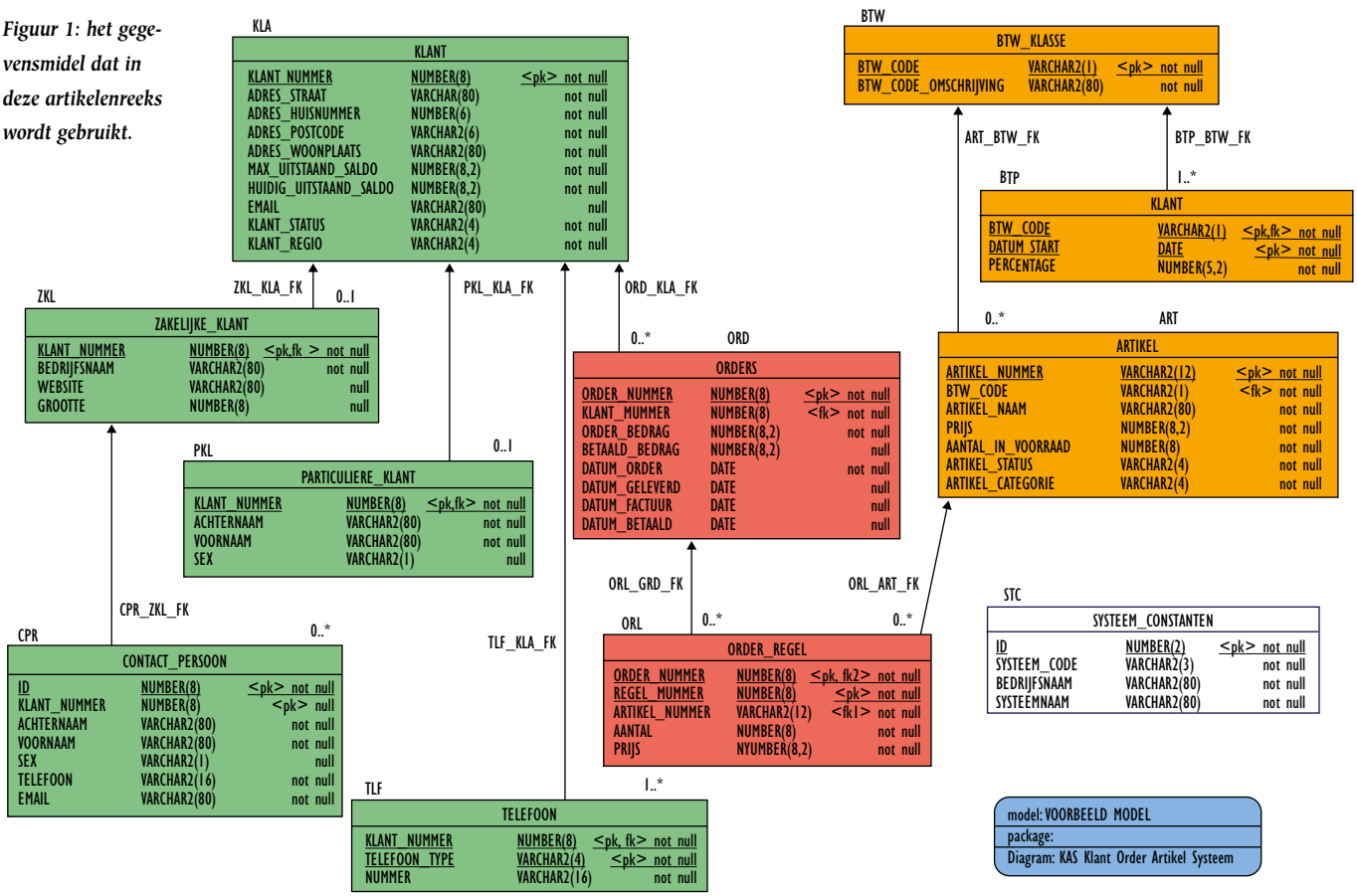
Inrichten van een Oracle-database

Deel 1: Naamgevingstandaard

Het installeren, inrichten en configureren van een Oracle-database is een lastige klus. In veel projecten wordt hierover telkens weer opnieuw nagedacht met een grote kans dat hierbij fouten gemaakt worden die later lastig te herstellen zijn. In een serie van drie artikelen laten we zien hoe deze inrichting op een gestructureerde manier vorm kan worden gegeven. In het eerste deel behandelen we de naamgevingstandaard.

