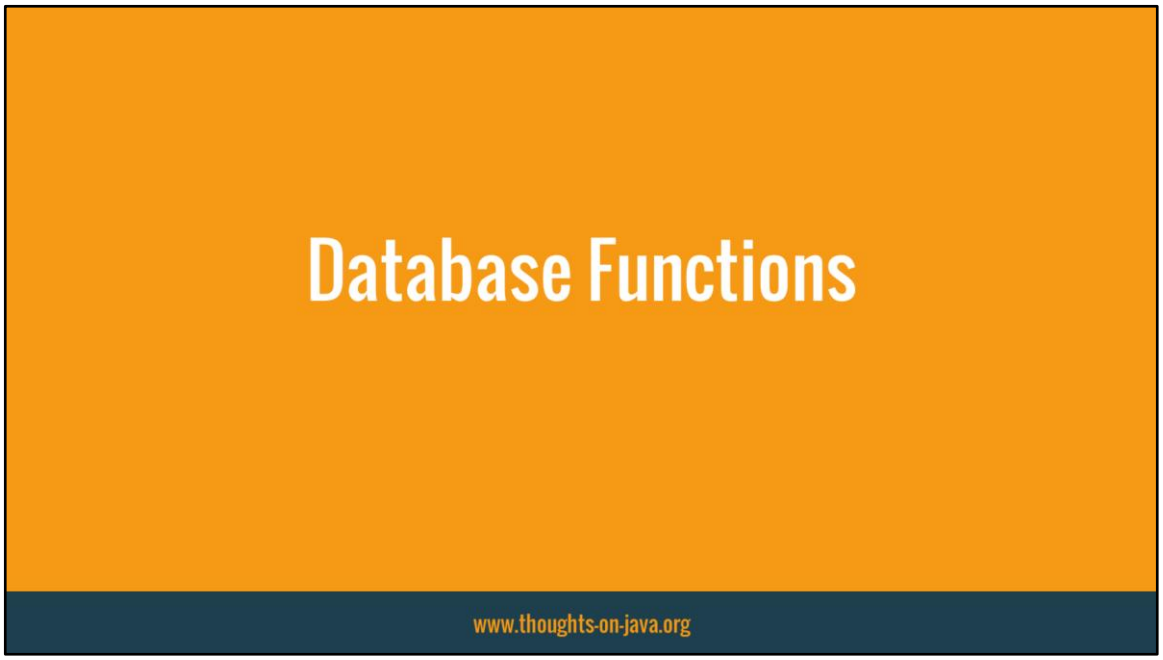
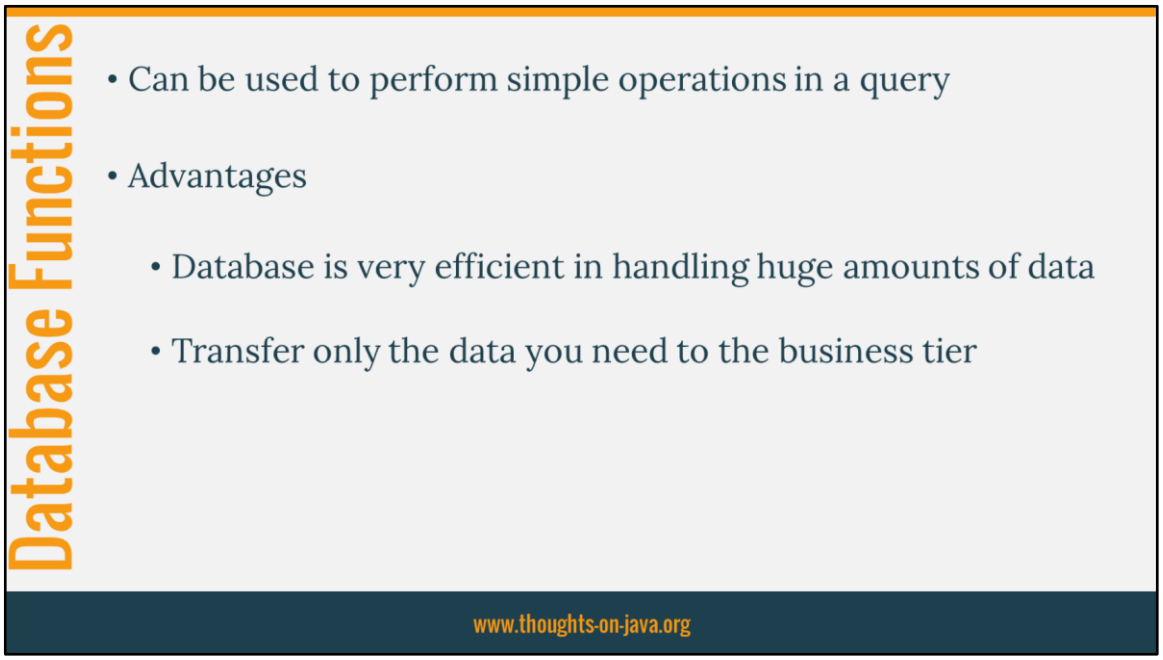
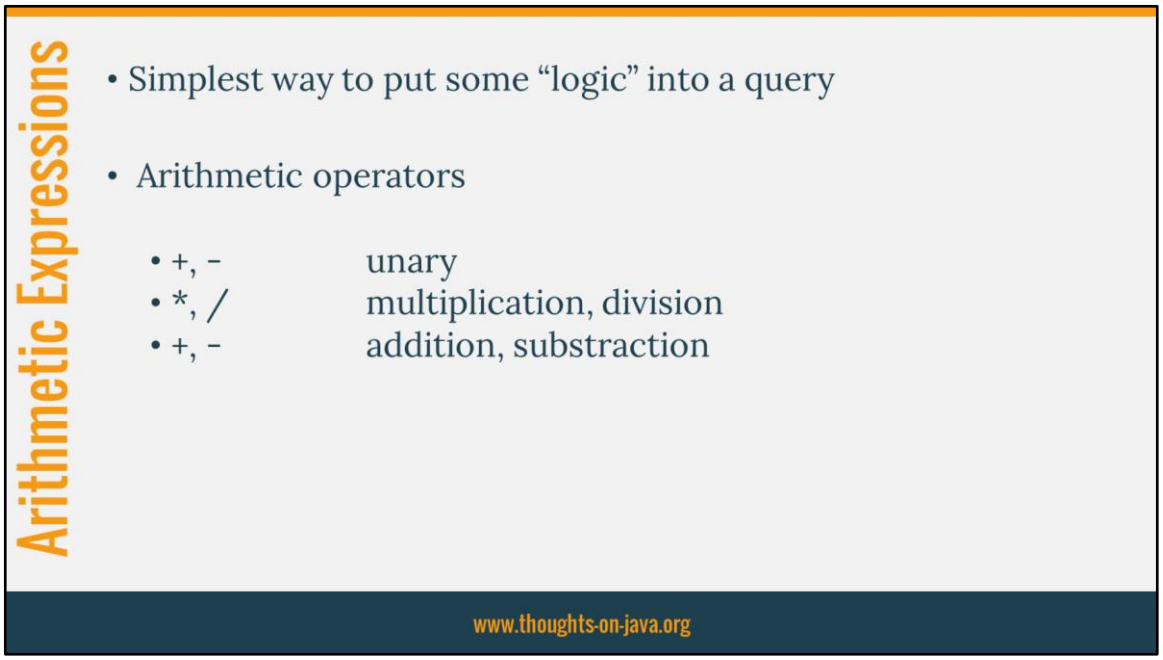


