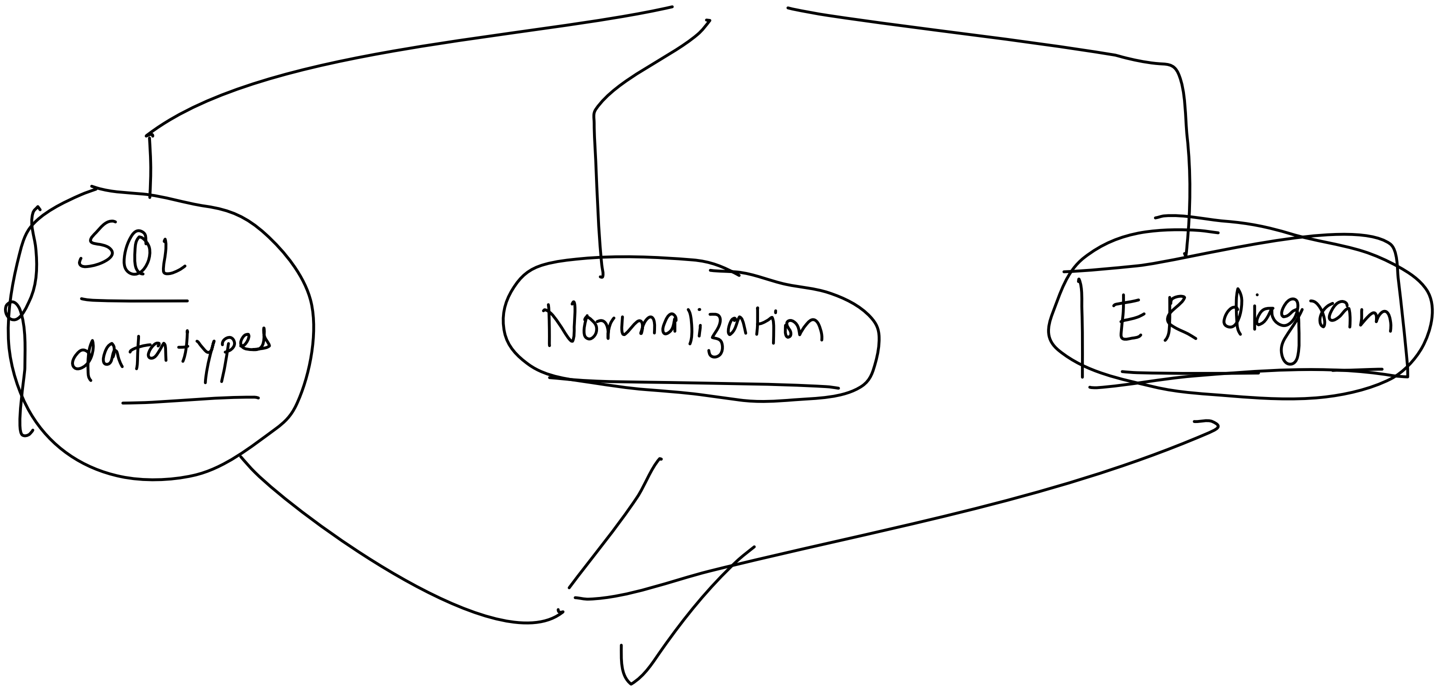
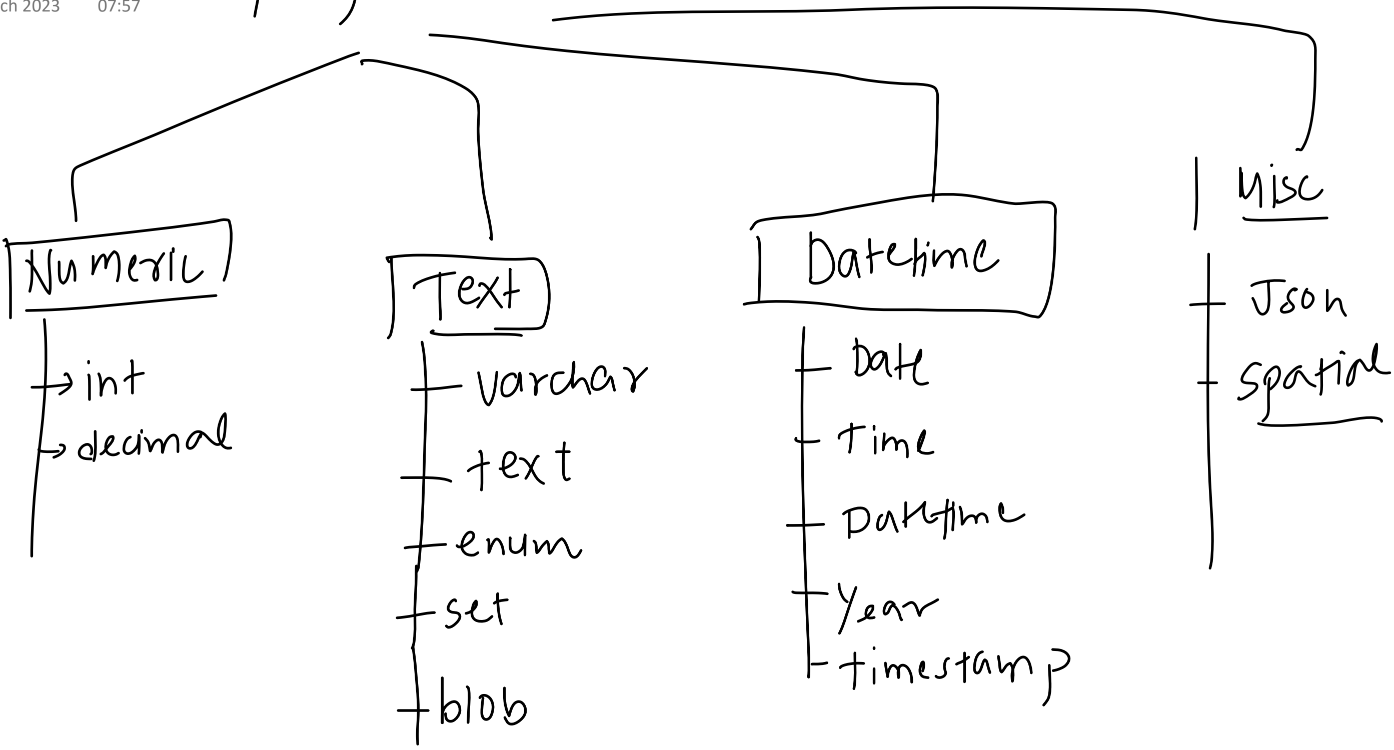
Plan of Attack

