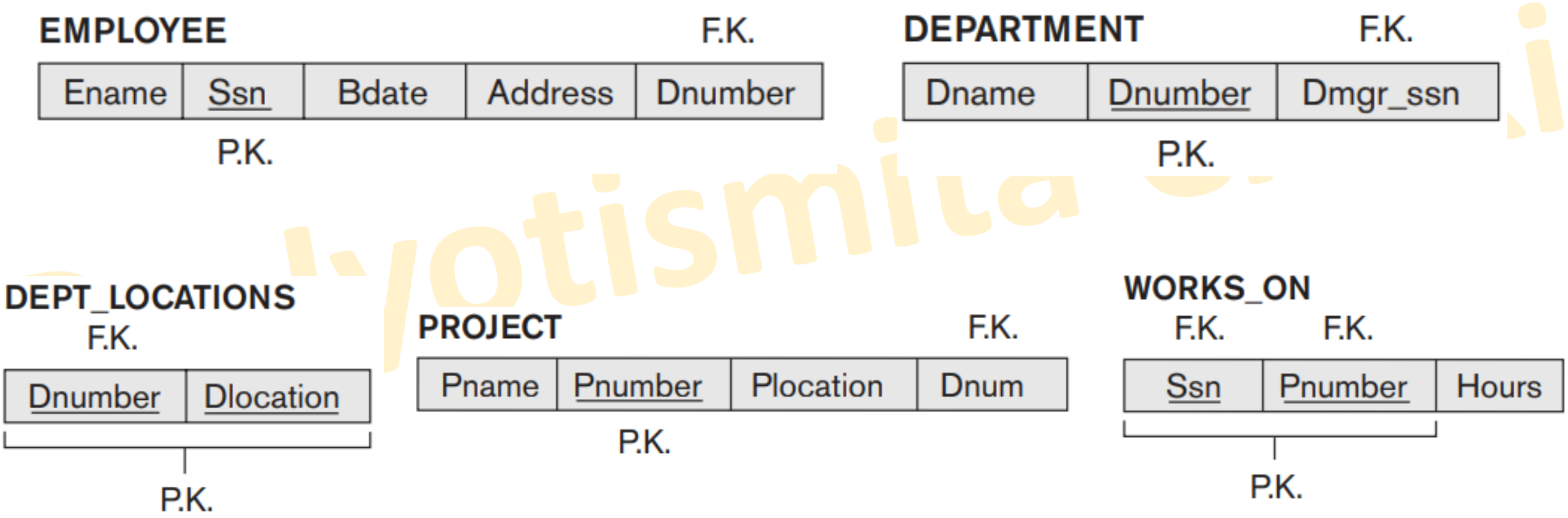
Schema Refinement

Dr. Jyotismita Chaki

Guidelines for relational schema

- Imparting Clear Semantics to Attributes in Relations
  - Design a relation schema so that it is easy to explain its meaning.
  - Do not combine attributes from multiple entity types and relationship types into a single relation.
  - If a relation schema corresponds to one entity type or one relationship type, it is straightforward to explain its meaning.
  - Otherwise, if the relation corresponds to a mixture of multiple entities and relationships, semantic ambiguities will result and the relation cannot be easily explained.
  - The semantics of a relation refers to its meaning resulting from the interpretation of attribute values in a tuple.

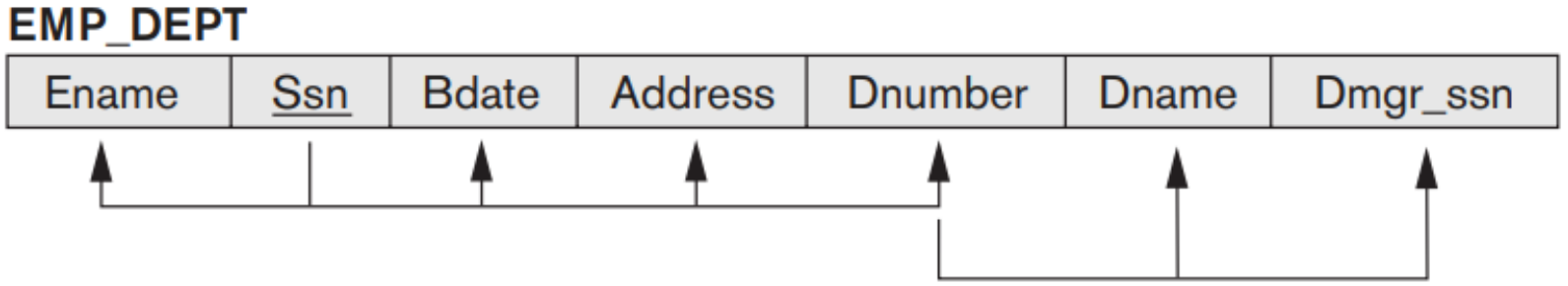
Guidelines for relational schema



A simplified COMPANY relational database schema.

Guidelines for relational schema

- Redundant Information in Tuples and Update Anomalies
  - Storing natural joins of base relations leads to a problem referred to as **update anomalies**
    - Insertion Anomalies:
      - For example, to insert a new tuple for an employee who works in department number 5, we must enter all the attribute values of department 5 correctly so that they are consistent with the corresponding values for department 5 in other tuples in EMP_DEPT.
      - It is difficult to insert a new department that has no employees as yet in the EMP_DEPT relation. The only way to do this is to place NULL values in the attributes for employee. This violates the entity integrity for EMP_DEPT because its primary key Ssn cannot be null.



Guidelines for relational schema

- Redundant Information in Tuples and Update Anomalies
  - Storing natural joins of base relations leads to a problem referred to as **update anomalies**
    - Insertion Anomalies:

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

Guidelines for relational schema

- Redundant Information in Tuples and Update Anomalies
  - Storing natural joins of base relations leads to a problem referred to as **update anomalies**
    - Deletion Anomalies:
      - The problem of deletion anomalies is related to the second insertion anomaly situation.
      - If we delete from EMP_DEPT an employee tuple that happens to represent the last employee working for a particular department, the information concerning that department is lost from the database.

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

Guidelines for relational schema

- Redundant Information in Tuples and Update Anomalies
  - Storing natural joins of base relations leads to a problem referred to as **update anomalies**
    - Modification Anomalies:
      - In EMP_DEPT, if we change the value of one of the attributes of a particular department—say, the manager of department 5—we must update the tuples of all employees who work in that department; otherwise, the database will become inconsistent.
      - If we fail to update some tuples, the same department will be shown to have two different values for manager in different employee tuples, which would be wrong.
  - Guideline: Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations. If any anomalies are present, note them clearly and make sure that the programs that update the database will operate correctly.

Guidelines for relational schema

EMPLOYEE

Ename	Ssn	Bdate	Address	Dnumber
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

No anomalies in this design

Guidelines for relational schema

- NULL Values in Tuples
  - As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL. [problem with NULLs is how to account for them when aggregate operations such as COUNT or SUM are applied. If NULL values are present, the results may become unpredictable in case of SELECT and JOIN operations]
  - If NULLs are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation.
  - For example, if only 15% of employees have individual offices, there is little justification for including an attribute Office_number in the EMPLOYEE relation; rather, a relation EMP_OFFICES(Essn, Office_number) can be created to include tuples for only the employees with individual offices.

Guidelines for relational schema

- Generation of Spurious Tuples
  - Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees that no spurious tuples (which is not valid) are generated.
  - Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples.

Guidelines for relational schema

EMPLOYEE

Ename	Ssn	Bdate	Address	Dnumber
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

Guidelines for relational schema

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

No spurious tuples if join is based on PK or FK

Guidelines for relational schema

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston

Guidelines for relational schema

	Ssn	Pnumber	Hours	Pname	Plocation	Ename
	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
*	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
	666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith, John B.

Spurious tuples if join is not based on PK or FK

Functional Dependency

- A functional dependency, denoted by **X → Y**, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R.
- The constraint is that, for any two tuples t1 and t2 in r that have $t_1[X] = t_2[X]$, they must also have $t_1[Y] = t_2[Y]$.
- The values of the Y component of a tuple in r depend on, or are determined by, the values of the X component; alternatively, the values of the X component of a tuple **uniquely** (or **functionally**) determine the values of the Y component.
- Terms:
  - There is a functional dependency from X to Y, or
  - Y is functionally dependent on X.

Functional Dependency

- The abbreviation for functional dependency is **FD** or **f.d**.
- The set of attributes X is called the **left-hand side** of the FD, and Y is called the **right-hand side.**
- If a constraint on R states that there cannot be more than one tuple with a given X-value in any relation instance r(R)—that is, X is a candidate key of R—this implies that X → Y for any subset of attributes Y of R (because the key constraint implies that no two tuples in any legal state r(R) will have the same value of X)**.**
- If X → Y in R, this does not say whether or not Y → X in R.
- A functional dependency is a property of the semantics or meaning of the attributes.

Functional Dependency

- Consider the relation schema EMP_PROJ. the following functional dependencies should hold:
  a. Ssn → Ename: The value of an employee's Social Security number (Ssn) uniquely determines the employee name (Ename)
  b. Pnumber → {Pname, Plocation}: The value of a project's number (Pnumber) uniquely determines the project name (Pname) and location (Plocation)
  c. {Ssn, Pnumber} → Hours: A combination of Ssn and Pnumber values uniquely determines the number of hours the employee currently works on the project per week (Hours)
- Ename is functionally determined by (or functionally dependent on) Ssn, or given a value of Ssn

Functional Dependency

- A relation state of TEACH with a possible functional dependency TEXT → COURSE.
- However, TEACHER → COURSE, and COURSE → TEXT are ruled out.

