

Veritabanı Yönetim Sistemleri (335)

Dr. Öğr. Üyesi Ahmet Arif AYDIN

L20-

PostgreSQL: View, Trigger

GÜZ -2022

Agenda

- View
- Trigger

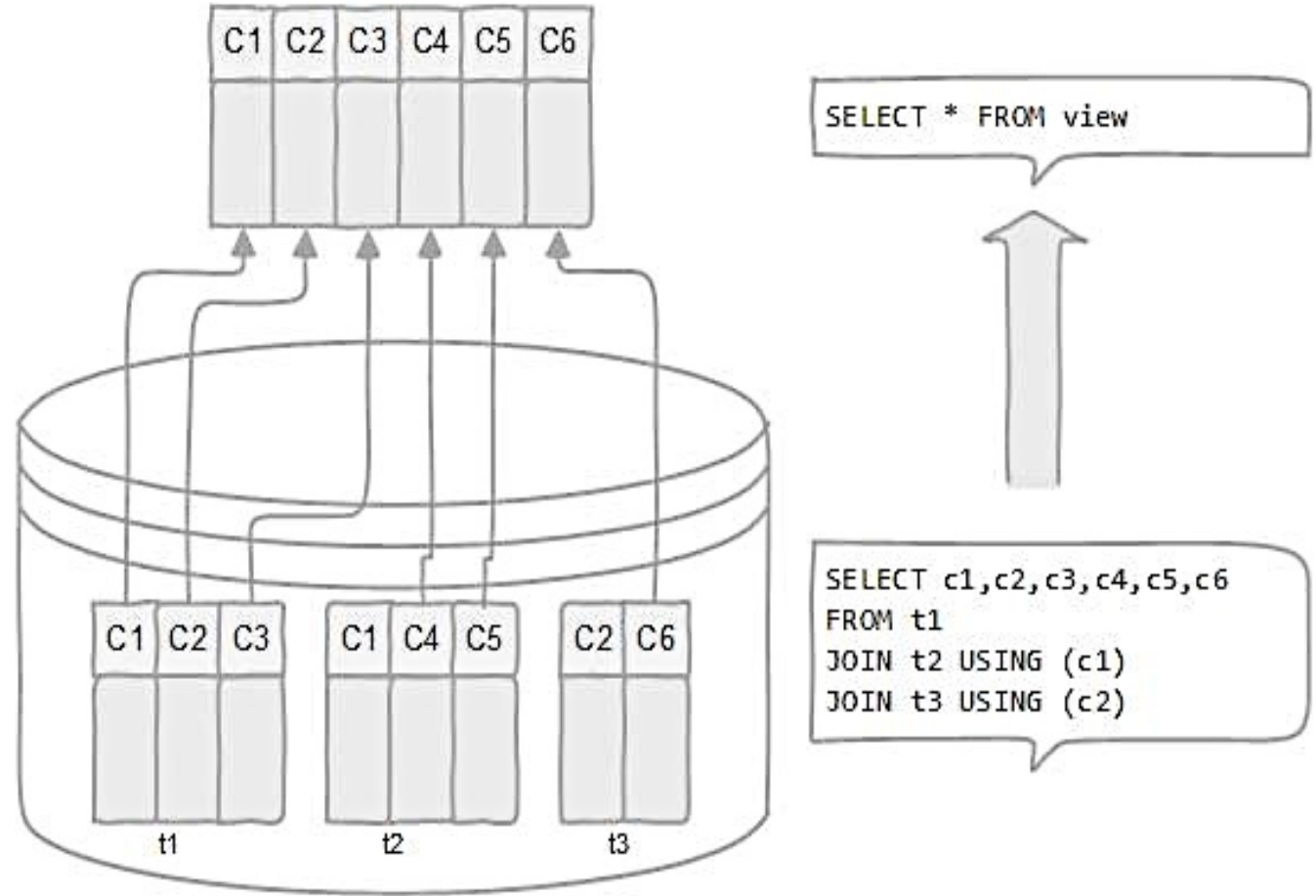
View

- Veritabanında tablolarda kayıtlı olan veriyi farklı görünüş ile sunmayı sağlar .
- View isimlendirilen bir soru gibi düşünülebilir.
- Sürekli kullanılan karmaşık sorgular için view oluşturulabilir.

