Week 6: Functional Dependency and Normal Forms (cont.)

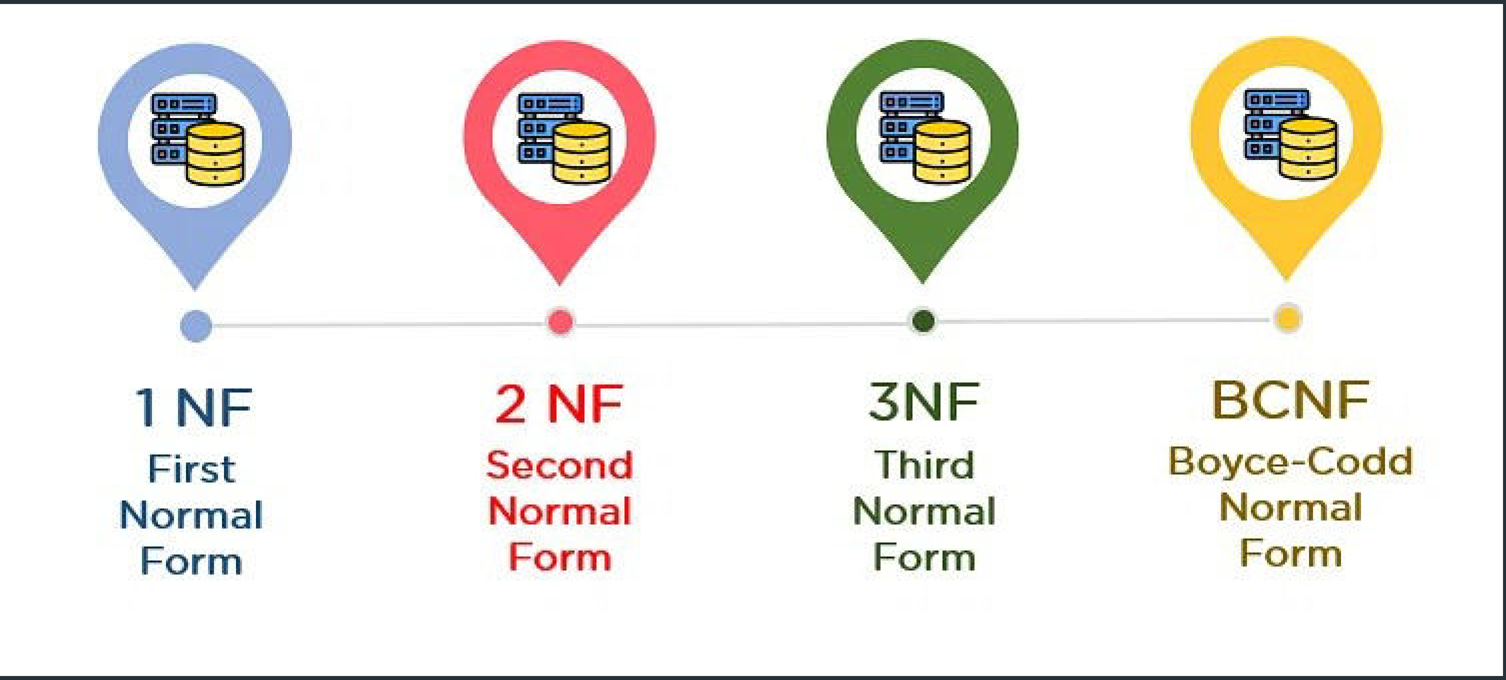
L6.1: Relational Database Design/6: Normal Forms

Normal Forms

- Normalization is used for mainly two purposes:
  - To eliminate redundant data.
  - To ensure data dependencies make sense, i.e. data is logically sorted.

$$\text{Normalization} \implies \text{Good decomposition} \implies \text{Minimization of Dependency} \implies \text{Redundancy} \implies \text{Anomaly}$$

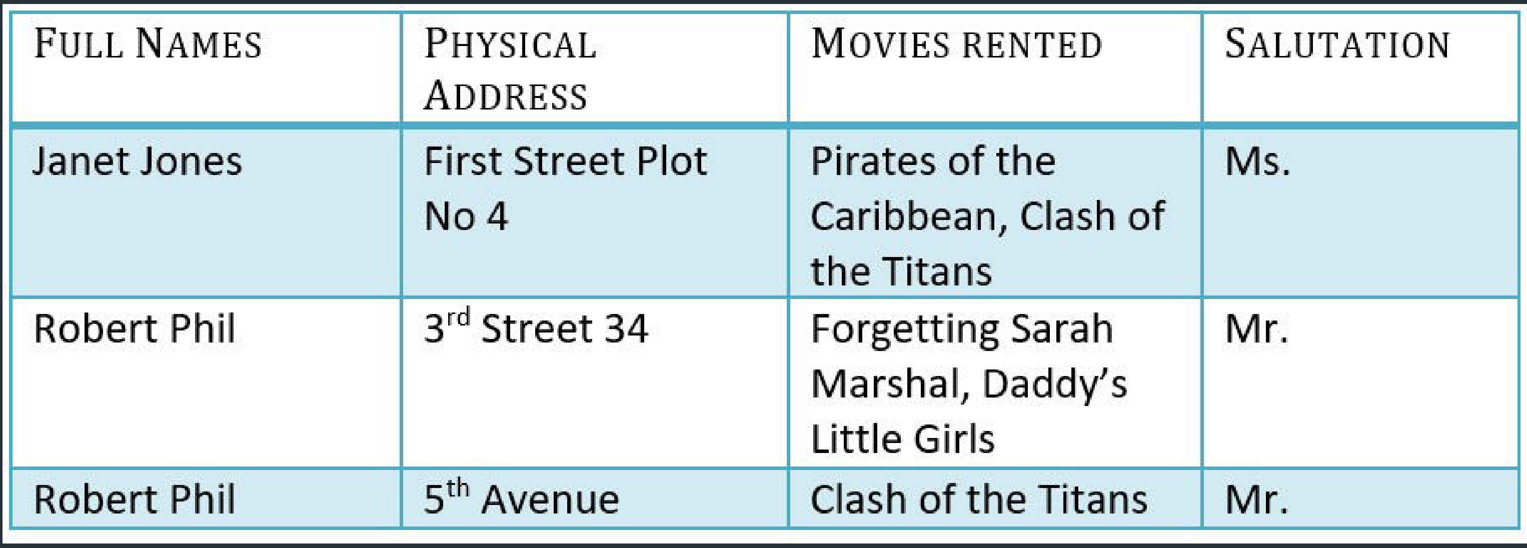
- To know more about normalization, check here



Normalization and Normal Forms

- A normal form specifies a set of conditions that the relational schema must satisfy in terms of its constraints.
- Some common normal forms are:
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)
- A relation is said to be "**normalized**" if it meets *third normal form (3NF)*.

    - Most 3NF relations are free of update, insertion, and deletion anomalies.
- We will consider this relation as example in this lecture:

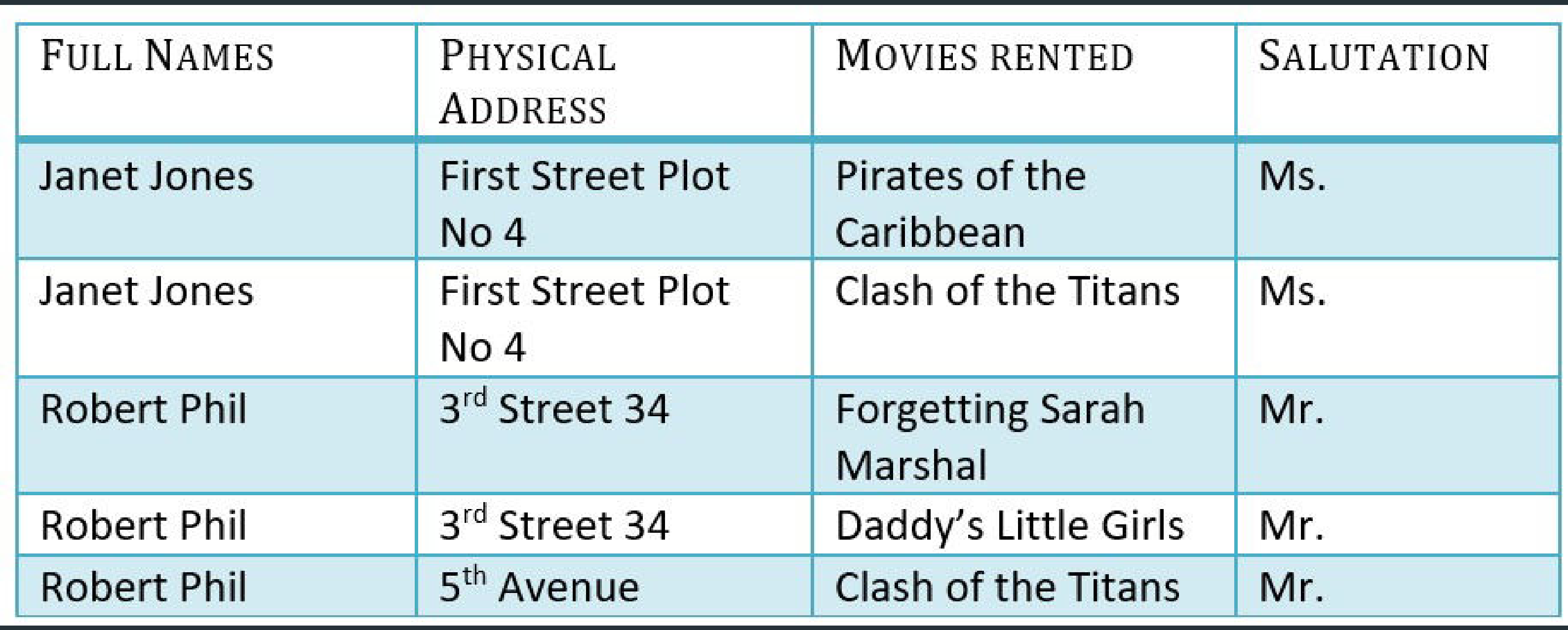


FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

First Normal Form (1NF)

- **Rule** - Domain of attribute must include only atomic values.

Relation in 1NF:

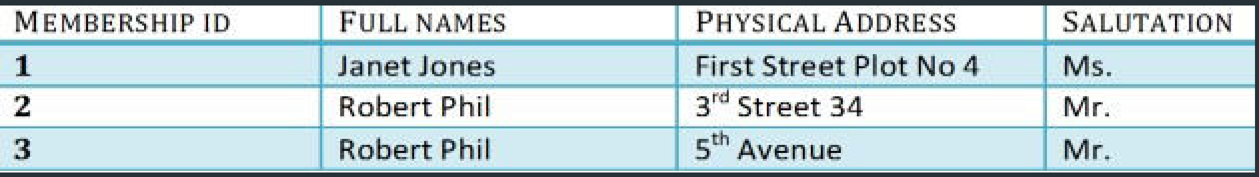


FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

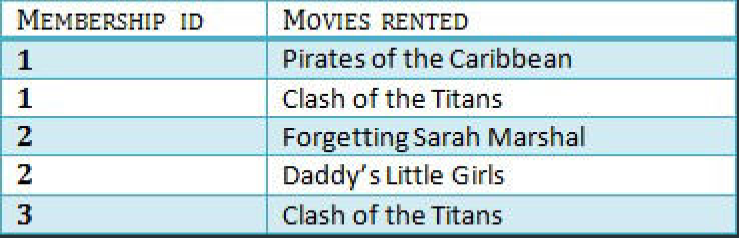
Second Normal Form (2NF)

- **Rule 1** - Relation must be in 1NF.
- **Rule 2** - Relation does not contain any partial dependency.
    - Single column primary key that does not functionally dependent on any subset of candidate key relation.

Relations in 2NF:



MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

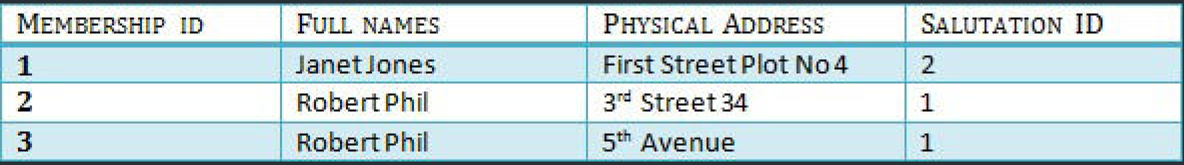


Membership ID	Movies rented
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

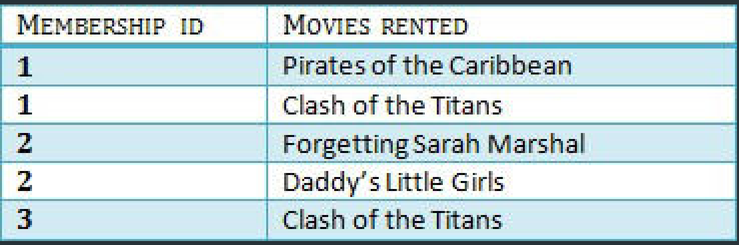
Third Normal Form (3NF)

- **Rule 1** - Relation must be in 2NF.
- **Rule 2** - Relation does not contain any transitive dependency.
    - Non-prime attribute is not functionally dependent on any other non-prime attribute.

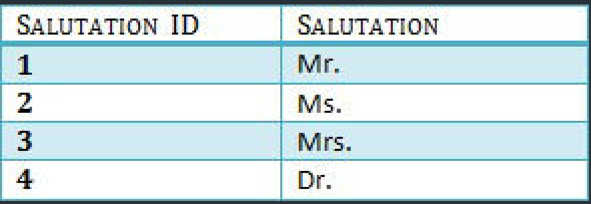
Relation in 3NF:



Membership ID	Full names	Physical Address	Salutation ID
1	Janet Jones	First Street Plot No 4	2
2	Robert Phil	3rd Street 34	1
3	Robert Phil	5th Avenue	1



Membership ID	Movies rented
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans



Salutation ID	Salutation
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

L6.2: Relational Database Design/7: Normal Forms

3NF Decomposition

Testing

- **Optimization**: Need to check only FDs in F, need not check all FDs in F^+.
- Use attribute closure to check for each dependency $\alpha \rightarrow \beta$, if α is a superkey.
- If α is not a superkey, then we have to verify if each attribute in β in a candidate key of R.
    - Decomposition into #NF can be done in polynomial time.

