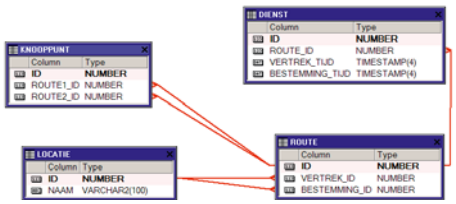
Puzzelen met SQL

Oh jee, we nemen het OV...

Met alle files op de weg is het misschien handiger en rustiger om met het OV te gaan. Maar als ik dan met het OV ga reizen, hoe moet dat dan. Hoe lang ben ik eigenlijk onderweg. Kan ik langere tijd werken of lezen tijdens de reis? Als je met de auto gaat, kun je gewoon van A naar B rijden. Het OV vertrekt waar je niet bent en brengt je waar je niet moet zijn. Dus om een route te bepalen moet je verschillende trajecten aan elkaar 'knopen'.

Door gebruik te maken van de site http://www.ov9292.nl kun je op redelijk eenvoudige wijze een reisplan maken, maar als je de informatie van de locaties hebt, kunnen we daar natuurlijk ook met SQL mee spelen en kijken of we zelf plannen kunnen maken, waarbij we ook andere parameters kunnen opgeven, die niet op de site zijn ingevoerd.
Als eerste hebben we natuurlijk een datamodel nodig.



Hierin moeten we (minimaal) de volgende gegevens opnemen:
vertrekpunt
vertrektijd
aankomstpunt
aankomsttijd

Wat is de kortste route vanaf mijn thuis adres naar de locatie waar ik wil/moet zijn. Zoals eerder gezegd, brengt het OV je niet van deur tot deur, dus kiezen we de meest logische en dichtstbijzijnde stations. Ik moet van Almere naar Den Haag reizen. Hier is geen directe verbinding voor aanwezig, maar via een knooppunt kan ik daar wel komen. Om te voorkomen dat we de op te geven parameters eventueel meerdere keren moeten opgeven maken we gebruik van query factoring. De parameters die we hebben is 'vertrek' en 'bestemming'. Eerst maar eens de routes bepalen die gebruikt moet worden:

```
with parameters as (select 'Almere' vertrek
                         , 'Den Haag' bestemming
                      from dual
                    )
, locaties as (select vertrek.id vertrek_id
                    , bestemming.id bestemming_id
                 from locatie vertrek
                    , locatie bestemming
                    , parameters
                where vertrek.naam = parameters.vertrek
                  and bestemming.naam = parameters.bestemming
          )
select route.id
  from route
     , locaties
where route.vertrek_id = locaties.vertrek_id
union
select route.id
  from route
     , locaties
 where route.bestemming_id = locaties.bestemming_id
```

In het bovenstaande SQL statement maken we gebruik van Subquery Factoring, zoals de WITH clause in de Oracle documentatie heet. De kracht van Subquery Factoring zit hem vooral in het meerdere keren gebruik kunnen maken van dezelfde inline view en het behalen van een Single Point Of Definition - in bovenstaande query LOCATIES genaamd. Deze LOCATIES view gebruiken we twee maal in de hoofdquery.
Het resultaat van deze query is een lijstje van te gebruiken routes. Als we het juiste knooppunt willen bepalen, dan willen we geen lijstje hebben, maar een record waarmee we verder kunnen. In Excel heet dit een draaitabel, in Oracle hebben we hier de PIVOT functie voor. Deze functie is in Oracle 11g geïntroduceerd en alleen op aggregate functions. Nu kunnen we natuurlijk een MAX loslaten op de id, maar een volgende verplichting die we hebben is het hard coderen van de waarden waarover we willen 'pivotten'. In dit geval is het niet echt nodig

