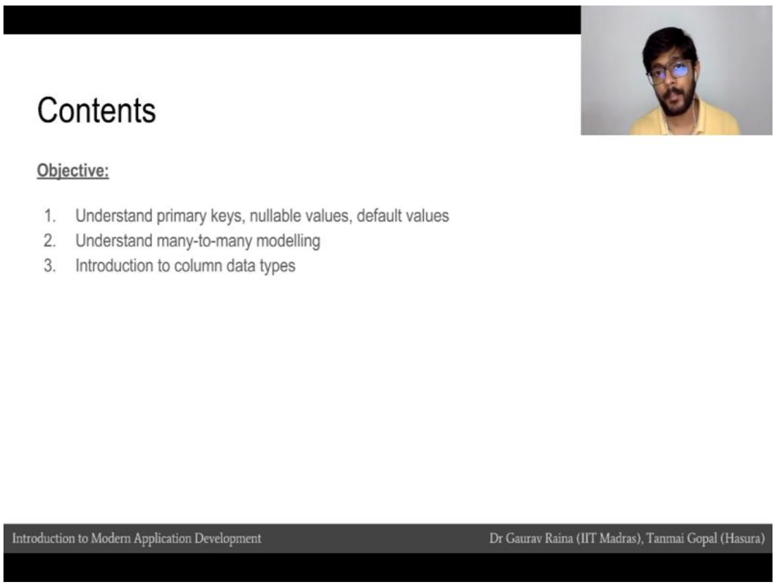
# Introduction to Modern Application Development

**Prof. Tanmai Gopal**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Madras**

**Module - P8**
**Lecture - 16**
**Practical: Deeper exploration of a DBMS (column types and more)**

Hi everybody. In this module, we will be looking at deeper exploration of DBMS. We will be looking primarily at different kinds of column types and slightly more advanced modelling.

(Refer Slide Time: 00:11)



Contents

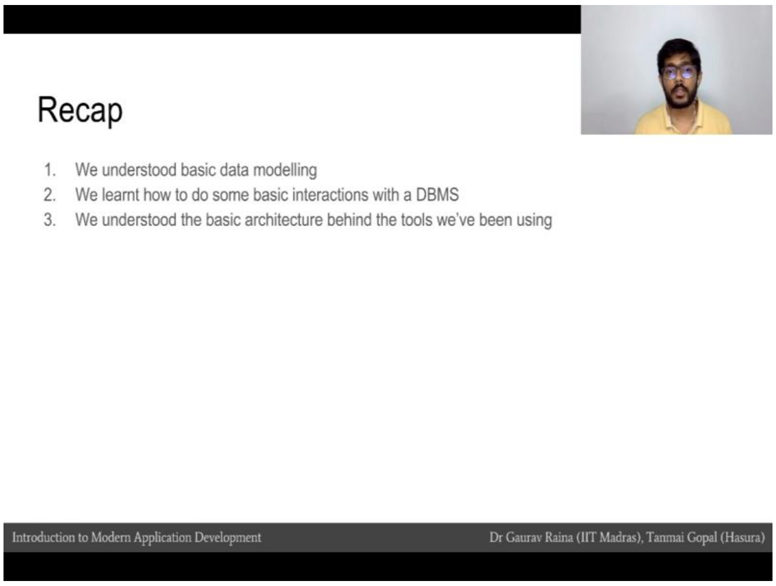
Objective:

1. Understand primary keys, nullable values, default values
2. Understand many-to-many modelling
3. Introduction to column data types

Introduction to Modern Application Development — Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

So, our primary objective in this quick practical module is to get little deeper in to understanding different column types. We will also try to understand primary keys in little more detail about what a primary key is. Some kind of heuristics or common ways of selecting what a good primary key for a table is. We will also look at kind of modelling known as many-to-many modelling, which is very common and very useful when modelling for relational data bases.

(Refer Slide Time: 00:35)



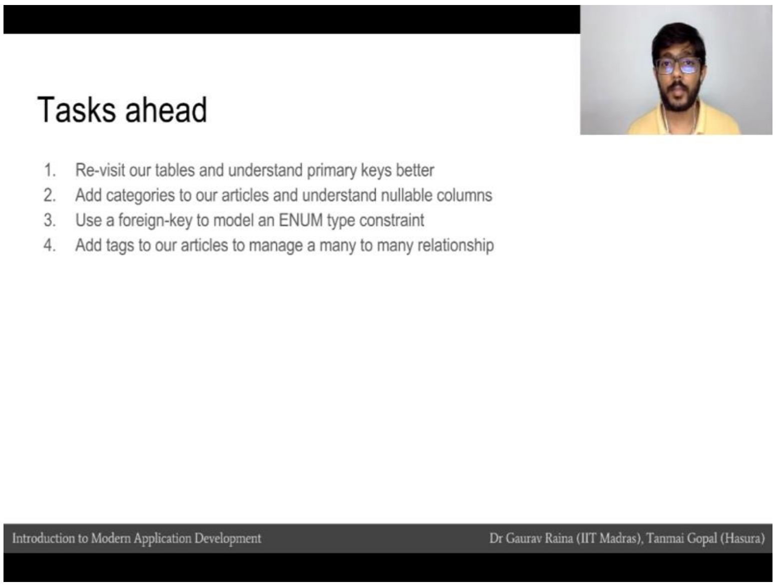
## Recap

1. We understood basic data modelling
2. We learnt how to do some basic interactions with a DBMS
3. We understood the basic architecture behind the tools we've been using

Introduction to Modern Application Development Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

Just to take a quick recap, we have understood basic data modeling. And, in the last practical module we looked at doing some basic interactions with DBMS. We have also understood what the basic architecture behind the tools we have being, has using so far.