<https://www.guru99.com/postgresql-view.html>

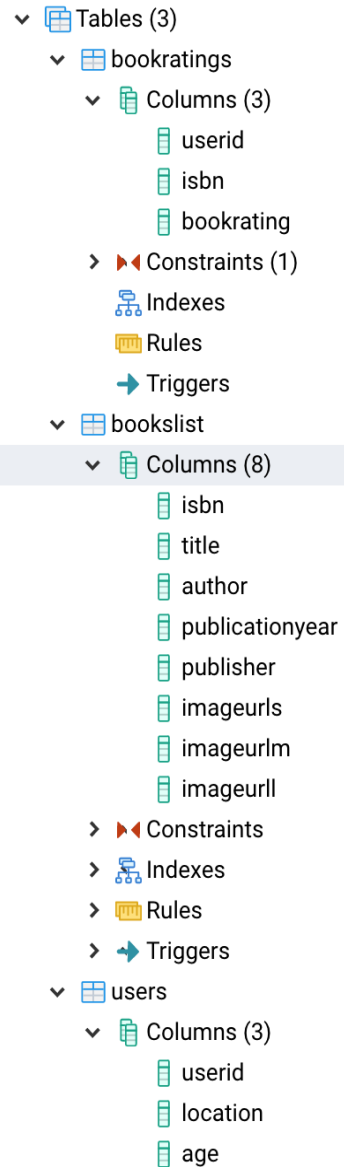
View

- Veritabanında kayıtlı olan bir sorgu (query) nesnesidir.
- Veri kaydetmez
- Ana tabloları görme yetkisi olmayan kullanıcılar için *verinin sadece istenilen kısımları* sunulabilir.



PostgreSQL: Create View

<https://www.postgresqltutorial.com/postgresql-views/>

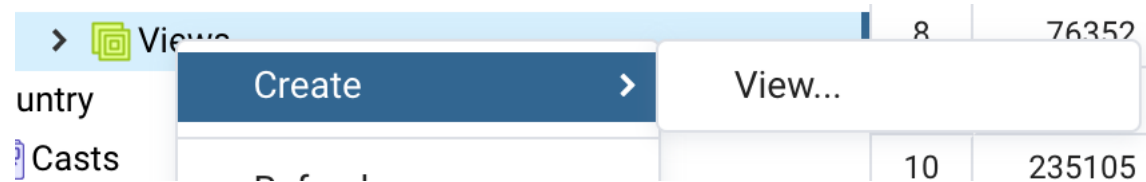


A tree view of a PostgreSQL database schema. The root is 'Tables (3)', which branches into 'bookratings', 'bookslist', and 'users'. 'bookratings' has columns 'userid', 'isbn', and 'bookrating', and a constraint. 'bookslist' has 8 columns: 'isbn', 'title', 'author', 'publicationyear', 'publisher', 'imageurls', 'imageurlm', and 'imageurll', and constraints, indexes, rules, and triggers. 'users' has columns 'userid', 'location', and 'age', and constraints.

Tables (3)
bookratings
Columns (3)
userid
isbn
bookrating
Constraints (1)
Indexes
Rules
Triggers
bookslist
Columns (8)
isbn
title
author
publicationyear
publisher
imageurls
imageurlm
imageurll
Constraints
Indexes
Rules
Triggers
users
Columns (3)
userid
location
age

bookratings	17,178	postgres	pg_default	1,007,155
bookslist	17,190	postgres	pg_default	265,575
users	17,167	postgres	pg_default	269,781

- *topten_users* view
- Listelenen kitaplar için en fazla değerlendirme (rating) yapan kullanıcılardan ilk 10'unun kullanıcı adı (userid), lokasyon ve rating sayısını ekrana yazdıran view'i oluşturunuz.



```
CREATE OR REPLACE VIEW public.topten_users AS
SELECT users.userid,
       users.location,
       count(users.userid) AS ratingcount
FROM   users
       JOIN bookratings ON users.userid = bookratings.userid
GROUP BY users.userid
ORDER BY (count(users.userid)) DESC
LIMIT 10;
```

```
ALTER TABLE public.topten_users
  OWNER TO postgres;
```

- *View ile verinin sadece istenilen kısımları sunulabilir.*
- PostgreSQL'in bütün komutları kullanılabilir.

