Skills for Hire
Data Analyst Program
Week 3

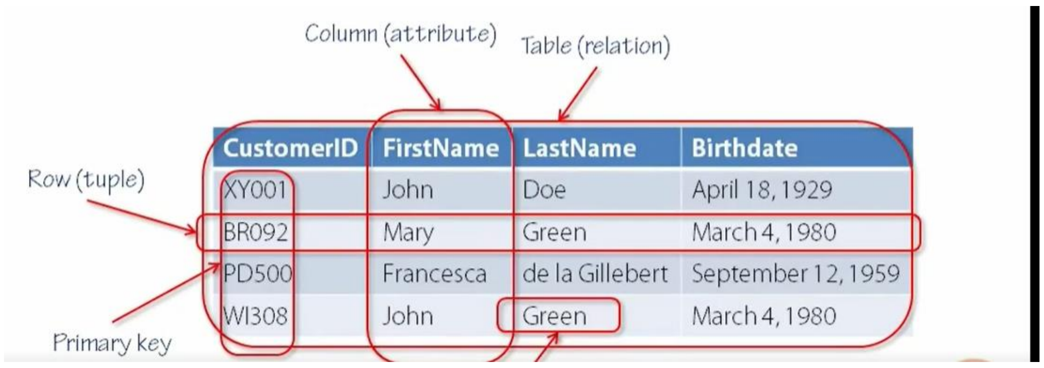
Relational Database Design

Development of storage hardware

- **A punch card** is made out of stiff paper, with holes punched into it. Information is encoded by punching holes in specific pre-defined positions.
- **Punch tape** uses the same basic idea as punch cards, punching holes in paper to store data, but instead of using a deck of cards, a long strip of paper was now fed through a machine that would either punch holes in the right locations or read the locations of the holes and translate it back into data for the computer to process.
- **The magnetic tape** is a long strip of plastic film coated with a magnetizable coating. This strip is then passed at great speed along the read/write heads that are able to detect if modify the magnetic field of this coding. Using plastic instead of paper and not punching holes in the tape resulted in a much better longevity of the medium and the information density of magnetic tape is much, much higher than that of punch tape. This, along with a much higher access speeds, revolutionize computing and storage
- **Spinning (Hard disk and Floppy disk)** Hard Disk is a magnetic disk made of aluminium. It is used as the main storage device on the computer. It uses a metallic disk which is known as a platter. Both sides of the disk are used for storing data except the upper side of the uppermost disk and the lower side of the lowermost disk. Magnetic oxide is used to coat the data storing surface.
  Floppy disk is a magnetic disk that contains a single plastic disk. In the initial days of the computer invention it was used as the main storage device, later on, it was used for carrying data from one computer to another but nowadays it is almost not used. It requires a floppy drive for operation.
- **Optical disc (Compact/DVD/Blu-ray)** Optical discs have a lot in common with magnetic discs. They also encoded data on a round surface that spins at high speed, while the read/write head moves to the appropriate location, but the information is encoded using optical rather than magnetic techniques.
- **Solid Sate Drives, or SSDs,** are the latest new developer in storage hardware. The technology has been used for several years already in thumb drive and consumer devices such as Smartphone's, but only recently have reliability capacity and pricing developed to a level where SSDs are a viable alternative for enterprise storage. An SSD does not have any moving parts. All data is stored on memory chips, usually flesh memory, a special kind of memory chips that retain their stored information even without power. Because there are no moving parts, access time is no longer limited by the time required to move a read/write head or to spin a disk to the correct position. Every location on the drive can be instantly accessed. This makes SSDs a lot faster than even the fastest magnetic discs

Relational Model Database

Based on set theory and first-order predicate logic.



CustomerID	FirstName	LastName	Birthdate
XY001	John	Doe	April 18, 1929
BR092	Mary	Green	March 4, 1980
PD500	Francesca	de la Gillebert	September 12, 1959
WI308	John	Green	March 4, 1980

E.F.Codd Rule

Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database. These rules can be applied on any database system that manages stored data using only its relational capabilities. This is a foundation rule, which acts as a base for all the other rules.

