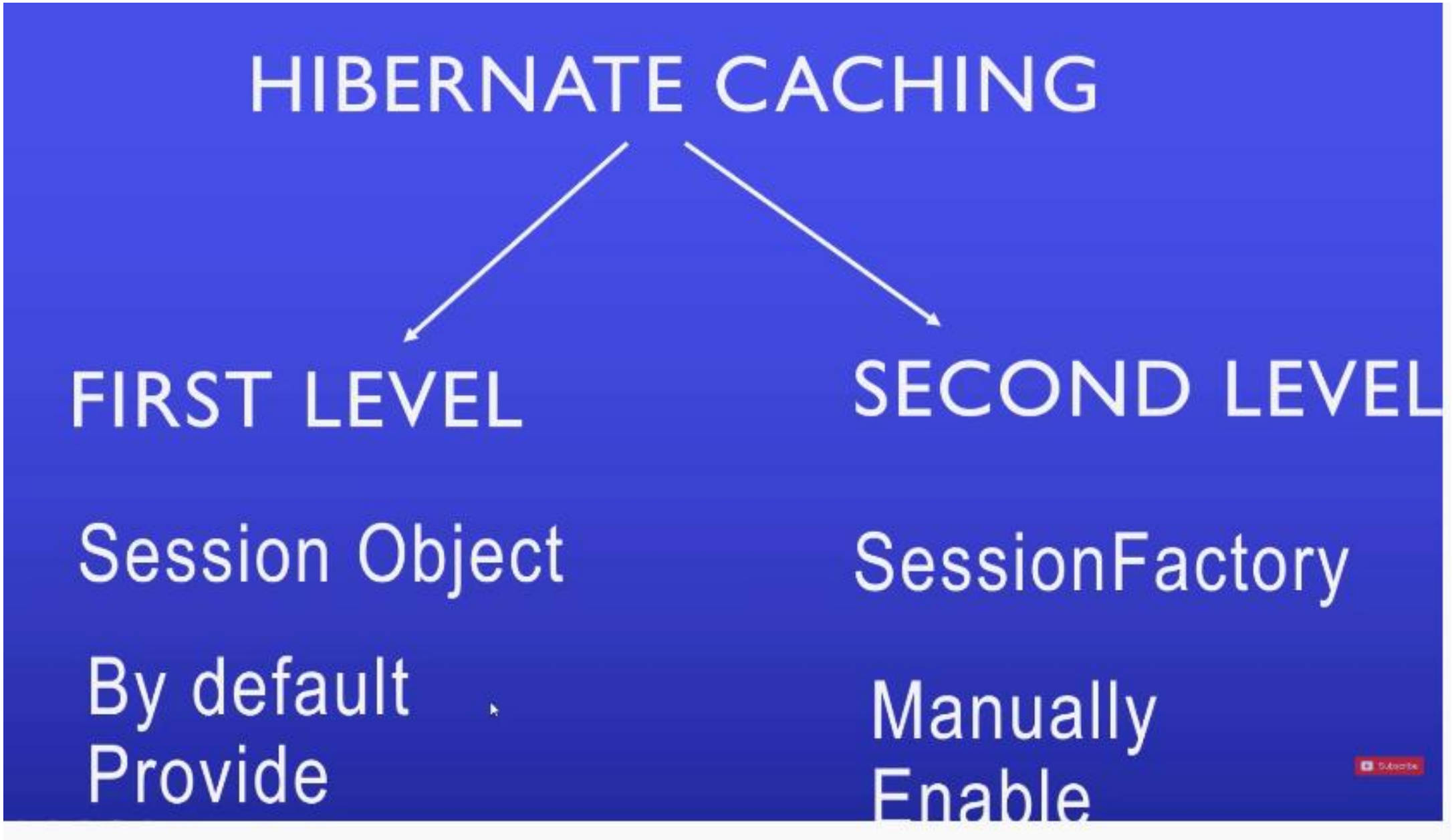
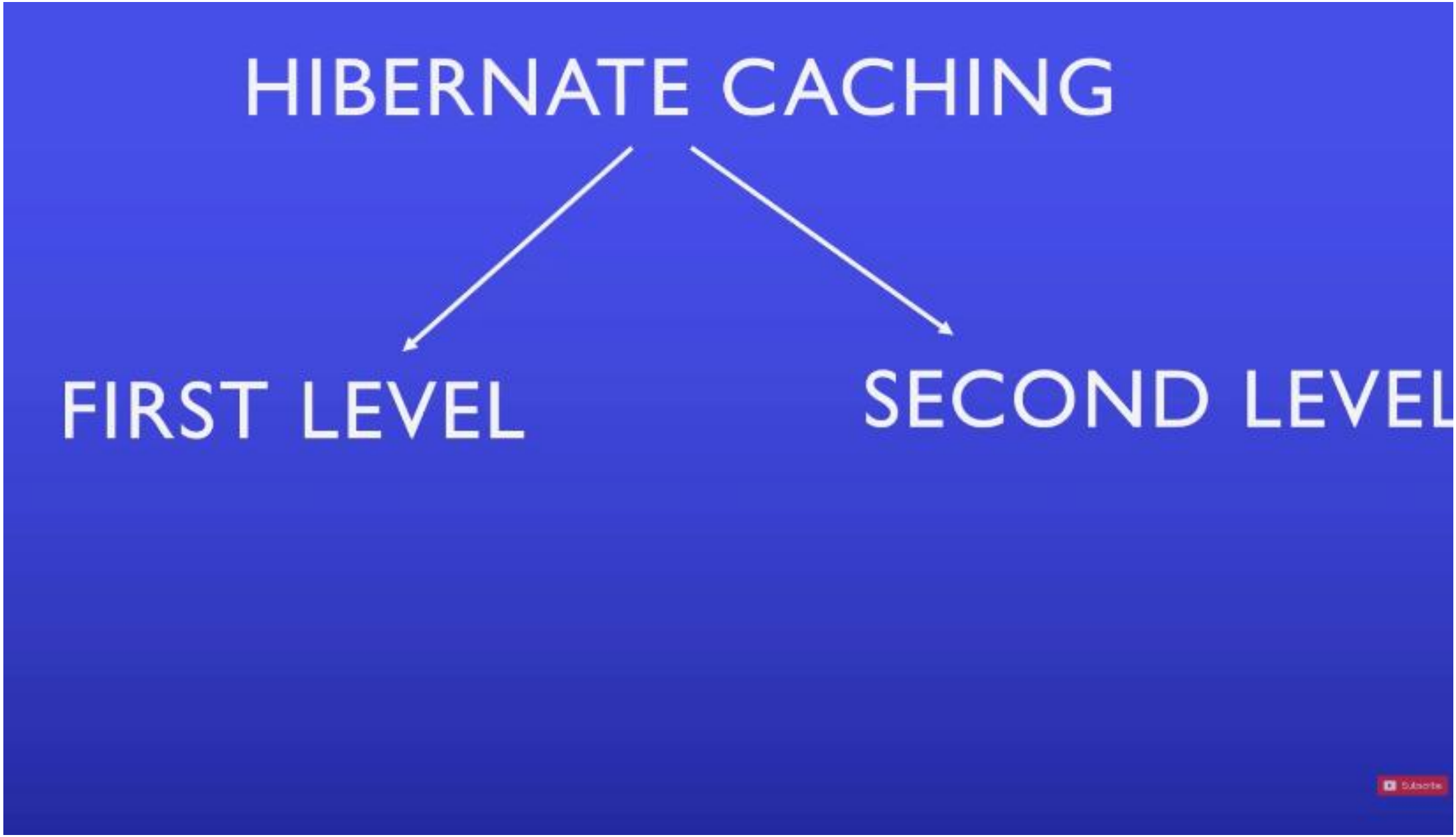

