DATABASE MANAGEMENT SYSTEMS (IT 2040)

LECTURE 06 - SCHEMA REFINEMENT

LECTURE CONTENT

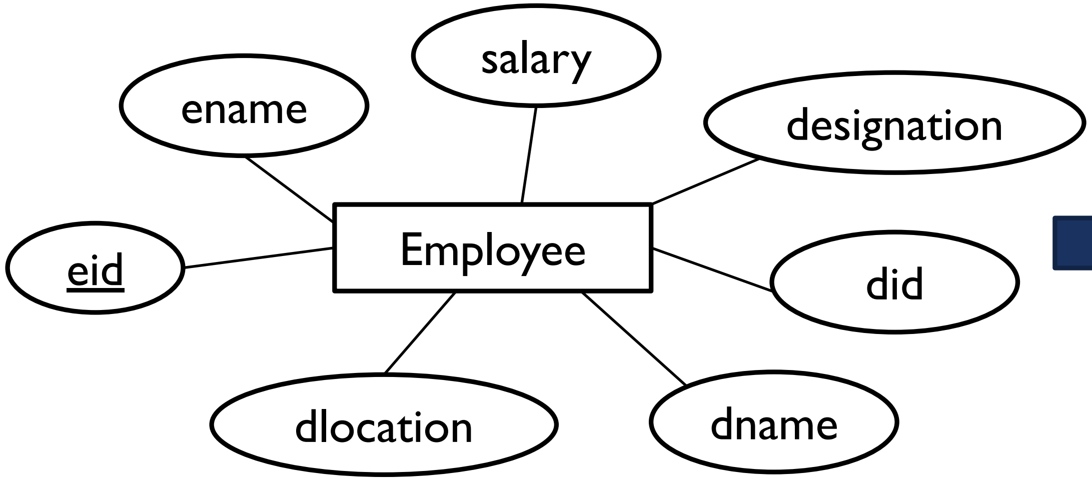
- Why schema refinement?
- Properties of good decomposition
- Functional dependencies
- Computing keys of relations
- Normalization and normal forms

LEARNING OUTCOMES

- Explain the pitfalls of incorrect grouping of attributes in relations.
- Explain the properties of a good decomposition.
- Compute keys from a given set of FDs in a relation
- Normalize a given schema to BCNF.

WHY SCHEMA REFINEMENT?

- The relations resulted through the logical database design may not be very good if your conceptual database design is not good.
- For example, what is wrong with the following schema resulted through mapping of an ER diagram



Employee

eid	ename	designation	salary	did	dname	location

WHY SCHEMA REFINEMENT? (CONTD.)

- Schemas such as in the previous slide lead to several anomalies while inserting, updating and deleting and wasting of space due to redundancies of data.
- Can you spot where data are duplicated and issues may cause during insert/update & delete?

eid	Ename	Designation	Salary	did	dname	location
1000	Ajith	Lecturer	60000	1	Academic	malabe
1001	Sunil	Executive	45000	3	Maintenance	Kandy
1002	Kamal	Lecturer	75000	1	Academic	malabe
1003	Piyumi	Manager	50000	2	Admin	metro
1004	Roshan	Lecturer	35000	1	Academic	malabe
1005	Nuwan	Lecturer	80000	1	Academic	malabe
1006	Jayamini	Assistant	25000	2	Admin	metro
1007	Nishani	Lecturer	42000	1	Academic	malabe
1008	Amal	Assistant	28000	4	ITSD	Matara

WHY SCHEMA REFINEMENT? (CONTD.)

- Insertion Anomaly
  - Inserting a new employee to the emp table
    - Department information is repeated (ensure that correct department information is inserted).
  - Inserting a department with no employees
    - Impossible since eid cannot be null
- Deletion Anomaly
  - Deleting the last employee from the department will lead to loosing information about the department
- Update Anomaly
  - Updating the department's location needs to be done for all employees working for that department

WHY SCHEMA REFINEMENT? (CONTD.)

- To solve the issues discussed previously, we should decompose the relations to smaller relations.
- For example, we can decompose the emp relation discussed previously into two relations as below to overcome the issues of redundancies and anomalies.

eid	Ename	Designation	Salary	did
1000	Ajith	Lecturer	60000	1
1001	Sunil	Executive	45000	3
1002	Kamal	Lecturer	75000	1
1003	Piyumi	Manager	50000	2
1004	Roshan	Lecturer	35000	1
1005	Nuwan	Lecturer	80000	1
1006	Jayamini	Assistant	25000	2
1007	Nishani	Lecturer	42000	1
1008	Amal	Assistant	28000	4

did	dname	location
1	Academic	malabe
2	Admin	metro
3	Maintenance	Kandy
4	ITSD	Matara

WHY SCHEMA REFINEMENT? (CONTD.)

