Datenbankreplikation in einem globalen Lösungsverbund

Stefan Brandl
Wacker Chemie AG
Burghausen

Schlüsselworte
Datenbankreplikation.

Einleitung

In dieser Session wird ein Projekt zur Einführung einer Replikationslösung für Oracle Datenbanken beschrieben.

Gegenstand des Projekts war, die bislang auf einer zentralen Oracle Datenbank basierenden Produktionsdatenerfassung auf mehrere Datenbanken aufzuteilen. Auslöser für das Projekt war die Erweiterung der Produktion auf mehrere weltweit verteilte Standorte und die hohe, unregelmäßige Auslastung der Produktionsdatenbank. Des Weiteren sollte die Verfügbarkeit der einzelnen Produktionsstandorte erhöht werden. Als erster Schritt wurde die Separierung von Auswertebenutzern und Applikationen, die für die Produktion im Betrieb notwendig sind, definiert. Dazu wurde eine eins zu eins Replikation der Produktionsdatenbank in eine neue Reportingdatenbank notwendig. Als weitere Schritte sind die Replikationen zwischen den verschiedenen Produktionsstandorten und die Archivierung von Altdaten geplant. In diesem Projektbericht geht es nur um den ersten Schritt der eins zu eins Replikation in eine Auswertedatenbank.

Doch so einfach sich auch eine eins zu eins Replikation einer Oracle Datenbank anhören mag, der Teufel steckt wie so oft im Detail. Im Verlauf des Projekts waren viele grundlegende Designentscheidungen zu treffen und technische Anforderungen des Produkts zu berücksichtigen. Nach einer Vorstellung der grundsätzlichen Funktionsweise der Replikationslösung werden die einzelnen zu berücksichtigenden Punkte im Detail vorgestellt.

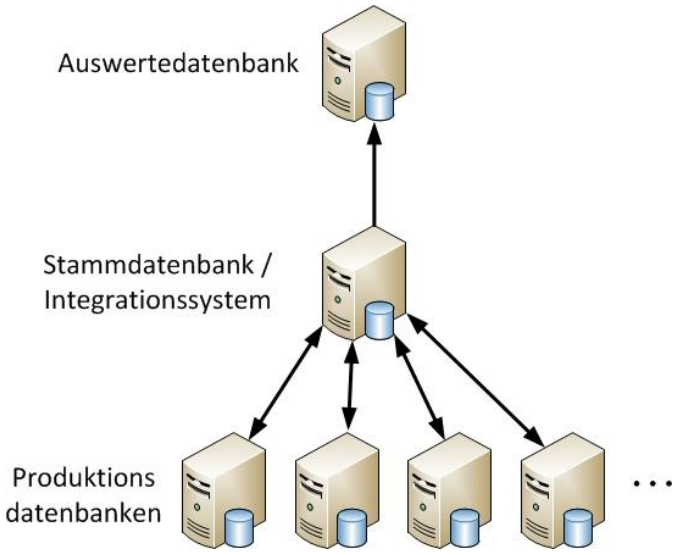

Abb. 1: Gesamtarchitektur

Projektinhalt

Ausgangssituation vor dem Projekt war eine gewachsene IT-Landschaft. Dabei existierte nur eine zentrale Datenbank, die viele Rollen erfüllte. Diese war für die Produktionssteuerung, die Messdatenerfassung und Auswertungen für mehrere Produktionsstandorte vorgesehen. Als weitere neue Produktionsstätten hinzukommen sollten, wurde diese Architektur als nicht mehr tragbar

beurteilt, so dass eine Aufteilung der Datenbanksysteme notwendig wurde. Ziel ist die in Abbildung 1 gezeigte flexibel erweiterbare Architektur. Da hierfür das Einführen einer im Unternehmen neuen Replikationssoftware notwendig wurde, sollte dies zunächst an einem einfachen Anwendungsfall durchgeführt werden. Dies war die Replikation in eine Auswertedatenbank. Dabei sollten Auswertungen unverändert von der Produktionsdatenbank auf die Auswertedatenbank verlagert werden können, so dass die Replikation als eins zu eins Replikation erfolgen musste.

