howest
hogeschool

Syllabus

Databases

Mileto Di Marco
Kristien Roels
Bart Soete

Inhoudsopgave

H6 – Views maken m.b.v. SQL

1 Inleiding

Een view kunnen we omschrijven als een opgeslagen SELECT-statement. In deze context moeten we een view zien als een *virtuele* tabel. Voor de gebruiker, zij het programmeur of databasebeheerder, presenteert een view zichzelf als een gewone tabel, compleet met kolomnamen en data. Een view bevat echter geen gegevens, de *resultaatset* van de uitgevoerde query wordt niet permanent opgeslagen maar telkens dynamisch gegenereerd. Een view is als het ware een filter naar de eindgebruiker toe en kan bestaan uit verschillende tabellen maar ook uit de combinatie van andere views.

Simpel uitgedrukt is een view een query (SELECT) die opgeslagen wordt op de databaseserver. Een opmerkelijk gegeven hierbij is dat bewerkingen die op een view worden uitgevoerd vertaald worden naar bewerkingen op de onderliggende tabel(len). We kunnen bijgevolg perfect een UPDATE statement uitvoeren op een view, op voorwaarde dat de view geen complexe bewerkingen bevat (zie Data wijzigen uit een view)

Waarom views gebruiken?

- Views verbergen de complexiteit van query's voor de gebruiker: een view kan een complexe join zijn van tabellen of andere views. De gebruiker ondervraagt de view zonder te weten hoe de join geschreven is. De view verbergt de complexiteit van de join(s) en biedt zinvolle informatie aan.
- Views geven aan verschillende gebruikers de gewenste data. De verschillende gebruikers krijgen elk een eigen 'zicht' op dezelfde data.
- Views zorgen voor logische dataonafhankelijkheid. Als het conceptueel datamodel aangepast wordt, dan kan dat verborgen blijven voor de gebruiker wanneer die op views werkt.
- Views zorgen voor gegevensbeveiliging. Een view toont aan de gebruiker enkel de data die hij nodig heeft.
- Views worden vaak gebruikt als basis voor ad hoc query's (gelegenheidsquery's) en rapportering.

De voorbeelddatabank die we in dit hoofdstuk gebruiken is de **bibliotheek-databank**.

2 Views, stored procedures of user-defined functions?

Op het eerste zicht kan er verwarring ontstaan tussen views, stored procedures (hoofdstuk 9) en user-defined functions (hoofdstuk 10), maar de onderstaande tabel maakt één en ander duidelijk.

Views	Stored Procedures	User Defined Functions
• Kan geen parameters ontvangen • Kan gebruikt worden als onderdeel van een andere query • Kan enkel 1 SELECT statement bevatten • Kan geen wijzigingen op de tabelstructuur uitvoeren • Kan wel onderdeel zijn van een INSERT, UPDATE of DELETE statement	• Kan parameters ontvangen • Kan geen onderdeel uitmaken van een andere query • Kan verschillende statements bevatten, alsook LOOPS, IF..ELSE • Kan de tabelstructuur wijzigen • Kan geen onderdeel zijn van een INSERT, UPDATE of DELETE statement	• Kan parameters ontvangen • Kan gebruikt worden als onderdeel van een andere query • Kan verschillende statements bevatten, alsook LOOPS, IF..ELSE • Kan de tabelstructuur niet wijzigen • Kan geen onderdeel zijn van een INSERT, UPDATE of DELETE statement

3 Een eenvoudige view aanmaken

De basissyntax van een view ziet er als volgt uit:

CREATE [OR ALTER] VIEW viewnaam [(kolomnaam, …)]
AS
SELECT-statement
[WITH CHECK OPTION]

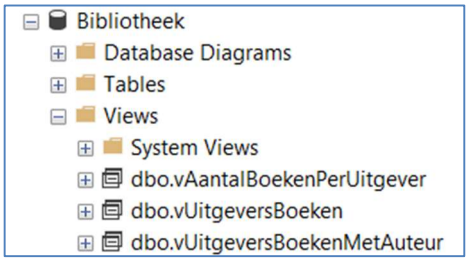
- Indien geen kolomnamen worden opgegeven, worden de kolomnamen uit het SELECT-statement overgenomen. Kolomnamen zijn verplicht als er in het SELECT-statement een onbenoemde kolom is (een expressie zonder alias).
- Het SELECT-statement definieert de view. In dit SELECT-statement kunnen meerdere tabellen gebruikt worden én ook andere views kunnen gebruikt worden.
- De ORDER BY-clausule is niet toegelaten in de definitie van de view, tenzij TOP ook gespecifieerd is.

Maak een view die de uitgevers teruggeeft die boeken hebben uitgegeven tussen 1980 en 1990. Geef de uitgevergegevens en de boekgegevens.
create or alter view vUitgeversBoeken as select u.uitgeverid, u.uitgever, b.boekid, b.titel, b.jaar from uitgevers u join boeken b on u.uitgeverid = b.uitgeverid where jaar between 1980 and 1990

Maak een view met het aantal boeken dat elke uitgever heeft uitgegeven.
create or alter view vAantalBoekenPerUitgever as select u.uitgeverid, count(boekid) as aantalBoeken from uitgevers u left join boeken b on u.uitgeverid = b.uitgeverid group by u.uitgeverid

Maak een view met de uitgevers en hun eventuele boeken met de auteur.
create or alter view vUitgeversBoekenMetAuteur as select u.uitgeverid, u.uitgever, b.boekid, b.titel, a.familienaam, a.voornaam from uitgevers u left join boeken b on u.uitgeverid = b.uitgeverid left join auteurs a on b.auteurid = a.auteurid

De aangemaakte views worden bewaard in de map Views:



4 Een view wijzigen

Een VIEW wijzigen kan ook het ALTER VIEW statement:

ALTER VIEW viewnaam [(kolomnaam, …)]
AS
SELECT-statement
[WITH CHECK OPTION]

5 Een view bevragen

Een view kunnen we oproepen door middel van een eenvoudig SELECT statement. Merk op dat we de view kunnen behandelen als ware het een tabel in onze databank.

