#### https://www.halvorsen.blog



# PostgreSQL

#### Hans-Petter Halvorsen

#### Contents

- Introduction
- <u>Getting Started with PostgreSQL</u>
  - pgAdmin, Query Tool, Create Database and Tables, ERD Tool
- <u>SQL Queries</u>
  - INSERT, SELECT, UPDATE and DELETE
- <u>Views</u>
- <u>Stored Procedures</u>
- <u>Triggers</u>

https://www.halvorsen.blog



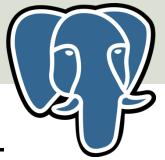
PostgreSQL

## Introduction

#### Hans-Petter Halvorsen



## Introduction



- PostgreSQL is an open-source objectrelational database system.
- Many other SQL databases exists like SQL Server, MySQL, MariaDB etc.
- We will focus on PostgreSQL in this Tutorial.



- PostgreSQL is an open-source object-relational database system.
- PostgreSQL exists for Windows, macOS and Linux.
- Homepage: <a href="https://www.postgresql.org">https://www.postgresql.org</a>
- EnterpriseDB (EDB) is the company that is one of the largest contributor to PostgreSQL and responsible for the installer.
- EDB offer paid services for enterprises, but PostgreSQL itself is free.
- ERD Download Page: <u>https://www.enterprisedb.com/downloads/postgres-postgresql-downloads</u>

#### Installation

<table-of-contents> Setup</table-of-contents>		Setup	- • ×
	Setup - PostgreSQL	Select Components	
😳 EDB	Welcome to the PostgreSQL Setup Wizard.	Select the components you want to install; clear the you are ready to continue.	components you do not want to install. Click Next when
		<ul> <li>PostgreSQL Server</li> <li>pgAdmin 4</li> <li>Stack Builder</li> <li>Command Line Tools</li> </ul>	Click on a component to get a detailed description
$\sim$			📲 Setup — 🗆 🗙
LA X			Password
PostgreSQL			Please provide a password for the database superuser (postgres).         Password         Retype password
		InstallBuilder	Make sure to remember the Password
	< Back Next :	>	
you need to	e default installatio create a password nat you need to ren		InstallBuilder 

https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

## pgAdmin

- pgAdmin is graphical tool for managing your PostgreSQL database.
- pgAdmin is part of the installer from EDB.
- If you prefer, you can also use "SQL Shell (psql)", which is a terminal based program where you can write and execute SQL syntax in the command-line terminal.

