Unit 3

Database Programming and Design

Functional Dependencies

* Need for functional dependencies - a ~~systema~~ framework for systematic design and optimization of relational schemas

→ They are a formal metric to specify the "goodness" of relational designs.

→ A set of attributes X functionally determines another set of attributes if the value of X determines a unique value for Y.

→ For any 2 tuples t_1 and t_2 in any relation instance r(R):

If $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$

In $X \rightarrow Y$

X: determinant Y: dependent

* Use of FDs : To describe the relational schema by specifying constraints on its attributes that must hold at all times.

* Constraints on Functional Dependencies

1. If a constraint on R states that, there cannot be more than one tuple with a given X- value in any instance => X is a candidate key of R.

2. If $X \rightarrow Y$ in R, it does not necessarily mean that $Y \rightarrow X$ in R.

3. An FD is a property of the relational schema R, not of a particular legal relation state r of R. (i.e cannot confirm an FD with just one legal dependency).

Example : Which FDs may exist in this relation?

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_4	d_3

Ans. $B \rightarrow C$

$C \rightarrow B$

$\{A, B\} \rightarrow D$

$\{C, D\} \rightarrow B$

$\{C, D\} \rightarrow A$

$\{A, B\} \rightarrow C$

* Time - Independent FDs : FDs that hold good for all possible additions to the relation, not just those that hold good for a particular time.

* Trivial FDs : when the right-side of the FD is a subset of the left

i.e $\{A, B\} \rightarrow A$

* Closure of FDs : The set of all FDs that are implied by a given set S of FDs is called the closure of S, written as S^+.

(another similar def on pg. 4)

* Armstrong's Axioms

→ A set of inference rules by which new FDs can be inferred from the given ones.

(i) Reflexivity : $B \subseteq A \Rightarrow A \rightarrow B$

(ii) Augmentation: $A \rightarrow B \rightarrow AC \rightarrow BC$

(iii) Transitivity : $A \rightarrow B$, $B \rightarrow C \Rightarrow A \rightarrow C$

(iv) Self-Determination : $A \rightarrow A$

(v) Decomposition : $A \rightarrow BC \Rightarrow A \rightarrow B$ and $A \rightarrow C$

(vi) Union : $A \rightarrow B$ and $A \rightarrow C \Rightarrow A \rightarrow BC$

(vii) Composition : $A \rightarrow B$ and $C \rightarrow D \Rightarrow AC \rightarrow BD$

set S>

Example: Given the relvar R, wol attributes A, B, C, D, E, F and the FDs :

$A \rightarrow BC$

$B \rightarrow E$

$CD \rightarrow EF$

Show that the FD $AD \rightarrow F$ holds for R, and is thus a member of the closure of the given set.

Ans

$A \rightarrow BC$

$A \rightarrow C$ & $A \rightarrow B$ (decomposition)

$AD \rightarrow CD$ (augmentation)

$CD \rightarrow EF$ (given)

$AD \rightarrow EF$ (transitivity)

$AD \rightarrow E$ & $AD \rightarrow F$ (decomposition)

* Closure Set of Attributes

Given a relval R

(i) a set Z of attributes of R

(ii) a set S of FDs that hold for R

The set of all attributes of R that are functionally dependent on Z is called the closure of Z+ of Z under S.

Procedure to find closure

CLOSURE [Z, S] : = Z

do forever :

for each FD $X \rightarrow Y$ in S

do :

if $X \subseteq$ closure [Z, S]

then closure [Z, S] : = closure [Z, S] $\cup$ Y

end

If CLOSURE [Z, S] did not change on this iteration then leave the loop

end;

Identifying the superkey : If the closure contains all the attributes of the relation R then the attributes in that set form the superkey.

Identifying the candidate key : Remove the attributes and check for closure individually. If the closure of the individual attributes does not contain all the attributes, then the superkey is the candidate key (i.e. it is an irreducible superkey)

* Equivalent Sets of FDs

Q sets of FDs F and G are equivalent if

- every FD in F can be inferred from G
- every FD in G can be inferred from F.

i.e $F^+ = G^+$

* Covers

F is said to cover G if every FD in G can be inferred from F

(i.e if G^+ is a subset of F^+)

F and G are equivalent if F covers G and G covers F.

Example: Given $A \rightarrow BC$

$E \rightarrow CF$

$B \rightarrow E$

$CD \rightarrow EF$

Compute the closure of $\{A, B\}^+$

$\{A, B\} \subseteq \{A, B\} \Rightarrow \{A, B\}$

Itr1

$A \subseteq \{A, B\} \Rightarrow \{A, B, C\}$

$E \not\subseteq \{A, B, C\} \Rightarrow$ no change

$B \subseteq \{A, B, C\} \Rightarrow \{A, B, C, E\}$

$CD \not\subseteq \{A, B, C, E\} \Rightarrow$ no change

Itr2

$A \subseteq \{A, B, C, E\} \Rightarrow \{A, B, C, E\}$

$E \subseteq \{A, B, C, E\} \Rightarrow \{A, B, C, E\}$ (same)

$B \subseteq \{A, B, C, E\} \Rightarrow \{A, B, C, E\}$ (same)

$CD \not\subseteq \{A, B, C, E\} \Rightarrow$ no change

Itr3 $A \subseteq \{A,B,C,E,F\} \Rightarrow \{A,B,C,E,F\}$

$E \subseteq \{A,B,C,E,F\} \Rightarrow \{A,B,C,E,F\}$

$B \subseteq \{A,B,C,E,F\} \Rightarrow \{A,B,C,E,F\}$

$CD \not\subseteq \{A,B,C,E,F\}$

no change at the end of Itr2 & Itr3

$\Rightarrow \{A,B\}^+ = \{A,B,C,E,F\}$

Example2 Given the FDs $A \rightarrow BC$

$E \rightarrow CF$

$B \rightarrow E$

and $CD \rightarrow EF$

Find the closure of $\{A,D\}^+$

Itr1 $\{A,D\} \subseteq \{A,D\} \Rightarrow \{A,D\}$

$A \subseteq \{A,D\} \Rightarrow \{A,B,C,D\}$

$E \not\subseteq \{A,B,C,D\} \Rightarrow$ no change

$B \subseteq \{A,B,C,D\} \Rightarrow \{A,B,C,D,E\}$

$CD \subseteq \{A,B,C,D\} \Rightarrow \{A,B,C,D,E,F\}$

The closure of $\{A,D\}^+$ has all the attributes $\{A,B,C,D,E,F\} \Rightarrow \{A,D\}$ is a superkey

Example3 Consider a relation R, with attributes A, B, C, D, E, F, G, H, I, J. Find the candidate key

$AB \rightarrow C$

$BD \rightarrow EF$ $\quad H \rightarrow J$

$AD \rightarrow GH$

$A \rightarrow I$

Ans: Find closure of each LHS. Check if the closure has all the attributes. Otherwise, start merging diff. combinations of attributes.

closure of $\{A, B\}^+ = \{A, B, C, I\}$

closure of $\{B, D\}^+ = \{B, D, E, F\}$

closure of $\{A, D\}^+ = \{A, D, G, H, I, J\}$

closure of $\{A\}^+ = \{A, I\}$

closure of $\{H\}^+ = \{H, J\}$

closure of $\{A, B, D\} = \{A, B, C, D, E, F, G, H, I, J\}$

$\{A, B, D\}$ is the candidate key.

Example 4: The relation R has attributes A, B, C, D, E, F.

The FDs are:

$A \rightarrow FC$

$C \rightarrow D$

$B \rightarrow E$

Find the candidate key of the relation

$\{A\}^+ = \{A, F, C\}$

$\{C\}^+ = \{C, D\}$

$\{B\}^+ = \{B, E\}$

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

$\{A, B\}$ is the candidate key

Example : Carsale has attributes

carno, car id, datesold, salesmanid, discountamt, commission %.

The functional dependencies are:

car-id → datesold

datesold → discount amt

salesmanid → comm %.

{carid}+ = {carid, datesold, discountamt}

{datesold}+ = {datesold, discountamt}

{salesmanid}+ = {salesmanid, comm %.}

{carid, salesmanid}+ = {carid, datesold, discountamt, salesmanid, comm %.}

* Irreducible Sets of Functional Dependencies (Minimal Cover)

A set of FDs is said to be irreducible if and only if it satisfies the following 3 properties:

1. The right side of every FD in S involves just one attribute.
2. No FD in S can be discarded in S w/o changing the closure set
3. The left side of every FD in S is irreducible - no attribute can be discarded from the left side w/o changing the closure set

irreducible sets
(also called minimal cover)

Procedure to find Irreducible Sets

make sure RHS has only 1 attributed

(i) compute closure of LHS of each FD

(ii) remove the FD, and find closure again

(iii) If the set of attributes in (i) & (ii) are the same, the FD is redundant, and can be eliminated.

Example1 : Reduce the following sets of FDs.

$X \rightarrow W$

$\{W, Z\} \rightarrow \{X, Y\}$

$Y \rightarrow \{W, X, Z\}$

Solution The FDs are

$X \rightarrow W$

$WZ \rightarrow X$

$WZ \rightarrow Y$

$Y \rightarrow W$

$Y \rightarrow X$

$Y \rightarrow Z$

(i) $X \rightarrow W$

closure of $\{X\}^+ = \{X, W, \}$

remove $X \rightarrow W$

new closure $= \{X\}$

(ii) $WZ \rightarrow X$

$\{WZ\}^+ = \{W, Z, X, Y\}$

remove $WZ \rightarrow X$

new closure $\{W, Z\}^+ = \{W, Z, Y, Z\}$

$X \rightarrow W$

$\boxed{WZ \rightarrow X}$

$WZ \rightarrow Y$

$Y \rightarrow W$

$Y \rightarrow X$

$Y \rightarrow Z$.

diff ⇒ cannot ignore

same ⇒

remove $WZ \rightarrow X$

(iii) $wz \to y$

closure of $\{wz\}^+ = \{w, z, y, z\}$

remove $\{wz \to y$

new closure = $\{w, z\}$

} diff. cannot ignore

$wz \to x$

$x \to w$

$wz \to y$

$y \to w$

$y \to x$

$y \to z$

(iv) $y \to w$

closure of $\{y\}^+ = \{y, w, x, z\}$

remove $y \to w$

new closure = $\{y, x, z, w\}$

} same ⇒ remove

(v) $y \to x$

closure of $\{y\}^+ = \{y, x, z, w\}$

remove $y \to x$

new closure: $\{y, z\}$

} diff. cannot remove

(vi) $y \to z$

closure of $\{y\}^+ = \{y, z, x, w\}$

remove $y \to z$

new closure = $\{y, x, w\}$

} diff, cannot remove

∴ The reduced set of FDs are:

$x \to w$

$wz \to y$

$y \to x$

$y \to z$

⇒

$x \to w$

$wz \to y$

$y \to xz$

Example 2: Find the irreducible sets of FDs.