Database functions are one of two options to perform some simple logic within the database, like simple arithmetic operations, counting records or simple string manipulations. Most databases support a lot of different functions, and Hibernate JPA support several of them. And if Hibernate does not support them out of the box, you can easily add them yourself. But more about that at the end of this video.

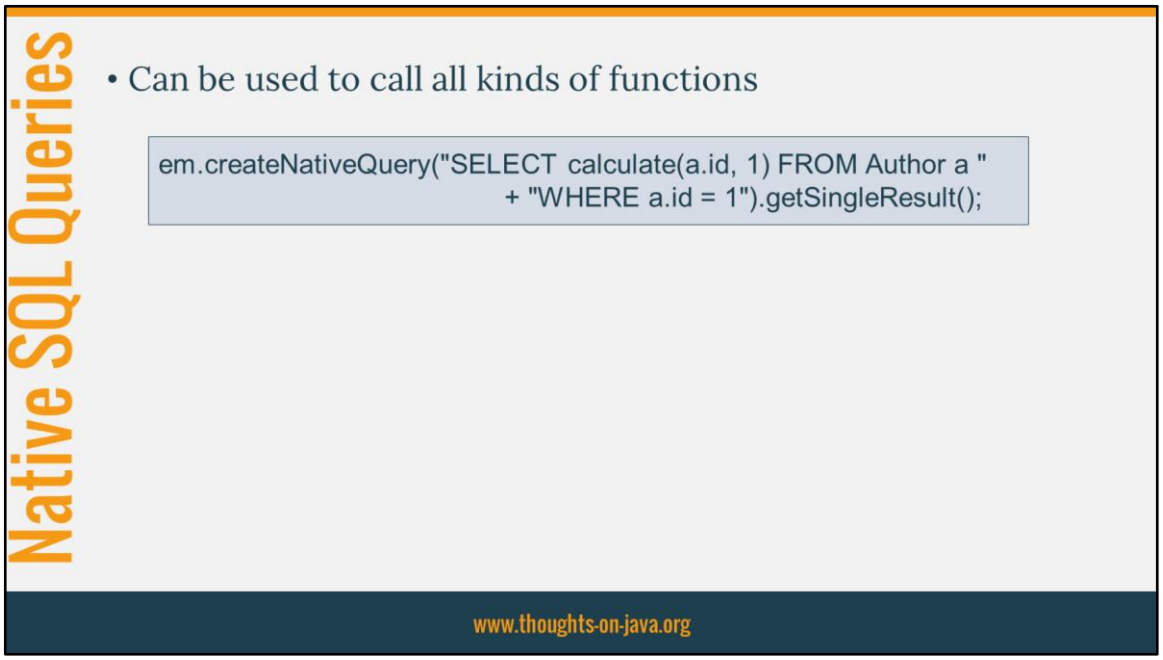


The main idea of using database functions is to let the database do the job it was designed for which is handling vast amounts of data and only transfer the return value of the function to your business tier. You could, for example, use a function to count all books in the database instead of selecting all of them and counting them within your Java application.
The call of the function reduces the amount of data the database has to select and transfer to your application which can provide huge performance benefits.



Arithmetic expressions are the simplest form of logic you can use and allow you to perform simple calculations within the database. You can see the arithmetic operators here on the slide.