https://www.halvorsen.blog

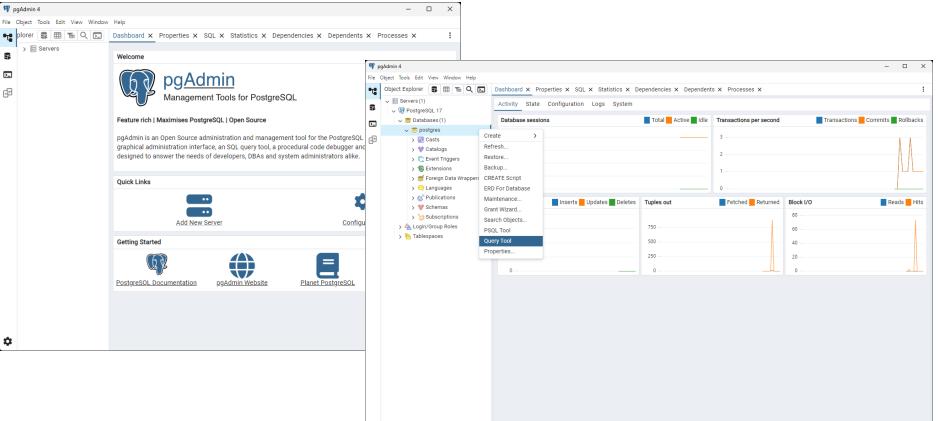
PostgreSQL

# Getting Started with PostgreSQL

Hans-Petter Halvorsen

**Table of Contents** 

## pgAdmin



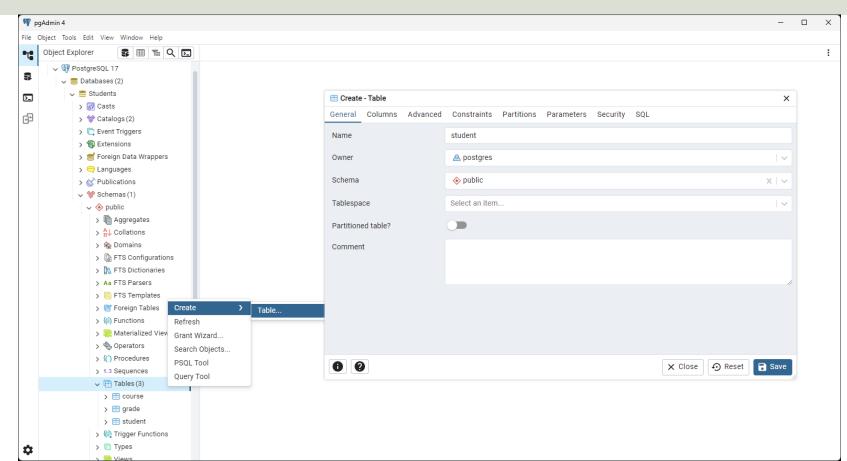
## Query Tool

Ф P	gAdmir	4		-		×
File	Object	Tools Edit View Window Help				
•	Weld	ome 🚦 postgres/postgres@PostgreSQL 17* 🗙				
	₩.	postgres/postgreSQL 17 🗸 🕏				0
072						
>_	Ouer	y Query History	Scratch Pad ×			2
문	1	select version();				•
	Data	Output Messages Notifications				2
	=+	🚡 🗸 🗂 🗸 🔹 🛃 🛃 🛃 🖍 SQL Showing rows: 1 to	1 🖍 Page No: 1	of 1 🛛 🖣	4 ••	►I.
		version				
	1	text PostgreSQL 17.4 on x86_64-windows, compiled by msvc-19.42.34436, 64-bit				
\$						
-	TOTA	I rows: 1 Query complete 00:00:00.050		CRLF Lr	1,00	10

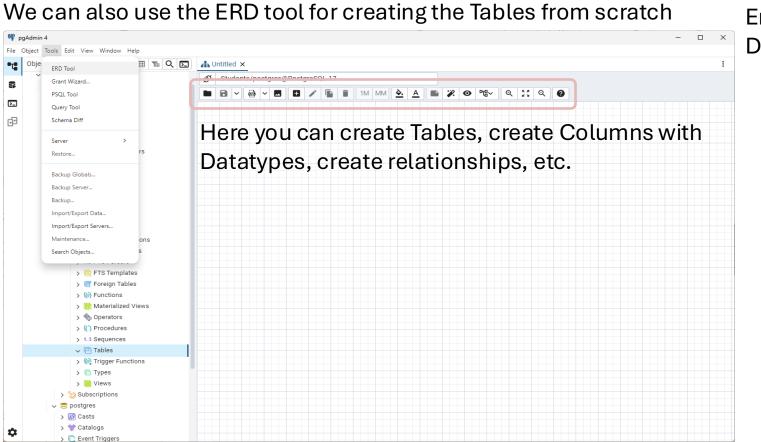
#### **Create new Database**

q	pgAdmin 4		– 🗆 X
Fil	e Object Tools Edit View Window Help		
•	Object Explorer 💲 🌐 🚡 🔍 💽	ashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes ×	:
	✓	Activity State Configuration Logs System	×
D	✓ ■ Databases (1) ✓ ● postgres Create	General Definition Security Parameters Advanced SQL	Transactions Commits Rollbacks
Ċ		Database Students	
	> 🛱 Event Triggers	4 — OID	
	> 💣 Foreign Data Wrappers > 🤤 Languages	2 Owner 🙇 postgres	
	> (S) Publications > (S) Schemas	Comment Tuple:	<b>k l/0</b> ■ Reads ■ Hits
	> 2 Subscriptions	100	// D
	> 🔁 Tablespaces	50	0 ———
		25	
		0	$ - \frac{1}{1} \frac$
		Close Reset S	ave
×	r		

#### **Create new Tables**



## **ERD** Tool in pgAdmin

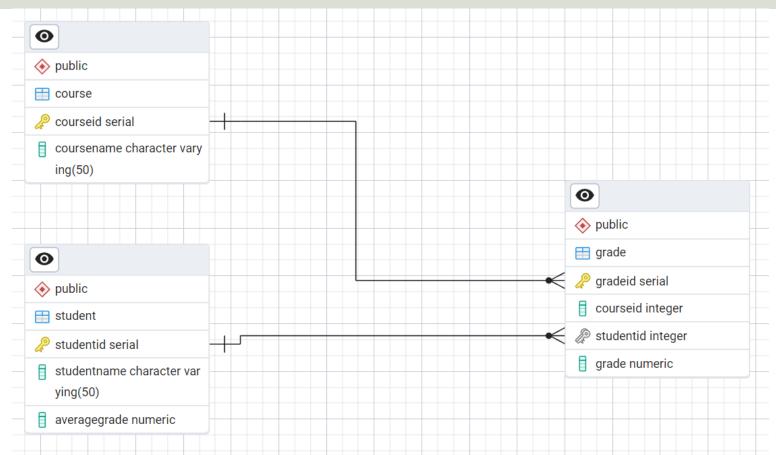


Entity Relationship Diagram (ERD)