om gebruik te maken van de PIVOT function, we kunnen dit met wat oude bekende functies ook oplossen.
Door gebruik te maken van de MIN() en MAX() functies kunnen we de rijen uit het resultaat omzetten naar kolommen. De query die we net hadden, nemen we ook op in het Subquery Factoring deel en noemen we ROUTES. Aan de ROUTES view kunnen we refereren in de hoofdquery.

```
with parameters as (select 'Almere' vertrek
                         , 'Den Haag' bestemming
                      from dual
                    )
, locaties as (select vertrek.id vertrek_id
                    , bestemming.id bestemming_id
                 from locatie vertrek
                    , locatie bestemming
                    , parameters
                where vertrek.naam = parameters.vertrek
                  and bestemming.naam = parameters.bestemming
          )
, routes as (select route.id id
  from route
     , locaties
where route.vertrek_id = locaties.vertrek_id
union
select route.id
  from route
     , locaties
 where route.bestemming_id = locaties.bestemming_id )
select min(id)
     , max(id)
  from routes;
```

	MIN(ID)	MAX(ID)
1	1	2

Als we het resultaat van deze query gebruiken, kunnen we de overstap locaties bepalen met de volgende query. Hiervoor verplaatsen we de query naar de Subquery Factoring, en leggen we een JOIN met de KNOOPPUNT tabel.

```
with parameters as (select 'Almere' vertrek
                         , 'Den Haag' bestemming
                      from dual
                    )
, locaties as (select vertrek.id vertrek_id
                    , bestemming.id bestemming_id
                 from locatie vertrek
                    , locatie bestemming
                    , parameters
                where vertrek.naam = parameters.vertrek
                  and bestemming.naam = parameters.bestemming
              )
, routes as (select route.id id
               from route
                  , locaties
             where route.vertrek_id = locaties.vertrek_id
            union
            select route.id
              from route
                 , locaties
             where route.bestemming_id = locaties.bestemming_id
           )
, de_route as (select min(id) vertrek
                    , max(id) bestemming
                 from routes
               )
select knooppunt.id
       , knooppunt.route1_id
       , knooppunt.route2_id
  from knooppunt
     , de_route
 where knooppunt.route1_id = de_route.vertrek
   and knooppunt.route2_id = de_route.bestemming
;
```

	ID	ROUTE1_ID	ROUTE2_ID
1	1	1	2

Nu zijn we bijna klaar met deze query. Het resultaat van de query gaan we in de laatste stap gebuiken om de gehele route te bepalen. We krijgen nu de volgende (complete) query:

```
with parameters as (select 'Almere' vertrek
                         , 'Den Haag' bestemming
                      from dual
                    )
, locaties as (select vertrek.id vertrek_id
                    , bestemming.id bestemming_id
                 from locatie vertrek
                    , locatie bestemming
                    , parameters
                where vertrek.naam = parameters.vertrek
                  and bestemming.naam = parameters.bestemming
              )
, routes as (select route.id id
               from route
                  , locaties
             where route.vertrek_id = locaties.vertrek_id
            union
            select route.id
              from route
                 , locaties
             where route.bestemming_id = locaties.bestemming_id
           )
, de_route as (select min(id) vertrek
                    , max(id) bestemming
                 from routes
               )
, knooppunten as (select knooppunt.id
                       , knooppunt.route1_id
                       , knooppunt.route2_id
                    from knooppunt
                       , de_route
                   where knooppunt.route1_id = de_route.vertrek
                     and knooppunt.route2_id = de_route.bestemming
                 )
select vertrek.naam vertrek_naam
     , bestemming.naam bestemming_naam
  from locatie vertrek
     , locatie bestemming
     , route
     , knooppunten
 where vertrek.id = route.vertrek_id
   and bestemming.id = route.bestemming_id
   and route.id = knooppunten.route1_id
union
select vertrek.naam vertrek_naam
     , bestemming.naam bestemming_naam
  from locatie vertrek
```

```
      , locatie bestemming
      , route
      , knooppunten
 where vertrek.id = route.vertrek_id
   and bestemming.id = route.bestemming_id
   and route.id = knooppunten.route2_id
;
```