(Refer Slide Time: 00:47)



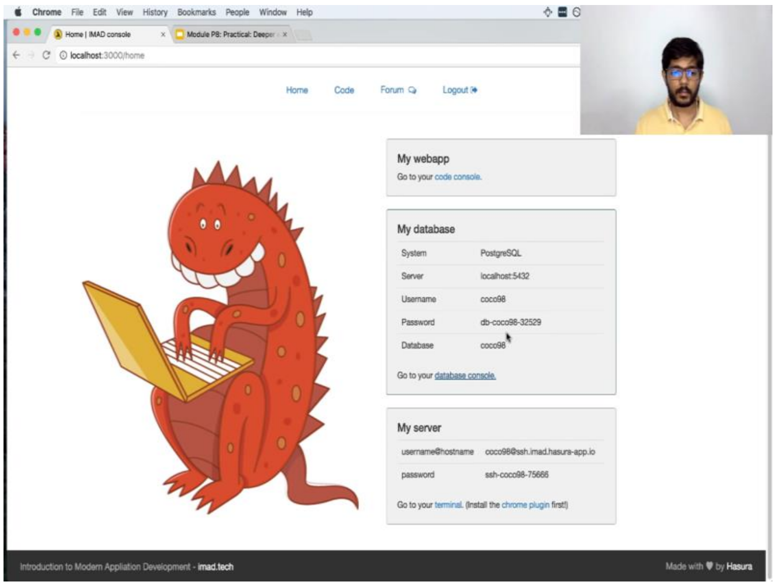
## Tasks ahead

1. Re-visit our tables and understand primary keys better
2. Add categories to our articles and understand nullable columns
3. Use a foreign-key to model an ENUM type constraint
4. Add tags to our articles to manage a many to many relationship

Introduction to Modern Application Development Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

In this module, the exercise that we will be doing is extending our blog example from one of the previous modules. We had the user table and the article table. So, we will add categories and tags to it. Until that, we will understand the different kinds of relationships that we can model in the database.

(Refer Slide Time: 01:01)


Chrome File Edit View History Bookmarks People Window Help
Home | IMAD console
Module P8: Practical: Deeper
localhost:3000/home
Home Code Forum Logout

My webapp

Go to your code console.

My database

| | |
|---|---|
| System | PostgreSQL |
| Server | localhost:5432 |
| Username | coco98 |
| Password | db-coco98-32529 |
| Database | coco98 |

Go to your database console.

My server

| | |
|---|---|
| username@hostname | coco98@ssh.imad.hasura-app.io |
| password | ssh-coco98-75666 |

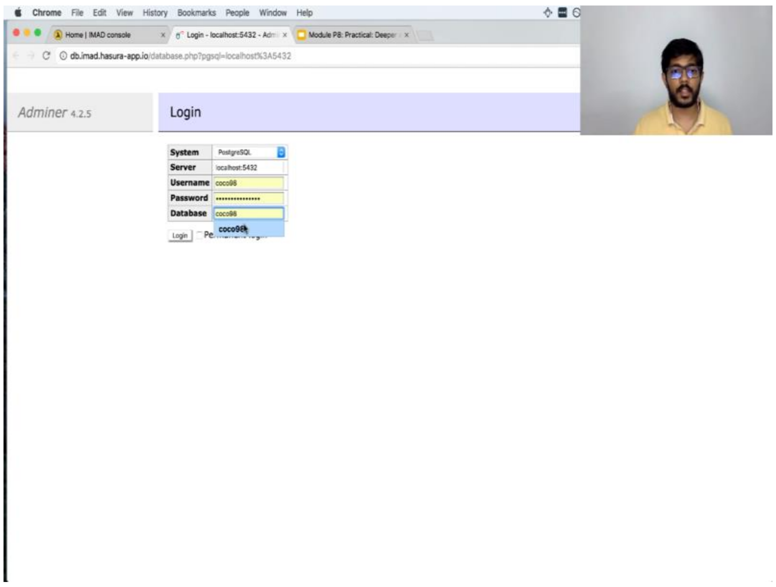
Go to your terminal. (Install the chrome plugin first!)

Introduction to Modern Appliation Development - imad.tech

Made with ♥ by Hasura

So, once again I click on this link.

(Refer Slide Time: 01:05)


Chrome File Edit View History Bookmarks People Window Help
Home | IMAD console
Login - localhost:5432 - Adm
Module P8: Practical: Deeper
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432
Adminer 4.2.5

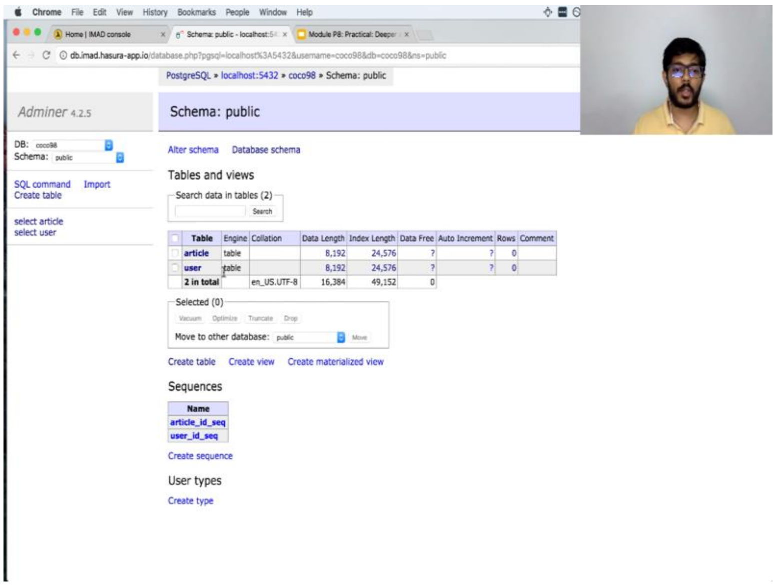
Login

| | |
|---|---|
| System | PostgreSQL |
| Server | localhost:5432 |
| Username | coco98 |
| Password | •••••••••••••••• |
| Database | coco98 |

Login Permanent login

And I am going to enter my database name as (Refer Time: 01:10) which is same as my username.