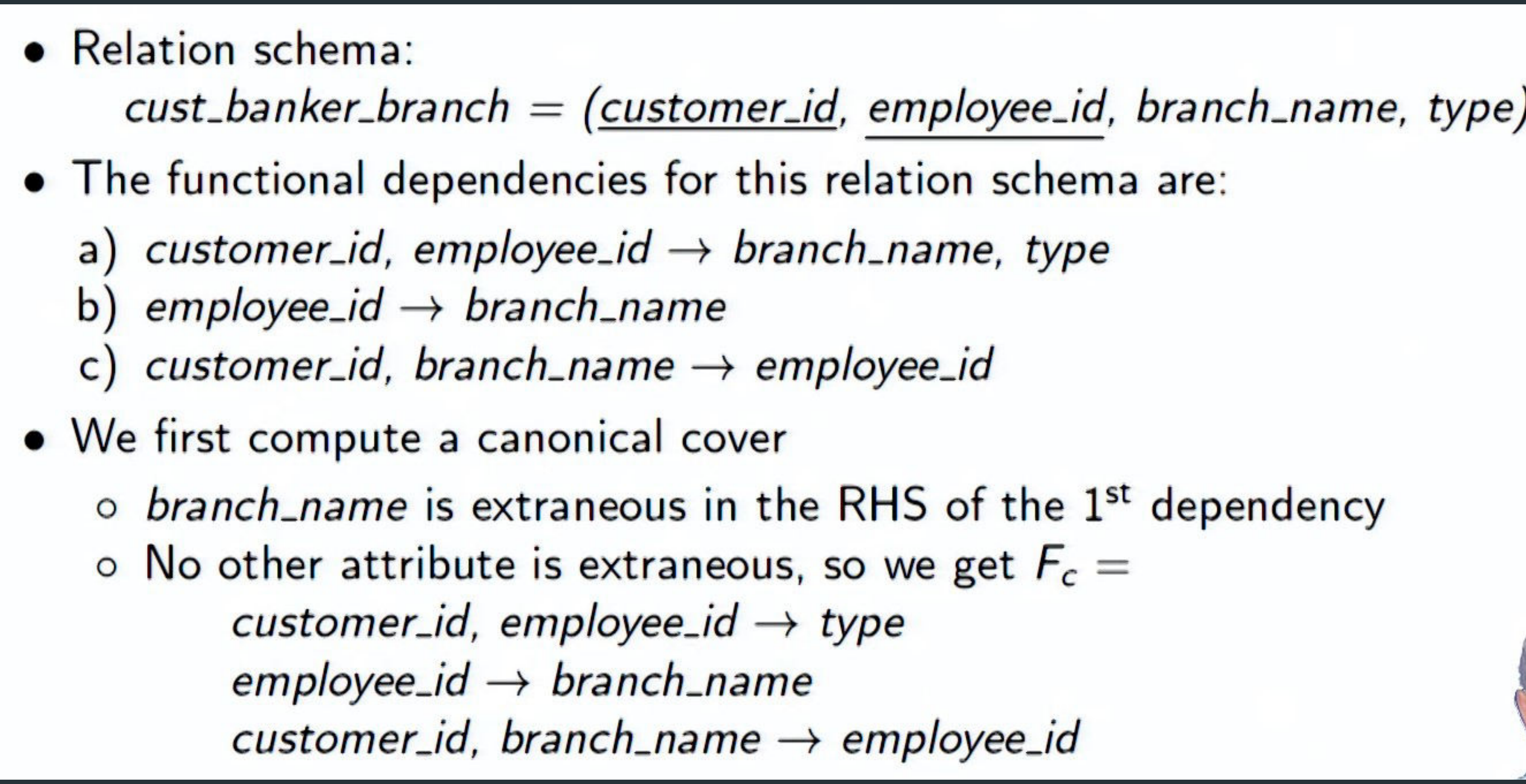
Algorithm

- Eliminate redundant FDs, resulting in a canonical cover F_c of F.
- Create a relation $R_i = XY$ for each FD $X \rightarrow Y$ in F_c.

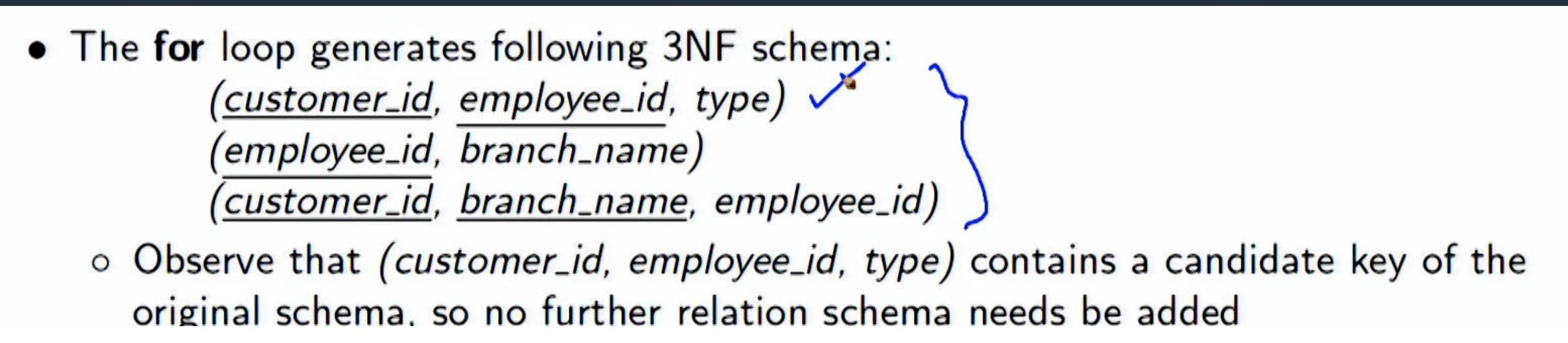
- If the key K of R does not occur in any relation R_i, create one more relation $R_j = K$.

```
Let Fc be the canonical cover of F;
i=0;
for each FD a->b in Fc do
    If none of the schema Rj, 1 <= j <= i contains ab
        then begin
            i = i + 1;
            Rj = ab;
        end
If none of the schema Rj, 1 <= j <= i contains a candidate key for R
     then begin
          i = i + 1;
          Rj = any candidate key for R;
     end
repeat
If any schema Rj is contained in another schema Rk
     then /* dleete Rj */
          Rj = R;
          i -= 1;
return (R1, R2, ..., Ri)
```

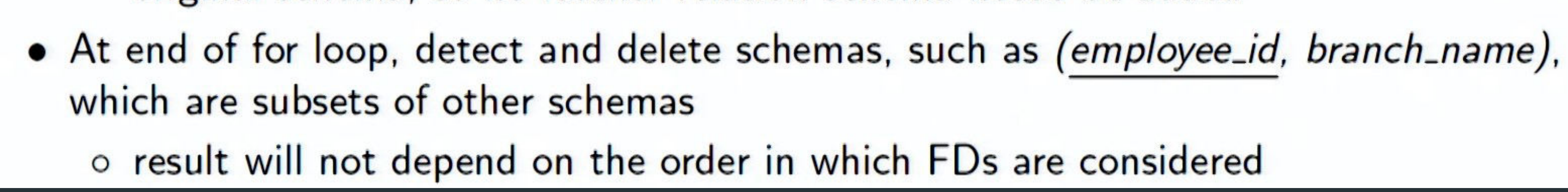
Example



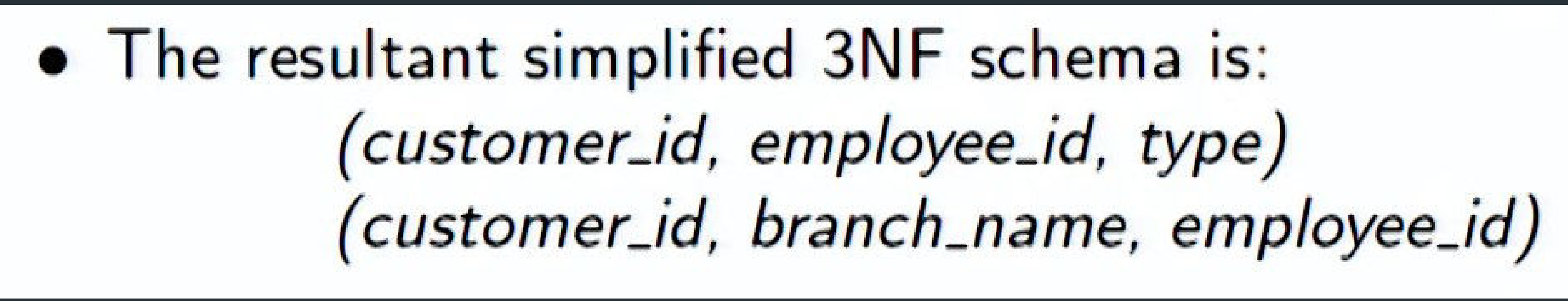
- Relation schema:
  cust_banker_branch = (*customer_id*, *employee_id*, *branch_name*, *type*)
- The functional dependencies for this relation schema are:
  a) *customer_id*, *employee_id* → *branch_name*, *type*
  b) *employee_id* → *branch_name*
  c) *customer_id*, *branch_name* → *employee_id*
- We first compute a canonical cover
  - *branch_name* is extraneous in the RHS of the 1st dependency
  - No other attribute is extraneous, so we get F_c =
    customer_id, *employee_id* → *type*
    employee_id → *branch_name*
    customer_id, *branch_name* → *employee_id*



- The **for** loop generates following 3NF schema:
  (*customer_id*, *employee_id*, *type*) ✓
  (*employee_id*, *branch_name*)
  (*customer_id*, *branch_name*, *employee_id*)
  - Observe that (*customer_id*, *employee_id*, *type*) contains a candidate key of the original schema, so no further relation schema needs be added



- At end of for loop, detect and delete schemas, such as (<u>*employee_id*</u>, *branch_name*), which are subsets of other schemas
  - result will not depend on the order in which FDs are considered



- The resultant simplified 3NF schema is:
  (*customer_id*, *employee_id*, *type*)
  (*customer_id*, *branch_name*, *employee_id*)

BCNF Decomposition

- A relation schema R is in BCNF with respect to a set F of Fds if for all FDs in F^+ of the form:
  - $\alpha \rightarrow \beta$ where $a \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:
  - $\alpha \rightarrow \beta$ is a trivial FD ($\beta \subseteq \alpha$)
  - α is a superkey for schema R.

