Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

Technisch Ontwerp Databases

Versie 0.51

Datum 27-6-2017

Status Definitief

Documenthistorie

Datum	Versie	Beschrijving	Auteur
	0.1	Oorspronkelijke naam 'TO BRP Database'.	Operatie BRP
20-3-2017	0.2	Diverse aanpassingen gedaan i.v.m. inrichting oplevering IV.	Operatie BRP
11-4-2017	0.3	Verdere aanvulling van nieuwe onderwerpen.	Operatie BRP
19-4-2017	0.4	Unaccent beschreven	Operatie BRP
31-5-2017	0.5	Elementtabel beschrijving	Operatie BRP
27-6-2017	0.51	Kosmetische aanpassingen n.a.v. review I&T	Operatie BRP

Reviewhistorie

Versie	Reviewer
0.2	Operatie BRP. Controle op volledigheid en aansluiting bij overige documentatie.
0.4	Operatie BRP. Controle op richtlijnen.

Inhoudsopgave

1 Inleiding

1.1 Doel

Dit document bevat een verzameling technische onderwerpen die allemaal betrekking hebben op de PostgreSQL databases van de BRP. De onderwerpen lopen uiteen van richtlijnen tot een beschrijving waarom bepaalde keuzes gemaakt zijn.

De beschrijving van de structuur van het gegevensmodel waar de database op gebaseerd is, is geen onderdeel van dit document. Zie hiervoor het de Toelichting op het gegevensmodel [1], het architectuurdocument [2] en de diverse functionele aspectbeschrijvingen.

1.2 Voorkennis

Dit document vereist kennis over concepten die gebruikelijk zijn binnen RDBMS systemen en in het bijzonder enige kennis over de specifieke (on)mogelijkheden van PostgreSQL.

1.3 Referenties

Nr.	Documentnaam	Organisatie	Versie	Datum
1	Architectuur Basisregistratie Personen	Operatie BRP	-	-
2	Toelichting Gegevensmodel	Operatie BRP	-	-

2 Databases en schema's

2.1 Databases

De gegevens van de BRP zijn verdeeld over drie databases: BRP, Berichten en Protocolleren. Deze drie databases hebben andere gebruiks- en opslagkarakteristieken. Mede omwille van de schaalbaarheid zijn de gegevens daarom verdeeld over drie aparte databases.

Het kan in de toekomst (mogelijk) zinvol zijn om de technische inrichting per database anders uit te voeren. Hierbij valt te denken aan andere replicatie instellingen of zelfs een ander type database.

Overzicht van de databases:

Database	Afkorting	Doel
BRP	BRP	Alle gegevens ten behoeve van de kern functionaliteit bijhouden en leveren.
Berichten	Ber	Gegevens inzake de inkomende en uitgaande berichten (Berichtarchief).
Protocolleren	Prot	Registratie van geleverde gegevens ten hoeve van protocollering.

De databases Berichtarchief en Protocolleren worden anders behandeld dan de BRP-database. In tegenstelling tot de BRP-database zullen deze databases een hoge mutatiegraad kennen. Iedere bijhouding zal leiden tot een toevoeging aan het Berichtarchief en iedere levering zal leiden tot een toevoeging aan zowel het Berichtarchief als het Protocol.

Om deze reden wijken de tabellen in deze databases op een aantal gebieden af van de tabellen in de BRP-database. Zo is er maar een beperkt aantal indexen en foreign keys gedefinieerd. Ook bevatten niet alle tabellen primary keys.

In de toekomst zou het kunnen zijn dat de gegevens die nu in deze twee databases zitten op een andere wijze opgeslagen worden om de gewenste performance te kunnen/blijven halen. Hierbij valt de denken aan een NoSQL oplossing. Omdat de huidige performance nog geen aanleiding geeft om deze stap te zetten, is ervoor gekozen om het aantal gebruikte opslag technologieën te beperken. Daarom worden deze gegevens gewoon in een PostgreSQL database opgeslagen.

2.2 Schema's

Onderstaande tabel toont de schema's, het gegevensgebied waarop ze betrekking hebben en een ruwe inschatting van de omvang alsmede eventuele specifieke eigenschappen. In het document Toelichting op het gegevensmodel [2] wordt een nauwkeuriger duiding van de inhoud gegeven.

