‘Operational analytics’ vereisen een grotere hoeveelheid data

Changed Data Capture

Jos van Dongen en Cees van Tilburg

Over het modelleren en vullen van datawarehouses is veel gepubliceerd, maar een belangrijk onderdeel van het extractieproces wordt daarbij vaak onderschat. Het gaat dan om het identificeren en extraheren van gewijzigde data in het bronsysteem, zodat alleen nieuwe of gewijzigde records hoeven te worden verwerkt in het ETL-proces. Over het algemeen wordt dit aangeduid met de term Changed Data Capture of CDC.

In zijn Data Warehouse ETL Toolkit stelt Kimball dat “capturing changes to the source system content is crucial to a successful Datawarehouse”. Toch worden er slechts zes pagina’s aan dit onderwerp gewijd met alleen beschrijvingen van verschillende mogelijkheden. Dit artikel is bedoeld om concrete handvatten te geven die direct toegepast kunnen worden en geeft tevens aanwijzingen in welke situaties een bepaalde oplossing meer of juist minder geschikt is. Overigens beperken we ons hierbij tot gestructureerde informatie die vastligt in relationele databases. Na de algemene beschrijving en toelichting van de technieken zal de laatste versie van de Oracle database (10g2) tegen het licht gehouden worden om te bezien hoe deze database het CDC-proces kan ondersteunen. Voor de volledigheid vermelden we nog dat ook andere DBMS’en zoals SQL Server en DB2 vergelijkbare mechanismen bevatten.
Om de principes toe te lichten maken we gebruik van een zeer eenvoudig voorbeeld. Een tabel met drie kolommen is hierbij voldoende, met een ID als sleutel, een tekstveld en een veld met de datum en tijd van de laatste wijziging. Er zijn twee toestanden van de data, één op 10 oktober 23:59:59 en één op 11 oktober 23:59:59, zie afbeelding 1. Er is dus een record gewijzigd, een record verwijderd en een record toegevoegd, wat voor de rest van dit verhaal voldoende is.

Globaal gezien kunnnen er twee typen CDC-oplossingen onderkend worden. De eerste is database-gebaseerd en maakt gebruikt van audit-kolommen of snapshots, de tweede is gebaseerd op het uitlezen van wijzigingen uit transactie-logs. Het eerste type wordt meestal aangeduid met de term ‘intrusive’ (binnendringend) of ‘invasive’, de tweede met ‘non-intrusive’ of ‘non-invasive’.
Intrusive technieken vereisen bewerkingen op de database en hebben direct effect op de performance omdat er tabellen verwerkt en query’s afgevuurd worden. De non-intrusive technieken hebben weinig tot geen impact op de performance van het bronsysteem maar zijn meestal wat complexer om in te richten. Eén ‘techniek’ die overigens nog vrij vaak voorkomt is de ‘full-load’. Hierbij wordt het complete datawarehouse telkens geheel leeggemaakt en opnieuw opgebouwd met de meest verse data uit het bronsysteem. We zullen hier niet verder op in gaan.

Intrusive technieken

Zoals gesteld bestaan deze technieken uit standaard database-bewerkingen en hebben dan ook als voordeel dat er geen speciale hulpmiddelen vereist zijn om veranderingen te identificeren, en het feit dat ze database-onafhankelijk zijn. Het nadeel is uiteraard dat de vereiste bewerkingen invloed hebben op de database performance en daarom eigenlijk alleen geschikt zijn wanneer er een batch-window beschikbaar is waarbinnen de database

Toestand 1:

ID	Tekst	DatumTijd
1	Zwart	10-10-2006 15:23:34
2	Groen	10-10-2006 17:23:22
3	Rood	10-10-2006 19:33:24
4	Blauw	10-10-2006 20:19:43

Toestand 2:

ID	Tekst	DatumTijd
1	Grijs	11-10-2006 09:56:41
2	Groen	10-10-2006 17:23:22
4	Blauw	10-10-2006 20:19:43
5	Paars	11-10-2006 14:12:18

Afbeelding 1: Toestand 1 en toestand 2.

