Keys

Uniquely identify the tuples. It is an attribute or set of attributes

Super key: It is a superset which is used to uniquely identify a row in a table. It can be a combination of all possible attributes. name✗ like ID & Aadhar no, ID & email, name & aadhar & ph.

Candidate key: Minimal super keys.
{ID}, {aadhar}, {email}, {name, phone}

Primary key: Should be unique & not null. {ID}. It is chosen from set of candidate key

Alternate key: The candidate key other than primary key. All the keys which are not primary keys. {aadhar}, {email}, {name, phone}

Unique key: Should be unique and contain null values also. {email}, {name, phone}

Composite key: Combination of more than one attribute {name, phone}

Foreign key: It is used to link two tables together. An attribute in one table refers to the primary key in another table. It is used to ensure referential integrity of data.

Employee Deptid (FK) → Deptid (primary key) Department

(Referencing) Child table

(Referenced) Parent table

Functional dependency

Relation between attributes

FD: $x \rightarrow y$ (dependent)
(determinant)

x	y
1	1
2	1
3	2
4	3
2	3

$t_1 \rightarrow$ $t_2 \rightarrow$

x determines y (or) y is functionally dependent on x

if $t_1.x = t_2.x \Rightarrow 2 = 2$

then $t_1.y = t_2.y$? 3 ✗

Each attribute should define uniquely Student

R.NO	Name	marks	Dept	Course
1	a	78	CS	C1
2	b	60	EE	C1
3	a	98	CS	C2
4	b	60	EE	C3
5	c	80	IT	C3
6	d	80	EC	C2

R.NO → Name ✓ FD.

Name → R.NO ✗

R.NO → marks ✗

Dept → Course ✗

Course → Dept ✗

marks → Dept ✗

{R.NO, name} → marks ✓

name → marks ✓

{name, marks} → Dept ✓

{name, marks} → {Dept, course} ✗

{R.No} → {name, marks} ✓

Types of FD

1. Fully functional dependency

X → Y is FD, if we remove any attribute of X violates the FD rule

(A, B) → C FD

remove A

if we remove B then A → C ✗ FD B → C ✗ FD

C is fully dependent on both A & B.

2. Partial functional dependency

X → Y is FD, if we remove any attribute of X doesn't violate the FD rule

(A, B) → C FD

if we remove A

Eg: if we remove B then A → C ✓ FD B → C ✓ FD

C is partially dependent on A or B.

3. Trivial functional dependency

A → B is FD

if B is subset of A

4. Non-Trivial functional dependency

A → B is FD

if B is not subset of A

5. Multivalued functional dependency

A → BC is FD

then B & C should not be dependent

B → C } Not FD

C → B

6. Transitivity functional dependency

A → B is FD B → C is FD then A → C is FD

$X \rightarrow Y$ is FD if it satisfies a constraint

If $t_1.x = t_2.x$ then $t_1.y = t_2.y$

if two x values are equal then y values should be equal

Sid	Name	Address	Course
1	A	Hyd	Python
1	A	Hyd	Java
1	A	blr	Python
2	B	delhi	C
3	B	chennai	Java
4	C	Hyd	Python
5	D		

FD's

Sid (x) → name (y) ✓

Sid → address ✓

Sid → Course ✗

Course → name ✗

name → sid ✗

{sid, name} → Course ✗

{sid, Course} → name ✓

Fully (Name, Course) → sid ✓

name → sid ✗

Course → sid ✗

$t_1.x = t_2.x$

$t_1.y = t_2.y$

Partial (sid, course) → name ✓

sid → name ✓

course → name ✗

Sid	Name	Age	College	Place
1	A	25	VITS	nlg
2	B	24	MVSR	hyd
3	C	25	CVR	wrl
4	B	23	GNIT	hyd

Trivial (sid, Name) → sid

(subset)

Non Trivial (sid, name) → age

Multivalued sid → (name, age)

name → age ✗ FD

age → name ✗ FD

Transitivity sid → college, college → place

sid → place

Properties of functional dependency / Armstrong

1. Reflexivity:
   if A - set of attributes
   B - subset of A
   then $A \rightarrow B$ is FD
   Dependent is subset of determinant