Testing

Simplified test

- To check if a relation schema R is in BCNF, it suffices to check only the dependencies in the given set F for violation of BCNF, rather than checking all dependencies in F^+.
  - If none of the dependencies in F causes a violation of BCNF, then none of the dependencies in F^+ causes a violation of BCNF.

Testing for BCNF Decomposition

- To check if a relation R_i in a decomposition of R is in BCNF,
  - Either test R_i for BCNF with respect to the **restriction** of F to R_i (i.e. all FDs in F^+ that contain only attributes of R_i)
  - Or use the original set of dependencies F that hold on R with following test:
    - For every set of attributes $\alpha \subseteq R_i$, check that α^+ either includes no attribute of $R_i - \alpha$ or includes all attributes of R_i.
    - If the condition is violated by some $\alpha \rightarrow \beta$ in F, then the following dependency can be shown to hold on R_i and R_i violates BCNF:

$$\alpha \rightarrow (\alpha^+ - \alpha) \cap R_i$$

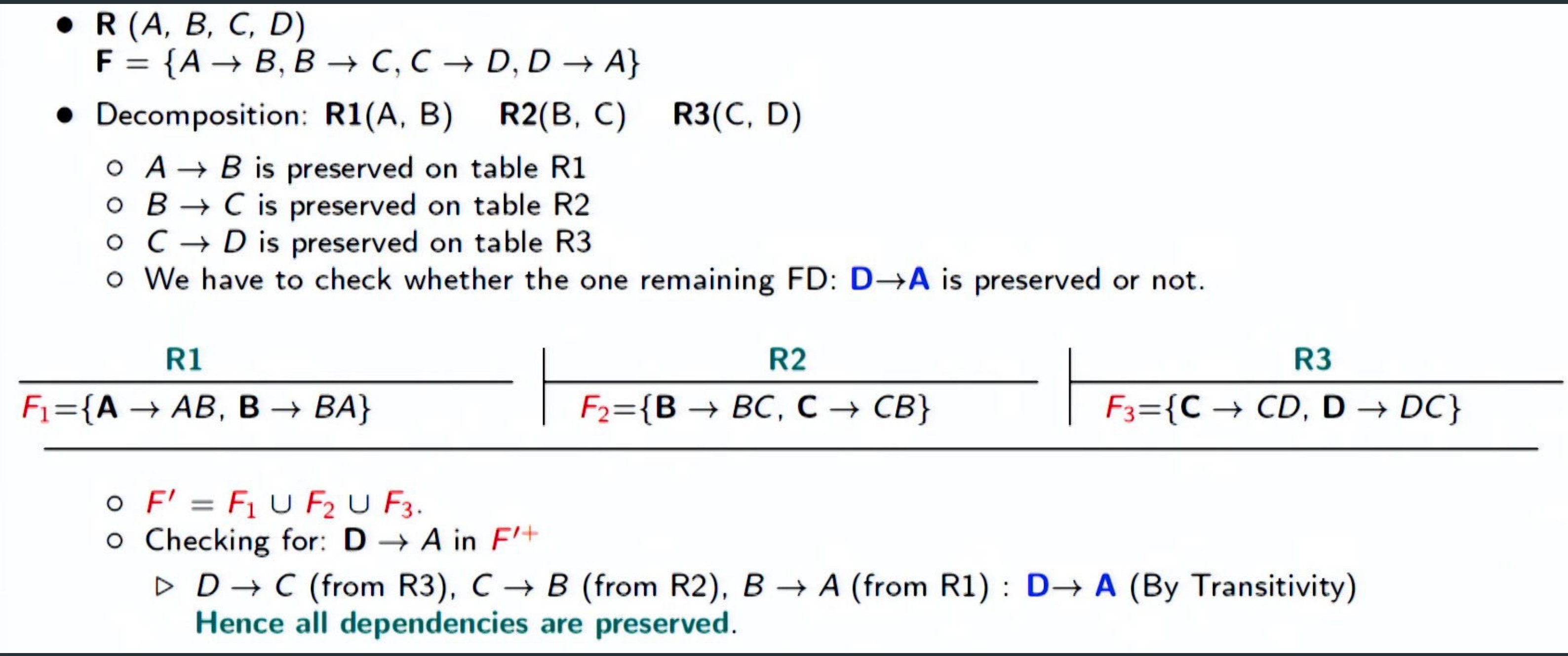
    - We use above dependency to decompose R_i.

Algorithm

1. For all dependencies $A \rightarrow B$ in F^+, check if A is a superkey by using attribute closure.

2. If not, then
    1. Choose a dependency in F^+ that violates BCNF, say $A \rightarrow B$.
    2. Create `R1 = AB`
    3. Create `R2 = (R - (B - A))`
3. Repeat For R1 and R2
    1. By defining $F1^+$ to be all dependencies in F that contain only attributes in R1, similarly $F2^+$.

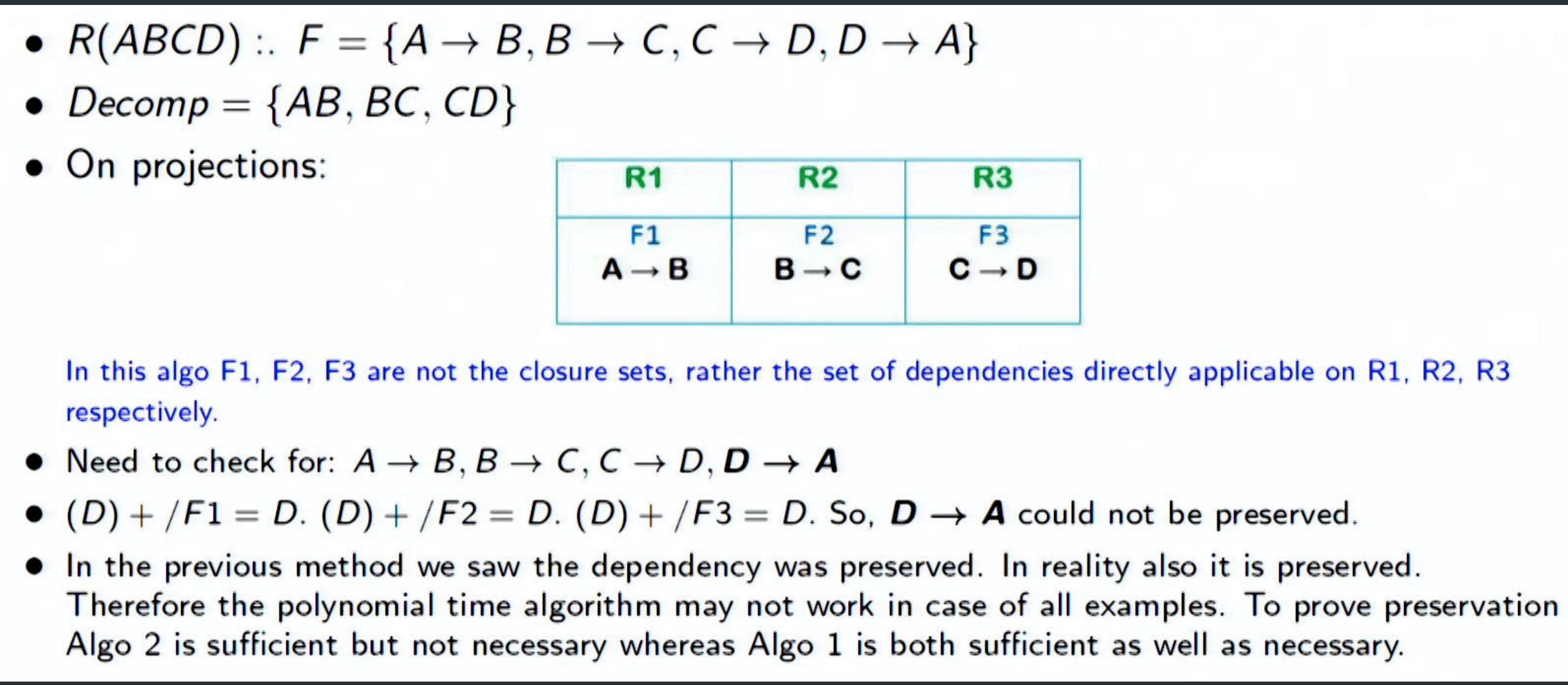
Example



- **R** (A, B, C, D)
  F $= \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$
- Decomposition: **R1**(A, B) **R2**(B, C) **R3**(C, D)
  - $A \rightarrow B$ is preserved on table R1
  - $B \rightarrow C$ is preserved on table R2
  - $C \rightarrow D$ is preserved on table R3
  - We have to check whether the one remaining FD: **D→A** is preserved or not.