1. $A \rightarrow BC$
2. $B \rightarrow C$
3. $A \rightarrow B$
4. $AB \rightarrow C$
5. $AC \rightarrow D$

Solution: The simplified set of FDs is:

$A \rightarrow B$
$A \rightarrow C$
$B \rightarrow C$
$AB \rightarrow C$
$AC \rightarrow D$

$A \rightarrow B$
$\boxed{A \rightarrow C}$
$B \rightarrow C$
$AB \rightarrow C$
$AC \rightarrow D$

(i) $A \rightarrow B$

closure of $\{A\}^+ = \{A, B, C, D\}$

remove $A \rightarrow B$

new closure $= \{A, C, D\}$

$\Rightarrow$ diff. cannot remove

(ii) $A \rightarrow C$

closure of $\{A\}^+ = \{A, C, D, B\}$

remove $A \rightarrow C$

new closure $= \{A, B, C, D\}$

same $\Rightarrow$ remove

(iii) $B \rightarrow C$

closure of $\{B\}^+ = \{B, C, \}$

remove $B \rightarrow C$

new closure $= \{B\}$

diff $\Rightarrow$ cannot remove

(iv) $AB \rightarrow C$

closure of $\{AB\}^+ = \{A, B, C, D\}$

remove $AB \rightarrow C$

new closure = $\{A, B, C, D\}$

$A \rightarrow B$

$A \rightarrow C$

$B \rightarrow C$

$AB \rightarrow C$

$AC \rightarrow D$

same → can be removed

(v) $AC \rightarrow D$

closure of $\{AC\}^+ = \{A, C, D, B\}$

remove $AC \rightarrow D$

new closure = $\{A, C, B\}$

diff → cannot remove

Simplified set of FDs

$A \rightarrow B$

$B \rightarrow C$

$AC \rightarrow D$

Normalization

* Informal Design Guidelines for Relational Databases

The quality of the relation schema design may be decided by:

(i) making sure the semantics of the attributes is clear in the schema

(ii) reducing redundant information in tuples

(iii) reducing null values in tuples

(iv) disallowing the possibility of generating spurious tuples

* Informal Guidelines

GUIDELINE 1

(i) Each tuple in a relation should represent one entity or relationship (attributes of different entities, say employees, departments, projects should not be mixed in the same relation)

GUIDELINE 2: Schema should be designed such that there are no update anomalies (insert, deletion or updation)

* Update Anomalies

A. Insertion Anomaly: If a tuple is inserted in the referencing relation, and the referencing attribute value is not present in the referenced attribute. (integrity constraint violation)

B. Update Anomalies: when updates are not correctly reflected across all referencing tables

C. Delete Anomalies: when deletion does not lead to the deletion of related records in referenced tables

GUIDELINE 3: Relations should be designed such that their tuples will have as few NULL values as possible. Attributes that are NULL frequently would be placed in separate relations, and not in the base relation.

Problems with NULL values

(i) wastage of storage space
(ii) difficult to understand semantics
(iii) difficult to perform select, agg and join operations
(iv) results become unpredictable

GUIDELINE 4: The relations should be designed so that they can be joined using primary and foreign key pairs. No spurious tuples should be generated by doing a natural join of any relations

* Spurious Tuples: Bad database design may result in spurious results for certain JOIN operations, leads to invalid / spurious tuples

* Normalization: The process of decomposing unsatisfactory relations by breaking up their attributes into smaller relations.

* Normal form: Conditions using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

(i) 1NF
(ii) 2NF
(iii) 3NF
(iv) BCNF (Boyce Codd Normal Form)

* Additional Properties for good database design

(1) Lossless Join - (Non-additive join property) - guarantees that spurious tuples do not occur when joining the tables created after decomposition. All original tuples must be obtained again

(2) Dependency Preservation: each FD should be preserved in the resulting relation

(1) is very important - cannot be compromised
(2) less stringent - may be sacrificed.

* First Normal Form (1NF)

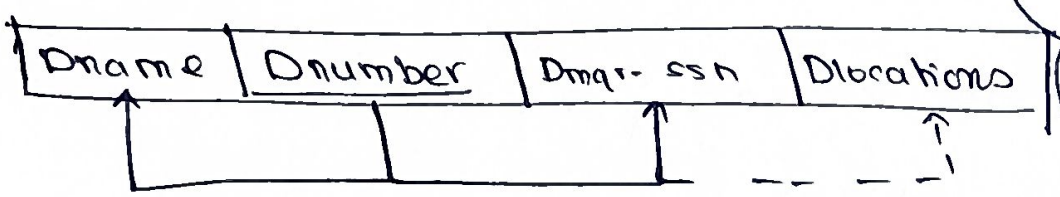
→ domain of an attribute must have only atomic values

→ disallows composite attributes, multivalued attributes & nested relations.

Example 1 : Consider the relation

Department

Department



Dlocation is multivalued ⇒ not in 1NF.

Solution

(i) Decompose as:

Dname	Dnumber	Dmgr-ssn

and

Dnumber	Dlocation

(no redundancy)

(ii) Expand the key such that there is a separate tuple for locations

Dname	Dnumber	Dmgr-ssn	Dlocation
R	5	33344555	Bellaire
R	5	33344555	Sugarland

(introduces redundancies)

Recap of Key Terminology

Superkey : subset of R s.t no 2 tuples have $t_1[S] = t_2[S]$

Key : a key K is a superkey w/ the additional property that removal of any attr. from K will cause K not to be a superkey anymore

A key is a minimal superkey.

→ If there is more than one key ⇒ candidate key

→ One of the candidate keys = primary key, others = secondary keys

prime attribute → member of a candidate key

non-prime attribute → not a member of any candidate key

If the max. no. of locations is known, introduce that many Dlocation fields (introduces null values)

Dname	Dnumber	Dmgr-ssn	Dloc1	Dloc2	Dloc3

* Second Normal Form - 2NF

A relation schema R is in second normal form (2NF) if every non-prime attribute is fully functionaly dependent on the primary key of R.

For a nested case: Example 2

EMP_PROJ

		Projs	
ssn	ename	Pno	Hrs

decompose as

ssn	ename

&

ssn	pnumber	hours

A. Full Functional Dependency:

An FD $X \rightarrow Y$ is a full FD if the removal of any attribute A from X means that the FD does not hold any more.

B. Partial Functional Dependency

An FD $X \rightarrow Y$ is a partial FD if the removal of any attribute A from X means that the FD still holds.

Example 1: Check if EMP_PROJ is in 2NF

Ssn	Pnumber	Hours	Ename	Pname	Plocation

FD1

FD2

FD3

Working Rule

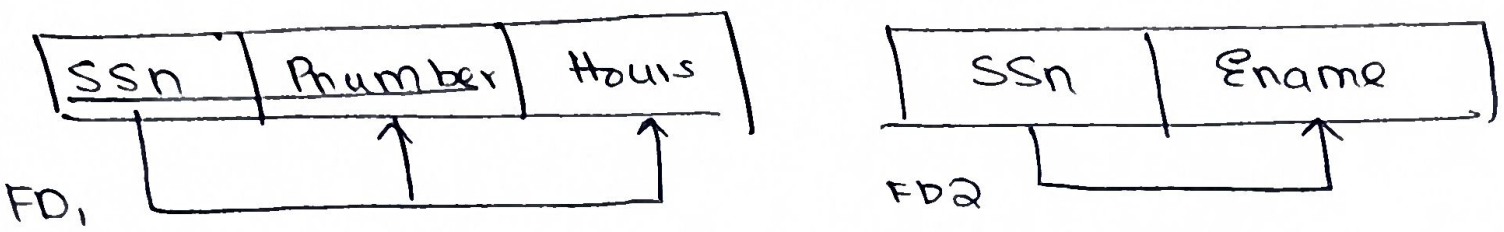
Check if a part of candidate key determines some non-prime attribute ⇒ partial FD

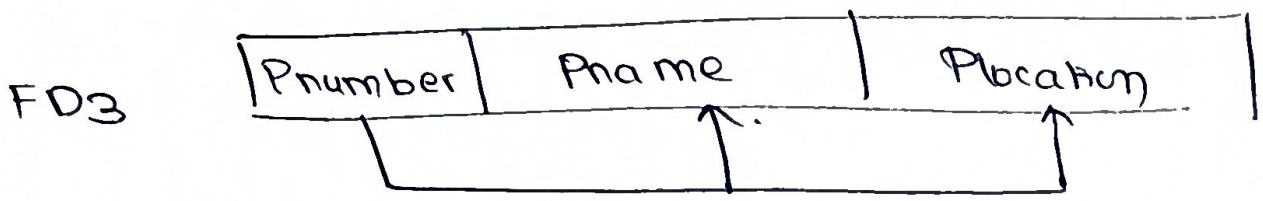
FD1: Fully Functional Dependency

FD2: Partial dependency (Ename is determined only by SSN)

FD3: Partial FD (Pname & Plocation determined only by Pnumber)