- Rule 0: Foundation Rule
  The following rules can be **applied on any database system** that manages stored data using only its relational capabilities.

- Rule 1: Information Rule
  The data stored in a database, may it be user data or metadata, **must be a value** of some table cell. Everything in a database must be stored in a table format.

- Rule 2: Guaranteed Access Rule
  Every single data element (value) is **guaranteed to be accessible logically** with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

- Rule 3: Systematic Treatment of NULL Values.
  The NULL values in a database must be given a **systematic and uniform treatment.** This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

- Rule 4: Active Online Catalog
  The structure description of the entire database **must be stored** in an online catalog, known as data dictionary, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

- Rule 5: Comprehensive Data Sub-Language Rule
  A database **can only be accessed** using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

- Rule 6: View Updating Rule
  All the views of a database, which can theoretically be updated, **must also be updatable** by the system.

- Rule 7: High-Level Insert, Update, and Delete Rule
  A database **must support** high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

- Rule 8: Physical Data Independence
  The data stored in a database **must be independent** of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

- Rule 9: Logical Data Independence
  The logical data in a database must be **independent of its user's view** (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

- Rule 10: Integrity Independence
  A database must be independent of the application that uses it. All its **integrity constraints** can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

- Rule 11: Distribution Independence
  The end-user must **not be able to see** that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

- Rule 12: Non-Subversion Rule
  If a system has an interface that provides access to low-level records, then the interface **must not be able** to subvert the system and bypass security and integrity constraints.

Data Modelling

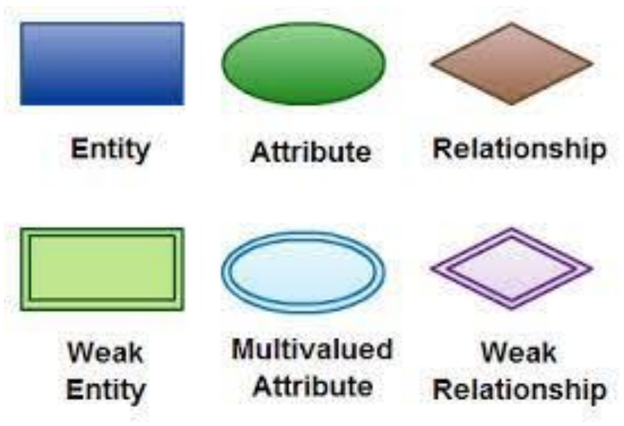
Data modeling (data modelling) is the process of creating a data model for the data to be stored in a database. This data model is a conceptual representation of Data objects, the associations between different data objects, and the rules. Data modeling helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data.

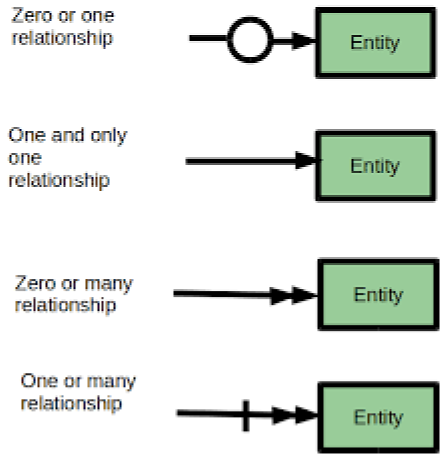
- Conceptual Data Model: This Data Model defines WHAT the system contains. This model is typically created by Business stakeholders and Data Architects. The purpose is to organize, scope and define business concepts and rules.
- Logical Data Model: Defines HOW the system should be implemented regardless of the DBMS. This model is typically created by Data Architects and Business Analysts. The purpose is to developed technical map of rules and data structures.
- Physical Data Model: This Data Model describes HOW the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database

