Databases(storage)

A database is a collection of inter-related data that models some aspects of our real world.

DBMS

A database management system DBMS is software that manages a database like (MySQL, Oracle, Microsoft SQL server).

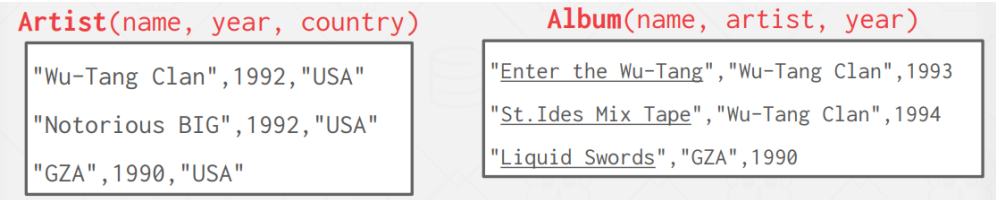
SQL

Structured Query Language is a computer language used for retrieval and management of data in the databases.

- Database interprets and make sense of SQL instruction with the use of DBMS.

Flat Files

In the past, flat files were the files we store data in. Database is stored as a comma separated value (CSV) and each entity will be stored in its own file like :



Issues with flat files

1) Data Integrity (Entity, Domain, Referential).
2) Implementation.
3) Durability.

Data Models.

A collection of concepts which describes the data in the database.

- Relational model (most common).

- NoSQL.

Database Schema.

Schema is about the structure of the database. It's a description of particular collection of data based on data model.

1) logical schema: describes the structure of the entire database for all users. It describes the structure in terms of entities and features of entities and the relationship between them. An Entity Relationship Diagram (ER-D) is usually drawn to represent the logical schema. At this level, you have no information about physical storage and retrieval data.

2) physical schema: describes how data stored on disk in the form of tables, columns, and rows.

- Database applications were difficult to build and maintain because there was a tight coupling between logical and physical layers. Early on, the physical layer was defined in the application code, so if we wanted to change the physical layer the application was using, we would have to change all the code to match the new physical layer.

So, Ted Codd proposes the relational model to avoid this.

Relational Model.

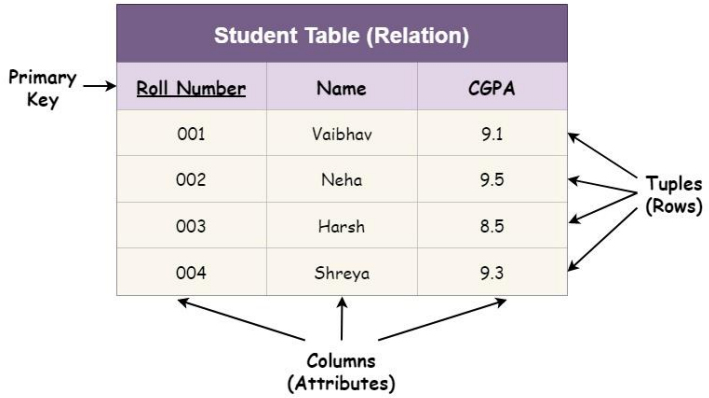
The relational model defines database abstraction based on relations to avoid maintenance overhead.

- A relation is an unordered set that contains the relationship of attributes that represent entities. Since the relationships are unordered, the DBMS can store them in any way it wants, allowing for optimization, A relation with n attributes is called an n-ary relation.

The relational data model defines three concepts:

- **Structure**: The definition of relations and their contents. This is the attributes the relations have and the values that those attributes can hold.
- **Integrity**: Ensure the database's contents satisfy constraints. An example constraint would be that any value for the year attribute must be a number.
- **Manipulation**: How to access and modify a database's contents

Relational Model in DBMS



Data Integrity

The following categories of data integrity exist with each Relational model.

- **Entity Integrity –** ensuring that each row of a table has a unique and non-null primary key value.
- **Domain Integrity** – Ensures that you use the correct value in the correct datatype.
- **Referential integrity –** Rows cannot be deleted, which are used by other records like foreign key.

Employee_id	Name	Salary	Age
1	Andrew	486522	25
2	Angel	978978	30
3	Anamika	697abc	35

This value is out of domain(not INTEGER)so it is not acceptable.

DOMAIN INTEGRITY

Primary Key

ID	Customer_Name	Age
1	Andrew	18
2	Angel	20
	Angel	20

This value cannot be NULL as we will not be able to identify customers uniquely

ENTITY INTEGRITY

artist_id	artist_name
1	Bono
2	Cher
3	Nuno Bettencourt

Link Broken

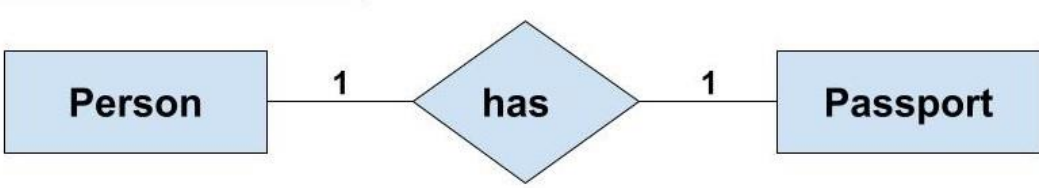
artist_id	album_id	album_name
3	1	Schizophonic
4	2	Eat the rich
3	3	Crave (single)

Relationships.

One-to-One Relationship

Such a relationship exists when each record of one table is related to only one record of the other table.

For example, If there are two entities 'Person' (Id, Name, Age, Address) and 'Passport'(Passport_id, Passport_no). So, each person can have only one passport and each passport belongs to only one person.

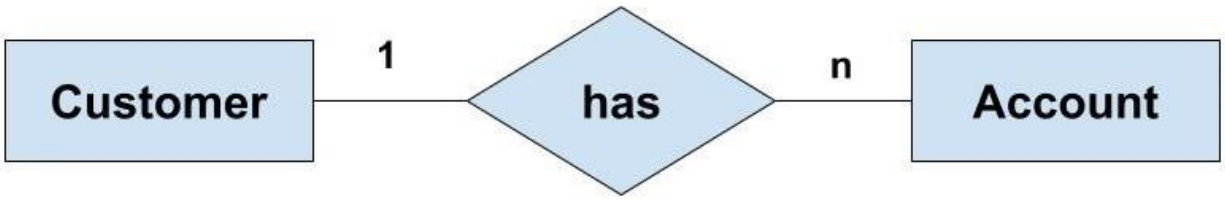


Such a relationship is not very common. However, such a relationship is used for security purposes. In the above example, we can easily store the passport id in the 'Person' table only. But, we make another table for the 'Passport' because Passport number may be sensitive data and it should be hidden from certain users. So, by making a separate table we provide extra security that only certain database users can see it.

One-to-Many or Many-to-One Relationship

Such a relationship exists when each record of one table can be related to one or more than one record of the other table. This relationship is the most common relationship found. A one-to-many relationship can also be said as a many-to-one relationship depending upon the way we view it.

For example, If there are two entity type ‘Customer’ and ‘Account’ then each ‘Customer’ can have more than one ‘Account’ but each ‘Account’ is held by only one ‘Customer’. In this example, we can say that each Customer is associated with many Account. So, it is a one-to-many relationship. But, if we see it the other way i.e many Account is associated with one Customer then we can say that it is a many-to-one relationship.



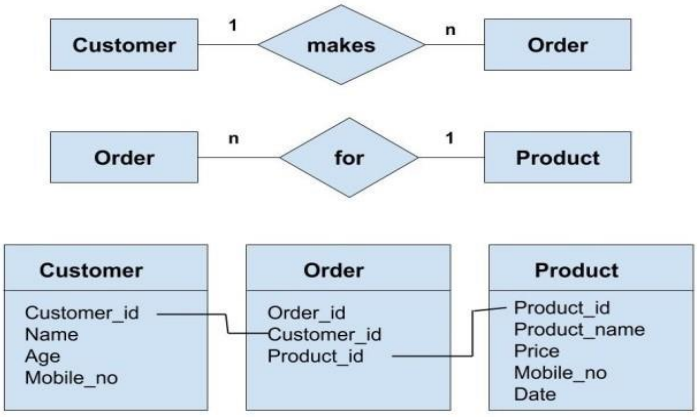
Many-to-Many Relationship

Such a relationship exists when each record of the first table can be related to one or more than one record of the second table and a single record of the second table can be related to one or more than one record of the first table. A many-to-many relationship can be seen as a two one-to-many relationship which is linked by a 'linking table' or 'associate table'. The linking table links two tables by having

fields which are the primary key of the other two tables. We can understand this with the following example.

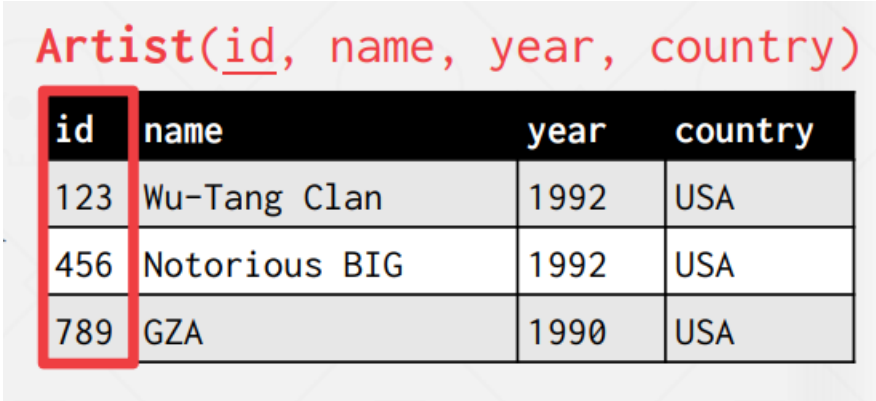
Example: If there are two entity type ‘Customer’ and ‘Product’ then each customer can buy more than one product and a product can be bought by many different customers.

Now, to understand the concept of the linking table here, we can have the ‘Order’ entity as a linking table which links the ‘Customer’ and ‘Product’ entity. We can break this many-to-many relationship in two one-to-many relationships. First, each ‘Customer’ can have many ‘Order’ whereas each ‘Order’ is related to only one ‘Customer’. Second, each ‘Order’ is related only one Product wheres there can many orders for the same Product.



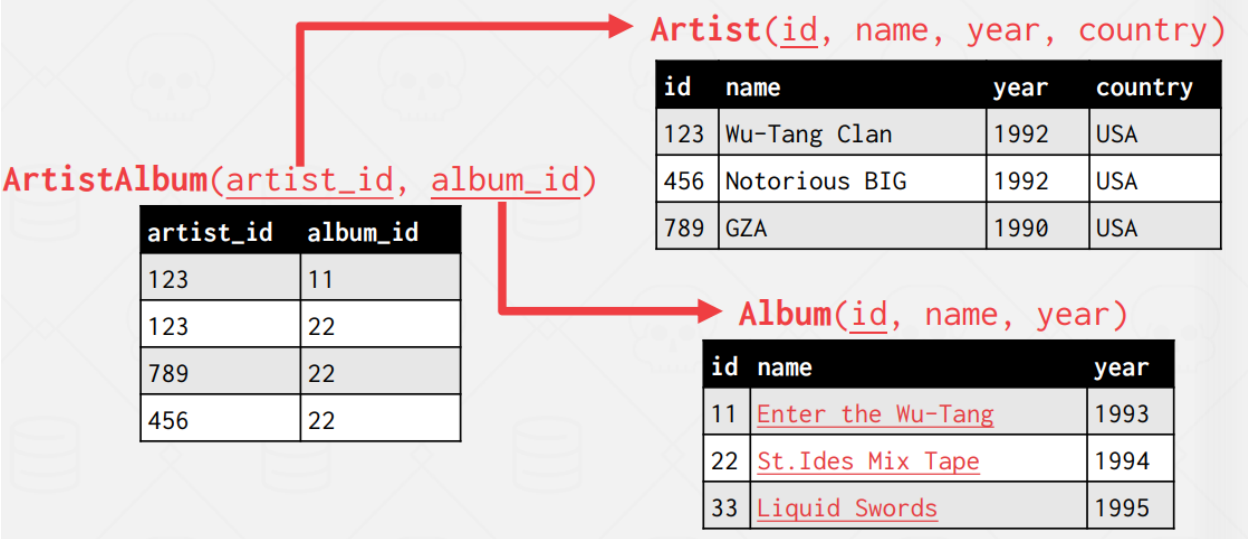
Relational Model: Primary Keys.

Primary key uniquely identifies a single tuple (row).



Relational Model: Foreign Keys.

It defines how an attribute (column) from one relation has to map to a tuple (row) in another relation.



Data Manipulation Languages (DMLs).

Methods to manipulate and retrieve the data in database.

- **Procedural:** defines how DBMS should use to find desired output based on sets (Relational algebra).
- **Non Procedural (decelerative)** : specifies only what data is wanted (Relational calculus).

Relational Algebra.

Relational Algebra is a set of operations to retrieve and manipulates tuples in relations. Each operator takes one or more relations as input and put the result in output relation.

Select operator.

- select operator takes in a relation and outputs relation with some tuples under a condition.

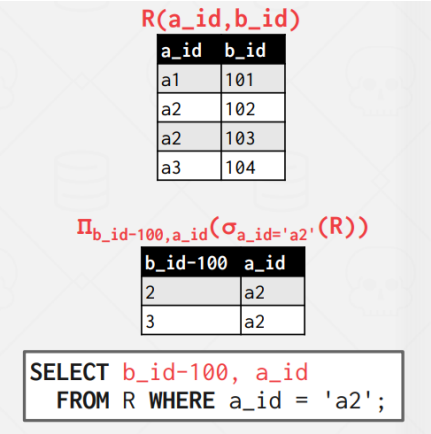
Syntax: $\sigma_{\text{predicate}}(R)$.

Example: $\sigma_{\text{a\_id='a2'}}(R)$

SQL: `SELECT * FROM R WHERE a_id = 'a2'`

Projection operator.

- Projection takes in a relation and outputs relation with attributes that we put on the predicate. And most of the time we use it with select operator.



Union operator.

- Union takes in a two relations and outputs relation with all tuples that at least in one of the two input relations. The two relations must have same attributes.

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

```
(SELECT * FROM R)
UNION ALL
(SELECT * FROM S);
```

(R ∪ S)

a_id	b_id
a1	101
a2	102
a3	103
a3	103
a4	104
a5	105

Intersection operator.

- Intersection takes in a two relations and outputs relation with tuples that appear in the input relations. The two relations must have same attributes.

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

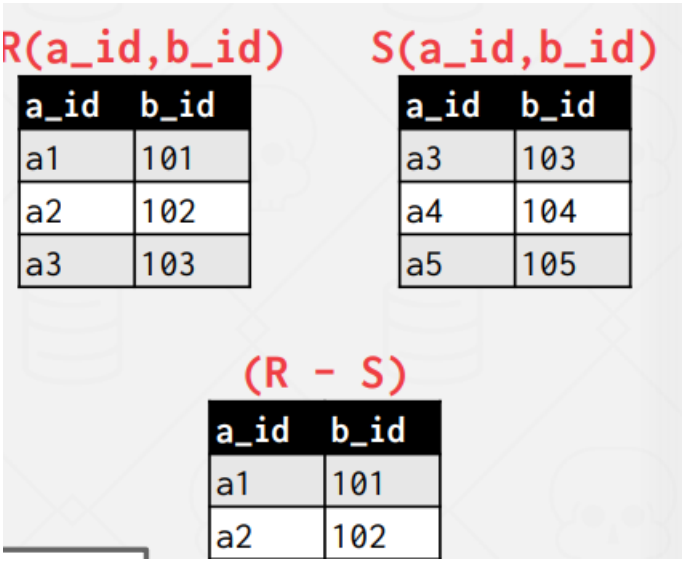
(R ∩ S)

a_id	b_id
a3	103

```
(SELECT * FROM R)
INTERSECT
(SELECT * FROM S);
```

Difference operator.

- difference takes in a two relations and outputs relation with tuples that appear in the 1st relation and not in 2nd relation. The two relations must have same attributes.



R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R - S)

a_id	b_id
a1	101
a2	102

```
(SELECT * FROM R)
EXCEPT
(SELECT * FROM S);
```

Product operator.

Generate a relation that contains all possible combinations of tuples from the input relations.

Syntax: (R × S)

```
SELECT * FROM R CROSS JOIN S;
```

```
SELECT * FROM R, S;
```

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R × S)

R.a_id	R.b_id	S.a_id	S.b_id
a1	101	a3	103
a1	101	a4	104
a1	101	a5	105
a2	102	a3	103
a2	102	a4	104
a2	102	a5	105
a3	103	a3	103
a3	103	a4	104
a3	103	a5	105

Join operator.

Generate a relation that contains all tuples that are a combination of two tuples (one from each input relation) with a common value(s) for one or more attributes.

Syntax: (R ⋈ S)

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R ⋈ S)

a_id	b_id
a3	103

```
SELECT * FROM R NATURAL JOIN S;
```

```
SELECT * FROM R JOIN S USING (a_id, b_id);
```

So, Relational algebra defines the primitives for processing queries on a relational database. it is the de facto standard for writing queries on relational model databases.