Funktionsweise der Replikation

Die eingesetzte Datenbankreplikation basiert auf dem Lesen der Online Redologdateien der Datenbank. Die darin enthaltenen Informationen werden ausgewertet und falls die Tabellen für die Replikation konfiguriert wurden, werden die Änderungen an Daten auf das Replikationsziel unverändert übertragen. Dadurch können auch am Replikationsziel Constraints aktiviert bleiben, da die Reihenfolge der Statements unverändert bleibt. Am Ziel werden die Daten mit einem gesonderten Datenbank-User für die Replikation mit DML Statements eingefügt. Die Lösung erfordert die Installation von Softwarekomponenten jeweils am Quell- und Zieldatenbankrechner. Diese werden mit Datastore bezeichnet. Jeweils zwei Datastores kommunizieren via TCP/IP und tauschen die zu replizierenden Daten aus.

Dabei ist es grundsätzlich möglich, auch Datenmanipulationen nach fest hinterlegten Regeln in der Replikationsvorschrift zu definieren. In der hier beschriebenen Umsetzung sollte diese Möglichkeit jedoch generell nicht genutzt werden, da auf Basis der replizierten Daten die Dokumentation zur Erfüllung der Spezifikationen von ausgelieferten Produkten erstellt werden. Diese dürfen naturgemäß nicht verändert werden.

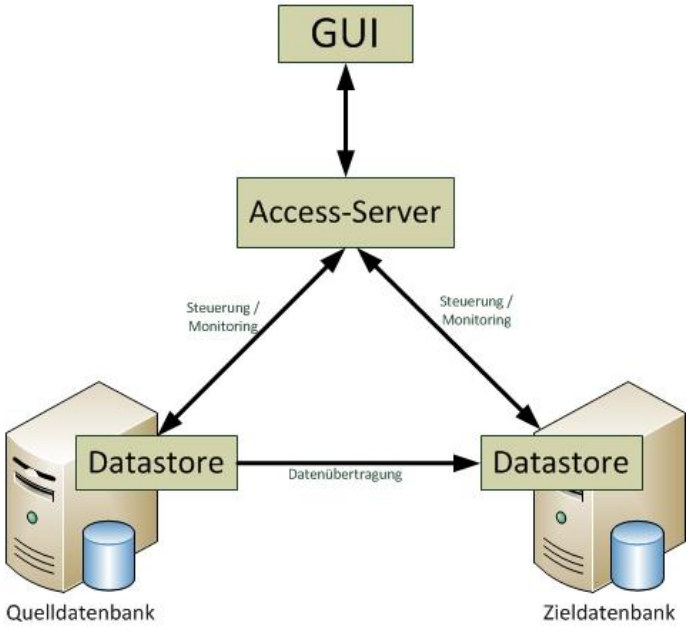

Abb. 2: Basisarchitektur der Replikationslösung

Designentscheidungen

Für den Betrieb der Replikationslösung waren mehrere Änderungen auf der Quelldatenbank durchzuführen.

Zunächst musste für jede zu replizierende Tabelle die Eigenschaft Supplemental Logging aktiviert werden. Dies wurde nicht für die gesamte Datenbank aktiviert, da dadurch das Logaufkommen der Datenbank signifikant ansteigt. In unserem Projekt wurden nur die für Replikation relevanten Tabellen

entsprechend geändert. Dadurch verdoppelte sich das Logaufkommen. Die Replikationslösung bot als Feature an, die entsprechenden Loggruppen anzulegen, jedoch wurden dadurch die Loggruppen nur für alle zum Ausführungszeitpunkt existierenden Spalten der Tabelle angelegt. Falls eine Tabelle um neue Spalten erweitert wird, sind diese Spalten nicht in der Loggruppe enthalten. Somit erkennt die Replikationslösung beim Wiederanstarten die neuen Spalten nicht und beendet sich mit einem entsprechenden Fehler. Um diese potentiellen Fehlerquellen auszuschließen, wurden die Supplemental Log Gruppen immer mit der Option all columns angelegt. Somit werden auch zukünftig neu angelegte Spalten gleich ins Supplemental Logging mit aufgenommen. Dies konnte jedoch nicht durch das Replikationstool abgebildet werden, sondern musste manuell vorgenommen werden. Dazu dient folgendes Statement:

ALTER TABLE SCOTT.EMP ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;

Beim Ausrollen in der Produktion auf allen 800 zu replizierenden Tabellen liefen wir dabei in ein Problem wegen Lock Situationen auf stark frequentierten Tabellen. Dies hatte entsprechende Auswirkungen auf die Applikationen. Im Nachhinein hätten wir dafür eine Wartungsmaßnahme ankündigen müssen, bzw. die Änderung frühzeitig in einem Wartungsfenster ausführen müssen.