R1	R2	R3
$F_1=\{\mathbf{A} \rightarrow AB, \mathbf{B} \rightarrow BA\}$	$F_2=\{\mathbf{B} \rightarrow BC, \mathbf{C} \rightarrow CB\}$	$F_3=\{\mathbf{C} \rightarrow CD, \mathbf{D} \rightarrow DC\}$

  - $F' = F_1 \cup F_2 \cup F_3$.
  - Checking for: $\mathbf{D} \rightarrow A$ in F'^+
    - ▷ $D \rightarrow C$ (from R3), $C \rightarrow B$ (from R2), $B \rightarrow A$ (from R1) : **D→ A** (By Transitivity)
      Hence all dependencies are preserved.



- $R(ABCD) :. F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$
- $Decomp = \{AB, BC, CD\}$
- On projections:

R1	R2	R3
F1 A→B	F2 B→C	F3 C→D

In this algo F1, F2, F3 are not the closure sets, rather the set of dependencies directly applicable on R1, R2, R3 respectively.

- Need to check for: $A \rightarrow B, B \rightarrow C, C \rightarrow D, \mathbf{D} \rightarrow \mathbf{A}$
- $(D)+/F1 = D$. $(D)+/F2 = D$. $(D)+/F3 = D$. So, $\mathbf{D} \rightarrow \mathbf{A}$ could not be preserved.
- In the previous method we saw the dependency was preserved. In reality also it is preserved. Therefore the polynomial time algorithm may not work in case of all examples. To prove preservation Algo 2 is sufficient but not necessary whereas Algo 1 is both sufficient as well as necessary.

3NF vs BCNF

3NF	BCNF
It concentrates on **Primary Key**.	It concentrates on **Candidate Key**.

3NF	BCNF
Redundancy is high as compared to BCNF.	0% redundancy.
It preserves all the dependencies.	It may not preserve all the dependencies.
A dependency $X \rightarrow Y$ is allowed in 3NF if X is a superkey or Y is a part of some key	A dependency $X \rightarrow Y$ is allowed in BCNF if X is a superkey.

L6.3: Relational Database Design/8: Case Study 🔗

L6.4: Relational Database Design/9: MVD & 4NF

MVD: Multivalued Dependency

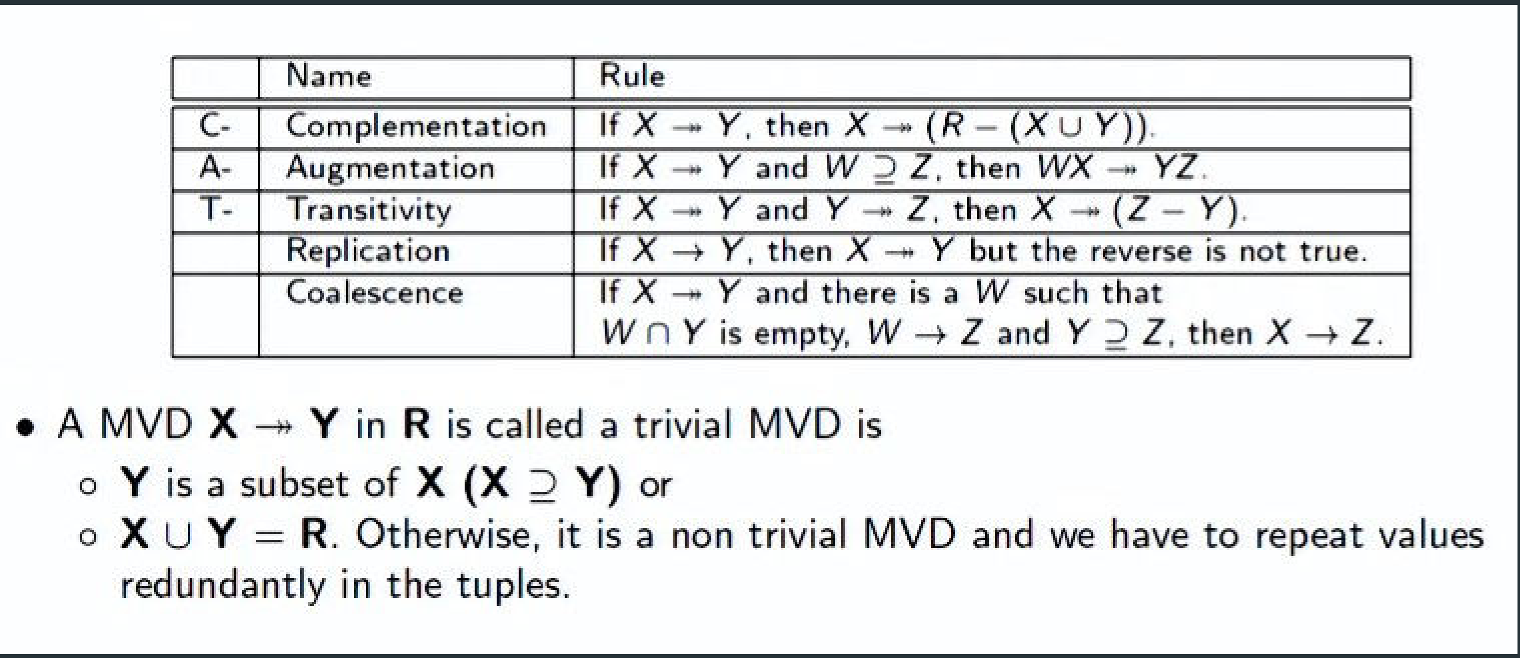
- If two or more independent relations are kept in a single relation, then MVD occurs.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.
- Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$. The **multivalued dependency** $\alpha \twoheadrightarrow \beta$ holds on R if, in any legal relation r(R), for all pairs of tuples t1 and t2 in r that agree on α, there exist tuples t3 and t4 in r such that:
  - t1[alpha] = t2[alpha] = t3[alpha] = t4[alpha]
  - t3[beta] = t1[beta] and t4[beta] = t2[beta]

Example:

- Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

BIKE_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	White

- Here columns `COLOR` and `MANUF_YEAR` are dependent on `BIKE_MODEL` and independent of each other.
- In this case, these two columns can be called as multivalued dependent on `BIKE_MODEL`. The representation of these dependencies is shown below:
    - `BIKE_MODEL ->> MANUF_YEAR`
    - `BIKE_MODEL ->> COLOR`



	Name	Rule
C-	Complementation	If $X \twoheadrightarrow Y$, then $X \twoheadrightarrow (R - (X \cup Y))$.
A-	Augmentation	If $X \twoheadrightarrow Y$ and $W \supseteq Z$, then $WX \twoheadrightarrow YZ$.
T-	Transitivity	If $X \twoheadrightarrow Y$ and $Y \twoheadrightarrow Z$, then $X \twoheadrightarrow (Z - Y)$.
	Replication	If $X \rightarrow Y$, then $X \twoheadrightarrow Y$ but the reverse is not true.
	Coalescence	If $X \twoheadrightarrow Y$ and there is a W such that $W \cap Y$ is empty, $W \rightarrow Z$ and $Y \supseteq Z$, then $X \rightarrow Z$.

- A MVD $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ in **R** is called a trivial MVD is
    - **Y** is a subset of **X** ($\mathbf{X} \supseteq \mathbf{Y}$) or
    - $\mathbf{X} \cup \mathbf{Y} = \mathbf{R}$. Otherwise, it is a non trivial MVD and we have to repeat values redundantly in the tuples.

