2. Introduction to Relational Database and SQL

Contents

[1. Introduction 1](#_Toc5739)

[2. Relational Databases 1](#_Toc7467)

[3. Demonstration - Relational Database 1](#_Toc28004)

[4. Summary 1](#_Toc7425)

# 1. Introduction

=>slides: Pg. 1-2

So we'll start off by doing an introduction to relational databases and SQL. We'll then introduce you to JDBC. And then we'll take a look at various aspects of JDBC, such as connecting to a database, using PreparedStatements, working with data from a PreparedStatement, and finally, working with a CallableStatement.

=>slides: Pg. 3

So in this module, we'll take a look at what a relational database is, and we'll also take a look at the basics of CRUD operations, so create, read, update, and delete operations.

# Relational Databases

=>slides: Pg. 4

So in broad terms, we can think that there are two categories of database, SQL database or relational database, and that's what we'll take a look at here. And then there's also something called a NoSQL database. NoSQL databases contain data, but that data isn't modeled using relationships. It's often documents and often held in JSON format that you can query in some way.

=>slides: Pg. 5

So what is a relational database? Well, in a relational database, data is organized into tables, and those tables have rows and columns. And then generally, we relate those tables through keys, hence the term relational databases. So typically, tables may have primary keys, and those tables are related somehow through something called a foreign key, and we'll see that in a sec.

=>slides: Pg. 6

So the database we'll be using through this course is going to hold gigs. So a gig will be held at a venue, and the venue will have a name, and an act will perform that gig. So here we have three tables to represent that. We have the Acts table, and each act has a name. And notice each act in the table has an id as well, and that id is a unique key, and that key in this diagram is shown with a little key symbol on the table. We have venues. And again, every venue has a unique key. That's the id field. And venues have a name, and venues have a capacity. So how many people can attend this gig? And then we have the Gigs table, and it's the Gigs table that relates these two things. So every gig will be held at a venue, and that's the venue id field. That's a foreign key identified on this diagram with a little blue key symbol. Every gig must have an act, so an act will perform the gig. And again, that's represented by that little blue symbol in the diagram. And then for a given gig, we need to know how many tickets we've sold and the date of the gig. Now obviously, a real‑life example would be much more complex than this, but we're keeping it simple here just for the sake of the course.

=>slides: Pg. 7

So how do we get at that data in these tables? So any data here is accessed through SQL, the structured query language. For the mathematically inclined, this is a set‑based language.

=>slides: Pg. 8

And typically when we talk about SQL, we think in terms of four operations, also known as the CRUD operations, create, read, update, and delete. So for create, this is when we're going to insert new data into the table, so we're creating data. Read is obviously when we read data from the table. Update is when we want to update data in the table. And delete is when we want to delete data from the table. I'm saying table here, but it could be tables. So we could be deleting, reading, creating, and updating data in multiple tables using these operations. And we also think in terms of rows. So when I create data, I'm inserting rows into a table. When I read data, I'm reading rows from a table. And we've said that data in a table is rows and columns. When we do read, we read specific rows. We also read specific columns. We don't always read all the columns from a given table or tables. So how do we do this?

=>slides: Pg. 9

So we have some examples here of reading data. So we have a simple read where call SELECT on the Gigs table, and we're asking for all the TicketsSold rows in the table. We can also filter. So here I'm asking for the id and the name from the Venues table, but only where a name has a specific form. So here we're saying get this data from the Venues table when the name looks like arena. And the percent signs here a sort of like a regular expression. We're saying any name that's like arena with any text before it and any text after it. We can also join on other tables. So here we can select multiple data from multiple tables. So here I'm saying take the Gigs table and join that to the Venues table and match the id in the Venues table to the venue id in the Gigs table. Also, we can filter at the same time. So again, we'll just get the rows where the venues names are like arena. And then finally, we can also use something called SQL functions. So here I'm saying tell me how many rows are in the Gigs table where I have fewer than 30 tickets sold.

=>slides: Pg. 10

So as well as querying, we can insert data into tables, and we do that by using the INSERT INTO clause. When we do this, we specify the name of the table and the columns to which we'd like to insert the data, and then we specify the values. So here we give it the name of the venue, in this case the arena, and also we give it the capacity of the venue. So we're adding a new row into our table, and we're specifying the value of two columns in that row.