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

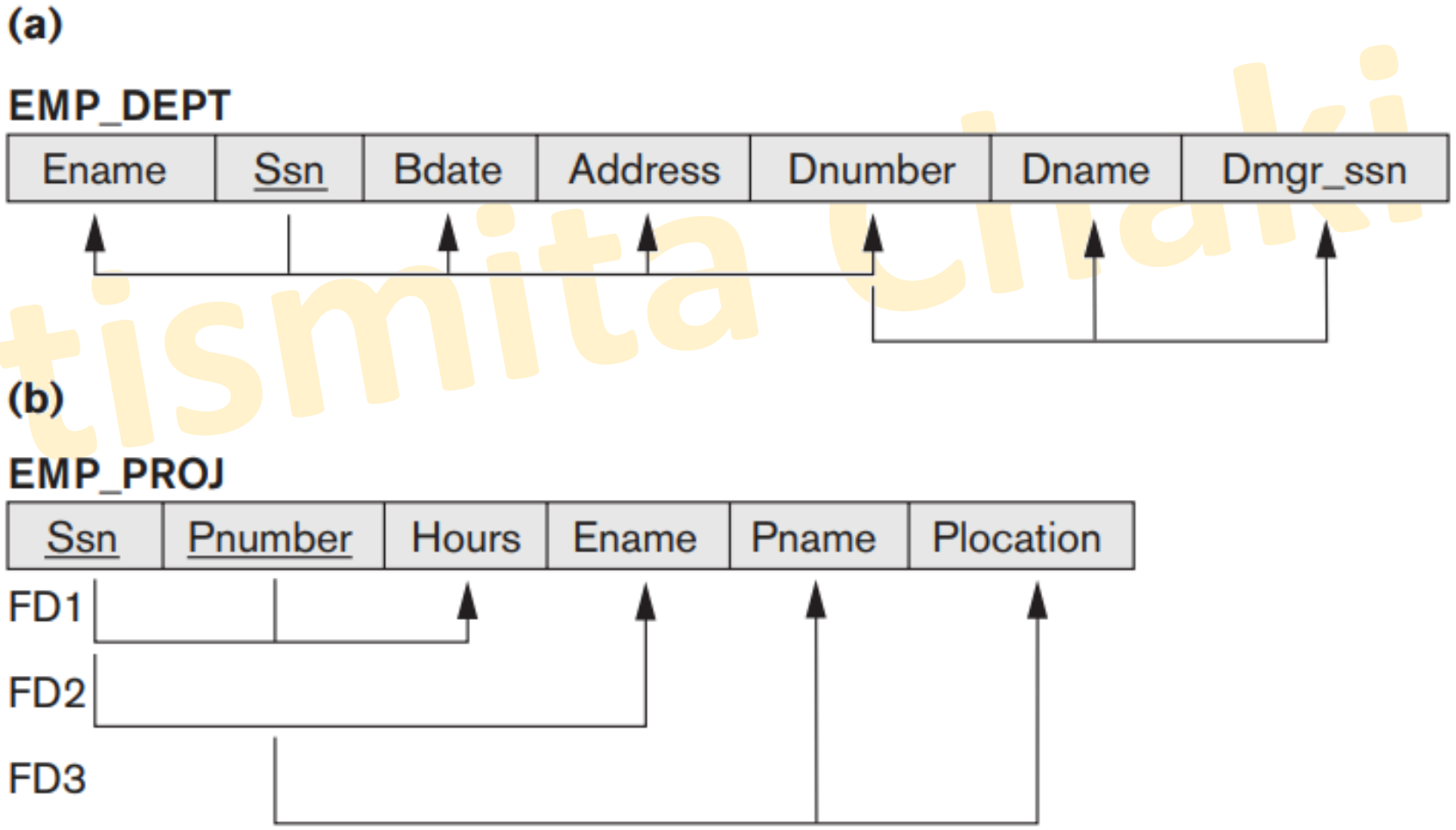
Functional Dependency

- The following FDs may hold because the four tuples in the current extension have no violation of these constraints: B → C; C → B; {A, B} → C; {A, B} → D; and {C, D} → B.
- However, the following do not hold because we already have violations of them in the given extension: A → B (tuples 1 and 2 violate this constraint); B → A (tuples 2 and 3 violate this constraint); D → C (tuples 3 and 4 violate it).

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

Functional Dependency

- Each FD is displayed as a horizontal line.
- The left-hand-side attributes of the FD are connected by vertical lines to the line representing the FD, whereas the right-hand-side attributes are connected by the lines with arrows pointing toward the attributes.



Closure of an attribute set

- The set of all those attributes which can be functionally determined from an attribute set is called as a closure of that attribute set.
- Closure of attribute set {X} is denoted as {X}+
- Following steps are followed to find the closure of an attribute set:
  1. Add the attributes contained in the attribute set for which closure is being calculated to the result set.
  2. Recursively add the attributes to the result set which can be functionally determined from the attributes already contained in the result set.

Closure of an attribute set: Example

- Consider a relation R (A , B , C , D , E , F , G) with the functional dependencies
  - A → BC
  - BC → DE
  - D → F
  - CF → G
- Now, let us find the closure of some attributes and attribute sets-

Closure of an attribute set: Example

- Closure of attribute A:
  - A + = { A }
    = { A , B , C } (Using A → BC)
    = { A , B , C , D , E } (Using BC → DE)
    = { A , B , C , D , E , F } (Using D → F)
    = { A , B , C , D , E , F , G } (Using CF → G)

  Thus, A + = { A , B , C , D , E , F , G }
- Closure of attribute D:
  - D + = { D }
    = { D , F } (Using D → F)

  We can not determine any other attribute using attributes D and F contained in the result set.

  Thus, D + = { D , F }

Closure of an attribute set: Example

- Closure of attribute set {B, C}:
  - { B , C }+ = { B , C }
    = { B , C , D , E } (Using BC → DE)
    = { B , C , D , E , F } (Using D → F)
    = { B , C , D , E , F , G } (Using CF → G)

  Thus, { B , C }+ = { B , C , D , E , F , G }

Finding the Keys Using Closure

- Super Key
  - If the closure result of an attribute set contains all the attributes of the relation, then that attribute set is called as a super key of that relation.
  - Example:
    - In the above example, The closure of attribute A is the entire relation schema. Thus, attribute **A is a super key** for that relation.
- Candidate Key
  - If there exists no subset of an attribute set whose closure contains all the attributes of the relation, then that attribute set is called as a candidate key of that relation.
  - Example:
    - In the above example, No subset of attribute A contains all the attributes of the relation. Thus, attribute A is also a candidate key for that relation.

Finding the Keys Using Closure

- **Question:** Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, M} and the set of functional dependencies
  - {E, F} -> {G},
  - {F} -> {I, J},
  - {E, H} -> {K, L},
  - K -> {M},
  - L -> {N} on R. What is the key for R?

Answer: Finding attribute closure of all given options, we get:

{E,F}+ = {EFGIJ}

{E,F,H}+ = {EFHGIJKLMN}

{E,F,H,K,L}+ = {{EFHGIJKLMN}

{E}+ = {E}

{EFH}+ and {EFHKL}+ results in set of all attributes, but EFH is minimal. So it will be candidate key.

Finding the Keys Using Closure

- **Question:** Consider a relation scheme R = (A, B, C, D, E, H) on which the following functional dependencies hold: {A–>B, BC–> D, E–>C, D–>A}. What are the candidate keys of R?
  (a) AE, BE
  (b) AE, BE, DE
  (c) AEH, BEH, BCH
  (d) AEH, BEH, DEH

Answer: (AE)+ = {ABECD} which is not set of all attributes. So AE is not a candidate key. Hence option A and B are wrong.
(AEH)+ = {ABCDEH}
(BEH)+ = {BEHCDA}
(BCH)+ = {BCHDA} which is not set of all attributes. So BCH is not a candidate key. Hence option C is wrong.
So correct answer is D.

Finding the Keys Using Closure

- Let R = (A, B, C, D, E, F) be a relation scheme with the following dependencies
  - C → F
  - E → A
  - EC → D
  - A → B
- Which is a key for R?
- Also, determine the total number of candidate keys and super keys.

Finding the Keys Using Closure

- Determine all essential attributes of the given relation.
- Essential attributes are those attributes which are not present on RHS of any functional dependency.
- Essential attributes are always a part of every candidate key.
- This is because they can not be determined by other attributes.
- The remaining attributes of the relation are non-essential attributes.
- This is because they can be determined by using essential attributes.

Finding the Keys Using Closure

- Essential attributes of the relation are- C and E.
- So, attributes C and E will definitely be a part of every candidate key.
- Now,
  - We will check if the essential attributes together can determine all remaining non-essential attributes.
  - To check, we find the closure of CE.
    - { CE }+ = { C , E }
      = { C , E , F } (Using C → F)
      = { A , C , E , F } (Using E → A)
      = { A , C , D , E , F } (Using EC → D)
      = { A , B , C , D , E , F } (Using A → B)

We conclude that CE can determine all the attributes of the given relation. So, CE is the only possible candidate key of the relation.

Finding FD from a given FD

- To check whether an FD A->B can be derived from an FD set F,
  1. Find (A)+ using FD set F.
  2. If B is subset of (A)+, then A->B is true else not true.

Question: In a schema with attributes A, B, C, D and E following set of functional dependencies are given
{A -> B, A -> C, CD -> E, B -> D, E -> A}
Which of the following functional dependencies is NOT implied by the above set?

