Universität Augsburg, Institut für Informatik
Datenbanken und Informationssysteme
Priv.-Doz. Dr. M. Endres

Datenbankprogrammierung (Oracle)

Wichtiger Hinweis: *Um nachfolgende Übungen sinnvoll durchführen zu können, hat der User/Schema, der unter Ihrem Namen angelegt wurde weitreichende Rechte erhalten (neue Benutzer anlegen, löschen, etc.), die es Ihnen ermöglichen aus (entweder aus Unachtsamkeit oder aus Absicht) verschiedene Formen von Vandalismus auf unserem Server zu veranstalten. Wir bitten Sie dringend, von derartigem abzusehen und den Anweisungen des Übungsblattes und der Tutoren genau zu folgen, da davon auch Ihre Mitstudenten, Examensarbeiter und Nachfolger betroffen sind. Oder wie es so schön im Kino heißt: "with great power comes great responsibility".*

Aufgabe 1: Benutzer

Legen Sie auf der Oracle Datenbank einen Benutzer `freund`<ihr Benutzername> an (also z.B. `freund0815`). Verwenden sie als Default-Tablespace den Tablespace `ORACLE_STUDENTS`!

- Melden Sie sich als dieser Benutzer neu bei der Datenbank an.
- Was geschieht? Warum? Was ist dagegen zu tun?
- Löschen Sie den User `freund`<ihr Benutzername> wieder.

Aufgabe 2: Rollen

In dieser Aufgaben sollen Sie sich mit Rollen vertraut machen.

- Erstellen Sie eine Rolle mit dem Namen `my<IHR BENUTZERNAME>VIEW` (z.B. `my0815VIEW`).
- Die Rolle soll alle Ihre Objekte sehen, aber nicht verändern können (Falls sich in Ihrem Schema noch keine Objekte befinden, legen Sie einige Tabellen an).
- Erstellen Sie zum Testen Ihrer Rolle einen Benutzer "`my<IHR BENUTZERNAME>DUMMY`" und weisen Sie ihm die Rolle `my<IHR BENUTZERNAME>VIEW` zu. Verwenden sie als Default-Tablespace den Tablespace `ORACLE_STUDENTS`!
- Wählen Sie als Benutzer `my<IHR BENUTZERNAME>DUMMY` die Rolle `my<IHR BENUTZERNAME>VIEW` ab und prüfen Sie die Zugriffsmöglichkeiten.
- Sichern Sie die Rolle `my<IHR BENUTZERNAME>VIEW` mit einem Passwort ab.
- Löschen Sie die Rolle `my<IHR BENUTZERNAME>VIEW` und den User `my<IHR BENUTZERNAME>DUMMY`.

Aufgabe 3: Materialized Views

Bei Views, die auf komplexen Abfragen definiert sind, kann es sinnvoll sein, die View-Daten nicht bei jedem Aufruf neu generieren zu müssen, sondern sie zu *materialisieren*, d. h. als Kopie abzuspeichern und nur zu bestimmten Zeitpunkten zu aktualisieren.

Oracle bietet hierfür das Konstrukt der *materialisierten Views*. Für materialisierte Views gibt es eine Trennung von Abfragen: Einfache Abragen selektieren nur Zeilen aus einer einzigen Tabelle (keinem View!) und führen keine Mengenoperationen, Joins oder Gruppierungen aus. Jede Ergebniszeile einer einfachen Abfrage entspricht einer Zeile der Ursprungstabelle. Bei komplexen Abfragen ist das nicht der Fall.

Die Syntax zur Erstellung eines materialisierten Views lautet:

```
CREATE MATERIALIZED VIEW <name>
[ REFRESH [ FAST | COMPLETE | FORCE ]
[ ON {DEMAND | COMMIT} ]
[ START WITH <startdatum> ] [ NEXT <datum> ]
[ WITH {PRIMARY KEY | ROWID} ] | NEVER REFRESH ]
[ FOR UPDATE ]
AS
SELECT ...
```