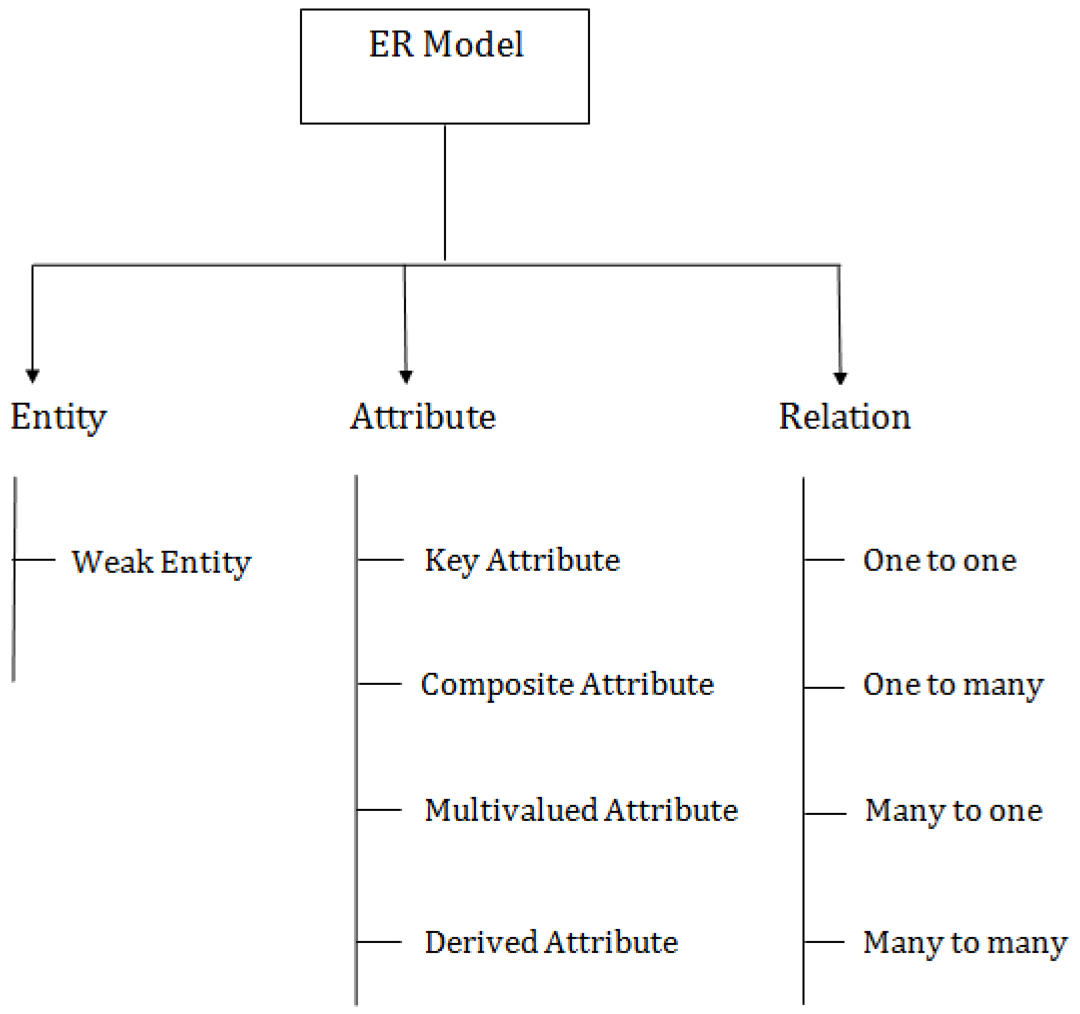
ER Modelling

Entity Relationship Model (ER Modeling) is a graphical approach to database design. It is a high-level data model that defines data elements and their relationship for a specified software system. An ER model is used to represent real-world objects.
An Entity is a thing or object in real world that is distinguishable from surrounding environment. For example, each employee of an organization is a separate entity also known as instance. Entity can have values.