A. CD -> AC
B. BD -> CD
C. BC -> CD
D. AC -> BC

Answer: Using FD set given in question,
(CD)+ = {CDEAB} which means CD -> AC also holds true.
(BD)+ = {BD} which means BD -> CD can't hold true. So this FD is no implied in FD set.
(BC)+ = {BCDEA} which means BC -> CD also holds true.
(AC)+ = {ACBDE} which means AC -> BC also holds true.

Normalization

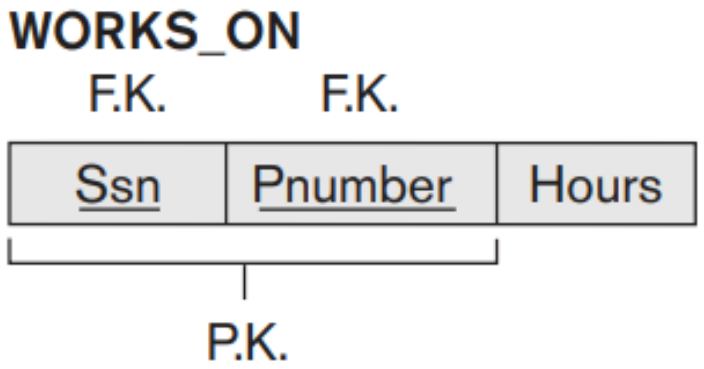
- The normalization process, as first proposed by Codd (1972a), takes a relation schema through a series of tests to certify whether it satisfies a certain normal form.
- The process, which proceeds in a top-down fashion by evaluating each relation against the criteria for normal forms and decomposing relations as necessary, can thus be considered as relational design by analysis.
- Initially, Codd proposed three normal forms, which he called first, second, and third normal form.
- A stronger definition of 3NF—called Boyce-Codd normal form (BCNF)—was proposed later by Boyce and Codd.
- All these normal forms are based on a single analytical tool: the functional dependencies among the attributes of a relation.
- Later, a fourth normal form (4NF) and a fifth normal form (5NF) were proposed, based on the concepts of multivalued dependencies and join dependencies, respectively

Normalization

- **Normalization of data** can be considered a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of
  - (1) minimizing redundancy and
  - (2) minimizing the insertion, deletion, and update anomalies
- It can be considered as a “filtering” or “purification” process to make the design have successively better quality.
- The **normal form** of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized.

Normalization

- An attribute of relation schema R is called a **prime attribute** of R if it is a member of some candidate key of R.
- An attribute is called nonprime if it is **not a prime attribute**—that is, if it is not a member of any candidate key.
- For example, both Ssn and Pnumber are prime attributes of WORKS_ON, whereas other attributes of WORKS_ON are nonprime.

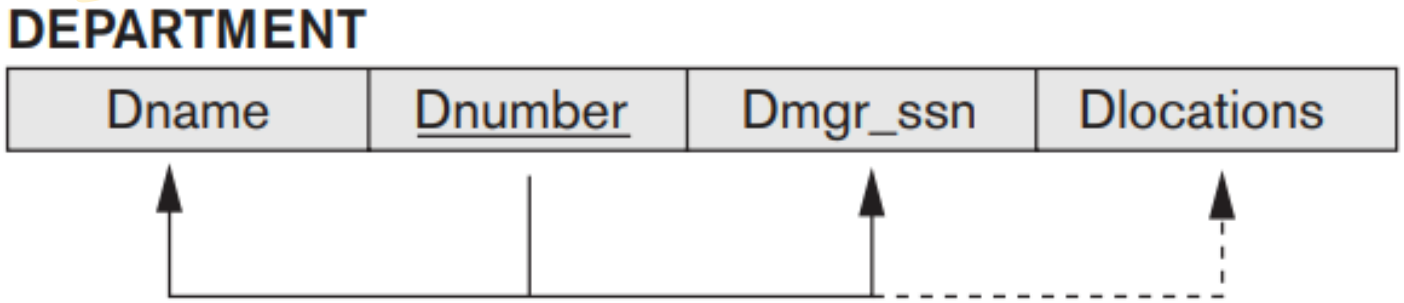


Normalization: First normal form (1NF)

- The domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute.
- Hence, 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a single tuple.
- In other words, 1NF disallows relations within relations or relations as attribute values within tuples.
- The only attribute values permitted by 1NF are **single atomic** (or **indivisible**) values.

Normalization: First normal form (1NF)

- There are two ways we can look at the Dlocations attribute:
  - The domain of Dlocations contains atomic values, but some tuples can have a set of these values. In this case, Dlocations is not functionally dependent on the primary key Dnumber.
  - The domain of Dlocations contains sets of values and hence is nonatomic. In this case, Dnumber → Dlocations because each set is considered a single member of the attribute domain.



Normalization: First normal form (1NF)

- There are three main techniques to achieve first normal form for such a relation:
  - Remove the attribute Dlocations that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key Dnumber of DEPARTMENT. The primary key of this newly formed relation is the combination {Dnumber, Dlocation}. A distinct tuple in DEPT_LOCATIONS exists for each location of a department. This decomposes the non-1NF relation into two 1NF relations.

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Normalization: First normal form (1NF)

- There are three main techniques to achieve first normal form for such a relation:
  - Expand the key so that there will be a separate tuple in the original DEPARTMENT relation for each location of a DEPARTMENT. In this case, the primary key becomes the combination {Dnumber, Dlocation}. This solution has the disadvantage of introducing redundancy in the relation and hence is rarely adopted.

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Normalization: First normal form (1NF)

- There are three main techniques to achieve first normal form for such a relation:
  - If a maximum number of values is known for the attribute—for example, if it is known that at most three locations can exist for a department—replace the Dlocations attribute by three atomic attributes: Dlocation1, Dlocation2, and Dlocation3. This solution has the disadvantage of introducing NULL values if most departments have fewer than three locations.
- Of the three solutions above, the first is generally considered best because it does not suffer from redundancy and it is completely general; it places no maximum limit on the number of values.

Normalization: First normal form (1NF)

- First normal form also disallows multivalued attributes that are themselves composite.
- These are called **nested relations** because each tuple can have a relation within it.
  - EMP_PROJ(Ssn, Ename, {PROJS(Pnumber, Hours)})
  - Ssn is the primary key of the EMP_PROJ relation, whereas Pnumber is the partial key of the nested relation; that is, within each tuple, the nested relation must have unique values of Pnumber.
- To normalize this into 1NF, we remove the nested relation attributes into a new relation and propagate the primary key into it; the primary key of the new relation will combine the partial key with the primary key of the original relation.

Normalization: First normal form (1NF)

(a)

EMP_PROJ

		Projs	
Ssn	Ename	Pnumber	Hours

(b)

EMP_PROJ

Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0

(c)

EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a nested relation attribute PROJS. (b) Sample extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

Normalization: First normal form (1NF)

- Ex:
  - CANDIDATE (Ssn, Name, {JOB_HIST (Company, Highest_position, {SAL_HIST (Year, Max_sal)})})
- After 1NF
  - CANDIDATE_1 (Ssn, Name)
  - CANDIDATE_JOB_HIST (Ssn, Company, Highest_position)
  - CANDIDATE_SAL_HIST (Ssn, Company, Year, Max-sal)

Normalization: First normal form (1NF)

- Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD_PHONE. Its decomposition into 1NF has been shown in table 2.

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

Normalization: First normal form (1NF)

TABLE_PRODUCT

Product ID	Color	Price
1	red, green	15.99
2	yellow	23.99
3	green	17.50
4	yellow, blue	9.99
5	red	29.99

TABLE_PRODUCT_PRICE

Product ID	Price
1	15.99
2	23.99
3	17.50
4	9.99
5	29.99

TABLE_PRODUCT_COLOR

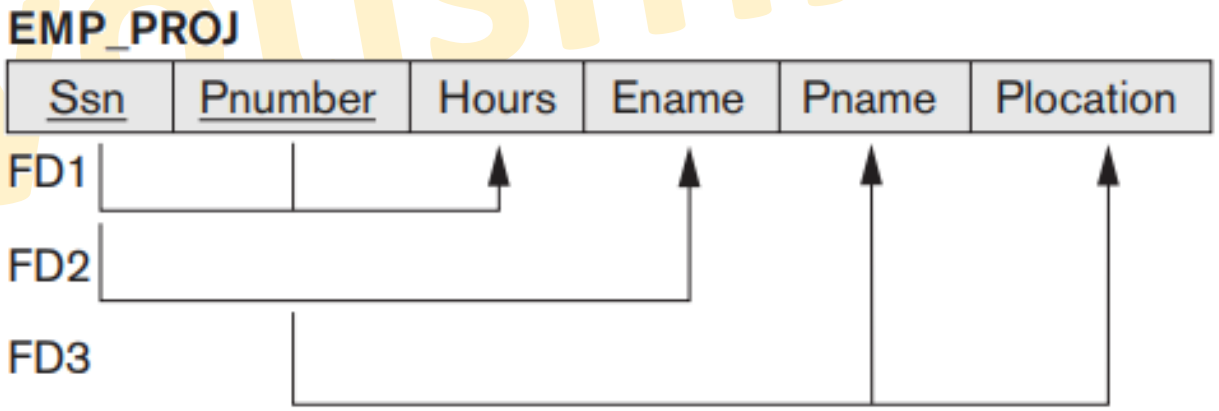
Product ID	Color
1	red
1	green
2	yellow
3	green
4	yellow
4	blue
5	red