#### **ERD** Tool in pgAdmin

🗣 pg	Admin 4				- 0 X	
ile C	bject Tools Edit View Window Help					
T.	Object Explorer 🛛 🕏 🎟	Count Rows	* ×		:	
	<ul> <li>&gt; S<sup>b</sup> Publications</li> <li>&gt; S<sup>b</sup> Publications</li> <li>&gt; Public</li> <li>&gt; Public</li></ul>	Create > Delete Delete (Cascade) Refresh Restore Backup	Jents/postgres@PostgreSQL 17			You can then
	<ul> <li>&gt; (i) Functions</li> <li>&gt; (ii) Materialized Views</li> <li>&gt; (ii) Procedures</li> <li>&gt; (i) Procedures</li> <li>&gt; 1.3 Sequences</li> <li>&gt; (iii) Tables (3)</li> </ul>	Truncate > View/Edit Data > Search Objects PSQL Tool Query Tool	ing(50)			generate a SQL Script:
	<ul> <li>&gt; ■ course</li> <li>&gt; ■ grade</li> <li>&gt; ■ student</li> <li>&gt; &gt; &gt; Types</li> <li>&gt; &gt; &gt; &gt; &gt; &gt; &gt; &gt; &lt;</li></ul>	Properties	studentid serial     studentid serial     studentiname character var     ying(50)     seragegrade numeric(10,     0)	Studentio digitati grade numeric( G	No limit	/choose If you find any bugs, Includi
∎ S	ervers > PostgreSQL 17 > Databases > 5	Students > Schemas > pu	blic > Tables > course			CHE 101001

## **ERD** Diagram



## SQL Table Script

```
CREATE TABLE student (
  studentid serial PRIMARY KEY,
  studentname varchar(50) NOT NULL,
  averagegrade numeric(10,0)
 );
```

```
CREATE TABLE course (
courseid serial PRIMARY KEY,
coursename varchar(50) NOT NULL
);
```

);

```
CREATE TABLE grade (
gradeid serial PRIMARY KEY,
courseid bigint NOT NULL REFERENCES course(courseid),
studentid bigint NOT NULL REFERENCES student(studentid),
grade numeric(10,0) NOT NULL
```

From the ERD Tool, we get a SQL script like this. We can also of course create this script from scratch.

#### **Create Tables from Script**

~ 5

#### We use the **Query Tool**:

	iory 1000.					
			Query History	2 5	Scratch Pad ×	2
<ul> <li>&gt; Im Aggregates</li> <li>&gt; Aggregates</li> <li>&gt;</li></ul>	Create Refresh Search Objects PSQL Tool Query Tool	2 3 4 5 6 7 7 9 9 10 11 12 13 13 14 15 5 16	<pre>studentname varchar(50) NOT NULL, averagegrade numeric(10,0) ); &lt; CREATE TABLE course ( courseid serial PRIMARY KEY, coursename varchar(50) NOT NULL ); &lt; CREATE TABLE grade ( gradeid serial PRIMARY KEY, courseid bigint NOT NULL REFERENCES course(courseid), studentid bigint NOT NULL REFERENCES student(atudentid),</pre>			
		Da	ata Output Messages Notifications			2
		CR	REATE TABLE			
		Qu	very returned successfully in 41 msec.			

🕸 pgAdmin 4

ୟ

5

File Object Tools Edit View Window Help

■ Welcome Students/postgres@PostgreSQL 17\* ×

Students/postgres@PostgreSQL 17

0

#### https://www.halvorsen.blog



#### PostgreSQL

# SQL Queries

#### Hans-Petter Halvorsen



#### We have the following main SQL Queries:

- INSERT
- SELECT
- UPDATE
- DELETE

CRUD operations, CRUD = Create (Insert), Read (Select), Update and Delete

### **INSERT Courses and Students**

Let's create some default data in our tables:

insert into course	<pre>(coursename) values ('Mathematics');</pre>
insert into course	<pre>(coursename) values ('Science');</pre>
insert into course	<pre>(coursename) values ('Programming');</pre>

<pre>insert into student (studentname</pre>	<pre>) values ('Elvis Presley');</pre>
<pre>insert into student (studentname</pre>	<pre>) values ('John Wayne');</pre>
<pre>insert into student (studentname</pre>	<pre>) values ('John Statham');</pre>