HIBERNATE CACHING
FIRST LEVEL
Session Object
By default
Provide
SECOND LEVEL
SessionFactory
Manually
Enable


HIBERNATE CACHING
FIRST LEVEL
SECOND LEVEL

NOW CACHING COMES

CACHING IN HIBERNATE

Caching is a mechanism to enhance the performance of a Application.

Cache is use to reduce the number of database queries.

HQL

- Database independent
- Easy to learn for programmer.

- from Student

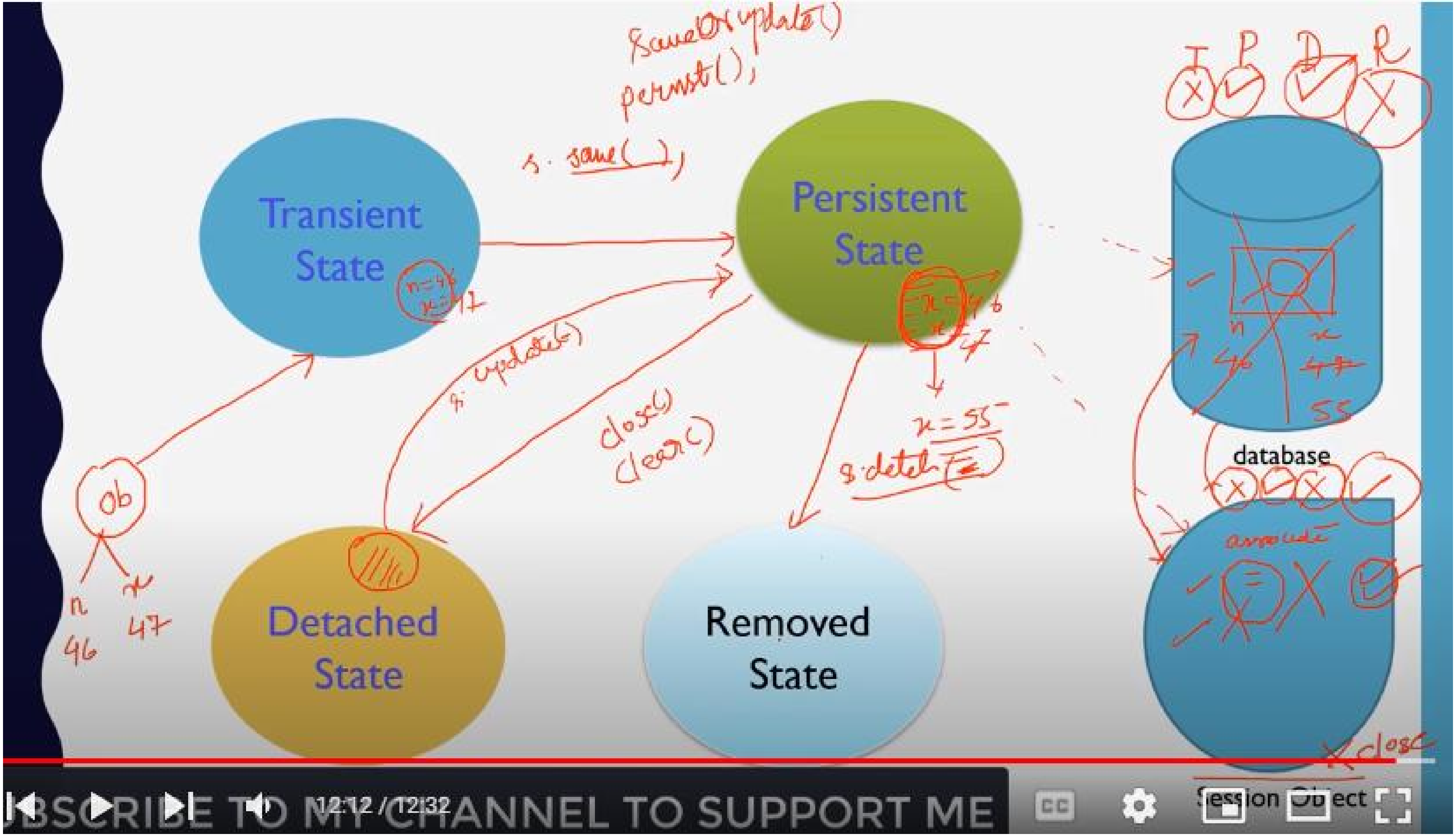
Entity Name

SQL

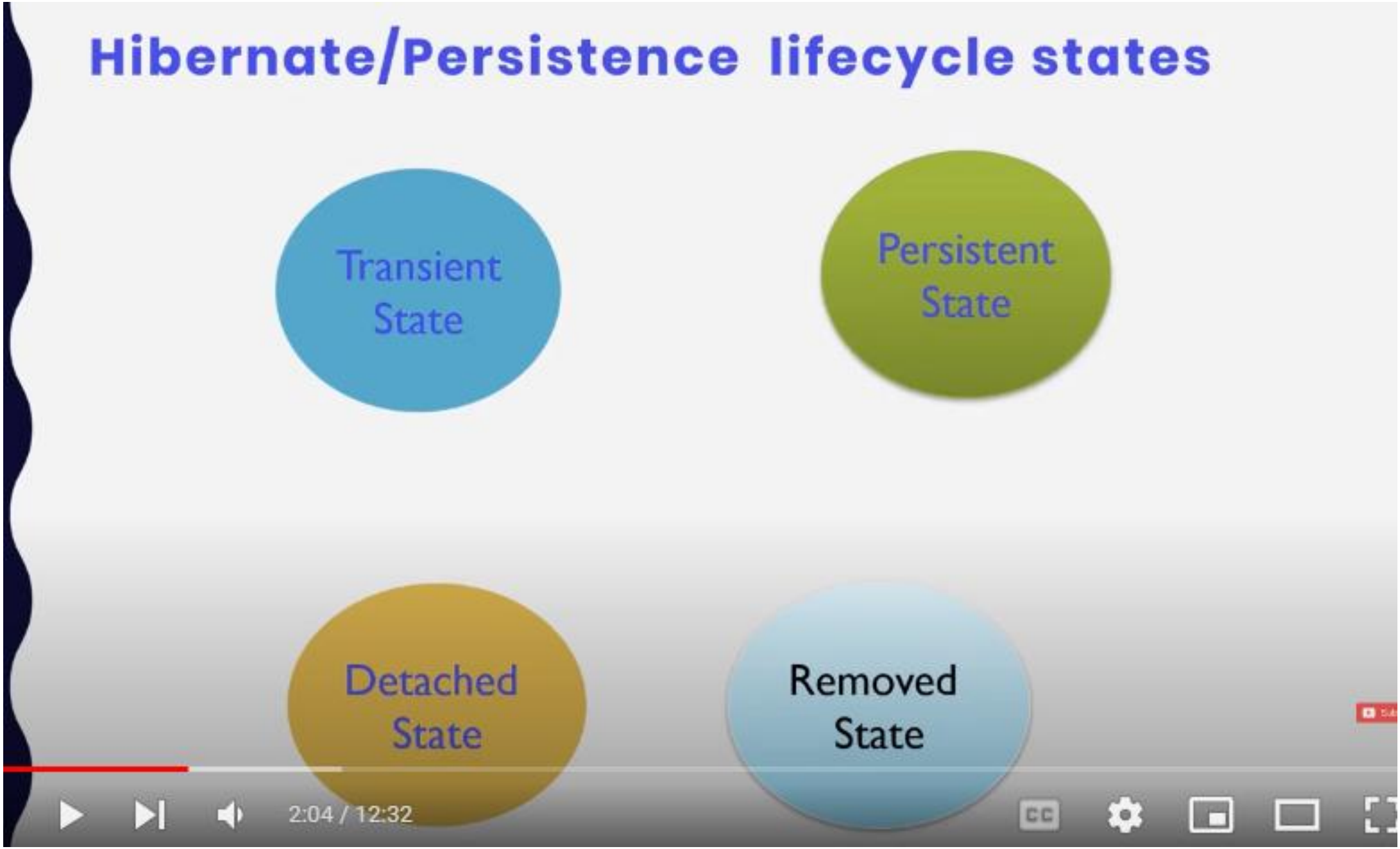
- Database dependent
- Easy to learn for DBA.

