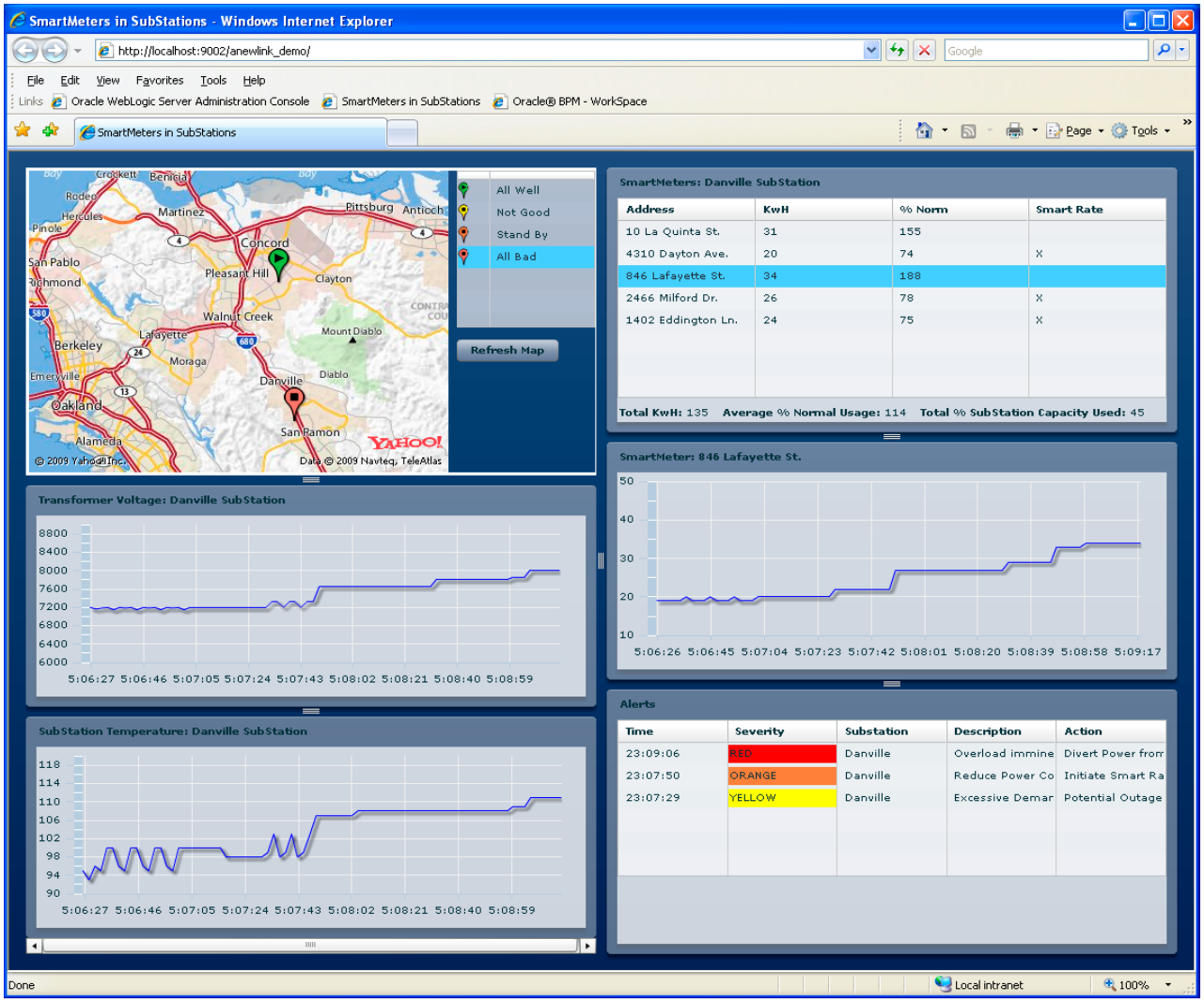
Oracle EDA Suite

Geavanceerd gereedschap voor Event Driven Architecture

In Optimize nummer 3 van dit jaar hebben we de concepten kunnen lezen van het jongste paradigma in de software-ontwikkeling: Event Driven Architecture. Ditmaal duiken we in de suite die Oracle levert als gereedschap om Event Driven applicaties te ontwikkelen. De belangrijkste component daarvan is de Complex Event Processor (CEP). CEP 11gR1 is als onderdeel van Fusion Middleware 11g inmiddels gereleased. Harold Gerritsen laat ons er een kijkje in nemen en geeft de verbeteringen aan ten opzichte van de vorige versie.



