Inrichten van een Oracle-database

Deel 2: Scrips voor het installeren

Het installeren, inrichten en configureren van een Oracle-database is een lastige klus. In een serie van drie artikelen laten we zien hoe deze inrichting op een gestructureerde manier vorm kan worden gegeven. In het tweede deel aandacht voor het installeren van scripts.

Dit deel bevat een beschrijving en voorbeelden van de scripts voor de transformatie van het fysieke datamodel naar de fysieke database. Het is gebaseerd op een Oracle-omgeving, maar kan gemakkelijk worden omgezet naar andere databaseproducten. Daarnaast zal het onderstaande aangepast moeten worden aan lokale wensen. Verder is het gebaseerd op de in het vorige deel beschreven naamgevingstandaard. Als daar een eigen invulling van gemaakt is, dan heeft dat ook consequenties voor de hierna beschreven scripts.
We veronderstellen hier dat er al een databaseserver is geïnstalleerd en, voor zover nodig, geconfigureerd voor de betreffende toepassing. Het installeren en configureren van een server valt buiten de scope van dit artikel.
Hierna volgt een beschrijving van de scripts om de objecten van elk type (tablespaces, rollen, users, tabellen, etcetera.) aan te maken. Aan het einde volgt een superscript om al deze scripts na elkaar te draaien. Dit superscript begint met enkele scripts om deze objecten eerst (in een omgekeerde volgorde) te verwijderen. Dan is het script herstartbaar. Zeker in de beginfase van een project zal het voorkomen dat men de inrichting nog een paar keer wil overdoen. Dat kan met dit superscript met een druk op de knop, nadat natuurlijk eerst de gewenst aanpassingen aan de objectscripts zijn gemaakt.

Er is een zipfile met een voorbeeld van deze scripts en het superscript beschikbaar via de website van Optimize (http://optimize.nl/Het-Blad/Optimize/Extra).

Tablespaces en datafiles

De eerste stap is het reserveren van diskruimte specifiek voor dit systeem/schema. Maak voor elke applicatie specifieke tablespaces en gebruik hiervoor niet de standaard tablespaces zoals USERS, TOOLS of SYSTEM. Dit vereenvoudigt het beheer sterk. Er worden standaard twee tablespaces voor respectievelijk tabellen en indexen gemaakt met naamgeving zoals we beschreven in het vorige deel. Als er meer tablespaces gewenst zijn, bijvoorbeeld voor gepartitioneerde tabellen of voor large objects, dan wordt het script hiervoor uitgebreid. In verband met de iets andere indeling van de directorynamen en bestandsnamen in Windows en Unix zijn er twee voorbeeldscripts:

- Voor UNIX: kas_110_cre_tablespaces_unix.sql ;
- Voor Windows: kas_110_cre_tablespaces_windows.sql .

Rollen

Maak altijd tenminste drie standaardrollen:

- Een rol voor de schema-eigenaar;
- Een rol voor de gebruikers die alleen gegevens mogen lezen;
- Een rol voor de gebruikers die ook gegevens mogen wijzigen (insert, update, delete).

Aanvullende specifieke rollen, bijvoorbeeld voor een application server of voor individuele functies/functieprofielen, kunnen worden toegevoegd als dat gewenst is voor de betreffende applicatie.
Zie voor een voorbeeldscript: kas_120_cre_roles.sql

Gebruikers

Maak een gebruiker aan voor de schema-eigenaar en aanvullende (standaard of specifieke) gebruikers, zoals gewenst voor de applicatie. Verleen aan deze gebruikers de gewenste rollen en andere privileges. Deze gebruikers/user id's kunnen direct aan een persoon zijn gekoppeld of bijvoorbeeld aan een applicatieserver. Het is niet gewenst om een user id voor een functie of functieprofiel te maken. Dan kan beter voor het functieprofiel een rol gemaakt worden. De persoon wordt weer aan deze rol of rollen gekoppeld.
Zie voor een voorbeeld script: kas_130_cre_users.sql

Sequences

Maak een script voor de sequences. Het is soms ook mogelijk

om de sequences in het CASE tool (PowerDesigner of ander gereedschap) te definiëren. Deze worden dan automatisch meegenomen in het volgende script. Een separaat script is hier dan niet meer nodig.
Zie voor een voorbeeld script: kas_200_cre_sequences.sql

