7.Normalisation

7.1 Redundancy

Redundancy is a waste of disk space, which means that use a lot of space to store the duplicate data.

- Bad design
  - Redundancy
  - Update / insertion / deletion anomalies
- Good design
  - Minimal redundancy
  - No update / insertion / deletion anomalies

7.2 Normalization

The solution to bad design, can normalization the design to avoid the mentioned anomalies.

7.2.1 Definition

Process that defines what is acceptable as good relational design. Can resolving issues surrounding changes to database.

7.2.2 Functional Dependencies (FDs)

A tool to capture semantic relationships between attributes

Detect and eliminate bad design.

- Using FDs to identify redundancies.
- Using FDs to decompose relations to eliminate the related update/insertion/delete issues.

7.2.3 Functional Dependencies

Informal definition : Value of attribute X determines value of attribute Y

X → Y

Assuming that X and Y are 2 sets of attributes, the relationship between X and Y value is modelled using a function

- 2 ways to determine functional dependencies
  - semantic meaning of attributes
  - actual data in tables

  In most cases, we use the former.

- Can deny one FD by looking at an instance of a relation, but can`t say there are one FD by looking at the whole instance of a relation.

- Armstrong's Axioms
  1. Reflexivity

$$\text{If } B \subseteq A, \text{ then } A \to B$$

  2. Augmentation

2. **Augmentation:** If $A \to B$, then $AC \to BC$ for any set of attributes C

- Example: *cpoints* $\to$ *wload* **implies** *cpoints, uos_name* $\to$ wload, *uos_name*

  3. Transitivity

3. **Transitivity:** If $A \to B$ and $B \to C$, then $A \to C$

- Example: *uos_code* $\to$ *cpoints*, *cpoints* $\to$ *wload* **implies** *uos_code* $\to$ *wload*

› Example

Products

Name	Color	Category	Dept	Price
Gizmo	Green	Gadget	Toys	49
Widget	Black	Gadget	Toys	59
Gizmo	Green	Whatsit	Garden	99

Functional Dependencies:

Name → Color
Category → Dept
Color, Category → Price

Given the above FDs, we *deduce* that *Name, Category* → *Price* must also hold on **any instance:** Let us use *augmentation* and *transitivity* as a proof

Name, Category → Color, Category
Color, Category → Price

Augmentation

Name, Category → Price	Transitivity

- Closure of a set of FDs F is the set F+
    - F+ means the F add the other FDs that can be deduced by F.
    - More formally

$$F^+ = \{X \rightarrow Y \mid F \models X \rightarrow Y\}$$

Initialize $F^+ = F$

Repeat

For each functional dependency FD in F^+

*Apply **reflexivity** and **augmentation** rules on F^+*

Add the result to F^+

For each pair of functional dependencies F_1 and F_2 in F^+

*If F_1 and F_2 can be combined using **Transitivity***

Add the result to F^+

Until F^+ does not change.

Example :

Assume that we have three attributes A, B, C in a relation R.

- With the following FDs, F = {A → B, B → C}.

› Using the previous algorithm, F^+ includes the following FDs:

A → A,	AB → A,	ABC → A,
A → B,	AB → B,	ABC → B,
A → C,	AB → C,	ABC → C,
B → B,	AC → A,	.... etc
B → C ,	AC → B,	
C → C	AC → C,	
	BC → B,	
	BC → C	
	....	

- Additional rules
    - Decomposition

      IF A → BC then A → B and A → C
    - Union

      If A → B and A → C, then A →BC

- Armstrong's Axioms are
    - Sound

      Generate only FDs in F+ when applied to a set F of FDs
    - Complete

      repeated application of these rules will generate all FDs in the closure F+

7.2.4 Functional Dependency and Keys

A superkey is a set of attributes that uniquely identify each tuple in a relation

- K is a superkey to relation R, then K → R and all attributes

A candidate key(or just key) is a minimal superkey

Example: Given a relation R, with attributes **ABCDE** (each letter denotes an attribute) where:

- **A** *uniquely identifies* each row in R
- **BC** also *uniquely identifies* each row in R (but *not B or C alone*)
- **A** is a superkey (*and candidate* key) for R
- **BC** is a superkey (*and candidate* key) for R
- **BCE** is a superkey (but ***not*** *a candidate key*)
  - because it is ***not*** minimal!