Decomposition to 4NF

- A relation schema R is in 4NF with respect to a set D of functional and multivalued dependencies if for all MVD in D^+ of the form:
    - $\alpha \twoheadrightarrow \beta$ where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:
        - $\alpha \twoheadrightarrow \beta$ is a trivial MVD ($\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
        - α is a superkey for schema R.
- If a relation schema R is in 4NF, then it is also in BCNF.

Restriction of MVD

- The restriction of D to R_i is the set D_i consisting of
    - All FDs in D^+ that include only attributes of R_i
    - All multivalued dependencies of the form
      $\alpha \twoheadrightarrow (\beta \cap R_i)$
      where $\alpha \subseteq R_i$ and $\alpha \twoheadrightarrow \beta$ is in D^+.

Algorithm

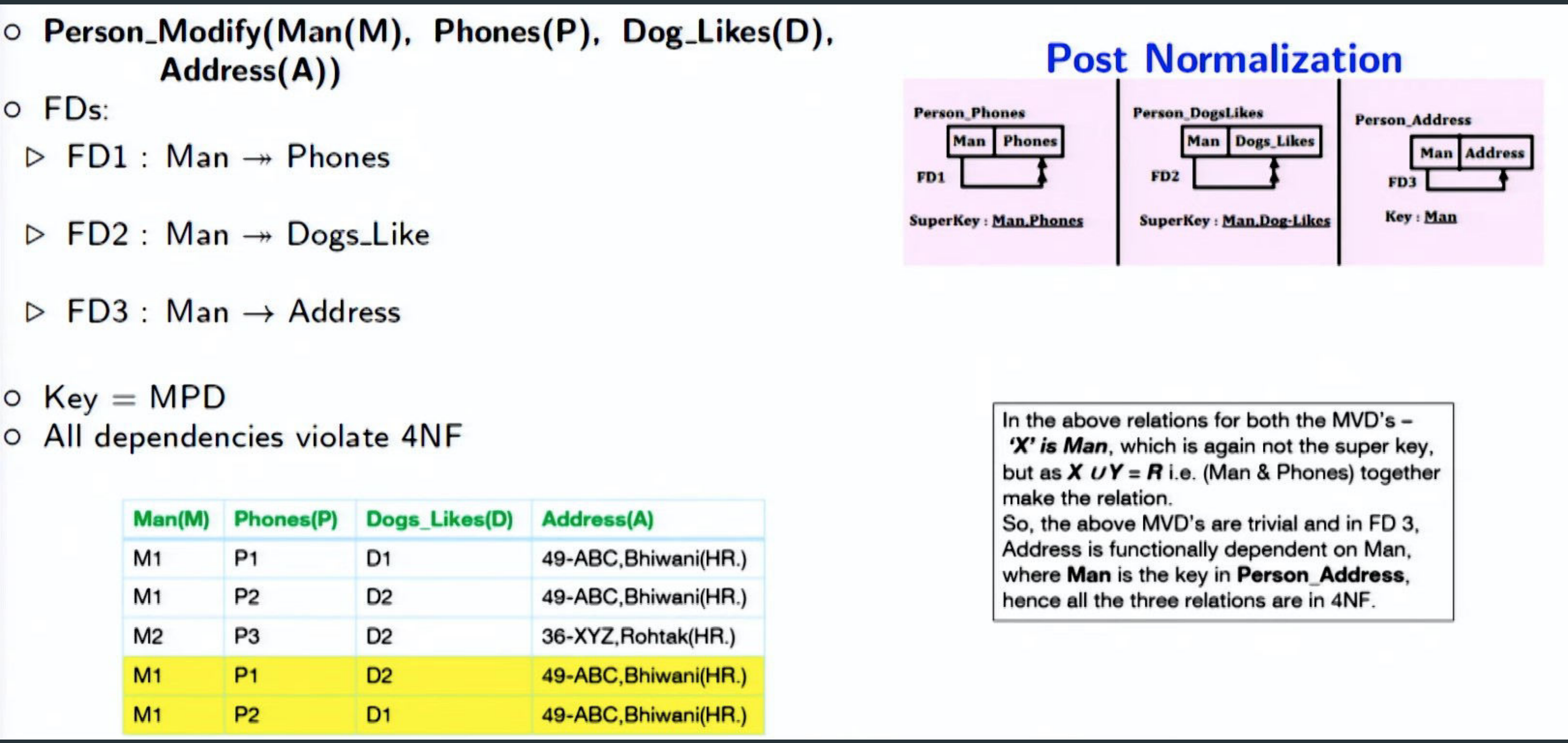
1. For all dependencies $A \twoheadrightarrow B$ in D^+, check if A is a superkey by using attribute closure.
2. If not, then:
    1. Choose a dependency in F^+ that violates 4NF, say $A \twoheadrightarrow B$.
    2. Create `R1 = AB`.
    3. Create `R2 = (R - (B - A))`.

3. Repeat for R1 and R2, by defining $D1^+$ to be all dependencies in D that contain only attributes in R1, similarly $D2^+$.

```
result = {R};
done = false;
compute D+;
Let Di denote the restriction of D+ to Ri;
while (not done)
     if (there is a schema Ri in result that is not in 4NF) then
          begin
               let A->>B be a nontrivial MVD that holds on Ri such that
                    A -> Ri is not on Di and a intersection B is not on Di;
               result = (result - Ri) U (Ri - (B - A)) U (A,B);
          end
     else
          done = true;
```

Example

1.



- Person_Modify(Man(M), Phones(P), Dog_Likes(D), Address(A))
- FDs:
  - ▷ FD1 : Man $\twoheadrightarrow$ Phones
  - ▷ FD2 : Man $\twoheadrightarrow$ Dogs_Like
  - ▷ FD3 : Man $\rightarrow$ Address
- Key = MPD
- All dependencies violate 4NF

Man(M)	Phones(P)	Dogs_Likes(D)	Address(A)
M1	P1	D1	49-ABC,Bhiwani(HR.)
M1	P2	D2	49-ABC,Bhiwani(HR.)
M2	P3	D2	36-XYZ,Rohtak(HR.)
M1	P1	D2	49-ABC,Bhiwani(HR.)
M1	P2	D1	49-ABC,Bhiwani(HR.)

Post Normalization

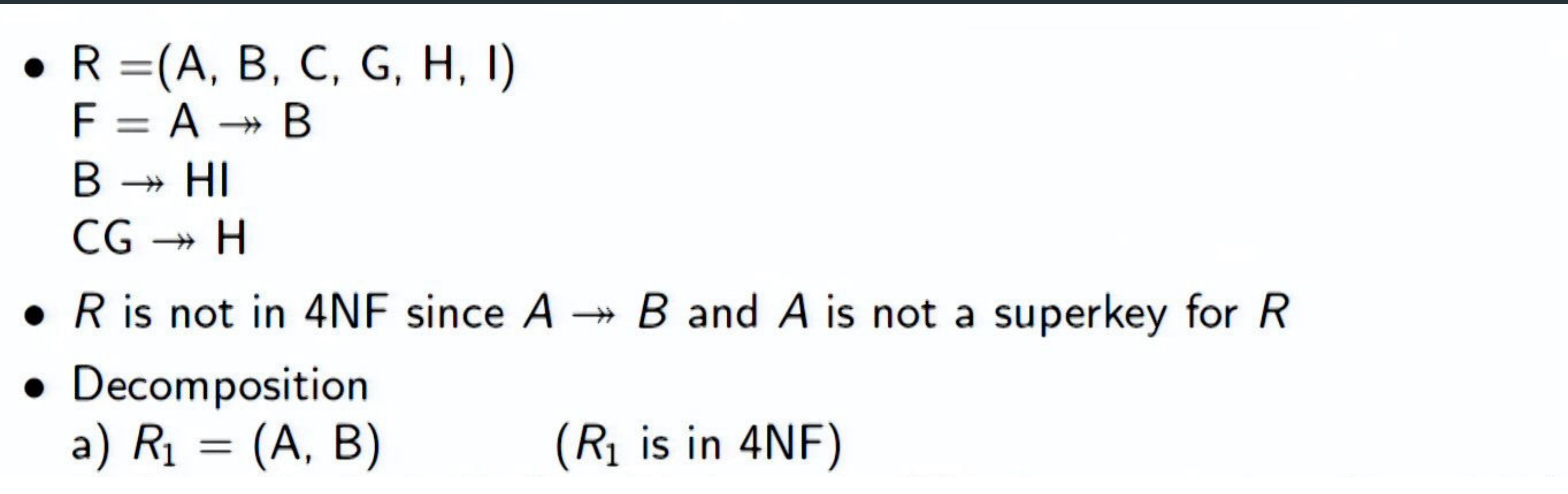
Person_Phones: | Man | Phones | — FD1 — SuperKey : Man,Phones