(Refer Slide Time: 01:12)


PostgreSQL » localhost:5432 » coco98 » Schema: public

Adminer 4.2.5

DB: coco98
Schema: public

SQL command Import
Create table

select article
select user

## Schema: public

Alter schema Database schema

Tables and views

Search data in tables (2)

| | Table | Engine | Collation | Data Length | Index Length | Data Free | Auto Increment | Rows | Comment |
|---|---|---|---|---|---|---|---|---|---|
| | **article** | table | | 8,192 | 24,576 | ? | ? | 0 | |
| | **user** | table | | 8,192 | 24,576 | ? | ? | 0 | |
| | **2 in total** | | en_US.UTF-8 | 16,384 | 49,152 | 0 | | | |

Selected (0)

Move to other database: public

Create table Create view Create materialized view

Sequences

| Name |
|---|
| article_id_seq |
| user_id_seq |

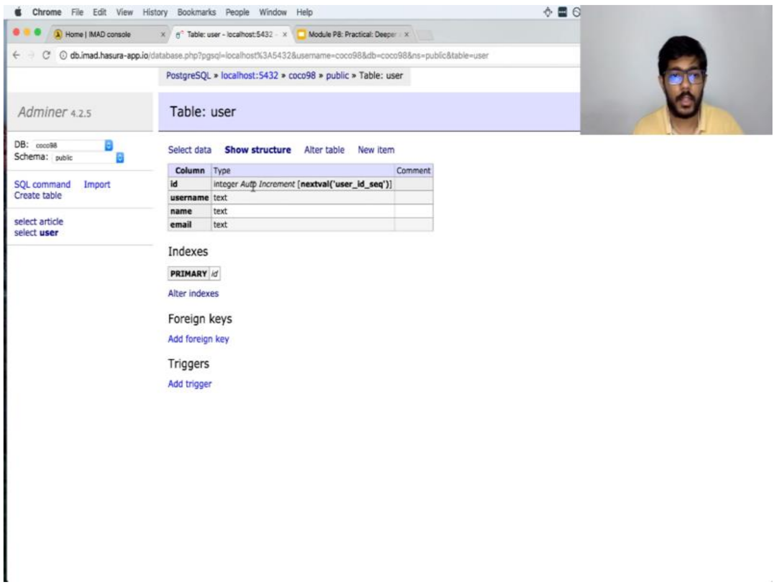
Create sequence

User types

Create type

And, I have logged in. So, the first thing in that we want to do is try to understand the notion of a primary key a little better.

(Refer Slide Time: 01:21)


PostgreSQL » localhost:5432 » coco98 » public » Table: user

Adminer 4.2.5

DB: coco98
Schema: public

SQL command Import
Create table

select article
select **user**

## Table: user

Select data **Show structure** Alter table New item

| Column | Type | Comment |
|---|---|---|
| **id** | integer *Auto Increment* [**nextval('user_id_seq')**] | |
| **username** | text | |
| **name** | text | |
| **email** | text | |

Indexes

| **PRIMARY** | *id* |
|---|---|

Alter indexes

Foreign keys

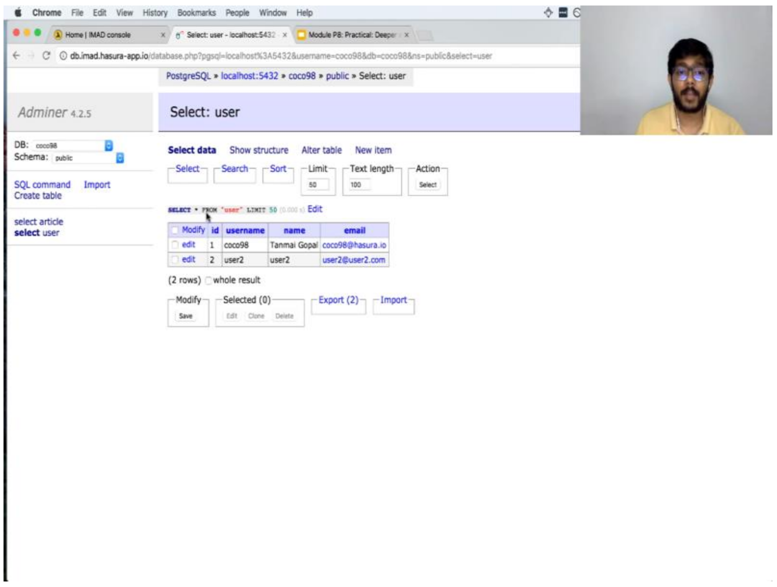
Add foreign key

Triggers

Add trigger

Let us look at the user table. Now, if you look at the user table we had the id column, the username column, the name column and the email column. And, we added the primary key. Now, what is a primary key?

(Refer Slide Time: 01:36)



PostgreSQL » localhost:5432 » coco98 » public » Select: user

Select: user

Limit: 50 Text length: 100

```
SELECT * FROM "user" LIMIT 50 (0.000 s) Edit
```

| Modify | id | username | name | email |
|---|---|---|---|---|
| edit | 1 | coco98 | Tanmai Gopal | coco98@hasura.io |
| edit | 2 | user2 | user2 | user2@user2.com |

(2 rows) whole result

A primary key is something that uniquely identifies a row in a table. And, not just that. But, a primary key is also a kind of index. And, we will cover what indexes are little later. But, for now it is important to understand that indexes help us to find rows faster. If you search by an index in a particular table; indexes, speed up access to particular rows in a table. For example, if we know that we are going to making queries very often using the id value. For example, we are going to say fetch me the details of user id 5, fetch me the details of user id 10. If we know that we are going to make these queries very often that column is a good candidate for being an index.

Now, when a column is a good candidate for being an index and the column is also good candidate for being a unique identifier for the row. That is typically when you should make it a primary key. Note that in this case even the user name could have been thought be the primary key or even the email could have been a primary key; because different users will probably not have the same email or user names or we might not want to allow different users in our data module to have this same user name or the same email.

In this case both username and email can also act as primary keys. However, the reason why I did not select them as a primary key is because the values of username and email might change. And if the value of the username and email are changing quite frequently, again I would prefer to have the primary key inside the separate column, whether the value of that column for that row does not change very often.

Which is why id is a primary key, and how do I ensure that id has a unique value every single time? It is because if we look at the structure of the table, I made it an auto incrementing value; that means that even if I do not supply the value of id when I am inserting a new row, the value of id will automatically be incremented from whatever the previous value of id is and inserted in to the row. So, which is why if I insert a new row with the new username, name and email, it will automatically have the id three.