## Using the Query Tool

ф р	agAdmin 4		– 🗆 X		
File	Object Tools Edit View Window Help				
-1	Welcome Students/postgres@PostgreSQL 17* ×	👎 p	gAdmin 4	- 0	×
-	🔗 Students/postgres@PostgreSQL 17 🗸 🗣	File (	Object Tools Edit View Window Help		
<b>1</b> 2		۳.	Welcome Students/postgres@PostgreSQL 17* ×		
▶_	Query Query History	5	🔗 Students/postgres@PostgreSQL 17 🗸 🗣		0
et.			■ B × ✓ ▼ × Nolimit ▼ ■ ▶ ▶ ▼ B ■ × 3 % ⊞ ♥		
نی	<pre>insert into course (coursename) values ('Mathematics'); insert into course (coursename) values ('Science');</pre>	⊵	Query Query History	Scratch Pad ×	2
	<pre>3 insert into course (coursename) values ('Programming');</pre> Data Output Messages Notifications	Ŧ	<pre>1 insert into student (studentname) values ('Elvis Presley'); 2 insert into student (studentname) values ('John Wayne'); 3 insert into student (studentname) values ('John Statham');</pre>		
			Data Output Messages Notifications		2
	INSERT 0 1		INSERT 0 1		
	Query returned successfully in 49 msec.		Query returned successfully in 37 msec.		
\$	Total rows: Query complete 00:00:00.049	\$	Total rows: Query complete 00:00:00.037	CRLF Ln 3, Col	59

#### SELECT

select \* from course

#### select \* from student

🗣 p	Admin	4									ſ	🗣 р	gAdmin	14												- 0		×
File (	Object	Tools Edit	View	Window Help								File (	Object	Tools Edit Viev	w Window	/ Help												
T	Welco	ome 📑 s	tude	nts/postgres@Po	stgreS	QL 17*	×					•1	Welc	ome 📑 Stude	nts/post	gres@Postg	greSQL 1	7* ×										
() <u>}</u>	₿Ÿ	Students/	post	tgres@PostgreSQ	)L 17			`	- 5			8	₿Ÿ	Students/pos	tgres@P	ostgreSQL 1	17			~	<b>\$</b>							<b>O</b>
			*~	🝸 👻 No lin	nit 🖣	-	I 🕨 🖻	-		<b>%</b>	<b>-</b> ∼](			8 - /-	<b>T</b>	No limit	•			-		0°	i≡~	0				
>_	Query	y Query H	istor	у						×	* Sc	>_	Quer	y Query Histo	у								2	Scratch	Pad 🗙			2
4	1	select *	fr	om course								<del>6</del> 4	1	select * fr	om stud	lent												
	Data	Output M	essa	iges Notification	ns								Data	Output Messa	ages No	otifications												2
	=+	• · •	~		~	SQL	Show	ing rows	: 1 to 3 📝	Page No:	1		=+	■ < <sup>1</sup> <			∽ SQL		Showing	g rows: 1 t	o 3 🖍	Page N	o: 1		of 1	4	••	ÞI
		courseid [PK] intege	1	coursename character varying (5	50) 🖍									studentid [PK] integer	studentn characte	ame varying (50)	1											
	1		1	Mathematics									1	1	Elvis Pre	sley												
	2		2	Science									2	2	John Wa	yne												
	3		3	Programming									3	3	John Sta	tham												
\$	Tota	l rows: 3	Que	ery complete 00:0	0:00.06	58						\$	Tota	al rows: 3 Qu	ery comp	lete 00:00:0	0.069								CRLF	Ln 1,	Col 2	2

#### **Insert Grades**

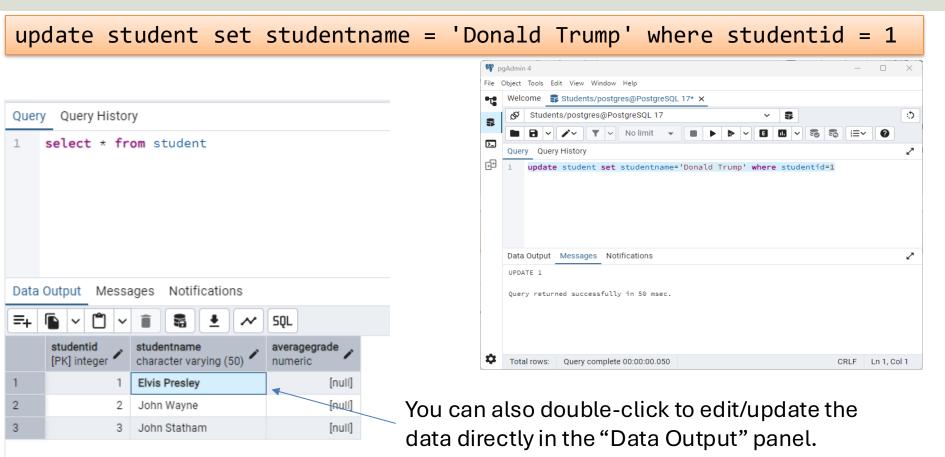
insert into grade (courseid, studentid, grade) values (1, 1, 2.5); insert into grade (courseid, studentid, grade) values (2, 1, 3.5); insert into grade (courseid, studentid, grade) values (3, 1, 1.5);

