Normalization

- **Normalization**
  - process of analyzing the given relation schemas based on their Functional Dependencies and primary keys to achieve the desirable properties:

  (1) Minimizing redundancy

  (2) Minimizing the insertion, deletion, and update anomalies

Redundant Information

redundancy

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
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

- In **EMP_DEPT**, the attribute values pertaining to a particular department (**DNUMBER**, **DNAME**, **DMGRSSN**) are repeated for every employee who works for that department.

Redundant Information

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	291 Berry,Bellaire,TX	4
Narayan,Remesh 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	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

- In contrast, each department's information appears only once in the **DEPARTMENT** relation.
- Only the department number (**DNUMBER**) is repeated in the EMPLOYEE relation for each employee who works in that department
- These can be classified into **insertion anomalies, deletion anomalies, and modification anomalies**.

What Is Anomalies?

- **Anomalies** are problems that can occur in poorly planned, unnormalized databases where all the data is stored in one table (a flat-file database).
- There are different types of anomalies which can occur in referencing and referenced relation:
  - **Insertion anomaly:** there are circumstances in which certain facts cannot be recorded.
  - **Deletion anomaly:** the unintended loss of data due to deletion of other data.
  - **Modification/Update anomaly:** the same information can be expressed on multiple rows; therefore updates to the relation may result in logical inconsistencies.

Insertion Anomalies

- can be differentiated into two types, based on the **EMP_DEPT** relation:
  - To insert a new employee tuple into **EMP_DEPT**, we must include either the attribute values for the department that the employee works for
  - or **nulls** (if the employee does not work for a department as yet).

redundancy

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
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

Insertion Anomalies

- 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 causes a problem because **SSN** is the **primary key** of **EMP_DEPT**, and each tuple is supposed to represent an employee entity-not a department entity.

Deletion Anomalies

- 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.
- This problem does not occur in this database because **DEPARTMENT** tuples are stored separately.

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	291 Berry,Bellaire,TX	4
Narayan,Remesh 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	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

Modification Anomalies

- In **EMP_DEPT**, if we change the value of one of the attributes of a particular department, For example 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.

For your Project

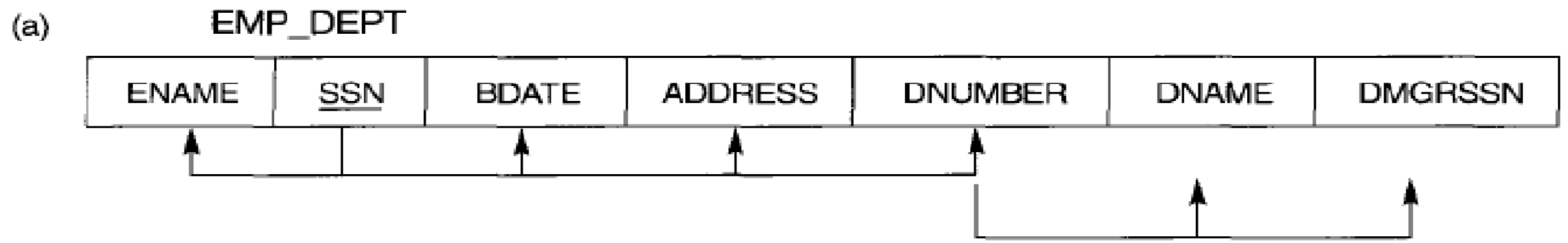
- Make sure to design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations.
- If any anomalies are present, note them clearly and make sure that the programs that update the database will operate correctly.

Functional Dependencies

- **Functional Dependency(FD):** is a relationship that exists when one attribute uniquely determines another attribute.
- An attribute **Y** is said to have a **functional dependency** on a set of attributes **X** (written **X → Y**) if and only if **each X** value is associated with precisely **one Y** value.
- It represented by an arrow sign **(→)** that is, **X→Y**, where **X** functionally determines **Y**. The left-hand side attributes **determine** the values of attributes on the right-hand side.
- Customarily we call **X** the determinant set and **Y** the dependent attribute.

Functional Dependencies

Example:



F= {SSN → {ENAME, BDATE, ADDRESS, DNUMBER},
DNUMBER→{DNAME, DMGRSSN}}

Some of the additional functional dependencies that we can infer from F are the following:

SSN → {DNAME, DMGRSSN}
SSN →SSN
DNUMBER →DNAME

Trivial functional dependency

- A functional dependency of an attribute on a superset of itself.