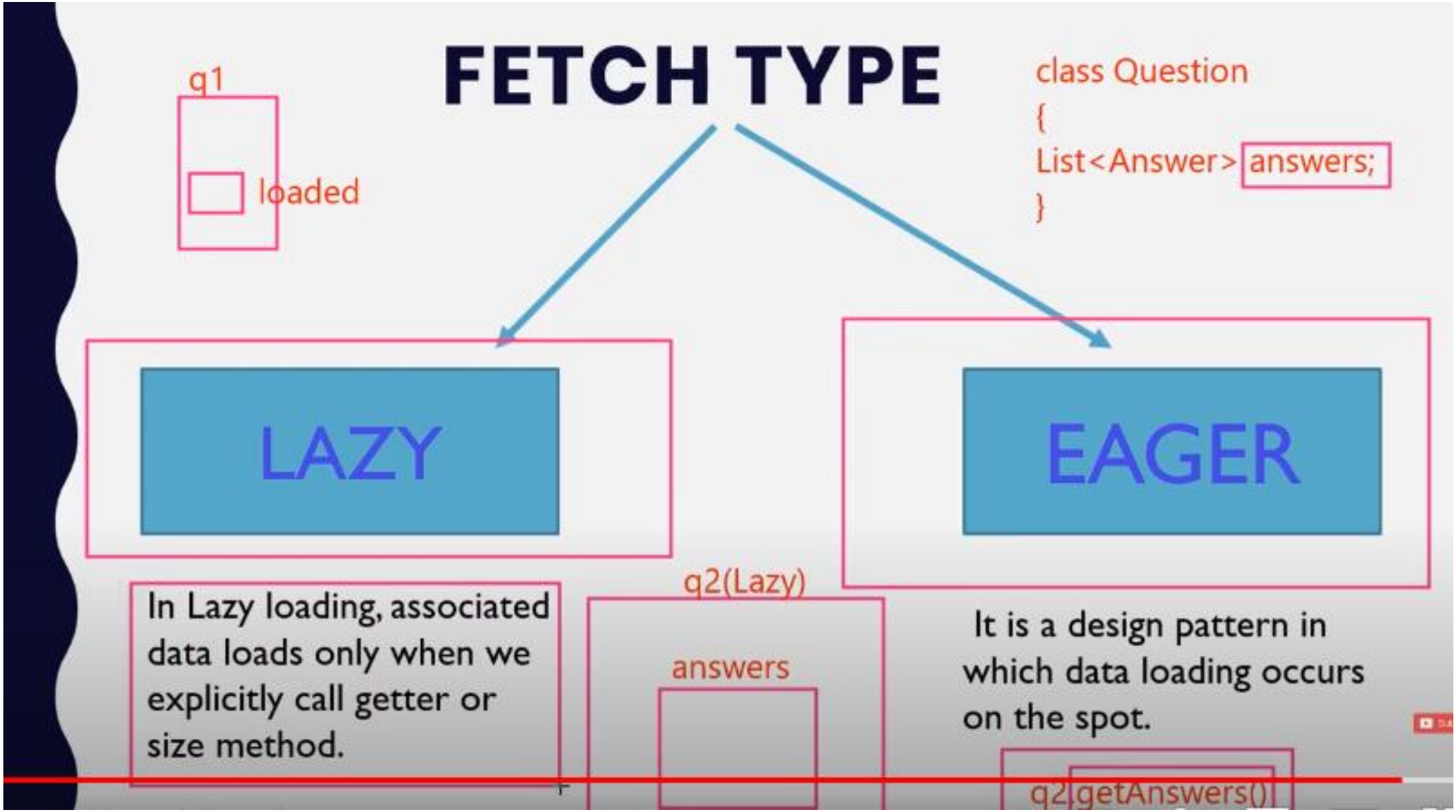
- Select * from Student

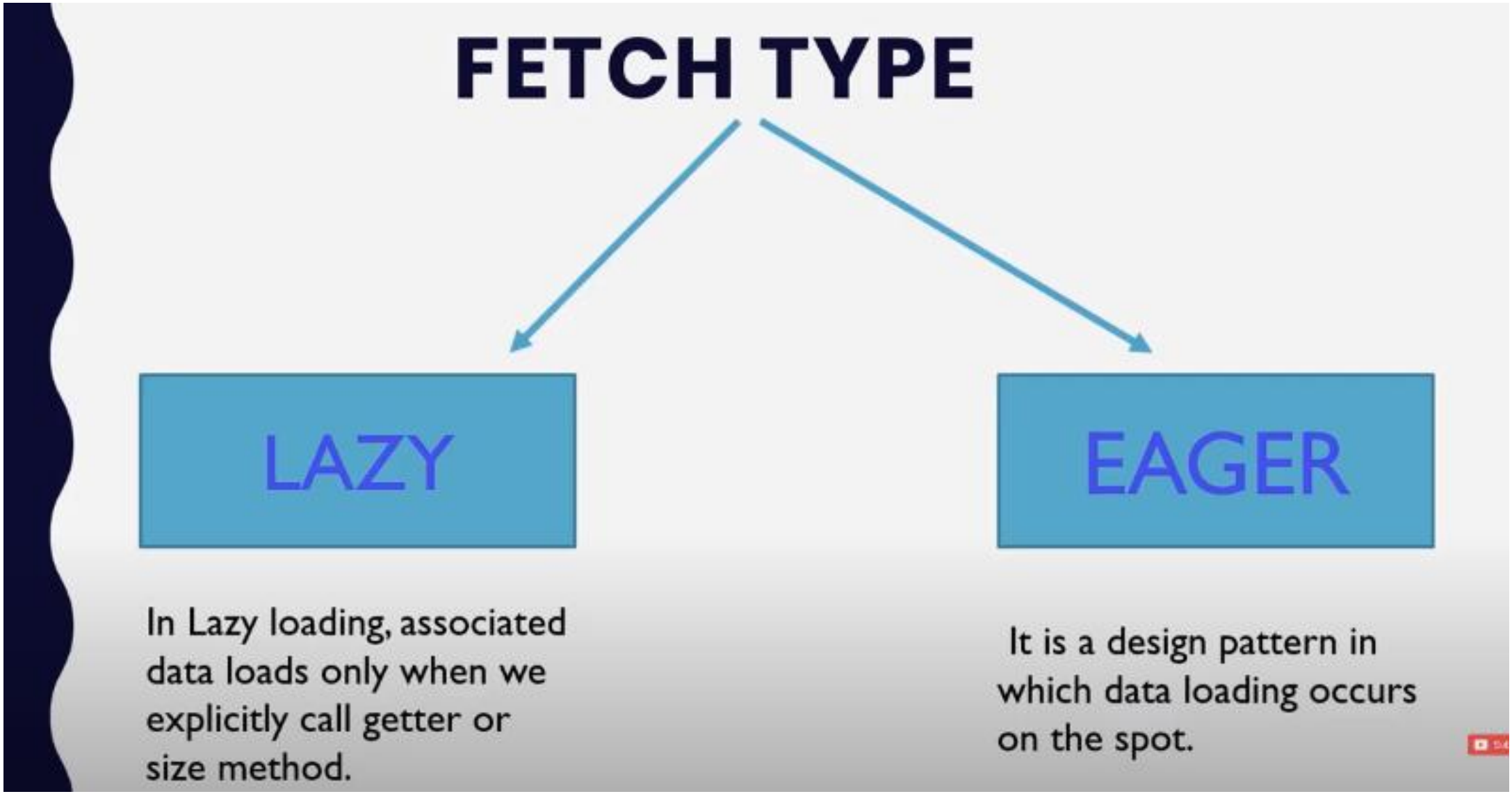
Table Name

