Whitepaper

Een temporele SQL Server database

Ronald Bosma

Hoofdkantoor
Kruisboog 42
3905 TG Veenendaal
Tel. +31(0)318 - 55 20 20
Fax +31(0)318 - 55 23 55

Kenniscentrum
De Smalle Zijde 39
3903 LM Veenendaal
Tel. +31(0)318 - 50 11 19
Fax +31(0)318 - 51 83 59

info.nl@infosupport.com
www.infosupport.com
K.v.K. 3013 5370
BTW NL8062.30.277.B01

IBAN NL92 RABO 0305 9528 89
BIC RABONL2U
IBAN NL74 INGB 0004 7385 93
BIC INGBNL2A

Whitepaper

Een temporele SQL Server database

Titel	Whitepaper
Project/Onderwerp	Een temporele SQL Server database
Versie	1.1
Status	Definitief
Datum	12-jan-2015
Bestand	Whitepaper Een temporele SQL Server database
Bedrijf	Info Support B.V.
Meer informatie	Voor vragen of meer informatie over deze whitepaper kunt u contact opnemen met Info Support door te bellen naar +31 (0) 318 55 20 20 en te vragen naar Sales Support & Marketing (Nederland) of te bellen naar +32 (0) 15 28 63 70 (België). U kunt ook een e-mail sturen naar sales@infosupport.com.

Inhoudsopgave

1. Inleiding

Aanleiding voor het schrijven van deze whitepaper is de afstudeeropdracht ‘Temporele datalaag converter/generator’ die uitgevoerd wordt door Ronald Bosma in opdracht van Info Support. Het doel van de opdracht is om een softwareontwikkelaar de mogelijkheid te geven automatisch een database in te richten als temporele database en automatisch data-accesslaag code te genereren voor het ophalen en opslaan van temporele data in de database. Om dit doel te bereiken zal er eerst uitgezocht moeten worden hoe een temporele database het best kan worden opgezet. Dit zal in deze whitepaper worden onderzocht.

Het doel van dit onderzoek is om de volgende hoofdvraag te beantwoorden: ‘Hoe richt ik een SQL Server database in als temporele database?’. Om deze vraag goed te kunnen beantwoorden zijn er een aantal deelvragen die onderzocht moeten worden. Deze deelvragen zijn:

- Wat is een temporele database?
- Welke verschillende temporele waarden kunnen er opgeslagen worden?
- Hoe kunnen temporele waarden worden bijgehouden?
- Op welke manier kan temporele data worden opgeslagen?
- Hoe kan ik temporele data toevoegen, opslaan en wijzigen?

De tool zal alleen ondersteuning bieden voor Microsoft SQL Server 2005 databases. De oplossingen die in deze whitepaper worden aangeboden zullen daarom ook gefocust zijn op dit type database.

In hoofdstuk 2 zal de vraag “Wat is een temporele database?” worden behandeld. Hierbij wordt onder andere gekeken naar de verschillende temporele waarden die kunnen worden opgeslagen en hoe deze kunnen worden opgeslagen.

In hoofdstuk 3 wordt de vraag “Op welke manier kan temporele data worden opgeslagen?” behandeld. Hierbij wordt gekeken naar het timestampen van tuples en van attributen.

Hoofdstuk 4 behandelt de deelvraag “Hoe kan ik temporele data toevoegen, opslaan en wijzigen?”. Hierbij wordt gekeken naar de verschillende databaseoperaties die op een database kunnen worden uitgevoerd. Denk hierbij aan toevoegen, wijzigen en verwijderen. De zaken waar rekening meegehouden moet worden in temporele databases komen hierin naar voren.

Hoofdstuk 5 kijkt naar de hoofdvraag ‘Hoe richt ik een SQL Server database in als temporele database?’. Hierin worden de verschillende datatypen en datamodellen beschreven die gebruikt kunnen worden binnen SQL Server.

In bijna alle literatuur die is gebruikt en gelezen voor deze whitepaper worden de verschillende temporele waarden als valid time, transaction time en bitemporal time wel beschreven. De meeste informatie hierover is uit [4] gehaald. De gebruikte voorbeelden zijn hierbij zelf verzonnen. De temporele datatypen instant, interval en periode worden uitgebreid beschreven in [6] en hier heb ik dan ook de meeste informatie uit geput om deze datatypen te beschrijven.

De verschillende timestamping scenario’s worden in [1] beschreven. Hierin worden de termen tuple timestamping en attribuut timestamping beschreven en noemen ze ‘vertical temporal anomaly’ en ‘horizontal temporal anomaly’. De toepassingsmogelijkheden van deze timestamp scenario’s zijn naar eigen inzicht beschreven.

Voor de databaseoperaties is naar [5] gekeken. Hier zijn de verschillende scenario's uitgehaald die mogelijk zijn bij het toevoegen, wijzigen en verwijderen van data uit de verschillende soorten temporele databases. De uitleg en voorbeelden, die bij de scenario's zijn beschreven, zijn zelf verzonnen.

De mogelijke datatypen die in SQL Server voor het opslaan van temporele waarden kunnen worden gebruikt, zijn mijn eigen idee. Details over de datatypen, zoals bereik en opslaggrootte, zijn uit [2] en [3] gehaald.

Wat betreft de datamodellen beschreef de meeste literatuur die is gelezen alleen de implementatie waarbij een tuple wordt getimestampt met een begindatum en einddatum kolom. De andere modellen zijn uit eigen ideeën voortgekomen.

2. Temporele databases

In dit hoofdstuk wordt de vraag ‘Wat is een temporele database?’ behandeld. Om deze vraag goed te kunnen beantwoorden is het belangrijk duidelijk te krijgen welke verschillende temporele waarden opgeslagen kunnen worden en hoe deze temporele waarden worden bijgehouden?

In een normale database wordt maar één toestand van informatie bijgehouden zoals deze op een bepaald moment is. Dit is meestal de toestand zoals die op dat moment in de werkelijkheid is. Bij het opvragen van gegevens zul je altijd alleen deze toestand van informatie terugkrijgen. Een normale database houdt van bijvoorbeeld de burgerlijke staat van personen maar één toestand bij, namelijk de toestand waarin de burgerlijke staat zich op dat moment bevindt. Dit kan bijvoorbeeld ongehuwd zijn.

Een temporele database is echter een database waarin historische gegevens worden bijgehouden. Dit kunnen zowel gegevens uit het verleden, het heden, als de toekomst zijn. De burgerlijke staat van een persoon bijvoorbeeld, kan meerdere toestanden hebben. Iemand is als hij wordt geboren altijd ongehuwd. Als hij trouwt verandert de toestand van zijn burgerlijke staat in gehuwd. Gaat de persoon scheiden, dan verandert zijn burgerlijke staat in gescheiden.

Een temporele database biedt de mogelijkheid om ‘terug in de tijd’ te kijken. Je kunt hierdoor de verschillende toestanden bekijken van informatie door de tijd heen. In ons voorbeeld kun je dus zien wanneer iemand ongehuwd was, wanneer hij getrouwd is geweest en vanaf wanneer hij gescheiden is. Door dit soort informatie temporeel op te slaan, kun je zien welke toestand deze gegevens op een bepaald tijdstip hadden.

Een temporele database kan denk ik het best omschreven worden als een database die het mogelijk maakt om alle toestanden van een object op te slaan en op te halen, die het tijdens zijn leven heeft aangenomen of nog zal aannemen [5]. Het bevat namelijk niet alleen een tijdsaspect, zoals in veel literatuur wordt beschreven, maar geeft ook nog eens de toestand van een object of feit weer voor een bepaald moment.

Deze toestanden kunnen opgeslagen worden zoals deze in de werkelijkheid zijn of zoals ze in de database zijn opgeslagen. Hierbij kun je de toestand opslaan voor een bepaald tijdstip of gedurende een periode. Deze opties zullen aan de hand van een voorbeeld worden uitgelegd. Hiervoor gaan we kijken naar de burgerlijke staat van de persoon Jan Jansen, zoals deze bekend is bij de gemeente.

Jan is geboren op 10 september 1955, maar is op deze dag nog niet bekend bij de gemeente. Een dag later meldt zijn vader hem aan en Jan wordt in de database van de gemeente ingevoerd met de burgerlijke staat ‘ongehuwd’.

Op 5 december 1980 trouwt Jan en zijn burgerlijke staat is ‘gehuwd’. Omdat het bij de gemeente echter een paar dagen duurt voordat deze wijziging is doorgevoerd, wordt op 7 december 1980 de burgerlijke staat van Jan in de database gewijzigd in ‘gehuwd’. Ongeveer één jaar later gaat het mis en Jan scheidt van zijn vrouw op 2 februari 1982. Dit wordt nog dezelfde dag ingevoerd.

Op 1 september 2008 komt Jan te overlijden. Op 4 september verwijdert een medewerker van de gemeente Jan uit de database. Al zijn gegevens bestaan hierdoor niet meer in de database. Zie tabel 2.1 voor de burgerlijke staat van Jan Jansen door de tijd heen in de database van de gemeente.

Datum	Burgerlijke staat in 'echte' wereld	Burgerlijke staat in de database
10 september 1955	Ongehuwd	-
11 september 1955	Ongehuwd	Ongehuwd
5 december 1980	Gehuwd	Ongehuwd
7 december 1980	Gehuwd	Gehuwd
2 februari 1982	Gescheiden	Gescheiden
1 september 2008	Overleden	Gescheiden
4 september 2008	Overleden	-

Tabel 2.1, burgerlijke staat van Jan Jansen door de tijd heen in de 'echte' wereld en de gemeentedatabase.

Zou de gemeente een normale database gebruiken, dan zou de burgerlijke staat van Jan elke keer gewijzigd worden in de burgerlijke staat van Jan op dat moment. Er wordt hierbij verder geen tijd bijgehouden. Vraag je vandaag de burgerlijke staat van Jan op, dan zal er niets terug worden gegeven doordat Jan is overleden en uit de database is verwijderd. Omdat er geen gebruik van een temporele database wordt gemaakt is het onmogelijk om te zien wanneer Jan is gescheiden en wanneer deze wijziging is ingevoerd in het systeem. Hij is namelijk helemaal niet bekend in het systeem. Hetzelfde geldt voor wanneer Jan getrouwd is.