Er zijn al diverse naamgevingstandaarden te vinden op internet. Google even op de termen “Oracle naming standard” en je vindt er diverse. We hebben hier toch gekozen om er nog een aan toe te voegen en wel om de volgende redenen:

- Er is een standaard nodig als uitgangspunt voor de in deel 2 beschreven inrichting;
- De meeste standaarden gebruiken geen korte code voor tabel of systeem. Deze is nodig voor een gestructureerde naamgeving van gerelateerde componenten (tablespaces bij een database, relaties bij een tabel);
- De meeste standaarden zijn minder uitgebreid als deze. Ze beschrijven bijvoorbeeld niet de componenten als systeem of domein.



Figuur 1: het gegevensmidel dat in deze artikelenreeks wordt gebruikt.

Deze standaard beschrijft geen naamgeving binnen PL/SQL code.

Uitgangspunt is dat het gegevensmodel zelf in een data dictionary wordt vastgelegd en dat vandaar uit de scripts voor de tabellen, indexen en de primary- en foreign key constraints worden gegenereerd. De scripts voor de andere componenten (tablespaces, users, rollen, views, procedures, etc.) zullen grotendeels handmatig worden geschreven. Deze scripts beginnen allemaal met een standaard stuk commentaar.
Het model in figuur 1 wordt gebruikt bij alle voorbeelden in deze artikelenreeks. Het is opgesteld met Power Designer maar dit had ook Oracle Designer, JDeveloper, Enterprice Architect of een ander CASE tool kunnen zijn.

Toelichting op het model

Het gegevensmodel in figuur 1 is bedoeld als voorbeeld voor deze artikelenreeks. Het pretendeert zeker niet de volledige functionaliteit voor een dergelijk systeem te bevatten. Het bevat wel:

- Een tabel met artikelen;
- Een tabel met BTW codes en percentages; van de BTW wordt het percentage historisch bijgehouden;
- Een tabel met klanten;
- Een klant kan een particuliere of een zakelijke klant zijn; hiervoor worden in twee extra tabellen de specifieke gegevens van deze twee types vastgelegd;
- Een tabel met contactpersonen die bij een zakelijke klant horen;