Stuurtabel

ID	Tabel	DatumTijdVan	DatumTijdTot
1	Voorbeeld	10-10-2006 00:01:00	11-10-2006 00:01:00

Afbeelding 2: Stuurtabel.

beschikbaar is voor dergelijke bewerkingen. Tevens zijn deze technieken over het algemeen nauwelijks geschikt om een (near-) real-time datawarehouse in te richten.

Audit of timestamp-kolommen.
In deze situatie bevat een tabel een of meerdere kolommen die aangeven wanneer een record is toegevoegd en/of gewijzigd, zie ook de voorbeeldtabel in afbeelding 1. Op basis van deze informatie kan bepaald worden welke records sinds een bepaald tijdstip zijn gewijzigd. In veel (standaard-) applicaties zijn dergelijke velden beschikbaar en het lijkt dan ook een eenvoudig en aantrekkelijk alternatief. Toch zitten er nogal wat haken en ogen aan deze aanpak:

- Bij een enkel timestamp-veld is het niet mogelijk om de inserts en updates te onderscheiden;
- Sommige systemen hebben datum en tijd in twee kolommen opgeslagen zodat een simpele vergelijkingsquery wat lastiger te maken is;
- Bijwerken van de timestamps kan in de applicatielaag of in de database met behulp van triggers zijn ingericht. Wanneer dit in de applicatielaag is geregeld is deze methode per definitie niet betrouwbaar: de applicatieprogrammeurs kunnen in verschillende modules verschillende logica hebben toegepast, bij maatwerkmodules kan het bijwerken van de timestamps helemaal vergeten zijn en wanneer batch-bewerkingen direct op de database worden uitgevoerd worden de timestamps ook niet bijgewerkt;
- Delete operaties zijn niet zichtbaar, tenzij records niet fysiek verwijderd worden maar alleen voorzien van een delete indicator (dit is ook weer afhankelijk van de applicatie en/of het datamodel);
- Er is een extra stuurtabel in het datawarehouse nodig om bij te houden welke tabellen tot en met welk tijdstip uitgelezen zijn; dit vereist weer extra stappen in het ETL-ontwerp.

Om ons voorbeeld te laten werken is een extra tabel nodig met de tabelnaam en de datum/tijd van de laatste transactie èn een veld met de bovengrens van het tijdvenster om overlap in resultaatsets te voorkomen, zie afbeelding 2. Met de volgende statements worden de onder- en bovengrens gezet en krijgen we de wijzigingen:

```
Update stuurtabel set DatumTijdVan = DatumTijdTot;
Update stuurtabel set DatumTijdTot = sysdate;

Select * from Voorbeeld
where datumtijd between
(select datumtijdvan from stuurtabel where tabel =
        'Voorbeeld') and (select datumtijdtot from
           stuur-tabel where tabel = 'Voorbeeld');
```

Het is resultaat is te zien in afbeelding 3. Het verwijderde record ontbreekt hier dus, evenals het onderscheid tussen de update en de insert. De meeste ETL-tools kunnen met dit laatste overigens prima uit de voeten doordat eerst een vergelijking met de doeltabel wordt uitgevoerd.

Snapshots.
De term 'snapshot' wordt hier als algemene term gebruikt en niet om een specifieke technologie aan te duiden. Wat bedoeld wordt is dat de inhoud van een tabel op enig moment wordt vergeleken met de inhoud van dezelfde tabel op een ander tijdstip. Het verschil tussen deze twee versies geeft het verschil weer, en omdat dit twee kanten op werkt kunnen met dit mechanisme ook de verwijderde records worden geïdentificeerd. Kimball noemt hierbij eliminatie en initial/incremental loads als twee verschillende oplossingen, maar feitelijk komen ze op hetzelfde neer. Terug naar het voorbeeld: in dit geval bestaan beide toestanden op het moment dat het extractieproces moet starten. Met behulp van een full outer join-operatie kunnen de verschillen inzichtelijk worden gemaakt, en door de query handig op te zetten krijgen we meteen alle insert, update en delete flags mee:

```
select * from
   (select
      case when t2.id is null then 'D'
           when t1.id is null then 'I'
           when t1.tekst <> t2.tekst then 'U'
           else 'N' end as flag,
      case when t2.id is null then t1.id else
                                    t2.id end as id,
      t2.tekst from toestand1 t1 full outer join
                  toestand2 t2 on t1.id = t2.id) a
where flag <> 'N'
```

Het resultaat is te zien in afbeelding 4. Dit lijkt een ideale set om aan een ETL-tool aan te bieden omdat alle benodigde informatie om het datawarehouse correct bij te werken hierin staat. Toch zijn er enkele potentiële nadelen, zoals de benodigde extra opslagruimte en query-tijd bij grote tabellen. Bovendien zijn meervoudige updates bij deze aanpak nog steeds niet zichtbaar.