- Random decompositions however, may introduce new problems.
- Two properties that could be looked up to ensure that the relations resulted from decomposition are good are as follows:
  - Loss-less join property
  - Dependency preserving property

WHY SCHEMA REFINEMENT? (CONTD.)

- Loss-less join property : the property enables recovery of original relation from a set of smaller relations resulted through decomposition.
  - For example, suppose S is decomposed in to R_1 and R_2. When we join R1 and R2 do we get S? if so, we say the decomposition is loss-less.

S

S	P	D
S1	P1	D1
S2	P2	D2
S3	P1	D3

R_1

S	P
S1	P1
S2	P2
S3	P1

R_2

P	D
P1	D1
P2	D2
P1	D3

- Dependency preserving property : enables to enforce any constraint on the original relation simply enforcing some constraints on each of the smaller relation.

SCHEMA REFINEMENT

- Schema refinement can be considered a systematic process for analyzing a relational schema with the aim of minimizing redundancies and minimizing insertion, deletion and update anomalies.
- The process performs a series of tests called normal form tests to check whether the schemas meet certain conditions, and those relations that are unsatisfactory are decomposed in to smaller relation.
- Normalization is based on functional dependencies

FUNCTIONAL DEPENDENCY IN GENERAL TERMS

- Functional dependency is a relationship that exists when one attribute uniquely determine another attribute.
- For example: Suppose we have a student table with attributes: id name, age.
  - Here id attribute uniquely identifies the name attribute of student table because if we know the student id we can tell the student name associated with it.
  - This is known as functional dependency and can be written as id->name or in words we can say name is functionally dependent on id.
- Redundancies in relations are based on functional dependencies

FUNCTIONAL DEPENDENCIES

- Mathematical definition of functional dependency (FD) :
  - A functional dependency, denoted by $X \rightarrow Y$, where X and Y are sets of attributes in relation R, specifies the following constraint:
    - Let t_1 and t_2 be tuples of relation R for any given instance

      Whenever $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$

      where $t_i[X]$ represents the values for X in tuple t_i

ACTIVITY

- Consider a person entity. With respect to attributes you know a person holds write three functional dependencies that can exist in a person table.
- Exchange what you have written with your peers.
- What have they written?

KEYS AND FDS

- A key constraint is a special case of a FD where the attributes in the key play the role of X and the set of all attributes play the role of Y.
- Normalization process analyzes schemas based on keys and on the functional dependencies among their attributes.
- Thus, to start normalization it is essential to find keys of a given relation.

COMPUTING KEYS

- Attribute closure of an attribute set X, denoted by X+ can be defined as set of all attributes which can be functionally determined from it.
- If X+ = all attributes, then X is a key.
- Attribute closure could be computed using Armstrong Axioms.
  - X, Y, Z are sets of attributes:
    - Reflexivity: If $X \subseteq Y$, then $Y \square X$
    - Augmentation: If $X \square Y$, then $XZ \square YZ$ for any Z
    - Transitivity: If $X \square Y$ and $Y \square Z$, then $X \square Z$
  - Couple of additional rules (that follow from Armstrong Axioms):
    - Union: If $X \square Y$ and $X \square Z$, then $X \square YZ$
    - Decomposition: If $X \square YZ$, then $X \square Y$ and $X \square Z$

COMPUTING KEYS (CONTD.)

- Consider a relation R (A, B, C, D), with the following set of functional dependencies over R:
  - $F = \{ A \rightarrow B, B \rightarrow C, B \rightarrow D\}$

$A \rightarrow A$ (reflexivity rule)
$A \rightarrow B$ (given)
$A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$ (transitivity)
$A \rightarrow B$ and $B \rightarrow D$ then $A \rightarrow D$ (transitivity)

$A \rightarrow ABCD$ (Union rule) , $[A]^+=\{ABCD\}$

$B^+= \{BCD\}$, $C^+ = \{C\}$, $D^+=\{D\}$
Therefore A is the key

ACTIVITY

- Consider a relation R (A, B, C, D, E), with the following set of functional dependencies over R:
- $\{A \rightarrow B, A \rightarrow C, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$
- Compute the keys for relation R.