(Refer Slide Time: 03:31)

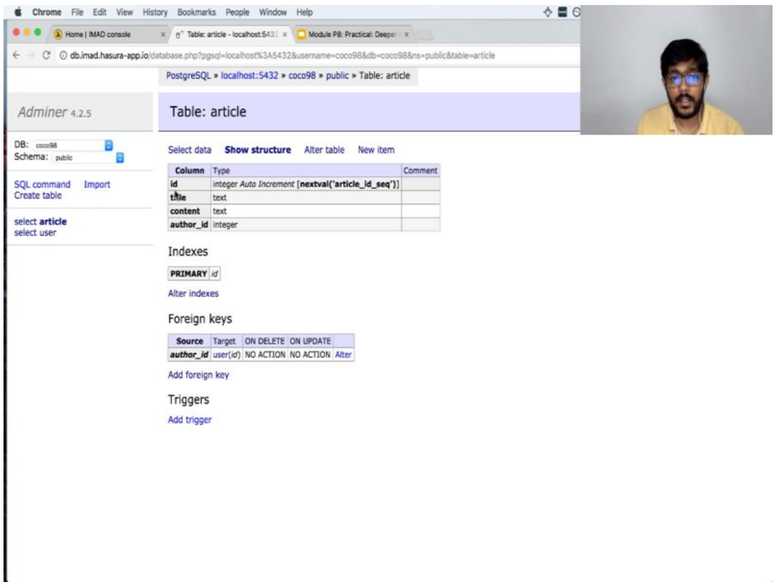

Chrome File Edit View History Bookmarks People Window Help
Home | IMAD console
Table: article - localhost:5432
Module P8: Practical: Deeper
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&table=article
PostgreSQL » localhost:5432 » coco98 » public » Table: article
Adminer 4.2.5
DB: coco98
Schema: public
SQL command
Import
Create table
select article
select user
Select data
Show structure
Alter table
New item
Alter indexes
Add foreign key
Add trigger

Table: article

| Column | Type | Comment |
|---|---|---|
| id | integer *Auto Increment* [**nextval('article_id_seq')**] | |
| title | text | |
| content | text | |
| author_id | integer | |

Indexes

| | |
|---|---|
| PRIMARY | *id* |

Foreign keys

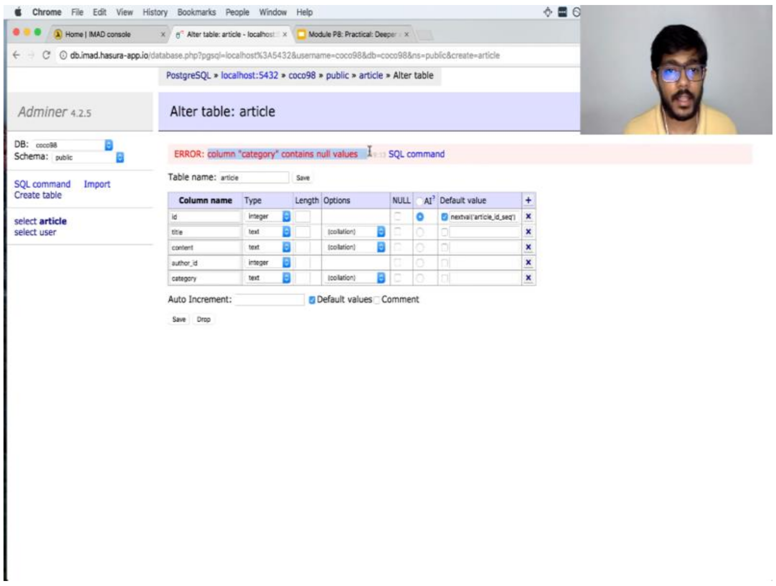
| Source | Target | ON DELETE | ON UPDATE | |
|---|---|---|---|---|
| *author_id* | user(*id*) | NO ACTION | NO ACTION | Alter |

Triggers

Let us go the article table. So, if you look at the article table, again the primary key I have selected to be id because title, content or author id, none of them uniquely identify the row as easily as id which uniquely identifies the row. Another; note that primary keys do not need to consist of a single column. A primary key could also consist of multiple columns.

For example, I could also make title, author id, a primary key; that because may be the same author. May be, in my data module I do not want to allow same author to have multiple articles of the same title. Title, author id is a unique value. And, may be I am going to use that value to uniquely identify my row, in which case I would make that a primary key. Our first exercise is to try to add the category column to our table. So, let us add a new column.

(Refer Slide Time: 04:25)


Alter table: article

ERROR: column "category" contains null values SQL command

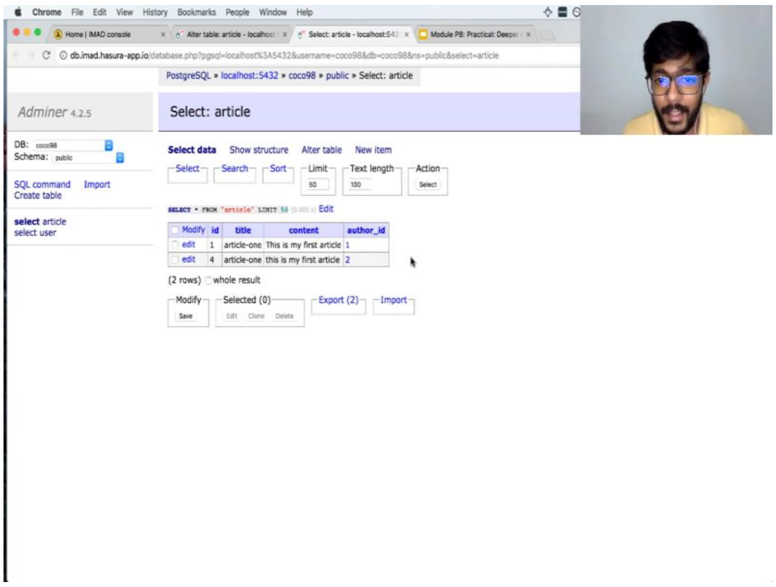
Table name: article

| Column name | Type | Length | Options | NULL | AI | Default value |
|---|---|---|---|---|---|---|
| id | integer | | | | ● | ✓ nextval('article_id_seq') |
| title | text | | (collation) | | | |
| content | text | | (collation) | | | |
| author_id | integer | | | | | |
| category | text | | (collation) | | | |

Auto Increment: ✓ Default values Comment

So, what I am going to do is alter table. I click on the plus button here to add a new column. And, I am going to say this column is called category and this is a text column. We can see many different types of columns that are provided by (Refer Time: 04:40). So, I will say text and I am going to add this. So, let us see what happens.