Reduction





* Third Normal Form

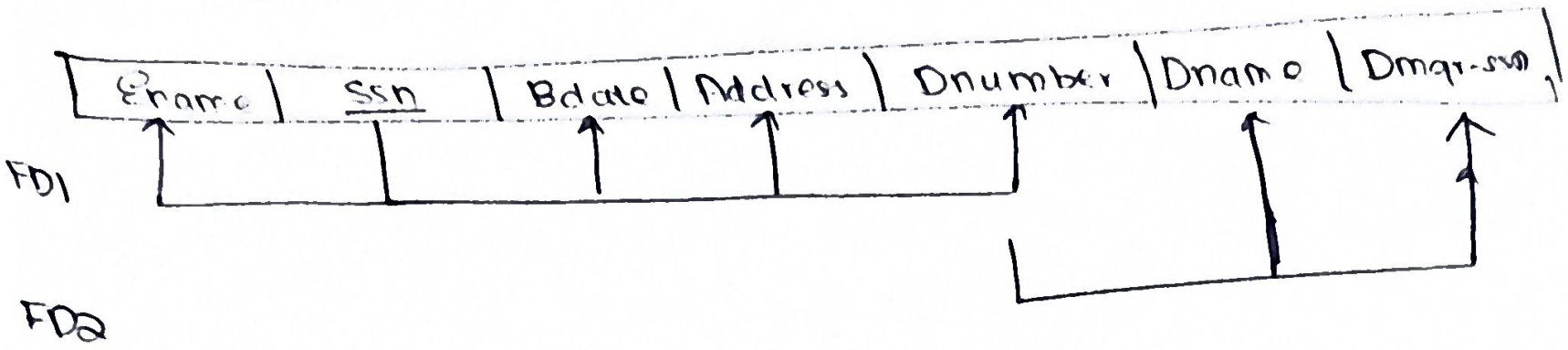
→ Based on the concept of Transitive Dependency.

Transitive dependency: An FD $X \rightarrow Y$ in R is a transitive dependency if there is a set of attribute Y, that is neither a candidate, nor a subset of any key of R, and $X \rightarrow Y$ and $Y \rightarrow Z$ holds.

→ A relational schema R is in 3NF if it is in 2NF and no non-prime attribute is transitively dependent on the primary key.

→ In $X \rightarrow Y$ and $Y \rightarrow Z$, if Y is a candidate key, then there's no problem with the transitive dependency.

Example: Check if EMP-DEPT is in 3NF



Solution Ssn → Dnumber

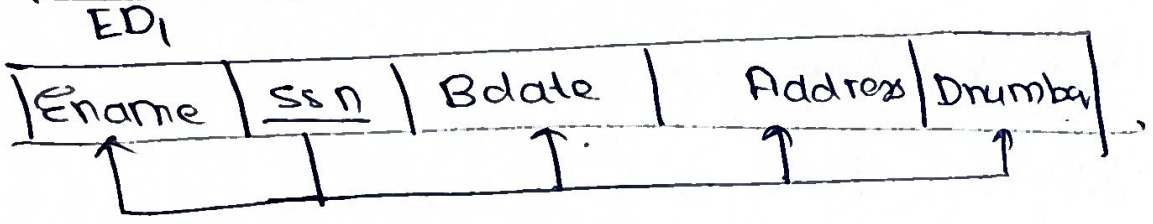
Dnumber → Dname, Dmgrssn

Dnumber is not a candidate key

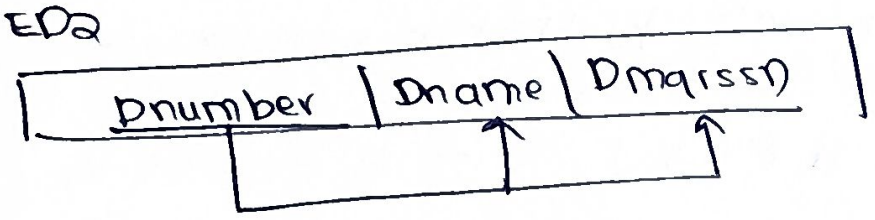
⇒ Not in 3NF

reduction

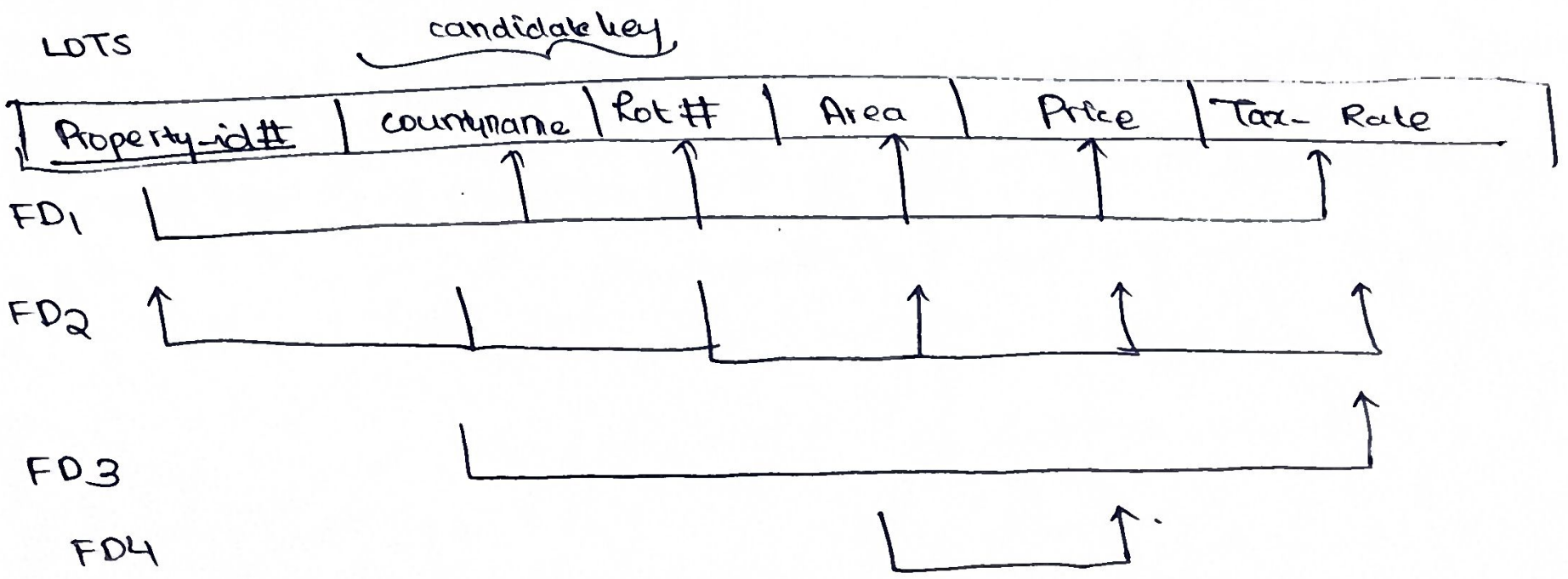
ED1



ED2



Question1: Normalize the given set of functional dependencies



1NF : no-multivalued attributes ⇒ 1NF is satisfied

2NF:

FD1 : fully functional dependency

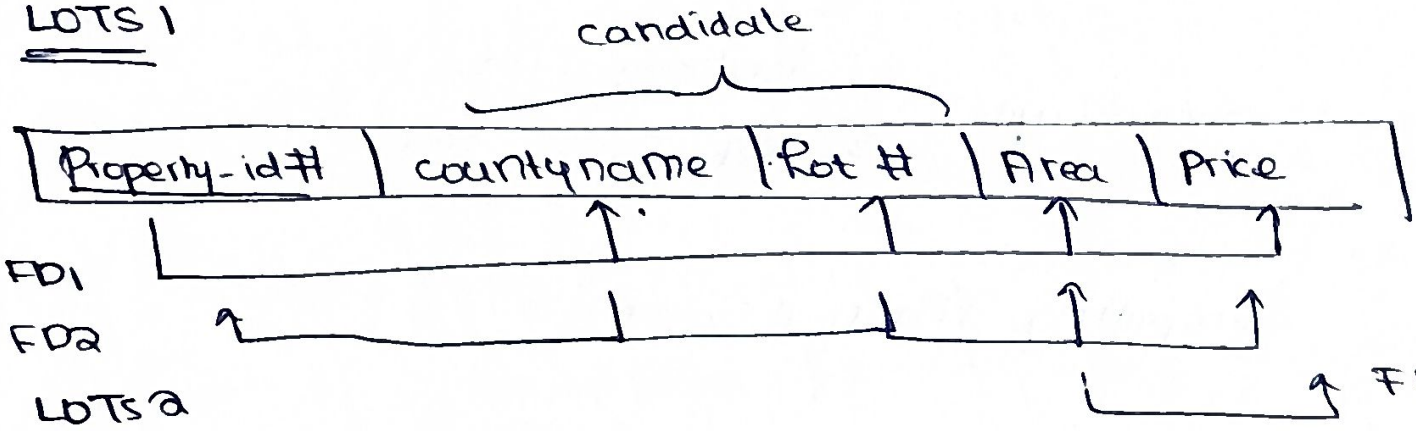
FD2 : fully functional dependency

FD3 : partial FD (only county name determines Tax-Rate)

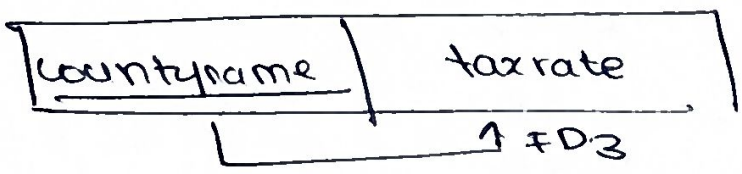
FD4 : doesn't matter, area not a prime key / candidate key

