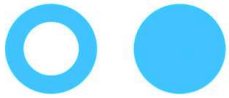

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Verteilte Datenbanken

- Übung 4 -
Schlüsselintegrität

Harm Knolle

- **Abgabe Übung 4 bis Sonntag, 17. Mai 2015, 24:00 Uhr**

DBV_Übung_4 vom 10.05.2015 23:01

Druck vom 10.05.2015 23:17

Prof. Dr. H. Knolle
Hochschule Bonn-Rhein-Sieg
Fachbereich Informatik
Grantham-Allee 20
53757 Sankt Augustin

Inhaltsverzeichnis

1 Vorüberlegungen 3

2 Realisierung der lokalen Schlüsselintegrität 4

2.1 Sequenzen 4
2.2 Trigger 4

3 Realisierung der globalen Schlüsselintegrität 7

4 Vorstellung der Ergebnisse 9

1 Vorüberlegungen

Die Datenbanken „Bonn“, „London“ und „New York“ sind im jetzigen Zustand lediglich für sich selbst verantwortlich. Jede der Datenbanken prüft daher lediglich die für ihre Daten vereinbarten Integritäts- und Konsistenzbedingungen. In den folgenden Übungen sollen Maßnahmen zur Sicherung der systemübergreifenden Konsistenz und Integrität getroffen werden. Hierbei stehen u.a. die folgenden Fragen und Problemlösungen im Vordergrund:

a) Welche Probleme müssen im Rahmen der globalen Primärschlüssel-Constraints gelöst werden?

    a. Wie gestalten sich Einfügevorgänge?

    b. Wie gestalten sich Löschvorgänge?

    c. Was muss bei Änderungsvorgängen beachtet werden?

b) Welche Probleme müssen im Rahmen der globalen Fremdschlüssel-Constraints gelöst werden?

    a. Wie gestalten sich Einfügevorgänge?

    b. Wie gestalten sich Löschvorgänge?

    c. Was muss bei Änderungsvorgängen beachtet werden?

c) Erörtern Sie zunächst ein grundsätzliches Lösungskonzept, das eine widerspruchsfreie knotenübergreifende Primärschlüsselvergabe ermöglicht. Beachten Sie, dass die Primärschlüsselvergabe für den Endanwender transparent geschehen soll.

2 Realisierung der lokalen Schlüsselintegrität

Häufig werden neue Primärschlüsselwerte über die SQL-Anweisung „SELECT MAX(<spalte> + 1) FROM <tabelle> berechnet. Diese Vorgehensweise garantiert die Berechnung des nächst höheren Spaltenwertes (sofern die Operation „+“ angewendet werden kann). Es kann aber nicht ausgeschlossen werden, dass zwei parallele Transaktionen dieselbe Berechnung zur gleichen Zeit ausführen. Die Folge ist, dass die schnellere Transaktion den neu berechneten Primärschlüsselwert zur Einfügung einer neuen Zeile verwenden kann, die langsamere dagegen auf eine Verletzung der Schlüsselbedingung stößt.

Zur Vertiefung und Anwendung der im Folgenden vorgeschlagenen Konzepte lesen Sie bitte in der Oracle-Dokumentation nach (insbesondere SQL Reference Manual: http://docs.oracle.com/database/121/SQLRF/toc.htm).

2.1 Sequenzen

Das Datenbanksystem Oracle besitzt ein Konzept, das sich zur Spezifikation und zur Verwaltung von anwendungsspezifischen Zählern eignet, die so genannte Sequenz: „`CREATE SEQUENCE ...`“. Sequenzen eigenen sich hervorragend zur Berechnung künstlicher (Primär)Schlüsselwerte. Sie repräsentieren Folgen numerischer Werte, die durch „`<sequenz>.NEXTVAL`“ oder „`<sequenz>.CURRVAL`“ angesprochen wird. Hierbei inkrementiert oder dekrementiert NEXTVAL die Sequenz um den bei der Erzeugung angegebenen Schrittwert.

Die Sequenzen werden automatisch verwaltet. Ihre Werte sind jedoch noch völlig unabhängig von den Tabellen, für die sie (Primärschlüssel)Werte generieren sollen. Die generierten Werte müssen daher wie gewohnt in die entsprechenden Tabellen eingetragen werden. Um auch diesen Prozess zu automatisieren, bieten sich so genannte Trigger an.

a) Definieren Sie spezifische Sequenzen pro Standort und pro lokaler Tabelle. Nutzen Sie hierfür das Konzept „`CREATE SEQUENCE ...` “

b) Nutzen Sie die von Ihnen bereits verwendeten lokalen Prefixe im Rahmen der Namenskonventionen.

c) Welche Parameter (siehe Dokumentation) werden benötigt und welche können weggelassen werden?