Afbeelding 1: EDA zichtbaar in actie. Dashboard van slimme energiemeters.

Om relatief eenvoudig EDA toepassingen te kunnen ontwikkelen heeft Oracle de EDA Suite samengesteld. Deze bevat zoals we eerder zagen een viertal producten. Dit zijn respectievelijk:

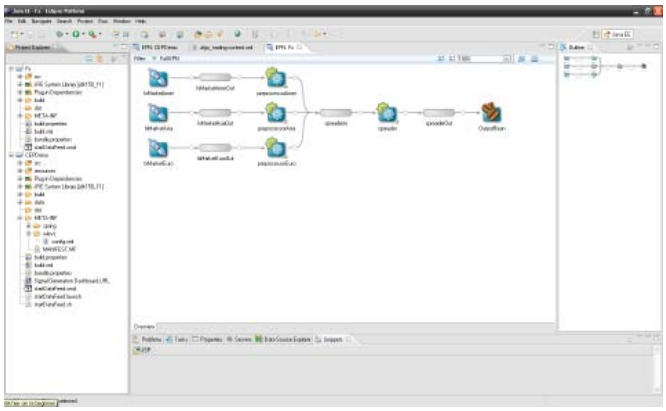
- Business Activity Monitoring (BAM)
- Oracle Service Bus (OSB)
- Business Rules (BR)
- Complex Event Processor (CEP)

Elk van deze producten vormt een stukje van de puzzel bij het ontwikkelen van EDA-applicaties. Allereerst de OSB: deze vormt de poort naar de buitenwereld. Elders waargenomen events dienen in electronische vorm, dat wil zegen bij voorkeur in de vorm van een XML-bericht, aan de servicebus aangeboden te worden. De OSB zal de berichten valideren en op basis van de inhoud selectief routeren naar de juiste ingang (bijvoorbeeld JMS Queue) van de CEP.
De CEP staat opgesteld om de aangeboden events te verwerken, wat eigenlijk neerkomt op het 'veredelen' van de informatie die uit de optredende events kan worden afgeleid. Op basis van het ontstane inzicht zal de CEP vervolgens die kennis gebruiken om een sterk uitgedunde stroom business events te genereren en deze bijvoorbeeld terug te geven aan de OSB.
Een andere mogelijkheid is dat de CEP zijn output afrondt door het triggeren van een stuk business logica in de vorm van een zogenaamd Plain Old Java Object (POJO).
In deze POJO is natuurlijk alles mogelijk. Zo kan de Business Rules Engine worden aangesproken, de derde component van de suite, om het onstane inzicht te toetsen tegen de geldende business rules.
Oracle BAM is aan de suite toegevoegd om ontstane inzichten te visualiseren voor de eindgebruiker (zie het voorbeeld in afbeelding 1). Om verwarring te voorkomen zij hierbij opgemerkt dat Oracle CEP zelf ook een component bevat met de naam 'Visualizer' (zie afbeelding 6). Deze is echter niet bedoeld voor eindgebruikers, maar als systeemmonitor voor de beheerder van de betreffende EDA applicatie.
Zo zien we dat elk product in de suite z'n bijdrage levert. Maar het product waar het bij EDA natuurlijk allemaal om draait is de Complex Event Processor. Omdat Oracle CEP een implementatie is van wat in de literatuur bekend staat als een DSMS, zullen we daar eerst kort bij stilstaan.

