, DEH**

Equivalence of Closure

- Two sets of FDs, F and G, are said to be **equivalent** if $F^+=G^+$
- For example:

  $\{(A,B \rightarrow C), (A \rightarrow B)\}$ and

  $\{(A \rightarrow C), (A \rightarrow B)\}$

  are equivalent.

- Q 1: Given a relational schema R(X, Y, Z, W, V) set of functional dependencies P and Q such that:
- P = { X → Y, XY → Z, W → XZ, W → V} and
- Q = { X → YZ, W → XV } using FD sets P and Q which of the following options are correct?

1. P is a subset of Q
2. Q is a subset of P
3. P = Q
4. P ≠ Q

- Q 1: Given a relational schema R(X, Y, Z, W, V) set of functional dependencies P and Q such that:
- P = { X → Y, XY → Z, W → XZ, W → V} and
- Q = { X → YZ, W → XV } using FD sets P and Q which of the following options are correct?

1. P is a subset of Q
2. Q is a subset of P
3. P = Q
4. P ≠ Q

Trivial Functional Dependency

- **Trivial** – If a functional dependency (FD) X → 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 → Y holds, where Y is not a subset of X, then it is called a non-trivial FD.

R={A,B,C,D}

GIVEN:
A->BC
C->A

Find the candidate keys?

R={A,B,C,D}

GIVEN:
A->BC
C->A

For Degree 1 : (Find the closure)

A^+ , B^+ , C^+ , D^+

R={A,B,C,D}

GIVEN:
A->BC
C->A

For Degree 1 : (Find the closure)

A^+ , B^+ , C^+ , D^+

- A^+ = {ABC} **(NOT A CANDIDATE KEY)**

R={A,B,C,D}

GIVEN:
A->BC
C->A

For Degree 1 : (Find the closure)

A^+ , B^+ , C^+ , D^+

- A^+ = {ABC} **(NOT A CANDIDATE KEY)**
- B^+ = {B} **(NOT A CANDIDATE KEY)**

R={A,B,C,D}

GIVEN:
A->BC
C->A

For Degree 1 : (Find the closure)

A^+ , B^+ , C^+ , D^+

- A^+ = {ABC} **(NOT A CANDIDATE KEY)**
- B^+ = {B} **(NOT A CANDIDATE KEY)**
- C^+ = {AC} **(NOT A CANDIDATE KEY)**

R={A,B,C,D}

GIVEN:
A->BC
C->A

For Degree 1 : (Find the closure)

A^+ , B^+ , C^+ , D^+

- $A^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $B^+ = \{B\}$ **(NOT A CANDIDATE KEY)**
- $C^+ = \{AC\}$ **(NOT A CANDIDATE KEY)**
- $D^+ = \{D\}$ **(NOT A CANDIDATE KEY)**

R={A,B,C,D}

GIVEN:
A->BC
C->A

For Degree 2 : (Find the closure)

AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+

- AB^+ = {ABC} **(NOT A CANDIDATE KEY)**

R={A,B,C,D}

GIVEN:
A->BC
C->A

For Degree 2 : (Find the closure)

AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+

- AB^+ = {ABC} **(NOT A CANDIDATE KEY)**
- AC^+ = {ABC} **(NOT A CANDIDATE KEY)**

R={A,B,C,D}

GIVEN:
A->BC
C->A

For Degree 2 : (Find the closure)

AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+

- AB^+ = {ABC} **(NOT A CANDIDATE KEY)**
- AC^+ = {ABC} **(NOT A CANDIDATE KEY)**
- AD^+ = {ABCD} **(A CANDIDATE KEY)**

R={A,B,C,D}

GIVEN:
A->BC
C->A

For Degree 2 : (Find the closure)

AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+

- AB^+ = {ABC} **(NOT A CANDIDATE KEY)**
- AC^+ = {ABC} **(NOT A CANDIDATE KEY)**
- AD^+ = {ABCD} **(A CANDIDATE KEY)**
- BC^+ = {BCA} **(NOT A CANDIDATE KEY)**

R={A,B,C,D}

GIVEN:
A->BC
C->A

For Degree 2 : (Find the closure)

AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+

- AB^+ = {ABC} **(NOT A CANDIDATE KEY)**
- AC^+ = {ABC} **(NOT A CANDIDATE KEY)**
- AD^+ = {ABCD} **(A CANDIDATE KEY)**
- BC^+ = {BCA} **(NOT A CANDIDATE KEY)**
- BD^+ = {BD} **(NOT A CANDIDATE KEY)**

R={A,B,C,D}

GIVEN:
A->BC
C->A

For Degree 2 : (Find the closure)

AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+

- AB^+ = {ABC} **(NOT A CANDIDATE KEY)**
- AC^+ = {ABC} **(NOT A CANDIDATE KEY)**
- AD^+ = {ABCD} **(A CANDIDATE KEY)**
- BC^+ = {BCA} **(NOT A CANDIDATE KEY)**
- BD^+ = {BD} **(NOT A CANDIDATE KEY)**
- CD^+ = {CDAB} **(A CANDIDATE KEY)**

R={A,B,C,D}

GIVEN:
A->BC
C->A

CANDIDATE KEY: {AD,CD}

PRIME ATTRIBUTE={A,C,D}
NON- PRIME ATTRIBUTE={B}

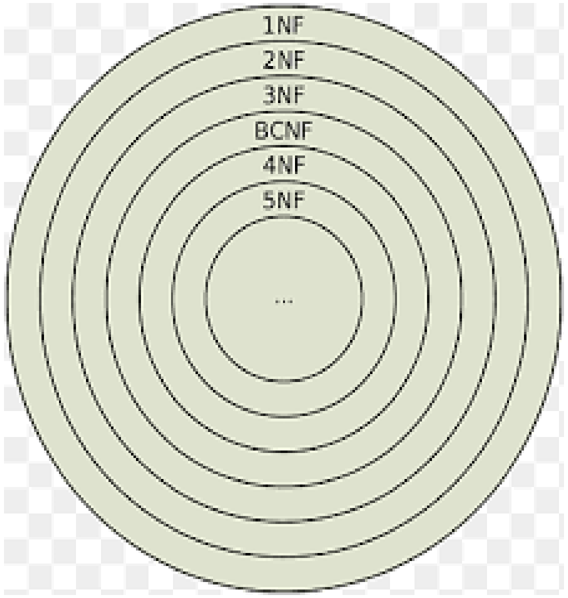
For Degree 2 : (Find the closure)

AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+

- AB^+ = {ABC} **(NOT A CANDIDATE KEY)**
- AC^+ = {ABC} **(NOT A CANDIDATE KEY)**
- AD^+ = {ABCD} **(A CANDIDATE KEY)**
- BC^+ = {BCA} **(NOT A CANDIDATE KEY)**
- BD^+ = {BD} **(NOT A CANDIDATE KEY)**
- CD^+ = {CDAB} **(A CANDIDATE KEY)**

Normalization

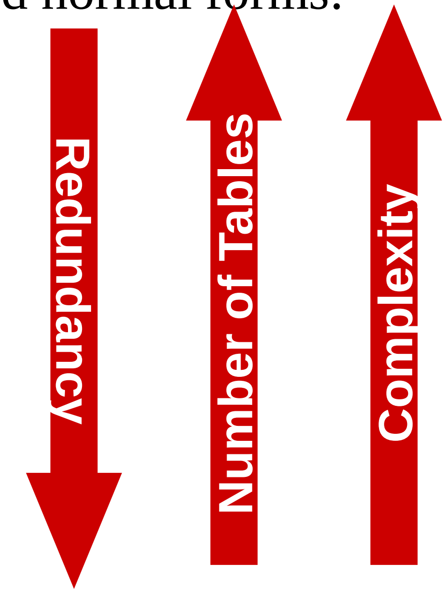
- Normalization is a process for assigning attributes to entities. It reduces data redundancies and helps eliminate the data anomalies.
- Normalization works through a series of stages called normal forms:
  - First normal form (1NF)
  - Second normal form (2NF)
  - Third normal form (3NF)
  - Boyce–Codd normal form (or BCNF or 3.5NF)
  - Fourth normal form (4NF)
  - Fifth normal form
  - ……



- The highest level of normalization is not always desirable.
- Which type of normalization is needed, basically depends on the given relations and specific applications

