Chapter 3 Design Theory for Relational Databases

Objectives

Understand concepts of:

- Functional Dependencies
- Normalization
- Decomposition
- Multi-valued Dependencies

Contents

- Functional Dependencies (FD)
- Rules about FDs
- Key & Super-Key
- Normal forms

Functional dependency

- A **functional dependency:** constraint between two sets of attributes in a relation
- Functional dependencies help to ensure the quality of data in the database.
- A functional dependency is indicated by an arrow →

Y dependent on X: X → Y

ID_employee	Name	Salary	living_Place
T001	Thuyền	2300	A105
T002	Hoa	5000	A107
T003	Vân	1900	A205

→ Have the value ID_ employee: we can get the employee name, their place of living and salary

Functional dependency

The advantages of functional dependencies

- Helps to avoid data redundancy
- Helps maintain the quality of data in the database
- Help define the meaning and constraints of the database
- Helps identify not good database designs
- Help find the facts related to the design of the database

Functional dependency

- A set of attributes X (include $A_1A_2\ldots A_n$) in R → $R(X)$

with $Y \subseteq$ **U** (include $B_1B_2\ldots B_m$)

functional dependency X **determine** Y (written $X \rightarrow Y$), is defined as:

if and only if each X value is associated with precisely one Y value.

- A functional dependency $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ holds on relation *R if two tuples of R agree on all of the attributes* A_1, A_2, …, A_n *then they must also agree on all of the attributes* B_1, B_2, …, B_m

∀ t, t' ∈ R if t.X = t'.X then t.Y = t'.Y

Functional dependency

title	*year*	*length*	*genre*	*studioName*	*starName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

Easy to see that: the following FD is true

- title,year $\rightarrow$ length, genre, studioName

Exercise: How about the FD

- title,year $\rightarrow$ starName

title, year → startName does not hold in Movies1 relation

→ Because there's more than one star in a movie

Functional dependency

roll_no	name	dept_name	dept_building
42	Thuyền	CO	A4
43	Hoa	IT	A3
44	Vân	CO	A4
45	Thủy	IT	A3
46	Hoàng	EC	B2
47	Anh	ME	B2

We have some valid functional dependencies:

- **roll_no → { name, dept_name, dept_building }**

→ roll_no can determine values: name, dept_name, and dept_building

-- **dept_name → dept_building**

--> Dept_name can identify the dept_building

- **{roll_no, name} → {dept_name, dept_building}**

Functional dependency

roll_no	name	dept_name	dept_building
42	Thuyền	CO	A4
43	Hoa	IT	A3
44	Vân	CO	A4
45	Thủy	IT	A3
46	Hoàng	EC	B2
47	Anh	ME	B2

Some invalid functional dependencies:

- **name → dept_name**

  --> Students with the same name can have different dept_name

- **dept_building → dept_name**

  --> There can be multiple departments in the same building

  Ex: ME and EC are in the same building B2

- **name → roll_no**
- **dept_building → roll_no**

Functional dependency

- Super-key
  - *Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation.*
  - *All super keys can't be candidate keys but the reverse is true.*
  - Every super-key satisfies the first condition of a key: it functionally determines all other attributes of the relation
  - If K is a key, L is a super key, then: $K \subseteq L$
  - And a key is also a super key

→ So, all those attributes in a table that is capable of identifying the other attributes → all super keys.

Functional dependency

R(Customer_name, Customer_ID, Social Security number (SSN), Address, Date_birth)

Superkeys are as follows:

SSN+Date_birth → Customer ID, Customer_name, Address

Customer_ID +SSN → Date_birth, Customer_name, Address

Key: Customer_ID +SSN

Functional dependency

- **Armstrong's Axioms**
  - Fundamental Rules: Let X, Y, Z are sets of attributes
    - **Reflexivity**

      If X is a subset of Y, then Y→X

      X ⊆Y, **Y → X**

      Ex: ABC →BC
    - **Augmentation (additional)**

      If X→Y, then XZ→YZ for any Z

      Ex: If C → D then ABC → ABD
    - **Transitivity**

    If X→Y and Y→Z, then X→Z

    Ex: If there is AB →C, C →EG, then AB →EG

Functional dependency

- **Union/Combining**: if X→Y AND X→Z then X→YZ

  Ex: If AB →CD and AB → EF, then AB → CDEF

- **Decomposition/Splitting**: if X→YZ, then X→Y and X→Z

  If X → Y and Z ⊆ Y then X → Z

  Ex: If AB → CDEF, then AB → CD and AB → EF

- **Pseudo transitivity** (like catching a bridge):

  If X → Y and WY → Z then WX →Z

  Ex: If AB → EF and DEF →G, then ABD → G

- Trivial FDs: right side is a subset of left side
  - Ex: FLD →FD

Functional dependency

- A set of FD's *S* **follows** from a set of FD's *T* if every relation instance that satisfies all the FD's in *T* also satisfies all the FD's in *S*

  A -> C follows from T = {A ->B, B -> C}

- Two sets of FD's *S* and *T* are **equivalent** if and only if *S* **follows** from *T*, and *T* **follows** *S*

  S = {A ->B,B -> C,A->C} and T={A-> B, B -> C} are equivalent

→ **These notions are useful in deriving new FDs from a given set of FDs.**

Functional dependency

Example

- Given the relation R (A, B, C, D, E, G, H)
- And the set of functional dependencies F = {B→D, AB→C, CD→E, EC→GH, G→A}