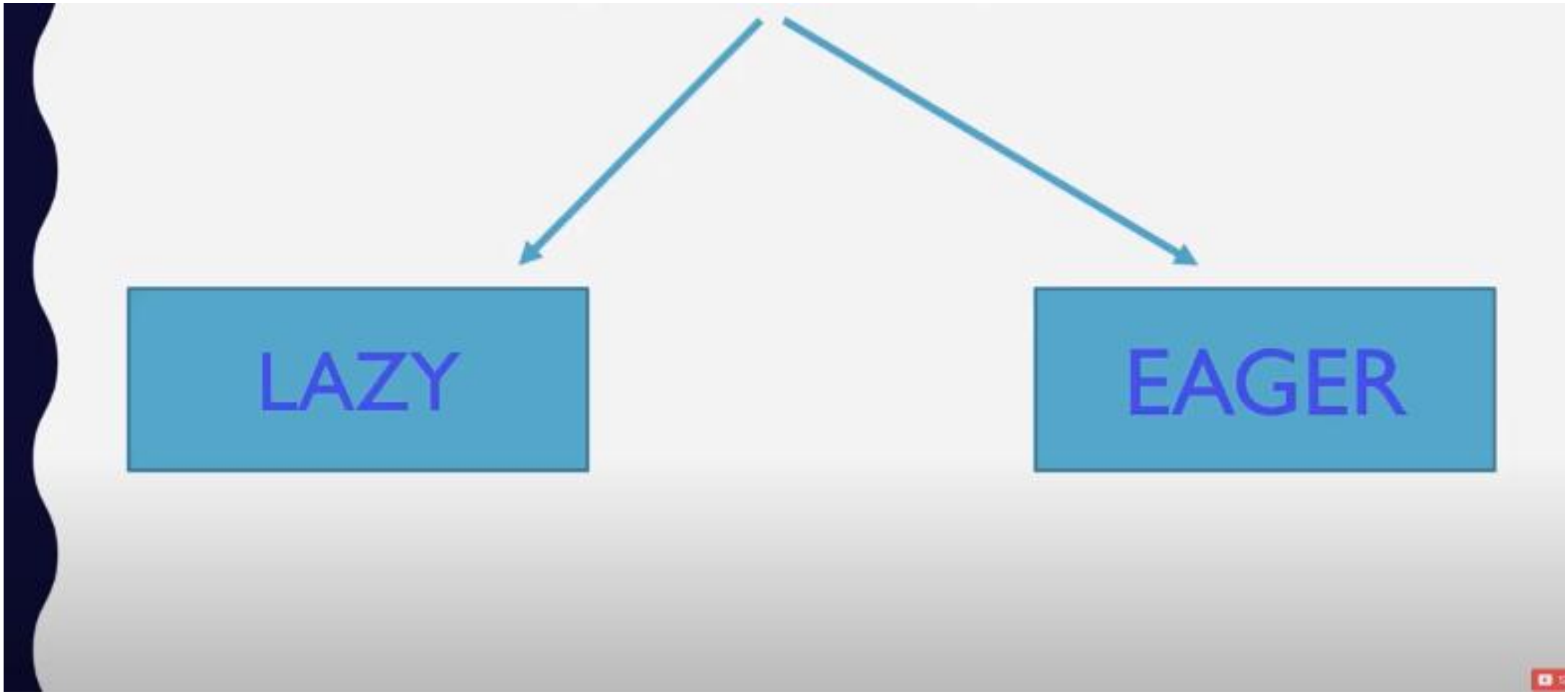


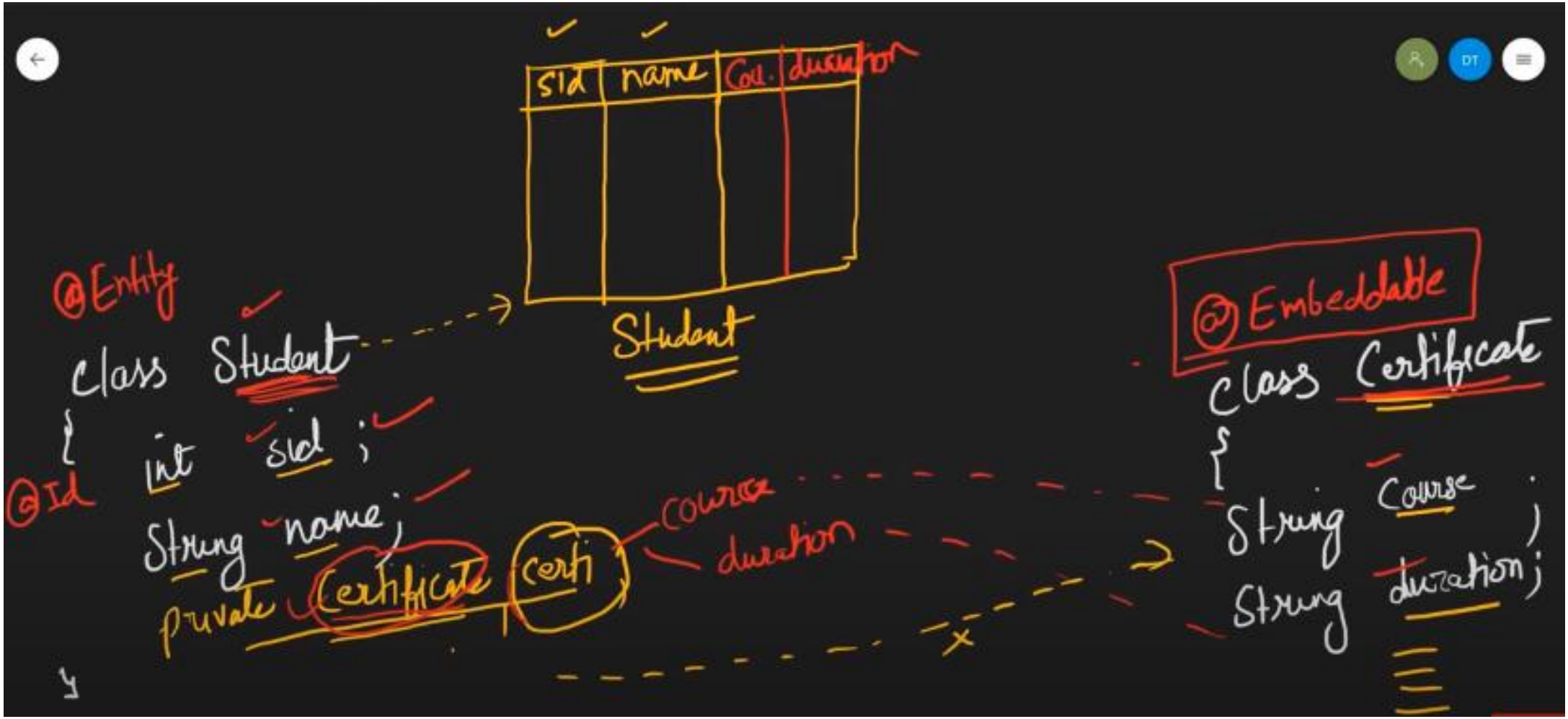
Transient State
Persistent State
Detached State
Removed State
database
saveOrUpdate()
persist();
s.save();
s.update()
close()
clear()
12:12 / 12:32