PostgreSQL: Create View

<https://www.postgresqltutorial.com/postgresql-views/>

- books
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
- Schemas (1)
 - public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions (1)
 - sent_rating_count(id integer)
 - Materialized Views
 - Sequences
 - Tables (3)
 - bookratings
 - bookslst
 - users
 - Trigger Functions
 - Types
 - Views (1)
 - topten_users**
 - Columns
 - Rules (1)
 - Triggers

```
1 -- View: public.topten_users
2
3 -- DROP VIEW public.topten_users;
4
5 CREATE OR REPLACE VIEW public.topten_users AS
6 SELECT users.userid,
7        users.location,
8        count(users.userid) AS ratingcount
9 FROM users
10 JOIN bookratings ON users.userid = bookratings.userid
11 GROUP BY users.userid
12 ORDER BY (count(users.userid)) DESC
13 LIMIT 10;
14
15 ALTER TABLE public.topten_users
16 OWNER TO postgres;
17
18 |
```

SELECT * FROM topten_users;

	userid integer	location character varying (250)	ratingcount bigint
1	11676	n/a, n/a, n/a	13599
2	198711	little canada, minnesota, usa	7550
3	153662	ft. stewart, georgia, usa	6109
4	98391	morrow, georgia, usa	5891
5	35859	duluth, minnesota, usa	5850
6	212898	la ronge, saskatchewan, ca...	4785
7	278418	omaha, nebraska, usa	4533
8	76352	olympia, washington, usa	3367
9	110973	wiley ford, west virginia, usa	3100
10	235105	st louis, missouri, usa	3067

- books
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Schemas (1)
 - public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions (1)
 - sent_rating_count(id integer)
 - Materialized Views
 - Sequences
 - Tables (3)
 - bookratings
 - bookslist
 - users
 - Trigger Functions
 - Types
 - Views (1)
 - topten_users
 - Columns
 - Rules (1)
 - Triggers

```
1 -- View: public.topten_users
2
3 -- DROP VIEW public.topten_users;
4
5 CREATE OR REPLACE VIEW public.topten_users AS
6 SELECT users.userid,
7        users.location,
8        count(users.userid) AS ratingcount
9 FROM users
10      JOIN bookratings ON users.userid = bookratings.userid
11 GROUP BY users.userid
12 ORDER BY (count(users.userid)) DESC
13 LIMIT 10;
14
15 ALTER TABLE public.topten_users
16     OWNER TO postgres;
17
18
```

**ALTER VIEW topten_users
RENAME TO top_ten_users;**

**DROP VIEW IF EXISTS
top_ten_users;**

Materyalized View

- View den farklı olarak veriyi kaydeder
- Veriye hızlı bir biçimde erişim için önemlidir.

```
CREATE MATERIALIZED VIEW test  
AS  
query  
WITH [NO] DATA;
```

```
REFRESH MATERIALIZED VIEW test;
```

```
REFRESH MATERIALIZED VIEW  
CONCURRENTLY test;
```

PostgreSQL: PL/pgSQL

PL/pgSQL

- Prosedürel bir programlama dilidir.
- Kullanıcı tanımlı fonksiyonların (user-defined functions)
- Stored procedure ve Trigger tanımlanmasına imkan sağlar

Trigger

- Bir tabloda INSERT, UPDATE, DELETE ve TRUNCATE işlemleri gerçekleştirildiğinde otomatik olarak çalışan bir fonksiyondur.
- Bir tablo ile ilişkilendirilen kullanıcı tanımlı bir fonksiyondur.
- Önce trigger fonksiyonu oluşturulur sonrasında tablo ile ilişkilendirilir.

Trigger

- Row triggers: Gerçekleşen işlemin etkilediği satır sayısında çalışır
- Statement triggers (1 defa çalışır)

PostgreSQL: Trigger

1

```
CREATE FUNCTION trigger_function()  
  RETURNS TRIGGER  
  LANGUAGE PLPGSQL  
AS $$  
BEGIN  
  -- trigger logic  
END;  
$$
```

2

```
CREATE TRIGGER trigger_name  
  { BEFORE | AFTER } { event }  
  
ON table_name  
  
[FOR [EACH] { ROW | STATEMENT }]  
  
EXECUTE PROCEDURE trigger_function
```