Resultaat:

ID	Tekst	DatumTijd
1	Grijs	11-10-2006 09:56:41
5	Paars	11-10-2006 14:12:18

Afbeelding 3: Resultaat timestamp selectie.

Database triggers.
Triggers kunnen een timestamp in een tabel bijwerken, maar natuurlijk ook de complete CDC-bewerking voor hun rekening nemen. Met behulp van triggers kunnen alle inserts, updates en deletes worden weggeschreven naar een aparte wijzigingstabel die door een ETL-proces kan worden uitgelezen. Elk record kan dan een timestamp en een operatie-label (I, U, D) meekrijgen zoals bij het snapshot voorbeeld. Bovendien zijn *alle* updates op deze manier beschikbaar, en niet alleen de laatste stand voor de start van een laadproces. Hoewel deze variant een aantrekkelijke optie lijkt kleven er aan deze werkwijze enkele nadelen:

- Performance, elke bewerking krijgt feitelijk een extra na-bewerking in de database, waardoor dit mogelijk een nadelig effect heeft op de performance;
- Garantie/support, indien de database bij een standaardapplicatie hoort, zoals bijvoorbeeld een ERP-pakket, zal de leverancier het niet toestaan (tenzij u het niet erg vindt dat u geen support meer ontvangt) dat er gesleuteld wordt aan de database;
- Onderhoud/beheer, de triggers maken deel uit van de BI-oplossing maar bevinden zich in het operationele systeem. Vaak zijn dit gescheiden werelden waardoor de kans bestaat dat er wijzigingen aan het bronsysteem worden doorgevoerd, waarbij de triggers worden vergeten.

Non-Intrusive technieken

Hoewel momenteel in de meeste gevallen gebruik gemaakt wordt van (een combinatie van) intrusive technieken zijn er diverse trends zichtbaar waardoor dit op termijn geen werkbare oplossing meer biedt. Er is een toenemende vraag naar meer 'operational analytics' die een grotere hoeveelheid data vereisen die ook nog zeer actueel zijn, de batch windows voor het laden van het datawarehouse worden bovendien steeds kleiner (24x7 beschikbaarheid, reconciliatie met andere systemen) terwijl datavolumes groeien.

Gelukkig zijn er alternatieven beschikbaar die steeds bruikbaarder worden. Databases slaan namelijk al automatisch alle wijzigingen op in transactie-logs. Deze logs zijn dus in principe een ideale bron om deze informatie uit te halen. Het feit dat niet op de database wordt ingegrepen maar alleen gecommitteerde transacties uit log-files worden gelezen, maakt deze oplossing ook zeer geschikt voor een zogenaamde 'trickle-feed' van het datawarehouse – waardoor de informatie desnoods near real-time beschikbaar kan komen. Log-files zijn echter database-specifiek en over het algemeen alleen leesbaar door gespecialiseerde tools. In het geval van Oracle kan bijvoorbeeld Oracle Logminer worden gebruikt of Lumigent Log Explorer voor SQL Server databases. Het kunnen lezen van log-files is echter niet voldoende om deze te kunnen gebruiken voor CDC-toepassingen. Er dient ook een mechanisme beschikbaar te zijn waarmee wijzigingen worden doorgesluisd naar een tabel en dat bijhoudt welke records al zijn verwerkt door een volgende stap in het proces. Ook hierin wordt door de standaard databases steeds beter voorzien, maar nog steeds op een database-specifieke manier. Een oplossing zoals bijvoorbeeld Attunity die biedt gaat weer een stapje verder door enerzijds database-specifiek de logs te monitoren en anderzijds de datastroom op een uniforme wijze aan een verwerkingsproces te presenteren.