2.1 Temporele waarden

2.1.1 Valid time

Eén van de mogelijkheden om informatie temporeel op te slaan is door het tijdstip toe te voegen waarop het feit waar is/was/zal zijn in de echte wereld. Dit wordt de valid time genoemd. Omdat voor deze term geen Nederlandse vertaling is gevonden, zullen we de Engelse tem gebruiken. In het voorbeeld hierboven wordt Jan geboren op 10 september 1955. Dit is de valid time waarop de burgerlijke staat van Jan 'ongehuwd' wordt.

Zou de gemeente een database gebruiken die de valid time bijhoudt, dan zou bij het toevoegen van Jan in de database op 11 september 1955 de valid time 10 september 1955 worden opgeslagen voor zijn burgerlijke staat. Dit betekent dat bij de gemeente bekend is dat Jan vanaf 10 september 1955 ongehuwd is.

Op 5 december 1980 trouwt Jan en deze wijziging wordt twee dagen later ingevoerd. Omdat de burgerlijke staat temporeel wordt bijgehouden willen we niet dat de bestaande gegevens van Jan (ongehuwd vanaf 10 september 1955) verloren gaan, want dan zou het geen temporele database zijn. Op 7 december 1980 wordt daarom de burgerlijke staat 'gehuwd' voor Jan toegevoegd en de valid time 5 december 1980 wordt hierbij opgeslagen. Op dezelfde manier wordt de wijziging na de scheiding van Jan in de database ingevoerd.

Bij het overlijden van Jan wordt opgeslagen dat de burgerlijke staat 'gescheiden' tot 1 september 2008 heeft geduurd. Dit geeft aan dat Jan op die datum is overleden en dus geen burgerlijke staat meer heeft. Dit kan bijvoorbeeld gedaan worden door een einddatum bij te houden voor de burgerlijke staat.

In de gemeentedatabase zijn nu de volgende gegevens bekend:

- Voor 10 september 1955 had Jan geen burgerlijke staat.
- Van 10 september 1955 tot en met 4 december 1980 was Jan ongehuwd.
- Van 5 december 1980 tot en met 1 februari 1982 was Jan getrouwd.
- Vanaf 2 februari 1982 tot 1 september 2008 was Jan gescheiden.
- Vanaf 1 september 2008 heeft Jan geen burgerlijke staat meer, omdat hij is komen te overlijden.

Databases die de valid time bijhouden worden ook wel valid time databases of historische databases genoemd. Dit type database kan namelijk de historie van de echte wereld bijhouden met betrekking tot het verleden, het heden en de toekomst.

2.1.2 Transaction time

Een tweede mogelijkheid van temporeel informatie opslaan is om de transaction time bij te houden. In het Nederlands zou dit vertaald kunnen worden naar transactietijd of het tijdstip van transactie. Er wordt nu niet bijgehouden wanneer een feit in de echte wereld juist is, maar wanneer een feit in de database is ingevoerd. In het voorbeeld wordt Jan na zijn geboorte op 11 september 1955 ingevoerd in de database en krijgt zijn burgerlijke staat de waarde ‘ongehuwd’. De transaction time is hier 11 september 1955 doordat dit de datum is waarop Jan is ingevoerd.

Op 7 december 1980 vindt de wijziging van Jan zijn burgerlijke staat in de database plaats van ongehuwd naar gehuwd. Net als bij de valid time worden de ‘oude’ gegevens niet overschreven, maar worden de nieuwe burgerlijke staat en transaction time toegevoegd. Op dezelfde manier wordt de wijziging na de scheiding van Jan in de database ingevoerd. Omdat deze wijziging op dezelfde dag als de daadwerkelijke scheiding plaatsvindt, zal hier dezelfde datum worden gebruikt als bij het bijhouden van de valid time.

Op 4 september 2008 wordt ingevoerd dat Jan is overleden. De einddatum bij de burgerlijke staat ‘gescheiden’ wordt op 4 september 2008 gezet. Dit geeft aan dat hij vanaf die datum geen geldige burgerlijke staat meer heeft doordat Jan is komen te overlijden.

In de gemeentedatabase zijn nu de volgende gegevens bekend:

- Voor 11 september 1955 was de burgerlijke staat van Jan niet bekend.
- Van 11 september 1955 tot en met 6 december 1980 is bekend dat Jan ongehuwd was.
- Van 7 december 1980 tot en met 1 februari 1982 is bekend dat Jan getrouwd was.
- Vanaf 2 februari 1982 tot en met 3 september 2008 is bekend dat Jan gescheiden was.
- Vanaf 4 september 2008 heeft Jan geen burgerlijke staat meer, omdat hij is komen te overlijden.

Zoals je kunt zien geef de transaction time niet weer wanneer Jan in de echte wereld is getrouwd, gescheiden of overleden. De transaction time geeft weer wat de toestand van informatie in de database was op een gegeven tijdstip.

Databases die de transaction time bijhouden worden ook wel rollback databases genoemd. Elke keer wanneer er een mutatie (toevoegen, wijzigen of verwijderen) wordt uitgevoerd op temporele data, wordt de mutatie opgeslagen samen met de tijd dat de mutatie plaatsvond. Dit wordt temporele rollback informatie genoemd. Hiermee kun je de data in een database terugzetten naar een bepaald punt in tijd, waarbij alle wijzigingen die na dit punt in de tijd zijn uitgevoerd, worden teruggedraaid.

2.1.3 Bitemporal time

Een temporele database kan ook zowel de valid time als transaction time bijhouden. Dit wordt bitemporal time genoemd. Een voordeel hiervan is dat je historische informatie (valid time) en temporele rollback informatie (transaction time) kunt bijhouden. Dit type database wordt ook wel een bitemporal database genoemd.

De valid time en transaction time hoeven dus niet gelijk te zijn en dit is soms zelf onmogelijk. Denk bijvoorbeeld aan een temporele database die alle persoonsgegevens opslaat van personen die in de 17e eeuw in Nederland zijn geboren. Computers bestonden nog niet in de 17e eeuw, laat staan een temporele database.

Uit dit voorbeeld komt ook het nut van temporele rollback informatie naar voren. Het kan bijvoorbeeld voorkomen dat er vandaag nieuwe persoonsinformatie is gevonden over een persoon die in het begin van de 17e eeuw is geboren. Deze informatie wordt meteen ingevoerd in de database. Als later blijkt dat deze informatie niet klopt, moet je de wijzigingen ongedaan kunnen maken. Met de temporele rollback informatie kun je alle wijzigingen op deze persoon, die na gisteren hebben plaatsgevonden, terugdraaien. Zou je alleen de valid time bijhouden, dan zou je de data over deze persoon niet zomaar terug kunnen draaien tot voor een bepaalde valid time, omdat correcte gegevens met een latere valid time dan verloren kunnen gaan.

2.1.4 Temporele datatypen

Om de hiervoor beschreven temporele waarden op te slaan, zijn er drie datatypen die gebruikt kunnen worden. Dit zijn instants, intervallen en perioden. Deze datatypen worden in deze paragraaf behandeld.

2.1.5 Instant

Jan Jansen is geboren op 10 september 1955 om 10:22. Zijn geboortedatum en tijd vormen een specifiek punt in de tijd. Dit wordt een instant genoemd. Dit is een belangrijk datatype omdat deze door andere datatypen kan worden gebruikt. Denk hierbij aan een periode waarbij met twee instants het begin en eind van de periode wordt aangegeven. De meeste database systemen ondersteunen alleen maar dit temporele datatype.

2.1.6 Interval

Het tweede temporele type is een interval. Een interval is bijvoorbeeld een week, een maand of 3 jaar. Het is een stuk aaneengesloten tijd waarvan het begin en eind niet bekend is. Waar een instant een punt in de tijd is, heeft een interval een richting. Deze kan positief of negatief zijn en geeft een ‘beweging’ naar de toekomst of het verleden aan. Een ander verschil tussen een interval en instant is dat een interval relatief is en een instant absoluut [6].

Het is wel mogelijk om een instant aan een interval te koppelen. Als je bijvoorbeeld een interval van een week hebt, weet je niet wanneer deze is begonnen, wanneer hij plaatsvindt en wanneer hij eindigt. Door met een instant aan te geven dat de week begint op 6 oktober 2008, weet je dat hij eindigt op 12 oktober 2008.

2.1.7 Periode

Het laatste temporele datatype is de periode. Een periode is een stuk aaneengesloten tijd waarvan het begin en eind bekend zijn. Jan Jansen uit ons voorbeeld is in de periode van 5 december 1980 tot en met 1 februari 1982 getrouwd geweest. Dit datatype wordt over het algemeen niet ondersteund door databases, maar kan doormiddel van twee instants voor het begin en eind van de periode wel worden geïmplementeerd.

De granulariteit (temporele resolutie) van de begin- en eindinstant is hierbij over het algemeen hetzelfde [6]. De granulariteit van een periode kan bijvoorbeeld een seconde, minuut, uur, dag, maand of jaar zijn. Het is het detailniveau dat wordt bijgehouden door, in dit geval, een periode. Als het begin van de periode met bijvoorbeeld een jaartal wordt aangegeven, dan zal het eind van de periode ook vaak met een jaartal worden aangegeven. De periode van 2005 tot en met 2008 is hier een voorbeeld van.

Een periode kan op een aantal manieren worden weergegeven/opgeslagen [6]. Als eerst is er de 'closed-closed' weergave waarbij het begin en eind van de periode beide binnen de periode vallen. De periode dat Jan Jansen getrouwd was van 5 december 1980 tot en met 1 februari 1982 is hier een voorbeeld van. Een 'closed-open' weergave kan ook worden gebruikt. Hierbij valt het eind van de periode net buiten de periode zelf. Als in ons voorbeeld de granulariteit een dag is, dan zal de periode van 5 december 1980 tot en met 2 februari 1982 zijn. Naast deze weergaven zijn 'open-closed' en 'open-open' ook nog mogelijk. Hierbij vallen respectievelijk het begin van de periode en zowel het begin als eind van de periode buiten de periode.

In de voorgaande voorbeelden heeft de periode altijd een begin en eind gehad. Het is echter ook mogelijk dat een periode is begonnen, maar nog niet is geëindigd. Jan Jansen is bijvoorbeeld nog niet overleden en is vanaf 2 februari 1982 tot op de dag van vandaag nog gescheiden. Deze periode heeft dus nog niet een eind en het is ook nog niet bekend wanneer deze zal eindigen. Deze periode kan weergegeven worden door bijvoorbeeld de eindinstant van de periode leeg te laten. Ook kan deze een waarde krijgen die aangeeft dat de periode nog voortduurt. Bijvoorbeeld een datumwaarde die ver in de toekomst ligt, zoals '9999-12-31'. Dit is de maximumwaarde bij SQL Server voor het datetime datatype (zie subparagraaf 5.1.3).

