Trapsgewijze parameters in gepagineerde rapporten gebruiken

28-1-2020 • 12 minutes to read • Edit Online

Dit artikel is bedoeld voor u wanneer u als auteur gepagineerde rapporten in Power BI gaat ontwerpen. Het bevat scenario's voor het ontwerpen van trapsgewijze parameters. Trapsgewijze parameters zijn rapportparameters met afhankelijkheden. Wanneer een rapportgebruiker een parameterwaarde (of waarden) selecteert, wordt deze gebruikt om de beschikbare waarden voor een andere parameter in te stellen.

> **NOTE**
> Een inleiding tot trapsgewijze parameters en hoe u deze kunt configureren, wordt niet behandeld in dit artikel. Als u niet volledig bekend bent met trapsgewijze parameters, raden we u aan eerst trapsgewijze parameters toe te voegen aan een rapport (Power BI Report Builder en SSRS).

Ontwerpscenario's

Er zijn twee ontwerpscenario's voor het gebruik van trapsgewijze parameters. Ze kunnen effectief worden gebruikt voor het volgende:

- *Grote sets* items filteren
- *Relevante* items weergeven

Voorbeeld database

De voorbeelden die in dit artikel worden weergegeven, zijn gebaseerd op een Azure SQL Database. De database registreert verkoopbewerkingen en bevat verschillende tabellen waarin wederverkopers, producten en verkooporders worden opgeslagen.

Een tabel met de naam **Reseller** slaat een record voor elke wederverkoper op en bevat vele duizenden records. De tabel **Reseller** bevat de volgende kolommen:

- ResellerCode (geheel getal)
- ResellerName
- Land-regio
- Staat-Provincie
- Plaats
- PostalCode

Er is ook een tabel met de naam **Verkoop**. Hier worden Verkooporder-records opgeslagen en er is een refererende-sleutelrelatie met de **Reseller**-tabel in de kolom **ResellerCode**.

Voorbeeld vereiste

Er is een vereiste voor het ontwikkelen van een profielrapport voor wederverkopers. Het rapport moet zijn ontworpen voor het weergeven van informatie voor één reseller. Het is niet gebruikelijk om de gebruiker van het rapport een reseller-code te laten invoeren, omdat ze deze zelden onthouden.

Grote sets items filteren

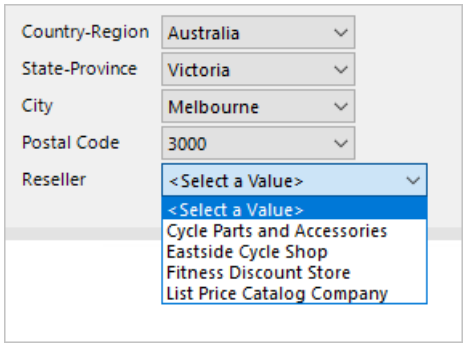
Laten we eens kijken naar drie voorbeelden om u te helpen grote sets beschikbare items te beperken, zoals

wederverkopers. Dit zijn:

- Filteren op gerelateerde kolommen
- Filteren op een groepeerkolom
- Filteren op zoekpatroon

Filteren op gerelateerde kolommen

In dit voorbeeld communiceert de gebruiker van het rapport met vijf rapportparameters. Ze moeten land/regio, staats provincie, plaats en vervolgens postcode selecteren. Een laatste parameter bevat vervolgens een lijst met wederverkopers die zich op die geografische locatie bevinden.



U kunt de trapsgewijze parameters als volgt ontwikkelen:

1. Maak de vijf rapportparameters, geordend in de juiste volgorde.
2. Maak de **CountryRegion**-gegevensset die de waarden van de land/regio ophaalt, met behulp van de volgende query-instructie:

```
SELECT DISTINCT
  [Country-Region]
FROM
  [Reseller]
ORDER BY
  [Country-Region]
```

3. Maak de **StateProvince**-gegevensset die afzonderlijke waarden voor de staat-provincie voor het geselecteerde land-regio ophaalt met de volgende query-instructie:

```
SELECT DISTINCT
  [State-Province]
FROM
  [Reseller]
WHERE
  [Country-Region] = @CountryRegion
ORDER BY
  [State-Province]
```

4. Maak de **City**-gegevensset die de waarden voor afzonderlijke steden voor het geselecteerde land/regio en de staat-provincie gebruikt, met behulp van de volgende query-instructie:

```
SELECT DISTINCT
  [City]
FROM
  [Reseller]
WHERE
  [Country-Region] = @CountryRegion
  AND [State-Province] = @StateProvince
ORDER BY
  [City]
```

5. Ga door met dit patroon om de **PostalCode**-gegevensset te maken.

6. Maak de **Reseller**-gegevensset om alle resellers voor de geselecteerde geografische waarden op te halen met behulp van de volgende query-instructie:

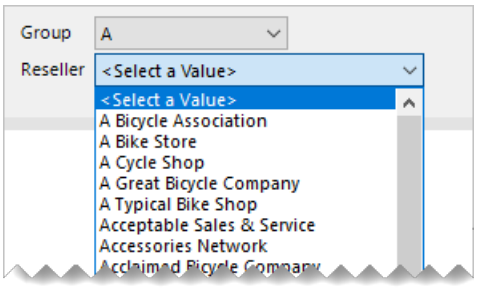
```
SELECT
  [ResellerCode],
  [ResellerName]
FROM
  [Reseller]
WHERE
  [Country-Region] = @CountryRegion
  AND [State-Province] = @StateProvince
  AND [City] = @City
  AND [PostalCode] = @PostalCode
ORDER BY
  [ResellerName]
```

7. Voor elke gegevensset, met uitzondering van de eerste, wijst u de query-parameters toe aan de bijbehorende rapportparameters.

> **NOTE**
>
> Alle query-parameters (voorafgegaan door het @-teken) die in deze voorbeelden worden weer gegeven, kunnen worden ingesloten in SELECT-instructies of doorgegeven aan opgeslagen procedures.
>
> Opgeslagen procedures zijn over het algemeen een betere ontwerpbenadering. De reden hiervoor is dat de query-plannen worden opgeslagen in de cache om sneller te kunnen worden uitgevoerd. In dat geval kunt u zo nodig meer geavanceerde logica ontwikkelen. Ze worden momenteel echter niet ondersteund voor gateway-relationele gegevensbronnen, dit zijn SQL Server, Oracle en Teradata.
>
> Ten slotte moet u er altijd voor zorgen dat er geschikte indexen bestaan om het efficiënt ophalen van gegevens te ondersteunen. Als dat niet het geval is, kunnen de rapportparameters slechts langzaam worden gevuld. De database kan worden overbelast. Zie SQL Server index-architectuur en ontwerphandleiding voor meer informatie over het indexeren van SQL Server.

Filteren op een groepeerkolom

In dit voorbeeld communiceert de gebruiker van het rapport met een rapportparameter om de eerste letter van de reseller te selecteren. In een tweede parameter worden vervolgens de resellers weergegeven wanneer de naam begint met de geselecteerde letter.



U kunt de trapsgewijze parameters als volgt ontwikkelen:

1. Maak de rapportparameters **ReportGroup** en **Reseller**, geordend in de juiste volgorde.
2. Maak de **ReportGroup**-gegevensset om de eerste letters op te halen die door alle wederverkopers worden gebruikt, met behulp van de volgende query-instructie:

```
SELECT DISTINCT
  LEFT([ResellerName], 1) AS [ReportGroup]
FROM
  [Reseller]
ORDER BY
  [ReportGroup]
```

3. Maak de **Reseller**-gegevensset om alle resellers die beginnen met de geselecteerde letter, op te halen met behulp van de volgende query-instructie:

```
SELECT
  [ResellerCode],
  [ResellerName]
FROM
  [Reseller]
WHERE
  LEFT([ResellerName], 1) = @ReportGroup
ORDER BY
  [ResellerName]
```

4. Wijs de query-parameter van de **Reseller**-gegevensset toe aan de bijbehorende rapportparameter.

Het is efficiënter om de kolomgroepering toe te voegen aan de tabel **Reseller**. Wanneer persistent en geïndexeerd, levert dit het beste resultaat op. Zie Berekende kolommen opgeven in een tabel voor meer informatie.

```
ALTER TABLE [Reseller]
ADD [ReportGroup] AS LEFT([ResellerName], 1) PERSISTED
```

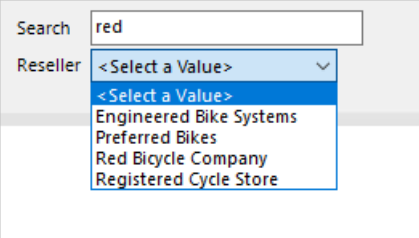
Deze techniek kan nog meer mogelijkheden leveren. Bekijk het volgende script waarmee een nieuwe groepeerkolom wordt toegevoegd om resellers te filteren op *vooraf gedefinieerde kant-en-klare brieven*. Er wordt ook een index gemaakt voor het efficiënt ophalen van de gegevens die vereist zijn voor de rapportparameters.

```
ALTER TABLE [Reseller]
ADD [ReportGroup2] AS CASE
  WHEN [ResellerName] LIKE '[A-C]%' THEN 'A-C'
  WHEN [ResellerName] LIKE '[D-H]%' THEN 'D-H'
  WHEN [ResellerName] LIKE '[I-M]%' THEN 'I-M'
  WHEN [ResellerName] LIKE '[N-S]%' THEN 'N-S'
  WHEN [ResellerName] LIKE '[T-Z]%' THEN 'T-Z'
  ELSE '[Other]'
END PERSISTED
GO

CREATE NONCLUSTERED INDEX [Reseller_ReportGroup2]
ON [Reseller] ([ReportGroup2]) INCLUDE ([ResellerCode], [ResellerName])
GO
```

Filteren op zoekpatroon

In dit voorbeeld communiceert de gebruiker van het rapport met een rapportparameter om een zoekpatroon te selecteren. In een tweede parameter worden vervolgens de resellers weergegeven wanneer de naam het patroon bevat.