Resultaat:

Flag	ID	Tekst
U	1	Grijs
D	3	NULL
I	5	Paars

Afbeelding 4: Resultaat full outer join query.

Bij het werken met log-files wordt er over het algemeen gewerkt met een publish/subscribe-model, waarbij vanuit een centrale publisher één of meerdere publicaties bijgewerkt worden. Een publicatie kan hierbij uit meerdere (delen van) tabellen bestaan. Op deze publicaties kan een gebruiker of proces vervolgens een 'abonnement' nemen. Dit mechanisme wordt niet alleen voor datawarehouse-toepassingen gebruikt maar ook voor database-replicatie in het algemeen. Op welke wijze dit in een Oracle database wordt toegepast wordt hierna beschreven.

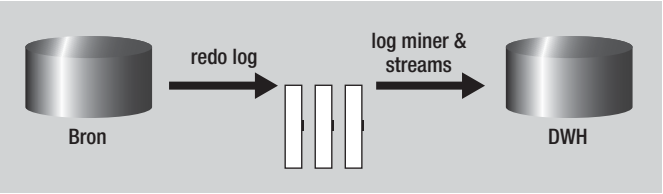
Oracle 10g2 Change Data Capture

Binnen een Oracle 10g2 database kunnen alle hiervoor beschreven technieken worden toegepast. In het kort:

- Timestamps, dit werkt met elk DBMS, dus ook met Oracle;
- Snapshots, hiervoor biedt Oracle met Materialized Views (voorheen snapshot) een standaardoplossing, waarbij met behulp van Materialized View logs zelfs de snapshots automatisch kunnen worden bijgewerkt;
- Triggers, de synchrone CDC-oplossing van Oracle zoals deze vanaf 9i wordt aangeboden is gebaseerd op database triggers. Er zijn diverse tools beschikbaar om het CDC-proces in te richten. Deze worden hier niet verder besproken.

De meest interessante opties zijn echter de asynchrone varianten die allen gebaseerd zijn op de Oracle logs. Oracle kent op dit moment twee vormen in elk twee varianten van asynchrone CDC:

- Hotlog, maakt gebruik van de redo logs waarbij de veranderingen direct na het voltooien van de transactie worden weggeschreven in de change tables. Het datatransport vindt plaats met flat-files;
- Distributed, idem als bovenstaand maar het datatransport gaat via een database link. Bron en doeldatabase kunnen in dit geval verschillend en zelfs van verschillende versies zijn (minimaal 9.2 voor de bron en 10g2 voor het doel, of 10.1 of hoger voor beide);
- Autolog on-line, de gehele redo log-file dient hierbij als invoer voor de staging-area;



Afbeelding 5: Het CDC-databaseproces.

- Autolog archive, veranderingen worden gelezen uit de archive logs. Beide autolog-varianten hebben een batch-georiënteerd karakter en zijn dus niet zo geschikt voor real-time toepassingen.

De achterliggende techniek is ge baseerd op de reeds langer beschikbare Logminer en Streams, maar de implementatie van CDC geschiedt geheel met behulp van PL/SQL. Dit maakt deze oplossing bruikbaar voor iedereen die met PL/SQL overweg kan en maakt tevens de integratie met ETL-tools eenvoudig omdat vrijwel iedere ETL-tool de aanroep van handmatig geschreven PL/SQL mogelijk maakt.