Apply Armstrong's:

1) AB → E
2) and AB→G

1. AB -> C (Hypothesis)
2. AB -> BC (Augmentation B)
3. B -> D (hypothesis)
4. BC -> DC (Augmentation C)
5. CD --> E (hypothesis)
6. BC -> E (Transitivity 4 and 5)
7. **AB -> E (Transitivity 2 and 6**)

8. AB -> EC (combining /Union 1 and 7)
9. EC -> GH (hypothetical)
10. AB -> GH (Transitivity 8 and 9)
11. **AB -> G (Decomposition/Splitting)**

Functional dependency

Example:

- Given the relation R= {A, B, C, E, F }
- And F= { AB--> C, C--> B , ABC --> E, F--> A}.
- Apply Armstrong's : FB -> E

1. F -> A (hypothesis)
2. FB -> AB (Augmentation B)
3. AB -> C (hypothesis)
4. ABC -> C (Augmentation)
5. ABC -> E (hypothesis)
6. ABC -> EC (combination of 4 and 5)
7. AB -> E (Decomposition/Splitting 6)
8. FB -> E (Transitivity 2 and 7)

Functional dependency

Example:

- Given the relation R= {A, B, C, D }
- And F = {A → B, A → C, BC → D}
- Apply Armstrong's : A → D

1. A → B
2. A → C
3. A → BC (combining /Union)
4. BC → D
5. A → D (transitivity)

The Closure of Attributes

- The closure of a set of attributes $\{A_1, A_2, \ldots, A_n\}$ under FD's in *S* *(denoted* $\{A_1, A_2, \ldots, A_n\}^+$ *or* F+ *)* is the set of attributes *B* such that every relation that satisfies all the FD's in set *S* also satisfies $A_1A_2\ldots A_n \rightarrow B$ (A^+) *(Armstrong's)*
- That is, $A_1A_2\ldots A_n \rightarrow B$ follows from the FD's of *S*
- $A_1, A_2, \ldots, A_n \in \{A_1, A_2, \ldots, A_n\}+$, because $A_1A_2\ldots A_n \rightarrow A_i$ is trivial

The Closure of Attributes

Example

- Given the relation schema Q(A,B,C,D)
- And F {A→ B; B →C; A→D; B →D}

Then, F+ = { A →B; B →C; A → D; B → D; A → BD; A → BCD; A → C; A → CD; A →BC; B →CD; …}

→ F ⊆ F+

The Closure of Attributes

Algorithm 3.7: Closure of a set of attributes

- Input: A set of attributes $\{A_1, A_2, \ldots, A_n\}$ and a set of FD's S
- Output: The closure $\{A_1, A_2, \ldots, A_n\}^+$

1. If necessary, split the FD's of *S*, so each FD in *S* have singleton right side

   (Split the FD's so that each FD has a single attribute on the right side)

2. Let *X* be a set of attributes that will become the closure. Initialize *X* to be $\{A_1, A_2, \ldots, A_n\}$

   (Initialize the closure set X by the set of given attributes.)

3. Repeatedly search for some FD: $B_1B_2 \ldots B_m \rightarrow C$, such that B_1, B_2, …, B_m are in *X*, but *C*
   a) If such C is found, add to X, and repeat the search
   b) If such C is not found, no more attributes can be added to X

4. The set X is the correct value of $\{A_1, A_2, \ldots, A_n\}+$

The Closure of Attributes

Example 1

- R(A,B,C,D,E,F)
- And the FD's AB-->C, BC-->AD, D-->E, and CF-->B (1)
- Compute {A,B}+

1. Split: BC --> AD into BC-->A and BC-->D

→ FD: AB-->C, BC-->A, BC-->D , D-->E, and CF-->B (2)

2. Start with X={A,B}

   1) AB-->C; A and B are in X

   → add C to X --> X={A,B,C}

   2) BC--> D, add D --> X={A,B,C,D}

   3) D-->E, add E --> X={A,B,C,D,E}

   4) Nothing new can be added

→ X^+ ={A,B,C,D,E}

Or {A,B}+ = {A,B,C,D,E}

The Closure of Attributes

Example 2

- R(A,B,C,D,E,F)
- And the FD's: AB -->C, BC --> AD, D-->E, and CF-->B
- AB--> D follows from the set of FD’s?

→ {A,B}+ which is {A,B,C,D,E}.

--> D is a member of the closure, we conclude that it follows FD’s

The Closure of Attributes

Example 3

- Suppose R(A,B,C,D,E,F) and the
- FD's AB -->C, BC-->AD, D-->E, and CF-->B

D--> A follows from the set of FD's?

- We compute {D}+
- We start from X={D}
- From D--> E, add E to the set --> X= {D,E}.
- No other FD's you can find that the left side is in X.

Since A is not in X. So, D --> A doesn't follow FD's

The Closure of Attributes

❑The key of the relation

- Given the relation r(R), the set $K \subset R$ is said to be the key of the relation r if: $K^+=R$
- If one element is removed from K, its closure will be different from R.
- Thus, the set $K \subset R$ is the key of the relation if $K^+=R$

$$(K \setminus A)^+ \neq R \text{ , } \forall A \subset R$$

The Closure of Attributes

Example 4

Student (Ssn, Sname, address, hscode, hsname, hscity, gpa, priority)

- ssn --> sname, address, gpa
- hscode --> hsname, hscity
- Gpa --> priority

{ssn, hscode}+ = {ssn, hscode}
= {ssn, hscode, sname, address, gpa}
= {ssn, hscode, sname, address, gpa, hsname, hscity }
= {ssn, hscode, sname, address, gpa, hsname, hscity, priority }

→ This forms a key for the relation

The Closure of Attributes

R(A, B, C, D)

$S=\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

Compute $\{A\}^+$?

