Databázové systémy

08. pohľady, synonymá, PL/SQL

KKUI
Katedra kybernetiky
a umelej inteligencie

Pohľady

- logická reprezentácia tabuľky alebo kombinácie viacerých tabuliek resp. pohľadov
- virtuálna tabuľka definovaná dopytom
- vždy „obsahuje“ aktuálne údaje
- neobsahuje údaje iba svoju definíciu t.j. spôsob ako prislúchajúce údaje získať
- nezaberá miesto
- umožňujú prispôsobiť prezentáciu údajov rôznym typom používateľov

Pohľady - vytvorenie

Syntax:

```
CREATE VIEW <názov pohľadu> AS
  SELECT ...;
```

- pohľad sa dá vytvoriť z každého dopytu

```
CREATE VIEW emp_view AS
  SELECT last_name, salary*12 annual_salary
  FROM employees
  WHERE department_id = 20;
```

Pohľady - vytvorenie

```
CREATE OR REPLACE VIEW v_emp_clerk AS
  SELECT first_name, department_id, salary,
  job_id
  FROM employees
  WHERE job_id like '%CLERK'
  WITH CHECK OPTION
  CONSTRAINT v_emp_con;
```

- CHECK OPTION zabráni vkladaniu, aktualizácii dát, ktorých výsledkom by boli riadky, ktoré by už neboli súčasťou daného pohľadu

Pohľady - vytvorenie

```
CREATE VIEW customer_ro(name,language,
  credit) AS
  SELECT cust_last_name, nls_language,
  credit_limit FROM customers
  WITH READ ONLY;
```

Pohľady - vlastnosti

- pohľad nezaberá takmer žiadne miesto na disku
- spojenie viacerých tabuliek do jednej
- obmedzenie prístupu k dátam (podmnožina údajov z jednej resp. viacerých tabuliek)
- ochrana pred prepísaním dát
- väčšia záťaž na procesor
- zobrazenie dát z pohľadu môže byť časovo náročnejšie ako pri priamom prístupe

Pohľady - vlastnosti

- štandardne ich nejde modifikovať, sú určené na zobrazenie dát
- obmedzené možnosti modifikovania:
  - nesmie obsahovať agregačné funkcie, množinové operácie, GROUP BY ...(viď. dokumentácia)

- ak nie je možné modifikovať pohľad klasickým spôsobom, môžme využiť triggre

Pohľady – mazanie a úprava

- odstránenie:

```
DROP VIEW <názov pohľadu>;
```

- zmena definície pohľadu možná iba nahradením pôvodného pohľadu:
  - pomocou príkazu `CREATE OR REPLACE`

- `ALTER VIEW <názov pohľadu> RECOMPILE;`

Materializované pohľady

- databázový objekt obsahujúci výsledok dopytu
- narozdiel od klasického pohľadu obsahuje dáta, ktoré boli výsledkom pri vytvorení dopytu
- je možné ho aktualizovať(urobiť refresh)
- defaultne nie je možné ho upravovať
- na prácu s mat. pohľadom slúži package dbms_mview

Príklad:

```
CREATE MATERIALIZED VIEW CAL_MONTH_SALES_MV AS
   SELECT   t.calendar_month_desc,  sum(s.amount_sold) AS
   dollars   FROM     sales s, times t
   WHERE    s.time_id = t.time_id
   GROUP BY t.calendar_month_desc;
```

Synonymá

- alias pre databázový objekt

Syntax:

```
CREATE [PUBLIC] SYNONYM <názov synonyma> FOR
  <meno užívateľa>.<názoy objektu>;

DROP [PUBLIC] SYNONYM <názov synonyma>;
```

Príklad:

```
CREATE OR REPLACE SYNONYM countries FOR
  hr.countries;
```

PL/SQL

- procedurálne rozšírenie jazyka SQL
- definovanie a spúšťanie programových jednotiek
- nie je case-sensitive
- Programové jednotky:
  - Procedúry
  - Funkcie
  - Balíky (Package)
  - Spúšťače (Trigger)
  - Natívne storované JAVA objekty

PL/SQL blok

```
[Záhlavie bloku] --záhlavie bloku
[DECLARE --deklaračná časť
  <konštanty>
  <premenné>
  <kurzory>
  <uživateľsky definované výnimky>]
BEGIN
  <PL/SQL kód> -- časť výkonných príkazov
[EXCEPTION --časť výnimiek
  <ošetrenie výnimiek> ]
END;
```