Voor een beter begrip van Oracle CDC zullen allereerst de databaseprocessen worden beschreven en vervolgens de wijze waarop CDC data ophaalt. In afbeelding 5 worden de databaseprocessen van CDC getoond. Alle veranderingen in een Oracle bron-database worden weggeschreven naar de data files en naar redo log-files. De gegevens in een redo log-file zijn opgeslagen in de vorm van transacties. Van de transacties wordt het tijdstip, het type operatie (insert/update/delete) en de gegevens die veranderd zijn (tabellen/kolommen) bijgehouden. Deze transacties worden vervolgens door het log-miner proces doorgezet naar het datawarehouse. Daar zijn deze gegevens weer beschikbaar in de vorm van zogenaamde 'change tables' welke door het ETL-proces als relationele tabellen te benaderen zijn.
Het instellen van CDC geschiedt door het nemen van een abonnement op de wijzigingen van een of meer tabellen. Afbeelding 6 toont de situatie dat een abonnement op de 'EMPLOYEE' tabel is genomen. Het abonnement is beperkt tot de kolommen EMPNO, NAME, SALARY. Veranderingen op deze tabel worden door het capture process uit de redo log-files van de brondatabase gehaald en weggeschreven in een change table. In deze change table vinden we de drie kolommen terug waarop het abonnement is genomen en in de operation-kolom de aanduiding voor de bewerking welke de volgende betekenis hebben:

I – insert;
UO – update before;
UA – update after;
D – delete.

Naast de bewerking kan de volgende informatie worden toegevoegd: de gebruiker die de wijziging heeft gedaan; datum en tijdstip van de wijziging (timestamp); het systeemnummer van de wijziging.

Belasting bronsysteem en performance

De belasting van het bronsysteem is in vergelijking met SQL query's beduidend lager en enigzins lager dan bij het gebruik van snapshots (met gebruik van materialized view logs). Door de asynchrone werking is de impact op batch-processen ook merkbaar lager dan bij het gebruik van snapshots of database triggers. Dit geeft tevens het belang en de motivatie voor het gebruik van deze technieken aan: als een datawarehouse vaker dan eenmaal daags geladen dient te worden of indien het nachtelijke laadproces niet meer past in de beschikbare tijd (backup's, factureringsbatches etcetera), moet wel uitgeweken worden naar een asynchrone variant. De performance van het laadproces zal bij het inzetten van CDC ook bij grote volumes niet omlaag gaan, omdat de techniek achter CDC, Streams, al enkele jaren gebruikt wordt om database te synchroniseren en dus berekend is op grote aantallen wijzigingen.

Een asynchroon CDC-proces inrichten vereist uiteraard de nodige kennis; processen stoppen en starten kan weliswaar gewoon via de Oracle Enterprise Manager, maar voor de overige stappen is de aanwezigheid (en beschikbaarheid!) van een DBA wenselijk.

Subscription

EMPNO	NAME	SALARY	MANAGER
123	SMIT	1000	1000
321	ALLEN	1250	1250
112	JONES	1200	1200
114	SMIT	1100	1100
115	BURLEY	1200	1200
116	ROSA	2000	2000
117	HEIJDEN	2500	2500

Change Table

OPERATION	EMPNO	NAME	SALARY
I	123	SMIT	1000
UO	321	ALLEN	1250
UA	321	ALLEN	1200
I	114	SMIT	1100
I	115	BURLEY	1200
D	222	JEWEL	2000

Afbeelding 6: CDC-dataverwerking.

In veel gevallen ontbreken wizards en tools (nog) zodat er via SQL Plus of TOAD PL/SQL functies moeten worden aangeroepen. Enkele andere belangrijke aandachtspunten voor inrichting en beheer zijn:

- Zorg voor een plan voor het opslaan en verwerken van log-files. Als bijvoorbeeld het DWH enkele dagen niet heeft kunnen laden kan er een behoorlijk beslag op beschikbare schijfruimte worden gelegd;
- Specificeer tablespaces expliciet, anders worden alle objecten in de SYS-tablespace aangemaakt;
- Zorg bij de invoering van CDC voor een goed testplan waarin databaseconfiguratie, redo files en piekbelasting worden getest;
- Zorg voor evenwicht in bron- en doelomgeving. Een OLTP database op een 64 CPU server met enkele duizenden *concurrent users* genereert nu eenmaal meer wijzigingen dan een single CPU PC kan behappen.

Wanneer deze punten in acht worden genomen en er is voldoende DBA-kennis in huis, dan is de asynchrone gedistribueerde hotlog-oplossing de meest geavanceerde oplossing die zelfs (near) real-time datawarehousing mogelijk maakt in een Oracle-omgeving.