Reduction

LOTS 1



LOTS 2



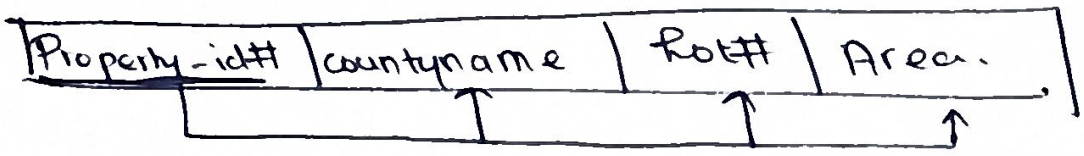
3NF Property-id# → Area

Area → Price

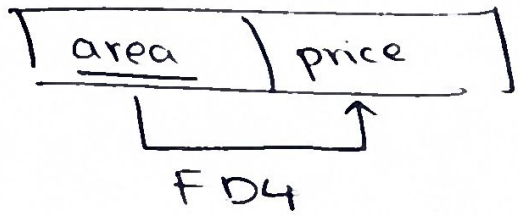
Area is not a candidate key

⇒ not in 3NF

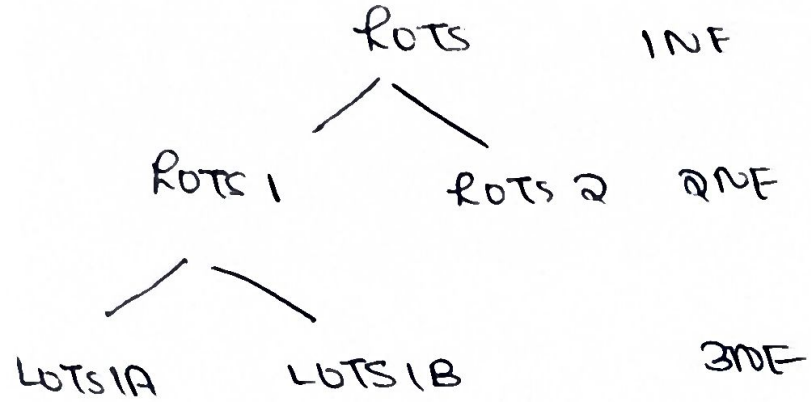
LOTS 1A



LOTS 1B



Reduced Table Structure



Question 2 : Given a relation $R = \{A, B, C, D, E, F\}$ and the functional dependencies:

$A \rightarrow C$ (not reduced here)

$A \rightarrow F$

$A \rightarrow D$

$C \rightarrow D$

$B \rightarrow E$,

normalize the relational scheme, upto 3NF

Steps

1. Find irreducible FDs (usually will be in reduced state)
2. Find closure to find candidate key
3. 1NF, 2NF, 3NF.

Irreducible FDS - Computing Minimal Cover

(i) $A \rightarrow C$

$\{A\}^+ = \{A, C, F, D\}$

remove $A \rightarrow C$

$\{A^+\} = \{A, F, D\}$

(ii) $A \rightarrow F$

$\{A^+\}^+ = \{A, F, D, C\}$

remove $A \rightarrow F$

$\{A$

} diff => cannot remove

$A \rightarrow C$

$A \rightarrow F$

$A \rightarrow D$

$C \rightarrow D$

$B \rightarrow E$

Embedded SQL

* Database Programming

→ most database interactions are executed through programs, called application programs

eg. canned transactions by end users

→ another application of dbms programming is to access a database through an application program that implements a web interface

eg. airline reservations / online purchases

* Database Programming Approaches

(i) embedded commands in general purpose programming lang.

(ii) use a library of database functions / classes

(iii) design a brand new full-fledged language

A. Embedded Commands

→ embedded in a general purpose programming lang

→ programming language is called. host language

→ database statements are identified by the prefix EXEC SQL

→ pre compiler scans program to identify db statements

→ called as embedded SQL.

B. Using a library of database functions

→ library functions available to the host lang. for db calls

→ those functions can:

(i) connect db

(ii) prepare a query

(iii) execute query loops

→ the actual db query & update commands are passed as parameters to the function calls

→ approach called as an API.

→ eg. usage of the JDBC class library

C. Designing a Brand New Language

→ a db programming language designed from scratch to be compatible with the db model and query language.

→ programming structures such as loops and conditional statements are added to the db language to make it a full-fledged programming language.

→ eg. Oracle's PL/SQL, SQL/PSM

* Impedance Mismatch

→ Incompatibilities between a host programming language and the database model

→ The datatypes of the prog. lang. differ from the attribute data types ⇒ require a new binding for each attribute type. (diff. binding needed for diff. prog. lang)

→ Datatypes in C / C++ & Java are different, and differ from the sql datatypes.

→ The results of most queries are multisets of tuples (each tuple is a set of attribute values)

→ A binding is needed to map the query result of the db with the appropriate data structure

→ A mechanism is needed to loop over tuples in a query result - need special iterators (cursor / iterator variable is used)

→ Impedance mismatch is less of a problem when a special db prog. lang. is designed eg. Oracle's PL/SQL

→ Also, object data model is similar to Java, use Java for lessening impedance mismatch.

* Steps in Database Programming

1. Client program opens a connection to the db server
2. Client program submits queries / updates db.
3. Client program closes (terminates) connection

* SQL Commands for connecting to a database

(i) Connection

CONNECT TO <server name> AS <connection-name>

AUTHORIZATION <user-account-info>

(ii) change from an active connection to another

SET CONNECTION connection-name

(iii) disconnection

DISCONNECT connection-name

* Embedded SQL

→ SQL statements within EXEC SQL or EXEC SQL BEGIN and END-EXEC or EXEC SQL END

→ shared variables, used in both languages, prefixed with a colon(:) SQL.

* SQLCODE and SQLSTATE

SQLCODE = 0 → indicates statement was executed successfully

SQLCODE < 0 → some error has occurred

SQLCODE > 0 → no data found

SQLSTATE - a communication variable, is a string of 5 characters

SQLSTATE = '00000' ⇒ no error or exception

SQLSTATE = '02000' ⇒ no more data

* Variable Declaration

```
EXEC SQL BEGIN DECLARE SECTION;
    varchar fname[16];
    char    ssn[10]
    int     dno
    int SQLCODE;
```

```
char SQLSTATE;
EXEC SQL END DECLARE SECTION;
```

Example: To print employee details based on ssn in a loop

```
int loop = 1
while (loop) {
    prompt ("Enter ssn :", ssn);
    EXEC SQL
        Select FNAME, LNAME, SALARY
        into :fname, :lname, :salary
        FROM EMPLOYEE WHERE SSN = :ssn ',

    If (SQLCODE = 0)
        printf (fname, lname, salary);
    else
        printf ("No data available");
    prompt ("More ssn? (1=y, 0=0):", loop);
    END EXEC
}
```

* Processing Query Results with Cursors

→ To bridge the gap between set-level retrieval capabilities of SQL and row-level retrieval capabilities of host - a cursor is used

→ A cursor is a logical pointer pointing to each of the rows providing addressability to those rows one at a time.

FETCH → moves cursor to the next tuple

CLOSE CURSOR → indicates that the processing of query results has been completed.

(or)

```
EXEC SQL DECLARE <cursor name> CURSOR

EXEC SQL FETCH <cursor name> INTO <host var>

EXEC SQL CLOSE <cursor name>
```

Example : Fetch the supplier no, supplier name & status for a given city.

```
EXEC SQL DECLARE X CURSOR FOR
        SELECT S.supplier_no , S.supplier_name, S.status
        FROM Supplier S
        WHER S.city = :Y

        EXEC SQL FETCH X INTO  :supplier_no, :supplier_name,
                               :status,

        END

EXEC SQL CLOSE X;
```

* Usage of Cursors for Updation

Example: To enter the department ~~name~~ name, and update each employee's salary by a specific amount.

Ans:

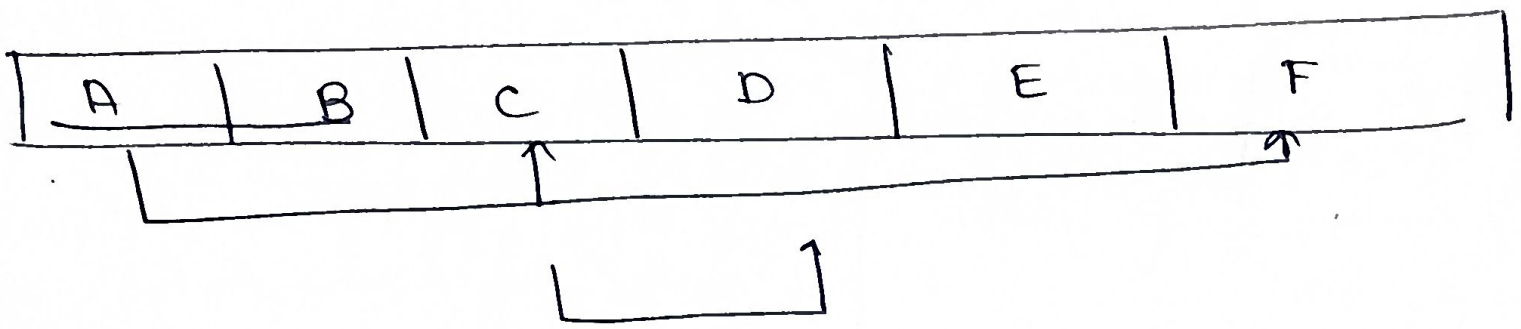
```
prompt ("Enter dept name", dname);
EXEC SQR
    SELECT Deptno INTO : deptno
    FROM Department WHERE
      dname = :dname;
    EXEC SQR DECLARE EMP CURSOR FOR
          SELECT Ssn, fname, lname, salary
          FROM employee
          WHERE
            Dno = :dnumber
    FOR UPDATE OF Salary
    EXEC SQR OPEN Emp,
    EXEC SQR FETCH FROM EMP INTO :ssn, :fname,
        :lname, :salary;
        while (SQRCODE == 0) {
              prompt ("Enter raise amt:", raise);
          EXEC SQL
              UPDATE EMPLOYEE
              SET SALARY = SALARY + :raise
              WHERE CURRENT OF EMP
```

```
    EXEC SQR FETCH FROM EMP INTO :ssn, :fname, :lname,
        :salary
}
EXEC SQR CLOSE EMP;
```