$\{B\}^+$?

→ What are some the keys of R?

Closing Sets of Functional Dependencies

- Give a set of FD's *S*, any set of FD's T equivalent to *S* is said to be a ***basis*** for *S*.
  Then we say T is a ***basis*** for *S*
- A ***minimal basis*** for FD's *S* is a ***basis B*** that satisfies three conditions:
  - All the FD's in *B* have singleton right sides
  - If any FD is removed from *B*, the result is no basis
  - If for any FD in *B* we remove one or more attributes from the left side, the result is no basis

Closing Sets of Functional Dependencies

Example

- R(A,B,C)
- S={A→ B, A → C, B → A, B → C, C → A, C → B, AB → C, BC → A, AC → B, A → BC, B → AC, C → AB}
- have several minimal basis

- FD1={A → B, B → A, B → C, C → B}, or

- FD2={A → B, B → C, C → A}

we say set FD's {FD1, FD2} is a basis for S

What happens to …

… a set of FD's *S* of *R* when we project *R* on some attributes?

That is, give a relation *R* with set of FD's *S*, and $R_1 = \pi_L(R)$. What FD's hold in R_1?

Projecting Functional Dependencies

To find a functional dependencies of projection:

- Follow from *S*, and
- only attributes of R_1

Projecting Functional Dependencies

Algorithm 3.12: Projecting a Set of FD's

- **Input**: R, $R_1 = \pi_L(R)$, *S* a set of FD's that hold in R
- **Output**: the set of FD's that hold in R_1
- **Method**:
  - T is the set of FD's that hold in R_1. Initially, T is empty
  - For each set of attributes X of R_1, compute *X*+.

  Add to *T* all non-trivial FD's $X \rightarrow A$ such that *A* is both in *X*+ and an attribute of R_1

  → *Now, T is a basis for the FD's that hold in R1 but may not be a minimal basis. Modify T as follows:*

  - Construct a minimal basis from *T*

Projecting Functional Dependencies

- Compute a minimal basis from *T*

  a)If there is an FD *F* in *T* that follows from other FD's in *T*, then remove *F* from *T*

  b) Let Y → B is a FD in *T* , with at least two attributes in *Y. Remove one* attribute from Y and call it *Z.*

  If *Z* → *B* follows from the other FD's in T (including *Y* → *B*), then replace *Y* → *B* by *Z* → *B*

  c) Repeat the above steps in all possible ways (b) until no more changes to *T* can be made

Projecting Functional Dependencies

Two notations

- (1) Closing the empty set and the set of all attributes cannot create new FD's
- (2) The closure of some set *X* is all attributes, then we cannot find any new FD's by closing supersets of *X*

Projecting Functional Dependencies

Example: Suppose R(A,B,C,D)

FD's $A \rightarrow B$, $B \rightarrow C$, and $C \rightarrow D$

$R1 = \pi_{A,C,D}(R)$.

Find the FD's of R1?

We should find all subsets of {A,C,D}

- note that:

+{} and {A,C,D} will give us trivial FD's.

+ If the of some set X has all attributes , then we cannot find closured any new FD's by closing supersets of X.

Projecting Functional Dependencies

- Compute the closure of the single set

- {A}+={A,B,C,D}.

  Thus, A→A, A→B, A→C, and A→D hold in R.

  But A→A is trivial, A→B contains B that is not in R1.

  So, we pick new FD's A → C, and A→ D that would hold on R1.

- {C}+={C,D}, then new FD's C→D
- {D}+={D}, no new FD's

Projecting Functional Dependencies

- Compute the closure of the double set
  - Since {A}+ include all attributes of R ({A}+={A,B,C,D}), no care any more for supersets of {A}

→ thus, we cannot find any new FD's by closing supersets of A. So, we don't need to compute {A,C}+, {A,D}+

  - {C,D}+={C,D}, no new FD's holds in R1

(*Thus, CD --> C, and CD-->D hold for R which both are trivial*)

- Finally, there are three FD's A→C, A→D, C→D hold in R1
- A→D is transitive from A→C, and C→D

  (transitive rule)

- So, minimal basis of R1 {A→C, C→D}

Anomalies introduction

Careless selection of a relational database schema can lead to redundancy and related anomalies

So, in this session we shall tackle the problems of relational database designing

Problems such as redundancy that occur when we try to cram too much into a single relation are called *anomalies*

Anomalies

title	*year*	*length*	*genre*	*studioName*	*starName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With The Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

The principal kinds of anomalies that we encounter are:

Redundancy*:* information maybe repeated unnecessarily in several tuples (**exp**: the title, year, length, genre and studioName: Star Wars, 1977, 124, SciFi, and Fox is repeated.)

Update Anomalies*:* We may change information in one tuple but leave the same information unchanged in another (**exp**: if we found that *Star Wars* is 125 minutes long, we may change the length in the first tuple but not in the second and third tuples) → you will lose the integrity.

