Normal Forms Based on Primary Keys

- ***Normalization of Relations***
- The normalization process, as first proposed by Codd (1972), **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.
- 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.**

Normal Forms Based on Primary Keys

- ***Normalization of Relations***
- **Normalization of data is 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.**
- An unsatisfactory relation schema that does not meet the condition for a normal form—the normal form test—is decomposed into smaller relation schemas that contain a subset of the attributes and meet the test that was otherwise not met by the original relation.

Normal Forms Based on Primary Keys

- ***Normalization of Relations***
- Definition:
  - The normal form of a relation is the highest normal form condition that a relation meets,
  - and hence indicates the degree to which it has been normalized.
- Existing designs are evaluated by applying the tests for normal forms, and normalization is carried out in practice so that the resulting designs are of high quality and meet the desirable properties stated previously.
- **Denormalization is a technique used by database administrators to optimize the efficiency of their database infrastructure. This method allows us to add redundant data into a normalized database to alleviate issues with database queries that merge data from several tables into a single table.**

Normal Forms Based on Primary Keys

- ***Definitions of Keys and Attributes Participating in Keys***
- Definition: A **superkey** of a relation schema R = {A1, A2, … , An} is a set of attributes S ⊆ R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S].
- A **key K is a superkey** with the additional property that removal of any attribute from K will cause K not to be a super key anymore.
- The difference between a key and a super key is that a key has to be minimal.
- **{Ssn} is a key** for EMPLOYEE,
- whereas **{Ssn}, {Ssn, Ename}, {Ssn, Ename, Bdate}**, and any set of attributes that includes Ssn are all superkeys.

Normal Forms Based on Primary Keys

- ***Definitions of Keys and Attributes Participating in Keys***
- If a relation schema has more than one key, each is called a **candidate key**.
- One of the candidate keys is arbitrarily designated to be the primary key, and the others are called **secondary keys**.
- In a practical relational database, each relation schema must have a primary key.
- **If no candidate key is known for a relation, the entire relation can be treated as a default superkey.**
- {Ssn} is the only candidate key for EMPLOYEE, so it is also the primary key.

Normal Forms Based on Primary Keys

- ***Definitions of Keys and Attributes Participating in Keys***
- Definition. An attribute of relation schema R is called a **prime attribute** of R if it is a member of some candidate key of R.
  - For example, {SSn,Ename} is a CK in R, then prime attributes are SSn and Ename.
- An attribute is called **nonprime** if it is not a prime attribute—that is, if it is not a member of any candidate key.

Properties of Functional Dependencies

- ***Reflexivity***
  - *If* X → Y & y is the subset of X , then X → X
  - An attribute determines itself
  - Always valid
  - Trivial FD
- ***Transitivity***
  - *If (*X → Y & Y → Z), then X → Z (might or might not be valid if any one of the condition in if case is false)
- ***Augmentation***
  - *If (*X → Y), then XA → YA
- ***Union***
  - *If (*X → Y & X → Z), then X → YZ
- ***Decomposition***
  - *If (*X → YZ), then X → Y , X → Z
  - Converse is not true
- ***Pseudo Transitivity***
  - *If (*X → Y & YZ → A), then XZ → A
- ***Composition***
  - If (X → Y & A → B), then XA → YB

Roll No	Name	Marks	Department	Course
1	Maryam	78	CS	OOP
2	Maira	60	AI	OOP
3	Maryam	78	CS	DB
4	Maira	60	AI	ML
5	Mohammad	80	SE	DB

Identifying SK and CK in a relation

- **Closure set / Attribute closure to find CK:**
  - Represented as X^+
  - X could be an attribute or a set of attributes
  - **SUPERKEY:** whose closure set contains all attributes of a relation R
    - Find closure set
    - If the closure set contains all attributes then it is a SK
  - **Candidate Key:** a key whose proper subset is not a SK
    - find proper subsets of a SK
    - and check if the closure sets of these are not SK then
    - It's a candidate key
  - Question:
    - R(A,B,C,D,E)
    - FD ($A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$)

Identifying SK and CK in a relation

- **How to find SK and CK:**
  - Question:
    - R(A,B,C,D,E)
    - FD ($A \rightarrow B$, $D \rightarrow E$)
  - STEPS:
    - Find closures sets
    - Identify SK
    - Check if it's a CK
      - find proper subsets of a SK
      - and check if the closure sets of these are not SK then
      - It's a candidate key