Hibernate/Persistence lifecycle states
Transient State
Persistent State
Detached State
Removed State
2:04 / 12:32


FETCH TYPE
q1
loaded
class Question
{
List<Answer> answers;
}
LAZY
EAGER
In Lazy loading, associated data loads only when we explicitly call getter or size method.
q2(Lazy)
answers
It is a design pattern in which data loading occurs on the spot.
q2.getAnswers()


FETCH TYPE
LAZY
EAGER
In Lazy loading, associated data loads only when we explicitly call getter or size method.
It is a design pattern in which data loading occurs on the spot.


LAZY
EAGER

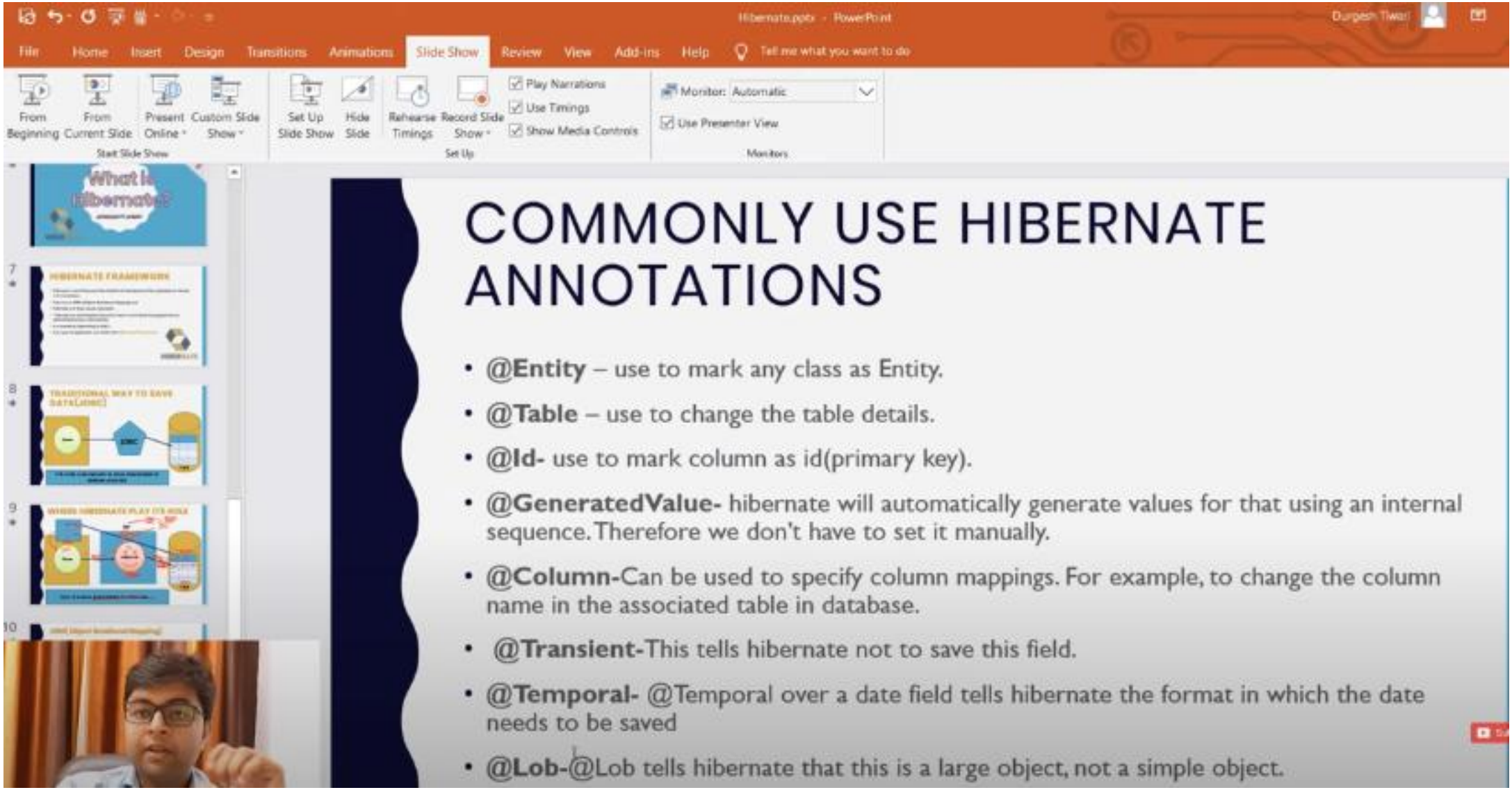

sid
name
Cou.
duration
Student
@Entity
class Student
{
@Id int sid;
String name;
private Certificate certi
course
duration
@Embeddable
class Certificate
{
String course
String duration;