Normalization: Second normal form (2NF)

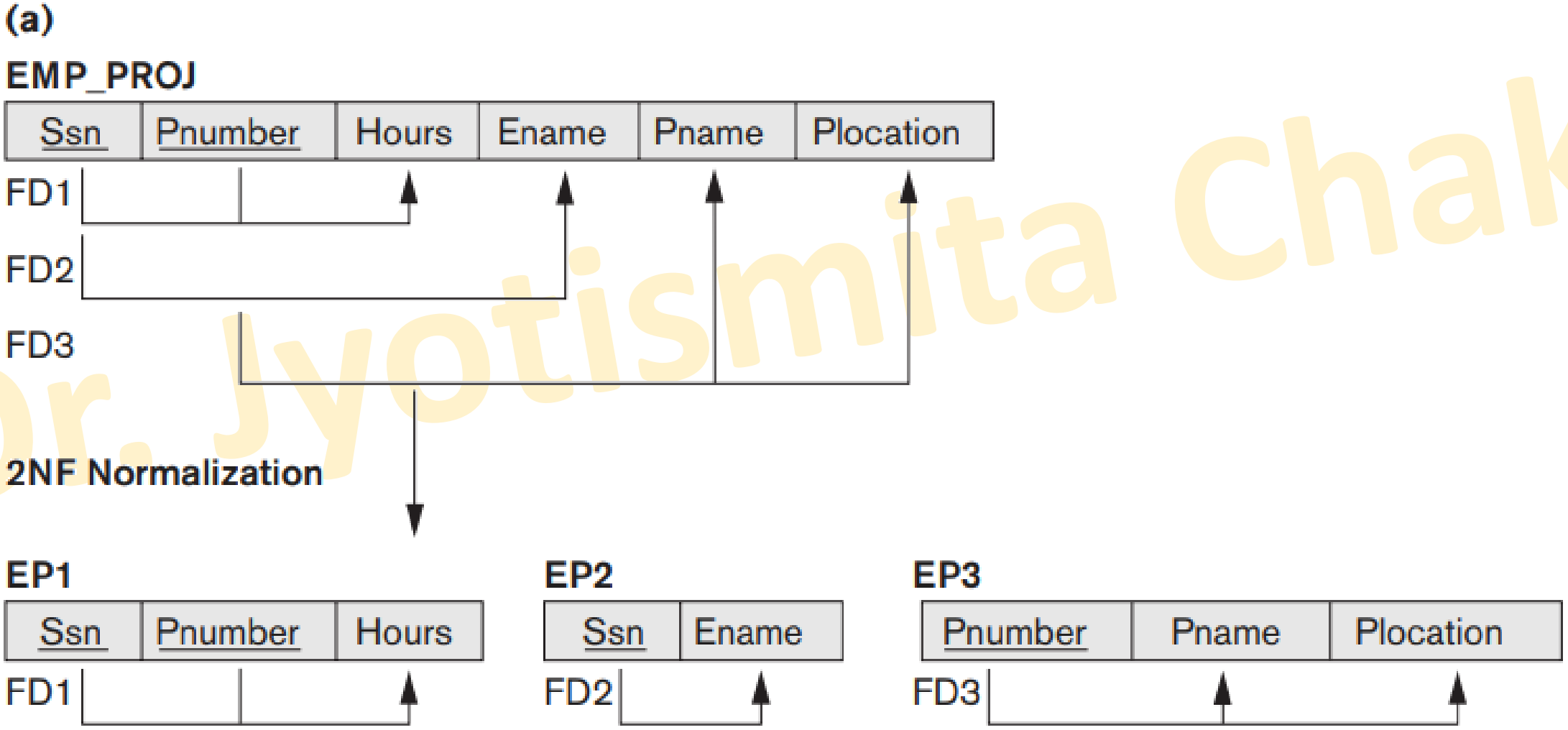
- **Second normal form (2NF)** is based on the concept of full functional dependency.
- A functional dependency X → Y is a **full functional dependency** if removal of any attribute A from X means that the dependency does not hold anymore.
- A functional dependency X → Y is a partial dependency if some attribute A ε X can be removed from X and the dependency still holds; that is, for some A ε X, (X − {A}) → Y.
- **Definition:** A relation schema R is in 2NF, only if a relation is in 1NF and if every nonprime attribute A in R is fully functionally dependent on the primary key of R.

Normalization: Second normal form (2NF)

- {Ssn, Pnumber} → Hours is a **full dependency** (neither Ssn → Hours nor Pnumber → Hours holds). However, the dependency {Ssn, Pnumber} → Ename is **partial** because Ssn → Ename holds



Normalization: Second normal form (2NF)



Normalization: Second normal form (2NF)

- In the above table, we have partial dependency:
  - The prime key attributes are **StudentID** and **ProjectID**.
  - The **StudentName** can be determined by **StudentID**, which makes the relation Partial Dependent.
  - The **ProjectName** can be determined by **ProjectID**, which makes the relation Partial Dependent.
- Therefore, the <**StudentProject**> relation violates the 2NF in Normalization and is considered a bad database design.

StudentID	**ProjectID**	**StudentName**	**ProjectName**
S89	P09	Olivia	Geo Location
S76	P07	Jacob	Cluster Exploration
S56	P03	Ava	IoT Devices
S92	P05	Alexandra	Cloud Deployment

Normalization: Second normal form (2NF)

- To remove Partial Dependency and violation on 2NF, decompose the above tables
- **StudentInfo**

StudentID	StudentName
S89	Olivia
S76	Jacob
S56	Ava
S92	Alexandra

ProjectInfo

ProjectID	ProjectName
P09	Geo Location
P07	Cluster Exploration
P03	IoT Devices
P05	Cloud Deployment

Stu_ProInfo

StudentID	ProjectID
S89	P09
S76	P07
S56	P03
S92	P05

Normalization: Second normal form (2NF)

- Suppose a following stu_proj relational schema:

stu_id	proj_id	stu_name	proj_name	proj_lang_used
s101	p001	Rakesh	Online Chatting	python
s102	p001	Kritika	Online Chatting	python
s102	p002	Kritika	Text Editor	Java
s103	p002	Mahesh	Text Editor	Java
s104	p002	Ram	Text Editor	Java
s104	p003	Ram	Online Shopping	PHP

- Identify functional dependencies in above table and find out that relation is in 2NF or not? If not decompose it in 2NF. (A student can work on many projects and a project can have many students associated with it.)

Normalization: Second normal form (2NF)

- The table is in 1NF because each attribute in the table have atomic (single) value.
- The following FDs are identified based on the value and descriptions given about table:
  - stu_id → stu_name (Student name can be determined by student id)
  - proj_id → proj_name, proj_lang_used (Project name and language used can be determined by project id)
- Since (stu_id, proj_id) uniquely identifies each record in the table, so, (stu_id, proj_id) is the candidate key in table.

Normalization: Second normal form (2NF)

- **Prime attributes** - stu_id, proj_id (because these are the part of the candidate key)
- **Non prime attributes** - stu_name, proj_name, proj_lang_used (because these are not part of the candidate key)
- Now, the partial dependency exists in the table, it means violation of rule of 2NF because non prime attribute is dependent on part of the candidate key.
  - stu_id → stu_name
  - proj_id → proj_name, proj_lang_used

both FD violating rule of 2NF, since (stu_id, proj_id) is the candidate key)

Normalization: Second normal form (2NF)

- Therefore, to convert the relation in 2NF, It is divided into three relations:
  - student (stu_id, stu_name) [Since stu_id → stu_name]
  - project (proj_id, proj_name, proj_lang_used [Since proj_id → proj_name, proj_lang_used]
  - stu_proj_alloc (stu_id, proj_id) [Since stu_id, proj_id is candidate key]
- Above three tables (student, project, stu_proj_alloc) are following the rules of 2NF.

Normalization: Second normal form (2NF)

- See, how the data redundancy is minimized after converting table into 2NF (compare it with original table):

table1: **student**

stu_id	stu_name
s101	Rakesh
s102	Kritika
s103	Mahesh
s104	Ram

table2: **project**

proj_id	proj_name	proj_lang_used
p001	Online Chatting	python
p002	Text Editor	Java
p003	Online Shopping	PHP

table3: **stu_proj_alloc**

stu_id	proj_id
s101	p001
s102	p001
s102	p002
s103	p002
s104	p002
s104	p003

Normalization: Second normal form (2NF)

- **Candidate Keys**: {teacher_id, subject}
- **Non prime attribute**: teacher_age
- The table is in 1 NF because each attribute has atomic values.
- However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table".

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Normalization: Second normal form (2NF)

- To make the table complies with 2NF we can break it in two tables like this:

teacher_details table:

teacher_id	teacher_age
111	38
222	38
333	40

teacher_subject table:

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Normalization: 1NF and 2NF of video library DB

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.

PK

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.