Person_DogsLikes: | Man | Dogs_Likes | — FD2 — SuperKey : Man,Dog-Likes

Person_Address: | Man | Address | — FD3 — Key : Man

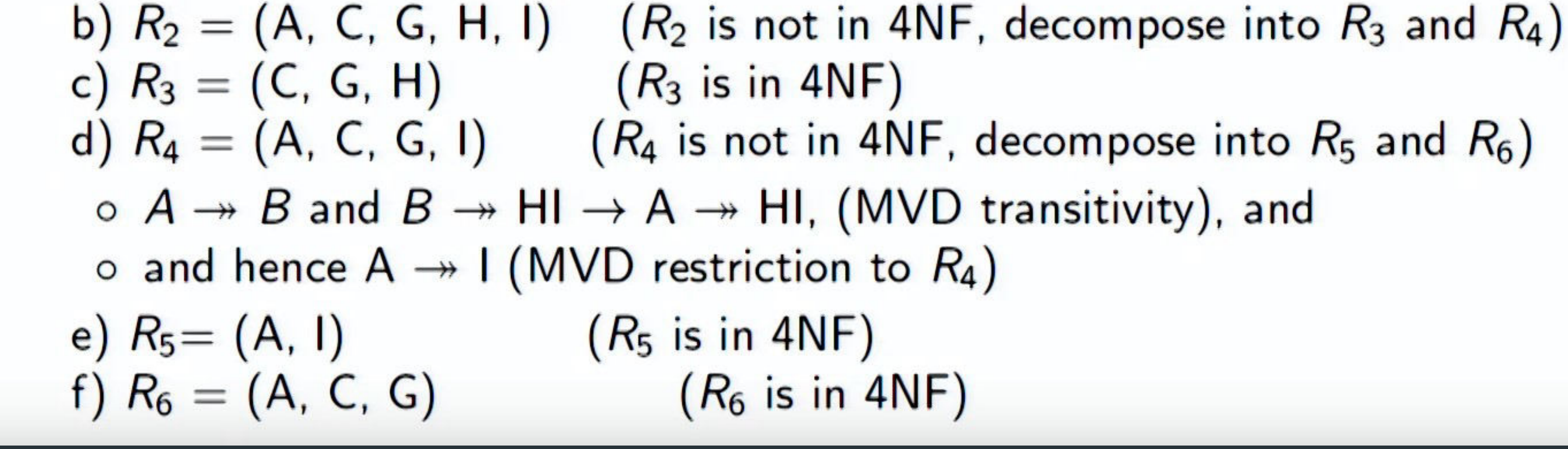
In the above relations for both the MVD's – *'X' is Man*, which is again not the super key, but as ***X ∪Y = R*** i.e. (Man & Phones) together make the relation.
So, the above MVD's are trivial and in FD 3, Address is functionally dependent on Man, where **Man** is the key in **Person_Address**, hence all the three relations are in 4NF.

2.



- R =(A, B, C, G, H, I)
  F = A $\twoheadrightarrow$ B
  B $\twoheadrightarrow$ HI
  CG $\twoheadrightarrow$ H
- R is not in 4NF since $A \twoheadrightarrow B$ and A is not a superkey for R
- Decomposition
  a) R_1 = (A, B)   (R_1 is in 4NF)

param302.bio.link



b) $R_2 = (A, C, G, H, I)$ (R_2 is not in 4NF, decompose into R_3 and R_4)
c) $R_3 = (C, G, H)$ (R_3 is in 4NF)
d) $R_4 = (A, C, G, I)$ (R_4 is not in 4NF, decompose into R_5 and R_6)

- $A \twoheadrightarrow B$ and $B \twoheadrightarrow HI \rightarrow A \twoheadrightarrow HI$, (MVD transitivity), and
- and hence $A \twoheadrightarrow I$ (MVD restriction to R_4)

e) $R_5 = (A, I)$ (R_5 is in 4NF)
f) $R_6 = (A, C, G)$ (R_6 is in 4NF)

L6.5: Relational Database Design/10: Design Summary & Temporal Data

Database Design Process

Design Goals

- Goal for a relational database design is:
    - BCNF / 4NF
    - Lossless Join
    - Dependency Preserving
- If we cannot achieve this, we accept one of:
    - Lack of dependency preserving
    - Redundancy due to use of 3NF

Normal Forms

- Further NFs
    - **Elementary Key Normal Form** (EKNF)
    - **Essential Tuple Normal Form** (ETNF)
    - **Join Dependencies And 5NF** (JD/5NF)
    - **Sixth Normal Form** (6NF)
    - **Domain/Key Normal Form** (DKNF)
- **Join dependencies** generalize multivalued dependencies which lead to *project-join normal form* (PJNF) or 5NF.
- A class of even more general constraints, leads to normal form called **domain-key normal form** (DKNF).

We have assumed schema R is given: - R could have been generated when converting E-R diagram to a set of tables. - R could have been a single relation containing all attributes that are of interest (universal relation). - Normalization breaks R into smaller relations - R could have been the reuslt of ad hoc design of relations, which we then test/convert to normal form.

ER Model and Normalization

- When an E-R diagram is carefully designed, identifying all entities correctly, the tables generated from the E-R diagram should not need further normalization.
- However, in real design, there can be functional dependencies from non-key attributes of an entity to other attributes of the entity.
- FDs from non-ky attributes of a relationship set possible, but rare, as most relationship sets are binary.

Denormalization for Performance

- We may want to use non-normalized schema for performance
- Example: Displaying `prereq` along with `course_id`, and `title` requires join of course with `prereq`
  - `Course(course_id, title, ...)`
  - `Preqrequisite(course_id, prereq)`
- **Alternative 1**: Use denormalized relation containing attributes of course as well as prereq with all above attributes.
  - `Course(course_id, title, preqreq)`
- **Alternative 2**: Use a materialized view defined as cross join of course and prereq.

Temporal Databases

- A temporal data have an association time interval during which the data are valid.
- A snapshot is the value of the data at a particular time.
- In practice, database designers may add start and end time attributes to relations.
- Example: `course(course_id, course_title)` is replaced by `course(course_id, course_title, start, end)`.
- Constraint: no two tuples can have overlapping valid times and are Hard to enforce efficiently.
- Foreign key references maybe to current version of data, or to data at a point in time.

