Op dit moment de enige open source CEP engine

Esper, de omgekeerde database

Jos van Dongen

Open Source databases en query tools zijn er volop te vinden, maar als het gaat om complexere producten, zoals bijvoorbeeld voor event processing toepassingen, is het nog steeds gesloten software die de boventoon voert. In het vorige nummer is al aandacht besteed aan één van deze producten waarbij vooral gekeken is naar het gebruik als real-time BI-tool. Dit artikel legt de nadruk meer op het gebruik van de regels die toegepast kunnen worden op binnenkomende events. Dat zullen we doen aan de hand van de tools van EsperTech. Dit bedrijf heeft als slogan 'Event Stream Intelligence: Continuous Event Processing for the Right Time Enterprise', wat meteen duidelijk maakt waar ze zich mee bezighouden.

Het EsperTech project is opgezet door Thomas Bernhardt, een software engineer werkzaam bij een grote financiële instelling. De ontwikkeling is pas begin 2005 gestart en resulteerde in juli 2006 in versie 1.0 van Esper, een Java based event processing engine. In maart van dit jaar is versie 2.1 opgeleverd met als belangrijkste verbeteringen de mogelijkheid om ongelijke events toch als één event type te behandelen en de ondersteuning voor update events. Het product is eveneens in een .Net/C# variant beschikbaar en heet dan Nesper, echter deze laatste loopt wel 1 of 2 versies achter. Beide producten worden onder een GPL v2 open source licentie beschikbaar gesteld, waarmee Esper/Nesper het enige volwaardige open source CEP product op de markt is. Espertech levert van Esper ook een Enterprise Edition: dezelfde codebase maar een andere – en betaalde – licentievorm. Support is alleen beschikbaar voor de betaalde variant, de OS versie moet het doen met community en forum ondersteuning.
Voor het zware werk is een 'high availability' variant EsperHA beschikbaar, eveneens onder commerciële licentievoorwaarden. Hierin zijn onder andere mogelijkheden voor clustering, in-memory caching en een transparante memory naar disk overflow. De commerciële voorwaarden gelden helaas ook voor EsperJDBC, de service endpoint waarmee vanuit welke omgeving dan ook ingeprikt kan worden op de Esper streams. Zonder EsperJDBC bent u dus aangewezen op andere (maatwerk) software om de query-resultaten te benaderen en verwerken. Ondanks de korte historie kan EsperTech bogen op een aantal in het oog springende resulaten. Merryl Lynch staat bijvoorbeeld prominent op de klantenlijst, en in juli 2007 is een OEM contract gesloten met Bea (nu Oracle) zodat Esper nu het CEP deel vormt van de Bea Weblogic Event Server. Het laatste nieuws is te vinden in de Business Objects Labs. Ook deze BI-mastodont heeft het real-time licht gezien en levert een Event-Driven BI-oplossing met Esper als CEP engine onder de motorkap. Vooralsnog alleen als prototype, maar gezien de oplossingen die concurrenten nu al bieden zal het niet lang meer duren voordat dit als product op de markt komt.

De database op zijn kop

Event stream processing kan eenvoudig gevisualiseerd worden als een omgekeerde database. Waar in een reguliere database de gegevens statisch zijn en bevraagd worden door een dynamische set van query's, is het in een event stream precies andersom. De query's zijn statisch en de gegevens 'stromen' als het ware door de query's heen. Ook de resultaten komen dus als een continue stroom beschikbaar voor gebruik, wat event processing bij uitstek geschikt maakt voor operational BI-toepassingen. Het schema in afbeelding 1 toont hoe Esper globaal is opgebouwd. Hierin is duidelijk te zien dat de data door de statements worden geleid en dit resulteert in resultaatobjecten, in dit geval 'plain old java objects' oftewel POJO's.