<pre>select * from grade</pre>		gradeid [PK] integer 🖍	integer	studentid integer	grade numeric 🖍
	1	1	1	1	2.5
	2	2	2	1	3.5
	3	3	3	1	1.5

Here student "Elvis Presley" (StudentId=1) gets the following grades in the different courses:

- "Mathematics" (Courseld=1) => Grade = 2.5
- "Science" (CourseId=2) => Grade = 3.5
- "Programming" (Courseld=3) => Grade = 1.5

#### UPDATE



#### DELETE

delete from tablename where column = ...

When using DELETE it is important to include a **where** statement, unless you want to delete all the data in that table.

Example:

delete from student where studentid = 3

This query will only delete the specific student where studentid =3

Or like this:

delete from student where studentname = 'John Statham'

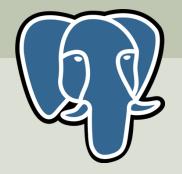
#### https://www.halvorsen.blog

PostgreSQL











## **Problem Description**

We need to use 3 different SQL queries to get information:

sel	lect * fr	om course							
	courseid [PK] integer	coursename character varying (50)							
1	1	Mathematics							
2	2	Science							
3	3	Programming							
<pre>select * from student</pre>									
se]		om student							
se]	Lect * fro studentid [PK] integer	om student studentname character varying (50)							
se]	studentid	studentname							
	studentid [PK] integer	studentname character varying (50)							
1	studentid [PK] integer	studentname character varying (50) 🖍 Elvis Presley							
1	studentid [PK] integer 1 2	studentname character varying (50) Elvis Presley John Wayne							

#### But we want to get information like this:

Data Output Messages Notifications ≡+ SQL studentname orade character varving (50) numeric character varying (50) Elvis Preslev Mathematics 2.5 Elvis Preslev Science 3.5 2 1.5 Elvis Presley Programming

But it is not possible because the information is stored in 3 different tables.

=> The solution is to create and use a **View**.

		gradeid [PK] integer	integer	studentid integer	grade numeric 🖍
	1	1	1	1	2.5
	2	2	2	1	3.5
e	3	3	3	1	1.5

#### Views

- A View is a "virtual" table that can contain data from <u>multiple</u> tables.
- Basically, a View is a SQL query that links 2 or more tables together making it possible to get data from these tables in a single query.

## **View Example**

CREATE OR REPLACE VIEW studentdata

AS

- SELECT
- student.studentName,
- course.courseName,
- grade.grade
- FROM student

INNER JOIN grade ON student.studentid = grade.studentid INNER JOIN course ON grade.courseid = course.courseid

In a View we typically use "INNER JOIN" to join information stored in different Tables.

#### **Create the View**

		•																
pg	Admin	4														_		> 2
• 0	bject	Tools	Edit View	Windov	v Help													e
	Welcome 🗣 Students/postgres@PostgreSQL 17* 🗙 🗣 Students/postgres 🗙																	
	₿ B	Students/postgres@PostgreSQL 17 🗸 🗣										1						
			· · ·	<b>T</b> ~	No limit	•		•	~	E		, 10 10	5	ΙΞ	~ 0			
ו	Query	Que	ery History								I							¥
<pre>2 AS 3 4 SELECT 5 student.studentName, 6 course.courseName, 7 grade.grade 8 FROM student 9 INNER JOIN grade ON student.studentid = grade.studentid 10 INNER JOIN course ON grade.courseid = course.courseid</pre>																		
	Data	Data Output Messages Notifications										ĸ						
		TE VIE / retu		cessful	ly in 40 m	sec.												
;	Tota	rows:	Query	comple	te 00:00:00.	040									CRLF	Lr	n 10, Co	ol 54

#### 👎 pgAdmin 4

> 📄 Types 🗸 🔟 Views (1) 🗸 回 studentdata Columns (3) f studentname coursename 🚦 grade Kules(1) > 🛟 Triggers