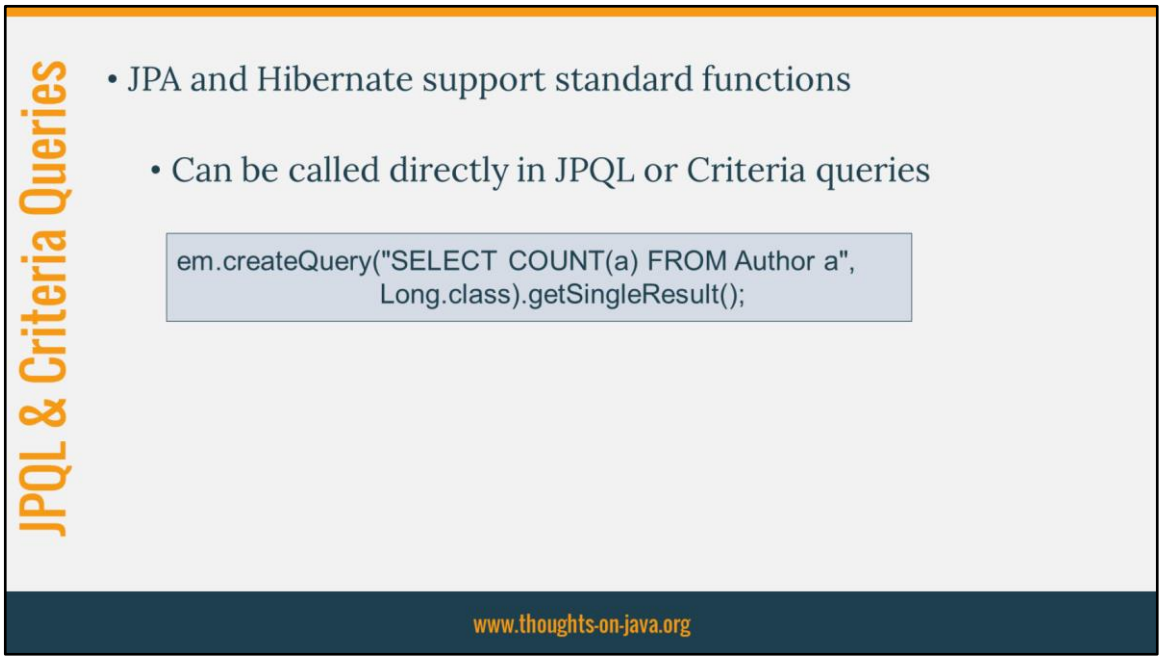
Functions allow you to use and implement more complex logic than the arithmetic expressions, and you can either call them in native SQL queries or JPQL or Criteria queries.
Let's have a look at the native SQL queries first.



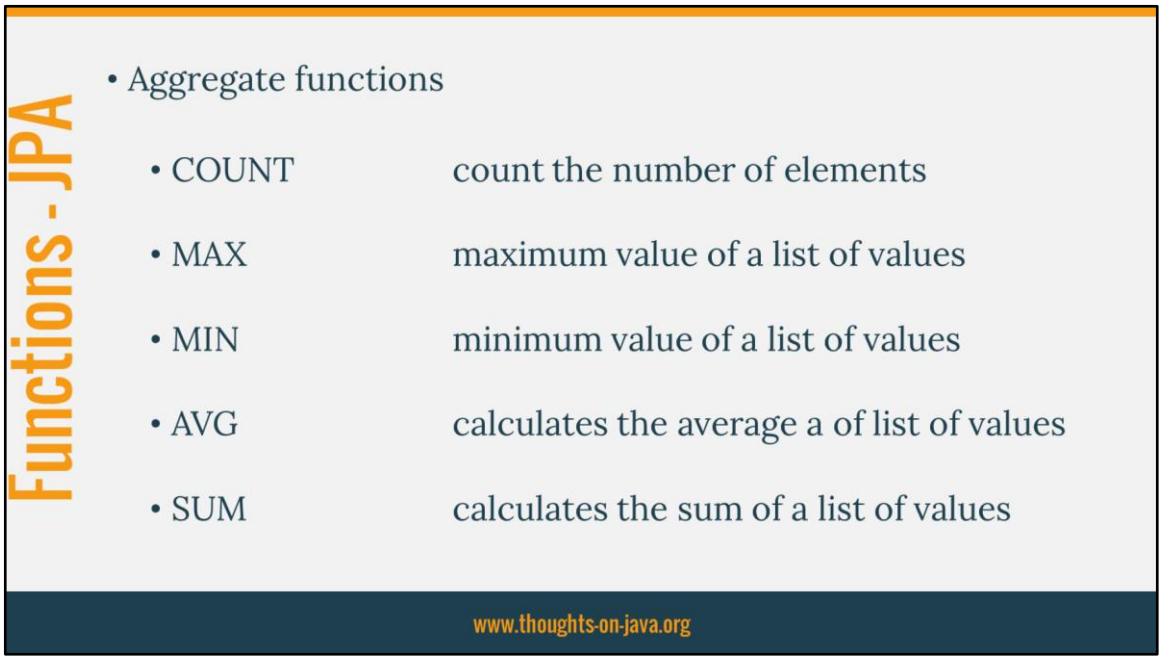
Native queries are the easiest way to use database functions with Hibernate. You only need to call the function within your query and provide the required parameters. You can see an example of such a query in the code snippet.
You already provide an SQL query when you create a native query, so Hibernate does not need to do any additional transformation and does not need to support the function you want to call. This allows you to call all functions supported by your database which includes the ones defined by the SQL standard but also database specific and custom functions.
If you want to do the same in a JPQL or Criteria query, you often need to spend some more effort. But more about that later. Let's just have a look at the demo application and call a custom database function.