PL/SQL blok

```
BEGIN
dbms_output.put_line('Hello world!');
END;
```

Syntaktické pravidlá a obmedzenia PL/SQL

KKUI
Katedra kybernetiky a umelej inteligencie

- Identifikátory
  - maximálna dĺžka 30 znakov
  - musí začínať písmenom (na 1. pozícii nesmie byť $, #, _ a číslica)
  - nesmie obsahovať bodky, medzery, pomlčky
  - nesmie obsahovať kľúčové slová

Pozn.:HELP RESERVED WORDS (SQL)/(PL/SQL)

pohľad v_$reserved_words

  - “SELECT”, “Procedúra 1“

Syntaktické pravidlá a obmedzenia PL/SQL

KKUI
Katedra kybernetiky a umelej inteligencie

- Oddeľovače:
  - +, -, * , /
  - ..
  - <>, !=, <=, >=
  - << >> - návestie
  - --, /* */
  - % (TYPE, ROWTYPE, implicitný kurzor ako sql%ROWCOUNT…)
  - ‘ vs. “
  - @
  - ;
  - :=
  - ||

Syntaktické pravidlá a obmedzenia PL/SQL

- komentáre
  - **--** jednoriadkový komentár

    napr.:

    DECLARE

    -- v tejto časti sú deklarované premenné

    BEGIN

    ...

    END;
  - **/* */** viacriadkový komntár

    napr.:

    BEGIN

    /* v tejto časti budu napísané príkazy,

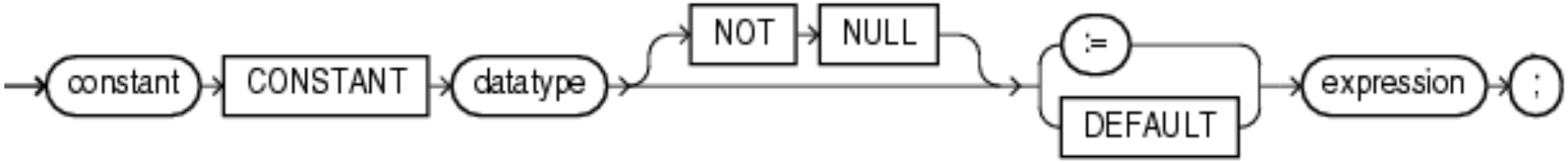
    ktoré budú vykonané */

    ...

    END;

Konštanty

- syntax:

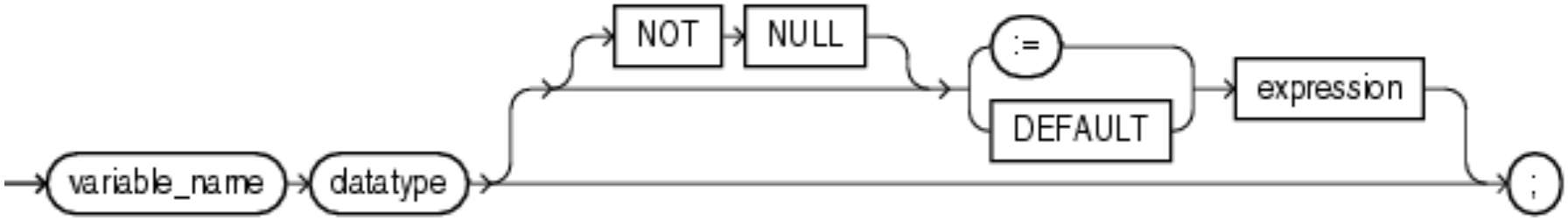


- ak nie je premenná inicializovaná, bude nadobúdať hodnotu NULL

Príklad:

```
DECLARE
   max_days_in_year CONSTANT INTEGER := 366;
BEGIN
   ...
END;
```

Premenné



- *priama deklarácia* -> výber typov:
  - VARCHAR2, NVARCHAR2, CHAR, NCHAR
  - NUMBER(10,2), INTEGER, BINARY_INTEGER...
  - DATE, TIMESTAMP [WITH TIME ZONE]
  - BOOLEAN (TRUE alebo FALSE alebo NULL)
  - BLOB, CLOB, NCLOB

  Príklad:

```
DECLARE
   v_meno VARCHAR 2;
   v_priezvisko VARCHAR2;
   datum_nar DATE;
BEGIN
...
END;
```

Premenné

- *deklarácia referovaním* -> mapovanie na existujúci stĺpec tabuľky resp. riadok tabuľky
  - <názov tabuľky>.<meno stĺpca>**%TYPE**
  - < názov tabuľky >**%ROWTYPE**

Príklad:

```
DECLARE
   v_plat EMPLOYEES.SALARY%TYPE;
   v_zamestnanec EMPLOYEES%ROWTYPE
BEGIN
...
END;
```

Pozn.:

- na konkrétnu položku záznamovej premennej pristupujeme: *v_zamestanec.salary*

Premenné

- priraďovanie hodnot:
  - počas deklarácie pomocou klauzuly **DEFAULT** alebo **:=**
    - mesto VARCHAR2 DEFAULT ‘Košice’;
    - plat NUMBER := 1000;
    - zac_dna DATE DEFAULT trunc(SYSDATE)
  - v tele programu/bloku
    - priamym priradením
      plat:=1.1*plat;
    - pomocou SELECT-u
      SELECT salary INTO plat FROM employees [WHERE …];

Premenné

```
DECLARE
   v_salary NUMBER;
   v_name VARCHAR2(10) NOT NULL := 'Milan';
 BEGIN
   SELECT salary INTO v_salary FROM employees
   WHERE name = v_name;
   dbms_output.put_line(,EMPLOYEE ' ||
   var_name || ' earns ' || v_salary ||' € ');
END; /
```

Trigger

- programová jednotka automaticky spúšťaná Oraclom pri výskyte určitej udalosti

Syntax:

```
CREATE [OR REPLACE] TRIGGER názov
  BEFORE|AFTER|INSTEAD OF udalosti
  [ premenne odkazujúce sa na staré a nové
  záznamy ]
  [FOR EACH ROW ]
  [WHEN ( podmienka )]
BEGIN
  <postupnosť príkazov>
END;
```

Trigger

- sa spusti pred (BEFORE) alebo po (AFTER) alebo namiesto (INSTEAD OF) udalosti, ktorá ho vyvolá
- udalosti:
  - *DML statement*

    DELETE, INSERT, UPDATE
  - *DDL statement*

    CREATE, ALTER, DROP
  - *Databázové operácie*

    SERVERERROR, LOGON, LOGOFF, STARTUP, SHUTDOWN

Trigger

- klauzula ***FOR EACH ROW***
  - ak je prítomná: trigger sa spustí pre každý riadok, ktorý je ovplyvnený príkazom, ktorým bol vyvolaný
    - row trigger
  - ak nie je prítomná: trigger sa spustí len raz
    - statement trigger
- klauzula ***WHEN***
  - špecifikuje podmienku, pre ktoré riadky sa má trigger spustit
  - syntax ako pre WHERE klauzulu

Trigger - príklad

```
CREATE OR REPLACE TRIGGER print_salary_changes
  BEFORE DELETE OR INSERT OR UPDATE ON employees
  FOR EACH ROW
  WHEN (NEW.job_id <> 'AD_PRES')  -- do not print
   information about President
DECLARE
  sal_diff  NUMBER;
BEGIN
  sal_diff  := :NEW.salary  - :OLD.salary;
  DBMS_OUTPUT.PUT(:NEW.last_name || ': ');
  DBMS_OUTPUT.PUT('Old salary = ' || :OLD.salary || ',
   ');
  DBMS_OUTPUT.PUT('New salary = ' || :NEW.salary || ',
   ');
  DBMS_OUTPUT.PUT_LINE('Difference: ' || sal_diff);
END;
```

Trigger – modifikované údaje

- predvolené názvy pre údaje, ktoré trigger spracováva sú NEW a OLD

  For an **INSERT** trigger, **OLD** contains **no values**, and **NEW** contains the **new values**

  For an **UPDATE** trigger, **OLD** contains the **old values**, and **NEW** contains the **new values**

  For a **DELETE** trigger, **OLD** contains the **old values**, and **NEW** contains **no values**

- pri príkazových trigger-och odkazujú na tabuľku, ktorá pozostáva zo spracovavaných riadkov

- pri riadkových sa odkazujú na každý jeden riadok postupne

Trigger - použitie

- Automaticky generované hodnoty závislých stĺpcov
- Vynútenie referenčnej integrity naprieč uzlami v distribuovanej databáze
- Vynútenie complexnej business logiky (pravidiel)
- Prevádzkovanie transparentného záznamu udalostí
- Prevádzkovanie auditovania
- Udržanie synchronizácie replikovaných tabuliek
- Zbieranie štatistík ohľadom prístupu k tabuľkám
- Modifikácia tabuliek počas DML príkazu pre views
- Publikovanie informácií ohľadom databázových udalostí, používateľských udalostí a SQL statements pre popis applikácie
- Obmedzenie DML operácií nad tabuľkami na pracovnú dobu
- Vynútenie autorizácie (security)
- Vyhnutie sa vadným transakciám

Trigger

- CREATE OR REPLACE
- ALTER TRIGGER trigger_name ENABLE/DISABLE/COMPILE;
- ALTER TABLE table_name DISABLE/ENABLE ALL TRIGGERS;
- DROP TRIGGER trigger_name;