> 🏷 Subscriptions

qy	pgAdmir	n 4										
File	Object	Tools	Edit	View	Window	Help						
•1	Obje	ct Exp					<b>\$</b>	Ξ	Te Q			
			> 🤤	Langu	ages							
÷.			> 🖉	Public	ations							
_			~ 💖	Schem	nas (1)							
>_			$\sim$	📀 pub	lic							
ф				> <u> </u>	Aggregate	S						
ىن				> <mark>A</mark> ↓ 0	collations							
				> 🏠 🕻	omains							
				> 🕼 F	TS Confi	gurations						
				> 🕅 F	TS Dictio	naries						
				> Aa F	TS Parse	rs						
				> 💽 F	TS Temp	lates						
				> 🖭 F	oreign Ta	bles						
				> (jii) F	unctions							
				> 💽 N	/aterializ	ed Views						
				> 争(	perators							
				> {() F	rocedure	s						
				> 13 S	Sequence	S						
				v 🖻 T	ables (3)							
				> 🗄	course							
				> 8	grade							
				> 🗄	student							
				> (@ т	rigger Fu	nctions						

### Using the View

🗣 р	🕴 pgAdmin 4												
File (	Object	Tools Edit View Window	/ Help										
•	Welco	Welcome 🗣 Students/postgres 🗙 🕵 Students/postgres@PostgreSQL 17* 🗙											
85	₿ B	Students/postgres@PostgreSQL 17 🗸 🕏											
		8	No limit 👻 🔳			5 i=~ 0							
D	Query Query History									2			
Ð,	1 select * from studentdata												
										ſ			
										ſ			
										ĺ			
										ĺ			
	Data	Output Messages No	otifications							2			
	=+		🛓 🛃 🖍 SQL		Showing rows: 1 to 3	Page No:	1 of 1	14	44 >>	•			
		studentname character varying (50)	coursename character varying (50)	grade numeric									
	1	Elvis Presley	Mathematics	2.5						ſ			
	2	Elvis Presley	Science	3.5						ſ			
	3	Elvis Presley	Programming	1.5						ĺ			
										ſ			
										ſ			
\$	Tata	0.000	ata 00:00:00 000							1.06			
~	Tota	rows: 3 Query comp	lete 00:00:00.080				CRL	r l	Ln 1, Co	26			

### **Views Queries Examples**

You can use Views almost as you use Tables. Here are some basic examples:

select \* from studentdata

• •

select coursename, grade from studentdata where studentname = 'Elvis Presley'

select studentname, grade from studentdata where coursename = 'Mathematics'

select avg(grade) as avgrade from studentdata where studentname = 'Elvis Presley'

https://www.halvorsen.blog



#### PostgreSQL

## **Stored Procedures**

#### Hans-Petter Halvorsen



### **Problem Description**

To create/insert Grades we need to create and execute queries like this:

insert into GRADE (CourseId, StudentId, Grade) values (1, 1, 2.5)
insert into GRADE (CourseId, StudentId, Grade) values (2, 1, 3.5)
insert into GRADE (CourseId, StudentId, Grade) values (3, 1, 1.5)

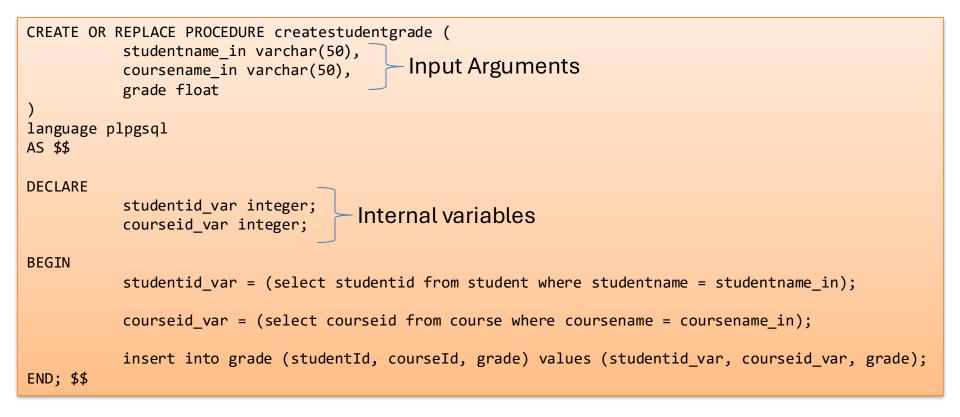
The "drawback" is that we need to remember the Courselds and the StudentIds, typically we only remember and want to use their names.

=> The solution is to create and use a **Stored Procedure**.

### **Stored Procedures**

- A Stored Procedure is very similar as a Method/Function in C# or Python - it is a piece of code with SQL commands that do a specific task – and you can reuse it.
- A Stored Procedure can have Input Arguments and Return values (just like a Method/Function).
- It also adds a layer of security, because you can do a lot of harm by creating the wrong queries. In that way you can create a set of Stored Procedures that is well implemented and tested properly.
- Stored Procedures can also prevent "SQL Injection" used by "hackers", etc.