Database. Schema	Gegevens-gebied	Ruwe schatting omvang, specifieke eigenschappen
BRP.Kern	Kern	Trage groei, omvang ca. 600GB, grootste tabel 500 miljoen rijen.
BRP.AutAut	Autorisatie & Authenticatie	Beperkte omvang. Aantal leveringsautorisaties is ruwweg evenredig met het aantal afnemers. Omvang grootste tabel in de orde van enkele tienduizenden rijen. Het aantal afnemerindicaties is significant, huidige schatting is meer dan een half miljard records.
Ber.Ber	Bericht	Grote omvang; Lineaire groei in de tijd (groei verwacht naar enkele miljarden per jaar).
BRP.Conv	Conversie	Klein, configuratieachtige tabellen, met een klein aantal rijen.

BRP.IST	Interstelsel tabellen	Groot, vergelijkbaar met het kernschema.
Prot.Prot	Levering / Protocolleren	Grote omvang; Lineaire groei in de tijd (groei verwacht naar enkele miljarden per jaar).
BRP.VerConv	Verantwoording conversie	Groot, vergelijkbaar met het kernschema.
BRP.MigBlok	Migratie blokkering	Klein, beperkt aantal rijen.
BRP.Beh	Beheer	Middelklein.

3 PostgreSQL versie en instellingen

De minimaal vereiste databaseversie is PostgreSQL 9.5.x. Hierbij staat de .x voor de meest recente minor 9.5 release.

De databases moeten aangemaakt zijn met, in ieder geval, de volgende instellingen:

Instelling	Waarde
Encoding	UTF-8
Collation (LC_COLLATE setting)	nl_NL.UTF-8
Character type (LC_CTYPE setting)	nl_NL.UTF-8

4 Benodigde opslagruimte

Een globale inschatting van de grootte van de databases. Er is onderscheid gemaakt tussen de het uitvoeren van de initiële vulling (IV) en de definitieve situatie na de IV.

Database	Tijdens IV	Definitief (Na IV)	Toelichting
GBA-V	500 GB	300 GB	Tijdens IV worden tussentabellen aangemaakt die 300 GB ruimte gebruiken. Daarnaast is er tijdelijk 200 GB aan temp space benodigd.
BRP	750 GB	750 GB	
Berichtarchief	?	?	
Protocolleren	100 GB	100 GB	

5 Indexen

5.1 Inleiding

De meeste indexen op de tabellen zijn (standaard) B-tree indexen. Indexen zijn aangelegd voor een aantal doeleinden:

- Indexen ten behoeve van navigatie door het datamodel. Op verwijzende gegevens, bijvoorbeeld de Persoon van Persoon \ Adres ligt een index om snel de adressen van een Persoon te vinden.
- Unique constraints dwingen de integriteit van de gegevens af. Zo mogen in de HIS tabellen niet meerdere feiten bij één gegeven op hetzelfde registratie tijdstip worden vastgelegd. Dit is de reden dat de index over twee attributen ligt; de verwijzing naar de B-laag en het tijdstip registratie. Voor veel zoekoperaties is de toevoeging van het tijdstip registratie niet zinvol. In het kader van opslag en snelheid kan in de toekomst het unique constraint vervallen.
- Indexen ten behoeve van bevraging. De veel gebruikte zoekpaden binnen bevraging zijn geïndexeerd om te garanderen dat de bevraging een goede responsetijd hebben.

5.2 Zoeken op diakritische tekens

Bij bevraging is de eis dat op een aantal attributen zoeken 'hoofdletter ongevoelig' en 'zonder diakrieten' kan. Om dit te bereiken zijn twee opties:

1. De attributen dupliceren, diakriet-loos in hoofdletters opslaan en op dit attribuut zoeken.
2. Expressie indexen gebruiken waarbij de index op een functie resultaat wordt gelegd waarbij de functie het gegeven zonder diakritische tekens retourneert.

Er is gekozen voor de tweede oplossing omdat dit redundantie voorkomt. Dit betekent dat er ook altijd gezocht moet worden met een zoekvraag die gelijk is aan de index-expressie, anders wordt de index niet gebruikt. Zie hoofdstuk 15 voor meer informatie over het verwijderen van de diakritische tekens..

Dit type index wordt toegepast bij systeemregels van het type *Index UPPER*.