U kunt de trapsgewijze parameters als volgt ontwikkelen:

1. Maak de rapportparameters **Search** en **Reseller**, geordend in de juiste volgorde.
2. Maak de **Reseller**-gegevensset om alle resellers die de zoektekst bevatten, op te halen met behulp van de volgende query-instructie:

   ```
   SELECT
     [ResellerCode],
     [ResellerName]
   FROM
     [Reseller]
   WHERE
     [ResellerName] LIKE '%' + @Search + '%'
   ORDER BY
     [ResellerName]
   ```

3. Wijs de query-parameter van de **Reseller**-gegevensset toe aan de bijbehorende rapportparameter.

> **TIP**
>
> U kunt dit ontwerp verbeteren zodat u meer controle hebt over uw rapportgebruikers. Hiermee kunnen ze hun eigen patroon-vergelijkingswaarde definiëren. Met de zoekwaarde "rood%" wordt bijvoorbeeld gefilterd op resellers met namen die *beginnen* met de tekens "rood".
>
> Zie LIKE (Transact-SQL) voor meer informatie.

Hier kunt u de rapportgebruikers de mogelijkheid geven hun eigen patroon te definiëren.

```
WHERE
  [ResellerName] LIKE @Search
```

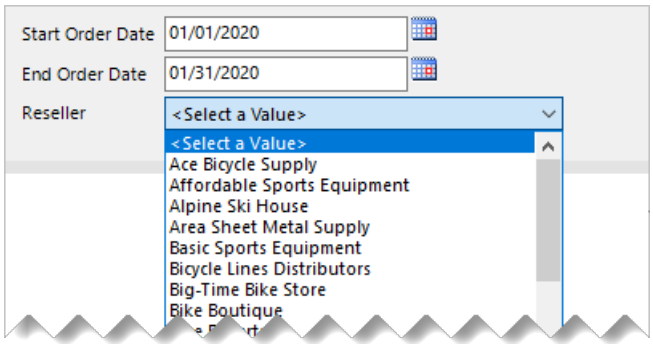
Veel niet-database-professionals zijn echter niet bekend met het percentage(%)-Jokerteken. In plaats daarvan zijn ze bekend met het sterretje (*). Door de component WHERE te wijzigen, kunt u hen toestaan dit teken te gebruiken.

```
WHERE
  [ResellerName] LIKE SUBSTITUTE(@Search, '%', '*')
```

Relevante items weergeven

In dit scenario kunt u feitelijke gegevens gebruiken om de beschikbare waarden te beperken. Rapportgebruikers worden weergegeven met items waarin de activiteit is vastgelegd.

In dit voorbeeld communiceert de gebruiker van het rapport met drie rapportparameters. De eerste twee stellen een datumbereik van de verkooporderdatums in. De derde parameter bevat dan een lijst met wederverkopers waar orders zijn gemaakt tijdens die periode.



U kunt de trapsgewijze parameters als volgt ontwikkelen:

1. Maak de rapportparameters **OrderDateStart**, **OrderDateEnd** en **Reseller**, geordend in de juiste volgorde.
2. Maak de **Reseller**-gegevensset om alle resellers die in de gespecificeerde periode orders hebben aangemaakt, op te halen met behulp van de volgende query-instructie:

```
SELECT DISTINCT
  [r].[ResellerCode],
  [r].[ResellerName]
FROM
  [Reseller] AS [r]
INNER JOIN [Sales] AS [s]
  ON [s].[ResellerCode] = [r].[ResellerCode]
WHERE
  [s].[OrderDate] >= @OrderDateStart
  AND [s].[OrderDate] < DATEADD(DAY, 1, @OrderDateEnd)
ORDER BY
  [r].[ResellerName]
```

Aanbevelingen

U wordt aangeraden uw rapporten zo mogelijk te ontwerpen met parameters. Dit komt doordat ze:

- Een intuïtieve en nuttige ervaring bieden voor uw rapportgebruikers
- Efficiënt zijn, omdat ze kleinere sets beschikbare waarden ophalen

Zorg ervoor dat u uw gegevensbronnen optimaliseert door:

- Waar mogelijk opgeslagen procedures gebruiken
- Juiste indexen toevoegen voor het efficiënt ophalen van gegevens
- Kolomwaarden materialiseren—en zelfs-rijen—om te voorkomen dat er dure query-tijd evaluaties worden gedaan

Volgende stappen

Bekijk de volgende resources voor meer informatie over dit artikel:

- Rapportparameters in Power BI Report Builder
- Trapsgewijze parameters toevoegen aan een rapport (Power BI Report Builder en SSRS)
- Vragen? Misschien dat de Power BI-community het antwoord weet
- Suggesties? Ideeën bijdragen om Power BI te verbeteren