Functional Dependencies

Data Anomalies

Name	NRIC	Phone Number	Home Address
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

List of anomalies:

- Redundancy → Alice address is duplicated
- Update anomalies → Accidentally update one of Alice's addresses, leaving other unchanged → corrupt table
- Deletion anomalies → Can't remove PhoneNumber → Primary key attributes cannot be NULL
- Insertion anomalies → Cannot insert record w/o PhoneNumber

How to resolve? NORMALIZATION (decomposition)

Normalization

Name	NRIC	Home Address
Alice	1234	Jurong East
Bob	5678	Pasir Ris

NRIC	Phone Number
1234	67899876
1234	83848384
5678	98765432

ok to have NRIC duplicates as it is part of the key. (still preferred to be unique)

List of anomalies (solved):

- Redundancy → Alice address no longer duplicated
- Update anomalies → Alice address can only be updated in one table
- Deletion anomalies → If PhoneNumber remove, Bob's record on right table deleted entirely.
- Insertion anomalies → If no PhoneNumber, record only appears on left table.

Functional Dependencies : Intuition

Name	NRIC	Phone Number	Home Address
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

Questions to ponder:

- Why was this table bad? A lot of anomalies
- Why does it have a lot of anomalies? Contains bad combination of attributes
- How do we spot bad combinations of attributes? Check correlations among attributes
- What kind of correlations? Functional Dependencies (FD)

Example:

Given NRIC, can we always tell the name of the person? Yes → need help from ICA

Given Name, can we always uniquely identify their NRIC? No → Different people can have same name, but different NRIC

∴ NRIC determines Name, but not the other way round. FD: NRIC → Name

Meaning: There do not exist two persons that have same NRIC but different names.

→ General Definition: There do not exist 2 records that have same A value but different B values in FD, A → B.

Where do FDs come from?

→ FDs either come from common sense or application requirements

Example of application requirements:

→ Purchase (CID, PID, SID, Price, Date)

→ Requirement: No shop to sell the same product to the same customer on the same date at two different prices

CID	PID	SID	Date	Price
C1	P1	S1	D1	99
C1	P1	S1	D1	33

CID	PID	SID	Date	Price
C1	P1	S1	D1	99
C2	P1	S1	D1	33

FD Implied: CID, PID, SID, Date → Price

Armstrong's Axioms

Axiom of Reflexivity	Axiom of Augmentation	Axiom of Transitivity
• A set of attributes → A subset of the attributes	• Given A → B • We always have AC → BC for any C	• Given A → B and B → C • We always have A → C
NRIC, Name → NRIC	Given NRIC → Name	If NRIC → Addr, and Addr → Postal
StudentID, Name, Age → Name, Age	Then NRIC, Age → Name, Age	Then NRIC → Postal
ABCD → ABC		
ABCD → BCD		
ABCD → AD		

Example:

Given A → C, AC → D, AD → B. Can you infer that A → B?

Given A → C, we have A → AC (Augmentation) AA → A (Reflexivity)

Given A → AC and AC → D, we have A → D (Transivity)

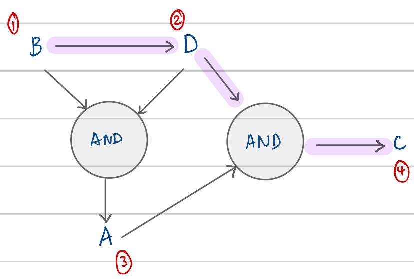
Given A → D, we have A → AD (Augmentation)

Given A → AD and AD → B, we have A → B (Transivity)

Reasoning with FDs: An Intuitive Solution

Example:

Four attributes A, B, C, D
Given: B → D, DB → A, AD → C
Prove B → C?



1. First, we reached B
   - Reachable set = { B }
2. Second, reach whatever B can reach
   - Reachable set = { B, D }, since B → D
3. Third, use all reachable elements to reach more
   - Reachable set = { B, D, A }, since DB → A
4. Repeat 3, until no more reaching is possible
   - Reachable set = { B, D, A, C }, since AD → C; done

Closure

→ Let $S = \{A_1, A_2 \dots A_n\}$ be a set S of attributes

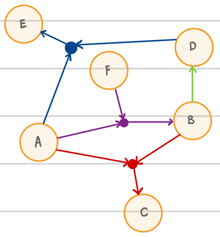
→ The closure of S is the set of attributes that are reachable from $A_1, A_2, \dots A_n$

→ Notation : $\{A_1, A_2, \dots, A_n\}^+$

→ Example :

- Given $A \to B$, $B \to C$, $C \to D$, $D \to E$
- $\{A\}^+ = \{A, B, C, D, E\}$
- $\{B\}^+ = \{B, C, D, E\}$
- $\{D\}^+ = \{D, E\}$
- $\{E\}^+ = \{E\}$

Closure & FD



$AB \to C$, $AD \to E$

$B \to D$, $AF \to B$

Closures :
→ $\{B, C\}^+ = \{B, C, D\}$
→ $\{A, B\}^+ = \{A, B, C, D, E\}$
→ $\{A, F\}^+ = \{A, F, B, C, D, E\}$

- To prove that $X \to Y$ holds, we only need to show that $\{X\}^+$ contains Y
- $AB \to C$, $AD \to E$, $B \to D$, $AF \to B$
- Prove that $AF \to D$ holds
- $\{AF\}^+ = \{AFBCDE\}$, which contains D
- Therefore, $AF \to D$ holds