Entity and Attributes
An employee of an organization is an entity. If “Peter” is a programmer (an employee) at Microsoft, he can have attributes (properties) like name, age, weight, height, etc. It is obvious that those do hold values relevant to him.


Entity
Attribute
Relationship
Weak Entity
Multivalued Attribute
Weak Relationship


Zero or one relationship
Entity
One and only one relationship
Entity
Zero or many relationship
Entity
One or many relationship
Entity



Relationships

a. One-to-One Relationship
   - When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.
   - For example, A female can marry to one male, and a male can marry to one female.

b. . One-to-many relationship
   - When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.
   - For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.

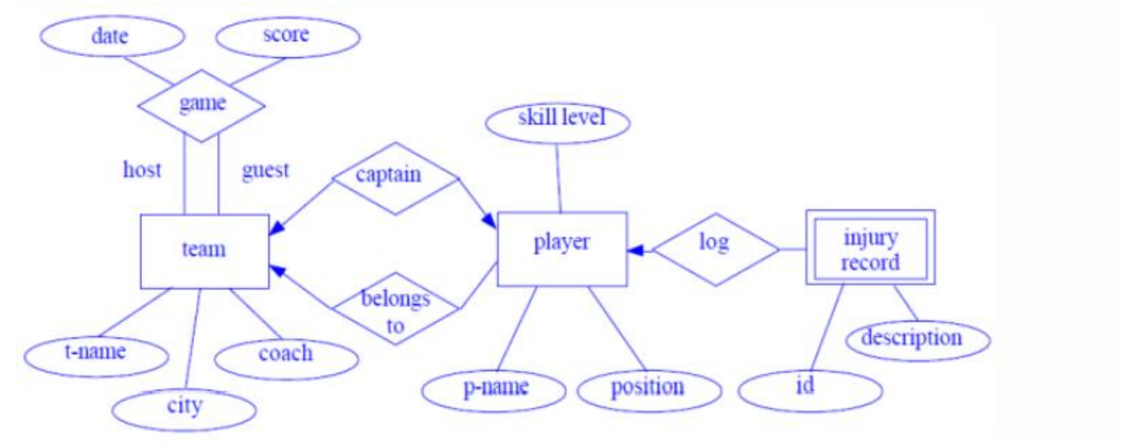
c. Many-to-one relationship
   - When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.
   - For example, Student enrolls for only one course, but a course can have many students.

d. Many-to-many relationship
   - When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.
   - For example, Employee can assign by many projects and project can have many employees.

e. Recursive Relationship

- A relationship between two entities of a similar entity type is called a recursive relationship. Here the same entity type participates more than once in a relationship type with a different role for each instance.

ER diagram case study
Suppose you are given the following requirements for a simple database for the National Hockey League (NHL):
• the NHL has many teams,
• each team has a name, a city, a coach, a captain, and a set of players,
• each player belongs to only one team,
• each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records, • a team captain is also a player,
• a game is played between two teams (referred to as host_team and guest_team) and has a date (such as May 11th, 2019) and a score (such as 4 to 2).



Normalization
Why don't we like single large databases?

- It isn't easy to maintain and update data as it would involve searching many records in relation
- Wastage and poor utilization of disk space and resources (why?)
- The likelihood of errors and inconsistencies increases

Review of Keys:

- **Primary Key –** is a column or group of columns in a table that uniquely identify every row in that table.
- **Candidate Key –** is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- **Super keys -** The set of attributes that can uniquely identify a tuple is known as Super Key. For Example, STUD_NO, (STUD_NO, STUD_NAME), etc.

We analyse and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties
Normalization is the **process of organizing data in a database.** This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

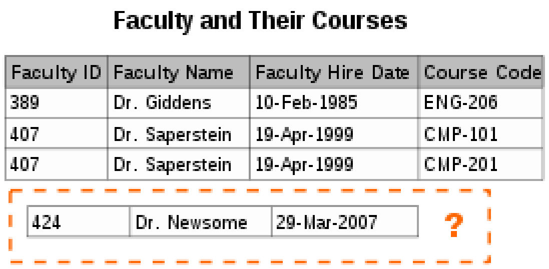
Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. For example, a customer address change is much easier to implement if that data is stored only in the Customers table and nowhere else in the database.