2. Augmentation:
   if $A \rightarrow B$ is FD
   if C is attribute (or) set of attributes added to both determinant & dependent
   then $AC \rightarrow BC$ is also FD

3. Transitivity:
   if $A \rightarrow B$ is FD &
   if $B \rightarrow C$ is FD
   then $A \rightarrow C$ is also FD.

Inference rules

These are derived from armstrong axioms

1. Union: if $A \rightarrow B$ is FD
   $A \rightarrow C$ is FD
   then $A \rightarrow BC$ is also FD

2. Composition: if $A \rightarrow B$ is FD
   $C \rightarrow D$ is FD
   then $AC \rightarrow BD$ is also FD

3. Decomposition: if $A \rightarrow BC$ is FD
   then $A \rightarrow B$ & $A \rightarrow C$ FD.

4. Pseudo transitivity:
   if $A \rightarrow B$ is FD, $BC \rightarrow D$ is FD then $AC \rightarrow D$ is

Attribute closure

closure → is a set of attributes that are functionally determined by attribute set.
We can closure on single attribute or attribute set
→ It is represented by +
closure of X → X^+ = { }

R = (A, B, C, D, E)

FD
A → B
B → C
C → D
D → A

A^+ = {A, B, C, D}
D^+ = {D, A, B, C}
E^+ = {E}
$(B, C)^+$ = {B, C, D, A}

R = (A, B, C, D, E, F, G)

FD
A → BC ⇒ A → B, A → C
BC → DE ⇒ BC → D, BC → E
D → F
CF → G

A^+ = {A, B, C, D, E, F, G}
D^+ = {D, F}
$(B, C, D)^+$ = {B, C, D, E, F, G}

Finding Superkeys & Candidate keys

Super key ⇒ If closure of attribute / set of attribute determines all the attributes of relation

R = (A, B, C)

$(AB)^+$ = {A, B, C}
$(ABC)^+$ = {A, B, C}
$(B)^+$ = {A, B, C}

ABC → SK
AB → SK
BC
AC
A
B → SK

closure of these should contain all attributes → CK (minimal)

To find super keys and candidate keys in a relation, first we need to find the closure of all the combinations in a relation.
After finding the closure, the attribute or set of attributes that determines all the attributes (single ones) can be considered as super keys.

$R = (A, B, C, D, E)$

FD: LHS → RHS

$A \to B$ $\quad (ABCDE)^+ = \{A, B, C, D, E\}$ → SK (ABCDE)

$C \to D$ $\quad (ACDE)^+ = \{A, C, D, E, B\}$ → Sk (ACDE)

$D \to E$ $\quad (ACE)^+ = \{A, C, E, B, D\}$ → Sk (ACE)

$D \to C$ $\quad A \overset{\uparrow}{C} \times$

$AC \to AD$ ∴ ACE $\quad (AC)^+ = \{A, C, B, D, E\}$ → SK (AC) ↓ minimal ∴ CK

Combination $\quad (CE)^+ = \{C, E, D\}$ — × SK

$(AE)^+ = \{A, E, B,\}$ — × Sk

$(A)^+ = \{A, B\} \times$ - Sk

$(C)^+ = \{C, D\} \times$ Sk

$(E)^+ = \{E\} \times$ Sk

Prime attributes → which are available in the CK.

CK → AC / Prime attributes - A, C

To check for more candidate keys

* Check whether any one of prime attributes are on RHS of any FD.

If Not → There is only one candidate key

If Yes → Replace prime attributes in CK with LHS of FD

Repeat finding Superkey & CK.

$R = (X, Y, Z, W)$

FD
LHS → RHS
$X \to Y$
$Y \to Z$
$Z \to X$.