2.2 Trigger

Trigger werden in Oracle mit Hilfe der prozeduralen Erweiterung von SQL (PL/SQL) spezifiziert und mit den entsprechenden Tabellen der Datenbank verknüpft. Im Gegensatz zu normalen Anwendungen, die ein Anwender bzw. ein Programm explizit starten muss, werden Trigger beim Vorliegen der spezifizierten Ereignisse vom DBMS automatisch „gezündet“. Für

Tabellen sind dieses die Ereignisse INSERT, DELETE und UPDATE. Dabei wird weiterhin unterschieden zwischen BEFORE- und AFTER-Triggern. BEFORE-Trigger werden gezündet, bevor die Datenbank die eigentliche Datenänderung vorgenommen hat, AFTER-Trigger zünden unmittelbar im Anschluss an das Ereignis. Bei Triggern können die jeweiligen neuen Attributwerte mit „`:NEW.<attribut>`“ angesprochen werden. Bei AFTER-Trigger stehen die überschriebenen Werte zusätzlich mit „`:OLD.<attribut>`“ für Vergleiche zur Verfügung.

Beispiel für die Programmierung eines Triggers:

```
CREATE OR REPLACE TRIGGER bnn_trg_article
    BEFORE INSERT ON bnn_article
  FOR EACH ROW
  WHEN (NEW.ID_article IS NULL)
  BEGIN
    SELECT bnn_s_article.NEXTVAL
              INTO :NEW.ID_article
              FROM dual;
  EXCEPTION
    WHEN OTHERS THEN
      NULL;
  END;
/
```

Dieser Trigger „`BNN_trg_article`“ wird „vor“ jedem Einfügevorgang in die Tabelle „`article`“ gezündet. Es wird geprüft, ob ein Primärschlüsselwert für die Spalte „`ID_article`“ vorhanden ist. Ist der vorhandene Primärschlüsselwert „`NULL`“, so wird in die neu einzutragende Zeile der nächste Wert der Sequenz „`bnn_s_article`“ eingetragen. Die Tabelle „`dual`“ ist hierbei eine Dummy-Tabelle, von der man grundsätzlich immer selektieren kann. Im EXCEPTION-Teil kann auf diverse Fehlerfälle beim Ablauf des Triggers reagiert werden. In diesem Beispiel wird zwar jeder Fehler abgefangen, es wird jedoch nichts unternommen.

Bitte beachten Sie den „/“ am Anfang der Zeile am Ende einer Trigger-Definition. Es ist ein Steuerzeichen das signalisiert, dass der Befehl abgeschlossen ist und vom Datenbanksystem interpretiert und kompiliert werden kann. Normalerweise geschieht das bei SQL schon bei einem „;“. Prozedurale SQL-Befehle wie Trigger und gespeicherte Prozeduren können jedoch mehrere Einzelbefehle enthalten, deren Interpretation alleine keinen Sinn ergibt. Wichtig: das „/“ muss am Anfang einer Zeile stehen.

a) Definieren Sie spezifische Trigger pro Standort und pro lokale Tabelle. Nutzen Sie dabei das Konzept „`CREATE TRIGGR ...`“, das im Folgenden die Vergabe der lokalen Primärschlüsselwerte automatisieren soll.

b) Nutzen Sie hierbei die von Ihnen bereits verwendeten lokalen Prefixe im Rahmen der Namenskonventionen.

c) Nutzen Sie für die Schlüsselvergabe die zugehörigen oben erstellten Sequenzen.

d) Innerhalb des Oracle-Kommandointerpreters können Sie mögliche Fehler mit dem Kommando „show error“ anzeigen lassen.

e) Testen Sie den Trigger interaktiv und schrittweise durch das gezielte Einfügen einzelner Datensätze direkt in Ihre lokalen Tabellen.

a. Warum wird es vereinzelt Fehlermeldungen der Form „ORA-00001: unique constraint (<name>.<primary key constraint name>) violated“ geben (nicht zwingend)?

b. Passen Sie die Sequenz bzw. den Trigger an und testen Sie den Trigger erneut.

c. Wählen Sie geeignete Start-Werte für die Sequenzen.

f) Warum ist es sinnvoller, die „`WHEN`“-Bedingung im Trigger ganz zu streichen?

g) Wie verhält es sich mit Primärschlüsseln, deren Werte aus Zeichen bestehen?

h) Testen Sie Ihre Primärschlüsselvergabe anhand von Einfügevorgängen in Ihre globalen Views. Welche Probleme sind zu erwarten?

3 Realisierung der globalen Schlüsselintegrität

Zur Realisierung der physikalischen Unabhängigkeit darf der globale Einfügevorgang nicht lokal erfolgen, sondern vollständig ortstransparent und somit global. Das bedeutet, dass der Einfügevorgang in die globalen Sichten erfolgen muss. Dieses wird jedoch nicht funktionieren, da die Sichten nicht wissen, an welche ihrer zugrunde liegenden Tabellen sie den Datensatz weiterleiten sollen (Frage: Warum können die Sichten das nicht wissen?).