18 March 2023 18:25



SQL Datatypes (MySql)

18 March 2023 07:57



Numeric Types

18 March 2023 08:09

1. **INT**: The INT data type is used to store integers with a maximum value of 2147483647 and a minimum value of -2147483648. Examples of data that can be stored in INT include employee IDs, order numbers, and product IDs.

2. **TINYINT**: The TINYINT data type is used to store integers with a maximum value of 127 and a minimum value of -128. Examples of data that can be stored in TINYINT include Boolean values, such as 0 for false and 1 for true.

3. **SMALLINT**: The SMALLINT data type is used to store integers with a maximum value of 32767 and a minimum value of -32768. Examples of data that can be stored in SMALLINT include quantities of items, such as the number of products sold in a transaction.

4. **MEDIUMINT**: The MEDIUMINT data type is used to store integers with a maximum value of 8388607 and a minimum value of -8388608. Examples of data that can be stored in MEDIUMINT include the number of visitors to a website or the number of followers on a social media platform.

5. **BIGINT**: The BIGINT data type is used to store integers with a maximum value of 9223372036854775807 and a minimum value of -9223372036854775808. Examples of data that can be stored in BIGINT include the total revenue generated by a company or the number of views on a YouTube video. →7 ≃

6. **FLOAT**: The FLOAT data type is used to store single-precision floating-point numbers, which are numbers with a decimal point. Examples of data that can be stored in FLOAT include the price of a product or the temperature of a room. 4 bytes

7. **DOUBLE**: The DOUBLE data type is used to store double-precision floating-point numbers, which are numbers with a decimal point that can store more digits than FLOAT. Examples of data that can be stored in DOUBLE include very large or very small numbers, such as the distance between planets in the solar system or the size of an atom. 15 ≃ 8 bytes

8. **DECIMAL**: The DECIMAL data type is used to store exact decimal values with a fixed number of digits before and after the decimal point. Examples of data that can be stored in DECIMAL include financial values, such as the cost of an item or the total balance in a bank account.