Normalization: 1NF and 2NF of video library DB

2NF

Customer Table

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

Customer_Rent Table

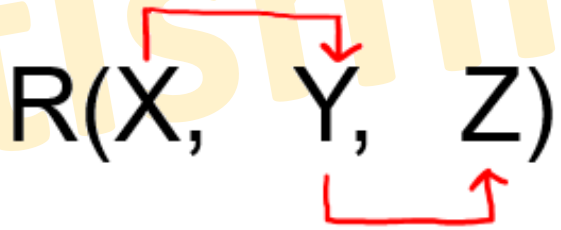
FULL NAMES	MOVIES RENTED
Janet Jones	Pirates of the Caribbean
Janet Jones	Clash of the Titans
Robert Phil	Forgetting Sarah Marshal
Robert Phil	Daddy's Little Girls
Robert Phil	Clash of the Titans

Normalization: Third normal form (3NF)

- **Third normal form (3NF)** is based on the concept of **transitive dependency**.
- A functional dependency X → Y in a relation schema R is a **transitive dependency** if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and both X → Z and Z → Y hold.
- **Definition 1:** According to Codd's original definition, a relation schema R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key.
- **Definition 2:** First it should be in 2NF and if there exists a non-trivial dependency between two sets of attributes X and Y such that X → Y (i.e., Y is not a subset of X) then
  1. Either X is Super Key
  2. Or Y is a prime attribute.
- Advantages:
  - Amount of data duplication is reduced.
  - Data integrity achieved.

Normalization: Third normal form (3NF)

- **Qs:** Given a relation R(X, Y, Z) and Functional Dependency set FD = { X → Y and Y → Z }, determine whether the given R is in 3NF? If not convert it into 3 NF.
- **Solution:** Let us construct an arrow diagram on R using FD to calculate the candidate key.



- **Let us calculate the closure of X**
- X + = XYZ (from the closure method we studied earlier)
- Since the closure of X contains all the attributes of R, hence **X is Candidate Key**

Normalization: Third normal form (3NF)

- **Now check the above table is in 2 NF.**
  1. FD: X → Y is in 2NF (as Key is not breaking and its Fully functional dependent)
  2. FD: Y → Z is also in 2NF(as it does not violate the definition of 2NF)
- **Hence above table R(X, Y, Z) is in 2NF.**
- **Let's check it for 3NF:**
  1. FD: X → Y is in 3NF (as X is a super Key)
  2. FD: Y → Z is not in 3NF (as neither Y is Key nor Z is a prime attribute)
- **Hence because of Y → Z, we can say that above table R is not in 3NF.**
- **Convert the table R(X, Y, Z) into 3NF:**
  - R1(X, Y)
  - R2(Y, Z)

Normalization: Third normal form (3NF)

- Qs: Given a relation R(X, Y, Z, W, P) and Functional Dependency set FD = { X → Y, Y → P, and Z → W}, determine whether the given R is in 3NF? If not convert it into 3 NF.
- Solution: We can see that an attributes XZ is not determined by any of the given FD, hence XZ will be the integral part of the Candidate key, i.e. no matter what will be the candidate key, and how many will be the candidate key, but all will have XZ compulsory attribute.
- **Let us calculate the closure of XZ**
  - XZ + = XZYPW (from the closure method that we studied earlier)
  - Since the closure of XZ contains all the attributes of R, hence **XZ is Candidate Key**

Normalization: Third normal form (3NF)

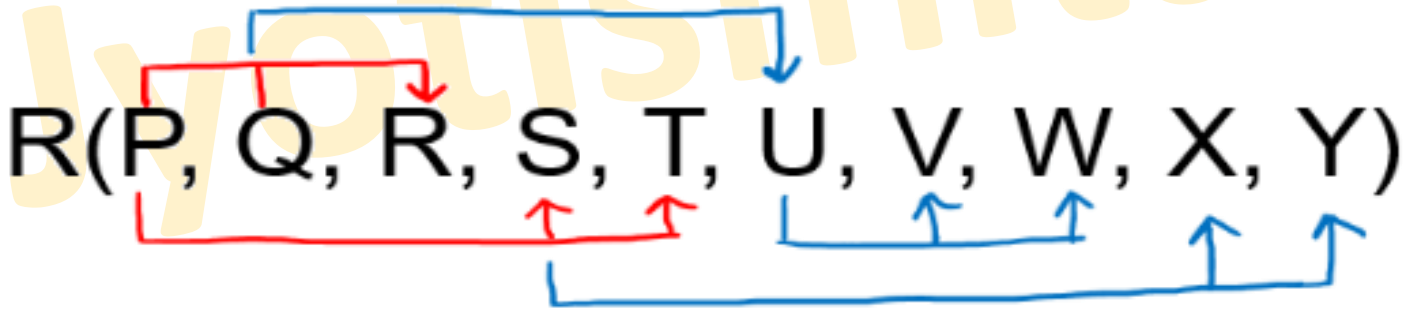
- Since R has 5 attributes: - X, Y, Z, W, P and Candidate Key is XZ, Therefore, prime attribute (part of candidate key) are X and Z while a non-prime attribute are Y, W, and P.
- Given FD are X → Y, Y → P, and Z → W and Super Key / Candidate Key is XZ
  1. FD: **X → Y** does not satisfy the definition of 3NF, that neither X is Super Key nor Y is a prime attribute.
  2. FD: **Y → P** does not satisfy the definition of 3NF, that neither Y is Super Key nor P is a prime attribute.
  3. FD: **Z → W** does not satisfy the definition of 3NF, that neither Z is Super Key nor W is a prime attribute.

Normalization: Third normal form (3NF)

- Convert the table R(X, Y, Z, W, P) into 3NF:
  - **Since all the FD =** { X → Y, Y → P, and Z → W} were not in 3NF, let us convert R in 3NF
    - **R1(X, Y)** {Using FD X → Y}
    - **R2(Y, P)** {Using FD Y → P}
    - **R3(Z, W)** {Using FD Z → W}
  - And create one table for Candidate Key XZ
  - **R4(X, Z)** { Using Candidate Key XZ }
  - All the decomposed tables R1, R2, R3, and R4 are in 2NF(as there is no partial dependency) as well as in 3NF.
  - Hence decomposed tables are: **R1(X, Y), R2(Y, P), R3(Z, W),** and **R4(X, Z)**

Normalization: Third normal form (3NF)

- Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Functional Dependency set FD = { PQ → R, P → ST, Q → U, U → VW, and S → XY}, determine whether the given R is in 3NF? If not convert it into 3 NF.



Normalization: Third normal form (3NF)

- We can see that an attribute PQ is not determined by any of the given FD, hence PQ will be the integral part of the Candidate key, i.e. no matter what will be the candidate key, and how many will be the candidate key, but all will have PQ compulsory attribute.
- **Let us calculate the closure of PQ**
  - PQ + = P Q R S T U X Y V W (from the closure method we studied earlier)
- Since the closure of XZ contains all the attributes of R, hence **PQ is Candidate Key**
- Since R has 10 attributes: - P, Q, R, S, T, U, V, W, X, Y, V, W and Candidate Key is PQ, Therefore, prime attribute (part of candidate key) are P and Q while a non-prime attribute are R S T U V W X Y V W

Normalization: Third normal form (3NF)

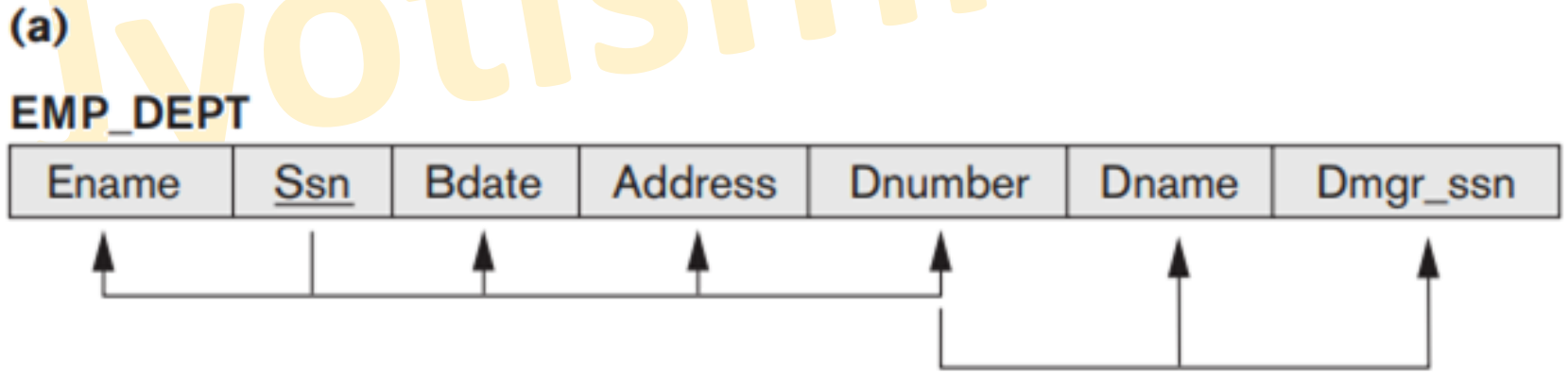
- Given FD are {PQ → R, P → ST, Q → U, U → VW and S → XY} and Super Key / Candidate Key is PQ
  1. FD: **PQ → R** satisfy the definition of 3NF, as PQ Super Key
  2. FD: **P → ST** does not satisfy the definition of 3NF, that neither P is Super Key nor ST is the prime attribute
  3. FD: **Q → U** does not satisfy the definition of 3NF, that neither Q is Super Key nor U is a prime attribute
  4. FD: **U → VW** does not satisfy the definition of 3NF, that neither U is Super Key nor VW is a prime attribute
  5. FD: **S → XY** does not satisfy the definition of 3NF, that neither S is Super Key nor XY is a prime attribute

Normalization: Third normal form (3NF)