Deletion Anomalies*:* If a set of values becomes empty, we may lose other information as a side effect (**exp**: if we delete "Fox" from the set of studios, then we have no more studios for the movie "Star Wars"

Decomposing Relations (divided)

- The accepted way to eliminate the anomalies is to decompose the relation into smaller relations.
- It means we can split the attributes to make two new relations.
- The new relations won’t have the anomalies.
- **But how can we decompose?**

Decomposing Relations (divided)

Definition: Given a relation $R(A_1,..,A_n)$, we say R is decomposed into $S(B_1,..,B_m)$ and $T(C_1,..,C_k)$ if:

+ $\{A_1,..,A_n\} = \{B_1,..,B_m\} \cup \{C_1,..,C_k\}$

+ $S = \prod_{B1,..Bm}(R)$

+ $T = \prod_{C1,..,Ck}(R)$

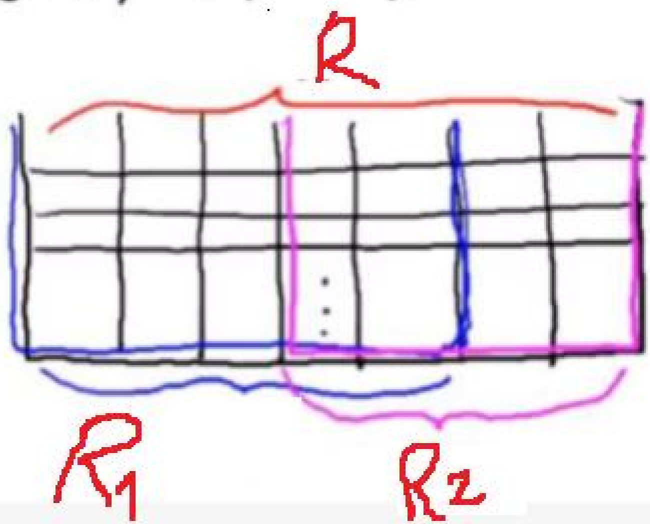
Decomposing Relations

$R(A_1, \ldots, A_n)$

$\hookrightarrow R_1(B_1, \ldots, B_k)$

$R_2(C_1, \ldots, C_m)$

$R_1 \bowtie R_2 = R$ ✓



Example: Decomposition

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With The Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

title	year	length	genre	studioName
Star Wars	1977	124	SciFi	Fox
Gone With The Wind	1939	231	drama	MGM
Wayne's World	1992	95	comedy	Paramount

title	year	starName
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone With	1939	Vivien Leigh
Wayne's W	1992	Dana Carvey
Wayne's W	1992	Mike Meyers

Discuss

title	*year*	*length*	*genre*	*studioName*
Star Wars	1977	124	SciFi	Fox
Gone With The Wind	1939	231	drama	MGM
Wayne's World	1992	95	comedy	Paramount

title	*year*	*starName*
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone With	1939	Vivien Leigh
Wayne's W	1992	Dana Carvey
Wayne's W	1992	Mike Meyers

Do you think that the anomalies are gone?

- Redundancy

- Update

- Delete

- The redundancy is eliminated (the length of each film appears only once)
- The risk of an update anomaly is gone (we only have to change the length of *Star Wars* in one tuple)
- The risk of a deletion anomaly is gone (if we delete all the stars for *Gone with the wind*, that deletion makes the movie disappear from the right but still be found in the left)

Decomposition: The Good, Bad and Ugly

We observed that before we decompose a relation schema into BCNF (Boyce Codd Normal Form), it can exhibit anomalies; That's the "Good"

However, decomposition can also have some bad:

- Maybe we can't recovery the original information; OR
- After reconstruction, the FDs maybe not hold

Example: Loss of information after decomposition

Deomposed: R1 and R2

R1 X R2 = R3

R (A,B,C)		
A	**B**	**C**
1	2	1
2	5	3
3	3	3

R1 (A,B)	
A	**B**
1	2
2	5
3	3

R2 (B,C)	
B	**C**
2	1
5	3
3	3

R3 (A,B,C)		
A	**B**	**C**
1	2	1
2	5	3
3	3	3

→ Thus, we conclude that the above decomposition is lossless when decomposing

Example: Loss of information after decomposition

Deomposed: R1 and R2

Natural Join: R1 x R2

R

A	B	C
1	2	3
4	2	5

R1

A	B
1	2
4	2

R2

B	C
2	3
2	5

R3

A	B	C
1	2	3
1	2	5
4	2	3
4	2	5

R3 = R1 X R2 (but R3 <> R1)

→ This relation is not same as the original relation R and contains some extraneous tuples

→ Clearly, R1 X R2 ⊃ R. Thus, we conclude that the above decomposition is lossy join decomposition

Normal Forms

- First Normal Form
- Second Normal Form
- Third Normal Form
- Boyce-Codd Normal Form
- Fourth Normal Form
- Fifth Normal Form

Dependencies: Definitions

- ***Multivalued Attributes:*** values of which are not uniquely identified

→ Ex: a large company could have many departments, some of them possibly in different cities.

