**VALIDATION**

* Admin user might directly connect to the database such as admin.

**ROW LEVEL VALIDATION**

* Things we should check , when a row is inserted , updated.
* Is given value defined?
* Is a value unique in its column
* Is a value >,< ,>=,<= some other value.
* The term hero level, means that we are trying to validate a row , or the information inside a row whatever.it is inserted or updated in a table.

**STEPS TO APPLY NULL CONSTRAINT**

* Use the query tool.
* Right click on validation database and click on query tool.
* Write the SQL code.
* Click on Run and you will get code inserted.

**SOLVING A GOTCHA WITH NULL CONSTRAINTS**

* We want to have values inside the column , unless we have any specific reason.

**APPLYING A UNIQUE CONSTRAINT TO ONE COLUMN**

* We can temporarily rename this column right here.

**TABLES AND HASH-TAGS TOGETHER**

* The only purpose of it is somehow relate a hashtag and a post together.

**MULTI COLUMN UNIQUENESS:**

* When creating the table..

CREATE TABLE PRODUCTS(

Id SERIAL PRIMARY KEY,

name VARCHAR(50),

department VARCHAR(50),

price integer ,

weight integer,

UNIQUE(name , department)

);

* After the table was created….

ALTER TABLE products

ADD UNIQUE (name ,department);

**APPLYING VALIDATION**

* User -> Web Server -> data products.

|  |  |
| --- | --- |
| Web server | Database |
| Easier to express more complex validation | Validation still applied even if you connect with a different client. |
| Far easier to apply validation rules. | Guaranteed that validation is always applied. |
| Many libraries to handle validation automatically. | Can only apply new validation rules if all existing rows satisfy,. |

* Adding a new value to a database tends to be more tricky.
* You can connect from some other client.

# **MORE COMPLICATED DESIGNS**

* Challenging to keep the structure/names of many different tables on your head.
* Nice to document your DB structure somehow!
* We can use schema designer to guide our schema.

**SQL SCHEMA DESIGNERS**

* Dbdiagram.io
* drawsql.app
* sqldbm.com
* quickdatabasediagrams.com
* ondras.zarovi.cz/sql/demo

**SQL DESIGN TOOL**

* One user can have many different commits.
* We need to place foreign key on the commits , table that look back on the idea of a particular tool.

**ADDITIONAL FEATURES AROUND POSTS**

* All location is all about physical real world location with a photo.

**PHOTO MENTIONS VS CAPTION MENTIONS**

* May be one person can be mentioned many times inside the same post.
* The tags table in the case does not actually record what the tag was added actually inside the photo.
* Highlighted text does not necessarily mean that we need to store something in the database.
* Mobile app could probably in charge of anything that looks in mention.

**DESIGNING A HASH-TAG SYSTEM**

* Can search for posts that contain hast-tags -implies that hashtags in a post caption are modeled in a database.
* Cannot search for comments or users with a hashtag -implies that they are not modeled.

**POSTS AND FOLLOWERS**

* Posts= Can be calculated in a running query , or a query that exists in the database.
* Followers=We call this ‘derived data’ We generally do not want to store derived data.

**FOLLOWER SYSTEM**

* A following system sets up a relationship between one user and another.
* We can follow user only one time.
* We cannot follow ourselves.

**STEPS TO CREATE A NEW DB**

* Create a new DB using PGadmin.
* Convert our design into a series of CREATE TABLE statements.
* Insert data into the database.
* Write some queries,

**CREATING TABLE WITH CHECKS**

* Not null = A value must be provided.
* Default=provide a default value if an insert statement just gives one.

**POSTS CREATION**

* 100% want an user to provide a value.
* We always want a value, but should be optional.