$(XYZW)^+ = \{X, Y, Z, W\}$ - SK
$(XZW)^+ = \{X, Z, W, Y\}$ - SK
$(ZW)^+ = \{Z, W, X, Y\}$ - SK
$(Z)^+ = \{Z, X, Y\}$ - X SK
$(W)^+ = \{W\}$ - X SK

Superkeys
XYZW
XZW
ZW → Candidate key

Prime attributes
↓
Z, W Yes in RHS
ZW
↓
YW

(YW) → CK.
Y, W Yes in RHS

$(YW)^+ = \{Y, W, Z, X\}$ - SK.
$(Y)^+ = \{Y, Z, X\}$ - X
$(W)^+ = \{W\}$ - X

YW
↓
(XW)

$(XW)^+ = \{X, W, Y, Z\}$ - SK
$(X)^+ = \{X, Y, Z\}$ - X
$(W)^+ = \{W\}$ - X

XW → CK.

X, W → P.A
Yes in RHS

XW
↓
(ZW)

$R = (A, B, C, D, E)$

FD

$AB \to C$

$C \to D$

$B \to E$

$(ABCDE)^+ = \{A, B, C, D, E\}$ – Sk

$(ABDE)^+ = \{A, B, D, E, C\} \to$ Sk

$(ABD)^+ = \{A, B, D, C, E\} \to$ Sk

$(AB)^+ = \{A, B, C, D, E\} \to$ Sk

$(BD)^+ = \{B, D, E, \} - X$

$(AD)^+ = \{A, D\} - X$

$(A)^+ = \{A\} \to X$

$(B)^+ = \{B, E\} \to X$

$(D)^+ = \{D\} - X$

Super keys : ABCDE, ABDE, ABD, AB – minimal

Candidate : AB

Prime Attributes : A, B Not in RHS of any FD

$\therefore$ AB is only ck.

$R = (A, B, C, D, E, H)$

FD

$A \to B$

$BC \to D$

$E \to C$

$D \to A$

reduce

$(ABCDEH)^+ = \{A, B, C, D, E, H\} \to$ Sk

$(ACDEH)^+ = \{A, C, D, E, H, B\} \to$ Sk

$(ADEH)^+ = \{A, D, E, H, B, C\} \to$ Sk

$(DEH)^+ = \{D, E, H, A, B, C\} \to$ Sk

$(DE)^+ = \{D, E, A, B, C\} \to X$

$(EH)^+ = \{E, H, C\} \to X$

$(DH)^+ = \{D, H, A, B\} \to X$

$(D)^+ = \{D, A, B\} \to X$

$(E)^+ = \{E, C\} \to X$

$(H)^+ = \{H\} X$

SK: ABCDEH, ACDEH, ADEH, DEH, BCEH, BEH, AEH

CK: DEH, BEH, AEH

PA: D, E, H — Yes in RHS

DEH
↓
BCEH

PA: B, E, H — Yes in RHS

BEH
↓
AEH.