FETCH DATA

get( )	load()
get method of Hibernate Session returns null if object is not found in cache as well as on database.	load() method throws ObjectNotFoundException if object is not found on cache as well as on database but never return null.
get() involves database hit if object doesn't exists in Session Cache and returns a fully initialized object which may involve several database call	load method can return proxy in place and only initialize the object or hit the database if any method other than getId() is called on persistent or entity object. This lazy initialization increases the performance.
Use if you are not sure that object exists in	Use if you are sure that object exists.

COMMONLY USE HIBERNATE ANNOTATIONS

- **@Entity** – use to mark any class as Entity.
- **@Table** – use to change the table details.
- **@Id-** use to mark column as id(primary key).
- **@GeneratedValue-** hibernate will automatically generate values for that using an internal sequence. Therefore we don't have to set it manually.
- **@Column-**Can be used to specify column mappings. For example, to change the column name in the associated table in database.
- **@Transient-**This tells hibernate not to save this field.
- **@Temporal-** @Temporal over a date field tells hibernate the format in which the date needs to be saved


Hibernate.pptx - PowerPoint
Durgesh Tiwari
File Home Insert Design Transitions Animations Slide Show Review View Add-ins Help Tell me what you want to do
From Beginning
From Current Slide
Present Online
Custom Slide Show
Start Slide Show
Set Up Slide Show
Hide Slide
Rehearse Timings
Record Slide Show
Play Narrations
Use Timings
Show Media Controls
Set Up
Monitor: Automatic
Use Presenter View
Monitors
What is Hibernate?
7
8
9
10
COMMONLY USE HIBERNATE ANNOTATIONS
• @Entity – use to mark any class as Entity.
• @Table – use to change the table details.
• @Id- use to mark column as id(primary key).
• @GeneratedValue- hibernate will automatically generate values for that using an internal sequence. Therefore we don't have to set it manually.
• @Column-Can be used to specify column mappings. For example, to change the column name in the associated table in database.
• @Transient-This tells hibernate not to save this field.
• @Temporal- @Temporal over a date field tells hibernate the format in which the date needs to be saved
• @Lob-@Lob tells hibernate that this is a large object, not a simple object.

> select expression > Right click on that > Choose Inspect

Ex:

```
if( a>b && c<d || (a>m+n-2) ) { }
```

e) Stop (Resume) [F8] : To finish current execution also used for next nearet breakpoint

f) Step Into (F5) : Go inside method body from method call.

g) Step Return (F7): Come back to method call from method body.

F11 - Debug
F5 - Go to method
F6 - Next Line
F7 - Return back to call
F8 - Finish Execution
ctrl+shift+I : Inspect
ctrl+shift+B : create/remove breakpoint

> select expression > Right click on that > Choose Inspect

Ex:

```
if( a>b && c<d || (a>m+n-2) ) { }
```

e) Stop (Resume) [F8] : To finish current execution.

f) Step Into (F5) : Go inside method body from method call.

g) Step Return (F7): Come back to method call from

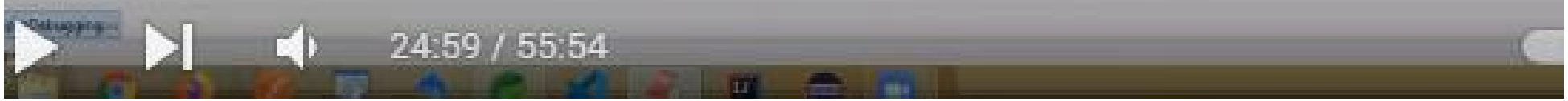
F11 - Debug

F6 - Next Line

F8 - Finish Execution

ctrl+shift+I : Inspect

ctrl+shift+B : create/remove breakpoint



> select expression > Right click on that > Choose Inspect

Ex:

```
if( a>b && c<d || (a>m+n-2) ) { }
```

e) Stop (Resume) [F8] : To finish current execution.

F11 - Debug
F6   - Next Line
F8   - Finish Execution
ctrl+shift+I : Inspect
ctrl+shift+B : create/remove breakpoint

b) Start Debug: Run Menu > Debug Option (F11 | Fn+F11)

c) Step Over (F6) : Execute current line and goto next line

d) Inspect (ctrl+shift+I) : Check selected expression value.
   > select expression > Right click on that > Choose Inspect

Ex:

```
if( a>b && c<d || (a>m+n-2) ) { }
```

e) Stop (Resume) [F8] : To finish current execution.

F11
F6
F8
ctrl+shift+I
ctrl+shift