Unit 3

Additional Questions on Normalization, Closure & Minimal Covers

(1) Consider the following schema. Reduce using 1NF, 2NF, 3NF and BCNF is possible

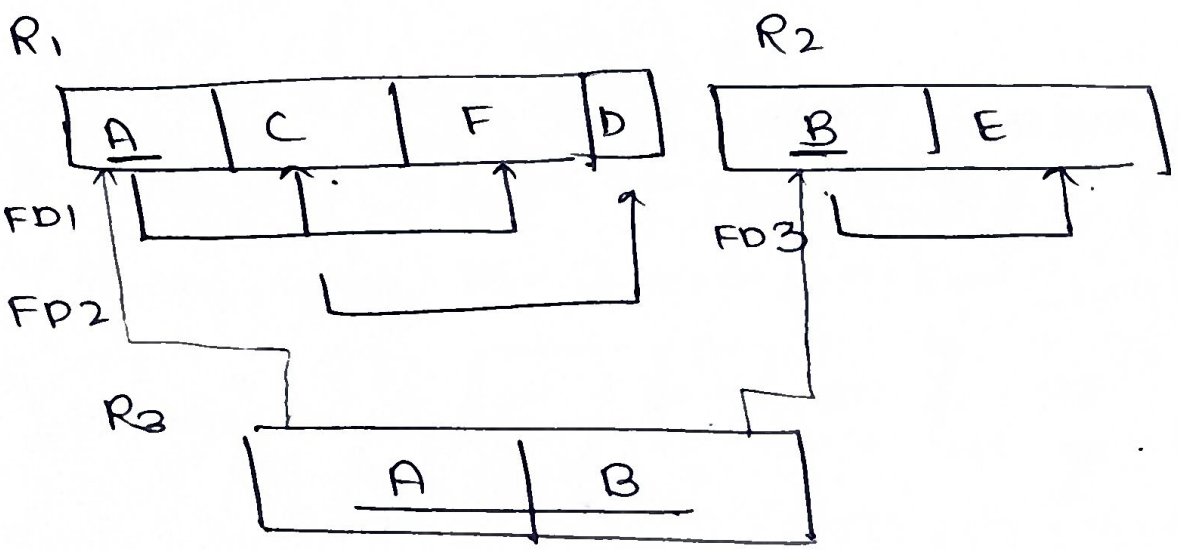


1NF: no multivalued attributes ⇒ in 1NF

2NF: FD1 = partial FD

FD2 = full FD

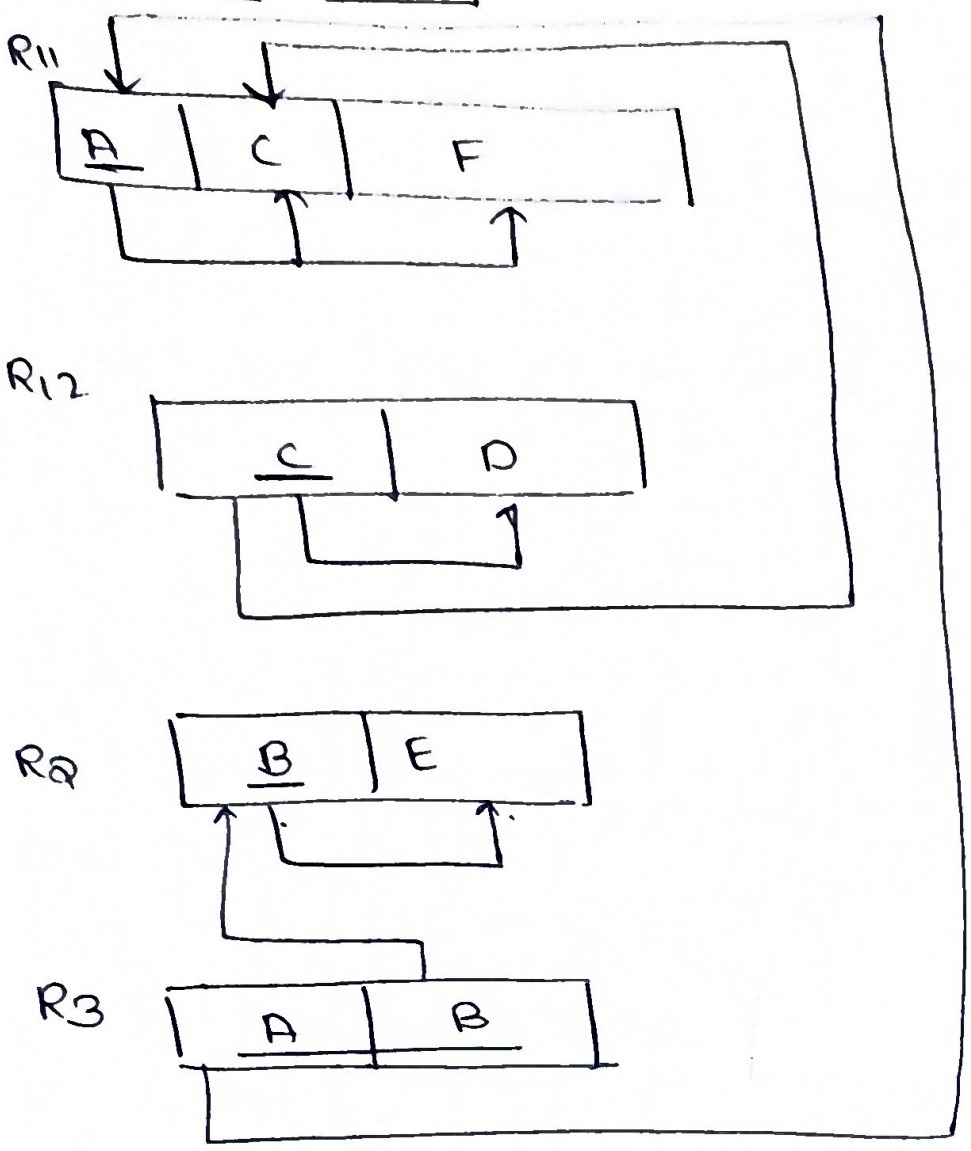
FD3 = partial FD



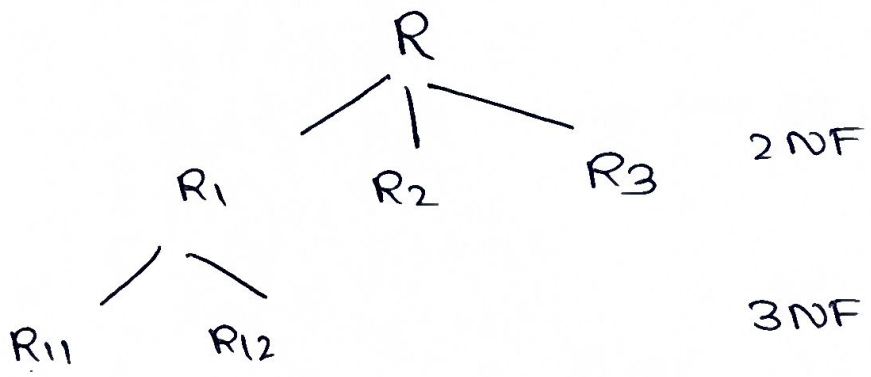
3NF: R_2 and R_3 are in 3NF

R_1 is not in 3NF as $A \rightarrow C$ and $C \rightarrow D$ (C is not a candidate key)

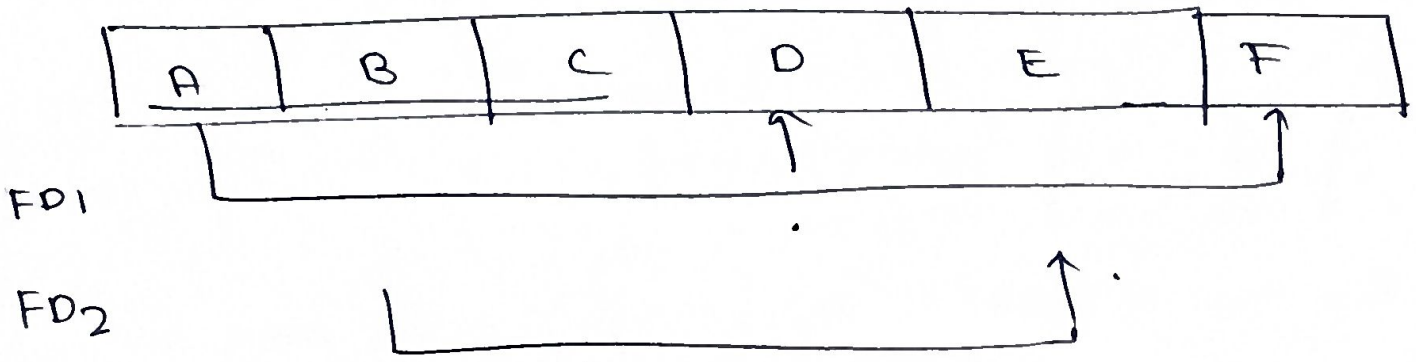
Reduced Tables



Reduction



(2) Consider the following schema. Normalize till 3NF, or BCNF if possible.