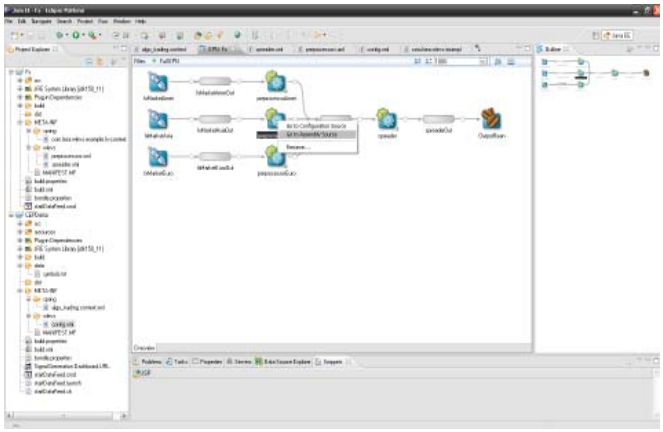
Chocola

Zoals we al hebben kunnen lezen draait het er bij Complex Event Processing om dat er van een ongrijpbare hoeveelheid ongerelateerde gebeurtenissen 'chocola wordt gemaakt', ofwel dat deze worden omgezet naar een beperkter aantal voor de business relevante gebeurtenissen, waarop vervolgens actie kan worden ondernomen. Hierbij is het voor de traditionele Oracle-ontwikkelaar wellicht moeilijk voor te stellen, maar een database heeft daarbij zo zijn beperkingen. Een database is sterk in het vastleggen (bij wijze van spreken 'in beton gieten') van data onder een zeer professioneel regime van beschikbaarheid, recoverability, autorisatie, auditing etcetera, waarna de data bij regulier gebruik meerdere keren met behulp van queries wordt geraadpleegd. Als we een database willen inzetten bij CEP betekent dit dat we eerst alle optredende events gaan vastleggen en daarna met behulp van SQL de overbodige events gaan verwijderen danwel de events gaan aggregeren tot relevante business events. Da's leuk, maar op het moment dat dit gedaan moet worden met wel een miljoen optredende events per seconde, dan is het al wat gemakkelijker voor te stellen dat dit niet helemaal de juiste weg is. Een seconde later staan er immers mogelijk al weer een miljoen nieuwe events voor de deur. Remedie is natuurlijk wat iedereen ook al weet uit de privésfeer: als je gaat verhuizen gooi je bij voorkeur eerst overbodige zaken weg voor je aan het verplaatsen slaat. Als je dit principe toepast op EDA-applicaties, dan betekent dit dat je er eigenlijk eerst 'chocola van moet maken', en dan pas verder verwerken in een applicatie, danwel opslaan in een RDBMS. Oracle-kenners zeggen dan natuurlijk meteen: "het kan echt wel in een RDBMS, we hebben immers Oracle Times-Ten, de in-memory database". Dat is dan niet eens zo ver bezijden de waarheid. Maar voor CEP is er meer nodig dan alleen opslaan en filteren. In essentie moeten we immers stromen van events kunnen monitoren en daarbij ogenschijnlijk niet gecorreleerde events kunnen correleren in patronen, zodat er relevante informatie ontstaat. En dan blijkt dat niet alleen de snelheid van het RDBMS beperkend te zijn, maar ook de taal die in een RDBMS wordt gebruikt. Voor het filteren en correleren van informatiestromen schiet SQL tekort en hebben we een taal nodig met een rijkere woordenschat. Eigenlijk willen we met een uitgebreidere versie van SQL realtime statements uitvoeren op eventstromen voor ze in het RDBMS terechtkomen. En, ook nèt even anders, de statements moeten continu uitgevoerd blijven worden, terwijl de events blijven toestromen. Hierbij dienen we te beseffen dat dit de omgekeerde wereld is met wat een RDBMS doet. Daar is de opgeslagen data immers (vrijwel) stabiel en worden er voortdurend SQL-statements op afgevuurd. In EDA-toepassingen echter definieer je tot nader order een SQL-achtige rule die vervolgens wordt toegepast op alle events die maar blijven toestromen en die bij wijze van spreken juist op de rule worden afgevuurd.
Een product dat zo iets mogelijk maakt, heet zoals eerder aangegeven een Data Stream Management System (DSMS). De requirements voor een goed DSMS (met tussen haakjes de voorziening in de EDA-suite die hierin voorziet) kunnen we samenvatten in de volgende zes punten:

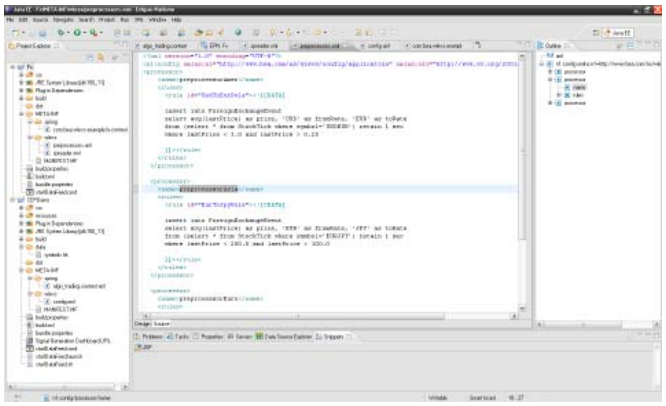
- We moeten makkelijk kunnen aansluiten op allerlei bronnen en bestemmingen van events (Adaptors);



Afbeelding 2a: Ontwikkelen van Event Processing Network in Eclips...



Afbeelding 2b: ...waarbij je met behulp van Right Mouse Click op een Processor component naar de Assembly Source gaat...



Afbeelding 2c: ...en je dan uitkomt in een XML-editor waarmee je de EPL rule kunt editen.

- We moeten makkelijk stromen events (Streams) kunnen samenvoegen en splitsen en de inhoud van die stromen kunnen monitoren (Profiles);
- We moeten makkelijk in bewerkingsstappen (Processors) patronen kunnen definiëren (Processor Rules) en daarmee complexe events omzetten in relevante business events;
- Als we hebben geconstateerd dat een event relevant is willen we daar eenvoudig actie op kunnen ondernemen in een applicatie (OutputBean) of visualiseren (Adaptor en BAM Dashboard);
- Dit alles willen we liefst eenvoudig kunnen ontwikkelen door dit in stappen grafisch aan elkaar te koppelen (Event Processing Network);
- Ten slotte willen we een standaard taal kunnen hanteren waarin we de op de event-stroom toe te passen statements kunnen uitdrukken (Continuous Query Language).

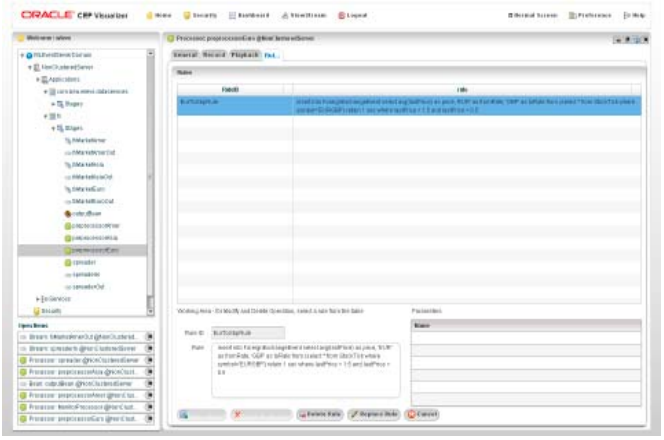
Complex Event Processor

De Complex Event Processor van Oracle is dus zo'n DSMS. De overname van BEA maakte het er voor Oracle op dit punt niet gemakkelijker op: zowel Oracle als BEA hadden een eigen CEP product. Die van Oracle had als voordeel dat de CEP statements in Continuous Query Language (CQL) konden worden geschreven, een taal die op dit moment uitgroeit tot de marktstandaard voor DSMS'en. BEA hanteerde door remmende voorsprong haar eigen Event Processing Language (EPL). Ondanks het niet ondersteunen van de marktstandaard heeft Oracle er voor gekozen de 10gR3 versie van de EDA Suite te baseren op BEA's CEP product. De belangrijkste drijfveer hiervoor was waarschijnlijk het fors hogere marktaandeel van de BEA WebLogic Event Server, zoals de BEA CEP heette toen het nog onderdeel uitmaakte van BEA's eigen productportfolio. Zoals Oracle het afgelopen jaar bewust vaag omschreef, zouden we 'in de zomer van 2009' worden verblijd met Oracle CEP 11gR1. De belangrijkste wijziging hierin is dat de EPL-engine is vervangen door c.q. uitgebreid tot een CQL-engine. Voorts is de look en feel in het ontwikkeltool en de beheermonitor (de 'Visualizer') wat strakker geworden. Zie voor een voorbeeld hiervan afbeelding 3 (10gR3) versus afbeelding 5 (11gR1). Daarnaast zijn de tools iets toegankelijker gemaakt door het toevoegen van bijvoorbeeld een Query Wizard voor het definiëren van CQL rules. Hierdoor is er absoluut minder kennis en ervaring nodig om met deze taal uit de voeten te kunnen.