(Refer Slide Time: 04:52)


Select: article

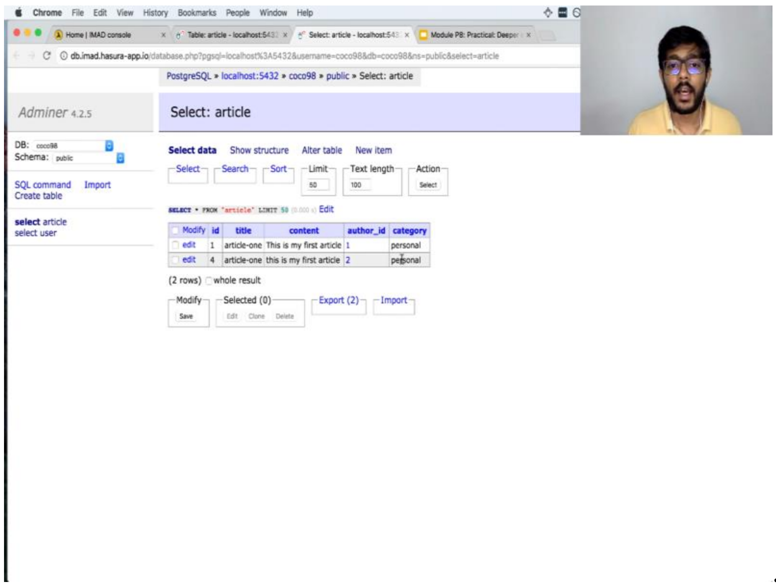
```
SELECT * FROM "article" LIMIT 50
```

| Modify | id | title | content | author_id |
|---|---|---|---|---|
| edit | 1 | article-one | This is my first article | 1 |
| edit | 4 | article-one | this is my first article | 2 |

(2 rows)

As soon as I clicked on save, I got an error saying column category contains null values. And, this is because I already have two articles. And, if I try to add a category column, the category column would not have a value. This column has no value. In a database, that is typically referred to as a null value; because the value is not an empty string. In this case the value is no value, which is also called as null.

(Refer Slide Time: 05:09)


Chrome File Edit View History Bookmarks People Window Help
Home | IMAD console
Table: article - localhost:5432
Select: article - localhost:5432
Module P8: Practical: Deeper
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&select=article
PostgreSQL » localhost:5432 » coco98 » public » Select: article
Adminer 4.2.5
DB: coco98
Schema: public
SQL command
Import
Create table
select article
select user
Select data
Show structure
Alter table
New item
Select
Search
Sort
Limit
Text length
Action
Select
Modify
Save
Selected (0)
Edit
Clone
Delete
Export (2)
Import

Select: article

Limit: 50 Text length: 100

```
SELECT * FROM "article" LIMIT 50
```

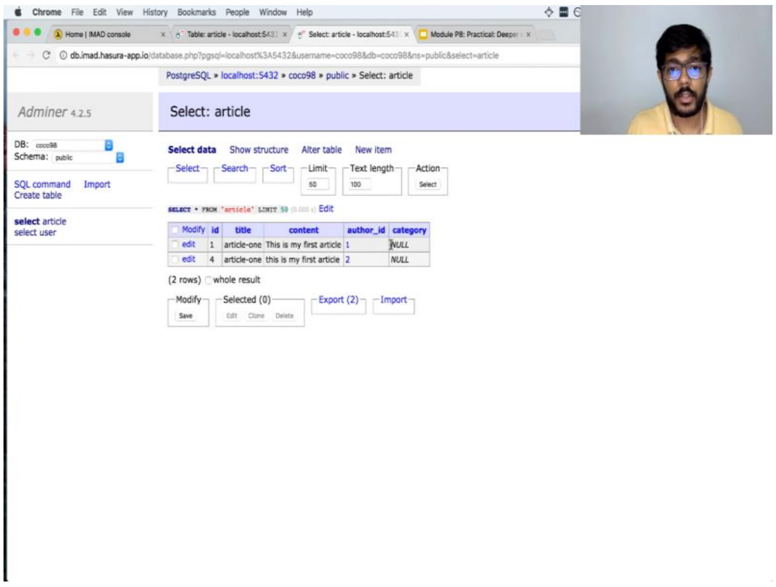
| Modify | id | title | content | author_id | category |
|---|---|---|---|---|---|
| edit | 1 | article-one | This is my first article | 1 | personal |
| edit | 4 | article-one | this is my first article | 2 | personal |

(2 rows) whole result

If we are adding a new column, we should allow, we should figure out what we want to do with the existing rows, which do not have that column. For example, do we want to give it a default category or do we want to give it? Do we want to allow null values? Let us say want to give it a default value.

In that case I go to the default value column and say that the category is personal. If I click on save, you will see the save goes through and refresh this page. And, you will see that a default category has been given. Suppose, I did not want to give the default category and I wanted to allow the category value to be null, to not have a value. Let us delete this column again. So, I go back to alter table. I click on the cross arrow and deletes the column. I save it. And, you can see that the category column is disappeared.

(Refer Slide Time: 06:02)



Select: article

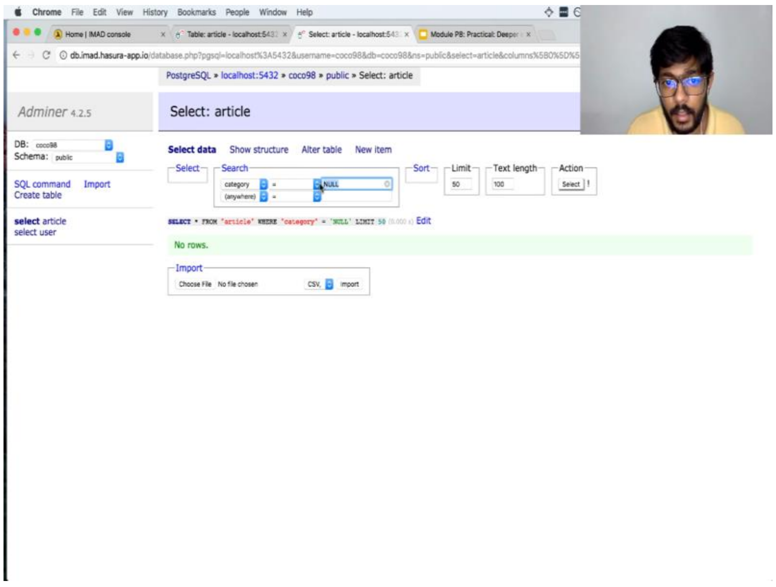
```
SELECT * FROM "article" LIMIT 50 (0.000 s) Edit
```

| Modify | id | title | content | author_id | category |
|---|---|---|---|---|---|
| edit | 1 | article-one | This is my first article | 1 | NULL |
| edit | 4 | article-one | this is my first article | 2 | NULL |

(2 rows)

Let us try to add the same column again. Sorry, let us click on the plus symbol category. I will make this text. And because now we are not going to specify default value, we want to allow the category column to contain null values. So, I am going to click on this column called null. And, if I save it and I refresh this page, you will see that this value is a value null. Now, it is important to understand that this is not a string value null.