REVISIT TO SOME DEFINITIONS

- Superkey: Set of attributes S in relation R such that no two distinct tuples t_1 and t_2 will have $t_1[S] = t_2[S]$
- Key: A key is a superkey with the additional property that removal of any attributes from the key will not satisfy the key condition
- Candidate Key: Each key of a relation is called a candidate key
- Primary Key: A candidate key is chosen to be the primary key
- Prime Attribute: an attribute which is a member of a candidate key
- Nonprime Attribute: An attribute which is not prime

NORMAL FORMS

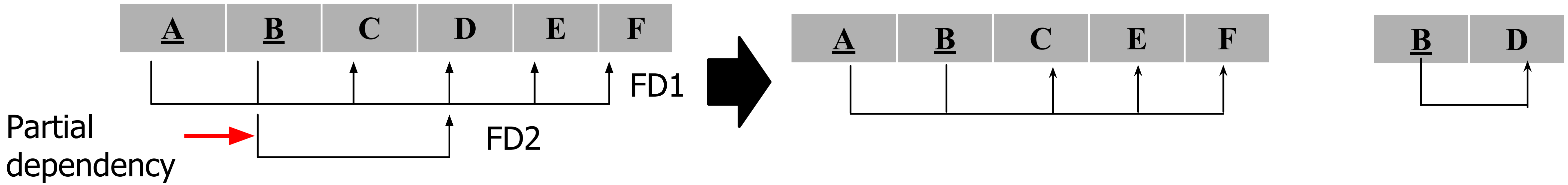
- Normal forms refers to a series of tests performed on relational schemas to improve their goodness.
- We discuss four normal forms namely,
  - 1st Normal Form
  - 2nd Normal Form
  - 3rd Normal Form
  - Boyce Codd Normal Form
- Test for each normal form is performed in a top-down fashion.
- The Normal form of a relation refers to the highest normal form condition it meets.

1ST NORMAL FORM

- A relation R is in first normal form (1NF) if domains of all attributes in the relation are *atomic* (simple & indivisible).
- 1NF is now considered to be part of the formal definition of a relation in the relational model since it allows only atomic values and disallows multivalued attributes and composite attributes.

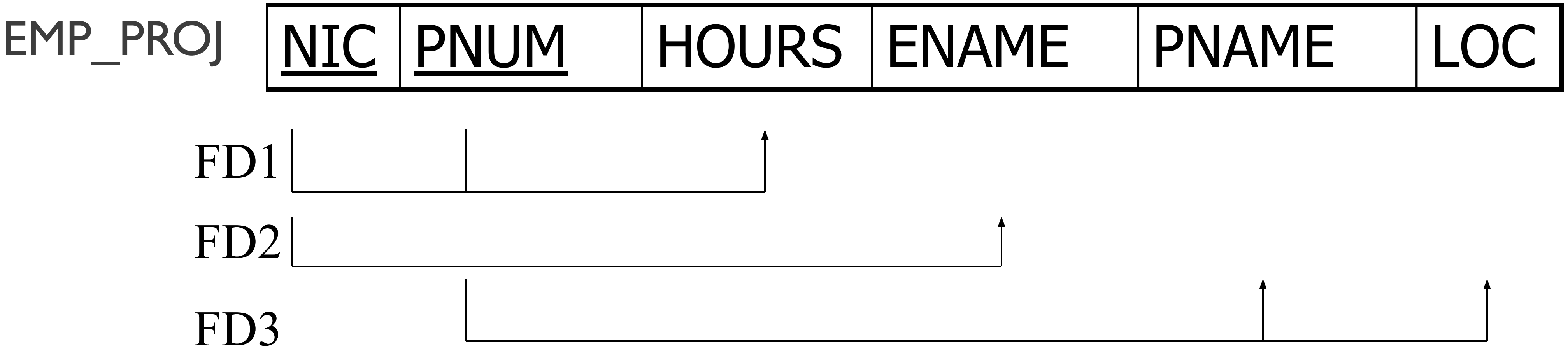
2ND NORMAL FORM

- A relation R is in second normal form (2NF) if every nonprime attribute A in R is not partially dependent on any key of R.
- Second normal form (2NF) is based on the concept of full functional dependency.
  - 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.
  - For example, in a relation R (ABCDE) where AB -> {ABCDE}, if A-> C , A->C is a partial dependency (not fully functional dependent)
- To normalize the relation to 2NF decomposition is performed as follows.