3. Timestamping scenario's

In dit hoofdstuk worden twee timestamping scenario's behandeld die in een temporele database kunnen worden gebruikt. Dit zijn tuple timestamping en attribuut timestamping. Daarnaast is het ook mogelijk om een verzameling van tuples of de database zelf te timestampen. Dit valt echter buiten de scope van deze whitepaper.

Voordat we naar de verschillende timestamping scenario's gaan kijken, is het belangrijk te weten wat timestamping inhoudt. Door een timestamp toe te voegen aan data kunnen we deze temporeel opslaan. De timestamp geeft aan welke periode de data valid is in de echte wereld en/of wanneer de data is opgeslagen in de database [1].

3.1 Tuple timestamping

Het eerste timestamping scenario is om tuples (records in een tabel) te timestampen. Hierbij wordt een heel record getimestampt. Bij een historische database gebeurt dit met de valid time periode die voor het record geldt. Bij een rollback database wordt een record getimestampt met de transaction time periode. Bij een bitemporal database wordt een record door beide voorgenoemde perioden getimestampt.

Een mogelijkheid om tuple timestamping te implementeren is door een startdatum en einddatum kolom aan een tabel toe te voegen die de valid time periode en/of transaction time periode aangeeft die voor een record geldt. Elke keer als temporele data in een record gewijzigd wordt zal een nieuw record worden aangemaakt met de wijzigingen en zal het getimestampt worden. Zie paragraaf 5.2 voor andere implementatieoplossingen van tuple timestamping.

In tabel 3.1 wordt het voorbeeld uit hoofdstuk 2 gebruikt, waarbij drie records van Jan Jansen worden bijgehouden (hij is hierbij nog niet komen te overlijden). Deze zijn getimestampt met een valid time periode. De datum '9999-12-31' in record drie geeft aan dat de valid time periode van die toestand nog niet is afgelopen.

Naam	Burgerlijke staat	Startdatum	Einddatum
Jan Jansen	Ongehuwd	1955-09-10	1980-09-05
Jan Jansen	Gehuwd	1980-09-05	1982-02-02
Jan Jansen	Gescheiden	1982-02-02	9999-12-31

Tabel 3.1, records Jan Jansen getimestampt met valid time periode.

Een nadeel van tuple timestamping is dat informatie over een persoon in de echte wereld over meerdere records wordt verspreid. Elk record representeert een toestand waarin de persoon zich bevond op een bepaald moment in de tijd. Dit wordt ook wel 'vertical temporal anomaly' genoemd [1]. Dit is een nadeel omdat een gebruiker alle gegevens van een persoon graag in één eenheid wil bekijken, bijvoorbeeld een tuple. Wil je in ons voorbeeld weten hoelang Jan Jansen zijn burgerlijke staat niet 'ongehuwd' is geweest, dan zul je dit over meerdere records van Jan Jansen moeten berekenen. De query om dit te doen kan hierdoor complex worden.

Zoals in het voorbeeld kan worden gezien is een ander nadeel van tuple timestamping dat redundantie optreedt. De naam Jan Jansen komt bijvoorbeeld drie keer voor in de tabel. Ook andere informatie zoals zijn geboortedatum e.d. kunnen worden bijgehouden in de tabel en zullen met tuple timestamping meerdere keren voorkomen.

Dit type timestamping kan het best gebruikt worden als de verschillende waarden in de kolommen van een record vaak tegelijk gewijzigd worden. Worden de waarden in verschillende kolommen van een record vaak onafhankelijk van elkaar gewijzigd en krijgen ze hierdoor elk een andere timestamp, dan is het een goed idee om attribuut timestamping toe te passen. Wordt in dit geval tuple timestamping gebruikt, dan zal de redundantie die optreed groot zijn. Het onafhankelijk van elkaar wijzigen van waarden in kolommen van een record wordt ook wel 'horizontal temporal anomaly' genoemd.

3.2 Attribuut timestamping

Het redundantieprobleem van tuple timestamping kan worden voorkomen door attribuut timestamping te gebruiken. Hierbij wordt niet getimestampt op een heel record, maar op een attribuut (kolom) van het record. Hierdoor hoeft data in niet temporele kolommen (bijv. de naam Jan Jansen) maar één keer opgeslagen te worden.

Een nadeel van attribuut timestamping is dat alle temporele attributen van een complex datatype moeten zijn of in een aparte tabel bijgehouden moeten worden. Er worden namelijk meerdere toestanden in een attribuut opslagen, samen met de timestamp die geldig is voor de waarde. Deze kunnen bijvoorbeeld in een array of collectie worden opgeslagen. Zie tabel 3.2 voor een voorbeeld. Andere oplossingen voor de implementatie van attribuut timestamping kunnen in paragraaf 5.3 worden gevonden.

Naam	Burgerlijke staat
Jan Jansen	{Ongehuwd, 1955-09-10, 1980-09-05}, {Gehuwd, 1980-09-05, 1982-02-02}, {Gescheiden, 1982-02-02, 9999-12-31}

Tabel 3.2, record Jan Jansen waarbij burgerlijke staat een valid time attribuut timestamp heeft.

De implementatie zoals deze in tabel 3.2 wordt weergegeven heeft nog een mogelijk nadeel. Het voldoet namelijk niet aan de eerste normaalvorm. Deze stelt dat elke waarde in een kolom atomair moet zijn. Een waarde die uit meerdere 'deelwaarden' is samengesteld, zoals de burgerlijke staat in onze voorbeeldimplementatie, is niet atomair. Is de eis aan de database gesteld dat hij aan de eerste normaalvorm moet voldoen, dan kan deze implementatie niet worden gebruikt. Zie paragraaf 5.3 voor andere implementatiemogelijkheden voor attribuut timestamping.

Het gebruik van attribuut timestamping wordt aangeraden als veel verschillende kolommen vaak onafhankelijk van elkaar wijzigen, ook wel 'horizontal temporal anomaly' genoemd. Wil je bijvoorbeeld één of twee kolommen in een tabel met twintig kolommen temporeel bijhouden, dan wordt attribuut timestamping ook aangeraden. Pas je op een van deze situaties tuple timestamping toe, dan kun je met veel redundantie te maken krijgen.

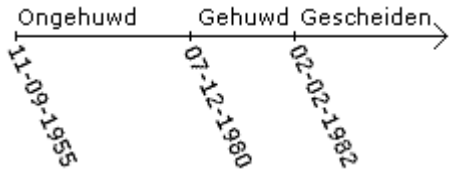
4. Databaseoperaties

Dit hoofdstuk behandelt de verschillende operaties die op data in een temporele database kunnen worden uitgevoerd. Hierbij wordt onderscheid gemaakt tussen transaction time databases, valid time databases en bitemporal databases. We gaan er hierbij vanuit dat gebruik wordt gemaakt van perioden om temporele waarden op te slaan. Voor de andere temporele datatypen zullen vergelijkbare scenario's van toepassing zijn.

4.1 Transaction time database

Bij een transaction time database wordt de tijd bijgehouden waarop een bepaalde transactie is uitgevoerd. Bijvoorbeeld het toevoegen van een persoon of het wijzigen van zijn burgerlijke staat. Deze datum wordt automatisch gegenereerd door de applicatie die de transactie uitvoert of door de database zelf. Een voordeel hiervan is dat transacties chronologisch worden uitgevoerd doordat je niet een transactie kunt uitvoeren in het verleden.

Als we het voorbeeld uit hoofdstuk 2 erbij pakken, dan ziet de tijdslijn van de burgerlijke staat er uit als in figuur 4.1 (Jan Jansen is hierbij nog niet komen te overlijden). Een volgende wijziging van de burgerlijke staat van Jan Jansen zal altijd na 2 februari 1982 liggen, namelijk de dag waarop de burgerlijke staat verandert.



Figuur 4.1, transaction time tijdslijn burgerlijke staat Jan Jansen.

4.1.1 Toevoegen en wijzigen

Wordt een nieuwe toestand in de database toegevoegd, dan zal deze als startdatum de datum krijgen waarop hij wordt toegevoegd. Bij het toevoegen van Jan Jansen met de burgerlijke staat 'ongehuwd' in de gemeentedatabase is dit 11 september 1955. Doordat op dat moment nog niet bekend is wanneer de burgerlijke staat van Jan Jansen zal wijzigen, zal de einddatum niet worden opgeslagen. Hiervoor kan bijvoorbeeld NULL worden gebruikt of een waarde ver in de toekomst, zoals '9999-12-31'.

Bij het wijzigen van een toestand zal deze nooit worden overschreven. De nieuwe waarde wordt toegevoegd aan de database door bijvoorbeeld een nieuw record in de tabel toe te voegen. In ons voorbeeld zal de waarde 'gehuwd' aan de database worden toegevoegd met als startdatum 7 december 1980. Doordat de burgerlijke staat 'ongehuwd' voor Jan Jansen niet meer geldig is vanaf de datum 7 december 1980, zal deze toestand als einddatum 6 december 1980 krijgen bij een 'closed-closed' weergave en een granulariteit van één dag.

Op dezelfde manier wordt de waarde 'gescheiden' aan de database toegevoegd met als startdatum 2 februari 1982. De waarde 'gehuwd', die al in de database staat, zal 1 februari 1982 als einddatum krijgen.

4.1.2 Verwijderen

Voor het verwijderen van temporele waarden zijn er twee mogelijkheden. Je kunt temporele waarden echt verwijderen. Bij het verwijderen van Jan Jansen uit de gemeentedatabase bijvoorbeeld, wordt alle data inclusief de temporele data (burgerlijke staat en bijbehorende perioden) verwijderd. Jan Jansen zal hierdoor niet meer bestaan in de database.

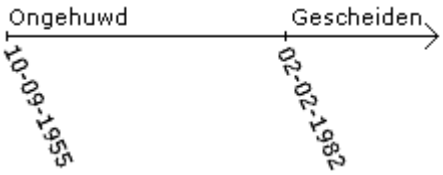
In productieomgevingen worden gegevens vaak niet verwijderd, maar bijvoorbeeld gemarkeerd als 'verwijderd' met een flag. Hierdoor blijven de gegevens behouden in de database. Bij het verwijderen van temporele waarden kun je de einddatum van de op dat moment actuele waarde in de huidige datum wijzigen. Dit geeft aan dat de toestand tot die datum geldig was en daarna is 'verwijderd'.