- Convert the table R(X, Y, Z, W, P) into 3NF:
  - **Since all the FD =** { P → ST, Q → U, U → VW, and S → XY } were not in 3NF, let us convert R in 3NF
    - **R1(P, S, T)** {Using FD P → ST }
    - **R2(Q, U)** {Using FD Q → U }
    - **R3(U, V, W)** { Using FD U → VW }
    - **R4(S, X, Y)** { Using FD S → XY }
    - **R5(P, Q, R)** { Using FD PQ → R, and candidate key PQ }
  - All the decomposed tables R1, R2, R3, R4, and R5 are in 2NF(as there is no partial dependency) as well as in 3NF.
  - **Hence decomposed tables are: R1(P, S, T), R2(Q, U), R3(U, V, W), R4(S, X, Y),** and **R5(P, Q, R)**

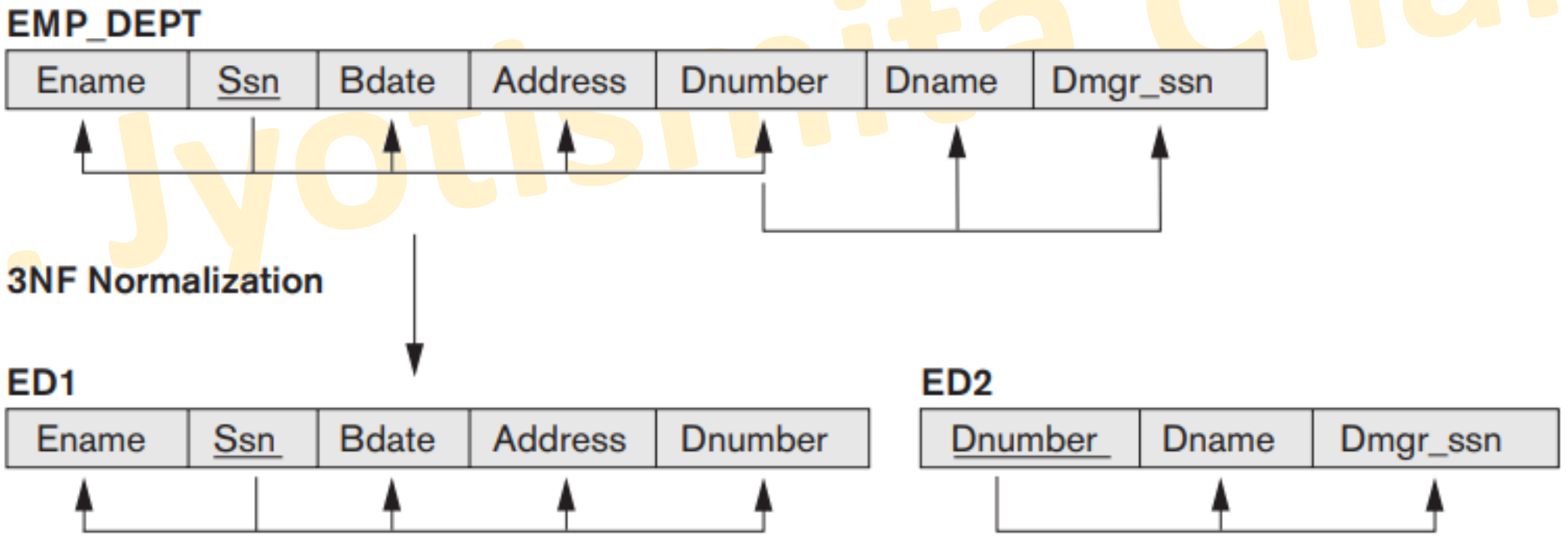
Normalization: Third normal form (3NF)

- The dependency Ssn → Dmgr_ssn is transitive through Dnumber in EMP_DEPT, because both the dependencies Ssn → Dnumber and Dnumber → Dmgr_ssn hold and Dnumber is neither a key itself nor a subset of the key of EMP_DEPT.



Normalization: Third normal form (3NF)

- We can normalize EMP_DEPT by decomposing it into the two 3NF relation schemas ED1 and ED2



Normalization: Third normal form (3NF)

- Suppose a school wants to store the address of each student, they create a table named **student_details** that looks like:

Rollno	State	City
1	Punjab	Chandigarh
2	Haryana	Ambala
3	Punjab	Chandigarh
4	Haryana	Ambala
5	Uttar Pradesh	Ghaziabad

- **Candidate Key**: {Rollno}
  Prime attribute: Rollno
  Non-prime attribute: {State, City}

Normalization: Third normal form (3NF)

- The above relation is not in third normal form, because as a rule says, there should be no transitive functional dependency in the relation.
- Here, **City** (a non-prime attribute) depends on **State** (a non-prime attribute), and **State** depends on **Rollno**.
- The non-prime attributes (State, City) are transitively dependent on the candidate key(Rollno).
- Thus, it violates the rule of third normal form.

Normalization: Third normal form (3NF)

- To covert the relation in 3NF, you have to decompose the relation as:

Table: Student_state

Rollno	State
1	Punjab
2	Haryana
3	Punjab
4	Haryana
5	Uttar Pradesh

Table:Student_city

State	City
Punjab	Chandigarh
Haryana	Ambala
Uttar Pradesh	Ghaziabad

Normalization: Third normal form (3NF)

- **EMPLOYEE_DETAIL table:**

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

- **Candidate key:** {EMP_ID}
- **Non-prime attributes:** In the given table, all attributes except EMP_ID are non-prime.

Normalization: Third normal form (3NF)

- Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID.
- The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.
- That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

Normalization: Third normal form (3NF)

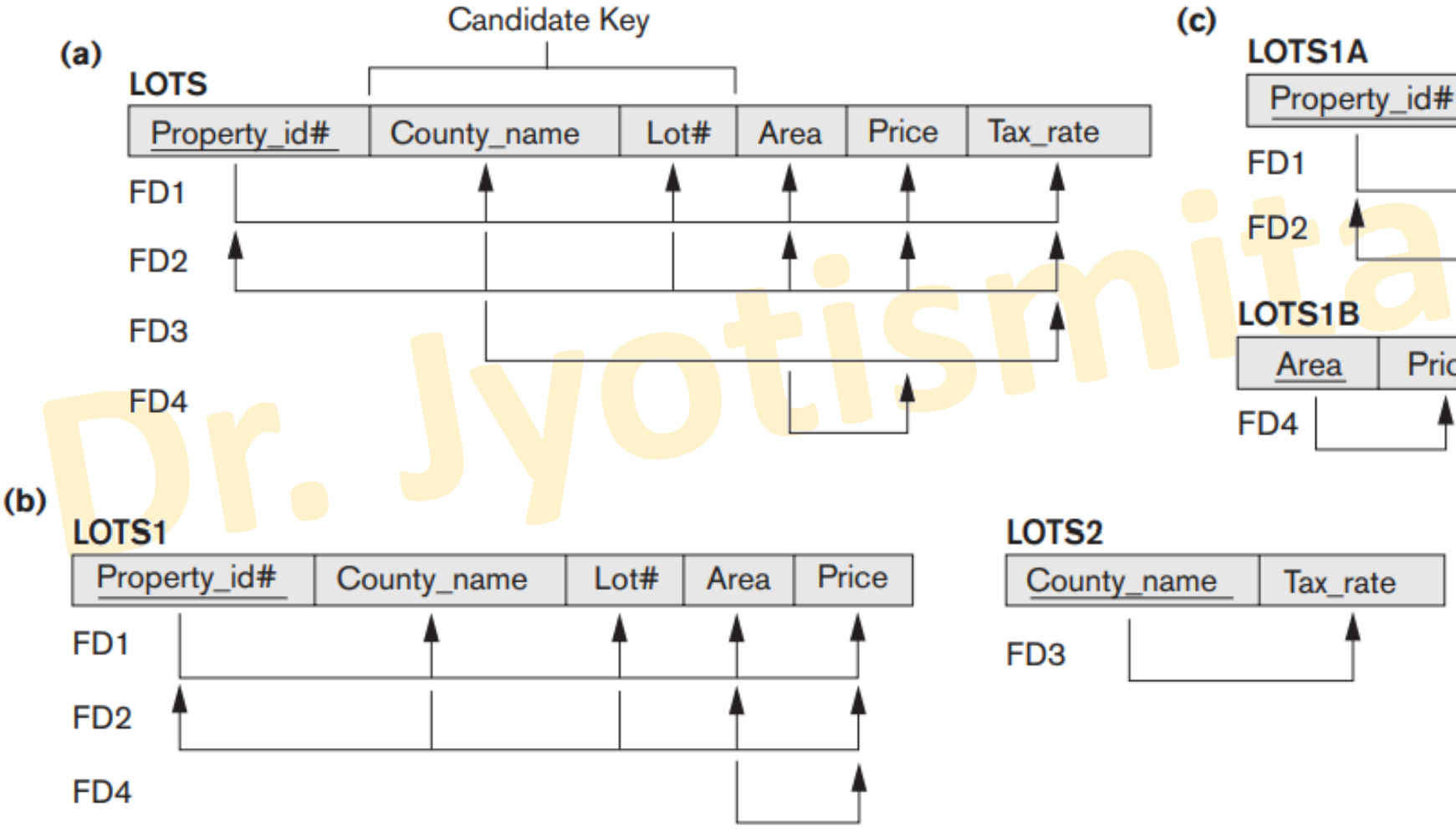
EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

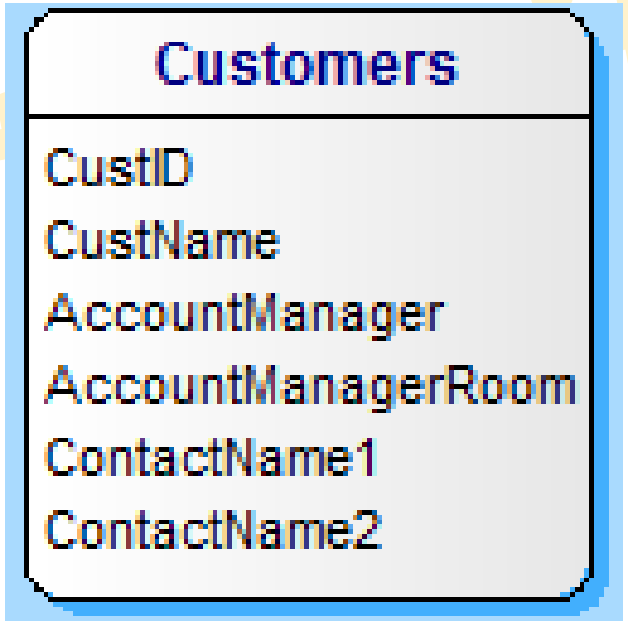
Normalization: Example



Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B.