### **Stored Procedure Example**



#### **Create the Stored Procedure**

👎 pgAdmin 4 — 🗆 🗙			👎 pgAdmi	n 4
File	Object Tools Edit View Window Help	File Object Tools Edit View Window Help		
•18	Welcome Students/postgres@PostgreSQL 17* ×		•te Obje	ect Explorer 🗣 🎟 🚡 🕰
<b>**</b>	Students/postgres@PostgreSQL 17       >         Image: Construction of the state o		\$ \$	<ul> <li>&gt; erreign Data wrappers</li> <li>&gt; clanguages</li> <li>&gt; clanguages</li> <li>&gt; clanguages</li> </ul>
Ŧ	<pre>1 * CREATE OR REPLACE PROCEDURE createstudentgrade ( 2 studentname varchar(50), 3 coursename varchar(50), 4 grade float 5 ) 6 language plpgsql 7 AS \$\$ 8 9 DECLARE 10 studentid integer; 11 courseid integer; 12 13 * BEGIN 14 studentid = (select studentid from student where studentname 15 courseid = (select courseid from course where coursename = courseid = (select courseid from course where coursename = courseid = (studentId, courseId, grade) values (studentId) 19 END; \$\$</pre>	<pre>studentname); oursename);</pre>		<ul> <li>Schemas(1)</li> <li>public</li> <li>Aggregates</li> <li>Collations</li> <li>Domains</li> <li>FTS Configurations</li> <li>FTS Dictionaries</li> <li>Aa FTS Parsers</li> <li>FTS Templates</li> <li>FTS Templates</li> <li>FF Foreign Tables</li> <li>Foreign Tables</li> <li>Foreign Tables</li> <li>Materialized Views</li> <li>Operators</li> <li>(1) Procedures(1)</li> <li>(1) createstudentgrade(IN studentnam)</li> <li>1.3 Sequences</li> </ul>
\$	Data Output       Messages       Notifications         CREATE PROCEDURE       Query returned successfully in 42 msec.         Total rows:       Query complete 00:00:00.042       CRLF       Ln 12			<ul> <li>✓ Image: Contract of the contract of</li></ul>

# Using the Stored Procedure

insert into GRADE (CourseId, StudentId, Grade) values (1, 1, 2.5)
insert into GRADE (CourseId, StudentId, Grade) values (2, 1, 3.5)

insert into GRADE (CourseId, StudentId, Grade) values (3, 1, 1.5)



call createstudentgrade('John Wayne', 'Mathematics', 1.0)

call createstudentgrade('John Wayne', 'Science', 2.0)

call createstudentgrade('John Wayne', 'Programming', 3.0)

# **Using the Stored Procedure**

Ф.							
File	Object Tools Edit View Window Help						
•	Welcome 🔓 createstudentgrade 🗙 Students/postgres@PostgreSQL 17* 🗙 🕏 Students/postgres 🗡	~					
07	Students/postgres@PostgreSQL 17 ~ \$	ð					
▶_	Query Query History	2					
Ð	<pre>call createstudentgrade('John Wayne', 'Mathematics', 1.0) 2</pre>						
	Data Output Messages Notifications	2					
	CALL						
	Query returned successfully in 37 msec.						
\$	Total rows: Query complete 00:00:00.037 CRLF Ln 2, Col	1					

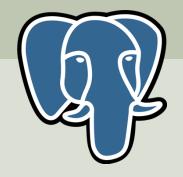
#### https://www.halvorsen.blog

PostgreSQL



#### Hans-Petter Halvorsen

Table of Contents



## **Problem Description**

Query History Ouerv

≡+

select \* from student

call createstudentgrade('John Wayne', 'Mathematics', 1.0) call createstudentgrade('John Wayne', 'Science', 2.0)

call createstudentgrade('John Wayne', 'Programming', 3.0)

Messages Notifications Data Output SQL studentid studentname averagegrade [PK] integer character varying (50) numeric (10) Elvis Presley [null] [null] John Wayne John Statham [null]

We want to automatically update the "averagegrade" for each student when inserting, updating or deleting Grades for a specific Student in a specific Course. => The solution is to create and use a **Trigger.** 

# Triggers

- A Trigger is executed when you insert, update or delete data in a Table specified in the Trigger.
- A trigger is a stored procedure in a database that automatically invokes whenever a special event in the database occurs.
- A Trigger is attached to a specific Table.
- You can use a Trigger to change data in the same table or in other tables.
- We typically first make a Trigger Function then we make the Trigger itself that is attached to a specific Table, this Trigger then basically executes the Trigger Function.