Bij het verwijderen van Jan Jansen uit de gemeentedatabase op 4 september 2008 wordt de einddatum voor de burgerlijke staat 'gescheiden' op 4 september 2008 gezet.

4.1.3 Valid time database

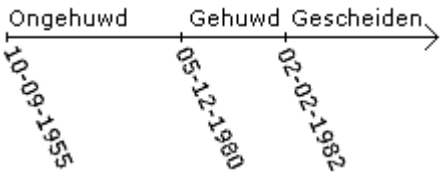
Bij operaties op een valid time database geeft de gebruiker zelf de periode op waarop temporele waarden geldig zijn. Bij het toevoegen van Jan Jansen in de gemeentedatabase zal een medewerker van de gemeente aangeven dat Jan Jansen vanaf 5 december 1955 ongehuwd is. Hierdoor hoeven wijzigingen aan temporele waarden in een database niet chronologisch te verlopen.

Theoretisch is het mogelijk dat bij de gemeente niet bekend is dat Jan Jansen is getrouwd op 5 december 1980. Er is alleen bekend dat hij vanaf 10 september 1955 ongehuwd was tot 2 februari 1982 en vanaf 2 februari 1982 is gescheiden. De tijdslijn van Jan Jansen zijn burgerlijke staat zou er dan uitzien als in figuur 4.2.



Figuur 4.2, valid time tijdslijn burgerlijke staat Jan Jansen zonder 'gehuwd'.

Komt de gemeente er op een gegeven moment achter dat Jan Jansen ook nog getrouwd is geweest, dan moet dit alsnog worden ingevoerd in de database. Als deze wijziging is doorgevoerd zou de tijdslijn van Jan Jansen zijn burgerlijke staat er uit kunnen zien als in figuur 4.3. Dit toont aan dat wijzigingen niet chronologisch uitgevoerd hoeven te worden bij een database die de valid time bijhoudt.



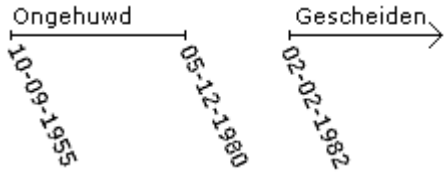
Figuur 4.3, valid time tijdslijn burgerlijke staat Jan Jansen.

4.1.4 Toevoegen en wijzigen

Bij het toevoegen van temporele waarden zal een gebruiker een startdatum en eventuele einddatum opgeven en deze data wordt in de database opgeslagen. Bij het toevoegen van Jan Jansen bijvoorbeeld wordt de burgerlijke staat 'ongehuwd' opgeslagen en zal de startdatum 5 december 2008 zijn. Hier zullen verder weinig problemen optreden. Bij het wijzigen echter, zijn er een aantal scenario's waar rekening mee moet worden gehouden [5].

1. De laatste valid time periode heeft geen einddatum en de startdatum van de nieuwe waarde ligt in deze periode.
In ons voorbeeld staat Jan Jansen in het systeem met de burgerlijke staat ongehuwd, waarvan de startdatum 10 september 1955 is. Er is nog geen einddatum bekend. Als zijn burgerlijke staat wordt gewijzigd in gescheiden met als startdatum 2 februari 1982, dan zal deze waarde aan de database worden toegevoegd met de opgegeven startdatum. De einddatum van de burgerlijke staat 'ongehuwd' zal de waarde 1 februari 1982 krijgen bij een 'closed-closed' weergave en granulariteit van één dag.

2. De valid time periode van de nieuwe waarde begint na de einddatum van de laatste valid time periode.
In het voorbeeld van Jan Jansen kan het zijn dat alleen bekend is in de database dat Jan Jansen van 10 september 1955 tot 5 december 1980 ongehuwd is geweest. Als de burgerlijke staat verandert in 'gescheiden' zal deze waarde worden toegevoegd aan de database met de opgegeven startdatum en eventuele einddatum. Tijdens de periode van 5 december 1980 tot 2 februari 1982 heeft Jan Jansen geen valid toestand gehad voor zijn burgerlijke staat.



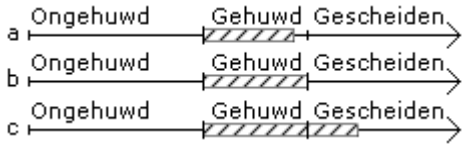
Figuur 4.4, van 5 december 2008 tot 2 februari 1982 heeft de burgerlijke staat van Jan Jansen geen valid waarde gehad.

3. De valid time periode van de nieuwe waarde begint op dezelfde datum als de valid time periode van een bestaande waarde.
Binnen dit scenario zijn er een aantal mogelijkheden. We gaan hierbij uit van een granulariteit van één dag en de weergave 'closed-closed' voor de valid time periode.

De eerste mogelijkheid is dat de einddatum van de nieuwe waarde voor de einddatum van de bestaande waarde ligt (zie figuur 4.5a). In dit geval zal het systeem de nieuwe waarde toevoegen en de startdatum van de bestaande waarde aanpassen aan de einddatum van de nieuwe waarde plus een dag.

Ook is het mogelijk dat de einddatum van de nieuwe waarde hetzelfde is als de einddatum van de bestaande waarde (zie figuur 4.5b). De valid time perioden zijn dus hetzelfde. In dit geval kan het systeem de bestaande waarde vervangen door de nieuwe waarde.

De laatste mogelijkheid is dat de einddatum van de nieuwe waarde na de einddatum van de bestaande waarde valt (zie figuur 4.5c). In dit geval kan de valid time periode van de nieuwe waarde één of meerdere bestaande waarden overlappen. Het systeem kan dan de overlapte waarden verwijderen uit de database. De laatste bestaande waarde kan deels overlapt worden, waarbij de begindatum van deze waarde wordt aangepast aan de einddatum van de nieuwe waarde plus een dag. Als de nieuwe waarde geen einddatum heeft en dus nog voortduurt, zullen alle waarden uit de database worden verwijderd die worden overlapt door de valid time periode van de nieuwe waarde.



Figuur 4.5, nieuwe valid time periode (gearceerd) en bestaande valid time periode beginnen op hetzelfde moment.

Bij de voorgaande mogelijkheden is uitgegaan van een nieuwe waarde die verschilt van de bestaande overlapte waarden. Bij mogelijkheid c van figuur 4.5, waar overlapping van meerdere valid time perioden plaatsvindt, is het echter ook mogelijk dat de nieuwe waarde dezelfde waarde heeft als één van de overlapte bestaande waarden. In ons voorbeeld is de nieuwe waarde bijvoorbeeld 'gehuwd'. In dit geval hoeft de bestaande waarde 'gehuwd' niet gewijzigd te worden, maar kan de einddatum worden aangepast aan de einddatum van de nieuwe waarde. Uiteraard moet de startdatum van de waarde 'gescheiden' ook aangepast worden, zoals hierboven is beschreven.

Bij het overlappen van de valid time periode van meerdere bestaande waarden door de valid time periode van de nieuwe waarde kan dit echter een complexe operatie worden. Je moet een bestaande waarde vinden met een overlapte valid time periode die dezelfde waarde heeft als de nieuwe waarde. De startdatum en einddatum moeten vervolgens worden aangepast. Daarna moeten alle andere overlapte waarden worden verwijderd. Bij veel overlapte bestaande waarden kan dit de performance verminderen. Of deze manier gebruikt moet worden is dus afhankelijk van de voorkeur van de ontwikkelaar en/of databasedesigner.

4. De valid time periode van de nieuwe waarde begint tijdens de valid time periode van een bestaande waarde.
Net als bij het vorige scenario zijn er binnen dit scenario een aantal mogelijkheden. We gaan hierbij uit van een granulariteit van één dag en de weergave 'closed-closed' voor de valid time periode.

De eerste mogelijkheid is dat de einddatum van de nieuwe waarde op dezelfde datum valt als de einddatum van de bestaande waarde (zie figuur 4.6a). In dit geval wordt de nieuwe waarde ingevoegd en zal de einddatum van de bestaande waarde de begindatum van de nieuwe waarde min een dag worden.

De tweede mogelijkheid is dat de valid time periode van de nieuwe waarde binnen de valid time periode van de bestaande waarde valt (zie figuur 4.6b). De einddatum van de nieuwe waarde valt dus voor de einddatum van de bestaande waarde. In dit geval zal de nieuwe waarde worden ingevoegd. De bestaande waarde wordt opgesplitst (bijv. in twee records afhankelijk van de implementatie), waarbij de einddatum van de eerste bestaande waarde de begindatum van de nieuwe waarde min een dag wordt. De begindatum van de tweede bestaande waarde krijgt de einddatum van de nieuwe waarde plus een dag.

De laatste mogelijkheid is dat de einddatum van de nieuwe waarde na de einddatum van de bestaande waarde valt (zie figuur 4.6c). De valid time periode overlapt hierbij één of meerdere bestaande waarden. De nieuwe waarde zal worden ingevoegd en de einddatum van de eerste overlapte bestaande waarde wordt de begindatum van de nieuwe waarde min een dag. De volledig overlapte waarden zullen worden verwijderd. Als de einddatum van de laatste overlapte waarde na de einddatum van de nieuwe waarde valt (hij wordt niet compleet overlapt) zal de begindatum van deze bestaande waarde de einddatum van de nieuwe waarde plus een dag worden. Heeft de nieuwe waarde geen einddatum en duurt zijn valid time periode momenteel nog voort, dan zullen alle volledig overlapte waarden worden verwijderd.

Figuur 4.6, nieuwe valid time periode (gearceerd) begint tijdens bestaande valid time periode.

Net als bij het voorgaande scenario is bij de hierboven genoemde mogelijkheden uitgegaan van een nieuwe waarde die verschilt van de bestaande overlapte waarden. Bij mogelijkheid c van figuur 4.6, waar overlapping van meerdere valid time perioden plaatsvindt, is het echter ook mogelijk dat de nieuwe waarde dezelfde waarde heeft als één van de overlapte bestaande waarden.

In ons voorbeeld is de nieuwe waarde bijvoorbeeld ‘ongehuwd’. In dit geval hoeft de bestaande waarde ‘ongehuwd’ niet gewijzigd te worden, maar kan de einddatum worden aangepast aan de einddatum van de nieuwe waarde. Uiteraard moet de bestaande waarde ‘gehuwd’ worden verwijderd en de startdatum van de waarde ‘gescheiden’ aangepast worden, zoals hierboven wordt beschreven.