Normalization: From 0NF to 3NF: 0NF

- Not normalized (0NF) table/entity in a data model
- A Customer has multiple Contacts (contact1, contact2,...).



Normalization: From 0NF to 3NF: 1NF

CustID	CustName	AccountManager	AccountManagerRoom	ContactID
171	ABNAmro	Hans	12	1
171	ABNAmro	Hans	12	2
190	RaboBank	Guus	15	3
190	RaboBank	Guus	15	4

Customers.

ContactID	ContactName
1	Piet
2	Koos
3	Mona
4	Mieke

Contacts.

Normalization: From 0NF to 3NF: 2NF

- The key is CustID + ContactID. If we remove the ContactID attribute we ensure that all attributes in the Customer entity depend on the new primary key CustID.

CustID	CustName	AccountManager	AccountManagerRoom
171	ABNAmro	Hans	12
190	RaboBank	Guus	15

Customers.

CustID	ContactID
171	1
171	2
190	3
190	4

CustomerContacts

ContactID	ContactName
1	Piet
2	Koos
3	Mona
4	Mieke

Contacts

Normalization: From 0NF to 3NF: 3NF

- Attribute AccountManagerRoom is transitive dependent on attribute AccountManager. A new entity AccountManager must be created so that we can remove attribute AccountManagerRoom from the Customer entity.

CustID	CustName	AccountManagerID
171	ABNAmro	1
190	RaboBank	2

Customers.

CustID	ContactID
171	1
171	2
190	3
190	4

CustomerContacts

ContactID	ContactName
1	Piet
2	Koos
3	Mona
4	Mieke

Contacts

AccountManagerID	AccountManager	AccountManagerRoom
1	Hans	12
2	Guus	15

AccountManagers

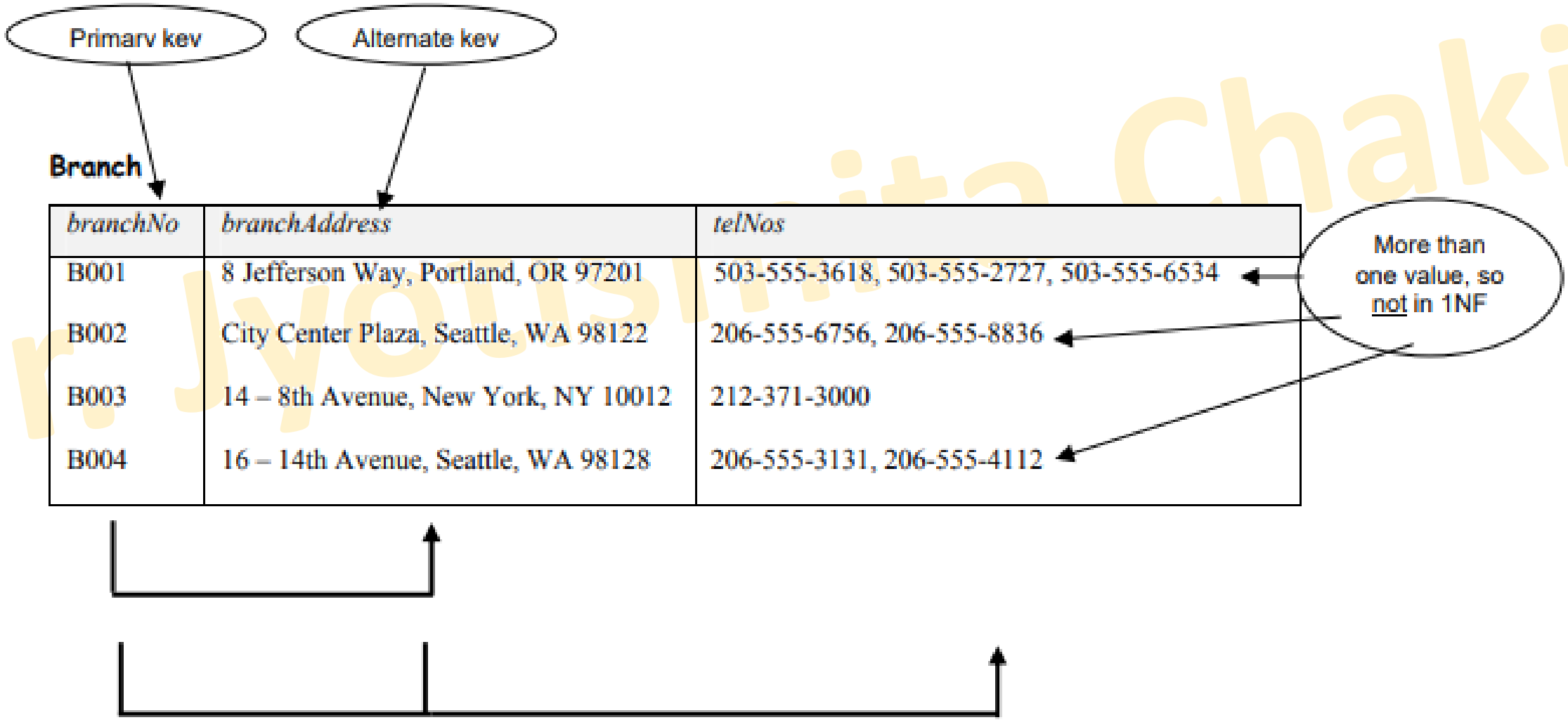
Normalization: Example

- Examine the table shown below.

branchNo	*branchAddress*	*telNos*
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618, 503-555-2727, 503-555-6534
B002	City Center Plaza, Seattle, WA 98122	206-555-6756, 206-555-8836
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131, 206-555-4112

- Why is this table not in 1NF?
- Describe and illustrate the process of normalizing the data shown in this table to third normal form (3NF).
- Identify the primary, alternate and foreign keys in your 3NF relations.

Normalization: Example



Normalization: Example

Branch

branchNo	*branchAddress*
B001	8 Jefferson Way, Portland, OR 97201
B002	City Center Plaza, Seattle, WA 98122
B003	14 – 8th Avenue, New York, NY 10012
B004	16 – 14th Avenue, Seattle, WA 98128

Primary key

Alternate key

BranchTelephone

branchNo	*telNo*
B001	503-555-3618
B001	503-555-2727
B001	206-555-6534
B002	206-555-6756
B002	206-555-8836
B003	212-371-3000
B004	206-555-3131
B004	206-555-4112

Becomes foreign key

Becomes primary key

Normalization: Example

- Examine the table shown below.

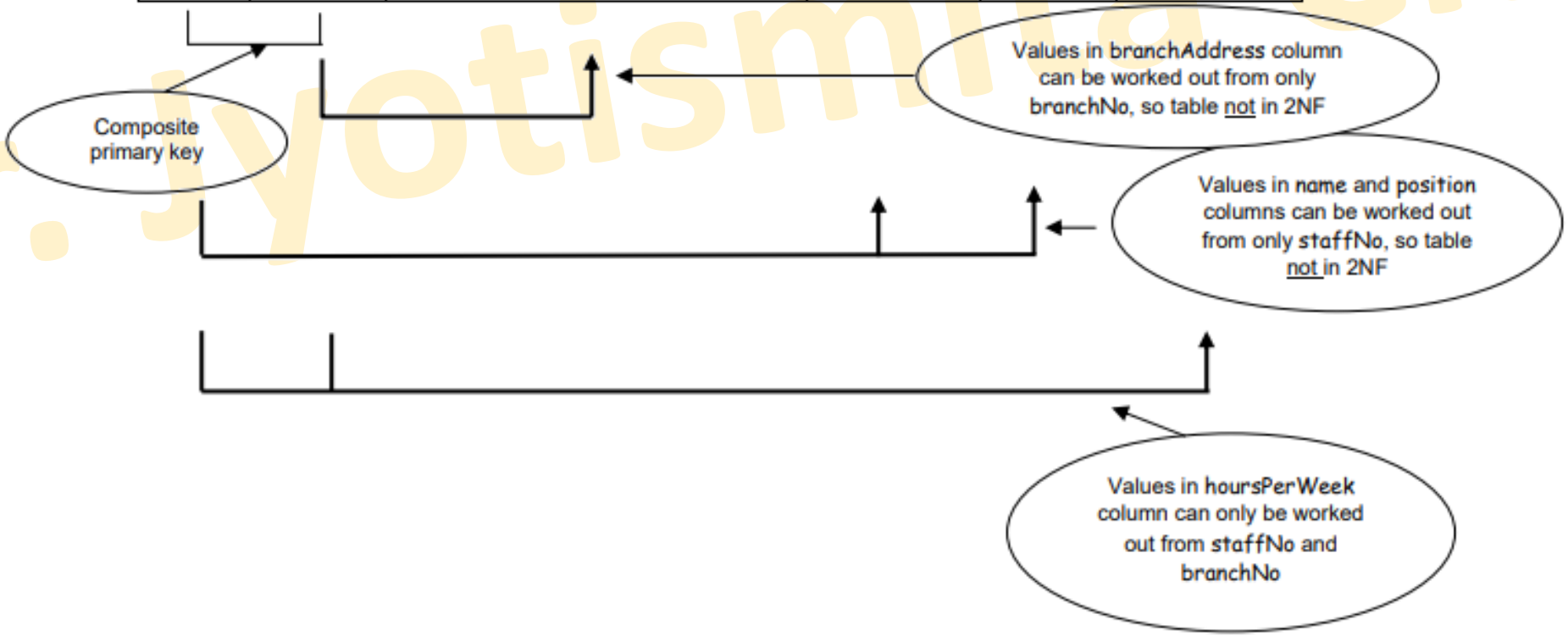
staffNo	*branchNo*	*branchAddress*	*name*	*position*	*hoursPerWeek*
S4555	B002	City Center Plaza, Seattle, WA 98122	Ellen Layman	Assistant	16
S4555	B004	16 – 14th Avenue, Seattle, WA 98128	Ellen Layman	Assistant	9
S4612	B002	City Center Plaza, Seattle, WA 98122	Dave Sinclair	Assistant	14
S4612	B004	16 – 14th Avenue, Seattle, WA 98128	Dave Sinclair	Assistant	10