Demo

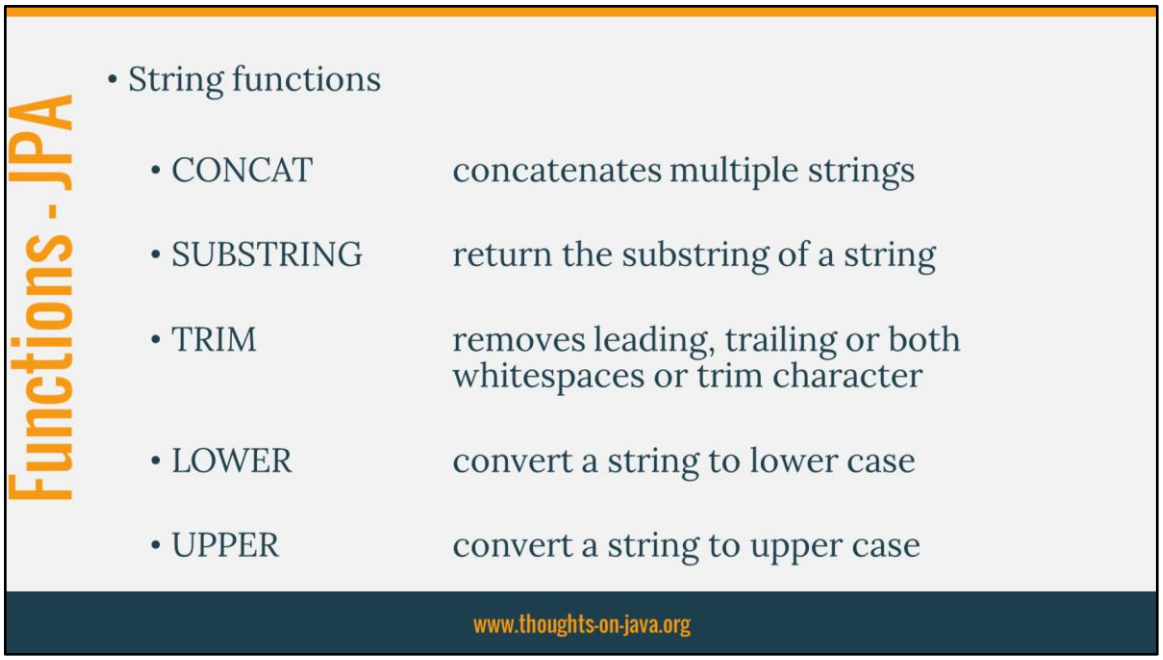
www.thoughts-on-java.org



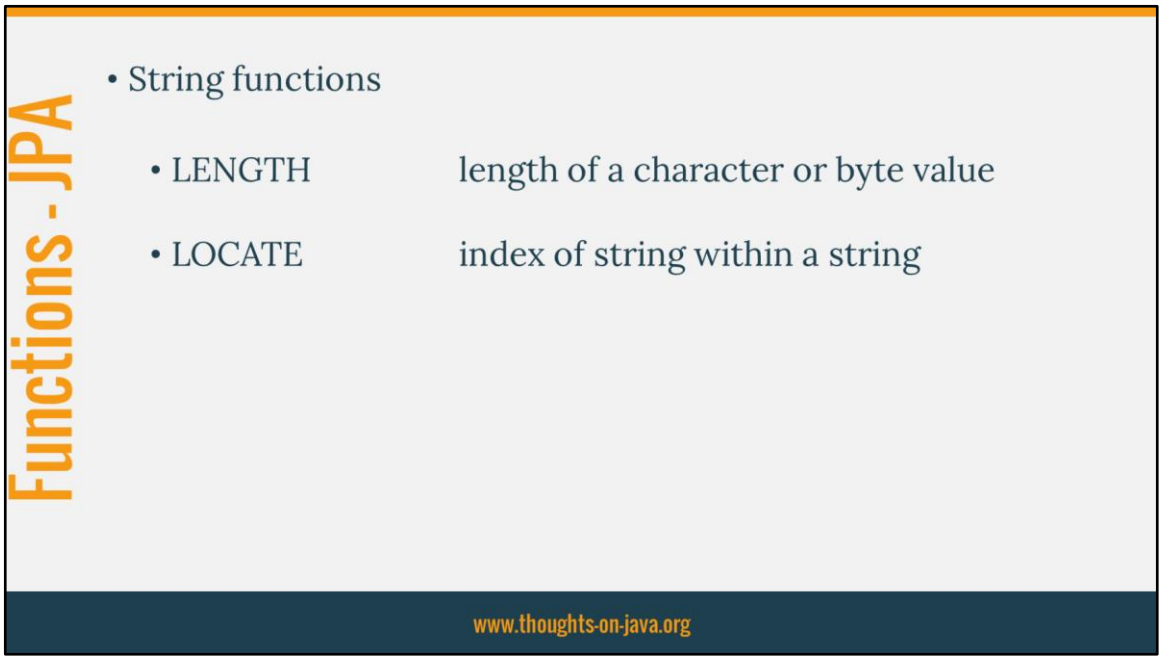
JPA and Hibernate support a set of standard functions, which you can use in the same way as in an SQL statement. You just call the function and provide the required parameters.
Let's have a look which functions are supported by default.



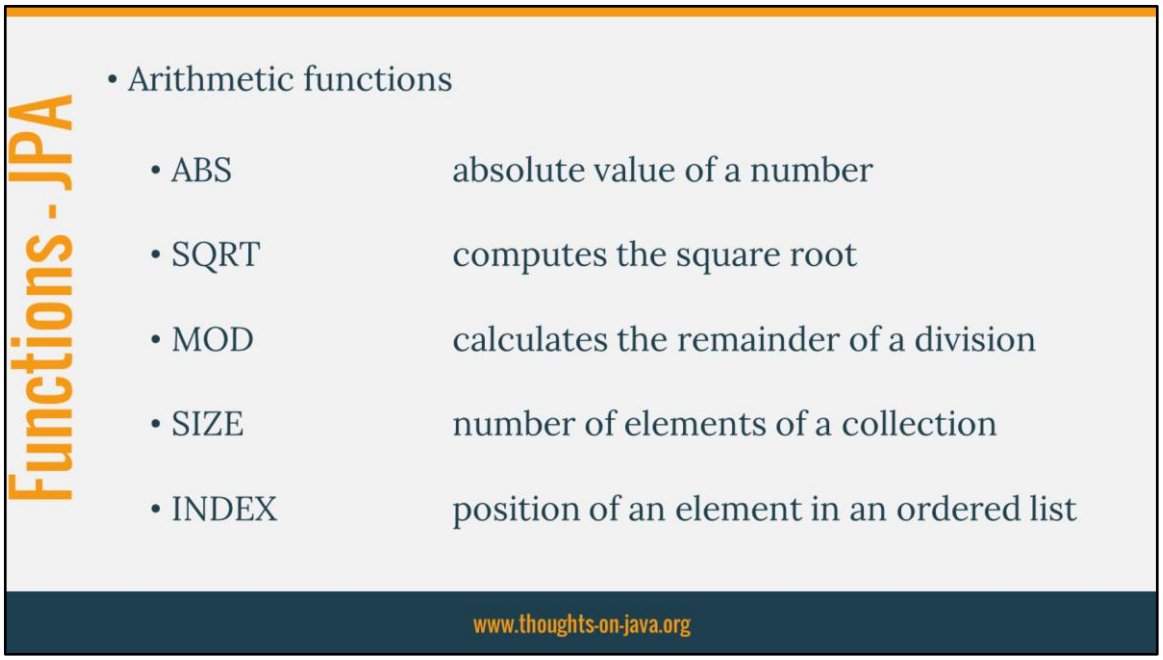
The JPA specification supports a set of aggregate functions which work on a set of values and aggregate them into one. You can see a list of aggregate functions here on the slide. You can count the selected elements, and if you have selected some numeric elements, you can determine the minimum and maximum value, calculate the average and add the values together.