Bij het overlappen van de valid time periode van meerdere bestaande waarden door de valid time periode van de nieuwe waarde kan dit echter ook weer een complexe operatie worden. Of deze manier gebruikt moet worden is dus afhankelijk van de voorkeur van de ontwikkelaar en/of databasedesigner.

4.1.5 Verwijderen

Voor het verwijderen van een valid time waarde gelden in feite dezelfde scenario’s als bij het wijzigen, alleen wordt er geen nieuwe waarde toegevoegd. De gearceerde delen in figuur 4.5 en figuur 4.6 zullen verdwijnen van de tijdslijn, waarbij de tijdslijn wordt onderbroken. Het wijzigen van de valid time periode van de waarden die blijven bestaan wordt op dezelfde manier gedaan als bij het wijzigen.

Bij het daadwerkelijk verwijderen van een waarde ontstaat een ‘gat’ in de tijdslijn. Het is ook mogelijk om een wijziging door te voeren, zoals in de vorige subparagraaf is beschreven, waarbij de nieuwe waarde de waarde NULL heeft. Hiermee wordt aangegeven dat de temporele waarde voor die periode geen valid toestand had.

4.2 Bitemporal database

4.2.1 Toevoegen en wijzigen

Omdat bij bitemporal databases naast de valid time ook de transaction time wordt bijgehouden heeft dit als voordeel dat er geen wijzigingen uitgevoerd hoeven te worden. Nieuwe waarden worden simpelweg in de database toegevoegd, ook bij overlapping van de valid time periode. Doordat de transaction time wordt bijgehouden kan het systeem nagaan welk waarde voor een bepaalde valid time geldig is.

Houden wij voor de burgerlijke staat van Jan Jansen zowel de valid time als transaction time bij, dan zou de database de gegevens kunnen bevatten zoals deze in tabel 4.1 worden weergegeven. In onderstaand voorbeeld is nog niet de burgerlijke staat ‘gehuwd’ doorgevoerd doordat dit nog niet bekend was bij de gemeente.

Burgerlijke staat	Valid time		Transaction time datum
	Startdatum	Einddatum	
Ongehuwd	1955-09-10	1982-02-01	1955-09-11
Gescheiden	1982-02-02	9999-12-31	1982-02-02

Tabel 4.1, burgerlijke staat Jan Jansen in de database op 30 september 2008 (‘closed-closed’ weergave).

Op 1 oktober 2008 komt iemand van de gemeente erachter dat Jan Jansen van 5 december tot en met 1 februari getrouwd is geweest. Dit wordt ingevoerd in de database, waarna de data eruit ziet als is weergegeven in tabel 4.2.

Burgerlijke staat	Valid time		Transaction time datum
	Startdatum	Einddatum	
Ongehuwd	1955-09-10	1982-02-01	1955-09-11
Gescheiden	1982-02-02	9999-12-31	1982-02-02
Gehuwd	*1980-12-05*	*1982-02-01*	*2008-10-01*

Tabel 4.2, burgerlijke staat Jan Jansen in de database op 1 oktober 2008 (‘closed-closed’ weergave).

De burgerlijke staat ‘gehuwd’ is toegevoegd met de juiste valid time periode. Ook is de datum van de transaction time ingesteld op 1 oktober 2008. De valid time periode van ‘gehuwd’ overlapt die van ‘ongehuwd’, maar de gegevens van deze burgerlijke staat worden niet gewijzigd.

Vraag je de burgerlijke staat van Jan Jansen voor bijvoorbeeld 1 januari 1981 op, dan zul je ‘gehuwd’ terugkrijgen. Hoewel de opgegeven datum ook in de valid time periode van ‘ongehuwd’ valt, kan aan de hand van de transaction time datum na worden gegaan welke waarde geldig is.

In bovenstaande voorbeelden wordt voor het vastleggen van de transaction time geen periode gebruikt, maar een instant. Als alleen de transaction time wordt bijgehouden kan de einddatum aangeven tot wanneer een waarde geldig was in de database en wanneer iets is verwijderd. Echter kunnen toestanden van bijvoorbeeld een burgerlijke staat in een bitemporal time database verschillende valid time periode bevatten die elkaar niet overlappen en dus allen geldig zijn. Hierdoor heeft het bijhouden van een einddatum voor de transaction time geen nut.

4.2.2 Verwijderen

Het verwijderen kan op een vergelijkbare manier worden uitgevoerd als bij het wijzigen van een temporele waarde. Verwijder je voor Jan Jansen op 3 oktober 2008 de burgerlijke staat van 1 januari 1981 tot en met 1 januari 1985, dan kun je een NULL waarde toevoegen met deze valid time periode en de transaction time waarop de verwijdering is uitgevoerd. Zie tabel 4.3 voor het resultaat van de verwijdering.

Burgerlijke staat	Valid time		Transaction time datum
	Startdatum	Einddatum	
Ongehuwd	1955-09-10	1982-02-01	1955-09-11
Gescheiden	1982-02-02	9999-12-31	1982-02-20
Gehuwd	1980-12-05	1982-02-01	2008-10-01
NULL	*1981-01-01*	*1985-01-01*	*2008-10-03*

Tabel 4.3, burgerlijke staat Jan Jansen in de database op 3 oktober 2008 (‘closed-closed’ weergave).

5. Temporele databases in Microsoft SQL Server

In dit hoofdstuk wordt gekeken hoe temporele databases in Microsoft SQL Server geïmplementeerd kunnen worden. Hiervoor wordt eerst gekeken naar de datatypen die gebruikt kunnen worden voor het timestampen van temporele waarden. Daarna worden een aantal mogelijkheden beschreven om een temporele database in te richten. Hierbij wordt rekening gehouden met de eisen die aan de te maken tool zijn gesteld (de aanleiding voor deze whitepaper).

5.1 Datatypen voor temporele waarden

Voor het opslaan van temporele waarden (in de vorm van een instant), zoals de valid time of transaction time, zijn een aantal datatypen beschikbaar in Microsoft SQL Server. Welk datatype gebruikt kan worden hangt onder andere af van de granulariteit (subparagraaf 2.1.7). Wordt er bijvoorbeeld op het niveau van jaren, dagen, minuten, etc. gekeken. Een aantal van de mogelijke datatypen wordt hierna behandeld.

5.1.1 smallint

Voor het opslaan van een jaartal kan een smallint gebruikt worden. Hierbij moet echter opgemerkt worden dat dit datatype ook negatieve waarden kan bevatten en de maximum op 32.767 ligt. Er zullen dus controles moeten worden uitgevoerd op de waarden die hierin worden opgeslagen. Dit kan bijvoorbeeld via constraints worden gedaan. Hoe dit in zijn werk gaat valt echter buiten de scope van deze whitepaper. Hoewel een int of bigint ook gebruikt kan worden voor de opslag van jaartallen, nemen deze meer ruimte in beslag in het geheugen en op disk.

5.1.2 int

Als er op maandniveau wordt gekeken is het mogelijk om een int te gebruiken. De maand september in het jaar 2008 kan bijvoorbeeld als 200809 worden opgeslagen. Omdat er net als bij een smallint negatieve waarden mogelijk zijn, het maximum getal dat opgeslagen kan worden 2.147.483.647 is en een waarde als 200800 mogelijk is, zullen ook hier controles moeten worden uitgevoerd bij de opslag van waarden. Net als bij jaar kan ook bij deze oplossing een bigint worden gebruikt als datatype, maar deze gebruikt meer ruimte dan een int.

Een ander nadeel van deze oplossing is dat 200809 een samengestelde waarde is en dus niet aan de eerste normaalvorm voldoet. Als dit een probleem is zou de waarde opgesplitst kunnen worden in een jaar en maand kolom. Een andere oplossing is om het aantal maanden bij te houden vanaf een bepaalde ‘beginmaand’, bijvoorbeeld januari 1900. Hierna kun je berekenen wat de maand is door de beginmaand plus het aantal opgeslagen maanden bij elkaar op te tellen.

Een int kan ook gebruikt worden als de granulariteit in dagen is. De datum 10 september 2008 kan bijvoorbeeld als 20080910 worden opgeslagen. Net als bij de vorige oplossing heeft deze mogelijkheid als nadeel dat validaties moeten worden toegevoegd om te controleren of de datum correct is en voldoet ook deze oplossing niet aan de eerste normaalvorm.

Net als een maand en datum kunnen waarden met een nog fijnere granulariteit in de int en/of bigint worden opgeslagen. Als je voor deze oplossing kiest houd dan goed in de gaten wat de maximumwaarde is die een int of bigint kan opslaan om problemen te voorkomen met de waarde die dit overschrijden.

5.1.3 datetime

Er is een datatype, genaamd datetime, waarin je een datum kunt opslaan tot op één-drie-honderdste van een seconde nauwkeurig [3]. Een voordeel van dit datatype is dat controles op bijvoorbeeld de maand- of dagwaarde niet zelf uitgevoerd hoeven te worden. Naast datetime is er ook smalldatetime. Dit type is vergelijkbaar met datetime, maar slaat waarden op tot op de minuut nauwkeurig. Ook is het bereik van mogelijke datums kleiner. Voordeel van dit type is dat het de helft minder ruimte in beslag neemt dan datetime.

Een jaar of maand zou eventueel ook in dit datatype opgeslagen kunnen worden. Het jaar 2008 kan bijvoorbeeld opgeslagen worden als 2008-01-01 waarbij altijd 1 januari voor het gewenste jaar wordt opgeslagen. In de database wordt deze als '2008-01-01 00:00:00.000' opgeslagen, waarbij de tijd altijd '00:00:00.000' zal zijn. De maand september 2008 kan als 2008-09-01 worden opgeslagen, waarbij altijd de 1e van de gewenste maand wordt opgeslagen. Hierdoor hoeven bepaalde validaties niet uitgevoerd te worden, maar kan de opslaggrootte van een waarde wel groter zijn.

5.1.4 User-defined type

Naast de datatypen die standaard beschikbaar zijn in SQL Server, kunnen er ook user-defined types worden aangemaakt. Dit zijn door de gebruiker gespecificeerde typen en kunnen in een .Net programmeertaal, zoals C#, worden geschreven. Het is bijvoorbeeld mogelijk een datatype 'periode' aan te maken, die een startdatum en einddatum bevat. In plaats van een startdatum en einddatum kolom van het type datetime te gebruiken voor timestamping, kan dit 'periode' datatype worden gebruikt. Hoe een user-defined type gemaakt en gebruikt kan worden valt buiten de scope van deze whitepaper. Voor meer informatie over dit onderwerp zie [2].