- Attribute Closure
  - X → X+

  If a set of attribute X, and X+ is all the attribute of the table, then X is a super key of the table.

  X will be candidate key if it is minimal.(None of its subset is a superkey)

  Algorithm to compute the closure X^+ of a set of attribute X:

  1. Initialise *result* with the *given set* of attributes, i.e., $X^+ = \{A_1, \ldots, A_n\}$ *(reflexivity rule)*
  2. Repeatedly search for some *FD* $A_1 A_2 \ldots A_m \rightarrow C$ such that all $A_1, \ldots, A_m$ are already in the set of attributes *result*, but C is *not.*
  3. Add C to the set *result. (transitivity and decomposition rules)*
  4. Repeat steps 2-3 until *no more attributes* can be added to *result*
  5. The set *result* is then the correct value of X^+

7.3 Normal Forms

- Goal : reduce different types of redundancies
- We focus on 1NF, 2NF, 3NF ,BCNF and 4NF
    - Based on FDs and MultiValued Dependencies (MVDs)
- Normalisation is the process to reduce specific types of redundancies.

7.3.1 First Normal Forms

a relation R is in 1NF if the domains of all attributes of R are atomic

- No multi Value

Student	UnitOfStudy
Mary	{COMP9120,COMP5318}
Joe	{COMP9120,COMP5313}

Violates 1NF

Student	UnitOfStudy
Mary	COMP9120
Mary	COMP5318
Joe	COMP9120
Joe	COMP5313

In 1NF

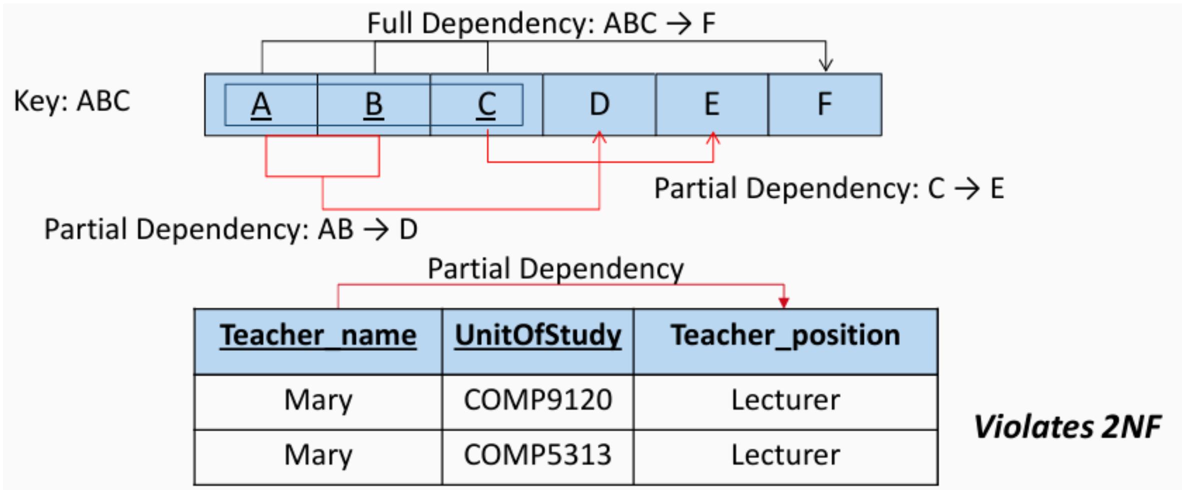
7.3.2 Second Normal Form

1NF + No partial dependencies

- Partial dependencies

  A non-trivial FD X → Y in R where X is a strict subset of some key for R and Y is not part of a key.

In other word, a not key attribute depend on the part of the candidate key.



Teacher_name	UnitOfStudy	Teacher_position
Mary	COMP9120	Lecturer
Mary	COMP5313	Lecturer

- Problem : redundancy

  A relation is in the 2NF if the closure F+ contains no functional dependency of the form:

  X → Y

  where Y is nonprime and X is a proper subset of a candidate key.