$(BCEH)^+ = \{B, C, E, H, D, A\}$ – SK

$(BEH)^+ = \{B, E, H, C, D, A\}$ – SK

$(BE)^+ = \{B, E, C, D, A\}$ – X

$(BH)^+ = \{B, H\}$ – X

$(EH)^+ = \{E, H, C\}$ – X

$(B)^+ = \{B\}$ – X

$(E)^+ = \{E, C\}$ – X

$(H)^+ = \{H\}$ – X

$(AEH)^+ = \{A, E, H, B, C, D\}$ – SK

$(AE)^+ = \{A, E, B, C, D\}$ – X

$(EH)^+ = \{E, H, C\}$ – X

$(AH)^+ = \{A, H, B\}$ – X

$(A)^+ = \{A, B\}$ – X

$(E)^+ = \{E, C\}$ – X

$(H)^+ = \{H\}$ – X

PA: A, E, H

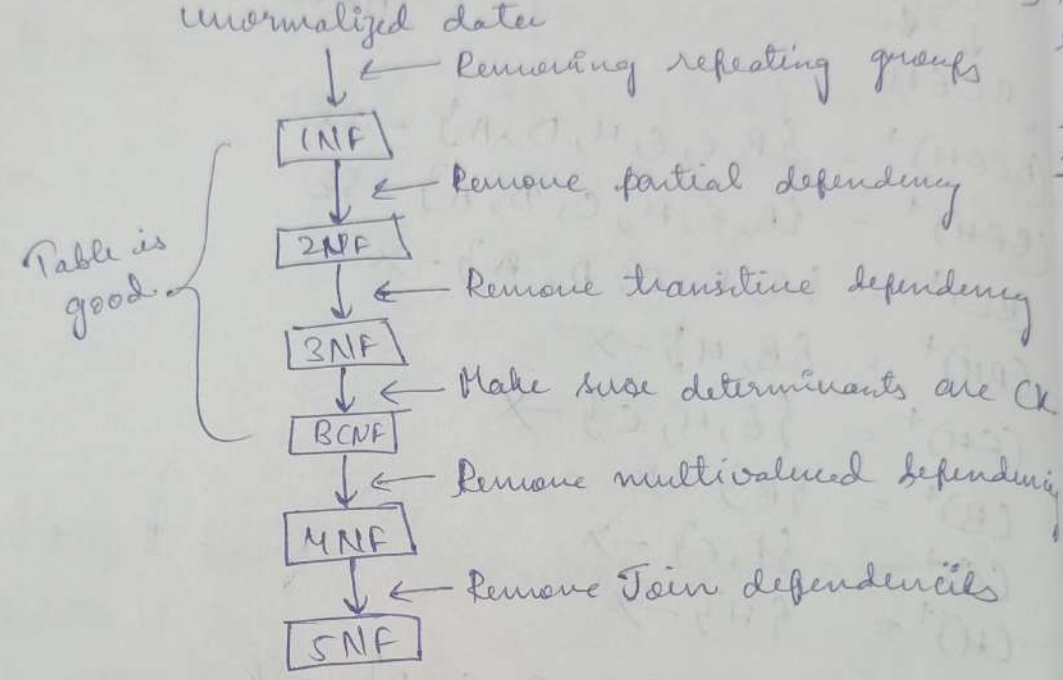
AEH
↓ Yes in RHS
DEH.

Normalization

Optimizing tables → Decomposed into sub tables

why

→ Avoid redundancy & anomalies (duplicate values)



1NF:

I/P: unnormalized data

O/P: Relation with 1NF

* All the attribute values should be atomic (Single)
* Attributes should not have multi-values, composite
* All the Attribute should be unique

Sid	name	Course
1	A	C, C++, Java
2	B	C++, Python
3	C	Java, Python

→

Sid	name	Course
1	A	C
1	A	C++
1	A	Java
2	B	C++
2	B	Python
3	C	Java
3	C	Python

2NF:

I/P: Relation with 1NF
O/P: Relation with 2NF

* Relation should be in 1NF
* Relation should not have partial FD.

Sid	Cid	C_Fee
1	Java	10000
2	Python	15000
1	C++	20000
3	C	10000
3	Java	10000
2	JS	20000

Sid → Cid ✗
Sid → C-Fee ✗
Cid → C-Fee ✓
Sid, Cid → C-Fee ✓
Sid, Cfee → Cid ✗
Cfee, Cid → Sid ✗

Sid, Cid → C_Fee

Sid → C_Fee ✗
Cid → C_Fee ✓ PFD

Sid	Cid
1	Java
2	Python
1	C++
3	C
3	Java
2	JS

Cid	Cfee
Java	10000
Python	15000
C++	20000
C	10000
JS	20000

3NF

I/P : Relation with 2NF
O/P : Relation with 3NF

* Relation should be in 2NF
* No transitive dependencies for NPA
  ↓
  Attributes not in Ck

* If $X \rightarrow Y$ (Y - P.A.)
  X should be PK
  Y should be part of Ck.

Sid	Course	Fee
1	Java	20000
2	C	10000
3	C++	15000
4	Python	25000
5	Java	20000
6	Python	25000
7	Java	20000
8	C++	15000

Sid → Course, Fee ✓
Sid → Course ✓
Sid → Fee ✓
Sid, Course → Fee ✓
Sid, Fee → Course ✓
Course → Fee ✓
↳ NPA

Sid
↓
PK
↓
Ck
↓
PA

X → Y
X → SL
Y → X part of Ck

Sid → Course
Course → Fee
} Transitive FD.