5.2 Tuple timestamping datamodellen

In deze paragraaf worden de verschillende mogelijkheden beschreven waarop tuple timestamping kan worden geïmplementeerd in een SQL Server database. Deze voorbeelden kunnen direct of indirect ook worden gebruikt in andere databaseomgevingen. Bij de gegeven voorbeelden gaan we uit van een periode als temporeel datatype. Hierbij vormen twee instant kolommen de startdatum en einddatum van de periode.

5.2.1 Timestamp toevoegen aan tabel

Eén van de mogelijkheden om tuple timestamping toe te passen op een database is om een startdatum en einddatum kolom aan de tabel toe te voegen waarop tuple timestamping plaats moet vinden. In figuur 5.1 wordt een voorbeeld gegeven van deze implementatie. Om na te gaan welke wijzigingen op een bepaald persoon zijn gedaan is er ook een kolom 'prevId' toegevoegd. Hierin wordt de id opgeslagen van het record waarop de wijzigingen zijn uitgevoerd.

Personen	
PK	**id**
	prevId naam burgerlijkeStaat salaris **startdatum** **einddatum**

Figuur 5.1, tuple timestamping op tabel 'Personen'.

De inhoud van de tabel 'Personen' kan er uitzien zoals wordt weergegeven in tabel 5.1. De persoon Jan Jansen is hierbij drie keer van burgerlijke staat veranderd. Ook heeft hij een salarisverhoging gehad toen hij trouwde en tijdens de periode dat zijn burgerlijke staat 'gescheiden' was.

Id	prevId	Naam	Burgerlijke staat	Salaris	Startdatum	Einddatum
1	NULL	Jan Jansen	Ongehuwd	1000	1955-09-10	1980-09-04
8	1	Jan Jansen	Gehuwd	1500	1980-09-05	1982-02-01
12	8	Jan Jansen	Gescheiden	1500	1982-02-02	2008-09-30
65	12	Jan Jansen	Gescheiden	2000	2008-10-01	9999-12-31

Tabel 5.1, deel van de inhoud van de tabel 'Personen'.

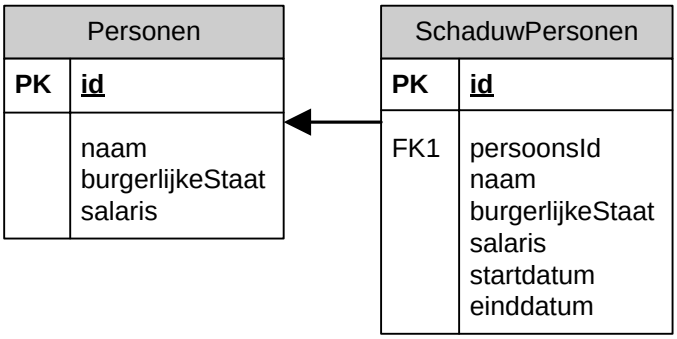
Het eerste record heeft voor de prevId de waarde NULL, wat aangeeft dat dit het eerste record voor Jan Jansen is. Wordt er nu een wijziging uitgevoerd op Jan Jansen, dan zal het nieuwe record 65 als prevId krijgen. Om alle records van Jan Jansen op te halen zal recursie moeten worden gebruikt.

In plaats van het opslaan van het vorige id is het opslaan van het originele id ook een mogelijkheid. Alle records voor Jan Jansen zouden dan in de kolom 'prevId' de waarde 1 krijgen met uitzondering van het eerste record. Hierdoor is het gemakkelijker om alle records op te halen die bij Jan Jansen horen, omdat je geen gebruik hoeft te maken van recursie.

Zoals al in paragraaf 3.1 is genoemd heeft dit datamodel als nadeel dat je redundantie krijgt. Verder vindt er ook geen scheiding plaatst tussen de actuele data en temporele data.

5.2.2 Schaduwtabel voor temporele data

Om een scheiding te krijgen tussen de actuele data en temporele data kun je een schaduwtabel toevoegen aan de database waarin alle temporele data wordt opgeslagen. Voor onze tabel 'Personen' zouden we een schaduwtabel 'SchaduwPersonen' kunnen aanmaken (zie figuur 5.2).



Figuur 5.2, tuple timestamping op tabel 'Personen' met behulp van schaduwtabel.

De tabel 'Personen' bevat de meest actuele gegevens van Jan Jansen (tabel 5.2). Een voordeel hiervan is dat de actuele gegevens snel kunnen worden opgevraagd zonder de temporele tabel te hoeven raadplegen. De tabel 'SchaduwPersonen' bevat alle temporele data (tabel 5.3). Aan de einddatum van de laatste valid time periode in de schaduwtabel kan gezien worden vanaf wanneer de gegevens in de brontabel geldig zijn.

Id	Naam	Burgerlijke staat	Salaris
1	Jan Jansen	Gescheiden	2000

Tabel 5.2, deel van de inhoud van de brontabel 'Personen'

Id	persoonsId	Naam	Burgerlijke staat	Salaris	Startdatum	Einddatum
1	1	Jan Jansen	Ongehuwd	1000	1955-09-10	1980-09-04
8	1	Jan Jansen	Gehuwd	1500	1980-09-05	1982-02-01
12	1	Jan Jansen	Gescheiden	1500	1982-02-02	2008-09-30

Tabel 5.3, deel van de inhoud van de schaduwtabel 'SchaduwPersonen'

Een nadeel van deze implementatie kan bij het verwijderen optreden. Bij het bijhouden van de transaction time bijvoorbeeld wil je alle gegevens van Jan Jansen bewaren om een eventuele rollback uit te kunnen voeren. Zijn gegevens mogen dus niet fysiek worden verwijderd. Een oplossing is om een kolom toe te voegen aan de tabel 'Personen' die aangeeft of een record verwijderd is of niet.

Een andere optie is om de waarden uit de tabel 'Personen' in de schaduwtabel toe te voegen met een einddatum die aangeeft wanneer Jan Jansen 'verwijderd' was. De kolommen van Jan Jansen in de tabel 'Personen' worden op NULL gezet, wat aangeeft dat hij is verwijderd. Ook zou je het record van Jan Jansen in zijn geheel uit de tabel 'Personen' kunnen verwijderen. Heb je echter een constraint aangemaakt, die aangeeft dat de waarde van de persoonsId in de schaduwtabel ook in de brontabel moet bestaan, dan kan dit record niet verwijderd worden.

Als laatste mogelijkheid kunnen de actuele gegevens altijd in de schaduwtabel worden opgenomen, waarbij de einddatum bijvoorbeeld de waarde NULL of '9999-12-31' krijgt. Hierdoor treedt extra redundantie op en heeft de schaduwtabel in feite dezelfde inhoud als de tabel uit de vorige subparagraaf (tabel 5.1). De tabel 'Personen' is in feite overbodig geworden, tenzij vaak de actuele gegevens van een persoon worden opgevraagd. Door deze gegevens uit de tabel 'Personen' te halen kan de performance verbeterd worden.

In ons voorbeeld kan het mogelijk zijn dat de naam nooit gewijzigd mag worden of dat de naam niet temporeel bijgehouden hoeft te worden doordat de gemeente hier niet in geïnteresseerd is. Om de redundantie te verminderen kan deze kolom uit de schaduwtabel weggelaten worden en alleen worden opgenomen in de brontabel. Dit kan handig zijn als verwacht wordt dat de temporele kolommen over het algemeen tegelijk worden gewijzigd en niet vaak onafhankelijk van elkaar wijzigen. Is dit wel het geval, dan wordt aangeraden om attribuut timestamping te gebruiken.

Zoals al eerder is genoemd bevat de tabel 'Personen' de meest actuele gegevens van Jan Jansen. Doordat de schaduwtabel niet doorzocht hoeft te worden bij het ophalen van de actuele gegevens, kan dit voor een verbetering van de performance zorgen. Worden echter ook gegevens over de toekomst in de tabel opgeslagen, dan kan dit datamodel voor een probleem zorgen. Gegevens in de brontabel die vandaag geldig zijn, kunnen morgen ongeldig zijn zonder dat er in de tussentijd wijzigingen op aangebracht zijn.

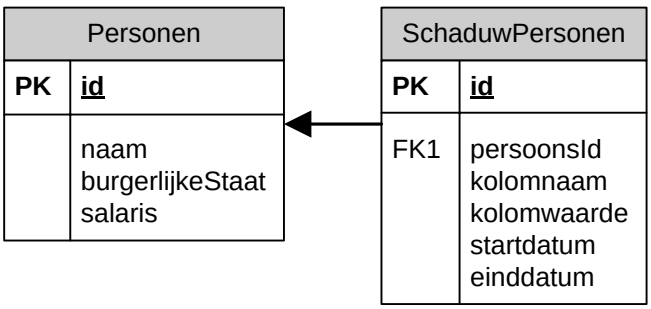
In deze situaties moet dus gecontroleerd worden of de waarden in de brontabel wel actueel zijn door de temporele periode van de waarden in de schaduwtabel te controleren. Zouden de waarden niet meer actueel zijn, dan moeten ze in de brontabel gewijzigd worden naar de actuele waarden. In dit geval heeft het bijhouden van de actuele gegevens in de brontabel weinig zin als je de performance van het opvragen van actuele gegevens zou willen verbeteren, omdat je toch in de schaduwtabel moet kijken naar de geldigheid van de waarden en misschien zelfs een extra update moet uitvoeren om de waarden in de brontabel te actualiseren. Er wordt daarom geadviseerd om in deze situatie de temporele kolommen uit de brontabel te verwijderen. De temporele kolommen 'burgerlijkeStaat' en 'salaris' zouden in ons voorbeeld in uit de brontabel 'Personen' verwijderd kunnen worden en altijd uit de schaduwtabel gehaald kunnen worden.

5.3 Attribuut timestamping datamodellen

In deze paragraaf worden de verschillende mogelijkheden beschreven waarop attribuut timestamping kan worden geïmplementeerd in een SQL Server database. Deze voorbeelden kunnen direct of indirect ook worden gebruikt in andere databaseomgevingen. Bij de gegeven voorbeelden gaan we uit van een periode als temporeel datatype.

5.3.1 Schaduwtabel voor alle temporele attributen van tabel

De eerste mogelijkheid is om één schaduwtabel aan te maken voor een tabel waarin attributen temporeel opgeslagen moeten worden. In figuur 5.3 wordt een voorbeeld gegeven waarbij de waarden in de kolommen 'burgerlijkeStaat' en 'salaris' temporeel opgeslagen moeten worden. De tabel 'Personen' bevat de actuele waarden en de tabel 'SchaduwPersonen' bevat de temporele waarden.