	VERTREK_NAAM	BESTEMMING_NAAM
1	Almere	Schiphol
2	Schiphol	Den Haag

De route die we moeten afleggen is nu helemaal compleet. Maar met een route alleen kom je nog niet op de plaats van bestemming. Het zou ook handig zijn om de dienstregeling in ogenschouw te nemen. Dan pas weet je hoe laat je op het station moet zijn, hoe lang je moet wachten en hoeveel tijd je hebt om een bakje koffie te halen.
Om het reisschema te bepalen, hebben we de DIENSTREGELING tabel nodig. We kunnen natuurlijk niet twee keer vanaf dezelfde locatie vertrekken, maar het kan wel voorkomen dat er meerdere diensten vanaf een knooppunt vertrekken. Hierdoor kan het gebeuren dat we in plaats van ons te haasten om de aansluiting te halen we de tijd kunnen nemen om rustig een kopje koffie of een broodje te halen.
Laten we eerst eens de mogelijke routes in combinatie met de dienstregeling bepalen:

```
select dienst.id dienst_id
      , route.id route_id
      , dienst.vertrek_tijd
      , dienst.bestemming_tijd
      , vertrek.naam vertrek_naam
      , bestemming.naam bestemming_naam
  from dienst
  join route on route.id = dienst.route_id
  join locatie vertrek on route.vertrek_id = vertrek.id
  join locatie bestemming on route.bestemming_id = bestemming.id;
```

	DIENST_ID	ROUTE_ID	VERTREK_TIJD
1	1	1	01-MAY-09 06.00.00.0000 AM
2	2	2	01-MAY-09 06.35.00.0000 AM
3	3	1	01-MAY-09 06.20.00.0000 AM
4	4	2	01-MAY-09 06.55.00.0000 AM

	DIENST_ID	ROUTE_ID	BESTEMMING_TIJD
1	1	1	01-MAY-09 06.30.00.0000 AM
2	2	2	01-MAY-09 07.10.00.0000 AM
3	3	1	01-MAY-09 06.50.00.0000 AM
4	4	2	01-MAY-09 07.40.00.0000 AM

	DIENST_ID	ROUTE_ID	VERTREK_NAAM	BESTEMMING_NAAM
1	1	1	Almere	Schiphol
2	2	2	Schiphol	Den Haag
3	3	1	Almere	Schiphol
4	4	2	Schiphol	Den Haag

Met deze query bepalen we de diensten en de route en nemen we ook de namen van de locaties op. We maken gebruik van ANSI SQL. Hiermee worden de joins in de from clause opgenomen en niet zoals we gewend zijn in de where clause. Dit maakt het een stuk overzichtelijker, zeker als we nog andere predicaten willen opnemen om de resultaatset te verkleinen. Deze query nemen we op via een with clause als een refactored query, kunnen we handig gebruik maken van inline views. Ook hier wordt dezelfde inline view twee keer gebruikt en hoeven we het maar 1 keer te schrijven. Dus minder code, minder kans op fouten, minder kans op RSI, groener (alles moet tegenwoordig groener, dus is dit een leuk argument).

```
with route_dienst as (
select dienst.id dienst_id
      , route.id route_id
      , dienst.vertrek_tijd
      , dienst.bestemming_tijd
      , vertrek.naam vertrek_naam
      , bestemming.naam bestemming_naam
  from dienst
  join route on route.id = dienst.route_id
  join locatie vertrek on route.vertrek_id = vertrek.id
  join locatie bestemming on route.bestemming_id = bestemming.id
)
select a.vertrek_naam, a.vertrek_tijd, a.bestemming_naam, a.bestemming_
tijd
      , b.vertrek_naam, b.vertrek_tijd, b.bestemming_naam, b.bestemming_
tijd
  from route_dienst a
  join route_dienst b on a.bestemming_tijd < b.vertrek_tijd
;
```

	VERTREK_NAAM	VERTREK_TIJD	BESTEMMING_NAAM	BESTEMMING_TIJD
1	Almere	01-MAY-09 06.00.00.0000 AM	Schiphol	01-MAY-09 06.30.00.0000 AM
2	Almere	01-MAY-09 06.00.00.0000 AM	Schiphol	01-MAY-09 06.30.00.0000 AM
3	Almere	01-MAY-09 06.20.00.0000 AM	Schiphol	01-MAY-09 06.50.00.0000 AM