Example:

{Employee ID, Employee Address} → {Employee Address} is trivial, as is {Employee Address} → {Employee Address}.

Properties of functional dependencies

- **Subset Property** (Axiom of Reflexivity):
  - If Y is a subset of X, then X → Y
- **Augmentation** (Axiom of Augmentation):
  - If X → Y, then XZ → YZ
- **Transitivity** (Axiom of Transitivity):
  - If X → Y and Y → Z, then X → Z
- **Union:**
  - If X → Y and X → Z, then X → YZ
- **Decomposition**:
  - If X → YZ, then X → Y and X → Z
- **Pseudotransitivity:**
  - f X → Y and WY → Z, then XW → Z

Functional Dependencies types

- **Full functional dependency:** an attribute is **fully functionally dependent** on a set of attributes X if it is:
  - Functionally dependent on X, and
  - Not functionally dependent on any proper subset of X.
  - X->Y is a full functional dependency if removal of any attribute from X means that the dependency does not hold anymore.
- **Partial dependency:** if some attribute in X can removed and the dependency still holds.
- **Transitive dependency:** an indirect functional dependency, one in which X→Z only by virtue of X→Y and Y→Z

Keys in Database

- Superkey: subset of attributes that separate any two rows.
  - For example, **{SSN}**, **{SSN, ENAME}**, **{SSN, ADDRESS}.**
- Key: a key has to be **minimal**.
- Candidate Key: has more than one key.
  - For example, **SSN and student_ID**.
- Primary key: one picked from the candidate key pool.
  - Most DBMSs require a table to be defined as having a single unique key, rather than a number of possible unique keys.
- Secondary keys: candidate key minus primary key.

Normal forms

- **Normal forms (NF):** provide criteria for determining a table's degree of vulnerability to logical inconsistencies and anomalies.
- The normal forms are applicable to individual tables; to say that an entire database is in **normal form n** is to say that all of its tables are in **normal form n**.

First Normal Form (1NF)

- A table is in **1NF** if and only it satisfies the following **five** conditions:
  - There's no top-to-bottom ordering to the rows.
  - There's no left-to-right ordering to the columns.
  - There are no duplicate rows.
  - Every row-and-column intersection contains exactly one value from the applicable domain (and nothing else).
  - All columns are regular [i.e. rows have no hidden components such as row IDs, object IDs, or hidden timestamps].
- Violation of any of these conditions would mean that the table is not strictly relational, and therefore that it is not in **1NF**.

First Normal Form (1NF)

Example of tables that would not meet this definition of 1NF are:

- A table that lacks a **unique key**. Such a table would be able to accommodate duplicate rows, in violation of condition 3.

- A table with at least one **nullable** attribute. A nullable attribute would be in violation of condition 4, which requires every field to contain exactly one value from its column's domain.

First normal form (1NF)

Example:

Suppose a novice designer wishes to record the names and telephone numbers of customers. He defines a customer table which looks like this:

Customer			
Customer ID	**First Name**	**Surname**	**Telephone Number**
123	Robert	Ingram	555-861-2025
456	Jane	Wright	555-403-1659
789	Maria	Fernandez	555-808-9633

First normal form (1NF)

- The designer then becomes aware of a requirement to record multiple telephone numbers for some customers. He reasons that the simplest way of doing this is to allow the "Telephone Number" field in any given record to contain more than one value:

Customer			
Customer ID	**First Name**	**Surname**	**Telephone Number**
123	Robert	Ingram	555-861-2025
456	Jane	Wright	555-403-1659 555-776-4100
789	Maria	Fernandez	555-808-9633

First normal form (1NF)

- Assuming, however, that the Telephone Number column is defined on some Telephone Number-like constraint (e.g. strings of 12 characters in length), the previous representation is not in 1Nf as it prevents a single field from containing more than one value from its column's domain.
- The designer might attempt to get around this restriction by defining multiple Telephone Number columns:

Customer					
Customer ID	**First Name**	**Surname**	**Tel. No. 1**	**Tel. No. 2**	**Tel. No. 3**
123	Robert	Ingram	555-861-2025		
456	Jane	Wright	555-403-1659	555-776-4100	555-403-1659
789	Maria	Fernandez	555-808-9633		

First normal form (1NF)

- This representation use of nullable columns, and therefore does not conform to definition of 1NF and causes logical problems. These problems include:
  - Difficulty in querying the table. Answering such questions as "Which customers have telephone number X?"
  - Inability to enforce uniqueness of Customer-to-Telephone Number links through the RDBMS. Customer 789 might mistakenly be given a Tel. No. 2 value that is exactly the same as her Tel. No. 1 value.
  - Restriction of the number of telephone numbers per customer to three. If a customer with four telephone numbers comes along, we are constrained to record only three and leave the fourth unrecorded.