(Refer Slide Time: 06:35)



Select: article

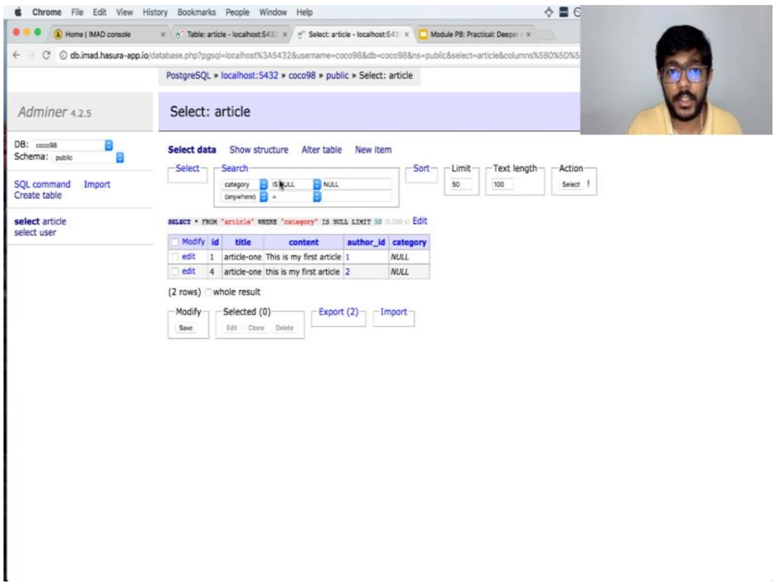
Search: category = NULL

```
SELECT * FROM "article" WHERE "category" = 'NULL' LIMIT 50 (0.000 s) Edit
```

No rows.

For example, if I try to search for category equal to null, and I say select, nothing comes up; because this is not a string called null. But, if I search for category, is null.

(Refer Slide Time: 06:49)


PostgreSQL » localhost:5432 » coco98 » public » Select: article
Select: article
Select data
Show structure
Alter table
New item

Search: category IS NULL NULL

Limit: 50 Text length: 100

```
SELECT * FROM "article" WHERE "category" IS NULL LIMIT 50
```

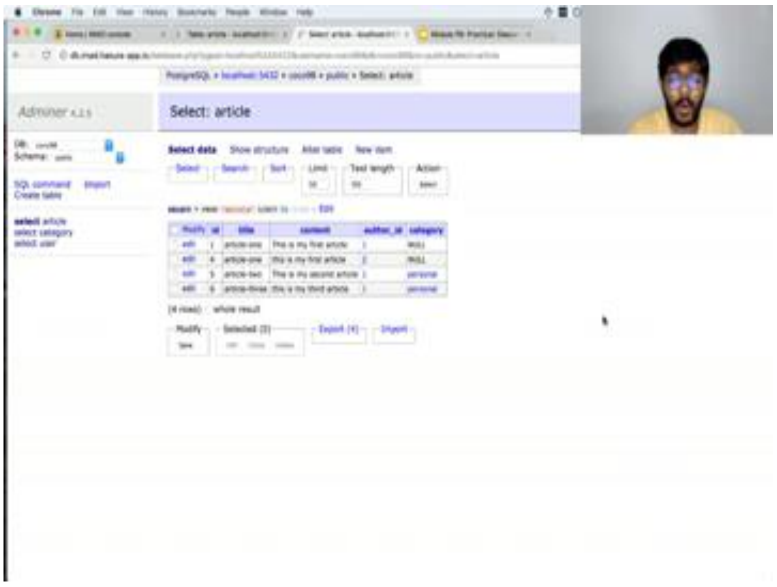
| Modify | id | title | content | author_id | category |
|---|---|---|---|---|---|
| edit | 1 | article-one | This is my first article | 1 | NULL |
| edit | 4 | article-one | this is my first article | 2 | NULL |

(2 rows) whole result

You can see that the null comes up, because you cannot compare a value to null. You cannot say that category is equal to null because null is not a value. So, the database provides you the special way for searching for a particular article that contains a value if it is null, which is called is null. It is not equal to. You cannot use brace, you cannot use greater than or lesser than, but you have to say is null for.

Similarly, you can say is not null. So if you search for is not null, you see no rows because both the columns have null set as the category. So, this is how we added the category.

(Refer Slide Time: 07:30)



Select: article

| Modify | id | title | content | author_id | category |
|---|---|---|---|---|---|
| edit | 1 | article-one | This is my first article | 1 | NULL |
| edit | 4 | article-one | this is my first article | 2 | NULL |
| edit | 5 | article-two | This is my second article | 1 | personal |
| edit | 6 | article-three | this is my third article | 1 | personal |

(4 rows) · whole result

Now what? The problem with this is that if I add a new item, like let say I say article two; content is, “this is my second article”. Author id 1; and let say I insert the category called personal. And, I saved this. So, you can see that a category called personal has been added. Now, if I add new item and let say title article 3; content, “this is my third article” by the same author ‘author 1’. And, let say I misspell personal.

Let say I misspell personal and I save it. You will notice that a different value of personal is getting saved. So, suppose I want to restrict the value of category to come from the predefined list, how would I do that? What I would do is I will use the foreign key constraint and I create another table. I will call this table ‘category’. And, this table will have only one column called name, which would be a text column.