	VERTREK_NAAM	VERTREK_TIJD	BESTEMMING_NAAM	BESTEMMING_TIJD
1	Schiphol	01-MAY-09 06.35.00.0000 AM	Den Haag	01-MAY-09 07.10.00.0000 AM
2	Schiphol	01-MAY-09 06.55.00.0000 AM	Den Haag	01-MAY-09 07.40.00.0000 AM
3	Schiphol	01-MAY-09 06.55.00.0000 AM	Den Haag	01-MAY-09 07.40.00.0000 AM

Nu hebben we de complete dienstregeling in kaart. Door nu de parameters uit eerdere queries toe te voegen kunnen we onze reis mogelijkheden bepalen.

```
with parameters as (select 'Almere' vertrek
                         , 'Den Haag' bestemming
                         , to_timestamp('06:00','HH24:MI') vertrek_tijd
                      from dual
                    )
, route_dienst as (
select dienst.id dienst_id
      , route.id route_id
      , dienst.vertrek_tijd
      , dienst.bestemming_tijd
      , vertrek.naam vertrek_naam
      , bestemming.naam bestemming_naam
  from dienst
  join route on route.id = dienst.route_id
  join locatie vertrek on route.vertrek_id = vertrek.id
  join locatie bestemming on route.bestemming_id = bestemming.id
)
select a.vertrek_naam, a.vertrek_tijd, a.bestemming_naam, a.bestemming_
tijd
      , b.vertrek_naam, b.vertrek_tijd, b.bestemming_naam, b.bestemming_tijd
  from route_dienst a
  join route_dienst b on a.bestemming_tijd < b.vertrek_tijd
  join parameters on (     a.vertrek_naam = parameters.vertrek
                       and b.bestemming_naam = parameters.bestemming
                       and a.vertrek_tijd >= parameters.vertrek_tijd
                     )
;
```

	VERTREK_NAAM	VERTREK_TIJD	BESTEMMING_NAAM	BESTEMMING_TIJD
1	Almere	01-MAY-09 06.00.00.0000 AM	Schiphol	01-MAY-09 06.30.00.0000 AM
2	Almere	01-MAY-09 06.00.00.0000 AM	Schiphol	01-MAY-09 06.30.00.0000 AM
3	Almere	01-MAY-09 06.20.00.0000 AM	Schiphol	01-MAY-09 06.50.00.0000 AM

	VERTREK_NAAM	VERTREK_TIJD	BESTEMMING_NAAM	BESTEMMING_TIJD
1	Schiphol	01-MAY-09 06.35.00.0000 AM	Den Haag	01-MAY-09 07.10.00.0000 AM
2	Schiphol	01-MAY-09 06.55.00.0000 AM	Den Haag	01-MAY-09 07.40.00.0000 AM
3	Schiphol	01-MAY-09 06.55.00.0000 AM	Den Haag	01-MAY-09 07.40.00.0000 AM

Zoals te zien is, kan een tabel (of een refactored query) op meer dan 1 veld gejoined worden met andere tabellen in de query. Het is natuurlijk ook mogelijk om 1 tabel aan meerdere andere tabellen te joinen.
Nu hebben we de mogelijkheden van de route bepaald en kunnen we met de parameters gaan spelen, zoals de vertrek tijd aanpassen. Maar we kunnen ook een extra parameter opnemen waarin we de overstaptijd opnemen.

```
with parameters as (select 'Almere' vertrek
                           , 'Den Haag' bestemming
                           , to_timestamp('06:00','HH24:MI') vertrek_
tijd
                           , numtodsinterval(5, 'MINUTE') overstap_tijd
                      from dual
                    )
, route_dienst as (
select dienst.id dienst_id
     , route.id route_id
     , dienst.vertrek_tijd
     , dienst.bestemming_tijd
     , vertrek.naam vertrek_naam
     , bestemming.naam bestemming_naam
  from dienst
  join route on route.id = dienst.route_id
  join locatie vertrek on route.vertrek_id = vertrek.id
  join locatie bestemming on route.bestemming_id = bestemming.id
)
select a.vertrek_naam, a.vertrek_tijd, a.bestemming_naam, a.bestemming_
tijd
     , b.vertrek_naam, b.vertrek_tijd, b.bestemming_naam, b.bestemming_
tijd
  from route_dienst a
  join route_dienst b on a.bestemming_tijd < b.vertrek_tijd
  join parameters on (    a.vertrek_naam = parameters.vertrek
                      and b.bestemming_naam = parameters.bestemming
                      and a.vertrek_tijd >= parameters.vertrek_tijd
                     )
 where (b.vertrek_tijd - a.bestemming_tijd) >= parameters.overstap_tijd
;
```