First normal form (1NF)

- A design that is unambiguously in **1NF** makes use of two tables:
  - a Customer Name table and
  - a Customer Telephone Number table.

Customer Name		
Customer ID	**First Name**	**Surname**
123	Robert	Ingram
456	Jane	Wright
789	Maria	Fernandez

Customer Telephone Number	
Customer ID	**Telephone Number**
123	555-861-2025
456	555-403-1659
456	555-776-4100
789	555-808-9633

Second normal form

- A table that is in first normal form (1NF) must meet additional criteria if it is to qualify for **second normal form**.
  - Specifically: a 1NF table is in **2NF** if and only if, given any candidate key K and any attribute A that is not a constituent of a candidate key, A depends upon the whole of K rather than just a part of it.
  - In slightly more formal terms: a 1NF table is in **2NF** if and only if all its **non-prime** attributes are functionally dependent on the whole of a candidate key. (**A non-prime attribute is one that does not belong to any candidate key.**)

Second normal form

- Consider a table describing employees' skills:

Employees' Skills		
Employee	**Skill**	**Current Work Location**
Jones	Typing	114 Main Street
Jones	Shorthand	114 Main Street
Jones	Whittling	114 Main Street
Bravo	Light Cleaning	73 Industrial Way
Ellis	Alchemy	73 Industrial Way
Ellis	Flying	73 Industrial Way
Harrison	Light Cleaning	73 Industrial Way

Second normal form

- Neither {Employee} nor {Skill} is a candidate key for the table.
- This is because a given Employee might need to appear more than once (he might have multiple Skills), and a given Skill might need to appear more than once (it might be possessed by multiple Employees).
- Only the composite key {Employee, Skill} qualifies as a candidate key for the table.
- The remaining attribute, Current Work Location, is dependent on only part of the candidate key, namely Employee.
- Therefore the table is not in 2NF.

Second normal form

- A 2NF alternative to this design would represent the same information in two tables:
  - an "Employees" table with candidate key {Employee}, and
  - an "Employees' Skills" table with candidate key {Employee, Skill}:

Employees	
Employee	**Current Work Location**
Jones	114 Main Street
Bravo	73 Industrial Way
Ellis	73 Industrial Way
Harrison	73 Industrial Way

Employees' Skills	
Employee	**Skill**
Jones	Typing
Jones	Shorthand
Jones	Whittling
Bravo	Light Cleaning
Ellis	Alchemy
Ellis	Flying
Harrison	Light Cleaning

Third normal form

- The **third normal form (3NF)** is a normal form used in database normalization.
- **3NF** was originally defined by E.F. Codd in 1971.
- Codd's definition states that a table is in **3NF** if and only if both of the following conditions hold:
  - The relation R (table) is in **second normal form (2NF)**
  - Every non-prime attribute of R is **non-transitively dependent** (i.e. directly dependent) on every candidate key of R.

Third normal form

Tournament Winners			
Tournament	**Year**	**Winner**	**Winner Date of Birth**
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

A 2NF table that fails to meet the requirements of 3NF is:

- Because each row in the table needs to tell us who win a particular Tournament in a particular Year, the composite key {Tournament, Year} is a minimal set of attributes guaranteed to uniquely identify a row.
- That is, {Tournament, Year} is a candidate key for the table.
- The breach of 3NF occurs because the non-prime attribute Winner Date of Birth is transitively dependent on the candidate key {Tournament, Year} via the non-prime attribute Winner.

Third normal form

- The fact that Winner Date of Birth is functionally dependent on Winner makes the table vulnerable to logical inconsistencies, as there is nothing to stop the same person from being shown with different dates of birth on different records.
- In order to express the same facts without violating 3NF, it is necessary to split the table into two:

Third normal form

Tournament Winners		
Tournament	**Year**	**Winner**
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Player Dates of Birth	
Player	**Date of Birth**
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

Update anomalies cannot occur in these tables, which are both in 3NF

Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

Table 14.1 Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

Notes on Normal Forms Based on Primary Keys

- Takes a relation schema through a series of tests to certify whether it satisfies a certain normal form.
- Decompose relations as necessary.
- 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.

Make your database into 3NF.