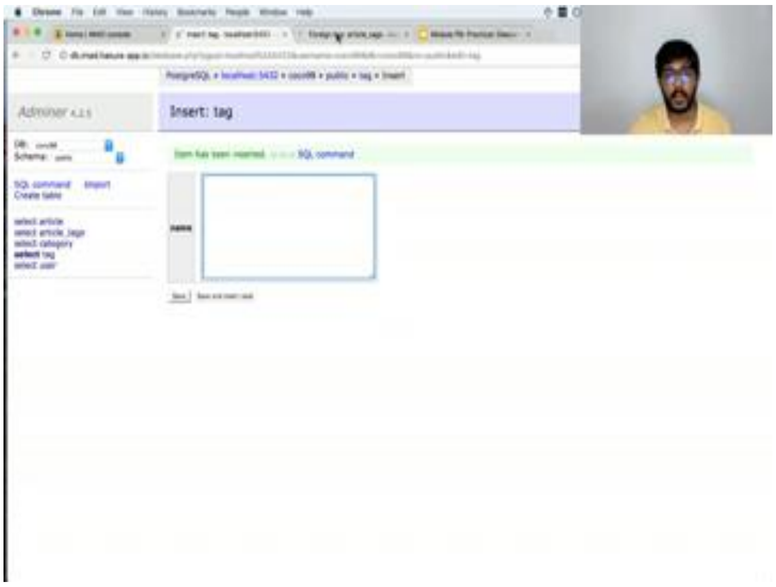
And, so let us save this. And, I will make this column, the primary key. So, the name column is the primary key. And, what I will do is I will insert some values here. So, I will say personal, work, technology.

Let us say these are the different categories. If I click on select, I will see the three particular categories have been added. Now, let us go back to our article table. And, let us restrict the value of category to only come from this category table’s name. So, if you would show structure, let us click on add foreign key.

Let us have the target table to category and say that our table's category values should be the target table's name. And, let us click on save. As soon as I click on save, we get this error which says the category personal, persona, personally, which is the misspelt personal is not present in the table category because the value is inconsistent. If you remember we did this last time. Let us go back to our article table and let us edit this and make this value null. Let us change it to null. So, I changed that value and let us go back and add the foreign key. And, so you can see the foreign key has been added.

Now, if I go back to the article table and I insert the value here, personal, it goes through. And, obviously if I try to change this to something else, I get an error saying that this key is not present in the table 'category'. And, so this how we are able to restrict our values of the category to come from a predefined set. This is very similar in programming languages to defining an ENUM type. For our next exercise, let us try to add tags to our article. So now unlike category, every article can have only category, but we want to allow every article to have multiple tags.

(Refer Slide Time: 10:42)



So, how should we model this? As we discussed in one of the previous modules, we could have added a tags column, but that would have made it hard to store multiple tags. The way to do it would be to create a new table called 'article_tags', where we give it a column called 'article id', which is an integer; because this refers to the article from our article table. And then, we will give it a column called tag, which is text. And, save this. Now, if you think about what the primary key for this particular table will be.

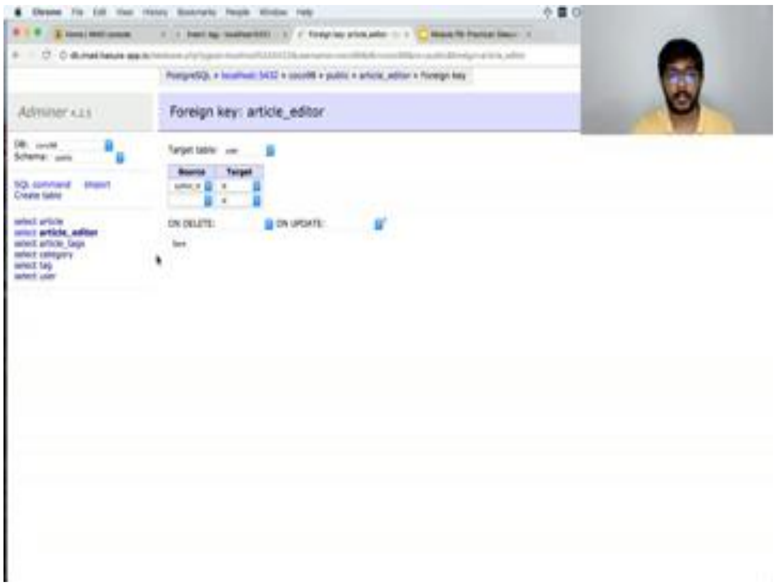
You notice that the primary keys actually not article id because if the primary key was article id, then we would not be able to add multiple tags. For example, ‘article id 1’ has tag one and tag two. There will be two rows in this table. And, so article id cannot be a primary key. Similarly, tag cannot be a primary key.

If the primary key will necessarily have to be both the columns, so article id and tag both together will define the primary key. Now again before we go ahead and insert some items, we have to remember to restrict these values. We do not want the random value of article id to be allowed inside this table; because this article id should definitely refer to an article that exist in the article table. And, the way of doing that is by adding foreign key. Let us have the foreign key. Let say the target table as article and article id must be the article id. Let us save this.

Now let us insert an item saying article id 1 contains the tag ‘mobile’. Article id 1 also contains the tag ‘web’. Now we can see that the article has the two tags. The article id 1 has tag ‘mobile’ and has tag ‘web’. The problem again is the same. What if I misspell a tag? What if I misspell the tag? And, I want tags to come from a predefined list of tags. Very simple. We will create a new table called ‘tag’; make it have a only one column called ‘name’ and save this. Let us make this name column, the primary key. So, that is done as well.

Let us go back to our article tags table. And, now we want to restrict the value of tag. So, let us go structure, add a foreign key. And, let us make the tag column point to the tag table’s name column. And, save this. As soon as I did this, you can see that are got an error saying the key tag ‘mobile’ is not present in the table ‘tag’. So, what we can do is I go to another tag. I am going to refresh this tag and go to the tag table. And, let me quickly insert two tags.

(Refer Slide Time: 13:38)


PostgreSQL » localhost:5432 » coco98 » public » article_editor » Foreign key

Adminer 4.2.5

## Foreign key: article_editor

DB: coco98
Schema: public

SQL command Import
Create table

select article
select **article_editor**
select article_tags
select category
select tag
select user

Target table: user

| Source | Target |
|---|---|
| author_id | id |
| | id |

ON DELETE: ON UPDATE:

Save

So, we had ‘web’ and we had ‘mobile’. And, now let us go back here and try to add this foreign key again. You can see that the foreign key goes through because this data is not present in the tag table. So, this is how we were able to model tags. And, we were able to model multiple tags in our articles. Now if we look at the article data, you have article data that is in this table that the category is also in this table. And, you also have the tag data for this article, which is in a separate table called article_tags.