Object	Naamgeving standaard	Voorbeeld
Systeem	Logische naam, bijvoorbeeld “Klant, Artikel en order systeem”. Het is nuttig om een 3 karakter code of mnemonic voor het systeem te maken of (alternatief) een 2 karakter code voor het systeem en een extra teken voor het deelsysteem.	KAS: Klant, Artikel en Order Systeem
Data model	Dezelfde naam als voor het systeem.	KAS
Tabel (Entiteit)	Logische naam in een enkelvoud, bijvoorbeeld: “klant” (geen “klanten”). NB Voor de tabel ORDERS is een uitzondering gemaakt en toch het meervoud ORDERS genomen omdat order al een reserved word is in SQL: Select * from order order by 1; Het is nuttig om een 3 karakter code voor elke tabel te maken: ORD voor orders, ORL voor orderregel, KLA voor Klant. De code van elke tabel is te zien in het voorbeeld model en boven elke tabel aangegeven.	Klant KLA
Domein of Datatype	Logische naam van het datatype. [zie ook Ref 2 en 5] Bij domein of datatype wordt op 1 plaats de definitie van een begrip, zoals de postcode vastgelegd, inclusief datatype (6 posities alfanumeriek) en controle (altijd 4 cijfers en 2 hoofdletters). Deze definitie wordt overgenomen bij elke kolom van dat type. Dit voorkomt redundantie in het model. Oracle ondersteunt geen domeinen in de database maar in de CASE tool die voor dit artikel / model is gebruikt is hiervan wel gebruik gemaakt.	Postcode
Kolom (Attribuut)	Logische naam. Vaak kan de naam van het domein worden gebruikt. De naam van de tabel wordt hier niet herhaald: de kolom “naam” voor de naam van de klant in de tabel klant, wordt gewoon “naam” en niet “klant_naam” of “klantnaam”. Het is duidelijk dat het om de naam van de klant gaat. In de programmatuur (SQL) kan altijd “klant.naam” worden gebruikt; Wanneer een domein vaker wordt gebruikt in een tabel, dan kan er een prefix of postfix aan worden geplakt: “voornaam”, “achternaam” of “datum_order”, “datum_levering”.	Naam Voornaam Datum_order Order_datum
Relatie	De code / mnemonic van de verwijzende (meer) tabel, een onderstreep teken, de code van de andere (een) tabel en ‘_FK’. Als er tussen twee tabellen meer relaties zijn, dan wordt er in de naam een verduidelijking van deze relatie gegeven. Bijvoorbeeld als in de orderregel, naast het artikelnummer ook een vervangend artikelnummer gestaan had (te leveren als het eerste niet geleverd kon worden) dan waren er twee relaties van orderregel naar artikel met namen als hiernaast.	ORD_KLA_FK ORL_ART_FK ORL_ART_VERV_FK
Primary key	De code van de tabel gevolgd door “_PK” [Zie ook Ref 6].	KLA_PK
Unieke key	De code van de tabel gevolgd door “_UK01”, “_UK02”, ..	KLA_UK01
Foreign key kolom	De naam van de foreign key kolom is gelijk aan de naam van de primary key kolom waarnaar wordt verwezen, eventueel, zoals ook hiervoor bij de foreign keys, aangevuld met een verduidelijking als er meer relaties zijn.	Artikel_nummer Artikel_nummer_verv
Functies	Alhoewel functies en schermen geen, direct aan de database gerelateerd objecten zijn, is het zinvol om ook hiervoor een codering af te spreken. Voor functies bijvoorbeeld de systeemcode en een volgnummer van 4 posities waarmee een groepering kan worden aangegeven.	KAS1000 KAS2301
Scherm	De schermen of andere componenten binnen een functie krijgen een code gelijk aan de functie waar ze bij horen en een extra letter.	KAS2301A KAS2301B
Routines	Algemene routines die door veel functies worden gebruikt krijgen een code als hiervoor maar met een serienummer tussen 9000 en 9999 en als letter een Z.	KAS9901Z

Tabel 1.

- Bij een klant kunnen enkele telefoonnummers worden vastgelegd voor bijvoorbeeld een vaste telefoon, een GSM en een FAX;
- Een tabel met orders en orderregels; deze hebben relaties met klanten resp. artikelen;
- Een tabel met systeemconstanten, bijvoorbeeld de systeemnaam en bedrijfsnaam.

Het voordeel van het opnemen van de bedrijfsnaam in de tabel met systeemconstanten is dat deze in de kop van rapporten getoond kan worden en dat bij wijziging van de bedrijfsnaam deze alleen in de tabel behoeft te worden aangepast en niet alle rapporten gewijzigd moeten worden. Ook kan het systeem zo voor meer bedrijven gebruikt worden.
Er mag altijd maar één record in deze tabel staan. Daarom is er een primary key (dus een unieke waarde) gedefinieerd die alleen de waarde 1 mag hebben. Het inserten van een tweede record is dan niet meer mogelijk.
Van alle scripts voor het maken van het databaseschema kan een zipfile met de voorbeelden worden gedownload via de website van Optimize (http://optimize.nl/Het-Blad/Optimize/Extra).
Voordat we een gestandaardiseerde werkwijze kunnen beschrijven voor de inrichting van een database of