5.3 Indexen bij verwijzende gegevens

Als er attributen zijn die verwijzen naar een andere tabel (foreign key), dan wordt er niet altijd een index aangemaakt op het verwijzende attribuut. De index wordt niet aangemaakt als deze toch niet gebruikt zal worden vanwege het beperkt aantal unieke voorkomens van het verwijzende attribuut in verhouding met het totaal aantal voorkomens in de tabel. In die gevallen is het direct volledig lezen van de tabel sneller dan de gegevens eerst via de index te benaderen.

Er wordt een index gemaakt als het attribuut:

- in de BRP database zit en
    - verwijst naar een dynamische tabel (dat wil zeggen: geen Stamgegeven). Of
    - verwijst naar een dynamisch stamgegeven en
        - dit stamgegeven in het Kern schema zit en meer dan 10.000 tupels bevat of
        - dit stamgegeven niet in het Kern schema zit en meer dan 2.000 tupels bevat

5.4 Bijzonderheden bij historie

Bij de Historie tabellen (HIS) zijn ook een aantal indexen aangelegd ten behoeve van bevraging. Bij bevraging geldt dat er nooit gezocht wordt in de formele historie. Hier zijn dus alleen gegevens van belang die niet vervallen zijn. Het lijkt daarom logisch om een partiele index aan te leggen met als clausule `WHERE TsVerval IS NULL`. Dit zorgt voor indexen die (veel) kleiner zijn.

Nu blijkt dat PostgreSQL (versie 9.5) de selectiviteit van een index niet goed kan bepalen als de partiele index gelimiteerd wordt door een attribuut welke niet in de index zit. Er wordt dan een (hardcoded) selectiviteit van '0,5' gebruikt. Dit betekent; 0,5% van alle rijen voldoen aan dit criterium.

Een voorbeeld query:

```
SELECT …
FROM kern.His_PersSamengesteldeNaam
WHERE
   upper(brp_unaccent(geslnaamstam)) like 'JACOBI%'
   AND TsVerval IS NULL
```

Bij een partiele index met TsVerval IS NULL gaf de queryplanner aan dat hier 249.815 rows terugkwamen (=0,5% van de gehele populatie). Bij het leggen van een volledige index werd de schatting teruggebracht tot 1001 rows.

Met name bij complexere queries zorgde het (onrealistische) plan bij partiele indexen voor queries die een hoge cost hadden maar soms wel snel uitvoerden, maar soms ook, conform cost schatting, traag waren.

Er is daarom afgezien van het gebruik van partiele indexen om alleen de actuele formele records te indexeren.

Hetzelfde geldt voor de tabellen die een index hebben op een (zeer) kleine hoeveelheid records (b.v. Buitenlands adres regel 2). Hier snapt de optimizer ook niet dat de `WHERE <veld> IS NOT NULL` (of `WHERE brp_unaccent(<veld>) IS NOT NULL`) zeer selectief is en geeft het ook 0,5% van de gehele populatie.

Daarom worden er bij functie gebaseerde indexen gewoon volledige indexen gebruikt, ook al hadden deze een stuk kleiner gekund.

6 Storage parameters

6.1 Inleiding

Bij iedere tabel zijn zogenaamde storage parameters op te geven. Deze storage parameters baseren we op het (geschatte) aantal initiële tupels en mutaties per jaar.

6.2 Fillfactor

Vinden er geen mutaties (updates) plaats op een tabel, dan wordt de `FillFactor` op 100 gezet.

6.3 Autovacuum en autoanalyze

Om te voorkomen dat de autovacuum en autoanalyze te vaak draaien, zijn de `autovacuum_vacuum_scale_factor`, `autovacuum_vacuum_threshold` en `autovacuum_analyze_threshold` aangepast.

Situatie	Scale factor	Threshold
Geen mutaties	0,2	10% van initiële tupels
Tabel met meer dan 1.000.000 tupels	0,05	2% van initiële tupels
Meer updates dan inserts	0,0005	0,2% van initiële tupels

Voorbeeld: een tabel met initieel 20.000.000 tupels, 1.000.000 inserts per jaar en 2.000.000 updates per jaar.

Hier zal de eerste vacuum plaats vinden na 40.000 mutaties.