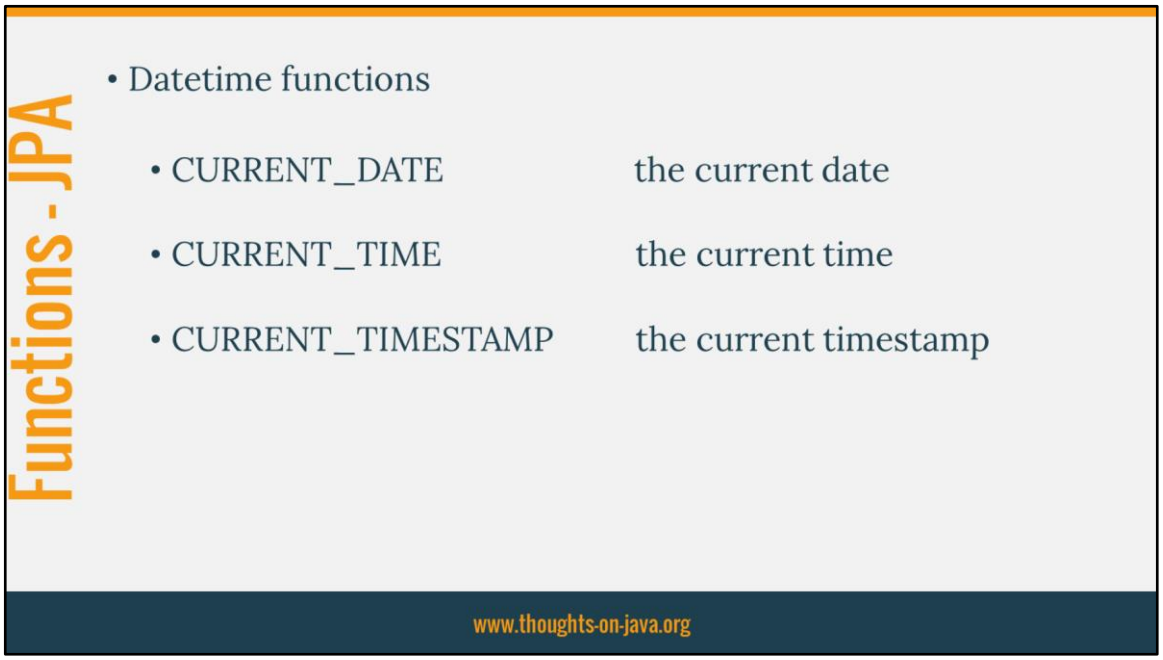
JPA also supports a set of functions to do basic string manipulation. As you can see here, you can concatenate multiple strings, extract a substring, trim whitespace or other characters and convert the string to lower or upper case.



You can also determine the length of the string or get the index of a substring within the selected string.

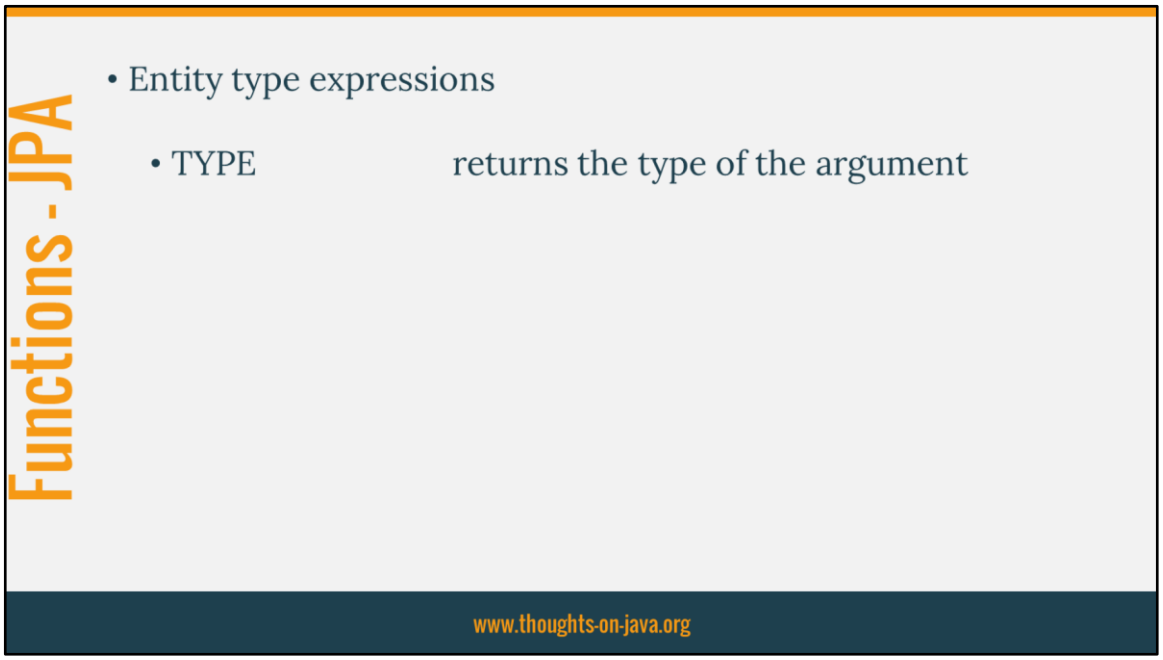


The arithmetic expressions allow you to do some very basic arithmetic operations. If you need to do something more complex, you can use arithmetic functions which help you to get the absolute value of a number, compute the square root or the remainder of a division. You can also use them to get the number of elements or the position of a particular element within a collection.