Normal Forms Based on Primary Keys

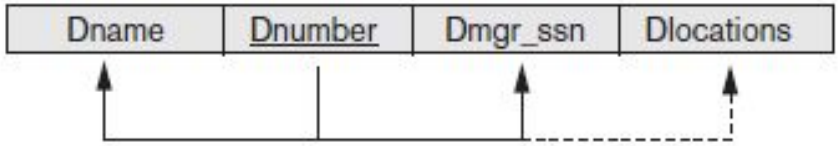
- ***First Normal Form***
- It states that the **domain of an attribute must include only atomic (simple, indivisible) values** and that the value of any attribute in a tuple must be a single value from the domain of that attribute.
- Hence, **1NF disallows having a set of values**, a tuple of values, or a combination of both as an attribute value for a single tuple.
- The only attribute values permitted by 1NF are **single atomic (or indivisible) values**.

Normal Forms Based on Primary Keys

- ***First Normal Form***
- Consider the DEPARTMENT relation schema shown in Figure 14.1, whose primary key is Dnumber, and suppose that we extend it by including the Dlocations attribute as shown in Figure 14.9(a).
- We assume that each department can have a number of locations.
- As we can see, this is not in 1NF because Dlocations is not an atomic attribute, as illustrated by the first tuple in Figure 14.9(b).

(a)

DEPARTMENT



(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

Normal Forms Based on Primary Keys

- ***First Normal Form***
- ***FIRST OPTION:***
- **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}, as shown in Figure 14.2.
- 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

Normal Forms Based on Primary Keys

- ***First Normal Form***
- ***SECOND OPTION:***
- **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 which leads to updated anomalies and hence is rarely adopted.

(c)

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

Normal Forms Based on Primary Keys

- ***First Normal Form***
- ***THIRD OPTION:***
- **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.**
- **Disadvantage:**
  - **NULL values ~** if most departments have fewer than three locations.
  - **Querying on this attribute becomes more difficult;**
    - for example, consider how you would write the query: List the departments that have 'Bellaire' as one of their locations in this design.
- For all these reasons, it **is best to avoid this alternative.**

Normal Forms Based on Primary Keys

- ***First Normal Form***
- ***THIRD OPTION:***
- 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.**
- In fact, **if we choose the second solution, it will be decomposed further during subsequent normalization steps into the first solution**.

Normal Forms Based on Primary Keys

- ***First Normal Form***
- ***THIRD OPTION:***
- 1NF also disallows multivalued attributes that are themselves composite.
- These are called **nested relations** because each tuple can have a relation within it.
- Figure 14.10 shows how the EMP_PROJ relation could appear if nesting is allowed.
- In EMP_Proj -> shows each employee entity with the projects on which he/she is working.
- The schema of this EMP_PROJ relation can be represented as follows:
- EMP_PROJ(Ssn, Ename, {PROJS(Pnumber, Hours)})
- The set braces { } identify the attribute PROJS as multivalued, and we list the component attributes that form PROJS between parentheses ( ).

Normal Forms Based on Primary Keys

- ***First Normal Form***
- Ssn is the primary key of the EMP_PROJ relation in Figures 14.10(a) and (b),
- whereas Pnumber is the partial key of the nested relation; that is, within each tuple, the nested relation must have unique values of Pnumber.
- Figure 14.10 (c) shows the solution.
- 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.

(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
999887777	Zelaya, Alicia J.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321	Wallace, Jennifer S.	30	20.0
		20	15.0
888665555	Borg, James E.	20	NULL

(c)

EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

Figure 14.10
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.

Normal Forms Based on Primary Keys

- ***First Normal Form***
- This procedure can be applied recursively to a relation with multiple-level nesting to un nest the relation into a set of 1NF relations.
- This is useful in converting an un normalized relation schema with many levels of nesting into 1NF relations.
- **As an example, consider the following:**
- **CANDIDATE (Ssn, Name, {JOB_HIST (Company, Highest_position, {SAL_HIST (Year, Max_sal)})})**
- **The first normalization using internal partial keys Company and Year,respectively, results in the following 1NF relations:**
- **CANDIDATE_1 (Ssn, Name)**
- **CANDIDATE_JOB_HIST (Ssn, Company, Highest_position)**
- **CANDIDATE_SAL_HIST (Ssn, Company, Year, Max-sal)**