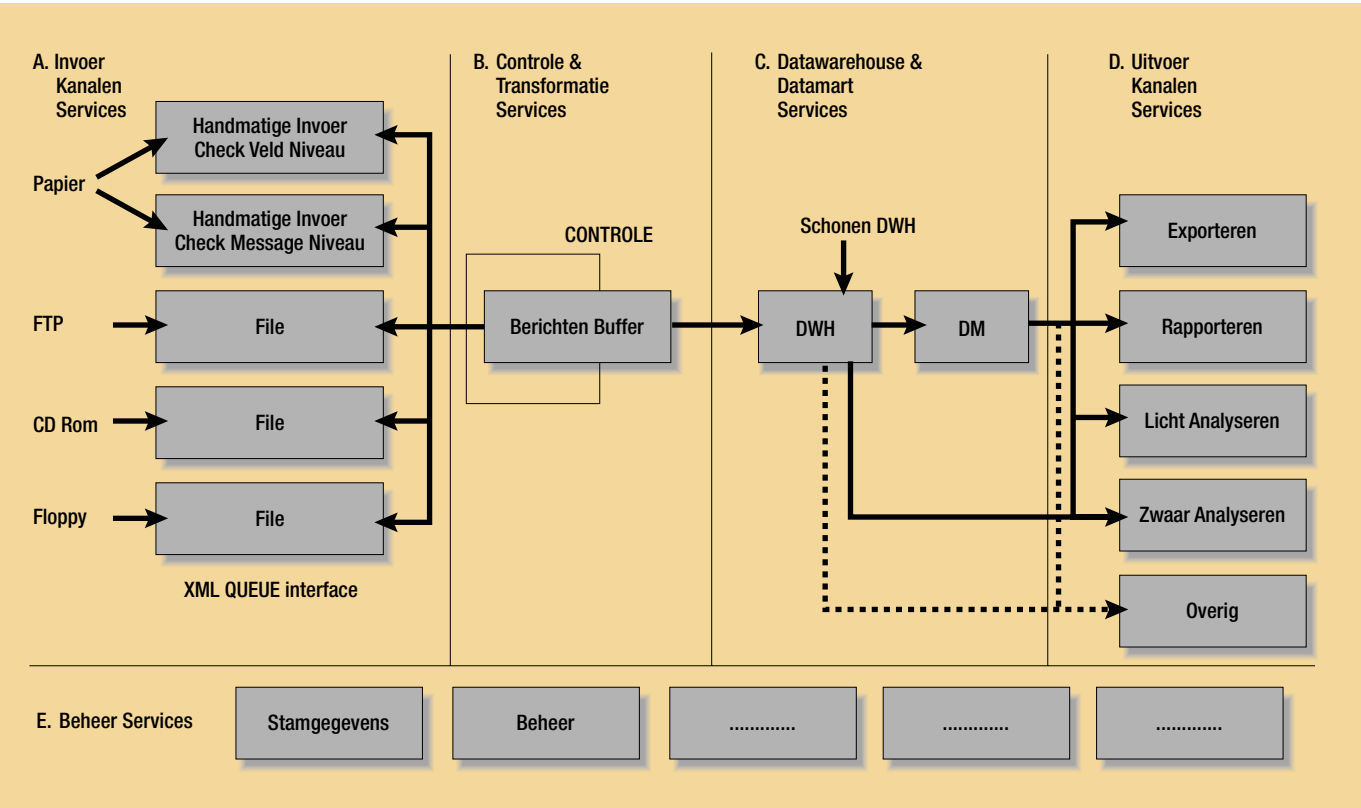
Autorisatiepolicy's in een datawarehouse

Oracle VPD en Oracle Label Security

Bij het opzetten van een datawarehouse speelt autorisatieproblematiek vaak een belangrijke rol. De data moeten voor management-gebruik ontsloten worden, maar daarbij gelden duidelijke restricties. In dit geval was er ook nog eens sprake van regionale autorisatie verschillen. In dit artikel wordt besproken hoe Oracle VPD en Oracle Lable Security daarbij ingezet kunnen worden.