Figuur 5.3, attribuut timestamping op attributen van tabel 'Personen' met één schaduwtabel.

In tabel 5.4 wordt een deel van de inhoud weergeven van de tabel ‘Personen’. Deze bevat de niet temporele waarden zoals de naam en de meest actuele waarden van de temporele kolommen. Tabel 5.5 geeft een deel van de inhoud van de schaduwtabel ‘SchaduwPersonen’ weer. De kolom ‘kolomnaam’ bevat de naam van de kolom waarop de temporele waarde betrekking heeft. In het voorbeeld kan dit ‘burgerlijkeStaat’ of ‘salaris’ zijn. De kolom ‘kolomwaarde’ bevat de daadwerkelijke temporele waarde en de start- en einddatum geven de temporele periode aan.

Id	Naam	Burgerlijke staat	Salaris
1	Jan Jansen	Gescheiden	1500

Tabel 5.4, deel van de inhoud van de brontabel 'Personen'.

Id	persoonsId	Kolomnaam	Kolomwaarde	Startdatum	Einddatum
1	1	burgerlijkeStaat	Ongehuwd	1955-09-10	1980-09-04
8	1	burgerlijkeStaat	Gehuwd	1980-09-05	1982-02-01
12	1	burgerlijkeStaat	Gescheiden	1982-02-02	9999-12-31
2	1	salaris	1000	1955-09-10	1980-09-04
9	1	salaris	1500	1980-09-05	9999-12-31

Tabel 5.5, deel van de inhoud van de schaduwtabel 'SchaduwPersonen'.

Een voordeel van deze implementatie is dat er weinig redundantie optreedt. Alleen de actuele waarden worden dubbel opgeslagen in zowel de brontabel als de schaduwtabel. Verder is het datamodel gemakkelijk te onderhouden en te wijzigen doordat er per tabel maar één tabel is waar de temporele waarden in worden opgeslagen.

Een nadeel is dat het datatype van de temporele waarde per attribuut kan verschillen. In ons voorbeeld is de burgerlijkeStaat van het type VARCHAR en het salaris van het type INT. In SQL Server kan dit opgelost worden door de kolom ‘kolomwaarde’ van het datatype SQL_VARIANT te maken. Hierin kunnen waarden van verschillende datatypen worden opgeslagen.

Bij het ophalen van de gegevens kan het echter mogelijk zijn dat je expliciet moet aangeven van welk type een waarde is. Dit wordt ook wel casten genoemd. Wil je bijvoorbeeld het gemiddelde salaris van Jan Jansen weten, dan moet je deze waarden omzetten naar een integer. Voor dit soort taken is dus een uitgebreide kennis van het datamodel nodig en maakt het maken van queries complexer.

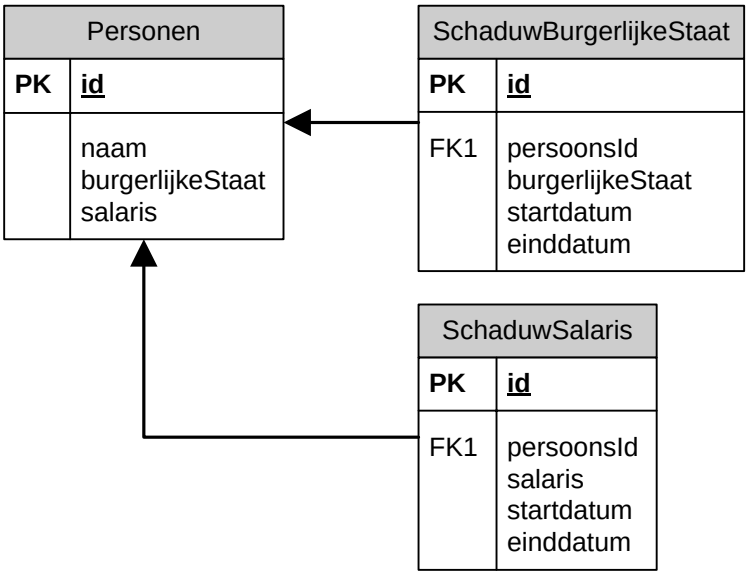
Een ander nadeel kan de hoeveelheid aan joins zijn die gebruikt moeten worden om met één SQL query de juiste waarden voor temporele attributen op te halen voor een persoon. Voor elke attribuut moet namelijk een join met de schaduwtabel worden gemaakt (in plaats van een join kan ook een subquery gebruikt worden). Door de hoeveelheid aan joins kan de performance slechter zijn dan gewenst. Met indexen en andere technieken kan dit worden verbeterd. Hoe dit moet valt buiten de scope van deze whitepaper.

Een ander nadeel van de hoeveelheid aan joins of subqueries is de complexheid van een query om een record op te halen. Om dit probleem te verhelpen kan bijvoorbeeld een view of stored procedure worden gebruikt. Ook dit valt

buiten de scope van deze whitepaper. Ga je gebruik maken van een view of stored procedure, houd dan in de gaten dat je bij het wijzigen van je datamodel, bijvoorbeeld het toevoegen van een nieuwe temporele kolom, ook de view of stored procedure moet aanpassen.

5.3.2 Schaduwtabel per attribuut

Om het gebruik van het SQL_VARIANT type te voorkomen kan ervoor worden gekozen om voor elk temporeel attribuut een eigen schaduwtabel te maken. Figuur 5.4 geeft een voorbeeld waarbij de tabel ‘Personen’ de brontabel is en de tabellen ‘SchaduwBurgerlijkeStaat’ en ‘SchaduwSalaris’ de temporele waarden van respectievelijk de burgerlijke staat en het salaris bijhouden.



Figuur 5.4, attribuut timestamping op attributen van tabel 'Personen' met voor elk attribuut een schaduwtabel.

In tabel 5.6 wordt een deel van de mogelijke inhoud van de tabel ‘Personen’ weergegeven. Deze bevat de niet temporele waarden zoals de naam en de meest actuele waarden van de temporele kolommen. Tabel 5.7 en tabel 5.8 geven de mogelijke inhoud van de schaduwtabellen ‘SchaduwBurgerlijkeStaat’ en ‘SchaduwSalaris’ weer.

Id	Naam	Burgerlijke staat	Salaris
1	Jan Jansen	Gescheiden	1500

Tabel 5.6, deel van de inhoud van de brontabel 'Personen'.

Id	persoonsId	BurgerlijkeStaat	Startdatum	Einddatum
1	1	Ongehuwd	1955-09-10	1980-09-04
8	1	Gehuwd	1980-09-05	1982-02-01
12	1	Gescheiden	1982-02-02	9999-12-31

Tabel 5.7, deel van de inhoud van de schaduwtabel 'SchaduwBurgerlijkeStaat'.

Id	persoonsId	Salaris	Startdatum	Einddatum
2	1	1000	1955-09-10	1980-09-04
9	1	1500	1980-09-05	9999-12-31

Tabel 5.8, deel van de inhoud van de schaduwtabel 'SchaduwSalaris'.

Een voordeel van deze aanpak is dat voor elk attribuut het juiste datatype gebruikt kan worden. Het casten van een waarde hoeft dus niet meer uitgevoerd te worden.

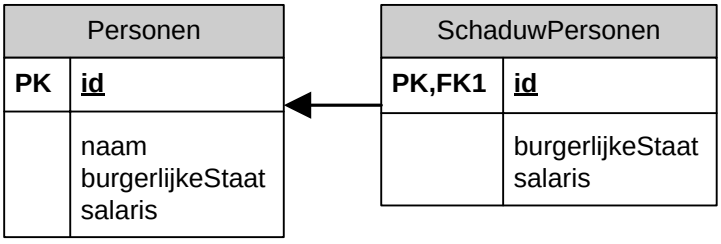
Een nadeel van deze aanpak is het aantal tabellen. Als er veel kolommen uit verschillende tabellen zijn die temporeel opgeslagen moeten worden zullen er veel schaduwtabellen ontstaan. Hierdoor kan het overzicht gemakkelijk verloren gaan. Voor de te maken tool zal dit geen probleem zijn doordat deze tabellen gegenereerd worden. De ontwikkelaar werkt met zijn initiële datamodel en hoeft de schaduwtabellen niet te zien.

Net als bij de oplossing in paragraaf 5.3.1 moet voor elk attribuut een join worden gedaan naar de bijbehorende schaduwtabel. Door de hoeveelheid aan joins (of subqueries) kan ook hier de performance onder de maat zijn en het maken van queries complex worden. Door views of stored procedures te gebruiken, kan de complexiteit van queries worden verminderd.

5.3.3 Schaduwtabel met XML als datatype voor attributen

Een complex datatype kan ook gebruikt worden om temporele data in op te slaan. In dit complexe datatype worden naast de temporele waarden, ook de start- en einddatum opgeslagen van de temporele periode die geldt voor de waarde. In een SQL Server database kan hiervoor XML gebruikt worden (een user-defined type is hiervoor ook een optie). Figuur 5.5 geeft hier een voorbeeld van.

De kolommen ‘burgerlijkeStaat’ en ‘salaris’ zijn beide temporeel en hebben in de schaduwtabel het datatype XML. Voor elk record in de tabel ‘Personen’ bestaat er maar één record in de tabel ‘SchaduwPersonen’, daarom is de primaire sleutel van deze tabel tevens de vreemde sleutel naar de tabel ‘Personen’ toe. Overigens zou het ook mogelijk zijn om de kolommen ‘burgerlijkeStaat’ en ‘salaris’ in de schaduwtabel te combineren in één XML kolom waarin alle waarden worden opgeslagen.



Figuur 5.5, attribuut timestamping op attributen van tabel 'Personen' met XML datatype voor temporele kolommen in schaduwtabel.

In tabel 5.9 wordt een deel van de mogelijke inhoud van de brontabel ‘Personen’ weergegeven. Hierin worden ook de meest actuele waarden van de burgerlijke staat en het salaris bijgehouden. Tabel 5.10 bevat een deel van de mogelijke inhoud van de schaduwtabel ‘SchaduwPersonen’. Voor het gemak zijn de burgerlijke staat en salaris

waarden in een leesbaar formaat weergegeven, maar in SQL Server zouden ze in een XML formaat worden opgeslagen.