> Naast de query-taal beschikt Esper over een 'event pattern language'

Esper werkt als een real-time engine die acties afvuurt op het moment dat een bepaalde stream voldoet aan een vastgelegde conditie. De query's die hiervoor zorgen worden gedefinieerd met behulp van EPL (Event Processing Language), een real-time query-taal die is gebaseerd op SQL. Naast de query-taal beschikt Esper over een 'event pattern language' die bedoeld is om patronen en correlaties tussen events op te sporen. Deze pattern

calls kunnen vervolgens gecombineerd worden binnen een query statement door het patroon als input te gebruiken. Als voorbeeld kunnen we kijken naar een 'stockticker' query die continu een stroom van aandelentransacties met bijbehorende prijzen analyseert. We zijn alleen geïnteresseerd in de transacties (events) van een bepaald aandeel over de laatste minuut waarvan de prijs boven een bepaalde waarde ligt:

```
Select * from pattern [every StockTickEvent
                    (symbol="IBM", price>80) where
                        timer:within (60 seconds)]
```

De mogelijkheden voor het definiëren van patterns zijn eindeloos en kunnen ook gebruikt worden voor het detecteren van ontbrekende data. In het volgende voorbeeld wordt gevraagd om alle events van type A te selecteren die niet binnen 10 seconden gevolgd worden door een event van type B.

```
select * from pattern [every EventA ->
           (timer:interval(10 sec) and not EventB)]
```

De patronen kunnen naast condities, tijdsintervallen en logische operatoren ook gebruik maken van regular expressions, wat een verdere verrijking betekent van de analysemogelijkheden.

Streaming query's

Zoals al aangegeven beschikt Esper over een eigen SQL-variant genaamd EPL waarvan hierboven al voorbeelden zijn opgenomen. Nu is 'select * from' niet echt een wereldschokkend fenomeen en gelukkig is de taal ook niet hiertoe beperkt. Een van de meest nuttige uitbreidingen is uiteraard het tijdsvenster dat gedefinieerd kan worden. In de meeste gevallen heeft een query betrekking op een aantal events (bijvoorbeeld de laatste 100 of 1000) of de events binnen een bepaald window (bijvoorbeeld de laatste vijf minuten). Om de gemiddelde prijs van alle aandelentransacties van de laatste dertig seconden te berekenen kan het volgende statement worden gebruikt:

```
select avg(price) from
                    StockTickEvent.win:time(30 sec)
```

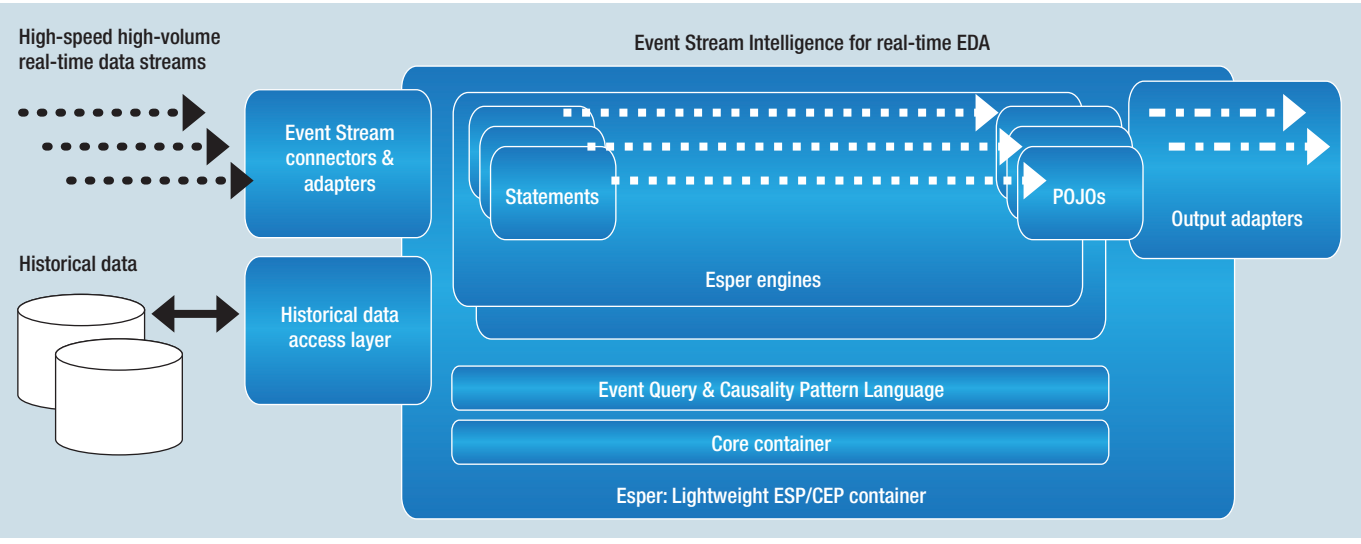
Als u zich misschien afvraagt hoe vaak deze query wordt uitgevoerd, ga dan even een paar paragrafen terug: de query is permanent, de data stromen. Het resultaat van het geheel is dan ook een permanente stroom van gemiddelde prijzen over de laatste 30 seconden. Voor de meeste toepassingen is dat niet zo handig, vandaar dat de taal beschikt over een output restrictie. Door 'output every 5 seconds' toe te voegen aan bovenstaande query zal het resultaat om de vijf seconden worden doorgegeven aan de output adapter.