Dit artikel is gebaseerd op de implementatie van een datawarehouse (DWH) uitgevoerd door Cap Gemini Ernst & Young. Dit datawarehouse is gebaseerd op de architectuur uit figuur 1. De data zijn afkomstig uit verschillende bronnen en worden in de invoerkanalen aangeboden aan de controle en transformatie services. Deze services dragen zorg voor het opnemen van de data in het DWH gedeelte van het datawarehouse. Vanuit het DWH schema worden de verschillende datamarts (DM) opgebouwd. Het DWH model is hierbij relationeel opgebouwd en de datamarts bestaan uit stermodellen.

De gerealiseerde datamarts worden in dit systeem aangeboden aan een grote groep eindgebruikers via het intra/internet. Dit datawarehouse wordt gevoed door een groot aantal regionale organisaties. Elke regionale organisatie mag alleen haar eigen data bekijken maar niet de data van andere regio's. Wel zijn er



Figuur 1. Vanuit het datawarehouse schema worden de verschillende datamarts opgebouwd

een aantal geaggregeerde gegevens die voor alle regio's toegankelijk zijn. Aangezien toegang via verschillende uitvoerkanalen (tools) mogelijk moet zijn ligt het voor de hand deze autorisatie in de database te regelen. Autorisatie op databaseniveau geldt immers automatisch voor alle te gebruiken tools. In dit artikel willen wij ingaan op hoe wij deze autorisatie-eisen in dit datawarehouse hebben gerealiseerd.

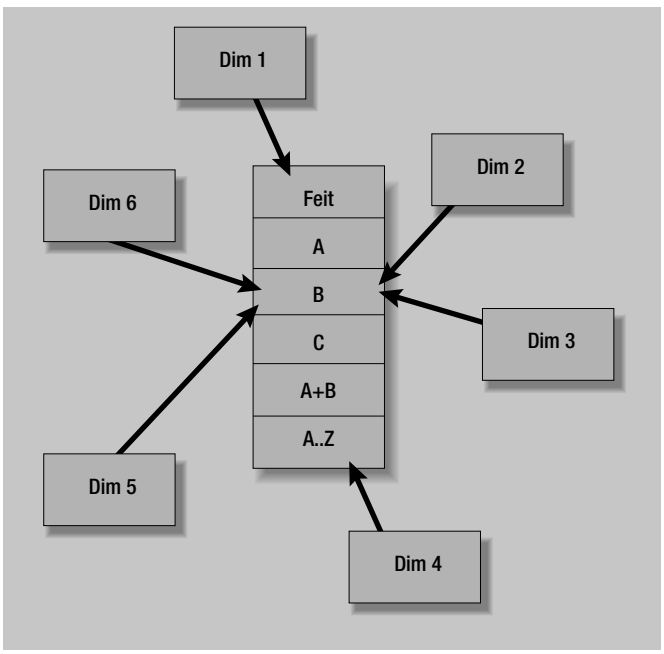
De problematiek in detail

In een datawarehouse speelt de autorisatieproblematiek voornamelijk in de datamarts. Deze worden immers door de eindgebruikers geraadpleegd. In dit datawarehouse zijn de datamarts gemodelleerd als sterren. Een ster bestaat uit een feit met daaromheen dimensies. Eindgebruikers mogen alle dimensies raadplegen en zij mogen een deel van de feiten raadplegen (zie figuur 2). Gebruiker A mag een deel van de feiten raadplegen, gebruiker B mag een ander deel van de feiten raadplegen, maar er is ook een gedeelte van de feiten die zij allebei mogen raadplegen.

Traditioneel werd deze vorm van autorisatie vaak geïmplementeerd door de feittabel op te splitsen in aparte feittabellen en views per groep gebruikers. Zodra de complexiteit van de organisatiestructuur toeneemt neemt ook de complexiteit van deze oplossing toe. Daarmee neemt dan ook de beheersbaarheid van deze manier van implementeren af (zie figuur 3).

Een betere oplossing

Een betere oplossing kan worden gerealiseerd met Oracle VPD of Oracle Label Security. Eerst zullen wij deze begrippen definiëren.



Figuur 2. Eindgebruikers mogen alle dimensies en een deel van de feiten raadplegen

Wat is Oracle VPD?

Virtual Private Database (VPD) is geïntroduceerd in Oracle 8i. Een VPD biedt een Fine-Grained Access Control (FGAC) voor een beveiligde data-toegang. Gebruikers krijgen alleen toegang tot data waartoe zij recht hebben. Deze optie maakt het zelfs mogelijk verschillende bedrijven van hetzelfde databaseschema gebruik te laten maken zonder dat zij dit van elkaar weten. VPD wordt geconfigureerd met de package DBMS_RLS (Row Level Security).