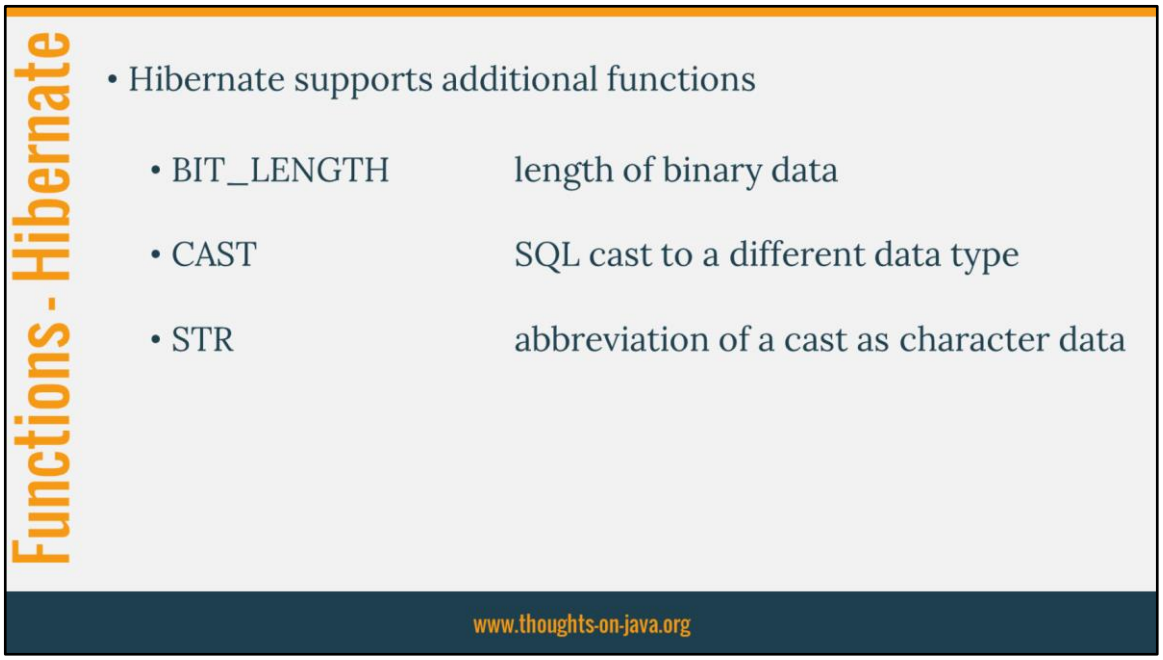


JPA also supports a set of very useful datetime functions which you can use to get the current date, time or timestamp from the database. But be careful, this function gets executed by the database and might return a different time than your Java application.

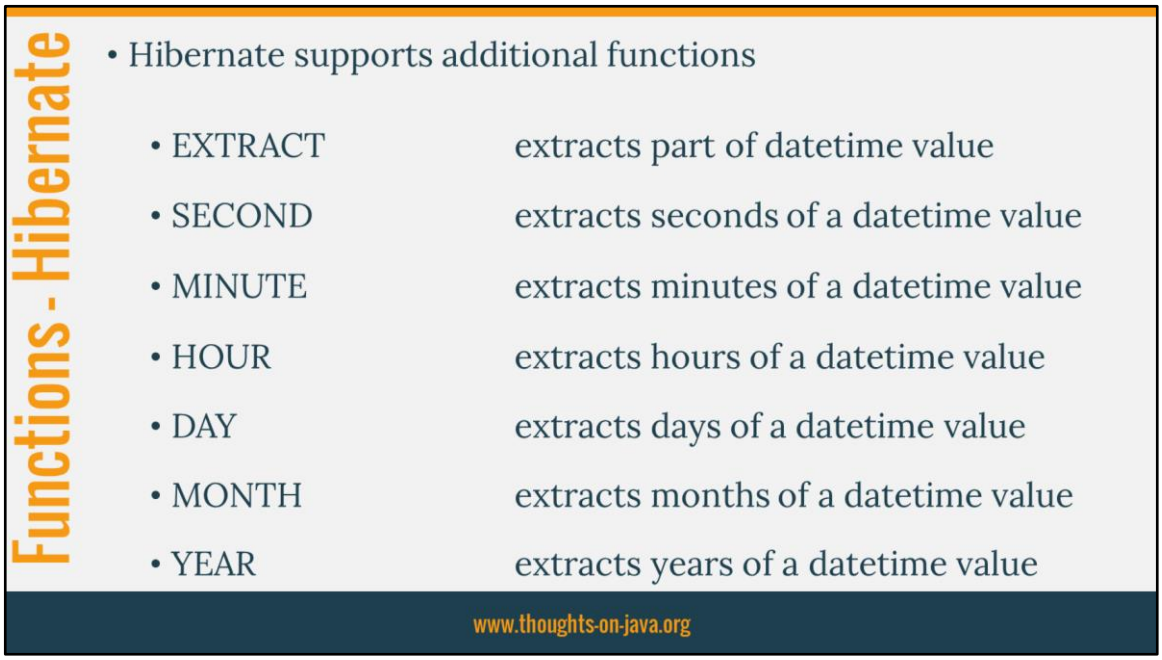
The case expressions allow you to implement simple control structures, like if statements and handling of null values.



And the last type of functions supported by JPA that I want to show you is the TYPE expression. It returns the type of a specific argument which you can use to adapt your query to specific entity types within an inheritance hierarchy.
As you have seen, JPA supports quite a few functions which you can use to implement complex queries, and Hibernate supports even more of them.