Eine weitere Anpassung im Datenbankbetrieb war die Erhöhung der Vorhaltezeit von Redologdateien. Bei einem Neustart der Replikationssoftware nach einem Fehler oder einer Änderung in der Konfiguration muss diese bis zur letzten offenen Transaktion auf der Quelle in den Redologdateien zurück springen und alle Änderungen nachlesen. Vor dem Einsatz der Replikationslösung wurden 2 Tage Redolog Speicherung als ausreichend erachtet. Einige Forms - Applikationen werden jedoch von den Anwendern gerne mehrere Tage offen gelassen. Dadurch entstand nun die Anforderung, dass die Redologs länger gespeichert werden. Dies wurde auf 7 Tage festgelegt. Wären bei einem Neustart der Replikation ältere Transaktionen auf der Datenbank aktiv, müssten die Archivlogs wieder zurückgespielt werden, bevor die Replikation wieder anlaufen kann. Dies würde zu weiteren Wartezeiten bei der Wiederherstellung der Funktionsfähigkeit führen. Um dies zu vermeiden, wurde zusätzlich zur Verlängerung der Vorhaltezeit ein Überwachungsjob eingerichtet, der vor alten Transaktionen warnt. Ab drei Tagen alten Transaktionen wird nun aktiv auf die User zugegangen und gebeten, die Applikationen zu schließen. Meist hat man dann auch nur vergessen, das Fenster zu schließen….

Ein weiterer zu beachtender Aspekt ist, wie mit DDL Änderungen umgegangen werden muss. Bei Änderungen (neue Spalten, gelöschte Spalten, umbenannte oder erweiterte Spalten) beendet sich die Replikation mit einem entsprechenden Fehler. Die Änderungen müssen in der Replikation bekannt gegeben werden. Dabei gäbe es generell zwei Möglichkeiten, dies zu tun: Vor der Ausbringung der Änderung die Replikation stoppen oder die Replikation auf einen Fehler laufen lassen. Im Projekt wurde letzteres als Lösung empfohlen. Hintergrund ist, dass dadurch bis zum letzten DML Statement alle Daten repliziert werden, erst dann wird die DDL Änderung im Log gelesen. Somit kann beim Start der Replikation das jeweils aktuell gültige DDL benutzt werden. Würde man die Replikation vorher stoppen, könnte es sein, dass zwischen dem Stoppen der Replikation und der Ausbringung des DDL Statements noch Datensätze in die relevante Tabelle geschrieben würden. Diese wären durch das Supplemental Logging mit der alten DDL Definition im Redolog enthalten. Somit würde bei Wiederanstarten der Replikation die Beschreibung der Tabelle innerhalb der Replikation (neues DLL) und die Beschreibung im Redolog (altes DDL) nicht zusammenpassen. Die Replikation könnte somit nicht gestartet werden. Für diesen Fall gäbe es zwar die Möglichkeit, Fehler zu ignorieren, und die Replikation trotzdem weiterlaufen zu lassen. Dies widerstrebt jedoch einem Datenbankanwender, da die Konsistenz der Daten dadurch nicht mehr gewährleistet ist. Im Change Management ist nun deshalb definiert, dass bei DDL Änderungen die Replikation auf einen Fehler laufen soll.

Eine weitere Besonderheit im Replikationsprojekt war die Berücksichtigung der Kombination mit einer geplanten Archivierung der Produktionsdatenbank. Dabei ist geplant, aus der Quelldatenbank alle Daten, die älter als eine noch zu definierende Grenze sind, zu löschen. Die Replikationslösung speichert am Ziel jeweils die zuletzt commitete SCN ab, um bei Neustarts oder nach Fehlern auf der