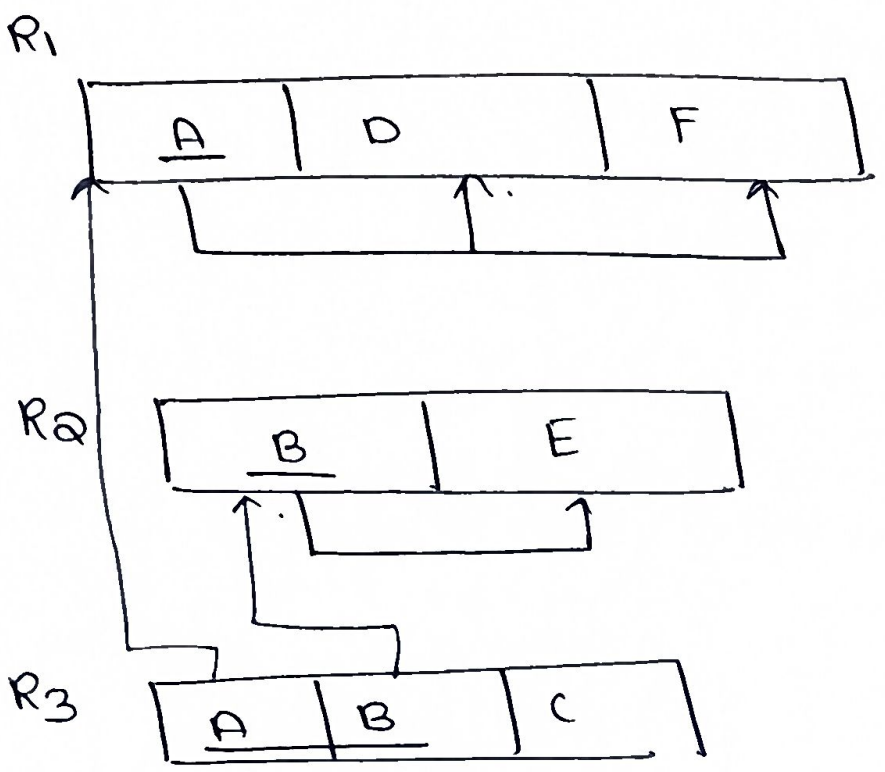


1NF : no multivalued attributes ⇒ in 1NF

2NF : FD1 ⇒ partial FD

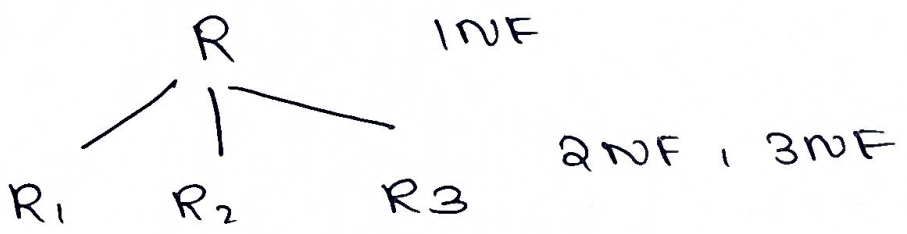
FD2 ⇒ partial FD.

Reduction



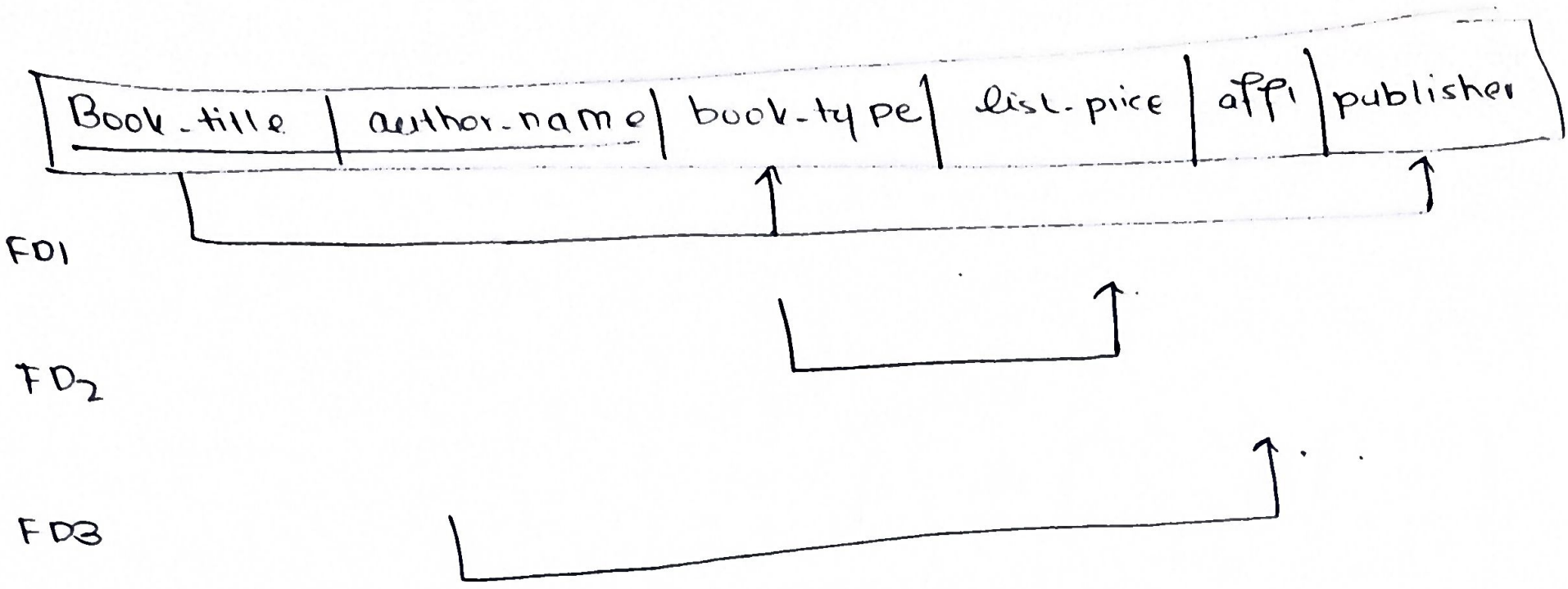
2NF is satisfied.

3NF : There are no transitive dependencies ⇒ 3NF is satisfied.



(3) Consider the following relation, with the given functional dependencies. Apply normalization until the relations cannot be decomposed further.

Book



Solution

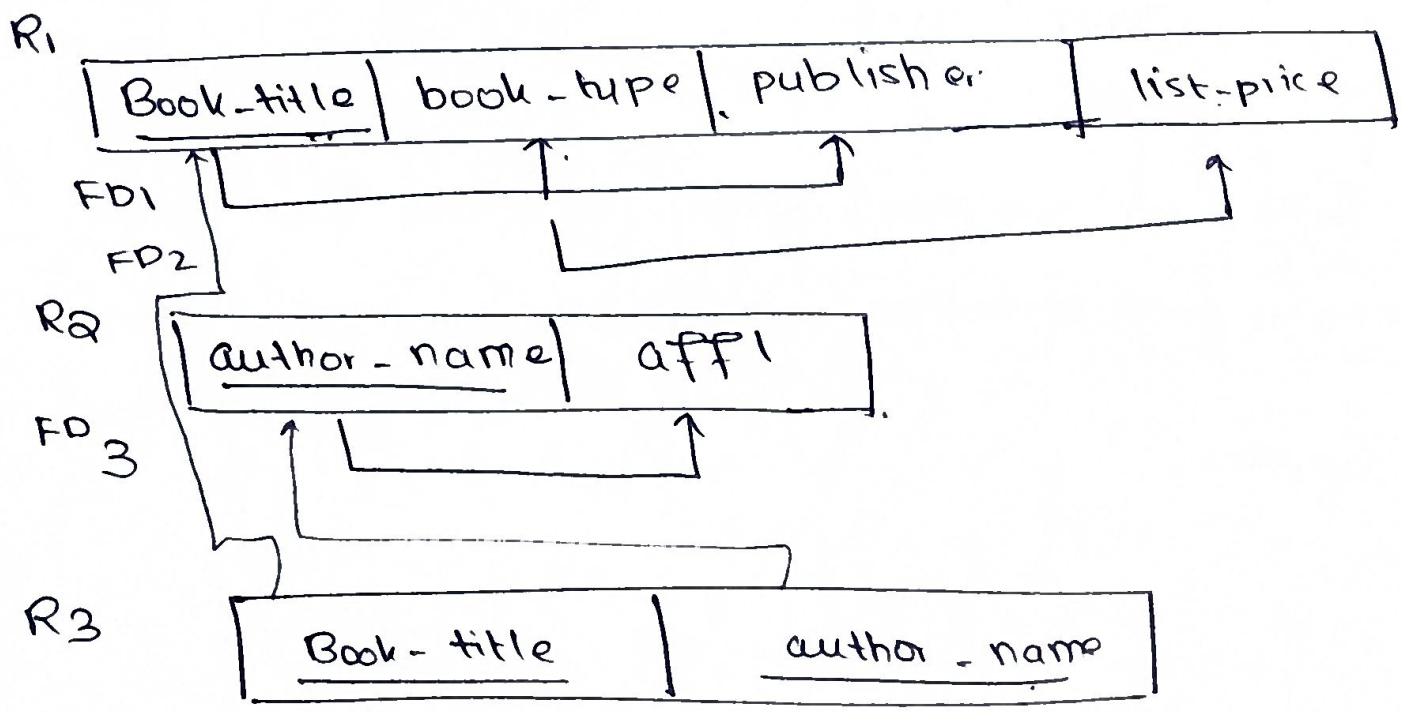
1NF: no multivalued attributes ⇒ 1NF is satisfied