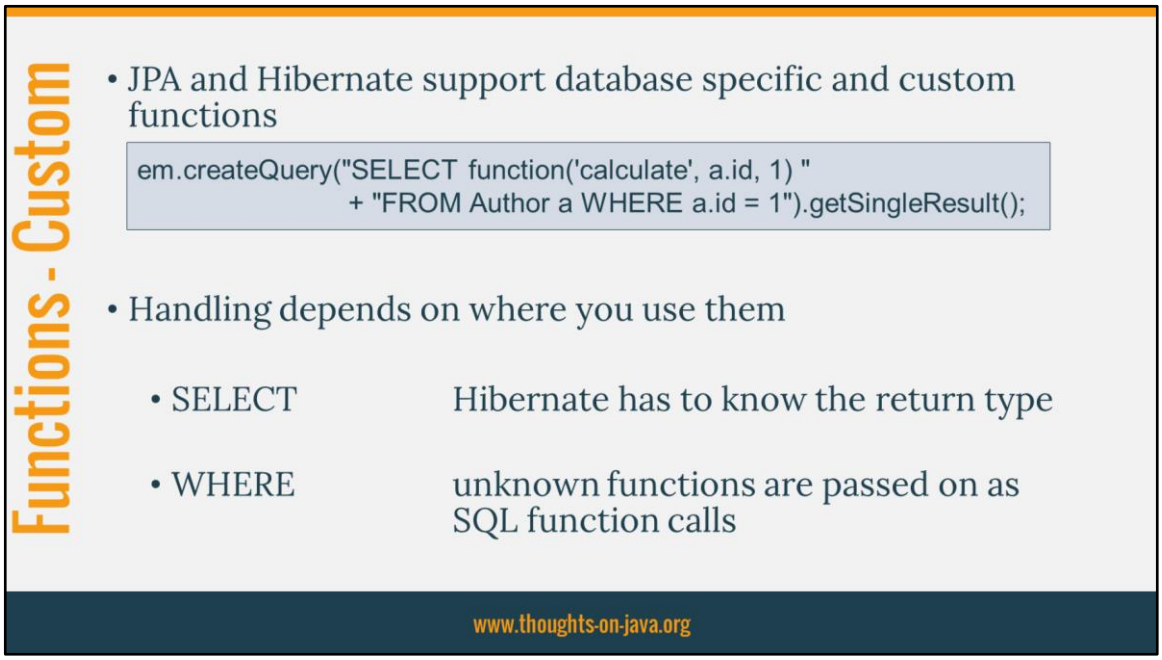
Hibernate offers additional functions to determine the length of binary data and to cast elements to different data types.



And it also offers functions to extract certain parts of a datetime value.
OK, but that is enough theory about the functions supported by JPA and Hibernate. Let's have a quick look at a code sample and then I want to show you how you can call custom and database specific functions with Hibernate.

Demo

www.thoughts-on-java.org

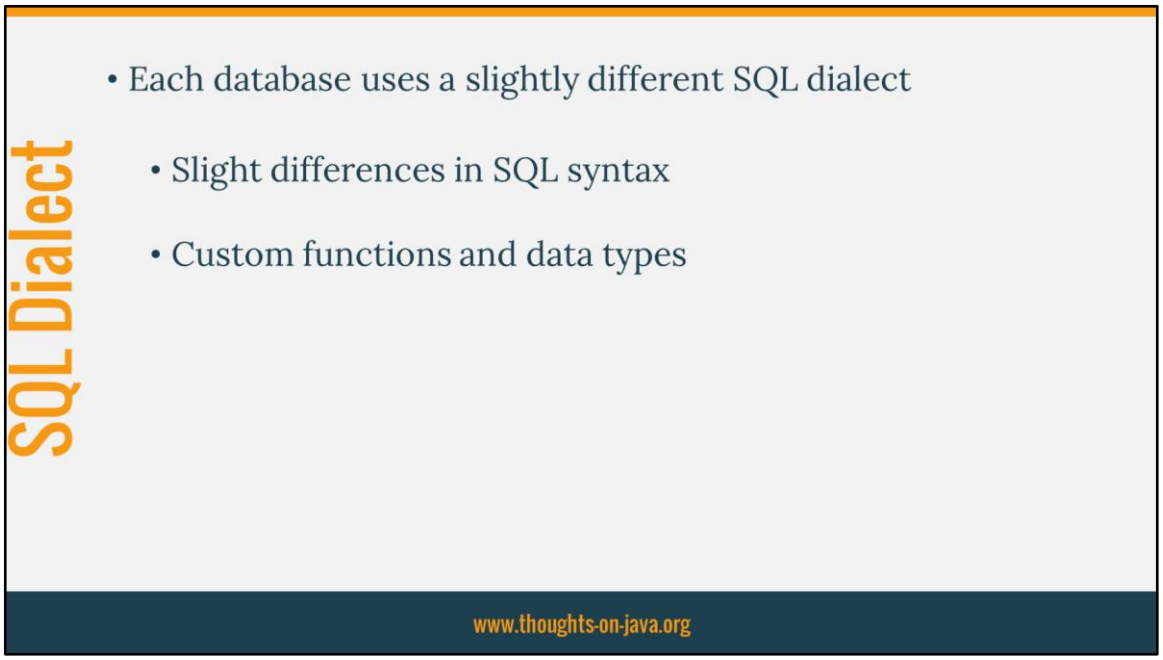


As I told you earlier, you can also call database specific and custom functions with JPA and Hibernate. JPA 2.1 introduced a new function called function which you can use to call any function that is known to the database. As you can see in the code snippet, you only need to provide the name of the function you want to call as the first parameter followed by all its parameters.