Met name de laatste situatie zorgt voor een vrij agressieve vacuum. Dit is gedaan omdat tabellen die voldoen aan dit criterium ook tabellen zijn waar veel op gezocht wordt (Administratieve handeling, Persoon). Veel ‘bloat’ in deze tabellen zal de werking van het systeem vertragen.

7 Partitionering

7.1 Inleiding

Een aantal tabellen is gepartitioneerd. Dit is om een aantal redenen gedaan:

- Snelheidsdegradatie bij inserts voorkomen. Tabellen hebben de neiging om trager te worden als de inhoud toeneemt, met name bij het gebruik van de B-Tree indexen.
- Eenvoud van wissen van gegevens. Een partitie is snel te verwijderen.
- Snelheid van selecteren van gegevens (mits op de juiste attributen gefilterd)

7.2 Protocollering en Berichtarchief

De Protocollering en Berichtarchiverings tabellen (Leveringsaantekening, Leveringsaantekening \ Persoon, Bericht, Bericht \ Persoon) zijn allen gepartitioneerd per kwartaal. Voor de '\ Persoon' tabellen betekende dit dat er een (afleidbaar) attribuut bij moest komen.

Alle gegevens van voor 1-1-2012 komen in één partitie (met de naam 2011). Vervolgens zijn er partities aangemaakt tot en met het 4e kwartaal 2021.

7.3 Administratieve handeling

De Administratieve handeling tabel is ook een tabel die voor partitionering in aanmerking zou kunnen komen. Hier is duidelijk een splitsing tussen administratieve handelingen die nog in levering zijn (Status levering = 1 of 3) en de overige administratieve handelingen die niet meer zullen muteren. Technisch heeft een partitie voor de 'in levering' handelingen grote voordelen omdat de mutaties dan in één kleine tabel plaats vinden. Dit is voor het vacuum proces een groot voordeel.

Nadeel van partitioneren is dat afdwingen van de integriteit van gegevens middels foreign keys niet meer kan. Controle op de verwijzingen naar Administratieve handelingen is dan niet meer mogelijk.

De integriteit van gegevens op deze Kern tabel weegt in dit geval op tegen de nadelen. Met name omdat er een aantal partiele indexen gemaakt zijn op het Status levering attribuut. Met de aangepaste vacuum instellingen op deze tabel om de hoge mutatiegraad te ondersteunen, zou dit geen beperking meer moeten zijn. Daarnaast zijn er een aantal significante wijzigingen te verwachten in toekomstige PostgreSQL versies waarbij het vacuum proces efficiënter werkt.

8 Numerieke codes:

Het gegevensmodel kent een aantal attributen van het type Numerieke code. Dit zijn bijvoorbeeld de Partij code, Land code of Gemeente code. De kenmerken van deze codes zijn;

- een vaste lengte;
- de inhoud is numeriek;
- de code kan beginnen met één of meer nullen.

Er zijn twee mogelijkheden om de codes vast te leggen in de database; vastleggen als getal (Integer/ BitInt) of als tekst (Varchar / Char)

Beide methoden hebben hun voor- en nadelen.

8.1 Getal

Voordelen: neemt veelal minder ruimte in beslag in de database, automatische validatie van de numerieke inhoud.

Nadeel: aanvullen van lengte moet in de software gebeuren tijdens het leveren.

8.2 Tekst

Voordelen; geen conversie nodig bij leveren en vergelijken.

Nadelen; minder efficiënt in opslag, extra validatie nodig op numerieke waarde.

8.3 Conclusie

Initieel zijn de numerieke codes uitgewerkt geweest als getal. Het werd echter lastig om de software consistent te houden. Overal moesten rekening gehouden worden met de voorloopnullen en waren uitzondering nodig voor het verschil in de interne opslagstructuur en de representatievorm. Bijvoorbeeld bij het toepassen van expressies; zijn deze twee expressies beide valide en zo ja, geven ze hetzelfde antwoord; *PartijCode = 123* en *PartijCode = '0123'*.

De numerieke codes zijn daarom nu geïmplementeerd als tekst. Er is gekozen voor de opslag als Char omdat dit het dichtst bij het 'vaste lengte' domein ligt.