Normalization is used to **minimize the redundancy from a relation or set of relations.** It is also used to eliminate undesirable characteristics like **Insertion, Update, and Deletion Anomalies.**

Data modification anomalies can be categorized into three types:

- **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.

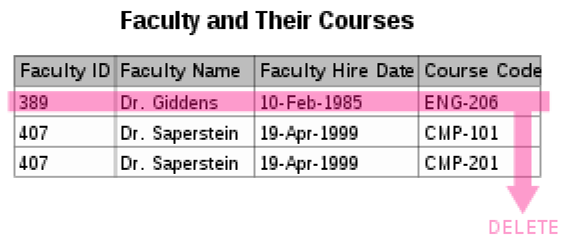
For example, each record in a "Faculty and Their Courses" relation might contain a Faculty ID, Faculty Name, Faculty Hire Date, and Course Code. Therefore, the details of any faculty member who teaches at least one course can be recorded, but a newly hired faculty member who has not yet been assigned to teach any courses cannot be recorded


Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201
424	Dr. Newsome	29-Mar-2007	?

- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.

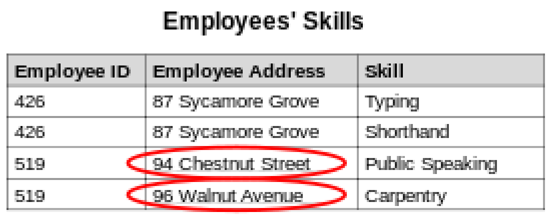
A deletion anomaly. All information about Dr. Giddens is lost if they temporarily cease to be assigned to any courses.


Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201

- **Updatation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

For example, each record in an "Employees' Skills" relation might contain an Employee ID, Employee Address, and Skill; thus a change of address for a particular employee may need to be applied to multiple records (one for each skill).