Normalization

- Normalization is a process for assigning attributes to entities. It reduces data redundancies and helps eliminate the data anomalies.
- Normalization works through a series of stages called normal forms:
  - First normal form (1NF)
  - Second normal form (2NF)
  - Third normal form (3NF)
  - Boyce–Codd normal form (or BCNF or 3.5NF)
  - Fourth normal form (4NF)
  - Fifth normal form
  - ……



- The highest level of normalization is not always desirable.
- Which type of normalization is needed, basically depends on the given relations and specific applications

Objective of Normalization

- Normalization presents a set of rules that tables and databases must follow to be well structured.
- It can solve different anomaly problems in write mode
- It reduces data redundancy

First Normal From

- A table is in the first normal form iff
  - The domain of each attribute contains only ***atomic values***, and
  - The value of each attribute contains only a ***single value*** from that domain.

In layman's terms atomic value means every column of your table should only contain *<u>single values</u>*

Example

- For a library

Patron ID	Borrowed books
C45	B33, B44, B55
C12	B56

1-NF Solution

Patron ID	Borrowed book
C45	B33
C45	B44
C45	B55
C12	B56

Example

- For an airline

Flight	Weekdays
UA59	Mo We Fr
UA73	Mo Tu We Th Fr

1NF Solution

Flight	Weekday
UA59	Mo
UA59	We
UA59	Fr
UA73	Mo
UA73	Tu
...	...

Second Normal Form

- A table is in 2NF iff
  - It is in 1NF and
  - no non-prime attribute is dependent on any proper subset of any candidate key of the table (i.e. no partial dependency exists)
- A ***non-prime attribute*** of a table is an attribute that is not a part of any candidate key of the table
- A ***candidate key*** is a minimal superkey

Example

- Library allows patrons to request books that are currently out

BookNo	Patron	PhoneNo
B3	J. Fisher	555-1234
B2	J. Fisher	555-1234
B2	M. Amer	555-4321

Example

- Candidate key is {BookNo, Patron}
- We have
  - Patron → PhoneNo (here phone no, a non-prime attribute is dependent on subset of candidate key, i.e. dependent on Parton, so partial dependency exists)
- Table is not 2NF
  - Potential for
    - Insertion anomalies
    - Update anomalies
    - Deletion anomalies

2NF Solution

- Put telephone number in separate Patron table

BookNo	Patron
B3	J. Fisher
B2	J. Fisher
B2	M. Amer

Patron	PhoneNo
J. Fisher	555-1234
M. Amer	555-4321

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)
GIVEN :

AB->C
C->D
B->E

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)
GIVEN :

AB->C
C->D
B->E

Solution:

Step 1: Find out the candidate key

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)
GIVEN :

AB->C
C->D
B->E

Solution:

Step 1: Find out the candidate key

AB is the only candidate key because $(AB)^+$ ={ABCDE}

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)
GIVEN :

AB->C
C->D
B->E

Solution:

Step 1: Find out the candidate key

AB is the only candidate key because $(AB)^+$ ={ABCDE}

Step 2: Find out the partial dependency if exist

AB->C (no partial dependency)

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)
GIVEN :

AB->C
C->D
B->E

Solution:

Step 1: Find out the candidate key

AB is the only candidate key because $(AB)^+ = \{ABCDE\}$

Step 2: Find out the partial dependency if exist

AB->C (no partial dependency)
C->D (no partial dependency)

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)
GIVEN :

AB->C
C->D
B->E

Solution:

Step 1: Find out the candidate key

AB is the only candidate key because $(AB)^+ = \{ABCDE\}$

Step 2: Find out the partial dependency if exist

AB->C (no partial dependency)
C->D (no partial dependency)
B-> E (partial dependency)

Partial dependency exist for the given relation hence it is in 1 NF but not in 2NF

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)
GIVEN :

AB->C
BC->D

Question: Find out whether the given relation is in 2nf or not?

R(ABCD)
GIVEN :

AB->C
BC->D

Solution:

Step 1: Find out the candidate key

AB is the only candidate key because $(AB)^+ = \{ABCD\}$

Step 2: Find out the partial dependency if exist

AB->C (no partial dependency)
BC->D (no partial dependency) (note: BC is not a proper subset of candidate key)

No Partial dependency exist for the given relation hence it is in 2 NF

Thank you..