2NF: FD1 ⇒ partial FD

FD2 ⇒ full FD

FD3 ⇒ partial FD

Reduction



3NF : FD1 : book_title → book_type and

and FD3 book_type → list_price

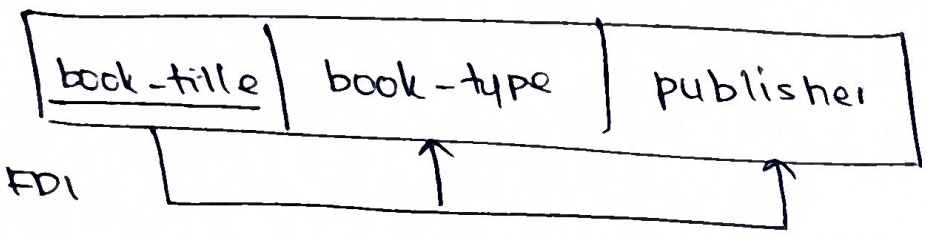
and book_type is not a candidate key

⇒ not in 3NF

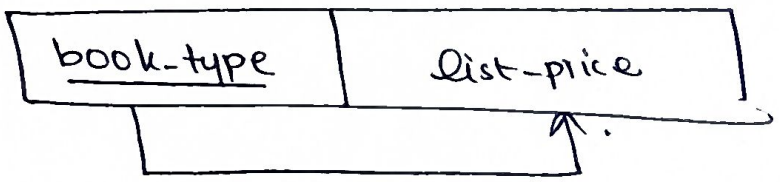
FD2 : in 3NF - no transitive dependency

Reductions

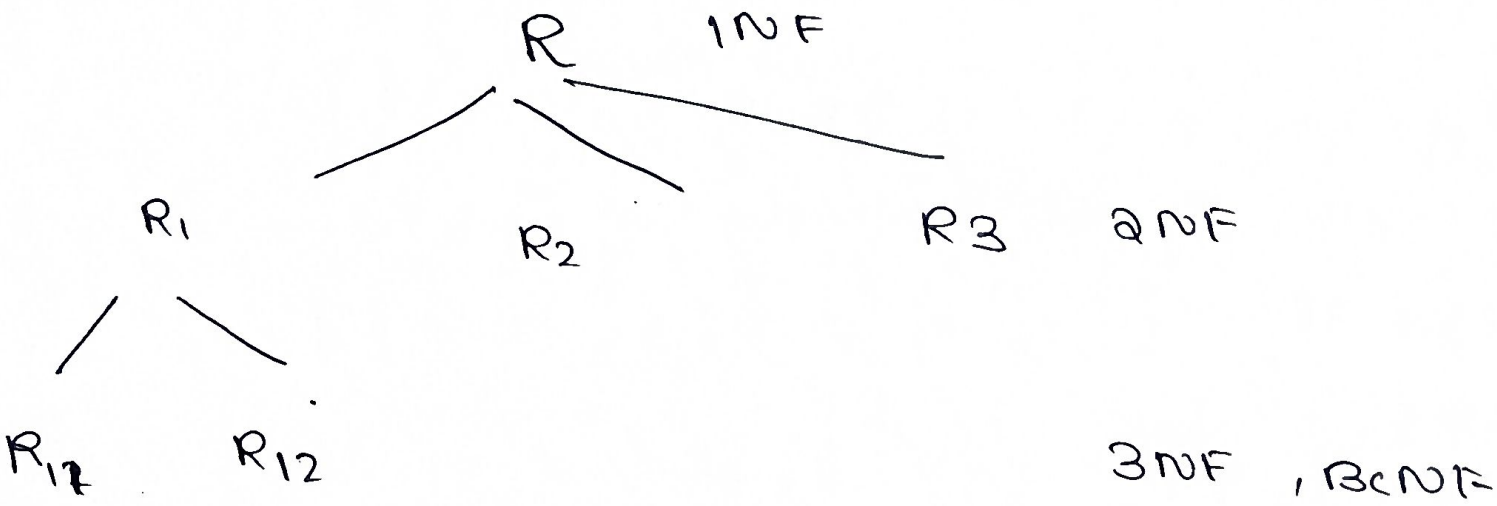
R_{11}



R_{12}



BCNF : is satisfied - determinants all superkeys.



(3) Consider the FDs : $A \rightarrow BC$

$E \rightarrow CF$

$B \rightarrow E$

$CD \rightarrow EF$

Compute the closure of $\{A, D\}^+$

Iter1 $\{AD\}^+ \subseteq \{A, D\}$

$A \subseteq \{A, D\} \Rightarrow \{A, B, C, D\}$

$E \not\subseteq \{A, B, C, D\}$ => no change

$B \subseteq \{A, B, C, D\} \Rightarrow \{A, B, C, D, E\}$

$CD \subseteq \{A, B, C, D\} \Rightarrow \{A, B, C, D, E, F\}$

Iter2 ~~$A \subseteq \{A, B, C, D, E, F\}$~~

$\{A, B, C, D, E, F\}$ is all the attributes

$\therefore$ closure of $\{A, D\}^+ = \{A, B, C, D, E, F\}$

(4) Consider the FDs

$A \rightarrow BC$

$B \rightarrow E$

$CD \rightarrow EF$

Use these FDs to show that the FD $AD \rightarrow F$ holds for R.

$A \rightarrow BC$ (given)

$B \rightarrow E$ (given)

$BC \rightarrow EC$ (augmentation)

Armstrong's Axioms

$A \rightarrow EC$ (transitivity)

$A \rightarrow E$
$A \rightarrow C$
} decomposition

$AD \rightarrow CD$ (augmentation)

$CD \rightarrow EF$ (given)

$AD \rightarrow EF$ (transitivity)

$AD \rightarrow E$
$AD \rightarrow F$
} decomposition

(5) Given $R = \{A, B, C, D, E\}$ with the FDs

~~$\{A, B$~~ $AB \rightarrow C$
$A \rightarrow D$
$D \rightarrow E$

Select the FD which does not satisfy 2NF.

Solution : Finding the candidate key of the relation R.

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

$\{A\}^+ = \{A, D, E\}$

candidate key = AB

$A \rightarrow D$ is a partial FD

$\Rightarrow$ does not satisfy 2NF.

(Marks) (6) Find the minimal cover for the given set of FDs

$B \rightarrow A$

$D \rightarrow A$

$AB \rightarrow D$

Solution

(i) closure of $B \rightarrow A$

$\{B\}^+ = \{B, A, D\}$

remove $B \rightarrow A$

$\{B\}^+ = \{B\}$

$B \rightarrow A$
$D \rightarrow A$
$AB \rightarrow D$

} not the same ⇒ cannot remove

(ii) closure of $D \rightarrow A$

$\{D\}^+ = \{D, A\}$

remove $D \rightarrow A$

$\{D\}^+ = \{D,\}$

} not the same ⇒ cannot remove

(iii) closure of $AB \rightarrow D$

$\{A, B\}^+ = \{A, B, D\}$

remove $AB \rightarrow D$

$\{A, B\} = \{A, B,\}$

} not the same ⇒ cannot remove

(7) Find the candidate key(s) of the relation R with the set of FDs:

(9)

$AB \rightarrow C$

$CD \rightarrow E$

$DE \rightarrow B$

(i) closure of $\{A, B\}^+ = \{A, B, C\}$

(ii) closure of $\{C, D\}^+ = \{C, D, E, B\}$

(iii) closure of $\{D, E\}^+ = \{D, E, B\}$

combine attributes

closure of $\{A, B, D\} = \{A, B, C, D, E\}$

candidate keys

$\Rightarrow \{A, B, D\}$ //.

(8) Consider the relation $R = \{A, B, C, D, E, F, G, H, I, J\}$ and the set of functional dependencies:

$AB \rightarrow C$

$BD \rightarrow EF$

$AD \rightarrow GH$

$A \rightarrow I$

$H \rightarrow J$

Identify the candidate key, and then decompose into 2NF and 3NF.