- ***Partial Dependency*** – when a non-key attribute is determined by a part, but not the whole primary key

StudentID	ProjectNo	StudentName	ProjectName
S01	199	Katie	Geo Location
S02	120	Ollie	Cluster Exploration

- The prime key: StudentID + ProjectNo
- StudentID --> StudentName   Partial Dependent
- ProjectNo --> ProjectName   Partial Dependent

Dependencies: Definitions

- ***Transitive Dependency*** – when a non-key attribute determines another non-key attribute

Show_ID	Film_ID	Film_Type	CD_Cost ($)
F08	S09	Thriller	50
F03	S05	Romantic	30
F05	S09	Comedy	20

- **Show_ID** -> Film_ID
- Film_ID -> Film_Type

--> Transitive type of functional dependency

1NF

1NF A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only (monovalue, indivisible) and the value of each attribute is also a single value

StudentID :primary key.

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English Maths Info Tech	$50 $50 $100	B A B+

Is it 1NF?

1NF

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

1NF

- There are repeating groups (subject, subjectcost, grade)

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English Maths Info Tech	$50 $50 $100	B A B+

How can you make it 1NF?

1NF

Create new rows so each cell contains only one value

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English Maths Info Tech	\$50 \$50 \$100	B A B+

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

1NF

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389,	Punjab

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

1NF

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

But now look – is the *studentID* primary key still valid?

1NF

No – the studentID no longer uniquely identifies each row

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

- *studentID* and *subject* together to uniquely identify each row.

- So the new key is StudentID *and* Subject

1NF

So. We now have 1NF

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

Is it 2NF?

2NF

A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key

Example

R = (ABCD)
Primary key: AB
F = {AB -> C, AB -> D}

--> C and D are non-key attributes --> fully functional dependent on the primary key

2NF

Example

R = (ABCD)

Primary key : AB

F = {AB -> C, AB -> D, B -> DC}

→Not 2NF

--> Because : B -> DC is a partial dependency (not fully functional dependent) on the Primary key

2NF

StudentName & Address are dependent on studentID (which is part of the key)

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

But they are not dependent on ***Subject*** (the *other* part of the key)

2NF

And 2NF requires…

Problem

All non-key fields are dependent key: studentID + subject

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

2NF

So it's not 2NF

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

How can we fix it?

Make new tables

- Make a new table for each primary key field

- Give each new table its own primary key

- Move columns from the original table to the new table that matches their primary key…

Step 1

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

Create 1 new table

STUDENT TABLE (key = StudentID)

Step 2

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

Create 1 new table

SUBJECTS TABLE (key = Subject)

Step 3

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
StudentID	Mary Watson	10 Charles Street	Bob	Red	Subject	$50	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X					Maths		A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

Create 1 new table

RESULTS TABLE (key = StudentID+Subject)

Step 3

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

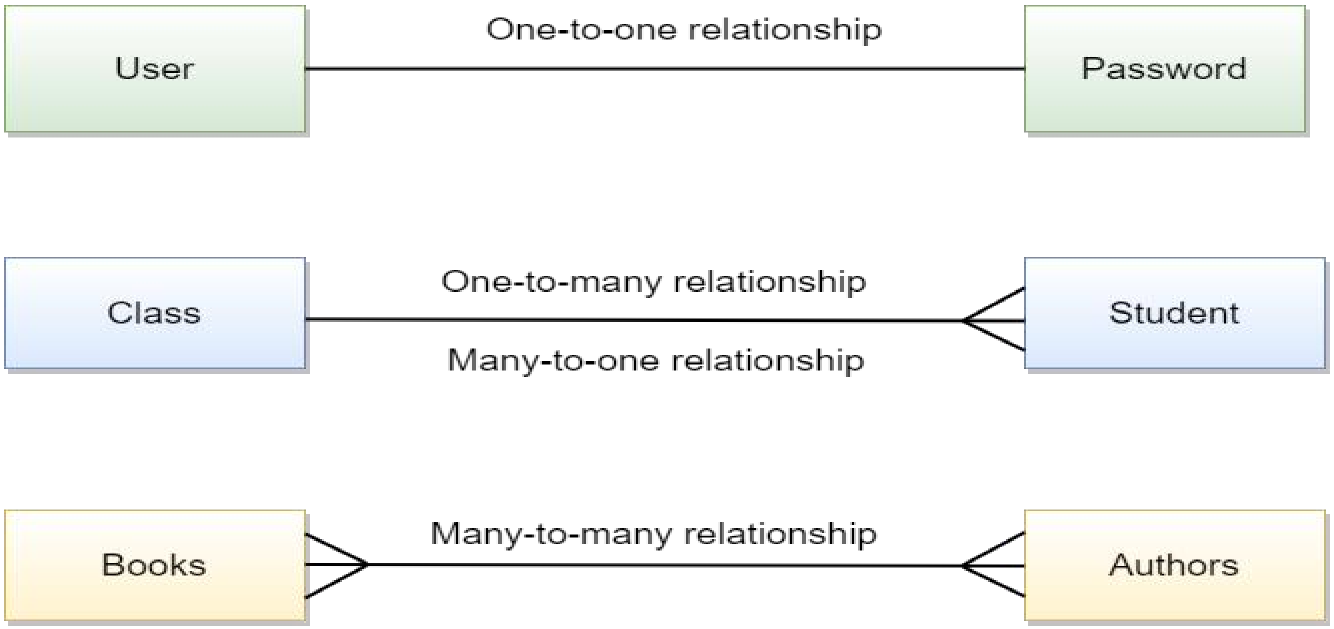
RESULTS TABLE (key = StudentID+Subject)