Id	Naam	Burgerlijke staat	Salaris
1	Jan Jansen	Gescheiden	1500

Tabel 5.9, deel van de inhoud van de brontabel 'Personen'.

Id	BurgerlijkeStaat		Salaris
1	Ongehuwd; Gehuwd; Gescheiden;	1955-09-10; 1980-09-04 1980-09-05; 1982-02-01 1982-02-02; 9999-12-31	1000; 1955-09-10; 1980-09-04 1500; 1980-09-05; 9999-12-31

Tabel 5.10, deel van de inhoud van de schaduwtabel 'SchaduwPersonen'.

Een voordeel van deze oplossing is dat elk record in de brontabel maar één record in de schaduwtabel heeft. Hierdoor hoeft maar één join uitgevoerd te worden met de schaduwtabel, in tegenstelling tot de voorgaande twee oplossingen.

Een nadeel is dat bij het uitvoeren van een SQL query je een complex datatype terugkrijgt voor temporele data en niet meteen de juiste waarde. In SQL Server is het wel mogelijk waarden uit een XML datatype te lezen met behulp van een SQL query. Dit kan echter wel een performance probleem opleveren doordat een XML datatype veel verschillende temporele waarden kan bevatten, waardoor het zoeken naar de juiste waarde de performance kan verminderen.

Wel is het mogelijk om XML types te indexeren, waardoor deze onder water om worden gevormd naar een relationeel model. Hierdoor houd je het overzicht over de temporele waarden doordat ze in een XML formaat worden weergegeven, maar kun je de performance verbeteren door dat onder water gebruik wordt gemaakt van een relationeel model.

Ook kunnen queries complex worden, doordat naast het ophalen van de data, ook de XML kolommen doorzocht moeten worden. Door het gebruik van bijvoorbeeld een view of stored procedure kan dit probleem worden verholpen.

Net als het tuple timestamping datamodel beschreven in subparagraaf 5.2.2, hebben de in deze paragraaf beschreven attribuut timestamping datamodellen hetzelfde probleem als ook gegevens over de toekomst kunnen worden opgeslagen. De actuele waarden die in de brontabel worden bijgehouden kunnen verlopen, waardoor ze niet meer actueel zijn. Dit kan bij de datamodellen uit subparagraaf 5.3.1 en 5.3.2 opgelost worden door de temporele kolommen geheel te verwijderen uit de brontabel en alle temporele waarden op te slaan in de schaduwtabel(len), inclusief de actuele waarden. Deze oplossing kan ook gebruikt worden voor het datamodel uit subparagraaf 5.3.3. Omdat hier de brontabel en schaduwtabel een 1-op-1 relatie met elkaar hebben kan de schaduwtabel ook in zijn geheel worden verwijderd, waarbij de temporele waarden in de brontabel worden opgeslagen. De brontabel zal dus de XML kolommen gaan bevatten.

Na het uitvoeren van een performancetest is gebleken dat het datamodel uit subparagraaf 5.2.2, waarbij een schaduwtabel voor elk attribuut wordt aangemaakt, de beste resultaten levert. Qua gebruikersgemak bij het schrijven van queries komt dit datamodel ook als beste naar voren.

6. Conclusies en aanbevelingen

Het antwoord op de hoofdvraag ‘Hoe richt ik een SQL Server database in als temporele database?’ hangt van een aantal zaken af. Er zijn een aantal dingen waar rekening mee moet worden gehouden en die de oplossing kunnen beïnvloeden.

Ten eerste is het belangrijk dat je weet welke temporele waarden je wilt opslaan. Wil je historische gegevens opslaan die de ‘echte’ wereld weergeven dan kies je voor een historische database. Deze houdt de valid time van temporele gegevens bij. Wil je bijhouden wanneer gegevens zijn toegevoegd, gewijzigd of verwijderd uit een database, dan kies je voor een database die de transaction time bijhoudt. Een voordeel van het bijhouden van de transaction time is dat je wijzigingen die zijn uitgevoerd kunt terugdraaien naar een bepaald punt in de tijd. Dit wordt een rollback genoemd. Wil je beide, dan kies je voor een bitemporal database. Deze houdt zowel de valid time als transaction time bij.

Voor de manier waarop temporele waarden worden opgeslagen zijn ook een aantal keuzes mogelijk. De eerste is het gebruik van een instant. Deze geeft een moment aan waarop iets waar was, bijvoorbeeld een datum. Een interval geeft een duur aan wanneer iets waar was, bijvoorbeeld een week. Het specificeert echter niet wanneer de week begint of eindigt. In combinatie met een instant kan een interval dit wel weergeven. Als laatste is er de periode die aangeeft van wanneer tot wanneer een temporele waarde geldig is. Deze optie is het meest flexibel doordat hiermee eventueel ook een instant kan worden aangegeven (begin- en eind van periode zijn hetzelfde) of een interval kan worden berekend.

Voor het opslaan van temporele gegevens in een tabel zijn er een aantal mogelijkheden. Je hebt tuple timestamping, waarmee je op recordniveau gaat bijhouden wanneer de gegevens in het record geldig waren. Dit type timestamping is vooral handig als gegevens in een record vaak tegelijk wijzigen en enige redundantie van gegevens geen probleem is.

Bij attribuut timestamping zul je per attribuut bijhouden wanneer een attribuut een bepaalde waarde had. Dit type timestamping kan het best gebruikt worden als de temporele attributen in een record vaak afzonderlijk van elkaar wijzigen. Zijn er in verhouding tot het aantal kolommen in een tabel maar weinig kolommen die temporeel moeten worden opgeslagen, dan is dit type timestamping ook een goede keuze doordat een grote hoeveelheid redundantie zal worden voorkomen.

Als er een keuze is gemaakt tussen de bovenstaande opties zijn er een aantal databasemodellen die gebruikt kunnen worden voor de inrichting van een temporele database. Voor de opdracht die tot deze whitepaper heeft geleid is een scheiding van gegevens wenselijk. Voor tuple timestamping is het daarom wenselijk om een schaduwtabel aan te maken waarin temporele waarden worden opgeslagen. Hierdoor is de temporele data gescheiden van actuele data. Voor attribuut timestamping is het gebruik van een schaduwtabel per attribuut de beste keuze qua performance en gebruikersgemak.

7. Begrippenlijst

Attribuut is een kolom in een tabel.

Attribuut timestamping is het timestampen van kolommen/attributen in een tabel.

Bitemporal time is een combinatie van valid time en transaction time.

Bitemporal time database is een temporele database die de bitemporal time van temporele waarden bijhoudt.

Datamodel beschrijft hoe de gegevens in een database zijn gestructureerd.

Granulariteit is het detailniveau dat wordt bijgehouden door een periode. Denk hierbij aan seconden, uren, minuten, dagen, maanden, jaren, etc.

Rollback is het terugdraaien van wijzigingen op een database tot voor een bepaald punt.

Schaduwtabel is een tabel waarin temporele data wordt bijgehouden over data uit een brontabel.

Temporele data is de combinatie van temporele waarden en de tijd waarvoor ze geldig zijn (valid time, transaction time of bitemporal time).

Temporele database is een database die het mogelijk maakt om alle toestanden van een object op te slaan en op te halen, die het tijdens zijn leven heeft aangenomen of nog zal aannemen.

Temporele datatypen zijn typen waarin de valid time, transaction time of bitemporal time van een temporele waarde kan worden opgeslagen. Mogelijkheden zijn instant, interval en periode.

Temporele waarde is een waarde die temporeel wordt opgeslagen. Voor bijvoorbeeld de burgerlijke staat kan dit ‘ongehuwd’ of ‘gehuwd’ zijn.

Timestamp is een datum en/of tijd die aangeeft waarop iets is gebeurt of geldig was.

Toestand van een object of database is de state van een object of database.

Transaction time is de tijd waarop een waarde/toestand van een object in een temporele database geldig was.

Transaction time database is een temporele database die de transaction time van temporele waarden bijhoudt.

Tuple timestamping is het timestampen van records in een tabel.

User-defined type is een datatype in SQL Server die is gespecificeerd door een gebruiker.

Valid time is de tijd waarop een waarde/toestand van een object in de werkelijkheid/’echte’ wereld geldig was.

Valid time database is een temporele database die de valid time van temporele waarden bijhoudt.

8. Literatuurlijst

[1] Andreas Steiner (1998). *A Generalisation Approach to Temporal Data Models and their Implementations*. http://www.globis.ethz.ch/people/former/thesisSteiner.pdf (22-09-2008).

[2] Microsoft Corporation (2008). *Creating a User-Defined Type.* http://msdn.microsoft.com/en-us/library/ms131106(SQL.90).aspx (06-10-2008).

[3] Microsoft Corporation (2008). *Datetime and smalldatetime.* http://msdn.microsoft.com/en-us/library/aa258277.aspx (24-09-2008).

[4] Nibblus (2004). *Temporal database*. http://en.wikipedia.org/wiki/Temporal_database (18-08-2008).

[5] Patricia Nogueira Hubler, Nina Edelweiss. *Implementing a Temporal Database on Top of a Conventional Database.* http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.8977 (01-10-2008).

[6] Richard T. Snodgrass (2000). *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann Publishers. http://www.cs.arizona.edu/people/rts/tdbbook.pdf

9. Index

10. Over Info Support

Info Support is opgericht in 1986 en is met ruim 350 medewerkers in Nederland een vooraanstaand IT-dienstverlener op het gebied van IT-consultancy, software -ontwikkeling, opleidingen en beheer. Info Support is niet beursgenoteerd en financiert de verdere ontwikkeling van de organisatie op basis van een beheerste groei uit eigen middelen.

Onze drive achter de oplossingen die wij realiseren voor onze klanten is er sterk op gericht bedrijfsprocessen sneller en beter te maken. Info Support ontwikkelt en beheert solide en innovatieve softwareoplossingen die organisaties ondersteunen bij het realiseren van hun doelstellingen.

De kernwaarden Soliditeit, Integriteit, Vakmanschap en Passie typeren onze werkwijze, waarin we sociaal en solide management belangrijker vinden dan omzetmaximalisatie. Ons hoogste doel is dat we met opdrachtgevers en medewerkers willen bouwen aan langetermijnrelaties. Daarbij houden we ons aan gemaakte afspraken. Dit maken we in de praktijk waar, getuige de jarenlange relaties die we met onze klanten hebben. Info Support mag zich al 16 jaar op rij TOP-IT-werkgever van het jaar noemen.

Zie voor meer informatie www.infosupport.com.