- Solution : Decompose the relation into 2 relations

Example above: Decompose R(Teach_name, UnitOfStudy, Teacher_position) into two relations: R1(Teacher_name, Teacher_position) and R2(Teacher_name, UnitOfStudy)

Teacher_name → Teacher_position

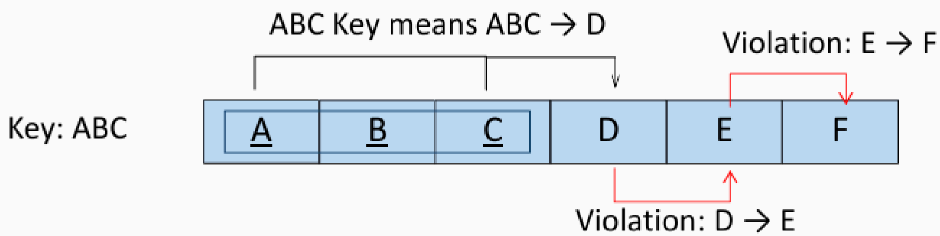
7.3.3 Third Normal Form

for each dependency X → y in F+, at least one of the following conditions holds :

- X → Y is a trivial FD (Y is the subset of X) (nature one)
- X is a superkey for R (nature one)

- Y is a proper subset of a candidate key for R

3 NF = 2 NF + any nonprime attribute cannot depend on candidate key through other nonprime attribute .



Employee_name	department	location
Jim	Research	Brisbane
John	Manufacturing	Melbourne
Mary	Research	Brisbane
Sue	Manufacturing	Melbourne

Violates 3NF

› There is a *functional dependency* between *Department* and *Location* (thus *transitive dependency*).

- Solution: split up the relation into 2 relations

- R1(Employee, Department) and R2(Department, Location)

Employee	Department

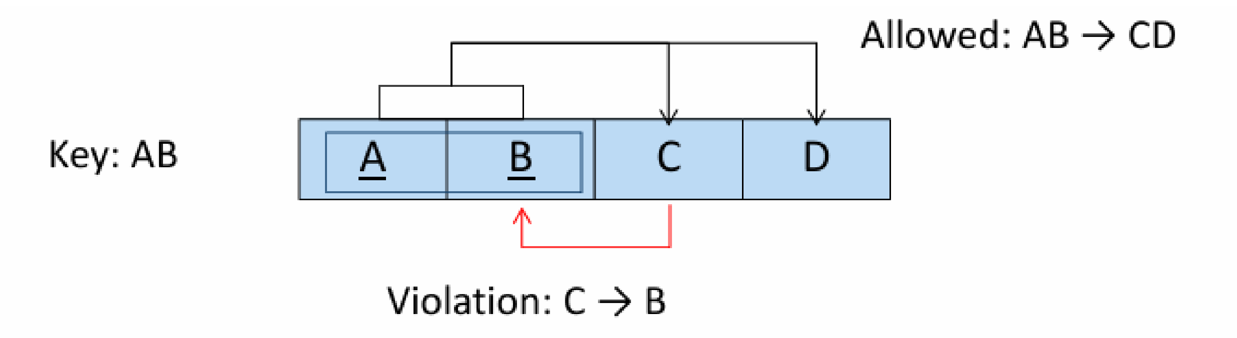
Department	Location

7.3.4 Boyce-Codd Normal Form

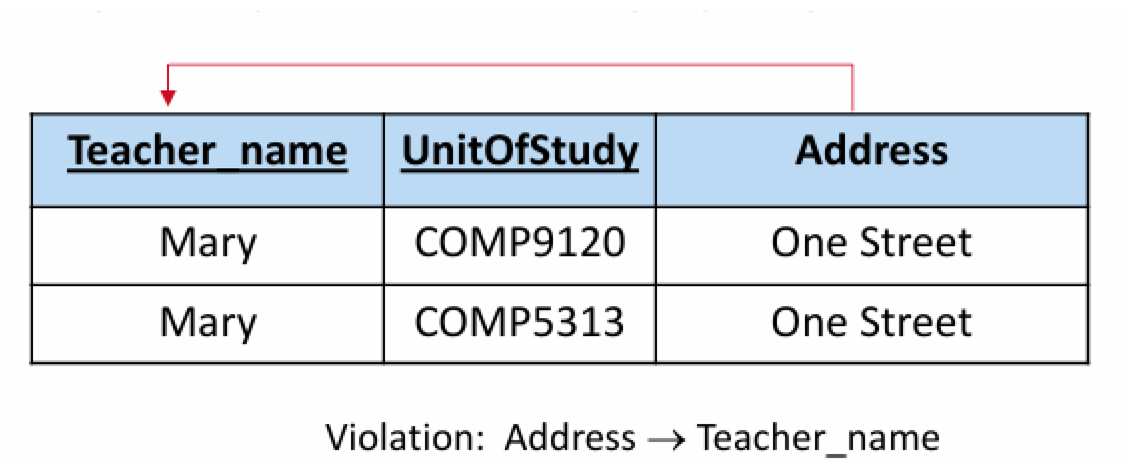
Stronger form of 3NF

For any non-trivial X → Y for R : X is a superkey for R

In other word : All dependency are from superkey.