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

3NF

There are 3 main types of relationship in a database

- One-to-one relation
- 1-n relation
- n-n relation



3NF

PERSONAL

EmployeeID	FirstName	LastName	Address	City	State	Zip
EN1-10	Carol	Schaaf	2306 Palisade Ave.	Union City	NJ	07087
EN1-12	Gayle	Murray	1855 Broadway	New York	NY	12390
EN1-15	Steve	Baranco	742 Forrest St.	Kearny	NJ	07032

PAYROLL

One-To-One Relationship

EmployeeID	PayRate
EN1-10	$25.00
EN1-12	$27.50
EN1-15	$20.00

3NF

CUSTOMERS

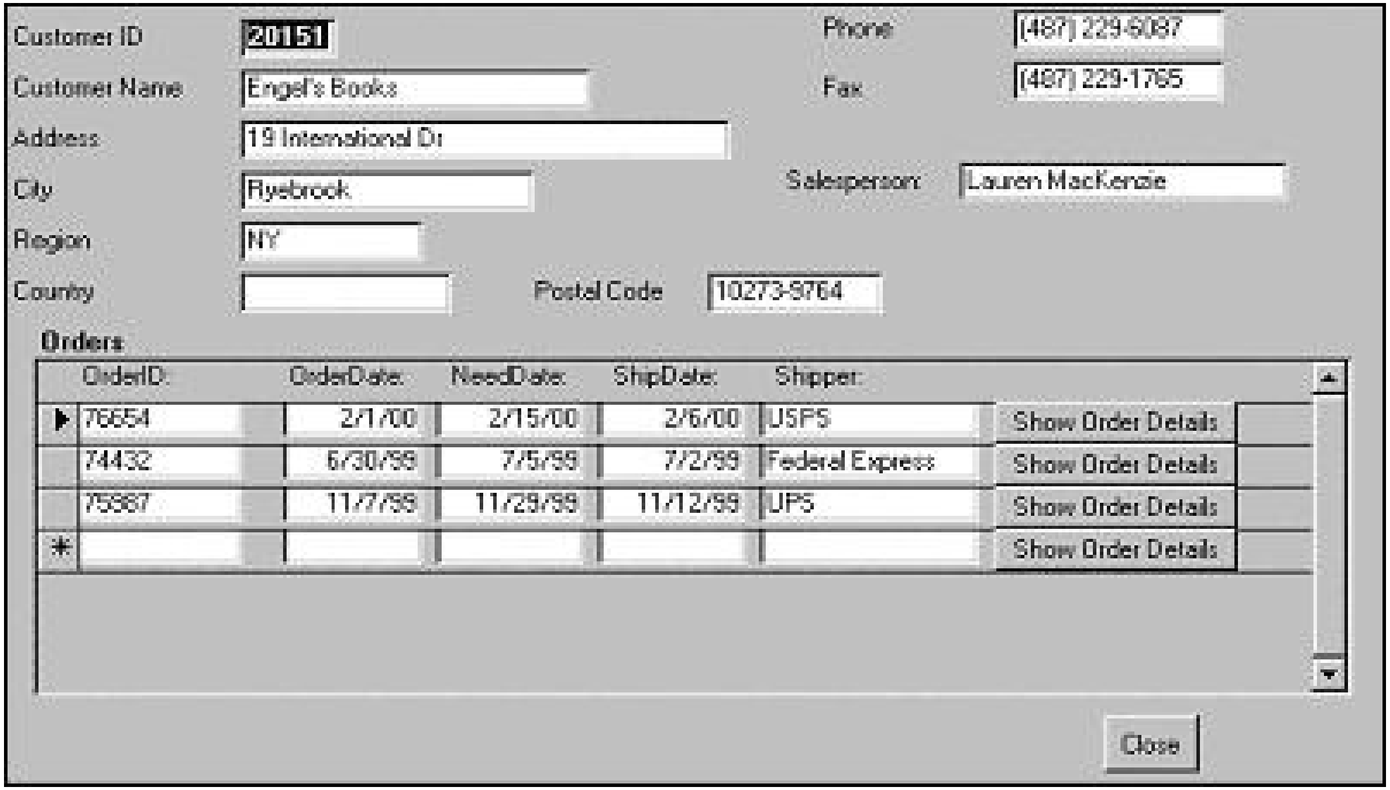
CustomerID	CustomerName	Address	City	State	Zip
20151	Engel's Books	19 International Dr	Ryebrook	NY	10273-9764
20493	Jamison Books	396 Apache Ave	Fountain Valley	CA	92708-4982
20512	Gardening Galore	79 Gessner Pk	Houston	TX	77024-6261

ORDERS

One-To-Many Relationship

OrderNum	CustomerID	OrderDate	ShipDate	Shipper
76654	20151	2/1/00	2/6/00	USPS
74432	20151	6/30/99	7/2/99	Federal Express
75987	20151	11/10/99	11/12/99	UPS

3NF



Customer ID 20151

Customer Name Engel's Books

Address 19 International Dr

City Rybrook

Region NY

Country

Postal Code 10273-9764

Phone (487) 229-6087

Fax (487) 229-1765

Salesperson: Lauren MacKenzie

Orders

OrderID:	OrderDate:	NeedDate:	ShipDate:	Shipper:	
76654	2/1/00	2/15/00	2/6/00	USPS	Show Order Details
74432	6/30/99	7/5/99	7/2/99	Federal Express	Show Order Details
75387	11/7/99	11/29/99	11/12/99	UPS	Show Order Details
					Show Order Details