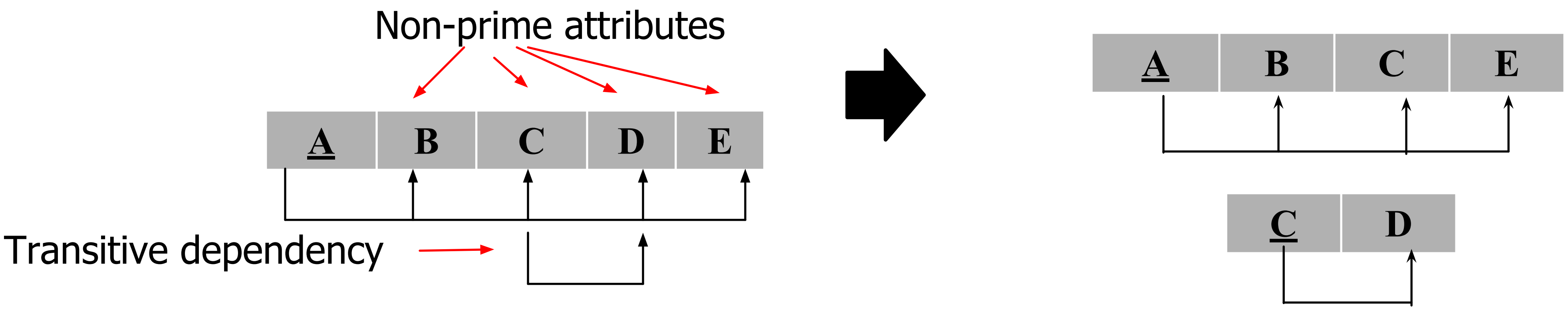
ACTIVTY

- What normal form the relation in the slide is? If the relation is not in 2NF normalize the relation to 2NF.



3^{RD} NORMAL FORM

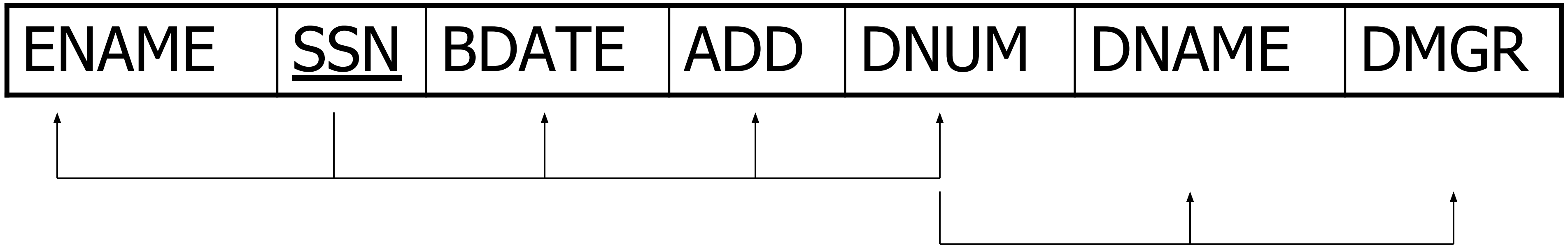
- A relation R is in 3rd normal form (3NF) if every
  - R is in 2NF, and no nonprime attribute is transitively dependent on any key
- Third normal form is based on the concept of transitive dependency.
  - A functional dependency X →Y in a relational schema R is a transitive dependency if there is a set of non-prime attributes Z where both X →Z and Z →Y hold.
- To normalize the relation to 2NF decomposition is performed as follows.