Solution

$AB \rightarrow C$

$BD \rightarrow EF$

$AD \rightarrow GH$

$A \rightarrow I$

$H \rightarrow J$

closure of $\{A,B\}^+ = \{A,B,C,I\}$

$\{B,D\}^+ = \{B,D,E,F\}$

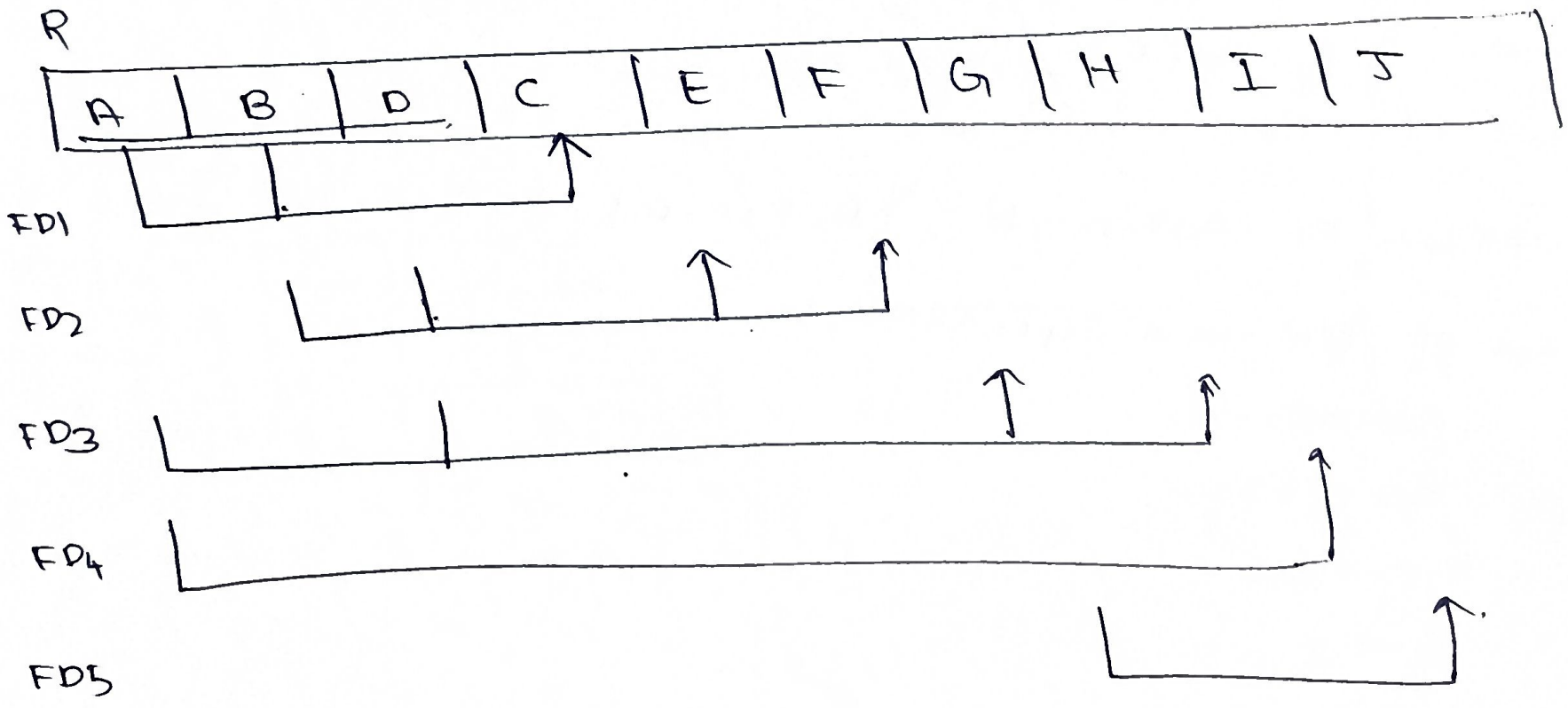
$\{A,D\}^+ = \{A,D,G,H,I,J\}$

$\{A\}^+ = \{A,I\}$

$\{H\}^+ = \{H,J\}$

closure of $\{A,B,D\}^+ = \{A,B,C,D,E,F,G,H,I,J\}$

The candidate key is $\{ABD\}$



② 1NF : no multivalued attributes

=> in 1NF

2NF : FD1 : partial FD

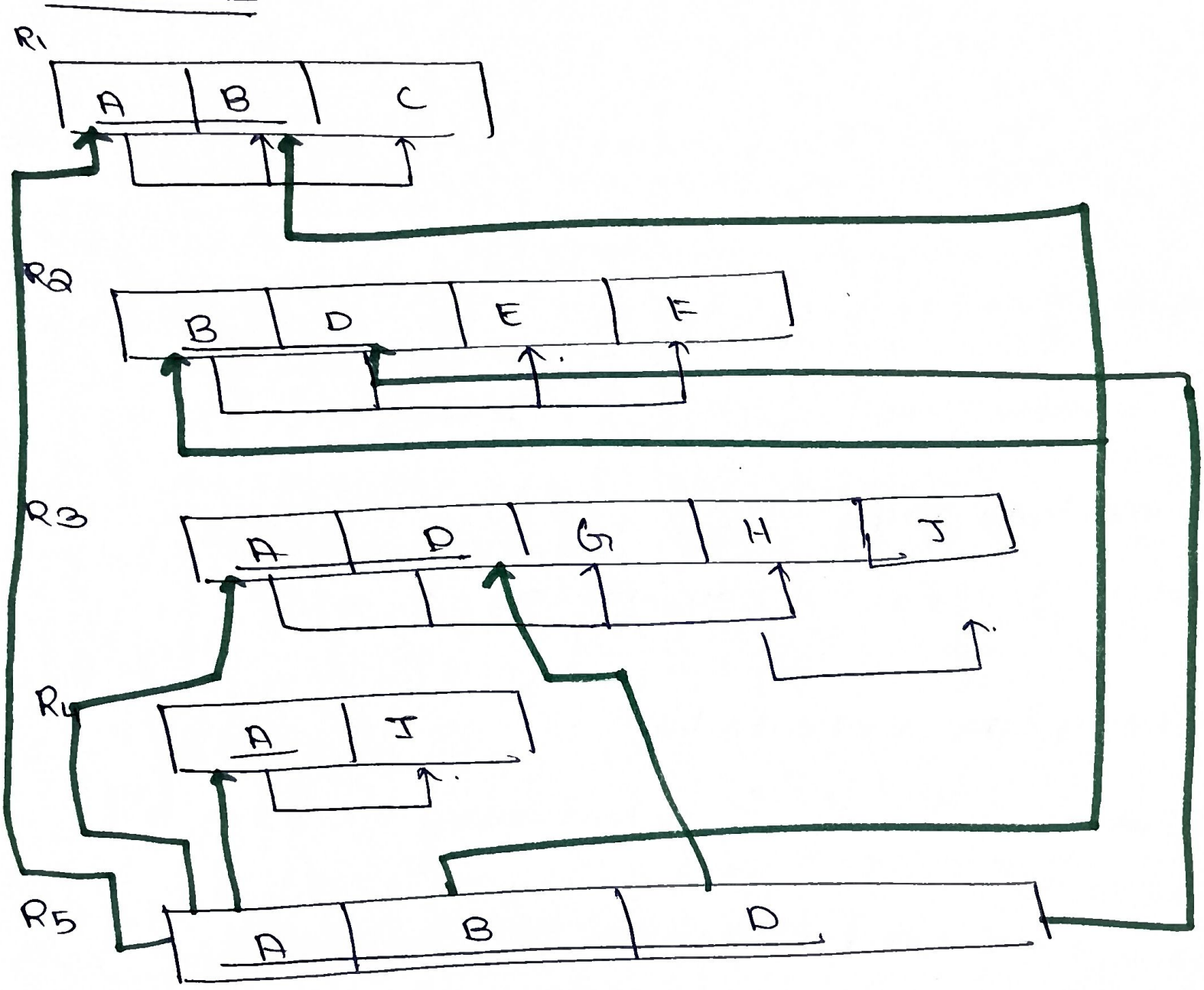
FD2 : partial FD

FD3 : partial FD

FD4 : partial FD

FD5 : full FD

Reductions



3NF FD1 : in 3NF

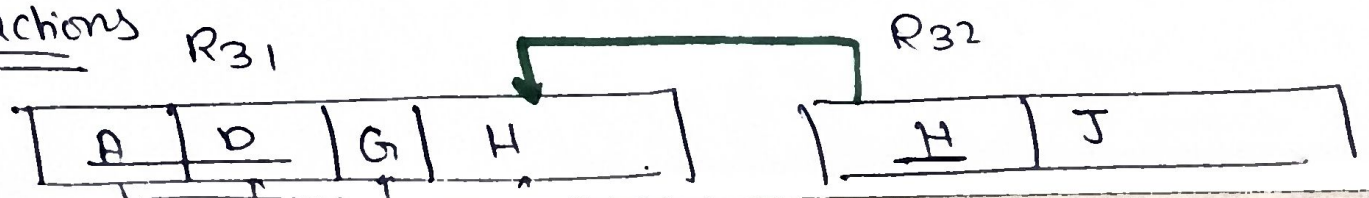
FD2 in 3NF

FD3 $A \rightarrow H$, $H \rightarrow J$, not in 3NF

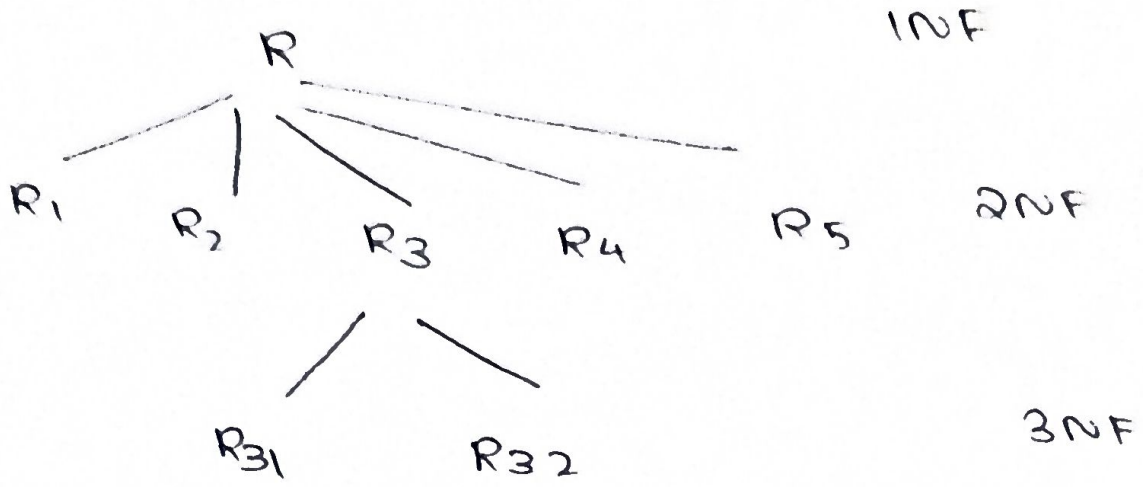
FD4 : in 3NF

FD5 : in 3NF

Reductions



Decompositions



(9) Consider the relation SST (Student, Subject, Teacher) and the set of FDs

{student, subject} → teacher

teacher → subject

(i) Find the candidate key

(ii) Decompose & check if in 3NF and BCNF.

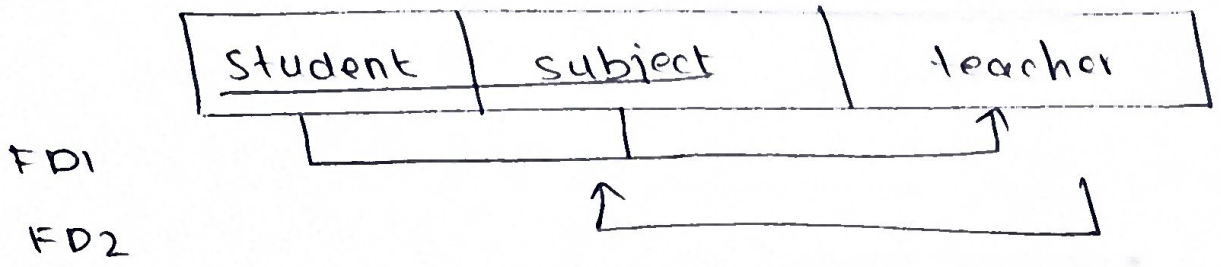
Solution Finding the candidate key

{Student, subject}$^+$ = {student, subject, teacher}

{teacher}$^+$ = {teacher, subject}

candidate key : {Student, subject}

Schema



2NF : FD1 = full FD

FD2 = full FD

in 2NF

3NF : student, subject → teacher & teacher → subject

transitive dependency where teacher is not a candidate key

Reductions