# **Trigger Function Example**

CREATE OR REPLACE FUNCTION calcavggrade\_function() RETURNS TRIGGER AS

```
$$
```

```
DECLARE
studentid_var int;
averagegrade_var float;
```

Note! "NEW" is a temporarily table containing the latest inserted data, and it is very handy to use inside a Trigger Function.

```
BEGIN
studentid var := NEW.studentid;
```

```
averagegrade_var = (select AVG(grade) from grade where studentid = studentid_var);
```

```
update student set averagegrade = averagegrade_var where studentid = @studentid_var;
```

```
RETURN NULL;
```

```
END;
$$
LANGUAGE 'plpgsql';
```

```
ቑ pgAdmin 4
                                                                                               ×
                                                                                          _
File Object Tools Edit View Window Help
    Welcome 📔 calcavggrade_function - Trigger Function.sql 🗙 🖺 calcavggrade_trigg... 🗙 🕏 Students/postgres... 🗙 🕏 Student 🗸
•
                                                       ~ $
     Students/postgres@PostgreSQL 17
                                                                                                    ୍
5
        🔒 🗸 🥕 🝸 🗸 No limit 👻
                                        ≻₋
    Query Query History
                                                                                                    2
фÐ
     1 • CREATE OR REPLACE FUNCTION calcavggrade_function()
         RETURNS TRIGGER AS
     2
     3
        $$
     4
     5
     6
         DECLARE
            studentid var int;
     7
     8
            averagegrade_var float;
     9
    10 V BEGIN
             studentid_var := NEW.studentid;
    11
    12
             averagegrade_var = (select AVG(grade) from grade where studentid = studentid_var);
    13
    14
    15
            update student set averagegrade = averagegrade_var where studentid = @studentid_var;
    16
    17
             RETURN NULL;
    18
         END;
    19
    20
        $$
    21 LANGUAGE 'plpgsql';
    Data Output Messages Notifications
                                                                                                    ~
    CREATE FUNCTION
    Query returned successfully in 54 msec.
$
    Total rows: Query complete 00:00:00.054
                                                                                     CRLF Ln 18, Col 1
```

# Trigger Example

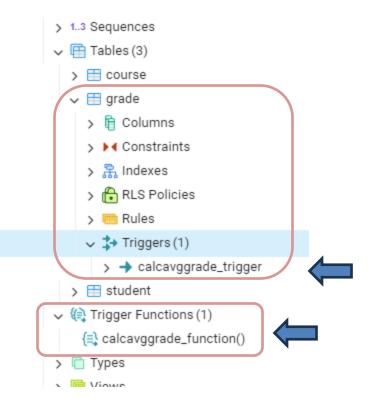
The Trigger basically just call/execute the Trigger Function calcavggrade\_function()

#### CREATE OR REPLACE TRIGGER calcavggrade\_trigger AFTER INSERT ON grade FOR EACH ROW EXECUTE FUNCTION calcavggrade function(); repairing the object Toole

You need to specify which Table the Trigger shall be attached to.

¶¶ p	Admin 4 – 🗆	×				
File	Ibject Tools Edit View Window Help					
•1	Welcome 🖹 calcavggrade_funct × 📑 calcavggrade_trigger - Trigger.sql × 🕏 Students/postgres ×	· ~				
63	🔗 Students/postgres@PostgreSQL 17 🗸 🕏	ð				
•						
2	Query Query History					
Ð	<pre>1 ~ CREATE OR REPLACE TRIGGER calcavggrade_trigger 2 AFTER INSERT ON grade 3 FOR EACH ROW 4 EXECUTE FUNCTION calcavggrade_function();</pre>					
	Data Output Messages Notifications	2				
	CREATE TRIGGER					
_	Query returned successfully in 44 msec.					
\$	Total rows: Query complete 00:00:00.044 CRLF Ln 4, Co	42				

# **Trigger Function + Trigger**



#### **Insert Grades**

We use the Stored Procedure created earlier:

call createstudentgrade('John Statham', 'Mathematics', 2.0)

call createstudentgrade('John Statham', 'Science', 3.0)

call createstudentgrade('John Statham', Programming', 1.0)

```
Query History
Query
     select * from student
Data Output Messages
                          Notifications
                           5
≡+
         \sim
                                 +
                                      \sim
                                            SOL
                  \sim
      studentid
                     studentname
                                            averagegrade
      [PK] integer 🖍
                     character varying (50) 🖍
                                            numeric (10)
                     Elvis Preslev
                                                      [null]
2
                     John Wavne
                                                      [null]
3
                     John Statham
```

## Hans-Petter Halvorsen

**University of South-Eastern Norway** 

www.usn.no

E-mail: <u>hans.p.halvorsen@usn.no</u> Web: <u>https://www.halvorsen.blog</u>