Creëer tabellen, indexen en constraints

Hierin worden alle tabellen, indexen, etcetera van het betreffende datamodel (het schema) gecreëerd. Dit script wordt volledig gegenereerd door het CASE-tool (PowerDesigner, Oracle Designer, etcetera.). Hierin mogen geen wijzigingen worden aangebracht, behalve het standaard commentaarblok aan het begin van het script. De reden: als er wijzigingen op het datamodel nodig zijn, dan kunnen die in de CASE-tool worden aangebracht en kan het script opnieuw worden gegenereerd. Zijn er aspecten die niet in de CASE-tool zijn gedefinieerd en daarna gegenereerd kunnen worden, dan is het beter om die (middels een ALTER TABLE) in een separaat script na dit script uit te voeren. Dit geldt bijvoorbeeld voor de spatial indexen bij gebruik van Oracle spatial of voor de partities van een gepartitioneerde tabel, etcetera. Op deze manier kan, bij een wijziging in het model, het script opnieuw gegenereerd worden en het oude gegenereerde script vervangen.

Dit script bevat de definitie van:
- Sequences, indien gedefinieerd in de CASE-tool;
- Tabellen;
- Indexen;
- Primary- en foreign key constraints;
- Mogelijk andere constraints die in de CASE tool gedefinieerd zijn.

Zie voor een voorbeeld script: kas_210_cre_tables.sql

Creëer extra constraints

Dit script bevat de constraints die handmatig (niet in de CASE-tool) worden geschreven.
Zie voor een voorbeeldscript: kas_220_cre_constraints.sql

Views

Dit script bevat de views. Als er veel views zijn kan er ook een script per view worden gemaakt met een extra script waarin al deze view scripts worden aangeroepen.
Zie voor een voorbeeldscript: kas_230_cre_views.sql

Initiële inserts

Dit script bevat de initiële vulling van tabellen met de gegevens die aanwezig moeten zijn bij de start van het systeem. In ons voorbeeld is dit de vulling van de tabellen:
- SYSTEEM_CONSTANTEN (een tabel waarin altijd 1 (en slechts 1) rij staat met systeemconstanten of parameters);
- BTW_KLASSE en BTW_PERCENTAGE.

Andere voorbeelden kunnen zijn:
- Een tabel waarin volgnummers worden bijgehouden;
- Een tabel met de code en tekst van foutboodschappen of andere vaste teksten die door de programma's gebruikt kunnen worden;
- Een of meer tabellen met vast gedefinieerde codes met omschrijving, bijvoorbeeld voor statussen, landencodes, codes als M (Man) en V (Vrouw), etcetera.

Voor enkele van deze tabellen hoeven dan geen onderhoudsfuncties meer gemaakt te worden. Het eventuele onderhoud dat later toch nodig blijkt moet dan (met een change procedure?) met SQL inserts statements en SQL*Plus door een beheerder worden gedaan. Maar ook als er wel een onderhoudsprogramma voor is dan heeft een initiële vulling nog voordelen: De ontwikkel-, test- en productieomgeving beginnen allemaal met dezelfde startsituatie waardoor verrassingen (het vergeten van de vulling van een essentiële code tabel) niet meer voorkomen.
Zie voor een voorbeeldscript: kas_310_initial_inserts.sql

Grants

Dit script bevat een PL/SQL procedure die grant statements genereert en uitvoert. In het voorbeeld worden voor elke tabel een twee statements gegenereerd:

```
Grant select on <tabel> to <Lees_rol>;
Grant all on <tabel> to <Wijzig_rol>;
```

Dit betreft natuurlijk de rollen die hiervoor al zijn gemaakt.
Zie voor een voorbeeldscript: kas_410_cre_grants.sql

Synoniemen

Dit script bevat een procedure die een create public synonym statement genereert en uitvoert voor elke tabel en voor elke view. Hierdoor kunnen alle andere gebruikers van het systeem deze tabellen gebruiken zonder telkens de schema owner voor de tabel aan te geven:

```
select * from klant;
```

in plaats van:

```
select * from KAS_DBA01.klant;
```

Door public synonyms te maken is het niet meer nodig om de synoniemen voor elke gebruiker te maken. Dit zou extra veel beheerwerk vergen bij nieuwe of vervallen tabellen en nieuwe of vervallen gebruikers. Wanneer er meer systemen op een databaseserver staan, kan er een naamgevingconflict ontstaan, bijvoorbeeld als beide systemen een tabel KLANT hebben. In dat geval kan moet men besluiten om alle synoniemen voor alle systemen te laten voorafgaan door een prefix met de applicatie- of schema code:

```
Create public synonym kas_klant for KAS_DBA01.klant;
```

Zie voor een voorbeeldscript: kas_420_cre_synonyms.sql

Het idee is dat de schema-eigenaar gebruiker (KAS_DBA01) alleen wordt gebruikt om het schema te onderhouden en niet voor reguliere queries of updates. Daarom wordt deze gebruiker altijd onmiddellijk na het maken van het schema gelockt en alleen tijdelijk ge-unlockt bij wijzigingen op het schema, bijvoorbeeld voor nieuwe tabellen of voor nieuwe of gewijzigde procedures of views.