9 Inter-database verwijzingen

De database kent verwijzingen tussen schema's binnen dezelfde database, bijvoorbeeld van *AutAut* naar *Kern*, en verwijzingen tussen de schema's die over meerdere databases liggen, bijvoorbeeld van *Bericht* naar *Kern*. Op gegevens die verwijzen binnen dezelfde database (intra-database) zijn foreign keys gelegd om de consistentie van de gegevens af te dwingen. Tussen twee database (inter-database) is dit niet af te dwingen. Er ligt daarom geen foreign key tussen bijvoorbeeld de *Bericht.Zendende partij* en de *Kern.Partij*.

9.1 Technische sleutel of Logische sleutel

Voor de verwijzende attributen in bijvoorbeeld het *Bericht* schema wordt de technische sleutel van het objecttype opgeslagen. Dus *Bericht.Zendende partij* bevat het *Partij.ID*. Er is hier bewust gekozen om de technische sleutel op te slaan en niet de logische sleutel *Partij.Code* om de volgende redenen:

- De schema's waar het om gaat (Protocol en Bericht) bevatten veel data. De logische sleutels zijn vaak 'groter' dan de technische sleutels. Het opslaan van de technische sleutel is dus efficiënter.
- Er is niet voor alle gegevens een echte logische identiteit. Bijvoorbeeld personen of administratieve handelingen zijn functioneel niet aan te wijzen met een logische sleutel. Het gebruik van de soort sleutel zou dan inconsistent zijn.
- De technische sleutel is net zo 'stabiel' als de logische sleutel. Beide zullen niet veranderen.
- Consistentie over de gehele database waardoor we overal op dezelfde manier verwijzen naar gegevens.

10 Technische sleutels

De technische sleutels (ID's) zijn technisch geïmplementeerd als SmallInt, Integer of BinInt. Deze keuze is afhankelijk van de verwachte hoeveelheid tupels die de sleutel identificeert.

Voor de (meeste) persoonslijstgegevens is een uitzondering gemaakt, deze hebben allemaal een technische sleutel van het type BigInt. Deze keuze is gedreven door de software. Het objecttype *Gegeven in onderzoek.Objectsleutel/Voorkomensleutel* verwijst naar persoonslijsttupels. Deze *Objectsleutel/Voorkomensleutel* zijn gedefinieerd als BigInt, deze moeten immers alle situaties aan kunnen. Bij definitie van de technische sleutel op basis van een reële vulling zou *Persoon.ID* een Integer zijn. Maar de *Gegeven in onderzoek.Objectsleutel* BigInt verwijzing naar een *Persoon.ID* Integer verwijzing gaf 'lelijke' Java code.

Uit tests met PostgreSQL bleek dat een BigInt definitie niet meer opslag vraagt dan een Integer, zolang de 'integer ruimte' gebruikt wordt. Daarom is besloten om alle persoonslijsttupels te identificeren middels een BigInt.

11 Indicatie ‘Actueel en Geldig’

De A-laag is afgeleid van de C/D-laag. Als een groep op C/D-niveau materieel en/of formeel beëindigd is, dan worden de attributen van die groep in de A-laag leeg (NULL) gemaakt.

In bepaalde situaties kan de programmatuur de A-laag gebruiken voor het zoeken van gegevens. Denk aan de nu geldende autorisaties of het bevragen van actuele persoonsgegevens. Hierbij is het wenselijk om te weten of het gegeven in de A-laag ‘Actueel en Geldig’ is, dat wil zeggen; het gegeven is zowel actueel op de formele tijdlijn als, indien relevant, geldig op de materiele tijdlijn. In veel gevallen is dit afleidbaar uit het gevuld zijn van verplichte attributen, maar hiermee wordt een niet gewenste afhankelijkheid gecreëerd (het attribuut kan immers in de toekomst optioneel worden). En niet alle groepen hebben verplichte attributen. Daarmee kan je niet bepalen of het attribuut functioneel leeg is of leeg is omdat het attribuut niet meer actueel/geldig is. Je zou dan alsnog naar de C/D laag moeten om te bepalen wat de actualiteit/geldigheid van het gegeven is.

Om het gebruik van de A-laag te vereenvoudigen, is gekozen om per groep een indicator vast te leggen die tijdens het afleiden van de A-laag gevuld wordt; het ‘Actueel en Geldig?’-attribuut.