Wat is Oracle Label Security?

Oracle Label Security (vroeger bekend onder de naam Trusted Oracle MLS RDBMS) maakt gebruik van de Oracle 8i VPD (Virtual Private Database) optie om row level security te realiseren. Toegang tot regels wordt beperkt tot het gebruikers beveiligingsniveau. Oracle Label Security wordt geconfigureerd, gecontroleerd en beheerd door de Policy Manager, een Enterprise Manager base GUI utility. Oracle VPD bevat de tools om Fine Grained Access Control te realiseren en zit standaard in de database. Oracle heeft, gebruikmakend van VPD, een row level security gerealiseerd in de vorm van Oracle Label Security. Oracle Label Security is een aparte database optie.

Vertaling naar onze problematiek

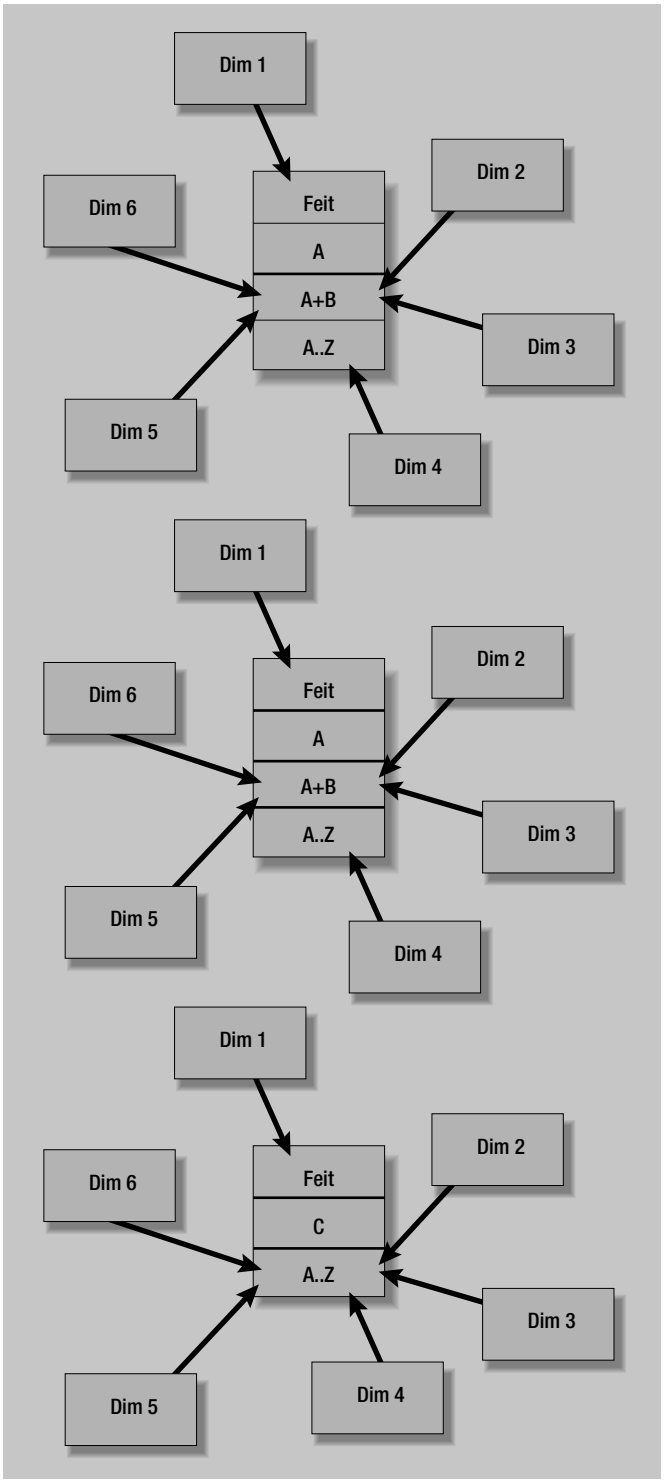
In eerste instantie hebben wij Oracle Label Security bestudeerd om de gewenste autorisatie te bereiken. Met deze kennis kwamen wij voor onze gewenste autorisatie tot het model zoals weergegeven in figuur 4. Met de hulp van labels wordt geïdentificeerd welke gebruikers(groep) welk gegeven mag zien. De entiteit label staat voor de kleinste te autoriseren groepen. Deze entiteit is enerzijds gerelateerd aan de te beveiligen data en anderzijds aan de gebruikers. Elke te beveiligen feit heeft een relatie naar de entiteit label. De feitentabel heeft een '1 op 1' relatie met de labeltabel. Bij een gebruiker horen een aantal labels. Deze relatie tussen gebruikers en labels is vastgelegd in het rechter deel van het model in figuur 4.