- Die `Refresh`-Option legt fest, wann und wie Oracle die Daten des materialisierten Views aktualisiert.
  - `FAST`, nur für einfache Views. Es werden nur die veränderten/neuen Zeilen der Mastertabelle übertragen. Um diese Methode verwenden zu können, muss zusätzlich ein View Log auf die Mastertabelle erzeugt werden: **`CREATE MATERIALIZED VIEW LOG ON`** `<mastertabelle>;`
  - `COMPLETE`: Der View wird komplett neu aufgebaut.
  - `FORCE`: veranlasst Oracle, zunächst einen FAST-Refresh zu versuchen. Ist das nicht möglich, wird der View komplett neu aufgebaut.
- Der Zeitpunkt von Aktualisierungen wird festgelegt mittels
  - `ON COMMIT`: Ein Update wird bei einem COMMIT auf die Mastertabelle(n) durchgeführt.
  - `ON DEMAND`: Ein Update erfolgt erst bei Eingabe eines Refresh-Befehls. Dies kann (unter anderem) durch den Aufruf des Befehls

    `EXECUTE` `DBMS_MVIEW.`**`REFRESH`**`('<Refresh-Group-Name>');`

    erfolgen. Mit diesem Befehl können auch materialisierte Views aktualisiert werden, für die eine andere Refresh-Methode angegeben wurde.
- `START WITH ... NEXT ...`: Ein Intervall, das zeitgesteuerte Aktualisierungen ermöglicht, , in dem mit `START WITH` ein Zeitwert und mit `NEXT` als darauffolgendem Zeitwert der Startzeitpunkt und die Intervalllänge angegeben werden. Ein täglicher Refresh um 03:00 Uhr morgens, beginnend am nächsten Tag, lässt sich wie folgt realisieren:

  `START WITH` `TRUNC(sysdate) + 3/24` **`NEXT`** `trunc(sysdate + 1) + 3/24`

  `TRUNC` entfernt dabei die Stundenzahl des aktuellen Zeitwerts, den `SYSDATE` liefert. Danach kann die Zeit in Tagen hinzuaddiert werden, wobei 3 genau 03:00 Uhr morgens ist. ,ll
- `NEVER REFRESH`: Es wird nie eine Aktualisierung durchgeführt.
- Über `WITH PRIMARY KEY | ROWID` wird der Bezug zur Mastertabelle hergestellt.
- `FOR UPDATE` gibt an, dass der materialisierte View nicht schreibgeschützt ist.

Bearbeiten Sie nun folgende **Aufgabe:** Gegeben sei die Relation *katalog(id, name, beschreibung, ek_preis, vk_preis)*. *ek_preis* ist der *Einkaufspreis*, *vk_preis* der *Verkaufspreis*. Erstellen Sie die Relation in der Datenbank und füllen Sie sie mit ein paar Beispieldaten.

Für Geschäftskunden soll ein materialisierter View erstellt werden. Der Einkaufspreis soll ausgeblendet und der Netto-Verkaufspreis ohne Mehrwertsteuer angezeigt werden. Im View sollen keine Veränderungen an Daten ausgeführt werden können. Ein (möglichst resourcensparendes) automatisches Update soll täglich um 2:00 Uhr morgens ausgeführt werden.

Aufgabe 4: SQL (Standard SQL)

In der Oracle-Datenbank finden Sie die beiden folgenden Tabellen eines Gebrauchtwagenhändlers:

- used_cars_in_stock (id, make, price, color, age)
- used_cars_sold (id, make, price, color, age)

Die erste Tabelle enthält Informationen über im Lager vorhandene Autos, die zweite über verkaufte Autos. Bearbeiten Sie die folgenden Aufgaben:

a) Geben Sie eine Liste aller Fahrzeuge in der Datenbank aus. Für jedes Auto soll in einer zusätzlichen Spalte `status` angegeben werden, ob es im Lager ("`in stock`") oder bereits verkauft ("`sold`") ist.

b) Autos mancher Marken verkaufen sich in bestimmten Farben nur sehr schlecht oder gar nicht. Geben Sie die Autos aus dem Lager aus, für deren Kombination aus Marke und Farbe es keine Verkäufe gibt.

c) Geben Sie das billigste Fahrzeug aus, das bereits verkauft wurde. Benutzen Sie dazu keine Aggregationsfunktion!

d) Geben Sie das teuerste Fahrzeug aus, das entweder im Lager ist oder bereits verkauft wurde. Benutzen Sie auch hier keine Aggregationsfunktion!

e) Ermitteln Sie diejenigen bereits verkauften Fahrzeuge, bei denen sowohl Preis als auch Alter höher waren als bei anderen verkauften Fahrzeugen mit gleichem Hersteller und gleicher Farbe. Ordnen Sie die Fahrzeuge nach Hersteller und Farbe.