Als ‘Actueel en Geldig? = False’, dan kan dit het volgende betekenen:

- Er zijn geen gegevens over deze groep bekend in de C/D laag.
- Er zijn gegevens bekend in de C/D laag, maar deze zijn formeel beëindigd.
- Er zijn gegevens bekend in de C/D laag, maar deze zijn materieel beëindigd.

11.1 Naamgeving

Iedere groep met formele historie en/of materiele historie krijgt een indicator met de naam: *Actueel en Geldig <groepsnaam>?*. Uitzonderingen is de standaardgroep, hier krijgt de indicator de naam *Actueel en Geldig?*. Ook bij een (uitzonderlijke) identiteitsgroep met formele historie krijgt de indicator de naam *Actueel en Geldig?*

12 Uniciteitconstraints

Het standaard uniciteitconstraint van PostgreSQL beschouwt twee NULL waarden als ongelijk. Een tabel met 1 kolom waar een uniciteitconstraint overheen ligt kan meerdere tupels bevatten die allemaal NULL als inhoud hebben.

Voor de BRP is dit gedrag meestal niet wenselijk. Eén onbekende waarde is toegestaan, maar niet meer dan één.

12.1 Implementatie

De bovenstaande situatie is opgelost door het introduceren van twee soorten uniciteit; de (reguliere) uniciteit en de ANSI uniciteit.

12.1.1 Reguliere uniciteit

Deze uniciteit wordt afgedwongen middels een regulier database uniciteits constraint.

12.1.2 ANSI uniciteit

Deze uniciteit is geïmplementeerd door een trigger op de tabel. Hierbij wordt <NULL> als unieke waarde gezien. Twee keer een <NULL> waarbij de overige waarden identiek zijn is niet toegestaan.

De onderstaande tabel wordt gelezen als 'van boven naar beneden worden de gegevens toegevoegd'.

Kolom 1	Kolom 2	Kolom 3	Toegestaan in uniciteit	Toegestaan in ANSI uniciteit
A	B	<NULL>	Ja	Ja
A	B	<NULL>	Ja	Nee
A	<NULL>	<NULL>	Ja	Ja
A	<NULL>	<NULL>	Ja	Nee

12.1.3 Uitzonderingen:

Op de ANSI uniciteit zijn een aantal uitzonderingen mogelijk.

- Tupels waar het attribuut *Voorkomen ten behoeve van levering mutaties* gevuld is. Hier zijn dubbele waarden toegestaan. Dit komt voor uit een functionele eis t.b.v. het leveren van mutaties vanuit migratie.
- Tupels uit groepen waarvan is aangegeven dat het is toegestaan dat de 'Datum aanvang geldigheid gelijk is aan de datum einde geldigheid' (DAG = DEG), hierbij geldt de uniciteit niet als de DAG = DEG. Dit komt voort uit functionele eisen waarbij er op één datum aanvang geldigheid meerdere mutaties kunnen voorkomen. Maar dit is een uitzonderlijke situatie. Daarom is besloten om het uniciteitsconstraint wel te behouden omdat dit de consistentie van het grootste deel van de data bewaakt.

13 Materiele historie en datum attributen

In het model zijn een aantal patronen en manieren voor het omgaan met datums. Er is een onderscheid tussen het vastleggen van gebeurtennissen die plaatsvinden binnen het systeem en het vastleggen van periodes. Gebeurtenissen in het systeem zijn altijd tijdstippen (Timestamp with time zone). Datums die onderdeel zijn van de persoonslijst zijn geïmplementeerd middels een getal (Integer). Dit vindt de oorsprong in het moeten kunnen vastleggen van incorrecte en (deels) onbekende datums. Dit gaat om bijvoorbeeld om de geboortedatum van een persoon en de aanvang/einde van de materiele historie. Vanwege consistentie is besloten om de datums gebruikt in stamgegevens ook vast te leggen als getallen, ondanks dat deze wel valide datums moeten bevatten. Via het attribuuttype van het element is na te gaan of het een *Datum* betreft of een *Datum evt. (deels) onbekend*.

Hieronder volgt een opsomming van de verschillende situaties waarin de datum binnen historie gebruikt wordt.