Conclusies

Welke van de beschreven technieken is nu het beste? Zoals zo vaak moet het antwoord zijn: 'dat hangt er vanaf'. Om dit wat meer handen en voeten te geven kan de tabel in afbeelding 7 houvast bieden. Dus wanneer deletes en meerdere record-wijzigingen gedurende de dag niet van belang zijn, er geen noodzaak is om vaker dan een keer per dag het datawarehouse bij te werken en niemand last heeft van de nachtelijke query's, dan kan er prima met timestamps worden gewerkt. Wie echter wil anticiperen op een toekomst waarin operationele systemen gebruik maken van data uit het datawarehouse doet er goed aan zich te oriënteren op de in dit artikel beschreven technieken. Met afbeelding 7 kan dus een eerste keuze gemaakt worden, waarna op basis van de tekst de keuze verder ingevuld kan worden.

	Timestamp	Snapshot	Triggers	Logbased
Insert/Update onderscheidbaar?	N	Y	Y	Y
Meervoudige updates zichtbaar?	N	N	Y	Y
Deletes zichtbaar?	N	Y	Y	Y
Lage belasting bronsysteem?	N	N	N	Y
Realtime DWH mogelijk?	N	N	Y	Y
DBMS onafhankelijk?	Y	Y	N	N
Zonder DBA in te richten?	Y	Y	N	N

Afbeelding 7: Selectietabel CDC-opties.

Literatuur

The Data Warehouse ETL Toolkit – Ralph Kimball en Joe Caserta.
Oracle 10g2 datawarehousing guide.
Attunity documentatie.

Jos van Dongen en Cees van Tilburg

Jos van Dongen (jvdongen@tholis.com) is Senior Consultant bij Tholis Consulting. Cees van Tilburg (cees.van.tilburg@oracle.com) is Technical Architect bij Oracle Consulting.

Update

InterSystems brengt CACHÉ 2007 uit

InterSystems heeft een nieuwe versie van CACHÉ uitgebracht. Daaraan zijn nieuwe technologieën en hulpmiddelen toegevoegd, die het ontwikkelen van webapplicaties met 40 procent kunnen versnellen. In de vernieuwde postrelationele database CACHÉ heeft InterSystems Zen geïntegreerd, een raamwerk voor het bouwen van platformonafhankelijke webapplicaties met uitgebreide functies. Sneller dan ooit kunnen deze applicaties nu operationeel gemaakt worden. Een andere nieuwe component, Jalapeño genaamd, ontlast Java-programmeurs van object-relationeel mappen; dit verkort de ontwikkeltijd aanzienlijk. CACHÉ 2007 is verkrijgbaar voor Windows, Linux, Mac, UNIIX en Open VMS.

CBIP-certificaat slaat aan in Nederland

CBIP staat voor Certified Business Intelligence Professional en is een testgebaseerd certificeringsprogramma binnen het BI vakgebied. Er zijn 5 specialisaties mogelijk, op twee niveau's: Practioner en Mastery. Om gecertificeerd te worden dienen drie pittige examens succesvol te worden afgelegd. In het voorjaar van 2006 heeft IT consultant Pecoma haar eerste groep mensen CBIP-gecertificeerd en op dit moment is de tweede groep zich aan het voorbereiden. Daarnaast zijn dit najaar LogicaCMG en een grote Nederlandse bank een opleidingstraject bij CIBIT gestart die afgesloten zal worden met de CBIP-examens.
Na het succesvol afronden van de examens mogen nu ook 7 DWBI professionals van Quintica zich officieel Certified Business Intelligence Professional (CBIP) noemen. Het afgelopen voorjaar heeft Quintica een opleidingstraject gevolgd met haar ervaren DWBI-consultants.
De CBIP-examens worden aangeboden door CIBIT in samenwerking met The Data Warehousing Institute (TDWI) en het onafhankelijke Institute for Certification of Computing Professionals. Sinds 2005 werken CIBIT en TDWI actief samen om de CBIP-certificering de standaard in het vakgebied te laten worden.
CBIP is ontwikkeld en wordt aangeboden in samenwerking met the Institute for Certification of Computing Professionals (ICCP), een non-profit organisatie opgericht in 1973. ICCP is mede verbonden aan onder andere DAMA, IEEE en CIPS.