=>slides: Pg. 11

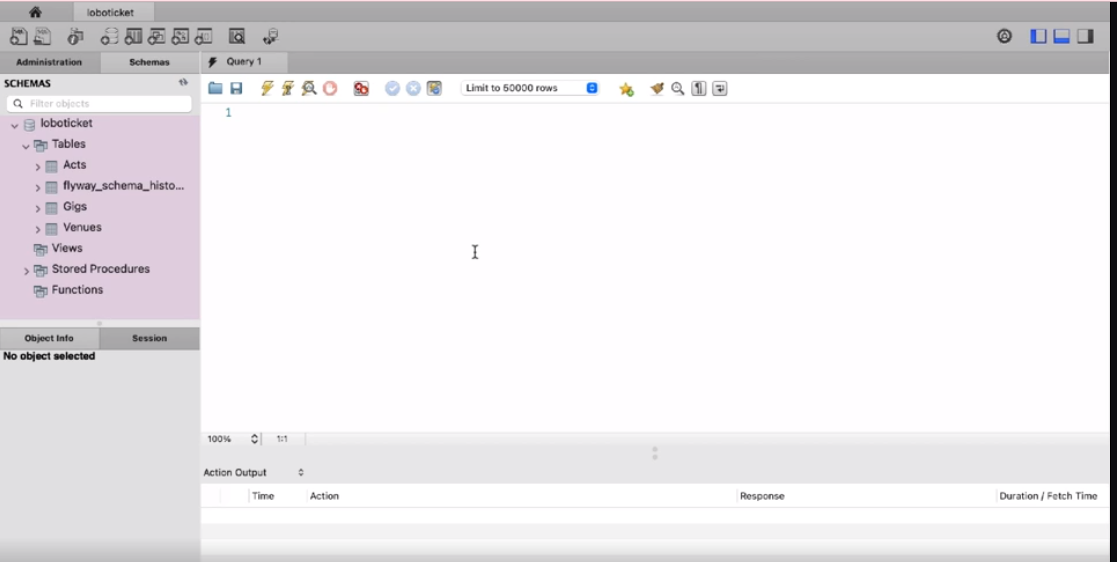
We can also update data. So to do that, we use the update keyword, and we specify the name of the table, in this case the Venues table. We specify the columns we want to update. So in this case, we're updating one column, the capacity column, and we're setting this value the 30. And then we tell it which column to change. So here we're saying anywhere in this Venues table where we have a venue row with an id of 4, set its capacity to 30. Now in this case, the id field is a primary key field, and primary key fields have unique values. So we should only change one row here. Beware of doing something like this. So here we're saying UPDATE venues SET capacity=30. So what this says is go to the Venues table and update the capacity for all rows in that table to the value of 30.

=>slides: Pg. 12

Similarly for delete. So we're saying DELETE FROM. We specify the name of the table. And again, we specify some filter. And here again, we are filtering on the primary key. So we should just delete one row from the table. The filters don't have to be on the primary key. Again, be aware of something like this. So DELETE FROM venues here will end up deleting all the rows from that table.

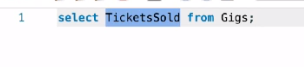
# Demonstration - Relational Database

=>slides: Pg. 3

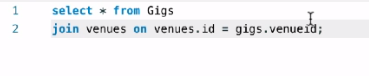


So I'm using MySQL as the database in this course, but you can use any database that you're comfortable with. Everything we show you here will apply across all relational databases. I'm also using MySQL Workbench to access the database to show you these demos here. So you can see on the left‑hand side, we have our tables, and we have the Acts table with the Id, Name, and RecordLabel fields. We have the Gigs table with the VenueId, ActId, TicketsSold, Price, and Date fields. And we have the Venues table with the Name and Capacity fields. So from the slides, we saw we can do things like this.





So I can say select TicketsSold from Gigs. And notice the result is here, and we have a list of all the tickets that have been sold for all gigs. Now obviously, if we look at it like this, it's slightly meaningless on its own. There's no extra information here to tell us what the gig is or what the venue is or what the artist is.



So to get the extra information, we can use a join. So I do a select \* from Gigs, join venues on, and then we specify the join columns, so venues.id = gigs.venueid. And if I run this query, then I'll get way more information. So we get the name of the venue, we get the date of the gig, we get how many tickets have been sold, we get the price, and so on and so forth.



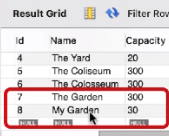
And we also said we can filter. So to filter, we use the where clause. And let's say where venues.name is like arena, so any venue with the word arena in the name. And again, if I run this query, we now get that information for the arena venue.



So what about inserts? So I can say insert into venues. I can specify the column names, so let's say name and capacity, and then we specify the values. So let's say My Garden for the name. And the capacity, let's set this to 30 people. And then if I run this query



and then do a select \* from venues,



you can see that My Garden has been inserted into the table with a capacity of 30. And notice this has an id of 8.



So we can now update the table. So I can say update table name, which is venues, set capacity equal to something else, let's say 35, where id=8. And again, if I run this



and then do the select,



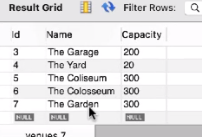
you can see that My Garden now has a capacity of 35. And finally, we can delete,



so delete from venues where id = 8. So if I execute that



and then do the select,



notice that My Garden venue has now disappeared.

# Summary

=>slides: Pg. 13

So what have we just seen? We've just talked about relational databases and the fact that these databases hold related data in tables. So these tables have rows, and the rows of columns, and we relate these tables through keys. We've seen that we can operate on this data, and the basic operations are known as the CRUD operations, create to insert new data into these tables, read to read data from the tables, update to update the data already in the tables, and delete to delete rows from the tables. =>slides: Pg. 14

Okay, so now that we've seen that, we'll now go on and take a look at JDBC. And in the next module, we'll do an introduction to JDBC.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*