Situatie	**Naamgeving**	**Kenmerken**
1. Patroon materiële historie	Datum aanvang geldigheid / Datum einde geldigheid	Optioneel. Type Datum evt. (deels) onbekend. Nooit in de toekomst.
2. Patroon bestaansperiode stamgegeven	Datum aanvang geldigheid / Datum einde geldigheid	Optioneel vanwege het groot aantal onbekende aanvangsdata in uit de Landelijke Tabellen GBA. Type Datum. Zowel ingang als einde kunnen in de toekomst liggen. Er is altijd maar één voorkomen en daarom geen sprake van een materieel patroon.
3. Stamgegevens binnen het AutAut schema	Datum ingang / Datum einde	Datum ingang is verplicht. Type Datum. Zowel ingang als einde kunnen in de toekomst liggen. Er is altijd maar één voorkomen en daarom geen sprake van een materieel patroon.

De keuze om voor situatie 2 een patroon te maken en 3 niet, is achteraf gezien arbitrair. De verschillen tussen deze twee situaties zijn op dit moment minimaal en hadden anders opgelost kunnen worden.

14 Indicatoren

Een aantal indicatoren worden in het Operationele gegevensmodel op een andere wijze vastgelegd dan gespecificeerd in het Logische gegevensmodel. Het betreft indicatoren die in de overgrote meerderheid van de persoonslijsten niet relevant is en waarvan het zetten dus een uitzondering is. Een voorbeeld is *Derde heeft gezag?* van het objecttype *Persoon*.

Deze indicatoren worden niet opgeslagen bij het betreffende objecttype, maar in een apart objecttype *Persoon \ Indicatie*. In het objecttype *Soort indicatie* is terug te vinden welke indicaties op deze wijze behandeld worden.

In het Operationele gegevens model zal bij dit soort attributen aangegeven zijn dat deze niet in het Operationele model zijn opgenomen, ze zijn dus echter wel geïmplementeerd.

Dit besluit is in een vroegtijdig stadium van de ontwikkeling genomen omdat deze werkwijze (veel) opslagruimte zou besparen in de database. Het gegeven hoeft immers uitsluitend voor de relevante gevallen vastgelegd te worden. Achteraf gezien valt het te betwisten of de uitzondering de besparing waard is.

15 Zoeken op diakritische tekens

Om het ‘slim zoeken’ te ondersteunen wordt gebruik gemaakt van de PostgreSQL extensie *unaccent*. Het gebruik van deze extensie heeft als voordeel dat alle standaard afgevangen worden op een zeer efficiënte manier.

Omdat in GBA-V een aantal bijzondere conversies zijn gedefinieerd die afwijken van de standaard unaccent, is er een set aangepaste unaccent regels gemaakt; *brp_unaccent.rules*. Bij het maken van de database worden deze rules gekoppeld aan de unaccent dictionary.

Voor het gebruik in de expressie index is er gebruik gemaakt van een wrapper functie *brp_unaccent*. Het gebruik van deze functie heeft twee doelen:

- Het maakt de functie *IMMUTABLE* zodat deze gebruikt kan worden in een index expressie. De standaard unaccent is uitsluitend *STABLE* omdat de achterliggende dictionary kan wijzigen. Dit zal echter niet gebeuren binnen de BRP.
- Om toe te staan dat de gegevens ‘hoofdletter ongevoelig’ te vinden zijn, de standaard zoekmethode binnen de BRP, wordt alles in uppercase geïndexeerd. Hierbij is de keuze voor upper- of lowercase willekeurig gekozen. Mocht er eventueel toch hoofdletter gevoelig gezocht moeten worden, dan is een dubbele filtering noodzakelijk; eerst op brp_unaccent om de index te raken en vervolgens op unaccent voor de vergelijking op inhoud.

16 Elementtabel

De elementtabel in de BRP database bevat de definitie van (een groot deel van) het gegevensmodel. De elementtabel wordt gevuld op basis van het BMR met alle elementen die van belang zijn voor de werking van de software. De betreft:

- Alle elementen uit het kernschema van het LGM.
- Alle patroonvelden die toegevoegd worden tijdens de transformatie van LGM naar OGM.
- Alle 'virtuele' elementen over gerelateerde die noodzakelijk zijn voor het reconstrueren van de juridische persoonslijst.