- To prove that $X \to Y$ does not holds, we only need to show that $\{X\}^+$ does not contains Y
- $AB \to C$, $AD \to E$, $B \to D$, $AF \to B$
- Prove that $AD \to F$ does not hold
- $\{AD\}^+ = \{ADE\}$, which does not contain F
- Therefore, $AD \to F$ does not hold

Superkeys of a table

Name	NRIC	Postal	Address
Alice	1234	939450	Jurong East
Bob	5678	234122	Pasir Ris
Cathy	3576	420923	Yishun

→ Definition of superkey : A set of attributes in a table that decides all other attributes

→ Example :

- $\{NRIC\}$ is a superkey
- Since $NRIC \to Name, Postal, Address$
- $\{NRIC, Name\}$ is a superkey
- Since $\{NRIC, Name\} \to Postal, Address$

Keys of a table

Name	NRIC	Postal	Address
Alice	1234	939450	Jurong East
Bob	5678	234122	Pasir Ris
Cathy	3576	420923	Yishun

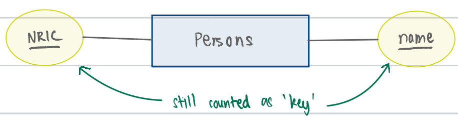
→ key : a minimal superkey

→ Example :

- $\{NRIC\}$ is a superkey
- Since $NRIC \to Name, Postal, Address$
- $\{NRIC, Name\}$ is a superkey
- Since $\{NRIC, Name\} \to Postal, Address$
- NRIC is a key (& superkey), $\{NRIC, Name\}$ is not a key (but is a superkey)

Note : Not to be confused with keys of entity sets because keys in entity sets need not be minimal.

Notion of keys in table ≠ keys in entity set
(need to be minimal) (need not be minimal)



→ Example :

- R (A , B , C , D)
- Given A → BCD , BC → A
- A is a key and a super key
- BC is also a key and a superkey ⇨ due to BC → A ⇨ minimal ; as removing B or C from BC will not allow B or C to reach A
- AB is not a key ; but is a super key

Candidate Keys

Name	NRIC	StudentID	Postal	Address
Alice	1234	1	939450	Jurong East
Bob	5678	2	234122	Pasir Ris
Cathy	3576	3	420923	Yishun

→ A table may have multiple keys

→ In that case, each key is referred to as a candidate key

→ Example :

- { NRIC } is a key
- Since NRIC → Name , Postal , Address , Student ID
- { Student ID } is a key
- Since Student ID → Name , NRIC , Postal , Address
- Both { NRIC } & { Student ID } are candidate keys

Primary & Secondary Keys

Name	NRIC	StudentID	Postal	Address
Alice	1234	1	939450	Jurong East
Bob	5678	2	234122	Pasir Ris
Cathy	3576	3	420923	Yishun

→ A table may have multiple keys

→ Choose 1 as primary key

→ Others referred to as secondary key

→ Example :

- { NRIC } is a key
- { Student ID } is a key
- If we choose { NRIC } as the primary key
- Then { Student ID } is the secondary key

Finding the keys : Algorithm

→ Check all possible combinations of attributes in the table
- Example : A, B, C, AB, BC, AC, ABC

→ For each combination, compute its closure
- Example : $\{A\}^+ = \dots$, $\{B\}^+ = \dots$, $\{C\}^+ = \dots$

→ If a closure contains all attributes, then the combination might be a key (or super key)
- Example : $\{A\}^+ = \{ABC\}$

→ Make sure that you select only keys
- Example : $\{A\}^+ = \{ABC\}$, $\{AB\}^+ = \{ABC\}$ → don't select AB (not minimal ; not key but is superkey)

Example :

→ A table R(A, B, C, D)

→ AB → C , AD → B, B → D

→ First enumerate all attribute combinations :
- $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$
- $\{AB\}$, $\{AC\}$, $\{AD\}$, $\{BC\}$, $\{BD\}$, $\{CD\}$
- $\{ABC\}$, $\{ABD\}$, $\{ACD\}$, $\{BCD\}$
- $\{ABCD\}$

→ Second, compute the closures :
- $\{A\}^+ = \{A\}$, $\{B\}^+ = \{BD\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$
- $\{AB\}^+ = \{ABCD\}$, $\{AC\}^+ = \{AC\}$, $\{AD\}^+ = \{ABCD\}$
- $\{BC\}^+ = \{BCD\}$, $\{BD\}^+ = \{BD\}$, $\{CD\}^+ = \{CD\}$
- $\{ABC\}^+ = \{ABD\}^+ = \{ACD\}^+ = \{ABCD\}$
- $\{BCD\}^+ = \{BCD\}$
- $\{ABCD\}^+ = \{ABCD\}$ ← superkeys (but not keys)

→ Finally output the keys → AB , AD

Tips :

→ Always check small combinations first
- Once key found in smaller combinations, no need to check others
- Others will be superkey but not key

→ If an attribute X is not determined by other attributes and yet it is still in the relation, then it must determine these attributes ; hence, appearing in every key.
- R (A , B , C , D) ; AB → C, AD → B , B → D
- A does not appear on RHS of any FD → no attribute combination that can reach A
- So A must be in every key
- In this case , keys are AB & AD