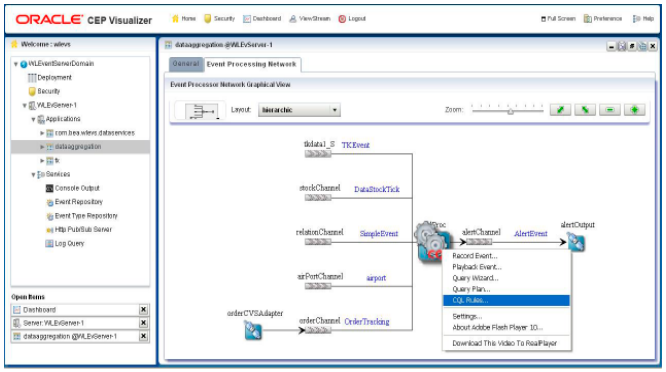
Ontwikkeling en beheer met CEP

Nu we weten wat Oracle CEP functioneel voor ons kan betekenen is het tijd om een kijkje te nemen onder de motorkap. Oracle CEP is eigenlijk een heel lichtgewicht Java-applicatie die je configureert in XML en die je na het deployen runtime kunt beheren. De werking van Oracle CEP wordt goed gevisualiseerd door de verschillende illustraties in dit artikel. In afbeelding 2a tot en met 2c wordt het ontwikkelproces weergegeven.

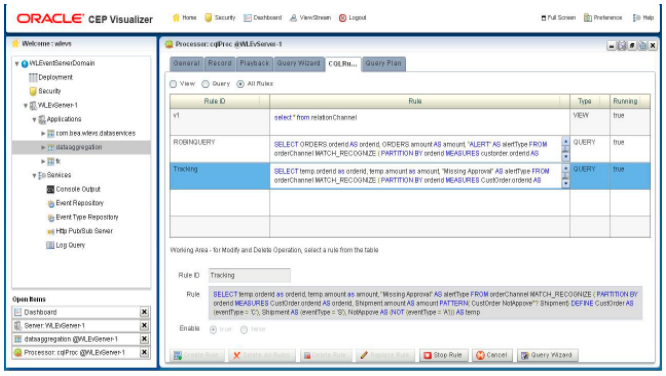
Het ontwikkelen gebeurde in 10gR3 nog uitsluitend op het Eclipse-platform, net zoals dat met BPEL na de overname van Collaxa aanvankelijk alleen op het Eclipse-platform kon wor-



Afbeelding 3: Het beheren van EPL Rules in de 10gR3 versie van de CEP Visualizer.

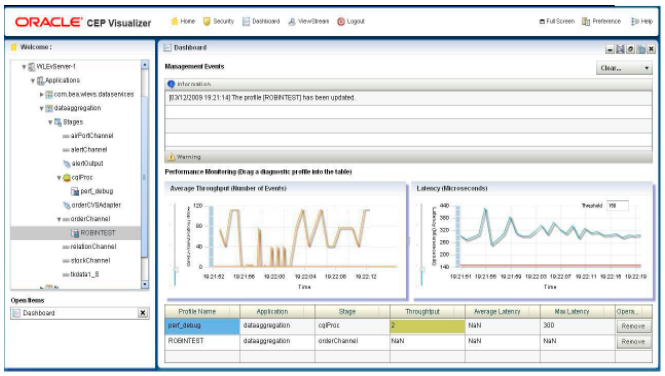


Afbeelding 4: Look and feel van het beheren van het Event Processing Network in de 11g versie van de CEP Visualizer.



Afbeelding 5: Het beheren van CQL Rules in de 11g versie van de CEP Visualizer (analoog met afbeelding 3).

den gedaan. Conceptueel bestaat er grote overeenkomst tussen het ontwikkelen voor Oracle CEP en het modelleren van een ETL (Extract-Transform-Load) flow in een Data Warehouse omgeving zoals m.b.v. Oracle Warehouse Builder. De ETL flow is in het geval van CEP een zogenaamd Event Processing Network (EPN).
Afbeelding 2a toont een EPN met Adaptors, Streams,



Afbeelding 6: System Management Dashboard in VIsualizer 11g voor het bewerken van de correcte werking van de Event Server.

Processors en een Output Bean. De voorbeeld-EPN in deze afbeelding heeft als taak de financiële markten van de Euro, de Dollar en de Yen in de gaten te houden en bij situaties waarop geacteerd moet worden m.b.v. een Javabean een Java applicatie dat te laten doen. Een ontwikkelaar kan met een Right Mouse Click (zie afbeelding 2b) op een Processorstap naar de XML source gaan van deze processor, waar dan de zogenaamde Processor Rule kan worden bewerkt (zie afbeelding 2c). Eerlijkheid gebiedt hierbij te zeggen dat het bewerken voor een ontwikkelaar in 10gR3 nog redelijk Spartaans genoemd mag worden. In plaats van wizard-achtige dialogen dien je direct de XML-source te editen, niet veel intelligenter dan het gebruik van Notepad voor die taak. In de nieuwste versie van Oracle CEP kan dit vanzelfsprekend ook vanuit JDeveloper worden gedaan.
Het beheer van CEP applicaties doe je met behulp van de Visualizer (afbeelding 3 t/m 6). Deze webapplicatie maakt het bijvoorbeeld mogelijk om een gedeployde EPN in beeld te brengen, daarbinnen een Processor-node te selecteren en dan -net als bij ontwikkeling binnen Eclipse- met een Right Mouse Click (zie afbeelding 4) de CQL source van de bijbehorende Rule te tonen. Desgewenst kan deze Rule gedisabled worden zodat in feite de deployment selectief ongedaan gemaakt kan worden. Een andere mogelijkheid van de Visualizer is de Dashboard functie. Hiermee kan een beheerder van de CEP applicatie systeemmanagement taken uitvoeren zoals het monitoren van de throughput en de latency van de eventstroom in de keten, analoog aan de taken van een netwerkbeheerder die op basis van dezelfde karakteristieken de 'gezondheid' van zijn netwerk in de gaten houdt.

Continuous Query Language

We hebben inmiddels de meeste aspecten van Oracle CEP bekeken. Om het overzicht compleet te maken is het wellicht aardig om nog even stil te staan bij de specifieke querytaal, de Continuous Query Language. In de documentatie op OTN zijn verschillende whitepapers te vinden waarin uitgebreid wordt

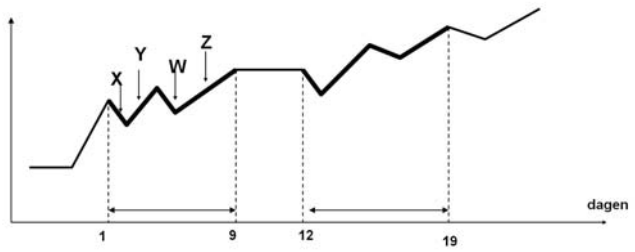
ingegaan op de vele constructies die mogelijk zijn in CQL. We zullen daarom volstaan met een klein voorbeeld. Kenmerkend van CQL is de mogelijkheid voor aggregeren, filteren, samenvoegen en splitsen en correleren van data. Bovendien is de periode waarover de data wordt verzameld te specificeren, alsmede het interval waarop moet worden gemeten.
De meest geavanceerde toepassing van CQL is die van de pattern-matching. De CQL syntax voor het specificeren van dergelijke patronen luidt als volgt:

```
SELECT <select-list>
FROM <stream|relation-name>
MATCH_RECOGNIZE (PARTITION BY)
MEASURES (…)
ONE | ALL ROWS PER MATCH
PATTERN <pattern-expression>
DEFINE <define-alphabets>)
```

Deze constructie kunnen we bijvoorbeeld toepassen bij patroonherkenning in koersverloop van een specifiek aandeel of een aandelenindex. In afbeelding 7 zien we een veel voorkomend patroon in koersontwikkeling van aandelen op de aandelenmarkt, het zogenaamde W-patroon. Dit patroon wordt gekenmerkt door het feit dat in de grafiek van de koersontwikkeling een vijftal buigpunten zijn waar te nemen die (hoe kan het ook anders) precies het beginpunt, het eindpunt danwel de knikpunten vormen van de letter W. Bij het optreden van een dergelijk patroon is vooral de duur van het patroon, d.w.z. de periode tussen het begin van de daling van de linkerarm van de W en het eind van de stijging van de rechterarm van de W, van belang. Met behulp van CQL is het bepalen daarvan door middel van een simpel statement te bereiken. Laten we kijken naar het volgende statement:

```
SELECT FIRST(x.time)
,            LAST(z.time)
FROM     ticker
MATCH_RECOGNIZE (ONE ROW PER MATCH PARTITION BY name
       PATTERN (X+ Y+ W+ Z+)
       DEFINE X AS (price < PREV(price))
                      Y AS (price > PREV(price))
                     W AS (price < PREV(price))
                      Z AS (price > PREV(price)))
```

Hierin zien we in de define-clause aan het eind van het statement dat X staat voor alle gemeten waarden waarvoor geldt dat ze lager waren dan hun voorganger (dit zijn dus alle punten die de linkerarm van de W vormen). Analoog hieraan staat Z voor alle hogere waarden dan hun voorganger, dus de rechterarm van de W. De select-clause kan zo eenvoudig het startpunt en eindpunt van het optredende W-patroon retourneren. In de afbeelding is dat respectievelijk dag 1 en dag 9 en later dag 12 en dag 19. Passen we dit statement toe als CQL-Rule in een Processor node van een EPN zoals we bijvoorbeeld zagen in



Afbeelding 7: W-patroon zoals dat typisch voorkomt in aandelenhandel.

afbeelding 4, dan bereiken we daarmee dat een veelvoud van optredende events (de continue stroom van gemeten aandelenkoersen), wordt teruggebracht tot een minimale stroom van begin- danwel eindpunten van de optredende 'W'-s.

Resumerend

Met de context van Event Driven Architecture in Optimize nummer 3 van dit jaar en de hoog-over beschouwing van de EDA Suite in het voorliggende artikel, kunnen we ons waarschijnlijk een redelijke voorstelling maken van wat de EDA Suite en meer specifiek Oracle CEP voor ons kunnen betekenen. Verrassend genoeg is de suite bijzonder laagdrempelig om mee te werken. Dit komt vooral doordat het in feite een combinatie is van concepten en technologie die al veel langer bestaan in het Oracle domein. Denk hierbij aan ETL flows, een message bus, concepten uit de Java wereld zoals POJO's en JMS Queues en natuurlijk Business Rules en Business Activity Monitoring. Kortom: een Oracle ontwikkelaar die de transitie heeft doorgemaakt van traditionele ontwikkeling met Designer en Developer naar ontwikkelen in Java, kan er in no-time mee aan de slag. Daarom zal de hoeveelheid toepassingen van EDA vooral worden beperkt door de creativiteit van business analisten. De uitdaging is om de mogelijkheden te zien van toepassing van EDA in reguliere bedrijfsprocessen.
Kortom: alles kan! De business is weer aan zet.

Referenties

ReCEPtor, publicatie over Complex Event Processing en de CEP engine van HP:
http://www.hpl.hp.com/techreports/2007/HPL-2007-176.pdf

Oracle SOA op OTN
http://www.oracle.com/soa

Oracle EDA op OTN
http://www.oracle.com/goto/eda,
http://www.oracle.com/technology/products/event-driven-architecture/index.html

Oracle CEP op OTN
http://www.oracle.com/goto/cep

Harold Gerritsen is directeur van A New Link bv.