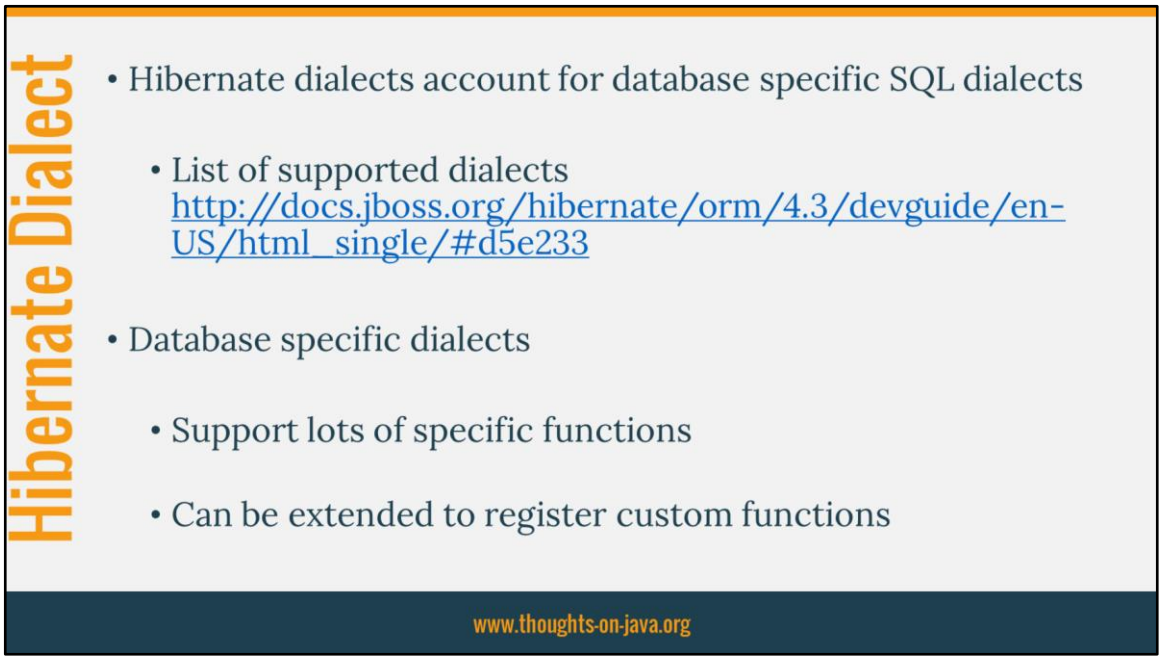
But depending on where you want to use them within your query, you have to do some additional work.
If you call the function in the WHERE part of your query, Hibernate will pass on the function call to the database, and you don't have to do anything on top.
But that is different if you want to call the function in the SELECT part of your query. Hibernate then needs to know the return type of the function which you can specify in the Hibernate dialect. But let's get into the IDE first and I will get into more details about Hibernate dialects afterwards.

Demo

www.thoughts-on-java.org



So why does Hibernate support multiple, database specific dialects? The reason is that each database uses a slightly different SQL dialect with slight differences in the SQL syntax and support for database specific functions and data types.

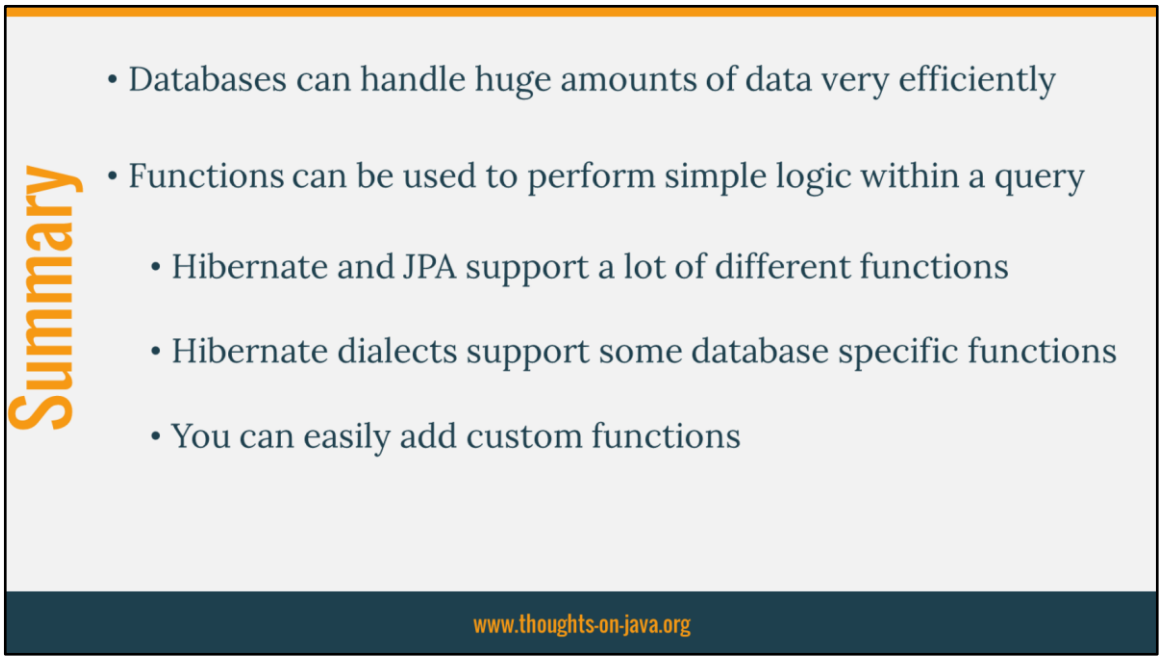


Hibernate offers its own dialects for most common database to support the different, database specific SQL dialects. You can find a list of all supported dialects in the documentation.
These dialects support a lot of database specific functions so you often only need to select the right dialect. But if you want to use your own functions, you need to create your own dialect. This can be easily done by extending the specific dialect for your database.

You can see an example of custom dialect here. I extended the PostgreSQL9Dialect and registered my custom function in the constructor. And then I referenced it in the persistence.xml file. That's all you need to do.
But registering the function in the dialect has a disadvantage if you need to support multiple databases. The dialect is database specific, and you need to create a custom dialect for each database you want to support.

Demo

www.thoughts-on-java.org



Databases handle huge amounts of data very efficiently and you can benefit from it, when you perform simple logic within your query and reduce the amount of data you need to select and transfer to your application.
As I have shown you in the beginning of this video, JPA and Hibernate support a lot of functions out of the box and the database specific Hibernate dialects add support for database specific functions.
If you want to use your own functions, you need to register them in the Hibernate dialect which you can create by extending the specific dialect of your database.

Exercises

www.thoughts-on-java.org