ACTIVITY

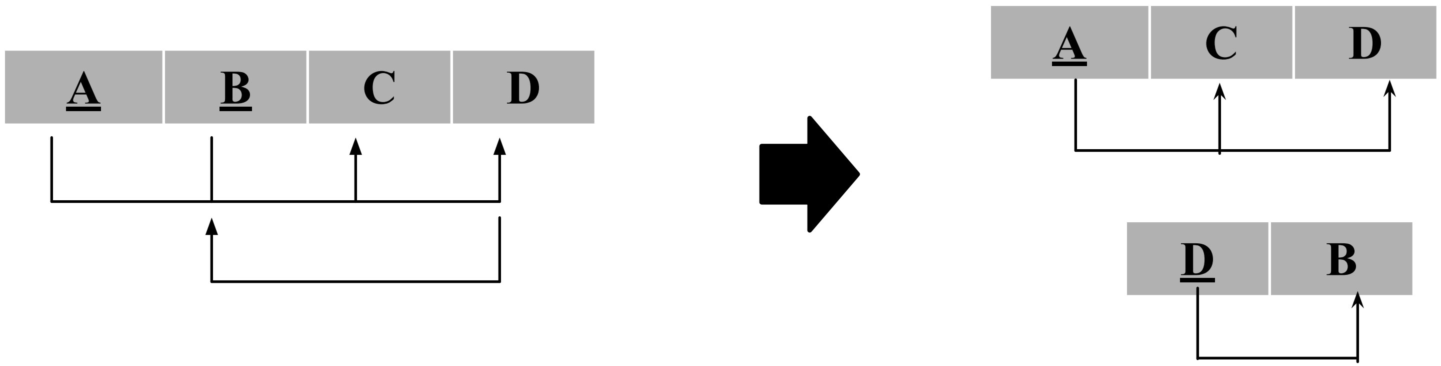
- What normal form the relation in the slide is? If the relation is not in 3NF normalize the relation to 3NF.

EMP_DEPT



BOYCE-CODD NORMAL FORM

- A relation schema is in Boyce-Codd Normal Form
  - If every nontrivial functional dependency X□A hold in R, then X is a superkey of R
- Decomposition into BCNF:
  - Consider relation R with FDs F. If X □ Y violates BCNF, decompose R into R - Y and XY