Bouwstenen

Een van de belangrijkste bouwstenen van EPL query's heeft u al in een aantal voorbeelden voorbij zien komen. Het gaat hierbij om de View die enigszins afwijkt van de views zoals we die in 'normaal' SQL kennen. Een view bepaalt de verzameling events waarop een selectie toegepast kan worden, zoals het win:time(30 sec) venster uit het laatste voorbeeld. Esper kent data window views zoals win:time(interval) en win:length(aantal) maar ook views die statistieken af kunnen leiden zoals bijvoorbeeld gewogen gemiddelden. Als voorbeeld volgt een query die het gewogen gemiddelde berekent van de Google aandelen over de laatste vijf seconden:

```
select average
       from StockTickEvent(symbol='GOOG').win:time(5
           seconds).stat:weighted_avg(price, volume)
```

Hier is ook te zien dat de stat:weighted_avg view twee parameters mee krijgt én een voorgedefinieerde output parameter



Afbeelding 1: Esper architectuur.

'average' voorschrijft. Het gedeelte achter de 'from' is de View. Andere krachtige mogelijkheden worden gevormd door het herschreven 'insert into' statement. In een RDBMS resulteert dit in een nieuwe tabel waarin het resultaat van een query wordt weggeschreven, in EPL is het resultaat een nieuwe event stream op basis van een selectie van één of meerdere bestaande streams. Deze nieuwe stream kan vervolgens weer als input dienen voor nieuwe query's en gejoined of gecorreleerd worden met al bestaande streams. EPL biedt eveneens de mogelijkheid om SQL data te benaderen en zelfs te joinen met event gegevens.

> Een van de meest nuttige uitbreidingen is het tijdsvenster dat gedefinieerd kan worden

En alsof dat nog niet genoeg is beschikt Esper ook over een techniek om niet-relationele data te joinen met event streams. Zo kan bijvoorbeeld een webservice worden benaderd, maar ook gegevens die door een applicatie in memory worden vastgehouden kunnen op deze manier worden gekoppeld. Alle soorten en maten joins worden ondersteund (inner, left outer, right outer, full outer), evenals simpele en gecorreleerde subquery's. En dat alles over een ongelimiteerd aantal streams en/of andere databronnen. Verder kunnen variabelen worden gebruikt en in een uitgebreide set operatoren en functies is eveneens voorzien. Mocht dit nog onvoldoende zijn dan kunt u zelfs uw eigen functies gebruiken, aangezien EPL gebruik kan maken van User Defined Functions.