Close

3NF

EMPLOYEES

EmployeeID	Last Name	First Name	ProjectNum
EN1-26	O'Brien	Sean	30-452-T3
EN1-26	O'Brien	Sean	30-457-T3
EN1-26	O'Brien	Sean	31-124-T3

PROJECTS

Many-To-Many Relationship

ProjectNum	ProjectTitle	EmployeeID
30-452-T3	Woodworking Around The House	EN1-26
30-452-T3	Woodworking Around The House	EN1-33
30-452-T3	Woodworking Around The House	EN1-35
30-457-T3	Basic Home Electronics	EN1-26
30-482-TC	The Complete American Auto Repair Guide	EN1-33
31-124-T3	The Sport Of Hang Gliding	EN1-26
31-124-T3	The Sport Of Hang Gliding	EN1-33

Step 4 - relationships

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Step 4 - cardinality

cardinality usually represents the relationship between the data in two different tables

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

Each student can only appear ONCE in the student table

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Step 4 - cardinality

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

SUBJECTS TABLE (key = Subject)

1

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

Each subject can only appear ONCE in the subjects table

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Step 4 - cardinality

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

1

A subject can be listed MANY times in the results table (for different students)

∞

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Step 4 - cardinality

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

1

A student can be listed MANY times in the results table (for different subjects)

∞

∞

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

A 2NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

1

SubjectCost is only dependent on the primary key, *Subject*

∞

∞

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

A 2NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Grade is only dependent on the primary key (*studentID + subject*)

A 2NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

Name, Address are only dependent on the primary key (*StudentID*)

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

1

∞

∞

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

So it is 2NF!

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

1

∞

∞

But is it 3NF?

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Bike_Warehouse

part	supplier	quantity	supplier country
Saddle	Bikeraft	10	USA
Brake lever	Tripebike	5	Italy
Top tube	UpBike	3	Canada
Saddle	Tripebike	8	Italy

part, supplier → quantity

supplier → Supplier country

Not 2NF?

- **{part, supplier}** : only candidate key
- **Supplier country**: functionally dependent on **supplier**

Bike parts

part	supplier	quantity
Saddle	Bikeraft	10
Brake lever	Tripebike	5
Top tube	UpBike	3
Saddle	Tripebike	8

Suppliers

supplier	supplier country
Bikeraft	USA
Tripebike	Italy
UpBike	Canada

3NF

A relation R is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively (bridge) dependent on the primary key.

- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.

- Non-Primary key columns shouldn't depend on the other non-Primary key columns

An attribute C is transitively dependent on attribute A if there exists an attribute B such that: A->B and B->C

3NF

Example

R = ABCDGH

Primary key: AB

F = {AB -> C, AB -> D, AB -> GH}

No transitive dependency --> 3NF

3NF

Example

R = ABCDGH

Primary key: AB

F = {AB -> C, AB -> D, AB -> GH, G -> DH}

→not 3NF

→Because, transitive dependency : G not primary

AB → GH

G --> DH

Fpt University

A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

Oh oh…

What?

SUBJECTS TABLE (key = Subject)

1

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

∞ ∞

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

HouseName is dependent on both ***StudentID* + *HouseColour***

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

Or **HouseColour** is dependent on both **StudentID + HouseName**

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

***But either way, non-key fields** are dependent on MORE **THAN THE PRIMARY KEY** (studentID)*

SUBJECTS TABLE (key = Subject)

1

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

*And **3NF** says that **non-key fields** must **depend on** nothing but the key → **not key***

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

WHAT DO WE DO?

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Again, carve off the offending fields

StudentTable

StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob

Primary key: StudentID

1

∞

∞

1

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

A 3NF fix

Create new table

StudentTable

StudentID	StudentName	Address
19594332X	Mary Watson	10 Charles Street

Primary key: StudentID

HouseTable

HouseName	HouseColor
Bob	Red

Primary key: HouseName

1

∞

∞

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

1

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

A 3NF fix

StudentTable

StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob

Primary key: StudentID

∞

1

1

HouseTable

HouseName	HouseColor
Bob	Red

Primary key: HouseName

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

∞

∞

1

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

A 3NF win!

STUDENTTABLE (key = StudentID)

StudentTable

StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob

Primary key: StudentID

HOUSE TABLE (key = HouseName)

HouseTable

HouseName	HouseColor
Bob	Red

Primary key: HouseName

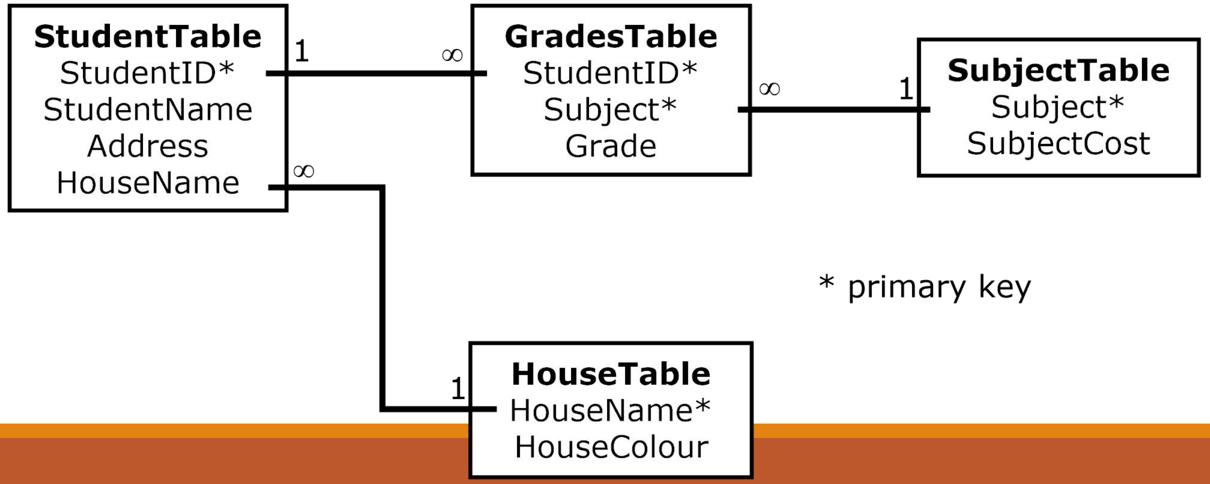
StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