BCNF is stronger form of 3NF



Teacher_name	UnitOfStudy	Address
Mary	COMP9120	One Street
Mary	COMP5313	One Street

- Problem : redundancy

  A teacher's name is repeated for every address that teaching a unit of study.

- Solution : split up the relation into 2 relations

- R1(Address, Teacher_name) and R2(Address, UnitofStudy)

Address	Teacher_name

UnitOfStudy	Address

7.3.5 4NF

- The issue of 1NF

name	profession	Language
John	{Electrician, Plumber}	French, Korean
Mary	{Doctor, Author}	Spanish, Chinese

name	profession	Language
John	Electrician	French
Mary	Doctor	Spanish
John	Plumber	Korean
Mary	Author	Chinese

THE values suggest the john is a electrician and speak French, and John is a plumber speaking Korean.

It has semantically incorrect.

- Solution

name	profession	Language
John	Electrician	French
Mary	Doctor	Spanish
John	Plumber	Korean
Mary	Author	Chinese
John	Plumber	French
Mary	Author	Spanish
John	Electrician	Korean
Mary	Doctor	Chinese

- MVD

  Multi Valued Dependency (MVD) between X and Y exist if no relationship can be inferred between X and Z (independent)

  $$X \twoheadrightarrow Y \ \ (X \ \textit{multidermines} \ Y)$$

  X can determine several Y, and Y is independent.

- 4NF

  Redundancy problem in MVDs:

  - For the first example : should list all professions for every language person speaks

  4NF deal with the redundancies created by multivalued dependencies

- R in 4NF if all MVDs of the Form X →> Y in f+, at least one of the following conditions holds:
  - X →> Y is a trivial MVD
  - X is a superkey of R

Assuming the *only key* to the following relation is the set : (Project-id, Personal-phone#):

employee_name	project_id	personal_phone_number
Bob	P1	047012345
Bob	P3	046098765
Bob	P1	046098765
Bob	P3	047012345
Lily	P1	045067543
Fiona	P7	043085432

Is this relation in 4NF?

No: There is at least *one non-trivial multivalued dependency*

employee_name ↠ Project-id (Note that employee_name is *not* a superkey).

Solution: split the above relation into two relations:

employee_name	Project_id

employee_name	Personal_phone_number

Now the two relations are in 4NF!

BCNF + `no 2+ MVDs`

7.4 Decomposition

Replace R by 2 or more distinct relations

- Each new relation schema contains a subset of the attributes of R
- Every attribute of R appears as an attribute in at least one of the new relations
- Many possible decompositions - not all good

Strong decomposition :

- Dependency preservation : No FDs are lost in the decomposition
- Lossless-join decomposition : Re-joining a decomposition of R should give back R.

- Dependency preservation

  F′ = F1 ∪ F2 ∪……∪ Fn-1 ∪ Fn

  If F` ≠ F , check F`+ = F+

- Lossless-join decomposition

  compose the split table into the original table, no extra data and no more data(tuple).

  If the intersection of the set of attributes between R1 and R2 functionally determines either R1 or R2.

  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

- Decomposing a schema into BCNF

  Suppose we have a schema R and a non-trivial dependency X → Y which causes a violation of BCNF. We decompose R into:

  - R1 = X U Y
  - R2 = R - Y

- Example schema that is *not* in BCNF:

loan_info = (*customer_id*, *loan_number*, *amount*) with *loan_number* $\rightarrow$ *amount*

but *loan_number* is *not a superkey*

- Assume,
  - X = *loan_number*
  - *Y* = *amount*

So, the relation *loan_info* is replaced by the following relations:

R_1 = (X ∪ Y) = (*loan_number*, *amount*)

R_2 = (*R* - *Y*) = (*customer_id*, *loan_number*)

Now both are in BCNF