Toegang

De Esper engine is geschreven in Java en kan opgenomen worden in een willekeurig Java proces. Voor gebruik van de real-time data in externe applicaties, zoals BI-tools, is de JDBC driver onontbeerlijk. Deze bestaat uit twee delen, een server endpoint en de client driver zelf. De JDBC driver kan stream data op twee manieren toegankelijk maken. Named Windows worden weergegeven als reguliere tabellen, terwijl EPL statement-resultaten worden vertaald in stored procedures. Business Objects heeft op een vergelijkbare wijze een real-time designer ontwikkeld waarmee streams gekoppeld kunnen worden aan universe-objecten. Voor client-applicaties als Xcelsius of WebIntelligence is het gegeven dat het om real-time data gaat dan ook volledig transparant. De JDBC driver is echter (helaas) geen open source product maar is op aanvraag wel als trial-versie verkrijgbaar. Het Business Objects prototype kan wel gewoon worden gedownload; dus bent u BO-gebruiker en wilt u zelf eens stoeien met event streams, dan is dit een laagdrempelige manier om met deze technologie kennis te maken.

Performance

Een van de eerste dingen waarover potentiële gebruikers van een dergelijke oplossing zich zorgen maken is natuurlijk de performance. Het feit dat het Java software betreft zal al enige wenkbrauwen doen fronsen, in dit geval echter volledig onterecht. Hoewel formele streaming benchmarks ontbreken, levert EsperTech wel een paar indrukwekkende cijfers af. Op een systeem met dual Intel 2 GHz processoren (een entry level server dus) worden 500.000 events per seconde verwerkt waarbij 1000 statements zijn gedefinieerd. Gemiddeld is de processorcapaciteit dan voor 85 procent benut en is de vertraging (latency) minder dan drie microseconden. Op een laptop zijn (afhankelijk van de configuratie) resultaten tussen 70.000 en 200.000 events per seconde behaald. En dat met een stuk software dat slechts een paar MB geheugen- en diskruimte nodig heeft.

Conclusie

Het is ongelooflijk wat EsperTech in zo'n korte tijd voor elkaar heeft gekregen. De hulp van de community is hierbij waarschijnlijk doorslaggevend geweest. Esper is op dit moment de enige open source CEP engine en gezien het prijskaartje dat aan commerciële varianten hangt, is het blijkbaar interessant om mee te werken aan dit project. Dit is trouwens ook de belangrijkste reden geweest voor Bernhardt om dit project te starten. Er was geen geld voor een dure CEP-oplossing of de beschikbare tools voldeden niet. Het resultaat is er echter naar, en over de engine en de mogelijkheden van EPL dan ook niets dan lof. Let echter op: u dient een doorgewinterde Java-programmeur te zijn of tot uw beschikking te hebben om iets met dit product uit te kunnen richten. Er is geen GUI en voor de JDBC driver die het simpel maakt om op de engine in te prikken en resultaten uit te lezen dient betaald te worden. Wilt u toch de kracht en de mogelijkheden zelf ontdekken, kijk dan eens in de labs van Business Objects of download een evaluatieversie van de Bea Weblogic Event Server.

Referenties

Espertech:	*www.espertech.com*
Esper community:	*http://esper.codehaus.org/*
BO demo:	*http://labs.businessobjects.com/edbi/*
Interview:	*http://rulecore.com/CEPblog/?p=12*
Open Source CEP	*www.ebizq.net/hot_topics/cep/features/9525.html?page=1*

Meer weten over Complex Event Processing:
http://knol.google.com/k/hans-gilde/complex-event-processing#
Gartner wijdt een complete summit aan complex event processing:
www.gartner.com/it/page.jsp?id=616710

Jos van Dongen

Jos van Dongen (jvdongen@tholis.com) is Senior Consultant bij Tholis Consulting.