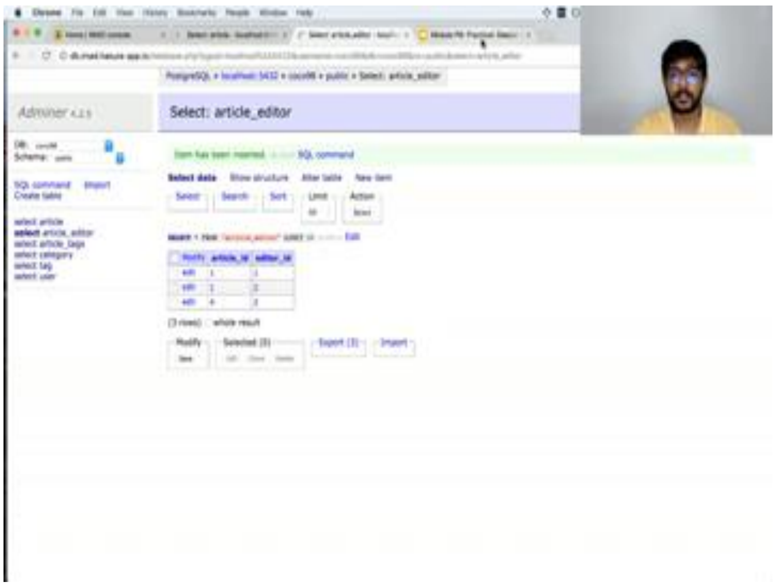
Now, this style of modelling is also referred to as a many-to-many modelling because many articles can have one or more tags. So, and tag itself comes from a separate table and the article itself comes from a separate table. And, we have this kind of table in the middle. This is the common way of modelling many-to-many relationships.

For example, if we had an article, every article had multiple authors. We obviously would not be able to have an ‘author id’ column because now a single article has multiple authors. So, what we would do? In that case, as you would create a new table called ‘article_author’. And at, which you would have two columns called ‘article id’ and ‘author id’, which would allow a single article to have multiple authors; ‘article id 1’ can have ‘author id 1’, ‘article id 1’ can also have ‘author id 2’ and so on and so forth.

In fact, let us do a quick modelling. Just to give an example, let us say our article has many editors. And, each editor has to be a user. So, what I can do is I can quickly create a table called ‘article underscore editor’, which will have two columns called ‘article id’ and ‘author id’. And, obviously the primary key is both article id and author id. It is not just one of the columns.

In addition, I would make article id, the foreign key to the article table and the author id, a foreign key to the author to the user table. So, now this becomes a way for me to model editors. So as you can see, I made a mistake here.

(Refer Slide Time: 15:56)


Adminer 4.2.5

Select: article_editor

Item has been inserted. SQL command

Select data  Show structure  Alter table  New item

| Modify | article_id | editor_id |
|---|---|---|
| edit | 1 | 1 |
| edit | 1 | 2 |
| edit | 4 | 2 |

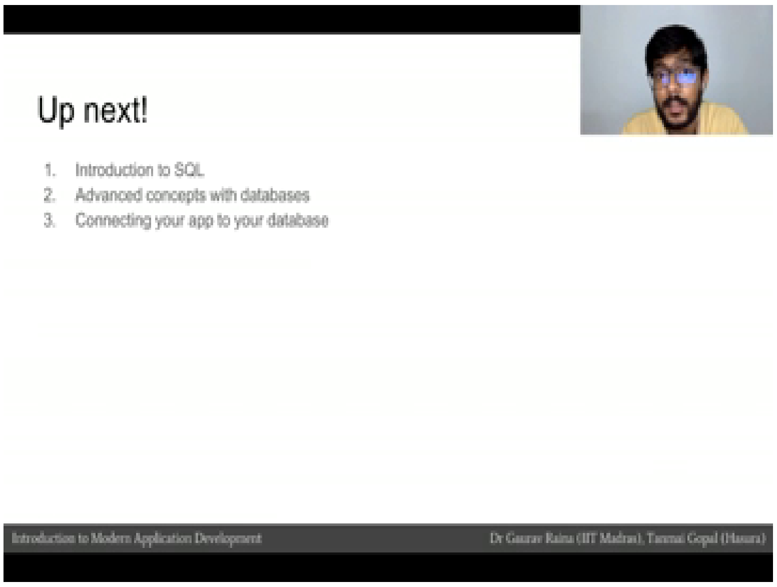
(3 rows) whole result

I am trying to model editors. And, I have named this column ‘author id’. Obviously, this column should be an ‘editor id’. So, I can click on the alter table, change the name to ‘editor id’. And, this goes through. Now I have renamed this column into an editor id.

Now this table can model multiple editors. Let us try to quickly insert some items. So, I can say ‘article id 1’ has ‘editor one’; ‘article id 1’ has ‘editor two’. If you look at this article, there are two editors for the same article. I can accept more items. I can say ‘article id 2’ as ‘editor 2’. So, this is a foreign key violation because we do not have ‘article id 2’, which we probably deleted a long time ago.

Let us see what articles we have. We have 4, 5 and 6. So, let me change this to 4. So, this means that article 1 has two editors, which is user 1 and user 2 are both editors and article id 4 as only one editor, which is user 2.

(Refer Slide Time: 17:19)



## Up next!

1. Introduction to SQL
2. Advanced concepts with databases
3. Connecting your app to your database

Introduction to Modern Application Development    Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

This is the way I can model what is called a many-to-many relationship because each article can have many editors. And, each editor can be an editor for the multiple articles. So, this is called a many-to-many relationship.

So, in this quick module we looked at slightly more advanced modelling. We looked at different kinds of column data types and we looked at how to model ENUM kinds of relationships and we looked at how to model many-to-many relationships.

In the next few modules, we will be looking, we will be getting an introduction to SQL and the theoretical modules. We look at slightly more advanced concepts with databases. And, the last practical module for the database's week will be connecting our app to the database.