Oracle Label Security is een implementatie van de hierboven geschetste oplossing. Het autorisatie / organisatie model bevat begrippen als compartments, groups en levels (zie de Oracle documentatie voor een uitleg van deze begrippen). Bij de bestudering van dit model bleek dit niet passen in onze gewenste autorisatie / organisatie structuur. Daarom hebben wij onze 'eigen' implementatie gemaakt van label security.

Implementatie

Onze implementatie bestaat de volgende onderdelen:

- Het autorisatie/organisatiemodel
- De policy functie
- Het aanmaken van de policy in de database
- De trigger voor het vullen van het label relatie attribuut



Figuur 3. Bij deze vorm van autorisatie wordt de feittabel opgesplitst in aparte tabellen of views per groep gebruikers

In het autorisatie/organisatiemodel worden de relaties tussen de labels en de gebruikers vastgelegd. De policy functie bepaalt aan de hand van het autorisatie/organisatie model wat aan een SQL statement moet worden toegevoegd om de autorisatie te implementeren. Door het aanmaken van de policy wordt de policy functie verbonden met de tabel. Vanaf dit moment gebruikt de database de policy functie om de tabel te beveiligen. Voor het vullen van de labels wordt een trigger gebruikt.

Autorisatie/organisatie model

Elke organisatie kan dit model naar eigen inzicht invullen. In ons geval hebben we vastgehouden aan de bestaande rollen en gebruikers van Oracle. Met de hulp van labels wordt geïdentificeerd welke gebruikers(groepen) welk gegeven mag zien. Labels worden in ons model gerelateerd aan rollen of gebruikers. Op deze wijze ontstaat er een transparante wijze van autorisatie uitdelen.

De beheerfuncties van rollen en gebruikers bestaan al binnen de bestaande database utility's en hoeven dus niet meer te worden gerealiseerd. Wel moeten er beheerfuncties worden gerealiseerd die de labels aan de juiste rollen of gebruikers koppelen. Dit is in ons geval relatief statische informatie (door het gebruik maken van rollen) die alleen bij organisatie veranderingen wijzigt. Een ieder is echter zo vrij om dit zo ingewikkeld te maken als men maar wil. Het enige waarop moet worden gelet is de performance van de 'policy functie'. Aan de hand van dit model bepaalt deze functie welke labels horen bij welke gebruiker. Deze wordt vaak aangeroepen en zal dus efficiënt zijn taak moeten kunnen vervullen.

De gerealiseerde datamarts worden in dit systeem aangeboden aan een grote groep eindgebruikers via het intra/internet

Policy functie

Doel van de policy functie is automatisch aan ieder SQL statement de where clause toe te voegen met een stuk SQL code die de toegang tot de tabel regelt:

De gebruiker wil het volgende statement uitvoeren:
*select * from tabelx*

De database maakt, voordat het statement wordt uitgevoerd, hiervan
*select * from tabelx where label in (select .....)*

De volgende alinea is een voorbeeld van een policy functie:

```
function geef_auto(object_schema in varchar2,
                      object_name    in varchar2) return varchar2 is

    l_return  varchar2(2000);

  begin

    if user in ('BEHEER', 'DISCOVERER', 'DM', 'DWH') then
      l_return := null;
    else


      if g_#_roles <= 0 then
        l_return := ' label IN (SELECT lbl.lbl_id FROM label_gebrui-
kers lbl WHERE lbl.gbr_naam = user)';

      else

        l_return := ' label IN (SELECT lbl.lbl_id FROM label_gebrui-
kers lbl WHERE lbl.gbr_naam  = user OR lbl.gbr_naam  IN (' || g_roles
|| '))';

      end if;

    end if;

    return l_return;

  end;
```

Er wordt gebruik gemaakt van sessie variabelen (g_). Dit verhoogt de performance. Alleen bij de start van de sessie worden de rollen bepaald waarvoor de gebruiker is geautoriseerd.