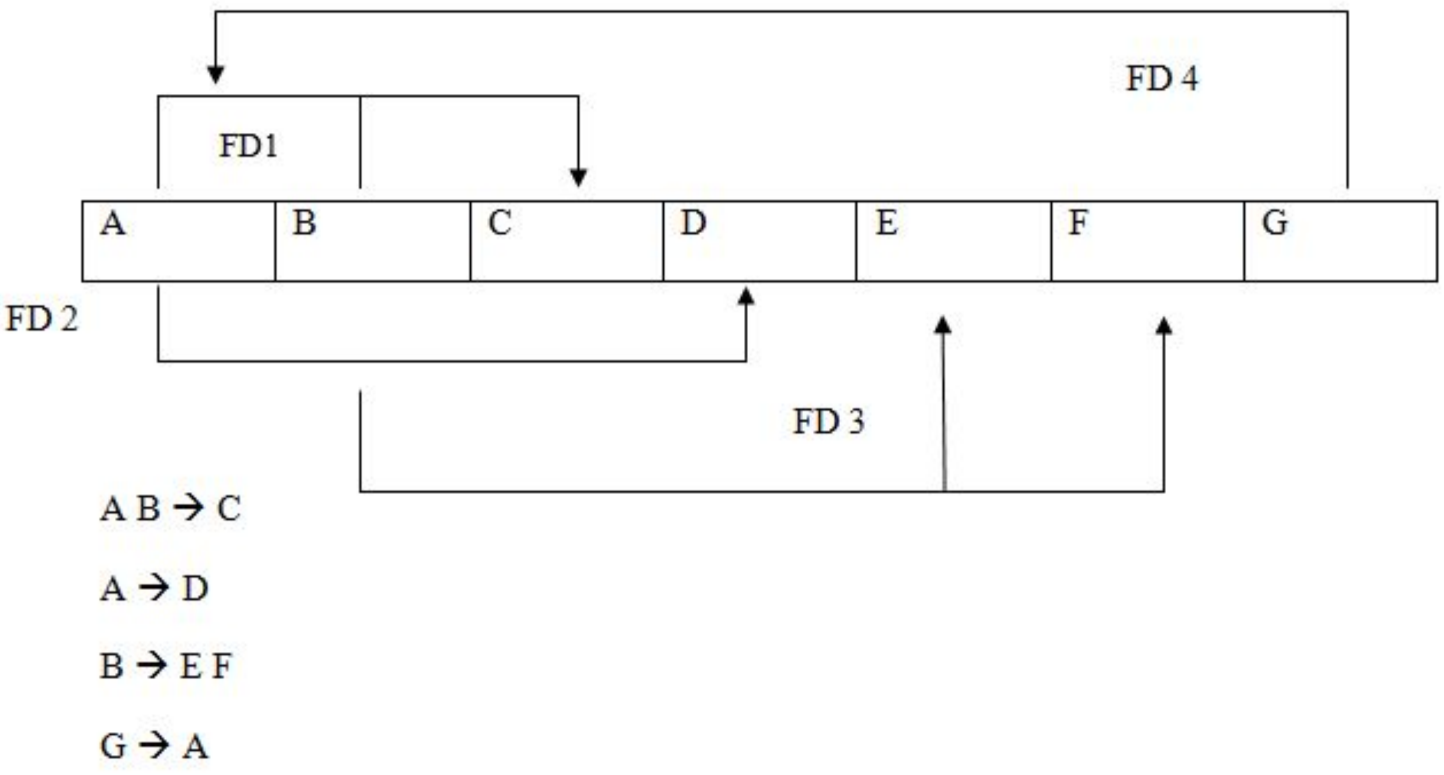


ACTIVITY

- Consider the following relational schema for R:

  R(A, B,C,D,E,F,G)

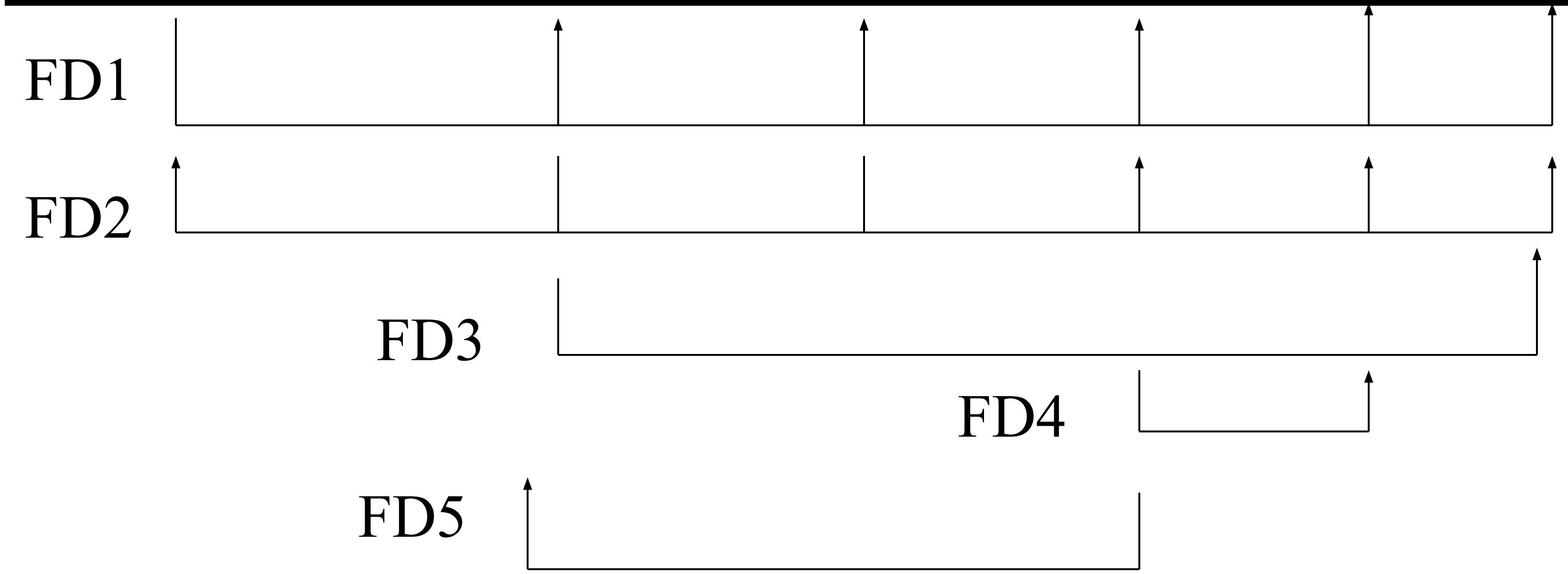
- AB is the primary key in the relation.



- What normal form is the relation in?
- If R is not in BCNF, convert it to BCNF.

ACTIVITY

PROPERTY_ID	COUNTY_NAME	LOT#	AREA	PRICE	TAX_RATE



WHAT YOU HAVE TO DO BY NEXT WEEK

- Try out the self-test questions on the course web.
- Complete the tutorial.