→ Numeric

Int (7,1,8)

Decimals 7.2, 8.25

- tiny 1
- small 2
- medium 3
- Int 4
- bigint 5

- Decimals
- float 7
- double 8

-128 127

signed int → -ve +ve

unsign → 0 → +ve

0-256

decimal(5,2) 3492.5

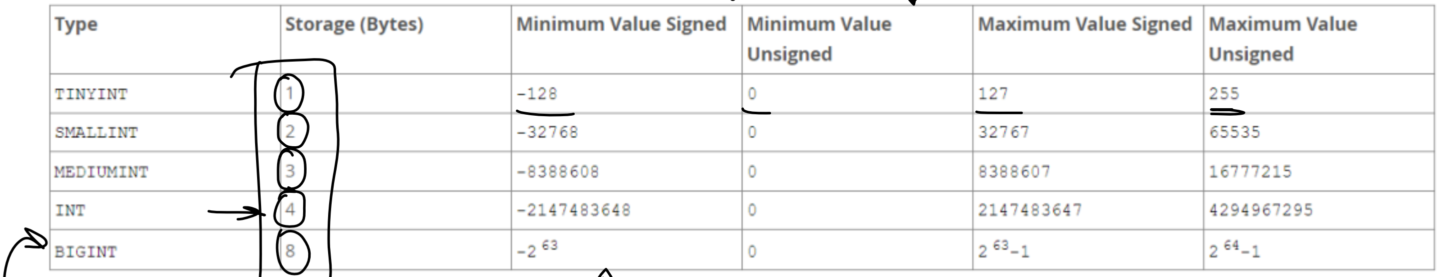
349.25

999.99

Decimal → total no of digit

DECIMAL(X, Y)

digits after .



Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^{63}	0	$2^{63}-1$	$2^{64}-1$

String Data Type

18 March 2023 08:15

1. **CHAR**: This data type is used to store fixed-length strings. The length of the string is specified when the table is created, and the field will always use that amount of space, regardless of whether the string stored in it is shorter or longer. For example, if you define a CHAR(10) field and store the string "hello" in it, MySQL will pad the string with spaces so that it takes up 10 characters. CHAR fields are useful when you have a field that always contains the same length of data, such as a state abbreviation or a phone number.

2. **VARCHAR**: This data type is used to store variable-length strings. The length of the string can be up to a specified maximum, but the field will only use as much space as it needs to store the actual data. For example, if you define a VARCHAR(10) field and store the string "hello" in it, MySQL will only use 5 characters to store the data. VARCHAR fields are useful when you have a field that can contain varying amounts of data, such as a user's name or address.

3. **TEXT**: This data type is used to store larger amounts of variable-length string data than VARCHAR. It can store up to 65,535 characters. TEXT fields are useful when you need to store large amounts of text data, such as blog posts or comments.

4. **MEDIUMTEXT**: This data type is used to store even larger amounts of text data than TEXT. It can store up to 16,777,215 characters. MEDIUMTEXT fields are useful when you need to store very large amounts of text data, such as long-form articles or legal documents.

5. **LONGTEXT**: This data type is used to store the largest amounts of text data. It can store up to 4,294,967,295 characters. LONGTEXT fields are useful when you need to store extremely large amounts of text data, such as entire books or large collections of data.

ENUM and SET

18 March 2023 08:16

male, female

1. **ENUM**: The ENUM data type is used to store a set of predefined values. You can specify a list of possible values for an ENUM column, and the column can only store one of these values. The ENUM data type can be used to ensure that only valid values are stored in a column, and it can also save storage space compared to storing string values. Example - gender
2. **SET**: The SET data type is similar to ENUM, but it can store multiple values. You can specify a list of possible values for a SET column, and the column can store any combination of these values. The SET data type can be used to store sets of values, such as tags or categories, in a single column. Example - hobbies

→ hobby [sports, gaming, travel]

BLOB

18 March 2023 08:16

Text

The BLOB (Binary Large Object) data type in MySQL is used to store large binary data, such as images, audio, video, or other multimedia content.

In MySQL, there are four types of BLOB data types that can be used to store binary data with different maximum sizes:

- TINYBLOB: Maximum length of 255 bytes. TINYBLOB is the smallest BLOB data type in MySQL. It can be used to store small binary data, such as icons, small images, or serialized objects.

- BLOB: Maximum length of 65,535 bytes (64 KB).BLOB is a medium-sized BLOB data type that can be used to store larger binary data, such as images, audio, video, or other multimedia files.