Policy

Om een policy te creëren is het mogelijk gebruik te maken van de Policy Manager die geleverd wordt als Oracle Label Security. Aangezien Oracle Label Security in ons geval niet voldeed voor deze toepassing moest de implementatie plaats vinden middels de onderliggende package DBMS_RLS. In deze package zitten vier procedures die van belang zijn voor het onderhoud aan policy's: ADD_POLICY, DROP_POLICY, ENABLE_POLICY en REFRESH_POLICY. Voor het maken van een policy is een functie noodzakelijk die als resultaat een conditie oplevert. De policy zal deze routine koppelen aan de tabel zodat in het vervolg de uit te voeren SQL-statements te maken krijgen met een extra conditie. De aanroep van de procedure ADD_POLICY ziet er als volgt uit:

```
BEGIN
DBMS_RLS.ADD_POLICY
(OBJECT_SCHEMA => 'DATABASE_SCHEMA',
 OBJECT_NAME => 'DATABASE_TABEL',
 POLICY_NAME => 'POLICY_NAAM',
 FUNCTION_SCHEMA => 'FUNCTIE_SCHEMA',
 POLICY_FUNCTION => 'POLICY_FUNCTIE',
 STATEMENT_TYPES => 'SELECT, INSERT, UPDATE, DELETE',
 UPDATE_CHECK => TRUE
);
END;
```

Object_schema	Het schema waarin de databasetabel is opgenomen
Object_name	Naam van de tabel
Policy_name	Naam van de policy
Function_schema	Schema waarin de functie is opgenomen
Policy_function	Naam van de functie
Statement_types	De SQL-statements waarop de policy van toepassing is
Update_check	Controle bij wijziging of deze wel is toegestaan

Trigger

Voor het vullen van het relatie attribuut naar de tabel label hebben we gebruik gemaakt van triggers met de volgende structuur:

```
create or replace trigger ftl_bifer
before insert on feit_tabel
for each row
/*********************************************************************
*********
Doel van het programma:
======================
Bij de creatie van een nieuw record in de tabel feit_tabel
wordt label gevuld door de keys id1 en id2 samen te voegen.
*********************************************************************
********/
begin
   :new.label := :new.id1 || :new.id2;
end ftl_bifer;
```

In ons geval kan het label rechtstreeks worden afgeleid uit een aantal velden van de primary key. Dat maakt deze trigger reuze eenvoudig. Ook hier geldt overigens dat je de trigger zo ingewikkeld mogelijk kunt maken bij complexe autorisatiestructuren. Indien er echter een hoge performance wordt geëist bij het vullen van de feitabel is het wel zaak de complexiteit niet te overdrijven.

Overig

Rechten en VPD

Alle gebruikers zijn altijd onderworpen aan de policy als deze op de tabel ligt. De enige gebruiker die default uitgesloten is van alle policy's in de database is SYS. Middels de systeem privilege EXEMPT ACCESS POLICY is het mogelijk om ook andere gebruikers uit te sluiten van de policy's op de tabellen. Dit is echter een niet gewenste situatie omdat dit het hele beveiligingsmechanisme omzeilt. Beter is het om een gebruiker (groep) via de policy functie eventuele volledige tabel toegang te verlenen. Denk hierbij vooral ook aan de user die voor de database back-up (imp en exp) wordt gebruikt.

Trace file

Het is mogelijk om de toegevoegde clausule te zien door het volgende statement uit te voeren:

```
ALTER SESSION SET EVENTS '10730 trace name context forever, level 12';
```

Er wordt nu in de tracefile het desbetreffende sql statement met de toegevoegde conditie alsook de naam van de tabel, de naam van de policy en de functie die de policy gebruikt opgenomen. Met dit alter-statement is het in ontwikkeling goed mogelijk om te kijken of de policy werkt en de conditie de juiste is.