	VERTREK_NAAM	VERTREK_TIJD	BESTEMMING_NAAM	BESTEMMING_TIJD
1	Almere	01-MAY-09 06.00.00.0000 AM	Schiphol	01-MAY-09 06.30.00.0000 AM
2	Almere	01-MAY-09 06.00.00.0000 AM	Schiphol	01-MAY-09 06.30.00.0000 AM
3	Almere	01-MAY-09 06.20.00.0000 AM	Schiphol	01-MAY-09 06.50.00.0000 AM

	VERTREK_NAAM	VERTREK_TIJD	BESTEMMING_NAAM	BESTEMMING_TIJD
1	Schiphol	01-MAY-09 06.35.00.0000 AM	Den Haag	01-MAY-09 07.10.00.0000 AM
2	Schiphol	01-MAY-09 06.55.00.0000 AM	Den Haag	01-MAY-09 07.40.00.0000 AM
3	Schiphol	01-MAY-09 06.55.00.0000 AM	Den Haag	01-MAY-09 07.40.00.0000 AM

Als we twee timestamps van elkaar aftrekken, dan krijgen we een interval terug. Dus moeten we de overstaptijd ook als interval opnemen om het correct met elkaar te kunnen vergelijken. Als we de overstaptijd 'te lang' maken, dan blijven er maar weinig mogelijkheden over, maar hebben we wel veel meer tijd voor het halen van koffie of iets anders. Het resultaat van de bovenstaande 3 queries is iedere keer hetzelfde, maar dat komt door de 'slimme' keuze van de parameters. Als er bijvoorbeeld 15 minuten overstaptijd opgegeven wordt, blijft er maar 1 mogelijke reis over.

Nu is het reizen met het openbaar vervoer misschien niet ideaal, eerst moet je naar het station – dan moet je wachten op de trein, misschien nog een overstap en dus weer wachten... Dan kom je weer op een station en moet je nog een manier hebben om op de plaats van bestemming te komen. Maar de tijd dat je in de trein zit, kan je wel besteden om iets anders te doen. Wat lezen, even slapen, misschien zelfs wat werken, gewoon met de telefoon in de hand bellen, al dit soort dingen die je niet kunt doen als je in de file staat, en het is nog beter voor het milieu ook. Ons advies is dan ook om vaker de trein te nemen. En als jullie als lezers dat nu eens en-masse doen, kunnen wij op de snelweg weer normaal doorrijden.

Patrick Barel is consultant bij AMIS Services in Nieuwegein. Hij schrijft op het blog van AMIS (http://technology.amis.nl/blog) en op zijn eigen blog (http://blog.bar-solutions.com).

Alex Nuijten is Oracle consultant bij AMIS Services. Hij schrijft op het Technology Blog van AMIS (technology.amis.nl/blog/) en op zijn eigen blog (nuijten.blogspot.com).

In de puzzel van November 2008 hebben we gebruik gemaakt van het spel SET om een puzzel te maken. Naar aanleiding van het contact met de makers van dit spel willen we nog de volgende copyright regel melden: Images reproduced with permission from Set Enterprises, Inc. Copyright © 1988, 1991 Cannei, LLC. SET® is a registered trademark of Cannei, LLC.