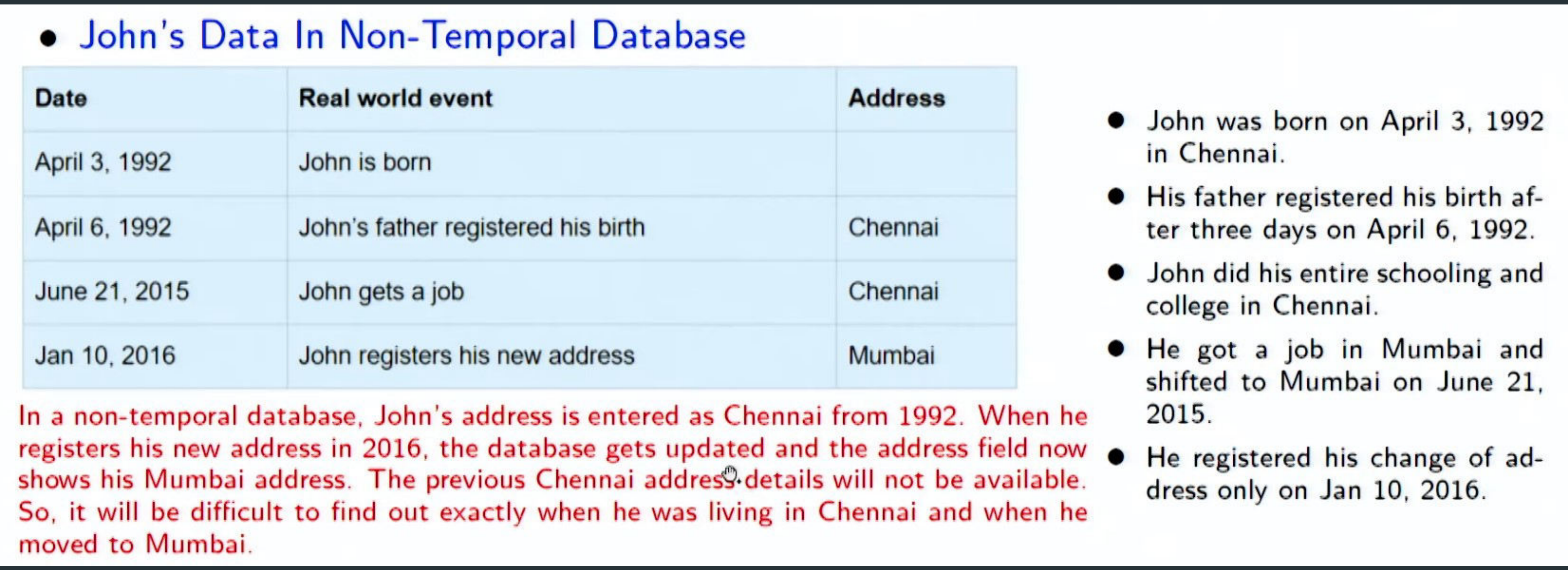
Temporal Data

- **Model of Temporal Domain** - 1-dimensional linearly ordered which maybe:
  - Discrete or dense
  - Bounded or unbounded
  - Single or multi-dimensional
  - Linear or non-linear
- **Temporal ER model** by adding valid time to *Attributes*, *Entities*, *Relationships*.

Modeling Temporal Data

- **Valid Time** - Time period during which a fact is true in the real world, provided by the application.
- **Transaction Time** - Time period during which a fact is stored in the database, based on transaction serialization order and is the timestamp automatically generated by the system.
- Temporal relation is one where each tuple has associated time; either valid time or transaction time or both.
  - **Uni-Temporal Relations**: Valid time or transaction time.
  - **Bi-Temporal Relations**: Both valid time and transaction time. It includes both start and end time.

Example:

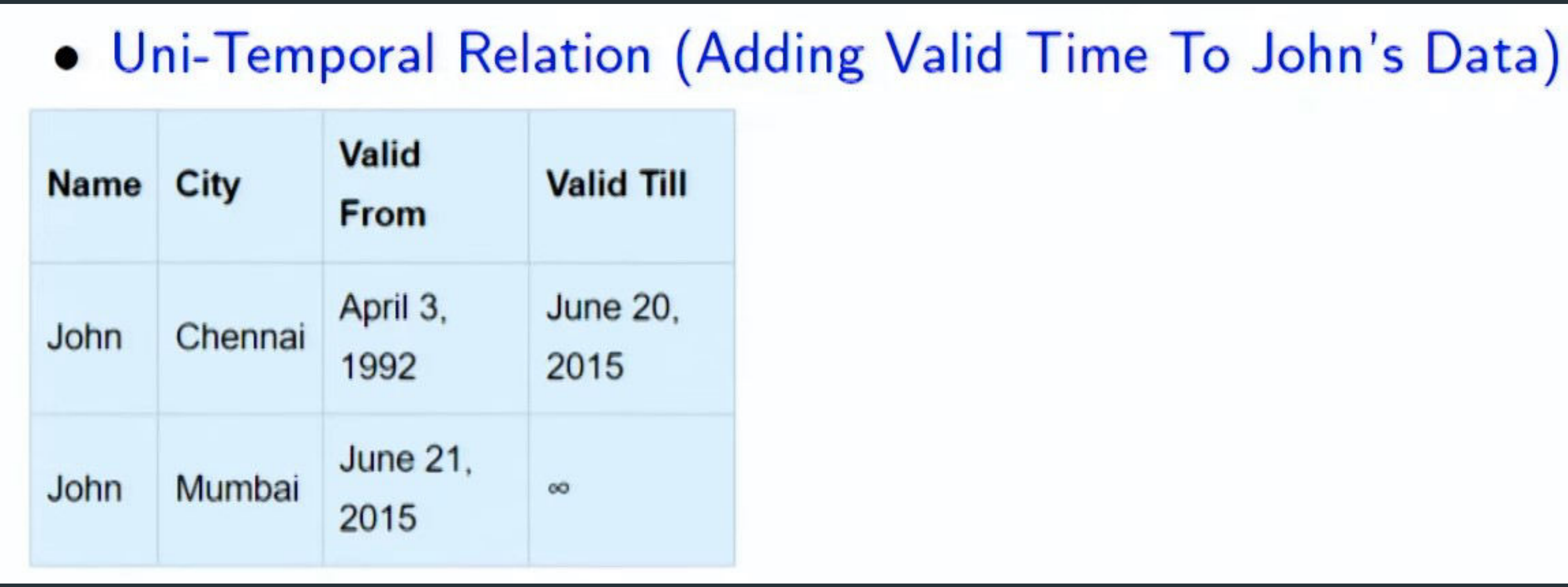


- John's Data In Non-Temporal Database

Date	Real world event	Address
April 3, 1992	John is born	
April 6, 1992	John's father registered his birth	Chennai
June 21, 2015	John gets a job	Chennai
Jan 10, 2016	John registers his new address	Mumbai

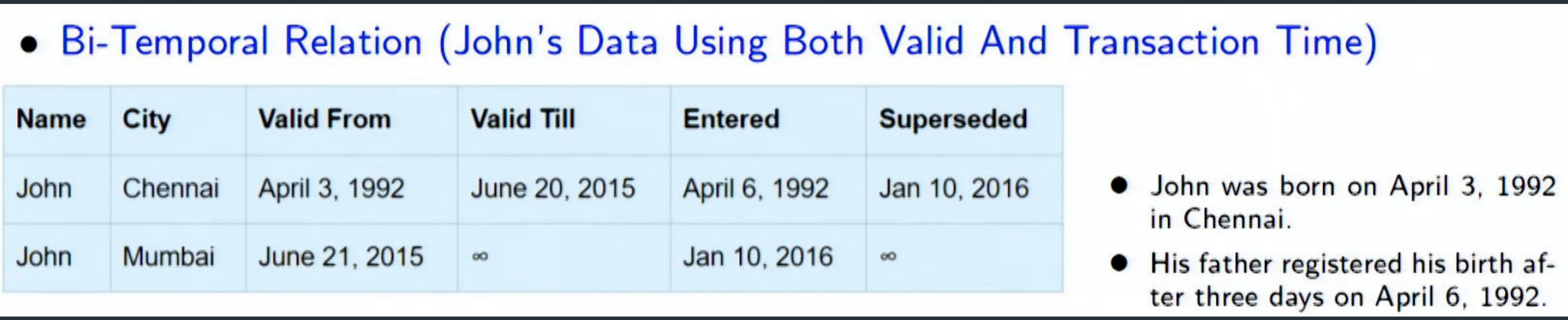
In a non-temporal database, John's address is entered as Chennai from 1992. When he registers his new address in 2016, the database gets updated and the address field now shows his Mumbai address. The previous Chennai address details will not be available. So, it will be difficult to find out exactly when he was living in Chennai and when he moved to Mumbai.

- John was born on April 3, 1992 in Chennai.
- His father registered his birth after three days on April 6, 1992.
- John did his entire schooling and college in Chennai.
- He got a job in Mumbai and shifted to Mumbai on June 21, 2015.
- He registered his change of address only on Jan 10, 2016.



- Uni-Temporal Relation (Adding Valid Time To John's Data)

Name	City	Valid From	Valid Till
John	Chennai	April 3, 1992	June 20, 2015
John	Mumbai	June 21, 2015	∞



- Bi-Temporal Relation (John's Data Using Both Valid And Transaction Time)

Name	City	Valid From	Valid Till	Entered	Superseded
John	Chennai	April 3, 1992	June 20, 2015	April 6, 1992	Jan 10, 2016
John	Mumbai	June 21, 2015	∞	Jan 10, 2016	∞

- John was born on April 3, 1992 in Chennai.
- His father registered his birth after three days on April 6, 1992.

Tutorials

6.1 Problem solving on normalization 🔗

6.2 Multivalued Dependency 🔗