VPD en Discoverer

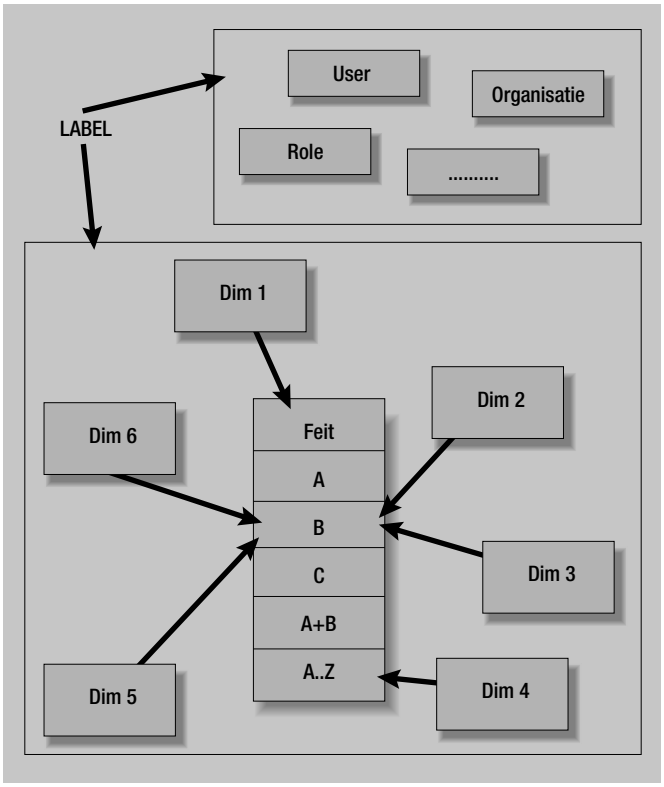
Oracle Discoverer

De autorisatie binnen Discoverer wordt vereenvoudigd door het gebruik van VPD. De onderliggende tabellen zijn nu afgeschermd middels een policy. Oplossingen middels views en/of meerdere bijna identieke werkboeken met alleen verschillende autorisaties behoren tot het verleden. Door gebruik te maken van VPD kan het werkboek aan alle gebruikers worden aangeboden en kunnen de gebruikers alleen die gegevens zien die ook werkelijk voor hen bestemd zijn.

Summary's / materialized views

In Discoverer admin is het mogelijk om automatisch summary's te laten aanmaken. Dit betekent dat Discoverer de performance van query's verbetert door het creëren van een of meerdere materialized views in de database. Discoverer admin geeft hiervoor adviezen. Bij het gebruik van VPD blijkt echter dat Discoverer de policy niet ziet en alleen zijn eigen opgebouwde query bekijkt en analyseert. De gecreëerde materialized views bevatten daarom niet de gewenste kolom LABEL. De query's van eindgebruikers lopen echter wel over de kolom LABEL heen. Zij kunnen daardoor nooit gebruik maken van deze materialized views en de daaraan gerelateerde performance winst. De oplossing is om de summary's, nadat Discoverer deze heeft bepaald, handmatig aan te passen door het toevoegen van de kolom LABEL.

De combinatie van VPD met materialized views en Oracle Discoverer is echter wel een uitdaging voor de DBA. De door Oracle Discoverer gegenereerde SQL kan verschillen per (sub)versie. Het gedrag van de query rewrite in combinatie met VPD en verschillende platformen verschilt bovendien. De verkeerde combinaties leiden tot de beruchte Oracle internal error 'ora-600 ...'. Bij het upgraden van de verschillende onderdelen is het noodzakelijk dit goed te testen. Test dit het liefst in een vroeg stadium uit, niet alleen in de ontwikkel/test omgeving maar ook op het uiteindelijke productieplatform.



Figuur 4. Met behulp van labels wordt geïdentificeerd welke gebruikers(groep) welk gegeven mag zien.

Conclusie

- VPD levert een adequate beveiliging op data niveau op. Autorisatie hoeft niet meer in tools worden verstopt waardoor alle tools automatisch gebruik maken van het correcte beveiligingsniveau.
- Oracle Label Security levert adequate beveiliging op row level op. Het bijbehorende autorisatie/organisatie model (relatie gebruiker / label) moet echter wel passen. Het is vaak eenvoudiger een eigen model te implementeren.
- Let op de performance aspecten van de policy functie.
- Zorg dat een database export/import correct werkt door deze in de vorm van een rol op te nemen in de policy functie.
- Test tijdig de combinaties van de te gebruiken producten / platformen.

Ben Binnendijk
is werkzaam als managing consultant bij Cap Gemini Ernst & Young, gespecialiseerd in data warehousing en business intelligence. Hij vervult daarbij verschillende rollen, waaronder architect en projectleider (e-mail: ben.binnendijk@cgey.nl).

Herald ten Dam
is werkzaam als senior consultant bij Cap Gemini Ernst & Young en gespecialiseerd in Oracle producten (e-mail: aldam@cgey.nl).