Object	Naamgeving standaard	Voorbeeld
Database	De code van het systeem (bijvoorbeeld "KAS") Plus 1 karakter voor het omgeving: D voor ontwikkeling (programmering en unit test) T voor (integratie) test A voor de acceptatietest P voor de productie, etc. voor mogelijk andere omgevingen. Plus eventueel een volgnummer als meer dan één database nodig is, bijvoorbeeld voor elke ontwikkelaar of tester een eigen database. Als een database voor meer systemen gebruikt wordt, met per systeem een schema, dan zal er voor de eerste 3 posities van de databasenaam een andere, meer algemene code gebruikt worden en wordt de systeemcode (KAS) gebruikt voor het schema.	 KASD01 KAST01, KAST02 KASA01 KASP
Tablespace	Systeem code plus 1 teken voor: • D voor gegevens • X voor de indexen plus een volgnummer van 2 posities indien meer dan één tablespace nodig is of nodig kan worden in de toekomst. Bij grote gepartitioneerde tabellen met bijvoorbeeld een partitie voor elke dag, week of maand, kan een tablespace gemaakt worden voor de (dag, week of maand) partities van één maand van alle tabellen. Men moet dan jaar en de maand toevoegen aan de naam van de partitie met de gegevens voor die maand. Als per maand en per tabel een tablespace gemaakt wordt, dan moet ook de code van de tabel aan de naam worden toegevoegd [Zie ook Ref 7].	KASD01 KASX01 KASD200904 KASX200904 KASD200904ORL KASX200904ORL
Datafile	Naam van de tablespace plus "_d" en een volgnummer van 2 positie plus achtervoegsel ".dbf" Het geheel in kleine letters.	kasd01_d01.dbf
Index	Algemeen: De code van de tabel plus "_" plus een suffix: Primaire sleutel index: suffix = PK Unieke index: suffix = UK plus 2 positie nummer: UK01, .. Andere index: suffix = IX plus 2 positie nummer: IX01, .. Indexen op een foreign key krijgen dezelfde naam als de relatie plus de toevoeging "_I".	KLA_??01 KLA_PK KLA_UK01 KLA_IX01 ORD_KLA_FK_I
Trigger	Tabel code plus toevoeging _T plus 2 posities voor het type trigger (BU = Before Update) plus 2 positie volgnummer indien nodig	KLA_TBU
View	Als de view specifiek gebaseerd is op een tabel, dan de tabelnaam plus achtervoegsel _VW plus 2 positie volgnummer, anders een logische naam.	Klant_VW02 Telefoon_lijst
Procedure	Voorvoegsel "PR_" plus een logische naam.	PR_insert_order
Package	Voorvoegsel "PK_" plus een logische naam.	PK_verwerk_order
Function	Voorvoegsel "FN_" plus een logische naam.	FN_uitstaand_saldo
Sequence	Specifieke sequence voor een tabel: "SQ_" plus tabel code. Algemene sequence: Voorvoegsel "SQ_" plus een logische naam.	SQ_KLA SQ_bericht_nummer
Constraint op tabel of kolom	Voorvoegsel "CN_" plus tabel code plus kolomnaam of andere logische naam.	CN_KLA_sex CN_KLA_datum_volgorde

Tabel 2.

databaseschema, moeten we eerst een aantal afspraken maken over naamgeving. Zonder een duidelijke standaardnaamgeving is er namelijk geen standaardinrichting mogelijk.
Het is een mogelijke standaard. Afwijken van deze standaard is zeker mogelijk, als deze afspraken maar worden vastgelegd en nageleefd. Case tools hebben vaak eigen standaarden voor namen van bijvoorbeeld primary- en foreign keys. Het kan werk besparen door deze standaard gewoon over te nemen. Andere afspraken hebben mogelijk ook consequenties voor de inrichting van de schema's of database.