Sid	Cou
1	
2	
3	
4	
5	
6	
7	
8	

Cou	Fee
Java	
C	
C++	
Python	

BCNF

3.5 NF
* Relati
* for a

Sid	Cou
1	Ja
1	Py
2	J
2	P

(S, C
(S
(
(
P.A →
P.

BCNF (Boyce Codd Normal Form)

3.5 NF

* Relation should be in 3NF
* for all FD's $X \to Y$
  X should be super key / CK.

S	C	T
Sid	Course	Tutor
1	Java	A
1	Python	B
1	Java	A
2	Python	C
2		

Tutor → Course ✓
Sid → Course ✗
Sid → Tutor ✗
Course → Tutor ✗
Sid, Course → Tutor ✗
Sid, ~~Tutor~~ → Course ✓
Course, Tutor → Sid ✗

T → C
S, C → T
S, T → C

$(S, C, T)^+ = \{S, C, T\}$ — Sk
$(ST)^+ = \{S, T, C\}$ → Sk
$(S)^+ = \{S\}$ ✗ Sk
$(T)^+ = \{T, C\}$ ✗ Sk

Sk : SCT, ST
CK : ST, SC

No NPA in CK's
Relation is in 3NF

P.A → T
↓
S, C
$(SC)^+ = \{S, C, T\}$ ✓ SK
$(S)^+ = \{S\}$ ✗
$(C)^+ = \{C\}$ ✗

P.A → S, C
↓
S, T

T → C
↓
Not a CK
} Relation not in BCNF

S, C → T
↓
$\S$ a CK
✓ Rela

S, T → C
↓
$\S$ a CK

Sid	Tutor
1	A
1	B
2	A
2	C

Course	Tutor
Java	A
Python	B
Python	C

Data Integrity rules.

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating and other processes have to be performed in a such a way that data integrity is not affected. Maintain data accuracy, consistency & reliability.

Types:
Domain constraint, Entity integrity constraint, Referential integrity constraint, key constraint

Domain: Valid set of values for an attribute. The datatype of domain includes string, character, integer, time, date.

id	Name	Age
1	Arya	-22 ✗
2	25	Bob ✗
null ✗	Ram	23

Entity Integrity: PK can't be null. Table can contain null other than PK.

Referential Integrity: It is a relation or link b/n two tables.

id	Name	FK DNo
1	A	33 ✗
2	B	11 ✓
3	C	22 ✓

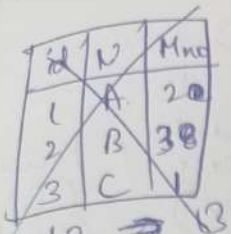
PK

DNo	Dname
11	XYZ
22	PQR

Key: Identify an entity within entity set uniquely. ~~Primary~~ Candidate key should not be repeated. It should be unique. (Email, Phone no)

id	Name	Age
1	A	25
1 ✗	B	30

Codd Rules



0 - 12 ⇒ 13 rules

0: Foundation rule - manage db capabilities (relational)
1: Information rule - tables
2: Guaranteed access - access data using tablename or PK or specifically
3: Systematic treatment of null values - dk the actual value
4: Rule of active and online relational catalog - Entire metadata is stored on 'Data Dictionary'. Data discription
5: Comprehensive Data Sub-language → create tables, insert/reference } support, update, delete, integrity constraints
6: Updating views - Modifications done on orginal table should be reflected in views.
7: Set level insertion, update & deletion
↳ Table level modification
8: Physical data independence - Changes done in physical storage will not affect the application programs
9: logical data independence - changes done in logical part (inserting, updating) will not affect application programs
10: Integrity independence - Store integrity constraints in data dictionary
11: Distribution independence - Databases can support centralised or distributed environment.
12: Non Subversion rule - Low level languages (single record) should follow integrity rules.