SUBJECTS TABLE (key = Subject)

Or…



* primary key

The Reveal

Before…

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English Maths Info Tech	$50 $50 $100	B A B+

After…

HouseTable

HouseName	HouseColor
Bob	Red

Primary key: HouseName

StudentTable

StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob

Primary key: StudentID

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

SUBJECTS TABLE (key = Subject)

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Order_informati

order_id	date	customer	customer email
1/2020	2020-01-15	Jason White	white@example.com
2/2020	2020-01-16	Mary Smith	msmith@mailinator.com
3/3030	2020-01-17	Jacob Albertson	jasobal@example.com
4/2020	2020-01-18	Bob Dickinson	bob@fakemail.com

- Candidate key: ***order_id***

- Non-prime attributes: ***date, customer, customer email***

order_id → date, order_id → customer, customer → customer email

--> customer email depends on customer

We split this into two relations:

Orders (with the attributes order_id, date, and customer)

Customers (with the attributes customer and customer email)

Orders

order_id	date	customer
1/2020	2020-01-15	Jason White
2/2020	2020-01-16	Mary Smith
3/3030	2020-01-17	Jacob Albertson
4/2020	2020-01-18	Bob Dickinson

Customers

customer	customer email
Jason White	white@example.com
Mary Smith	msmith@mailinator.com
Jacob Albertson	jasobal@example.com
Bob Dickinson	bob@fakemail.com

BCNF

Boyce-Codd Normal Form(**BCNF**) is an advanced version of 3NF as it contains **additional constraints** 3NF

- 3NF
- For every **non-trivial functional dependency X -> Y**, **X** is the **superkey** of the table.

A superkey is a set of one or more attributes that can uniquely identify a row in a database table

- Là 3NF
- Không có thuộc tính khóa mà phụ thuộc hàm vào thuộc tính không khóa.

BCNF

Example

R = ABCDGH

Primary key: AB

F = {AB -> C, AB -> D, AB -> GH}

--> **BCNF**

BCNF

Example

R = ABCDGH

Key: AB

F = {AB -> C, AB -> D, AB -> GH, H -> B}

--> **not BCNF**

Because, B that is functionally dependent on the non-key attribute H: H--> B

BCNF

– Remove key attributes that are functionally dependent on non-key attributes from the relation

– Separate them into new relation -> primary key is the non-key attribute causing the dependency.

BCNF

EmployeeProject

Employee_Code	Project_ID	Project_Leader
101	P03	Grey
101	P01	Christian
102	P04	Hudson
103	P02	Petro

Candidate key {Employee Code, Project ID}

- 3NF
- But it violates the rules of BCNF

Non-trivial functional dependency: Project_Leader -> Project_ID

Project_ID is a prime attribute

Project Leader is a non-prime attribute

→ This is not allowed in BCNF

BCNF decomposition algorithm (self studying)

Input: A relation R with a set of FD’s F

Output: A BCNF decomposition of R with lossless join **Method**:

- At each step compute the key for the sub-relation R
- if not in BCNF, pick any FD X->Y which violates
- break the relation into 2 sub-relations
  - R1(XY)
  - R2(S - Y)
  - this has a lossless join
  - project FD's onto each sub-relation
- continue until no more offending FD's

3NF decomposition algorithm – self studying

Input: A relation R with a set of FD's F

Output: A decomposition of R into a collection of relations, all of which are in 3NF. This decomposition has a lossless join and dependency-preservation.

Method:

- Find minimal basic for F, say G.
- ∀ X-A ∈ G, use XA as the schema of one relations in the decomposition.
- If none of the sets of relations from Step 2 is a super key for R, add another relation whose schema is a key for R.

Summary 1

Decompose a relation into BCNF is a solution for eliminating anomalies

But BCNF can cause information loss and dependency loss

3NF is a relax solution of BCNF that keep loss-less join and dependency-preservation properties

Summary 2:

2NF	3NF	Boyce-Codd
every nonprime attribute *A* in *R* is not partially dependent on *any* key of R	a *nontrivial* functional dependency: *X* => *A* holds in *R,* either (a) *X* is a superkey of *R,* or (b) *A* is a prime attribute of *R.*	a *nontrivial* functional dependency *X* => *A* holds in *R,* then: a) *X* is a superkey of *R*
Note:A functional dependency *X* => *Y* is a **full functional dependency** if removal of any attribute *A* from *X* means that the dependency does not hold any more; A **partial functional dependency** is not a **full functional dependency**		