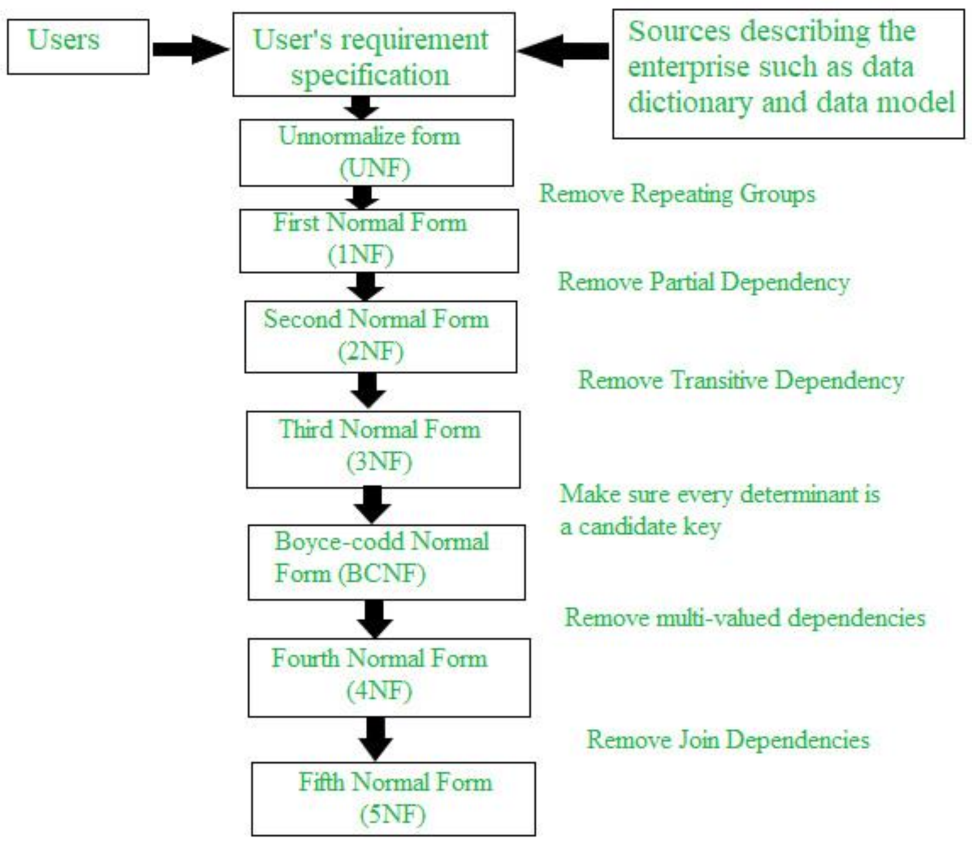

Employees' Skills

Employee ID	Employee Address	Skill
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519	94 Chestnut Street	Public Speaking
519	96 Walnut Avenue	Carpentry

Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

Types of Normalization


Users
User's requirement specification
Sources describing the enterprise such as data dictionary and data model
Unnormalize form (UNF)
Remove Repeating Groups
First Normal Form (1NF)
Remove Partial Dependency
Second Normal Form (2NF)
Remove Transitive Dependency
Third Normal Form (3NF)
Make sure every determinant is a candidate key
Boyce-codd Normal Form (BCNF)
Remove multi-valued dependencies
Fourth Normal Form (4NF)
Remove Join Dependencies
Fifth Normal Form (5NF)

1NF

Scenario: A video library maintains a database of movies rented out. Without any normalization in database, all information is stored in one table as shown below.

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.

Here you see Movies Rented column has multiple values.

1 NF Requirement: **A relation is in 1NF if it contains an atomic value. (NO multiple elements in one cell)**

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.

2NF
In the 2NF, relational must be in 1NF.
In the second normal form, all non-key attributes are fully functional dependent on the primary key
Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30

47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

3NF :

Third Normal Form (3NF)

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.

- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

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

- Eliminate fields that do not depend on the key.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X → Y.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

An entity is in the third normal form (3NF) if it is in the second normal form and all of its attributes are not transitively dependent on the complete primary key. Transitive dependency exists when a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key. In other words: The third normal form means that no attribute within an entity is dependent on an non-prime attribute that, in turn, depends on the primary key.

It may be more feasible to apply third normal form only to data that changes frequently. If some dependent fields remain, design your application to require the user to verify all related fields when any one is changed.

When an indirect relationship causes functional dependency it is called Transitive Dependency.

Super key: EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

Candidate key: {EMP_ID}

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.

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

Higher Forms of Normalization

Boyce-Codd Normal Form (BCNF):

Boyce–Codd Normal Form (BCNF) is based on functional dependencies that take into account all candidate keys in a relation; however, BCNF also has additional constraints compared with the general definition of 3NF.
For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:

- It should be in the Third Normal Form.
- And, for any dependency A → B, A should be a super key.

Super keys - The set of attributes that can uniquely identify a tuple is known as Super Key. For Example, STUD_NO, (STUD_NO, STUD_NAME), etc.

EMPLOYEE table:

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

Functional dependencies

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Fourth Normal Form

If two or more independent relation are kept in a single relation or we can say **multivalue dependency** occurs when the presence of one or more rows in a table implies the presence of one or more other rows in that same table. Put another way, two attributes (or columns) in a table are independent of one another, but both depend on a third attribute. A **multivalued dependency** always requires at least three attributes because it consists of at least two attributes that are dependent on a third.

For a dependency A -> B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency. The table should have at least 3 attributes and B and C should be independent for A ->> B multivalued dependency. For example,

STUDENT

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

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

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

STUDENT_HOBBY

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

Fifth Normal Form / Projected Normal Form (5NF):

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

- 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).

Example

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

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

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

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John

Math	John
Math	Akash
Chemistry	Praveen

P3

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