PostgreSQL: Trigger

1

```
CREATE FUNCTION trigger_function()
```

```
RETURNS TRIGGER
```

```
LANGUAGE PLPGSQL
```

```
AS $$
```

```
BEGIN
```

```
-- trigger logic
```

```
END;
```

```
$$
```

2

```
CREATE TRIGGER trigger_name
```

Trigger'in ne zaman
Çalışacağı tanımlanır

```
{ BEFORE | AFTER }
```

```
{ event }
```

INSERT, UPDATE
DELETE, TRUNCATE

```
ON table_name
```

FOR EACH ROW

[FOR [EACH] { ROW | STATEMENT }]

FOR EACH STATEMENT

```
EXECUTE PROCEDURE trigger_function
```

Trigger

- Örneğin
 - Müsteri tablosununa yeni bir kişi eklendiğinde müşteri ile ilgili veriler diğer gerekli tablolara otomatik olarak girilebilir.
 - Veritabanı içerisinde yapılan değişikliklerin kaydedildiği başka **güncellemegeçmişi** isimli bir tablo oluşturulabilir.

PostgreSQL: Trigger

course

	courseid [PK] character varying	coursename character varying	credit integer
1	BMG1	Bilg Muh Giris 1	5
2	PL315	PROGRAMLAMA DIL...	3
3	DBMS335	VERITABANI YONETI...	5

course_updates

	courseid character varying (10)	oldcredit character varying	update_time date
1	DBMS335	6	2021-11-29
2	DBMS335	7	2021-11-29

PostgreSQL: Trigger

course

	courseid [PK] character varying	coursename character varying	credit integer
1	BMG1	Bilg Muh Giris 1	5
2	PL315	PROGRAMLAMA DIL...	3
3	DBMS335	VERITABANI YONETI...	5

course_updates

	courseid character varying (10)	oldcredit character varying	update_time date
1	DBMS335	6	2021-11-29
2	DBMS335	7	2021-11-29

```
CREATE OR REPLACE FUNCTION course_changes2()  
RETURNS TRIGGER  
LANGUAGE PLPGSQL  
AS  
$$  
BEGIN  
    INSERT INTO course_updates(courseid,oldcredit, update_time)  
    VALUES(OLD.courseid,OLD.credit, CURRENT_DATE);  
    RETURN NEW;  
END;  
$$
```

PostgreSQL: Trigger

course

	courseid [PK] character varying	coursename character varying	credit integer
1	BMG1	Bilg Muh Giris 1	5
2	PL315	PROGRAMLAMA DIL...	3
3	DBMS335	VERITABANI YONETI...	5

course_updates

	courseid character varying (10)	oldcredit character varying	update_time date
1	DBMS335	6	2021-11-29
2	DBMS335	7	2021-11-29

```
CREATE OR REPLACE FUNCTION course_changes2()  
RETURNS TRIGGER  
LANGUAGE PLPGSQL  
AS  
$$  
BEGIN  
    INSERT INTO course_updates(courseid,oldcredit, update_time)  
    VALUES(OLD.courseid,OLD.credit, CURRENT_DATE);  
    RETURN NEW;  
END;  
$$
```

```
CREATE TRIGGER credit_changes  
BEFORE UPDATE  
ON course  
FOR EACH ROW  
EXECUTE PROCEDURE course_changes2();
```

PostgreSQL: Trigger

course

	courseid [PK] character varying	coursename character varying	credit integer
1	BMG1	Bilg Muh Giris 1	5
2	PL315	PROGRAMLAMA DIL...	3
3	DBMS335	VERITABANI YONETI...	5

course_updates

	courseid character varying (10)	oldcredit character varying	update_time date
1	DBMS335	6	2021-11-29
2	DBMS335	7	2021-11-29

```
CREATE OR REPLACE FUNCTION course_changes2()  
RETURNS TRIGGER  
LANGUAGE PLPGSQL  
AS  
$$  
BEGIN  
    INSERT INTO course_updates(courseid,oldcredit, update_time)  
    VALUES(OLD.courseid,OLD.credit, CURRENT_DATE);  
    RETURN NEW;  
END;  
$$
```

```
CREATE TRIGGER credit_changes  
BEFORE UPDATE  
ON course  
FOR EACH ROW  
EXECUTE PROCEDURE course_changes2();
```

```
UPDATE course  
SET credit=5  
where courseid='DBMS335';
```

Dinlediğiniz için
Teşekkürler...
İyi çalışmalar...