

Structured Query Language

The way of storing relational data

Data Engineering

How the data will be used in the future so that the format you use will make sense. Here are some of the questions you might want to consider

- How do I store multimodal data, e.g., a sample that might contain both images and texts?
- Where do I store my data so that it's cheap and still fast to access?

Data Engineering

Format	Binary/Text	Human-readable	Example use cases
JSON	Text	Yes	Everywhere
CSV	<i>Text</i>	<i>Yes</i>	<i>Everywhere</i>
<i>Parquet</i>	<i>Binary</i>	<i>No</i>	<i>Hadoop, Amazon Redshift</i>
Avro	Binary primary	No	Hadoop
Protobuf	Binary primary	No	Google, TensorFlow (TFRecord)
Pickle	Binary	No	Python, PyTorch serialization

Row-Major and Column Major

- Row-major formats are better when you have to do a lot of writes, whereas column-major ones are better when you have to do a lot of column-based reads.

Column-major:

- Data is stored and retrieved column by column
- Good for accessing features

Row-major:

- Data is stored and retrieved row by row
- Good for accessing samples

	Column 1	Column 2	Column 3
Example 1
Example 2
Example 3

Row-Major and Column Major

- Consider that you want to store the number 1000000. If you store it in a text file, it'll require 7 characters, and if each character is 1 byte, it'll require 7 bytes. If you store it in a binary file as int32, it'll take only 32 bits or 4 bytes.

```
In [2]: df = pd.read_csv("data/interviews.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17654 entries, 0 to 17653
Data columns (total 10 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Company         17654 non-null  object
1   Title           17654 non-null  object
2   Job             17654 non-null  object
3   Level           17654 non-null  object
4   Date            17652 non-null  object
5   Upvotes         17654 non-null  int64
6   Offer           17654 non-null  object
7   Experience       16365 non-null  float64
8   Difficulty       16376 non-null  object
9   Review          17654 non-null  object
dtypes: float64(1), int64(1), object(8)
memory usage: 1.3+ MB
```

```
In [3]: Path("data/interviews.csv").stat().st_size
```

```
Out[3]: 14200063
```



```
In [4]: df.to_parquet("data/interviews.parquet")
Path("data/interviews.parquet").stat().st_size
```

```
Out[4]: 6211862
```



Data Models

- Data models describe how data is represented. Consider cars in the real world. In a database, a car can be described using its make, its model, its year, its color, and its price
- Alternatively, you can also describe a car using its owner, its license plate, and its history of registered addresses. This is another data model for cars.
- Two types of Data Models: Relational models and NoSQL models.

Data Models

- Data models describe how data is represented. Consider cars in the real world. In a database, a car can be described using its make, its model, its year, its color, and its price
- Alternatively, you can also describe a car using its owner, its license plate, and its history of registered addresses. This is another data model for cars.
- Two types of Data Models: Relational models and NoSQL models.

Data Models

- Data models describe how data is represented. Consider cars in the real world. In a database, a car can be described using its make, its model, its year, its color, and its price
- Alternatively, you can also describe a car using its owner, its license plate, and its history of registered addresses. This is another data model for cars.
- Two types of Data Models: Relational models and NoSQL models.

Relational Data Model

Title	Author	Format	Publisher	Country	Price
Harry Potter	J.K. Rowling	Paperback	Banana Press	UK	\$20
Harry Potter	J.K. Rowling	E-book	Banana Press	UK	\$10
Sherlock Holmes	Conan Doyle	Paperback	Guava Press	US	\$30
The Hobbit	J.R.R. Tolkien	Paperback	Banana Press	UK	\$30
Sherlock Holmes	Conan Doyle	Paperback	Guava Press	US	\$15



Title	Author	Format	Publisher ID	Price
Harry Potter	J.K. Rowling	Paperback	1	\$20
Harry Potter	J.K. Rowling	E-book	1	\$10
Sherlock Holmes	Conan Doyle	Paperback	2	\$30
The Hobbit	J.R.R. Tolkien	Paperback	1	\$30
Sherlock Holmes	Conan Doyle	Paperback	2	\$15

Publisher ID	Publisher	Country
1	Banana Press	UK
2	Guava Press	US

NoSQL Data Model

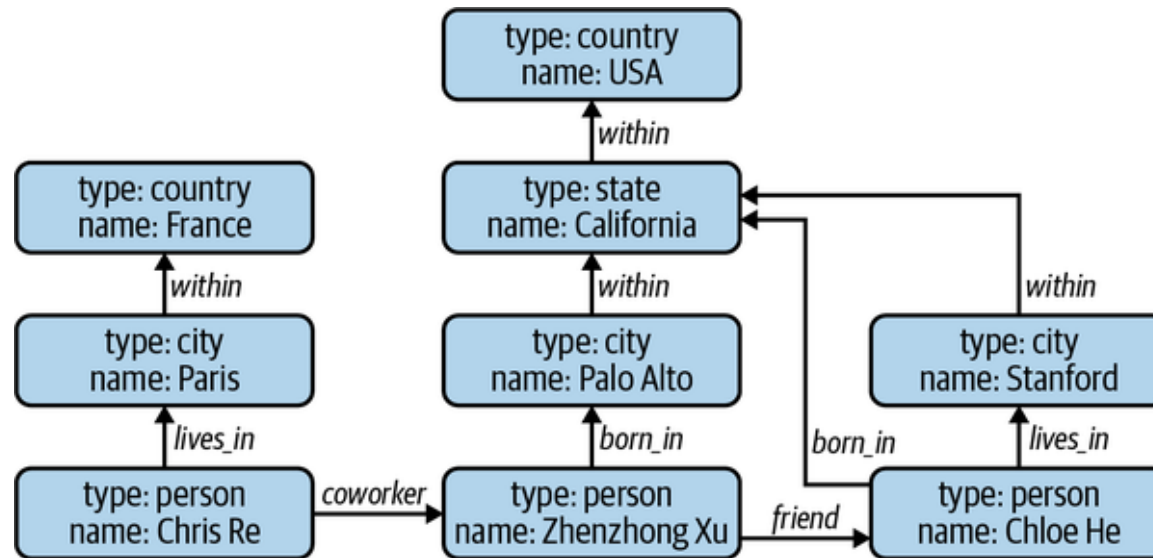
- All documents in a document database are assumed to be encoded in the same format.
- Each document has a unique key that represents that document, which can be used to retrieve it.
- A document is often a single continuous string, encoded as JSON, XML, or a binary format like BSON (Binary JSON)

```
{
  "Title": "Harry Potter",
  "Author": "J .K. Rowling",
  "Publisher": "Banana Press",
  "Country": "UK",
  "Sold as": [
    {"Format": "Paperback", "Price": "$20"},
    {"Format": "E-book", "Price": "$10"}
  ]
}
```

```
{
  "Title": "Sherlock Holmes",
  "Author": "Conan Doyle",
  "Publisher": "Guava Press",
  "Country": "US",
  "Sold as": [
    {"Format": "Paperback", "Price": "$30"},
    {"Format": "E-book", "Price": "$15"}
  ]
}
```

Graph Data Model

- The graph model is built around the concept of a “graph.”
- A graph consists of nodes and edges, where the edges represent the relationships between the nodes.
- A database that uses graph structures to store its data is called a graph database.



Structured and Unstructured

Structured data	Unstructured data
Schema clearly defined	Data doesn't have to follow a schema
Easy to search and analyze	Fast arrival
Can only handle data with a specific schema	Can handle data from any source
Schema changes will cause a lot of troubles	No need to worry about schema changes (yet), as the worry is shifted to the downstream applications that use this data
Stored in data warehouses	Stored in data lakes

Declarative & Imperative

- **In the declarative paradigm**, you specify the outputs you want, and the computer figures out the steps needed to get you the queried outputs.
- **In the imperative paradigm**, you specify the steps needed for an action and the computer executes these steps to return the outputs

OLTP vs OLAP

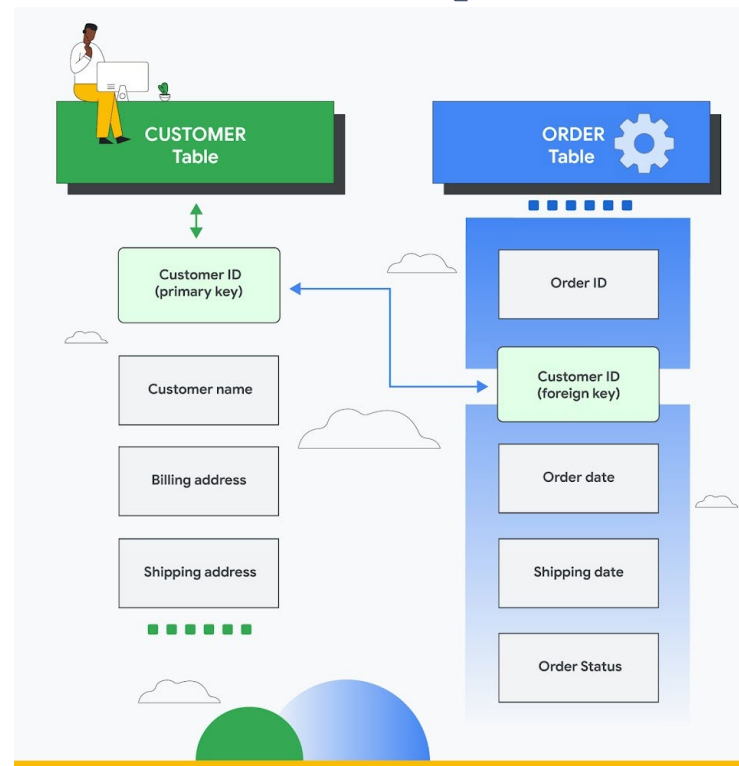
- **OLTP: Online Transaction Processing**
- OLTP systems are designed to support everyday transaction-oriented applications in industries such as banking, retail, logistics, etc.
- Prioritizes fast query processing and maintaining data integrity in multi-access environments.
- Data is often current, not historical.
- **Examples:** A bank's system where customers withdraw or deposit money; a retailer's system where customers make purchases.

OLTP vs OLAP

- **OLAP: Online Analytical Processing**
- OLAP systems are designed to support complex queries and offer business insights. They facilitate multi-dimensional analytical queries, providing a platform for business intelligence and data mining.
- Simple relationships with fewer joins.
- Aggregated data.
- Commonly uses schemas like star and snowflake.
- **Examples:** An e-commerce company analysing sales trends over the past year; a system providing business performance metrics.

What is Relational Database

- A relational database is **a collection of information that organizes data in predefined relationships where data is stored in one or more tables (or "relations") of columns and rows**



Different SQL Tools

- **MySQL:** An open-source relational database management system, owned by Oracle Corporation. One of the most popular databases for web-based applications
- **Microsoft SQL:** A relational database management system developed by Microsoft. Used for a variety of applications ranging from small applications to large scale enterprise applications. SQL Server uses T-SQL as its primary querying language
- **PostgreSQL:** It's an open-source relational database management system (RDBMS). Known for its extensibility and SQL compliance. It's not just an SQL processing tool but also offers "NoSQL" capabilities.

Different SQL Tools

- **PL/SQL(Procedural Language for SQL):** Predominantly used in Oracle Databases for writing stored procedures, functions, and triggers.
- **SQLite:** A C-language library that offers a lightweight, disk-based database, which doesn't require a separate server process. It's serverless, self-contained, and zero-configuration.

4 Stages of DBMS

- Database Management Systems are having 4 Important Characteristics.
 - **Data Definition** – Define the data being tracked
 - **Data Manipulation** – Add, Update & Remove the Data
 - **Data Retrieval** – Extract and Report the data available in database
 - **Administration** – defining users on the system, security, monitoring, system administration

Database Tables

- A database table is a lot like a spreadsheet.
- Data is kept in **Columns** and **Rows**.
- Each **Column** is assigned:
 - A Unique **Name**, identifying a human readable name of the column. (ie FIRST_NAME, LAST_NAME)
 - A **Data Type** (ie – String, Date, Time, Number, etc)
 - Optionally, constraints (ie – Is a value required?, Length of String, etc)
- Each **Row** is a distinct database **Record**.

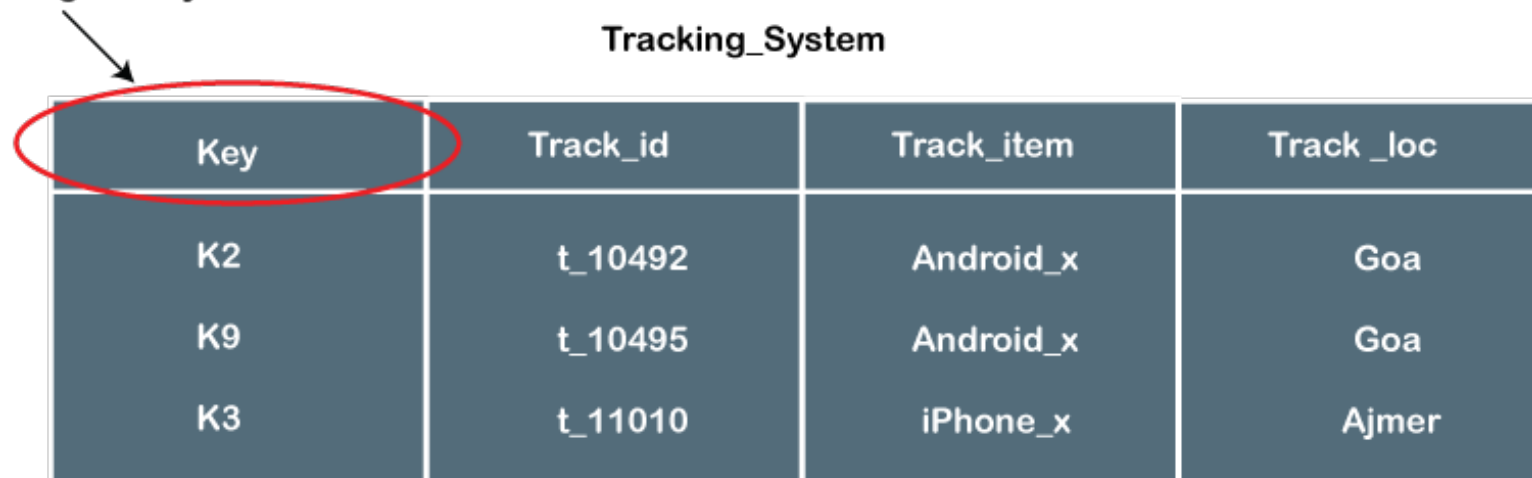
Primary Key & Surrogate Key

- A **Primary Key** is an optional special database column or columns used to identify a database record.

A **Surrogate Key** is a type of **Primary Key** which used a unique generated value.

- Should have no business value, and should never change.

Surrogate Key



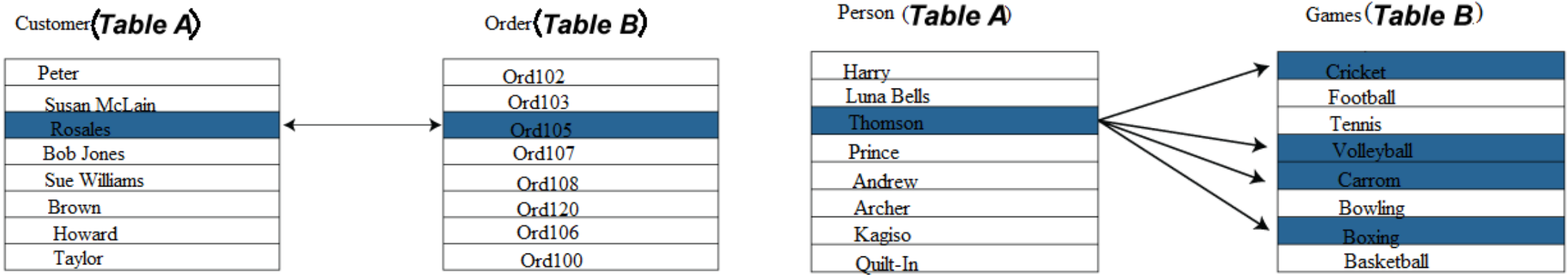
Key	Track_id	Track_item	Track_loc
K2	t_10492	Android_x	Goa
K9	t_10495	Android_x	Goa
K3	t_11010	iPhone_x	Ajmer

Data Relationships

- **One to One** – Record in Table A matches exactly one record in Table B
- **One to Many** – Record in Table A matches many in Table B, but Table B matches only one record

in Table A. (Think – An Order with multiple items)

- **Many to Many** – Record in Table A matches many in Table B, and Table B matches many records in Table A.

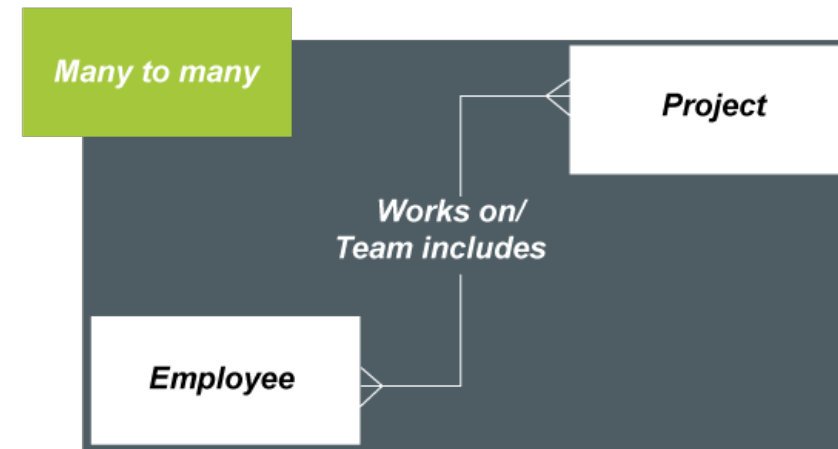
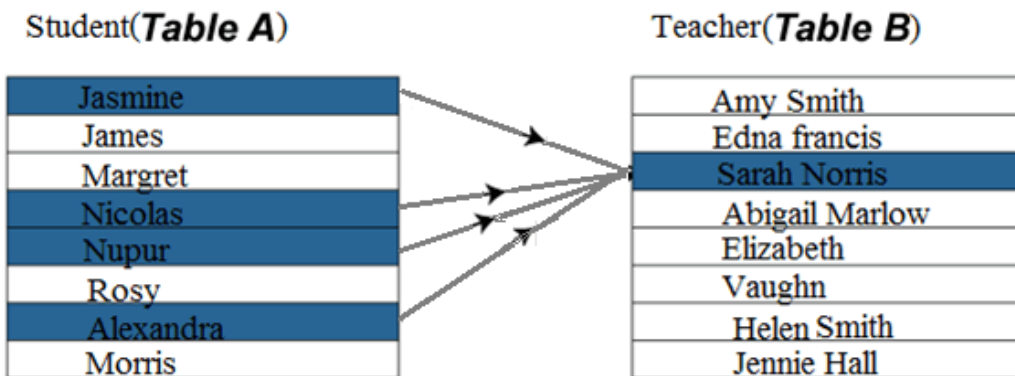


Data Relationships


- **One to One** – Record in Table A matches exactly one record in Table B
- **One to Many** – Record in Table A matches many in Table B, but Table B matches only one record


in Table A. (Think – An Order with multiple items)


- **Many to Many** – Record in Table A matches many in Table B, and Table B matches many records in Table A.

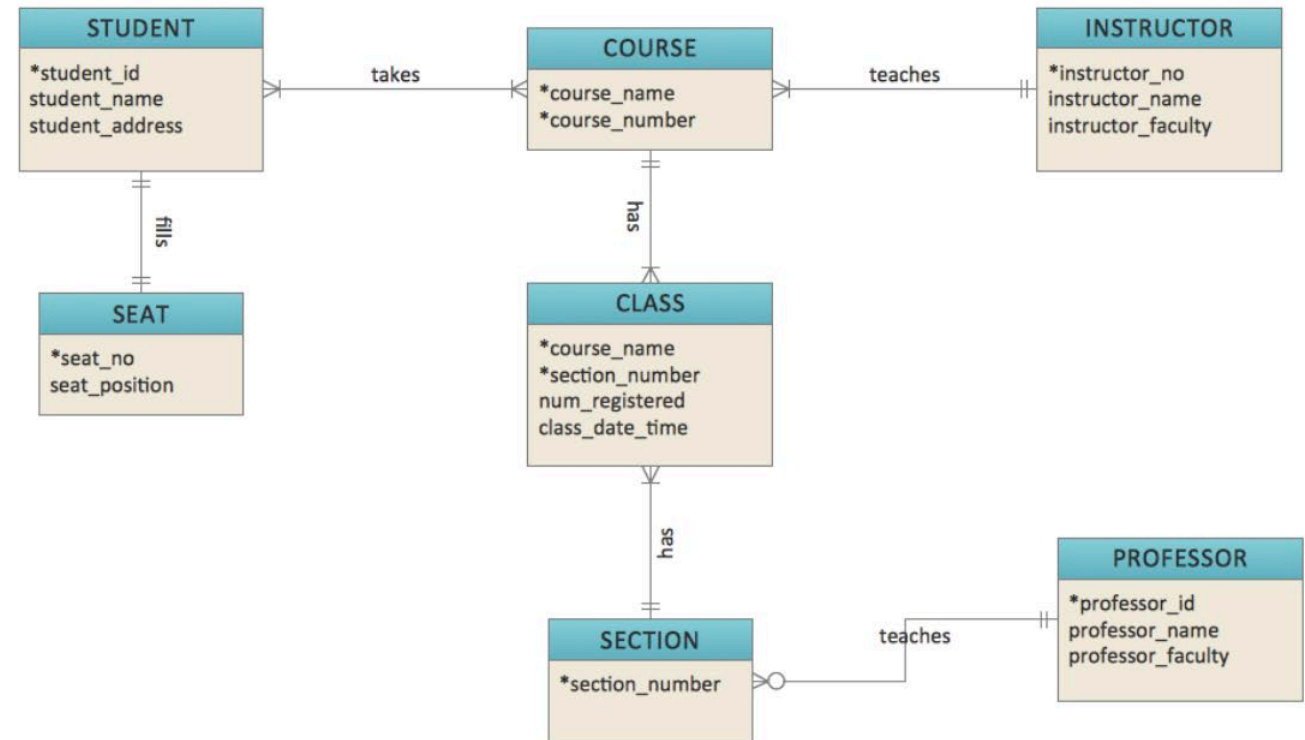


Data Relationships

▣ one to one: 

▣ one to many: 

▣ many to many: 



Data Relationships

ID	FIRST_NAME	LAST_NAME	ADDRESS	CITY	STATE	ZIP_CODE
1234	Michael	Weston	123 Brickel	Miami	FL	33135
1235	Fiona	Glenanne	123 Brickel	Miami	FL	33135
1236	Sam	Axe	222 Taimiami	Miami	FL	33109
1237	Madeline	Weston	945 Sunset Lane	Miami	FL	33114
1238	Jesse	Porter	9973 A1A	Miami	FL	33132
1239	Barry	Burkowski	3838 Orange Grove St	Miami	FL	33314

Table: CUSTOMER



ID	CUSTOMER_ID	DRINK_DESCRIPTION
122249	1234	Scotch
122250	1235	Pina Colada
122251	1236	Budwiser
122252	1237	White Wine
122253	1238	Stone IPA
122254	1239	Tequila Sunrise
122255	1234	Scotch
122256	1235	Pina Colada
122257	1236	Budwiser
122258	1237	Old Fashioned
122259	1238	Corona
122260	1239	Pina Colada

Table: DRINK_ORDER

DDL

- **DDL** – Data Definition Language (ie CREATE TABLE...) is used to define the relational model
- Under the covers, the RDBMS will store data about your tables in catalog tables
- The software is used to enforce data being stored conforms to the rules you've defined for the data.

DML

- **DML** – Data Manipulation Language

Allows you to add (INSERT), change (UPDATE), or remove (DELETE) data.

The RDBMS enforces data manipulation adheres to the rules of the **Data Definition**.

The RDBMS allows set up 'rules' for multi-user systems.

These rules manage what happens in competing conditions. (what happens when two users want to update the same data, at the same time)

Retrieval

- Data Retrieval is the act of pulling data out of the database
- The RDBMS determines the optimal way to retrieve data out of the database.
- Multi-table joins can become very complex.
- Consider tables with billions and billions of rows.
- Reports can go from seconds, to hours when the retrieval strategy is wrong.
- The RDBMS also considers what happens when updates occur while your report is running.

Character Set

- Computers are driven off of binary information – ie 1's and zeros. • A 'bit' is binary one or zero.
- A byte is a collection of eight bits (10000111) = 70
- ASCII – American Standard Code for Information Interchange
- One of the first 'character' sets
- Limited to 128 characters (mostly letters, numbers, common punctuation)
- UTF-8 is highly popular used for email / web. 1 – 4 bytes long.
- Up to 1,112,064 characters

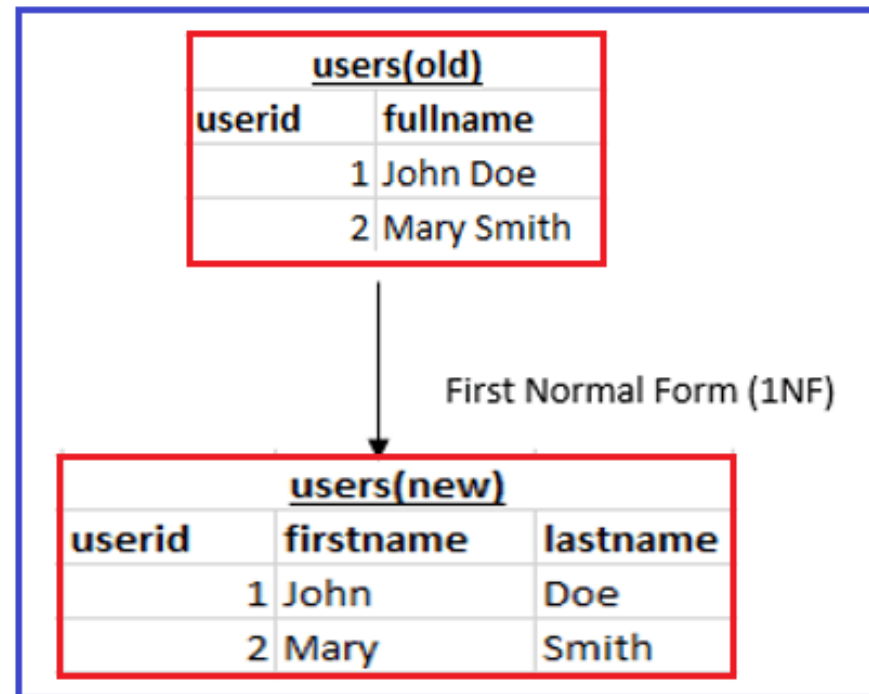
Data Normalization

Database Normalization is the most important factor in Database design or Data modeling. Database Normalization is the process to eliminate data redundancies and store the data logically to make data management easier

- **First Normal Form (1NF)**
- **Second Normal Form (2NF)**
- **Third Normal Form (3NF)**
- **Fourth Normal Form**
- **Fifth Normal Form**
- **Boyce Codd Normal Form(BCNF)**

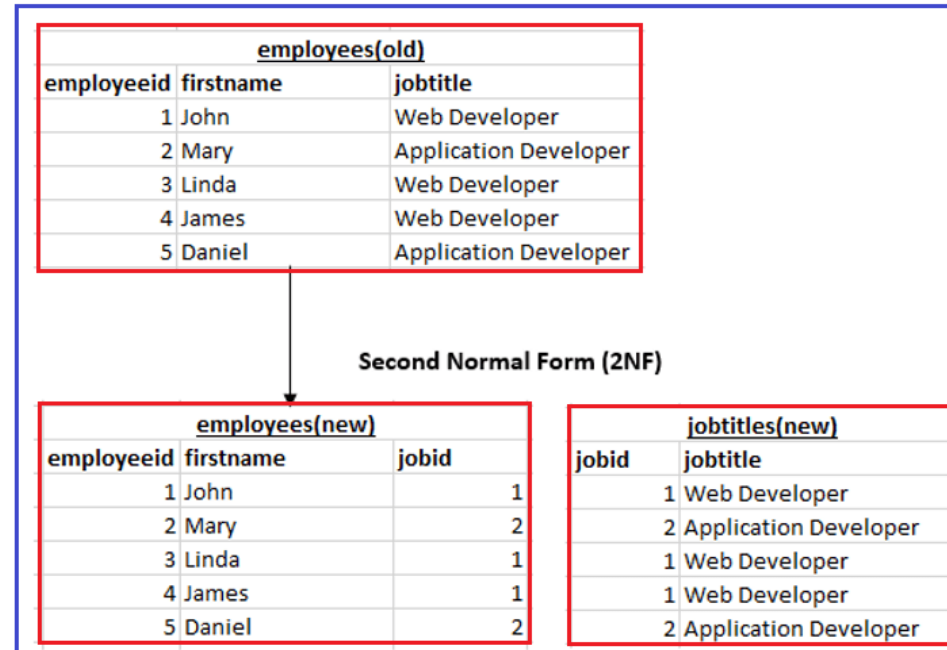
First Normal Form

In the first normal form, each column must contain only one value. No table should store repeating groups of related data. The easiest way to follow the first normal form is to inspect the database table horizontally.



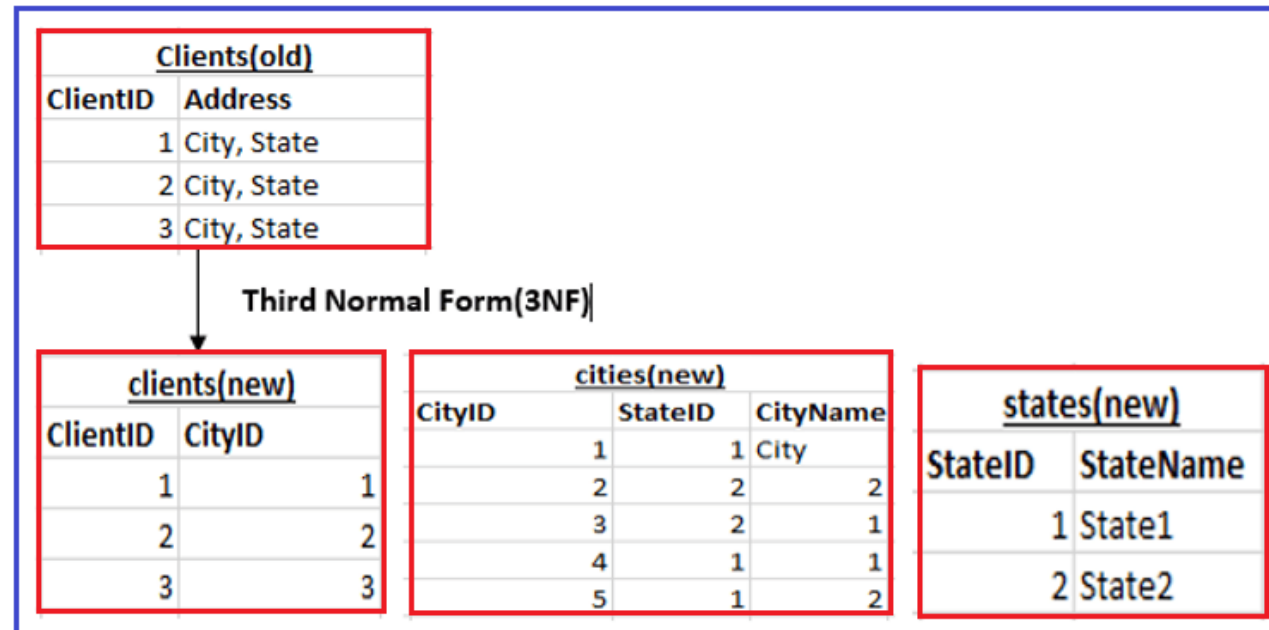
Second Normal Form

In the second normal form, first, the database must be in the first normal form and there should not be any partial dependency. If there are duplicate values in the row, they should be stored in their own separate tables and linked to the table using foreign keys.



Third Normal Form

In the third normal form, the database is already in the third normal form, if it is in the second normal form. Every non-key column must be mutually independent. Identify any columns in the table that are interdependent and break those columns into their own separate tables.



Third Normal Form

Functional Dependency: When there is a relationship exists between the primary key and non-key attribute within a table it is called functional dependency.

- **$X \rightarrow Y$**
- Here, X is known as determinant, and Y is known as the dependent.

Transitive Dependency: When there is an indirect functional dependency between the attributes it is called Transitive Dependency. If $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$ is called Transitive Dependency. To achieve Third Normal Form (3NF) we have to eliminate Functional Dependency.

Boyce Codd Normal Form

Boyce–Codd Normal Form or BCNF is an extension to the third normal form, and is also known as 3.5 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 $A \rightarrow B$, A should be a **super key**.
 - The second point sounds a bit tricky, right? In simple words, it means, that for a dependency $A \rightarrow B$, A cannot be a **non-prime attribute**, if B is a **prime attribute**.

Boyce Codd Normal Form

And, for any dependency $A \rightarrow B$, A should be a **super key**.

- The second point sounds a bit tricky, right? In simple words, it means, that for a dependency $A \rightarrow B$, A cannot be a **non-prime attribute**, if B is a **prime attribute**.

<u>College Enrolment</u>		
StudentID	Subject	Professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

Student Table

<u>Student</u>	
StudentID	ProfessorID
101	1
101	2
102	3
103	4
104	5

Professor Table

<u>Professor</u>		
ProfessorID	Professor	Subject
1	P.Java	Java
2	P.Cpp	C++
3	P.Java2	Java
4	P.Chash	C#
5	P.Java	Java

Fourth Normal Form

For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1. It should be in the **Boyce–Codd Normal Form**.
2. And, the table should not have any **Multi-valued Dependency**

A table is said to have multi-valued dependency, if the following conditions are true,

1. For a dependency $A \twoheadrightarrow B$, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.
2. Also, a table should have at-least 3 columns for it to have a multi-valued dependency.

Denormalization

It is a database optimization technique where we can add redundant data to one or more tables and optimize the efficiency of the database. It is applied after doing normalization. It also avoids costly joins in a relational database. It is used on the already normalized database to increase performance. In denormalization, we are including data from one table to another table to reduce the number of joins in the query which helps in speeding up the performance.

School	
school_id	school_name
1	school1
2	school2

Student	
student_id	student_name
1	Paul
2	Tim

School has Student	
school_id	student_id
1	1
1	2
2	1

School has Student			
school_id	school_name	student_id	student_name
1	schhol1	1	Paul
1	school1	2	Tim
2	school2	1	Paul

Data Integrity & Referential

Data integrity refers to the overall accuracy and consistency of the data in your database. You want high-quality data. People lose confidence in your data when they spot problems like a salary value that contains alpha characters or a percent increase value over 100 percent.

Referential integrity refers to the quality of the *relationships* between the data in your tables. If you have a complaint in the complaint table for a customer that doesn't exist in the customer table, you have a referential integrity problem. By defining customer_id as a foreign key, you can be assured that every customer_id in the complaint table refers to a customer_id that exists in the customer table.

Temporary Table

MySQL allows you to create temporary tables—that is, a temporary result set that will exist only for your current session and then be automatically dropped

To create a temporary table based on the results of a query, simply precede the query with the same **create temporary table** syntax

Common Table Expressions

Temporary result set that you name and can then select from as if it were a table. You can use CTEs only for the duration of one query (versus temporary tables, which can be used for the entire session)

We use the **with** keyword to give the CTE a name

Recursive CTE

Recursion is a technique that is used when an object references itself. When I think of recursion

Recursion is useful when your data is organized as a hierarchy or a series of values where you need to know the previous value to arrive at the current value.

Subquery

A subquery (or inner query) is a query nested within another query. A subquery is used to return data that will be used by the main query. When a query has a subquery, MySQL runs the subquery first, selects the resulting value from the database, and then passes it back to the outer query

Views, Functions & Procedures

- Views are useful in situations where you want to simplify a complex query or hide sensitive or irrelevant data.
- Functions and procedures are programs you can call by name. Because they're saved in your MySQL database, they are sometimes called *stored* functions and procedures.
- Collectively, they are referred to as *stored routines* or *stored programs*.
- When you write a complex SQL statement or a group of statements with several steps, you should save it as a function or procedure so you can easily call it by name later.
- Functions and procedures are saved in the database where you created them

Views, Functions & Procedures

- Function gets called from a SQL statement and always returns one value
- A procedure, on the other hand, gets called explicitly using a call statement.
- While procedures may return no values, one value, or many values, a function accepts arguments, performs some task, and returns a single value.
- Procedures are often used to execute business logic by updating, inserting, and deleting records in tables, and they can also be used to display a dataset from the database.
- Functions are used for smaller tasks, like getting one piece of data from the database or formatting a value. Sometimes you can implement the same functionality as either a procedure or a function.

Triggers

- Triggers are most often used to track changes made to a table or to enhance the data's quality before it's saved to the database.
- Like functions and procedures, triggers are saved in the database in which you create them.
- Triggers can be set to fire either before or after rows are changed.
- You can also write triggers that fire *before* rows are changed in a table, to change the data that gets written to tables or prevent rows from being inserted or deleted. This can help improve the quality of your data before you save it to the database.

Events

- Events can be scheduled to run once or at some interval, like daily, weekly, or yearly; for example, you might create an event to perform weekly payroll processing. You can use events to schedule long-running processing during off-hours, like updating a billing table based on orders that came in that day
- MySQL has an *event scheduler* that manages the scheduling and execution of events.