exakten Logposition wieder aufsetzen zu können. Bei Änderungen an den Metadaten der Tabelle in der Replikationslösung wird diese Information jedoch verworfen. Diese muss dann vom Ziel der Replikation wieder übernommen werden, um einen korrekten Aufsatzpunkt beim Wiederanstarten der Replikation zu finden. Ohne diesen Aufsatzpunkt würden relevante Sätze verpasst werden. Dieses Setzen der SCN wird mit Bookmark setzen bezeichnet und ist nur über ein Kommandozeilentool möglich. Dazu muss man sich direkt auf den Datenbankrechner verbinden. Dies ist bei Wacker Chemie jedoch strikt getrennt und Datenbankentwickler haben keinen Zugriff auf die Datenbankmaschinen. Somit benötigt man für DDL Änderungen zukünftig Mitarbeiter aus zwei verschiedenen Abteilungen, wodurch die Flexibilität weiter eingeschränkt wurde. Alternative wäre bei DDL Änderungen von Tabellen, diese komplett neu zu laden (refresh) und somit den Aufsatzpunkt wieder zu erreichen. Durch das oben erwähnte Archivieren kann dies aber nicht durchgeführt werden, da die Historie nur noch in der Auswertedatenbank vorhanden ist.
Für die Anwender wichtig waren auch die Latenzzeiten, d.h. die Verzögerung mit der durchschnittlich neue bzw. geänderte Datensätze in der Auswertedatenbank zur Verfügung stehen. Auf einigen Tabellen werden sekündlich Datensätze eingefügt. Im Normalbetrieb der Replikation ist eine Verzögerung von wenigen Sekunden zu beobachten. Problem dabei ist jedoch, welches SLA mit dem Datenowner abgesprochen wird. Im Fehlerfall oder bei einem Change kann es durchaus sein, dass die Replikation mehrere Minuten bis Stunden hinterherhinkt. Vor der Datentrennung waren die Daten immer hochaktuell. Für die neue Architektur wurde eine Meldezeit von zwei Stunden definiert. Das heißt, sobald die Replikation um mehr als zwei Stunden hinterherhinkt, müssen die Anwender informiert werden. Dies war seit der Produktivsetzung Ende 2011 bislang erst einmal der Fall.
Besonders beachtet werden muss auch ein mögliches Recovery einer der beiden beteiligten Datenbanken. Falls das Ziel recovered werden muss, ist es ausreichend, die Replikation einfach wieder anzustarten, da die zu verwendende Logposition für den Wiederanlauf durch das Recovery des Ziels korrekt wiederhergestellt wird. Komplizierter stellt sich der Fall dar, wenn die Quelle recovered werden muss. Dann muss nämlich auch das Ziel auf einen älteren Stand als die Quelle wiederhergestellt werden. Hintergrund ist, dass in der im Projekt verwendeten Einstellung der Replikation alle DML Statements unverändert übertragen werden. D.h. insert bleibt insert und update bleibt update. Bei einem Fehler, bspw. Constraintfehler bei wiederholten inserts oder versuchten updates bei nicht vorhandenen Datensätzen im Ziel würde die Replikation fehlerhaft beendet. Wenn die Zieldatenbank zeitlich gesehen hinter das Quellsystem zurückgesetzt wird, können die Unterschiede sauber aufgeholt werden.
Dies sind nur einige Beispiele, wie die Replikation Einfluß auf das Datenbanksystem bzw. dessen Einstellung hat. Weitere Punkte, die beachtet werden müssen sind:

- Kompletter Refresh von Tabellen in Zusammenhang mit der Archivierung des Quellsystems.
- Reihenfolge von Löschstatements bei selbstrekursiven Tabellen.
- Aktualisierung von Materialized Views.
- Verwendung von Datenbanklinks.

Diese Punkte werden in den Vortragsfolien kurz gezeigt.

Fazit und Ausblick

Das zunächst als Kennenlernen für die neue Replikationssoftware gedachte Projekt zum Aufbau einer Auswertedatenbank hatte also in der Durchführung doch einige Klippen zu bieten, die umschifft werden wollten. Diese konnten jedoch alle in der geplanten Durchführungszeit von sechs Wochen bewältigt werden. Die eingangs erwähnten Projektziele und Erwartungen haben sich insgesamt gesehen erfüllt. Als weitere Schritte zur Gesamtarchitektur aus Abbildung 1 werden aktuell Replikationen zwischen verschiedenen Produktionsnahen Datenbanken aufgebaut. Hier können wir von den erworbenen Kenntnissen profitieren.

Kontaktadresse:
Stefan Brandl
Wacker Chemie AG
Johannes-Hess-Straße 24
D-84489 Burghausen

Telefon: +49 (0) 8677-83-0
Fax:
E-Mail stefan.brandl@wacker.com
Internet: www.wacker.com