Geef de uitgevers en hun eventuele boeken met de auteur. Sorteer op naam van de uitgever.
select * from vUitgeversBoekenMetAuteur order by uitgever
Zie hoe de view gebruikt wordt als een tabel; bij gebruik van de view wordt de onderliggende query uitgevoerd. We hebben dus 1x het denkwerk moeten doen om een complexe query te schrijven. We hebben deze verpakt en bewaard in de databank en kunnen deze vanaf nu gebruiken. Dus: views verbergen de complexiteit van query's voor de gebruiker.

Dit betekent dat we een view ook kunnen opnemen in complexere SELECT statements. Bekijk onderstaande voorbeelden.

Welke uitgever(s) hebben het hoogst aantal boeken uitgegeven?
select top(1) with ties * from vAantalBoekenPerUitgever order by aantalBoeken desc
Ook hier: we hebben 1x het denkwerk moeten doen om de view te schrijven. Daarna kan de view, die de complexiteit van de query voor de gebruiker verbergt, gebruikt worden.

Geef nu ook de naam van uitgevers die we zonet gevonden hebben (de uitgevers met het hoogst aantal boeken).
select top(1) with ties * from vAantalBoekenPerUitgever join uitgevers on vAantalBoekenPerUitgever.uitgeverid = uitgevers.uitgeverid order by aantalBoeken desc
Je kan een view dus echt gebruiken als een tabel en een join maken van de view met een andere tabel of view. Nogmaals: het interessante is dat de moeilijke query met GROUP BY als het ware verborgen is voor de gebruiker; deze moeilijke query zit verpakt in de view en de view is bewaard in de databank.

6 Een view verwijderen

Om een view te verwijderen kunnen we gebruik maken van het DROP VIEW statement:

Verwijder de view vAantalBoekenPerUitgever
Drop view if exists vAantalBoekenPerUitgever

7 Data wijzigen uit een view

Views maken het mogelijk om, indien ze eenvoudig opgebouwd zijn, de weergegeven data te wijzigen door middel van een UPDATE statements. Met eenvoudig opgebouwd bedoelen we onderstaande voorwaarden:

- Een view mag geen aggregaatfuncties of een GROUP BY-clausule bevatten.
- Een view mag geen DISTINCT kolom bevatten.
- Een afgeleide (berekende) kolom is ook niet mogelijk.

Indien aan vorige voorwaarden is voldaan dan kunnen we de data uit een view onderwerpen aan een UPDATE statement.

Werk op de “eenvoudige” view vUitgeversBoekenMetAuteur. Wijzig de titel van het boek met boekid 12 in 'MSSQL Fundamentals'
update vUitgeversBoekenMetAuteur set titel = 'MSSQL Fundamentals' where boekid = 12

Merk op:

Wanneer de view aangemaakt wordt met WITH CHECK OPTION dan moeten de voorwaarden uit de WHERE clausule gerespecteerd worden bij elke update.

create or alter view vUitgeversBoeken as select u.uitgeverid, u.uitgever, b.boekid, b.titel, b.jaar from uitgevers u left join boeken b on u.uitgeverid = b.uitgeverid where jaar between 1980 and 1990 WITH CHECK OPTION
Breng een wijziging aan bij boek 12: zet het jaar op 1991.
update vUitgeversBoeken set jaar = 1991 where boekid = 12
Dit lukt niet omdat de wijziging aan de gegevens uit de view niet meer voldoen aan de voorwaarden van de view.
Dus: als de maker van de view enkel updates wil toelaten die de voorwaarden van de view respecteren, dan moet de maker van de view de WITH CHECK OPTION gebruiken.

8 Data verwijderen uit een view

Om data te verwijderen uit een view moeten we rekening houden met 1 belangrijke restrictie. Het is niet mogelijk om het DELETE statement te gebruiken op een view die meerdere tabellen refereert. Enkel op views die opgebouwd zijn uit één enkele tabel kunnen we het DELETE statement uitvoeren.

Maak een view op basis van één tabel, bijvoorbeeld een view die enkel de titel en het jaar weergeeft van de boeken.
create or alter view vBoeken as select titel, jaar from boeken

Verwijder het boek met de titel ‘Kaas’
delete from vBoeken where titel ='Kaas'

9 Data INSERT in een view

Ook wat betreft data inserts gelden dezelfde restrictie, enkel wanneer een view opgebouwd werd uit één enkele tabel kunnen we het INSERT statement op die view toepassen. Om dit te demonstreren maken we een (eigenlijk nutteloze) view van de tabel Uitgevers.

```
create or alter view vUitgevers
as
select * from uitgevers
```

Op deze view kunnen we nu een INSERT uitvoeren.

```
insert into vUitgevers (uitgeverid, uitgever)
values (88, 'Prometheus')
```