Abhilfe schaffen an dieser Stelle sogenannte Instead-Trigger, die den Einfügeversuch in eine Sicht bereits im Vorfeld der Ausführung abfangen und somit eine Fehlermeldung durch das System verhindern. Die so abgefangene Anweisung kann dann vom Aktionsteil des Triggers entsprechend programmiert werden, sodass der Trigger den Datensatz anhand des Fragmentierungsprädikat letztendlich korrekt an das Zielsystem weiterleitet.

Zur Vertiefung und Anwendung der im Folgenden vorgeschlagenen Konzepte lesen Sie bitte in der Oracle-Dokumentation nach (insbesondere SQL Reference Manual: http://docs.oracle.com/database/121/SQLRF/toc.htm).

Beispiel für die Programmierung eines Instead-Triggers:

```
CREATE OR REPLACE TRIGGER trg_<name>
    INSTEAD OF INSERT OR DELETE ON <global_view>
  FOR EACH ROW
  BEGIN
    IF INSERTING THEN
      IF <fragmentation1> THEN
        INSERT INTO <local_table1> values (NULL,
               :new.<attribut1>, :new.<attribut2>, ...);
      ELSIF <fragmentation2> THEN
        INSERT INTO <local_table2> values (NULL,
               :new.<attribut1>, :new.<attribut2>, ...);
      ELSIF <fragmentation3> THEN
        INSERT INTO <local_table3> values (NULL,
               :new.<attribut1>, :new.<attribut2>, ...);
      END IF;
    END IF;
    IF DELETING THEN
      IF <fragmentation1> THEN
        DELETE FROM <local_table1> WHERE <key_attribute> =
                  :old.<key_attribute>;
      elsif <fragmentation2> THEN
        DELETE FROM <local_table2> WHERE <key_attribute> =
                  :old.<key_attribute>;
      elsif <fragmentation3> THEN
        DELETE FROM <local_table3> WHERE <key_attribute> =
                  :old.<key_attribute>;
      END IF;
    END IF;
  END;
/
```

a) Erstellen Sie pro globaler View einen Trigger, der die Zuordnung der Einfügevorgänge in die jeweils zuständige lokale Tabelle vornimmt. Berücksichtigen (implementieren) Sie hierbei die zuvor von Ihnen festgelegten Fragmentierungskriterien.

b) Nutzen Sie hierbei die von Ihnen bereits verwendeten lokalen Prefixe im Rahmen der Namenskonventionen.

c) Wie müssen die globalen Trigger implementiert werden, damit die lokalen Trigger für die Primärschlüsselvergabe nicht übergangen werden?

d) Was passiert, wenn Sie einen Einfügevorgang in eine durch ein Synonym referenzierte Sicht vornehmen?

   a. Erörtern Sie in diesem Zusammenhang erneut die Problematik „zentralisierte und replizierte Katalogverwaltung“.

   b. Begründen Sie die von Ihnen favorisierte Vorgehensweise im Rahmen der Programmierung der globalen Trigger.

e) Berücksichtigen Sie gleichzeitig auch die Löschvorgänge in den globalen Views.

f) Testen Sie die Verteilung und die Primärschlüsselvergabe interaktiv und schrittweise

   a. durch das gezielte Einfügen einzelner Datensätze direkt in Ihre globalen Views.

   b. Kontrollieren Sie den globalen Einfügevorgang anhand der Daten Ihrer lokalen Tabellen.

g) Testen Sie den Löschvorgang interaktiv und schrittweise

   a. durch das gezielte Löschen einzelner Datensätze direkt aus Ihren globalen Views.

   b. Kontrollieren Sie den globalen Löschvorgang anhand der Daten Ihrer lokalen Tabellen.

4 Vorstellung der Ergebnisse

Im Rahmen der Vorstellung Ihrer Ergebnisse werden die folgenden Unterlagen **pro Gruppenmitglied** erwartet (fristgerechter Upload auf die Lernplattform):

a) Erstellung (unter Ihrer persönlichen Benutzerkennung):

   a. Lokale Sequenzen zur Spezifikation der Primärschlüsselwerte (siehe Kapitel 2).

   b. Lokale Trigger zur Generierung der Primärschlüsselwerte (siehe Kapitel 2).

   c. Globale Trigger zur Steuerung der globalen Einfüge und Löschaktionen (siehe Kapitel 3).

   d. Pro lokaler Tabelle Generierung von Testdatensätzen zum Testen der globalen Einfügung und entsprechender anschließender globaler Löschung.

b) Upload folgender Dateien (nutzen Sie die Vorlagen die in moodle zur Verfügung gestellt werden):

   a. Die eigenen drei Protokolle der Installation „`[bonn|london|newyork]_global_pk_database.log`“ (**nicht ein Protokoll pro Gruppe, sondern die individuellen Protokolle der einzelnen Gruppenmitglieder**).

   b. Testskript „`test_global_pk_database.sql`“ (keine Logdatei).

c) Vorstellung Ihrer Datenbank mit Hilfe des Oracle SQL Developers.

d) Beantwortung der im Text „eingebauten“ Fragen (mündlich).