- Why is this table not in 2NF?
- Describe and illustrate the process of normalizing the data shown in this table to third normal form (3NF).
- Identify the primary, (alternate) and foreign keys in your 3NF relations.

Normalization: Example

TempStaffAllocation

staffNo	*branchNo*	*branchAddress*	*name*	*position*	*hoursPerWeek*
S4555	B002	City Center Plaza, Seattle, WA 98122	Ellen Layman	Assistant	16
S4555	B004	16 – 14th Avenue, Seattle, WA 98128	Ellen Layman	Assistant	9
S4612	B002	City Center Plaza, Seattle, WA 98122	Dave Sinclair	Assistant	14
S4612	B004	16 – 14th Avenue, Seattle, WA 98128	Dave Sinclair	Assistant	10



Normalization: Example

Branch

branchNo	*branchAddress*
B002	City Center Plaza, Seattle, WA 98122
B004	16 – 14th Avenue, Seattle, WA 98128

Becomes primary key

TempStaff

staffNo	*name*	*position*
S4555	Ellen Layman	Assistant
S4612	Dave Sinclair	Assistant

Becomes primary key

TempStaffAllocation

staffNo	*branchNo*	*hoursPerWeek*
S4555	B002	16
S4555	B004	9
S4612	B002	14
S4612	B004	10

Becomes foreign key

Composite primary key

Becomes foreign key

Normalization: Boyce Codd Normal Form

- A relation schema R is in BCNF if whenever a **nontrivial functional dependency** X → A holds in R, then X is a superkey of R.
- If a functional dependency X->Y holds true where Y is not a subset of X then this dependency is called **non trivial Functional dependency**.
- **For example**:
  - An employee table with three attributes: emp_id, emp_name, emp_address. The following functional dependencies are non-trivial: emp_id -> emp_name (emp_name is not a subset of emp_id) emp_id -> emp_address (emp_address is not a subset of emp_id)
  - On the other hand, the following dependency is trivial: {emp_id, emp_name} -> emp_name [emp_name is a subset of {emp_id, emp_name}]

Boyce Codd Normal Form

- For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:
  1. It should be in the **Third Normal Form**.
  2. And, for any dependency A → B, A should be a **super key**.
     - It means, that for a dependency A → B, A cannot be a **non-prime attribute**, if B is a **prime attribute**.

Normalization: Boyce Codd Normal Form

- Below we have a college enrolment table with columns student_id, subject and professor.

student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

- In the table above:
  - One student can enrol for multiple subjects. For example, student with **student_id** 101, has opted for subjects - Java & C++
  - For each subject, a professor is assigned to the student.
  - And, there can be multiple professors teaching one subject like we have for Java.

Boyce Codd Normal Form

- In the table above student_id, subject together form the primary key, because using student_id and subject, we can find all the columns of the table.
- One more important point to note here is, one professor teaches only one subject, but one subject may have two different professors.
- Hence, there is a dependency between subject and professor here, where subject depends on the professor name.
- This table satisfies the **1st Normal form** because all the values are atomic, column names are unique and all the values stored in a particular column are of same domain.
- This table also satisfies the **2nd Normal Form** as their is no **Partial Dependency**.
- And, there is no **Transitive Dependency**, hence the table also satisfies the **3rd Normal Form**.

Boyce Codd Normal Form

- But this table is not in **Boyce-Codd Normal Form**.
- Why this table is not in BCNF?
  - In the table above, student_id, subject form primary key, which means subject column is a **prime attribute**.
  - But, there is one more dependency, professor → subject.
  - And while subject is a prime attribute, professor is a **non-prime attribute**, which is not allowed by BCNF.

Boyce Codd Normal Form

- To make this relation(table) satisfy BCNF, we will decompose this table into two tables, **student** table and **professor** table.
- Student Table

student_id	p_id
101	1
101	2
and so on...	

- Professor table

p_id	professor	subject
1	P.Java	Java
2	P.Cpp	C++
and so on...		

Boyce Codd Normal Form

- SportsClub

Ground	Begin_Time	End_Time	Package
G01	07:00	09:00	Gold
G01	10:00	12:00	Gold
G01	10:30	11:00	Bronze
G02	10:15	11:15	Silver
G02	08:00	09:00	Silver

The relation is in 1NF, 2NF, 3NF, but not in BCNF. Here is the reason –

Functional Dependency **{Package->Ground}** It has the determinant attribute Package on which Ground depends on is neither a Candidate Key nor a superset of the candidate key.

Boyce Codd Normal Form

- Package

Package	**Ground**
Gold	G01
Silver	G02
Bronze	G01

- TomorrowBookings

Ground	**Begin_Time**	**End_Time**
G01	07:00	09:00
G01	10:00	12:00
G01	10:30	11:00
G02	10:15	11:15
G02	08:00	09:00

Now the above tables are in BCNF.
Candidate key for **<Package>** table are Package and Ground
Candidate key for **<TomorrowBookings>** table are **{Ground, Begin_Time}** and **{Ground, End_Time}**

Multivalued dependency

- Definition. A multivalued dependency X → Y specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R:
  - If two tuples t1 and t2 exist in r such that t1[X] = t2[X], then two tuples t3 and t4 should also exist in r with the following properties, where we use Z (Z is shorthand for the attributes in R after the attributes in (X ∪ Y) are removed from R) to denote (R – (X ∪ Y)):
    - t3[X] = t4[X] = t1[X] = t2[X]
    - t3[Y] = t1[Y] and t4[Y] = t2[Y]
    - t3[Z] = t2[Z] and t4[Z] = t1[Z]

Multivalued dependency

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- 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.

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

- 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

Multivalued dependency

- Whenever X →→ Y holds, we say that X **multidetermines** Y.
- Because of the symmetry in the definition, whenever X →→ Y holds in R, so does X →→ Z. Hence, X →→ Y implies X →→ Z and therefore it is sometimes written as X →→ Y|Z.
- A multivalued dependency (MVD) X →→ Y in R is called a **trivial MVD** if
  - (a) Y is a subset of X, or
  - (b) X ∪ Y = R

EMP_PROJECTS

Ename	Pname
Smith	X
Smith	Y

EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

EMP_PROJECTS in Figure has the trivial MVD **Ename →→ Pname** and the relation EMP_DEPENDENTS has the trivial MVD **Ename →→ Dname** as satisfying (b).

Fourth Normal Form

- **Definition:** A relation schema R is in 4NF with respect to a set of dependencies F (that includes functional dependencies and multivalued dependencies) if, for every nontrivial multivalued dependency X →→ Y in F+, X is a superkey for R.
- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency

EMP

Ename	Pname	Dname
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

EMP_PROJECTS

Ename	Pname
Smith	X
Smith	Y

EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

Two MVDs: Ename →→ Pname and Ename →→ Dname

Fourth Normal Form

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

Two MVDs: Stu_ID →→ Course and Stu_ID →→ Hobby

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Join dependency

- If a table can be recreated by joining multiple tables and each of this table have a subset of the attributes of the table, then the table is in Join Dependency.
- It is a generalization of Multivalued Dependency.
- (πR1 (r), πR2 (r), … , πRn (r)) = r

Join dependency

Employee

EmpName	**EmpSkills**	**EmpJob** (Assigned Work)
Tom	Networking	EJ001
Harry	Web Development	EJ002
Katie	Programming	EJ002

Join Dependency:
{(EmpName, EmpSkills), (EmpName, EmpJob), (EmpSkills, EmpJob)}

EmployeeSkills

EmpName	**EmpSkills**
Tom	Networking
Harry	Web Development
Katie	Programming

EmployeeJob

EmpName	**EmpJob**
Tom	EJ001
Harry	EJ002
Katie	EJ002

JobSkills

EmpSkills	**EmpJob**
Networking	EJ001
Web Development	EJ002
Programming	EJ002

Fifth Normal Form

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless ($R1 \bowtie R2 = R$).
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

Fifth Normal Form

- Combination of all these fields required to identify a valid data.

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

Fifth Normal Form

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen