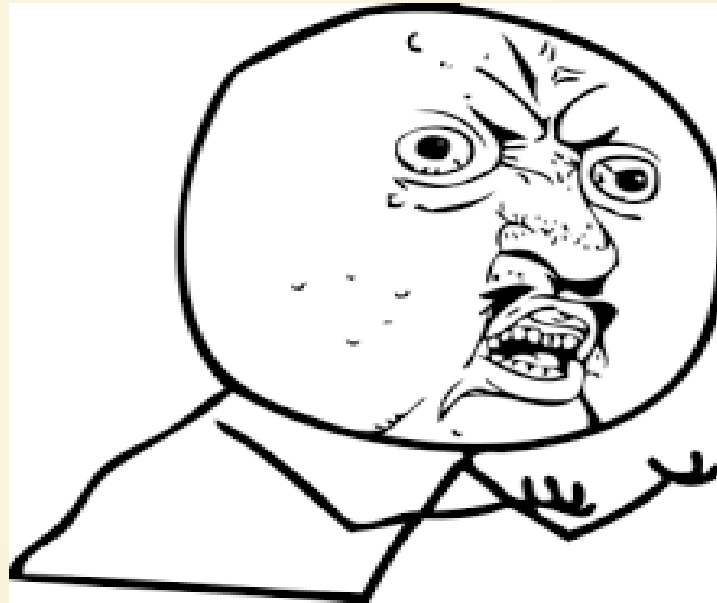


DATABASE

WHAT IS DATABASE

1. Program that allows for the storage and retrieval of data.
2. There are two types of databases: relational and non-relational.

BUT WHY DATABASE



PROBLEMS IF NOT USING DATABASE

1. Size of data
2. Ease of Updating Data
3. Accuracy
4. Security
5. Redundancy
6. Incomplete Data

TABLES AND PRIMARY KEY

1. Database is made up of (often multiple) tables.
2. Example:

Customers				
customer_id	username	email	password	birthday
1	william	william@example.com	kUdGR9ZWOC	10031980
2	john	john@example.com	dp3lylrkSh	5051982
3	sara	sara@example.com	qowIEPot7i	1132010
4	alice	alice@example.com	DcJFq3CXAK	10242003

3. Each row is known as a record
4. Each record has a primary key

PRIMARY KEY

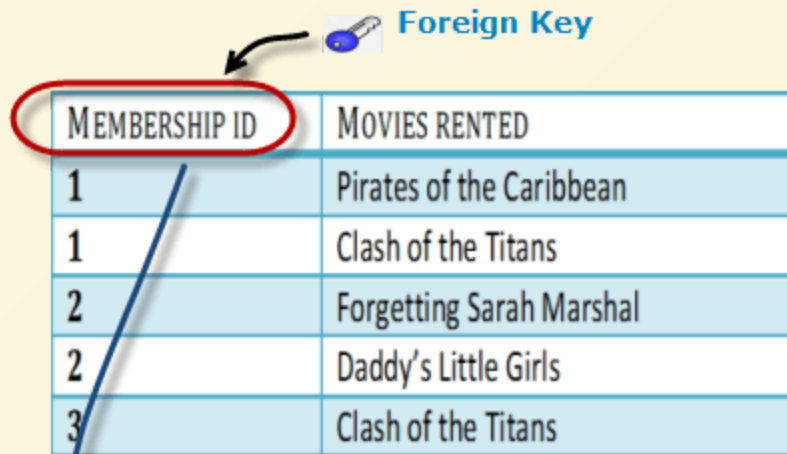
Unique and non-null number that refers to each record

To understand why, let's imagine that the user "william" wants to change his "username" to "bill." How do we know which "password" and "birthday" to associate with this user? Without a primary key, we don't.

FOREIGN KEY

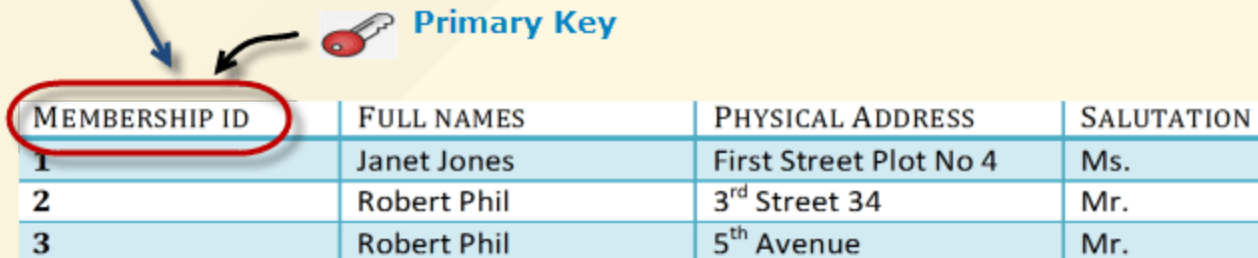
1. Foreign Key references the primary key of another Table! It helps connect your Tables
2. A foreign key can have a different name from its primary key
3. It ensures rows in one table have corresponding rows in another
4. Unlike the Primary key, they do not have to be unique. Most often they aren't

FOREIGN KEY



MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Foreign Key references Primary Key
Foreign Key can only have values present in primary key
It could have a name other than that of Primary Key



MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 rd Street 34	Mr.
3	Robert Phil	5 th Avenue	Mr.

DATABASE RELATIONSHIP

1. One-to-one
2. One-to-many
3. Many-to-many

One to One

Allows only one record on each side of the relationship.

The primary key relates to only one record – or none – in another table. For example, in a marriage, each spouse has only one other spouse. This kind of relationship can be implemented in a single table and therefore does not use a foreign key.

One to Many

Allows a single record in one table to be related to multiple records in another table.

Consider a business with a database that has Customers and Orders tables. A single customer can purchase multiple orders, but a single order could not be linked to multiple customers. Therefore the Orders table would contain a foreign key that matched the primary key of the Customers table, while the Customers table would have no foreign key pointing to the Orders table.

Many to Many

Complex relationship in which many records in a table can link to many records in another table.

For example, our business probably needs not only Customers and Orders tables, but likely also needs a Products table.

VIEWS

1. A view is a database object that is of a stored query.
2. A view can be accessed as a virtual table in PostgreSQL.
3. A view does not store data physically

Advantages

1. A view helps simplify the complexity of a query because you can query a view, which is based on a complex query, using a simple `SELECT` statement.
2. Like a table, you can grant permission to users through a view that contains specific data that the users are authorized to see.

Create a View

```
CREATE VIEW view_name AS query;
```

View Sample Query

```
CREATE VIEW customer_master AS SELECT cu.customer_id  
AS id, cu.first_name || ' ' || cu.last_name AS name,  
a.address, a.postal_code AS "zip code", a.phone,  
city.city, country.country, CASE WHEN cu.activebool  
THEN 'active' ELSE '' END AS notes, cu.store_id AS sid  
FROM customer cu INNER JOIN address a USING  
(address_id) INNER JOIN city USING (city_id) INNER  
JOIN country USING (country_id);
```


Alter View

```
ALTER VIEW customer_master RENAME TO customer_info;
```

Drop View

```
DROP VIEW IF EXISTS customer_info;
```

Materialized View

1. A view that allows to store data physically

Create Materialized View

```
CREATE MATERIALIZED VIEW view_name AS query WITH [NO]  
DATA;
```

Refresh Materialized View

```
REFRESH MATERIALIZED VIEW view_name;
```

When you refresh data for a materialized view, PostgreSQL locks the entire table therefore you cannot query data against it.

Refresh Materialized View Concurrently

```
REFRESH MATERIALIZED VIEW CONCURRENTLY view_name;
```

Concurrently

1. With `CONCURRENTLY` option, PostgreSQL creates a temporary updated version of the materialized view, compares two versions, and performs INSERT and UPDATE only the differences.
2. You can query against the materialized view while it is being updated.
3. One requirement for using `CONCURRENTLY` option is that the materialized view must have a `UNIQUE` index.

Create Unique Index

```
CREATE UNIQUE INDEX rental_category ON  
rental_by_category (category);
```


Drop Materialized View

```
DROP MATERIALIZED VIEW view_name;
```