- MEDIUMBLOB: Maximum length of 16,777,215 bytes (16 MB).MEDIUMBLOB is a larger BLOB data type that can be used to store even larger binary data, such as high-resolution images or longer audio or video files.

- LONGBLOB: Maximum length of 4,294,967,295 bytes (4 GB).LONGBLOB is the largest BLOB data type in MySQL, and it can be used to store very large binary data, such as very high-resolution images, long audio or video files, or even entire documents.

LOAD_FILE(PATH)

Pros of storing files in BLOB columns:

- BLOB columns allow you to store binary data directly in the database, without needing to store the file externally.
- Storing files in the database can simplify backup and restore procedures, as all the data is in one place.
- Access to BLOB data can be controlled through database user permissions.

Cons of storing files in BLOB columns:

- Storing large files in the database can slow down database performance and increase storage requirements.
- If you need to access the file outside of the database (e.g. to share it with another application or user), you'll need to extract it from the database.
- Some file types may not be well-suited for storage in BLOB columns, depending on their size, structure, and how they are accessed.

Datetime

18 March 2023 08:17

In MySQL, there are several temporal data types that can be used to store and manipulate time and date values. These include:

1. **DATE** - used for storing date values in the format **YYYY-MM-DD**.
2. **TIME** - used for storing time values in the format **HH:MM:SS**.
3. **DATETIME** - used for storing date and time values in the format **YYYY-MM-DD HH:MM:SS**.
4. **TIMESTAMP** - used for storing date and time values in the format **YYYY-MM-DD HH:MM:SS**. It has a range of 1970-01-01 00:00:01 UTC to 2038-01-19 03:14:07 UTC.
5. **YEAR** - used for storing year values in 2-digit or 4-digit format **(YYYY or YY)**. If the year is specified with 2 digits, it is assumed to be in the range 1970-2069 (inclusive).

Spatial Datatypes

18 March 2023 08:17

GEOMETRY - The GEOMETRY data type is a generic spatial data type that can store any type of geometric data, including points, lines, and polygons.

ST_ASTEXT(), ST_X(),ST_Y()

JSON

18 March 2023 08:17

[{ 'name' : 'nitish', 'gender' . 'male' }]

ecommerce

pid pname | pric | description (json)

mongodb

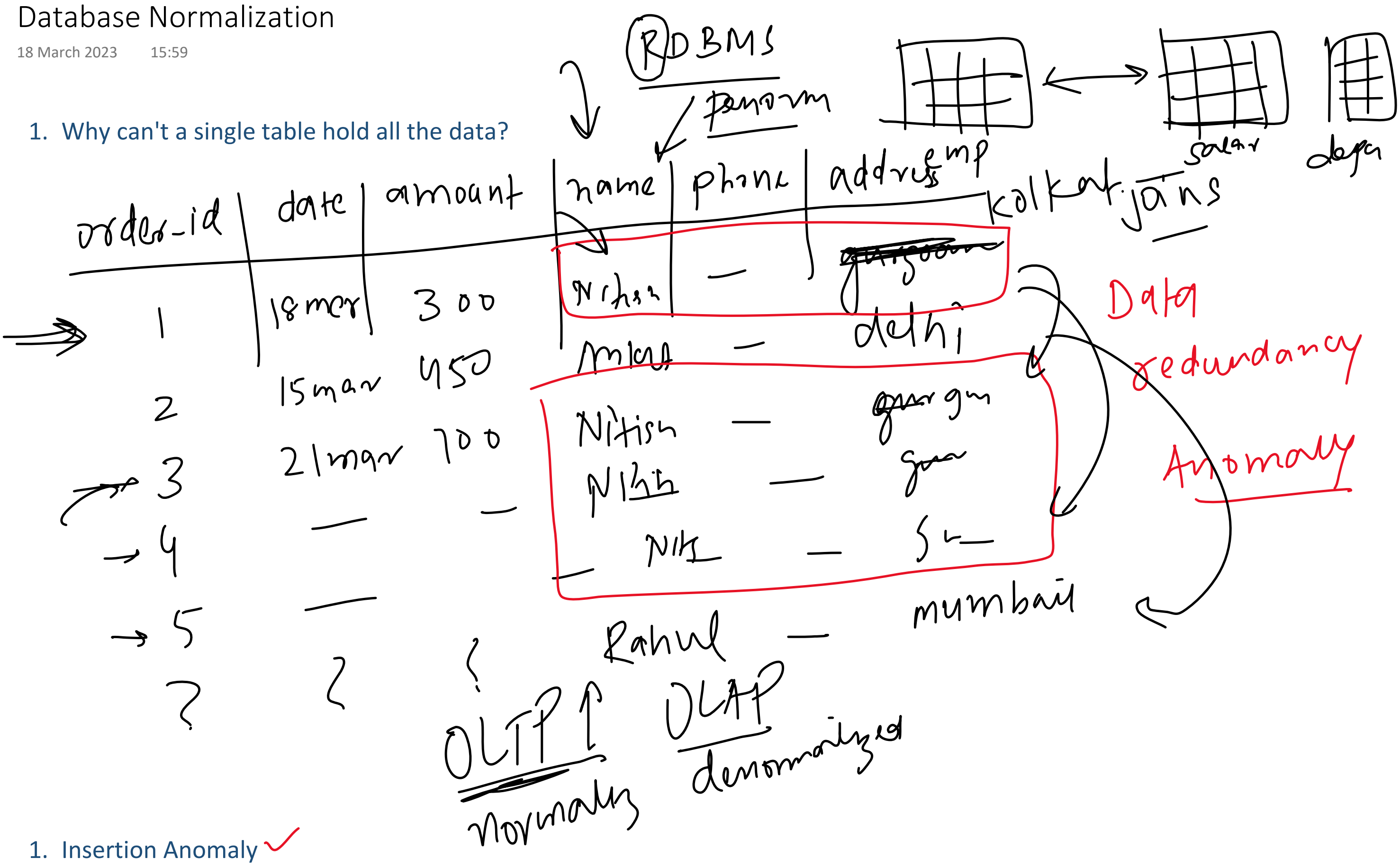
smartpone

pen

Database Normalization

18 March 2023 15:59

1. Why can't a single table hold all the data?



1. Insertion Anomaly ✓
2. Deletion Anomaly ✓
3. Update Anomaly ✓
4. What is the solution? -> Normalization

Database normalization is a process used to organize data in a database to reduce data redundancy and dependency. The goal of normalization is to ensure that each piece of data is stored in one place, in a structured way, to minimize the risk of inconsistencies and improve the overall efficiency and usability of the database.

There are several levels of database normalization, each with its own set of rules and guidelines. The most commonly used levels of normalization are:

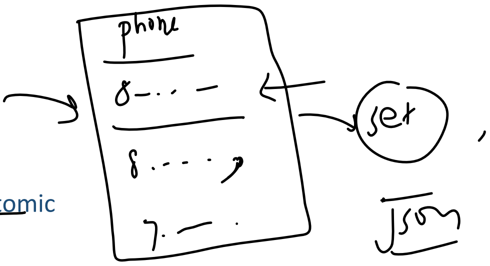
a. **First Normal Form (1NF)**: This level requires that all data in a table is stored in a way that each column contains only atomic (indivisible) values, and there are no repeating groups or arrays.
b. **Second Normal Form (2NF)**: This level requires that each non-key attribute in a table is dependent on the entire primary key, not just a part of it.
c. **Third Normal Form (3NF)**: This level requires that each non-key attribute in a table is dependent only on the primary key and not on any other non-key attributes.

There are higher levels of normalization, such as Fourth Normal Form (4NF) and Fifth Normal Form (5NF), but they are less commonly used in practice.

BCNF

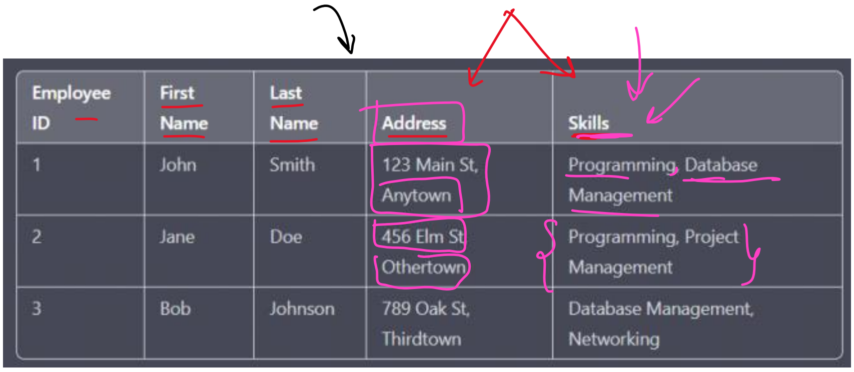
1st Normal Form

18 March 2023 15:59



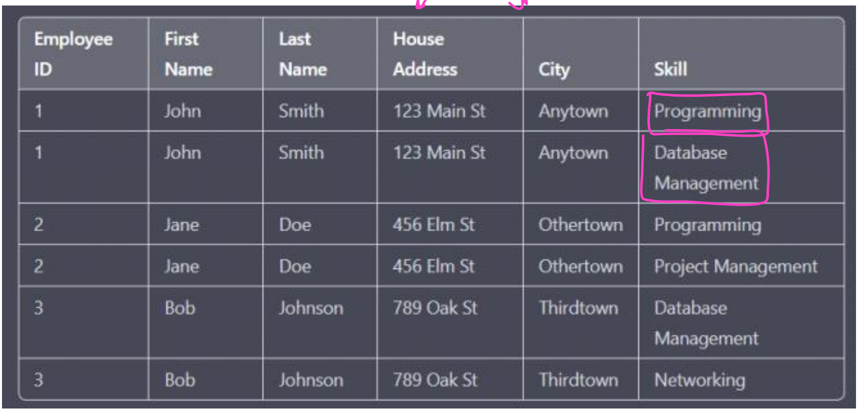
A table is in 1 NF if:

a. There are only Single Valued Attributes or each col should contain atomic values.
b. Attribute Domain does not change -> data type should not change.
c. There is a unique name for every Attribute/Column.
d. The order in which data is stored does not matter.

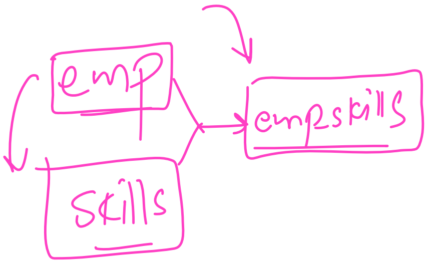


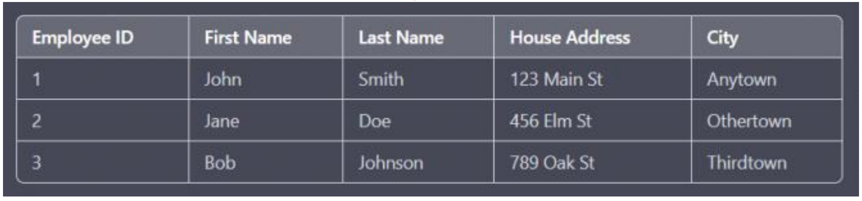
Employee ID	First Name	Last Name	Address	Skills
1	John	Smith	123 Main St, Anytown	Programming, Database Management
2	Jane	Doe	456 Elm St Othertown	Programming, Project Management
3	Bob	Johnson	789 Oak St, Thirdtown	Database Management, Networking

1 NF

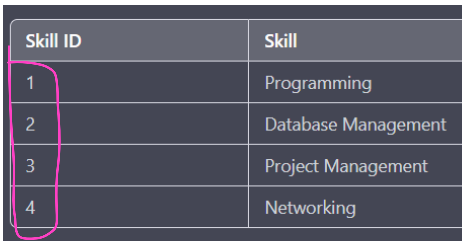


Employee ID	First Name	Last Name	House Address	City	Skill
1	John	Smith	123 Main St	Anytown	Programming
1	John	Smith	123 Main St	Anytown	Database Management
2	Jane	Doe	456 Elm St	Othertown	Programming
2	Jane	Doe	456 Elm St	Othertown	Project Management
3	Bob	Johnson	789 Oak St	Thirdtown	Database Management
3	Bob	Johnson	789 Oak St	Thirdtown	Networking

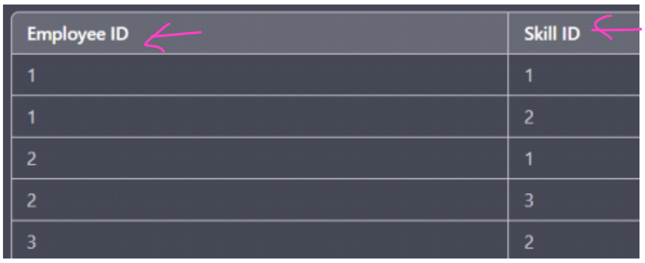




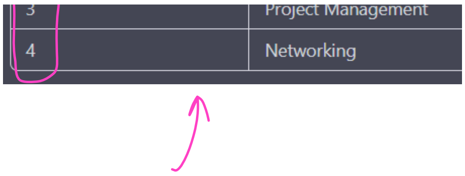
Employee ID	First Name	Last Name	House Address	City
1	John	Smith	123 Main St	Anytown
2	Jane	Doe	456 Elm St	Othertown
3	Bob	Johnson	789 Oak St	Thirdtown



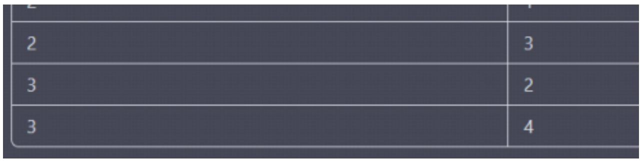
Skill ID	Skill
1	Programming
2	Database Management
3	Project Management
4	Networking



Employee ID	Skill ID
1	1
1	2
2	1
2	3
3	2



3	Project Management
4	Networking



2	3
3	2
3	4

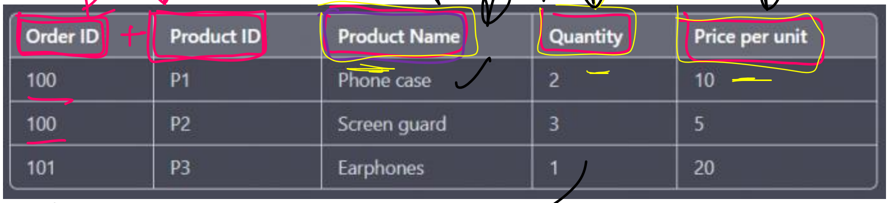
2nd Normal Form

18 March 2023 16:01

A table is in 2NF if:

1. It is already in 1NF
2. It does not contain any partial dependency

order details



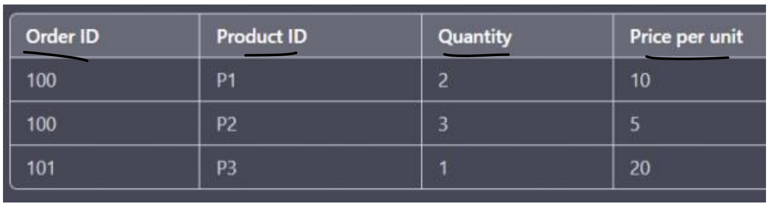
Order ID	Product ID	Product Name	Quantity	Price per unit
100	P1	Phone case	2	10
100	P2	Screen guard	3	5
101	P3	Earphones	1	20

102

phonecase

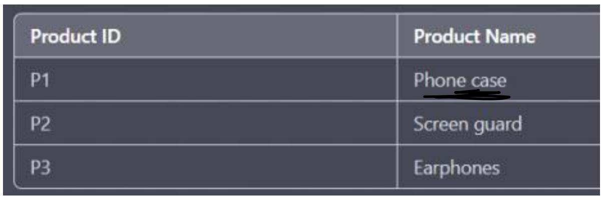
Partial dependency occurs when a non-key attribute is dependent on only a part of the primary key instead of the entire key.

order_details



Order ID	Product ID	Quantity	Price per unit
100	P1	2	10
100	P2	3	5
101	P3	1	20

product



Product ID	Product Name
P1	Phone case
P2	Screen guard
P3	Earphones

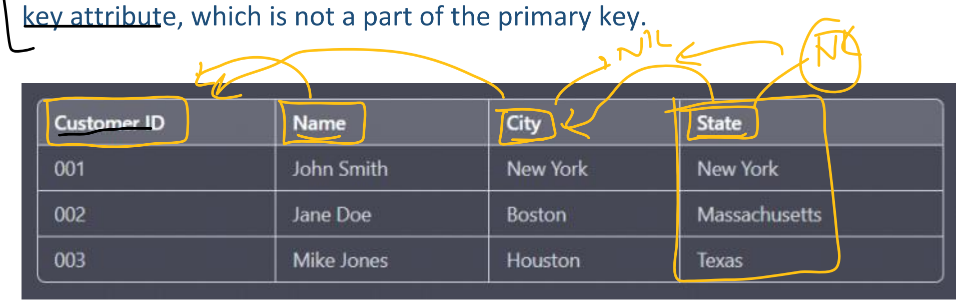
3rd Normal Form

18 March 2023 16:01

A table is in 3NF if:

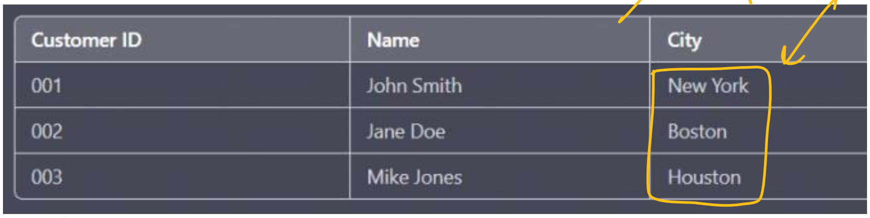
1. If it is already in 2NF
2. There is no transitive dependency.

A transitive dependency exists when a non-key attribute depends on another non-key attribute, which is not a part of the primary key.

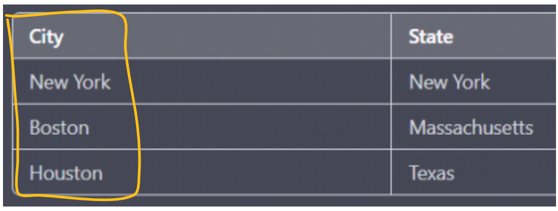

NIL
NO

Customer ID	Name	City	State
001	John Smith	New York	New York
002	Jane Doe	Boston	Massachusetts
003	Mike Jones	Houston	Texas

city id
customer
location



Customer ID	Name	City
001	John Smith	New York
002	Jane Doe	Boston
003	Mike Jones	Houston



City	State
New York	New York
Boston	Massachusetts
Houston	Texas

004
new york

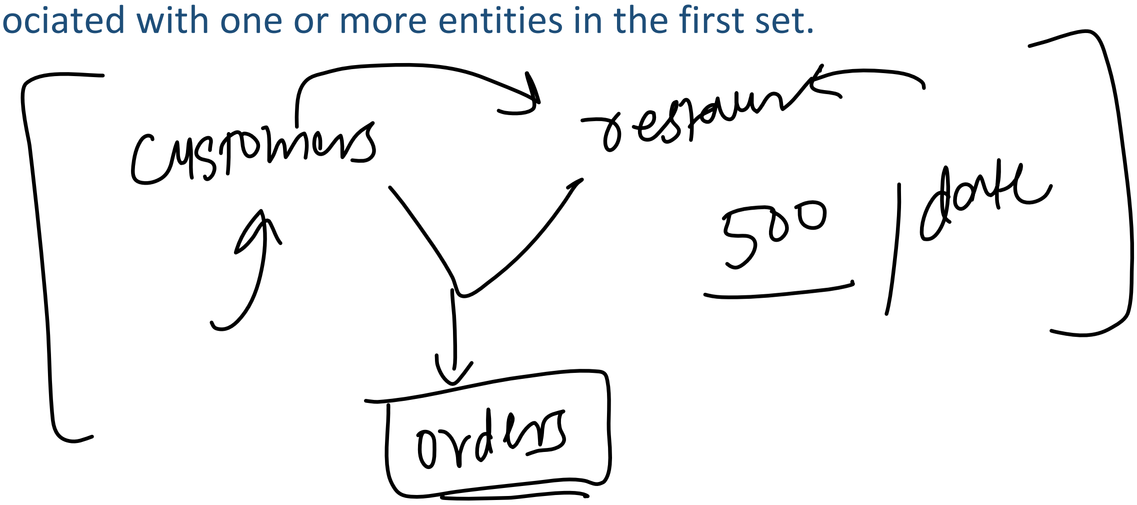
ER Diagram

18 March 2023 17:47

ER diagram stands for Entity-Relationship diagram. It is a graphical representation of entities and their relationships to each other. ER diagrams are used in database design to visualize the entities, attributes, and relationships involved in a system.

There are three basic types of relationships in an ER diagram:

a. One-to-One (1:1): Each entity in one set is associated with only one entity in the other set, and vice versa.
b. One-to-Many (1:N): Each entity in one set is associated with one or more entities in the other set, but each entity in the other set is associated with only one entity in the first set.
c. Many-to-Many (N:M): Each entity in one set is associated with one or more entities in the other set, and each entity in the other set is associated with one or more entities in the first set.


customers
restaurant
500 | date
orders