Databasekoppelingen

Deze scripts worden gebruikt voor het maken van de databasekoppelingen (verbinding met andere databases). Hiervan is geen voorbeeld gemaakt.

Triggers, procedures en functies

In dit script wordt de procedurele code, zoals voor triggers, packages, procedures en functies, gedefinieerd. Als er veel procedurele code is, dan kan er ook een script per object worden gemaakt met een extra script waarin al deze scripts worden aangeroepen. Hiervan is geen voorbeeld gemaakt.

Super script dat alle scripts draait

Tenslotte is er een script dat alle hiervoor genoemde scripts in één run uitvoert. Dit script zal achtereenvolgens:

- Inloggen als system (of een andere ADMIN gebruiker);
- Verwijderen van de public synoniemen die in een vorige run zijn aangemaakt;
- Verwijderen van de users, met name de user die eigenaar van het schema is. Hierdoor worden al zijn objecten (tabellen, views, triggers, procedures, etcetera.) automatisch verwijderd;
- Verwijderen van de rollen;
- Verwijderen van de tablespaces. Hier is gekozen om wel de tablespaces, maar niet de datafiles te verwijderen zodat deze in de volgende stappen hergebruikt kunnen worden;
- Aanmaken van de tablespaces;
- Aanmaken van de rollen;
- Aanmaken van de users;
- Inloggen als de schema eigenaar;
- Aanmaken van de sequences, tabellen en alle andere objecten van de hiervoor genoemde scripts;
- Genereren van testdata;
- Draaien van enkele verificatie scripts;
- Inloggen als system en locken van de schema eigenaar.

Dit script kan telkens opnieuw worden gedraaid na wijzigingen in het schema in verband met fouten of nieuwe inzichten. Het verwijdert daarmee ook de testdata en genereert deze opnieuw. Als een tester eigen testdata nodig heeft, dan zal hij het script waarin deze worden toegevoegd aan de gegenereerde testdata zelf opnieuw moeten uitvoeren. Bij heel veel testdata kan dit erg lang duren.
NB Bij de eerste run zullen er foutmeldingen komen in het eerste deel, omdat de te verwijderen objecten nog niet bestaan.

Bij wijzigingen in views of in de procedurele code, bijvoorbeeld bij nieuwe of gewijzigde procedures of triggers, is het runnen van deze hele procedure overkill. Dan kan beter alleen het betreffende script worden uitgevoerd. Maar het voordeel van deze werkwijze is het snel en gestandaardiseerd inrichten van een ontwikkel of test omgeving en de initiële productie omgeving.
Wat als twee ontwikkelaars of testers beide een eigen versie van het systeem willen hebben? Dat kan in elk geval door elk een eigen database instance te geven, maar vaak is dat ongewenst vanwege de grote overhead per instance op de server en eventueel ook i.v.m. extra licentiekosten. Een tweede mogelijkheid is de naam van de schema eigenaar te parametriseren: kas_dba01, kas_dba02, etcetera. Elk heeft dan zelf het volledige schema maar zij maken gebruik van dezelfde rollen en optioneel ook van dezelfde tablespaces. Alleen het maken van public synonyms is niet mogelijk, omdat er dan een naamconflict ontstaat. In dat geval

- kan de ontwikkelaar alles testen onder de userid van de schema eigenaar, tenslotte: het schema is ook specifiek voor hem/haar gemaakt;
- of kunnen er specifieke private synoniemen gemaakt worden.

Het opruimen van een systeem kan door het eerste deel van dit script uit te voeren. Daarna moeten nog wel de datafiles worden opgeruimd.
Zie voor een voorbeeld script het kader of: kas_000_create_script.sql

De doorlooptijd van dit script is in de orde van 10 seconden met Oracle 11g op een windows PC.

In het volgende deel

In dit deel hebben we een standaardinrichting voor een Oracle-database of schema beschreven met tablespaces, rollen, users, tabellen, etcetera. In het volgende deel beschrijven we hoe we de tabellen kunnen vullen met een kleine of grote hoeveelheid testdata. Dit gaat over het genereren van testgegevens.

Toon Loonen is werkzaam bij Capgemini. Hij is gespecialiseerd in (logisch en fysiek) gegevensmodellering en is bereikbaar via e-mail: toon.loonen@capgemini.com of toon.loonen@inter.nl.net.