16.1 Naamgeving

Voor de naamgeving van de elementen is vastgehouden aan de naam zoals we communiceren in de XSD's. Dit heeft tot gevolg dat de *Persoon.Bijhoudingspartij* als naam heeft *Persoon.Bijhouding.PartijCode*. De programmatuur zorgt er voor dat het Integer attribuut geleverd wordt als bijbehorende Code.

16.2 Representatie attributen

De Elementtabel bevat zowel attributen ten behoeve van structuur als van representatie. De representatie attributen kunnen een afgeleid gegeven zijn op basis van het attribuuttype en/of de attributen waarnaar verwezen wordt. Dit is met name gedaan bij elementen die naar stamgegevens verwijzen. Zo is bijvoorbeeld de *Minimum lengte* van het attribuut *Persoon.Bijhouding.PartijCode* gevuld op basis van het element *Partij.Code*, de logische identiteit van het stamgegeven waarnaar verwezen wordt.

16.3 Gerelateerde gegevens

De gerelateerde gegevens op de persoonslijst van de Persoon zijn technisch gezien verwijzingen naar (andere) Personen. De opslag van de geboortedatum van een kind van een persoon gebeurt fysiek dus in de *Persoon.Datum* kolom. Voor de persoonslijst is er echter een groot verschil tussen het *Persoon.Datum* kolom in de hoedanigheid van het kind of in de hoedanigheid van de ouder. Daarom bestaan er elementen voor gerelateerde gegevens die mappen op andere elementen die wel gekoppeld zijn aan fysieke tabellen.

16.3.1 Opbouw elementen

Alle elementen van de (directe) persoonsgegevens worden gerepresenteerd als *Persoon.Groep[.Attribuut]* of *Persoon.Subobjecttype[.Attribuut]*. Bijvoorbeeld. *Persoon.Geboorte, Persoon.Geboorte.Datum, Persoon.Adres.Huisnummer*.

De gerelateerde gegevens zijn vastgelegd als verwijzing via de subtypen van betrokkenheid en relatie. Het pad van de ouders van een persoon: *Persoon.Kind.FamilierechtelijkeBetrekking.Ouder* (De persoon in hoedanigheid van een Kind, navigatie naar de FamilierechtelijkeBetrekking en daar de personen in hoedanigheid van Ouder). Omdat dit lange paden geeft zijn de volgende afkortingen toegepast

Pad	**Afkorting**
Persoon.Kind.FamilierechtelijkeBetrekking.Ouder	GerelateerdeOuder
Persoon.Ouder.FamilierechtelijkeBetrekking.Kind	GerelateerdeKind
Persoon.Partner.Huwelijk.Partner	GerelateerdeHuwelijkspartner

Persoon.Partner.GeregistreerdPartnerschap.Partner	GerelateerdeGeregistreerdePartner

De geboortedatum van een ouder zijn dus *GerelateerdeOuder.Persoon.Geboorte.Datum*. De *GerelateerdeOuder.Objectsleutel* verwijst dus naar het Relatie.ID.

Het toepassen van de afkortingen heeft (achteraf) voor een aantal problemen gezorgd. Het bleek dat we de bij de familierechtelijke betrekking van een persoon in hoedanigheid van een kind en ouder ook moesten kunnen aanwijzen. Dit heeft tot gevolg dat (delen van) het afgekorte pad ook als elementen bestaan die niet afgekort zijn.

Een paar voorbeelden van de elementen die hierdoor ontstaan in hun context:

Element paden	**Omschrijving**
Persoon	Gegevens over de persoon
Persoon.Kind	Gegevens over de kind FRB (persoon in de hoedanigheid van kind). Onderdeel van het GerelateerdeOuder pad (Persoon.Kind.FamilierechtelijkeBetrekking.Ouder)
GerelateerdeOuder	Persoonsgegeven over de ouders van de persoon
GerelateerdeOuder.Ouderschap	Gegevens over de ouderschap groep van de ouder van de persoon
Persoon.Ouder	Gegevens over de ouder FRB (persoon in de hoedanigheid van ouder). Onderdeel van het GerelateerdeKind pad (Persoon.Ouder.FamilierechtelijkeBetrekking.Kind)
Persoon.Ouder.Ouderschap	Gegevens over de ouderschap groep van de persoon in de hoedanigheid van ouder
GerelateerdeKind	Persoonsgegevens over de kinderen van de persoon