NB: Het geheel is gebaseerd op een Oracle-omgeving, maar kan gemakkelijk worden omgezet naar andere database producten.
NB: We gebruiken hier de termen tabel en kolom. Die komen overeen met de termen entiteit en attribuut in het logisch model.

Logische elementen

Voor tabellen, domeinen of datatypes en kolommen wordt vaak een naam gebruikt die uit meerdere woorden is samengesteld. Deze woorden worden gescheiden door een onderstreep teken als in enkele voorbeelden hiervoor. De naam van deze objecten zal vaak gelijk zijn aan de naam van dit object in het logisch model behalve:

- In het logisch model zijn spaties en enkele andere tekens toegestaan die in het fysiek model verboden zijn. Deze worden in het fysiek model vervangen door onderstreep tekens;
- In het logisch model mag de entiteitnaam langer zijn dan 30 posities; in dat geval moet de fysieke tabelnaam hiervan worden afgeleid door deze korter te maken.

We gebruiken voor het systeem of voor een entiteit/tabel, naast de naam, ook altijd een code of mnemonic van drie posities. Afhankelijk van de eisen van de omgeving (veel of weinig systemen) kan er ook voor gekozen worden om altijd een code van vier posities of altijd een code van twee posities te gebruiken. Deze code moet zeker uniek zijn binnen een schema, maar liever ook over de schema's heen binnen een database.

Fysieke elementen

Speciale tabellen, bijvoorbeeld werktabellen of tijdelijke tabellen met tussentijdse resultaten hebben een naam die begint met een voorvoegsel (bijv. WK_) en vervolgens een logische naam of de naam van de functie waarvoor deze tabel nodig is (WK_KAS2301).
Gebruik van hoofdletters of kleine letters is erg afhankelijk van de gewoonten binnen een organisatie. Ik geef de voorkeur om alle items en codering in kleine letters, maar mnemonics, codes en andere afkortingen in hoofdletters te zetten.
Gebruik geen speciale tekens in tabel- of kolomnamen zoals spatie, of "-,.!@#%&*()+=| \ /? ".

Gebruik nooit versie-informatie in de database of schema namen, tenzij er gelijktijdig verschillende versies van het systeem onderhouden moeten/kunnen worden.
Voorkom het gebruikt van namen die gereserveerde woorden zijn (of zou kunnen worden). Bijvoorbeeld: in ons voorbeeld data model hebben we de tabel ORDERS ipv (enkelvoud) ORDER. Dit omdat het woord ORDER ook wordt gebruikt in het SQL select statement:

```
select * from order order by datum_order;
```

Deze query zal een foutmelding geven. Het kan wel met:

```
select * from "order" order by datum_order;
```

De tabel naam is hierbij ook hoofdlettergevoelig!
Beter is om deze situatie te voorkomen. Door toevoeging van een prefix voor elke tabel (KAS_ORDER, KAS_ARTIKEL, WK_KAS2301_FACTUUR_REGEL, etc) zal de tabelnaam nooit meer een gereserveerd woord worden.

Algemene regels voor het maken van codes of mnemonics voor lange namen:

- Gebruik de eerste letter van elk woord in een samengestelde naam;
- Verwijder klinkers, maar niet als dit het eerste teken van de naam is;
- Verwijder enkele laatste karakters van elk woord of lettergreep.

Enkele Voorbeelden:
NB: BTW wordt niet afgekort, het is al 3 posities en al een afkorting.

In dit deel hebben we een mogelijke naamgevingstandaard beschreven voor een Oracle-omgeving. Deze wordt weer gebruikt in het volgende deel waar we een standaardinrichting voor een Oracle-database of schema beschrijven.

Lange (volledige) naam	Korte naam	Code of Mnemonic
Zakelijke klant	zkl_klnt	ZKL
Contact persoon	cnt_prsn	CPR
Telefoon	tlfn	TLF
BTW Percentage	btw_prc	BPC

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.