SQL:2003

slajdy k přednášce DJ1

Jaroslav Pokorný

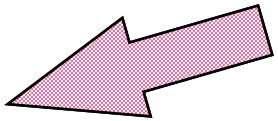
MFF UK, Praha

pokorny@ksi.mff.cuni.cz

Obsah

SQL:2003

- hodně korekcí chyb
- několik nových rysů



  - typy dat
    - operace
    - predikáty
  - Operace MERGE
  - OLAP: TABLESAMPLE
  - generované sloupce
  - identitní sloupce a generátory posloupností
  - Část 14 SQL/XML

Nové typy dat

- BIGINT
- MULTISET

Odstraněno

- BIT
- BIT VARYING

BIGINT

- přesnost BIGINT ≥ přesnost INTEGER ≥ přesnost SMALLINT
- stejný základ jako INT a SMALLINT
- stejné operátory jako na SMALLINT a INTEGER

MULTISET

- typy kolekcí:
  - MULTISET
  - ARRAY
- Multimnožina je neuspořádaná kolekce proměnné délky, její prvky mají specifikovaný typ
  - žádná (specifikovaná) maximální kardinalita
  - *nemusí* být specifikovaná ani u ARRAY

MULTISET - definice

- A INTEGER MULTISET
- B ROW(F1 BIGINT, F2 VARCHAR(4000)) MULTISET
- C INTEGER MULTISET()
  - prázdná multimnožina s celočíselným typem prvků (*ne* NULL!)
- D INTEGER MULTISET(2, 3, 5, 7)
  - multimnožina celých čísel s několika prvočísly
- E INTEGER MULTISET(SELECT A FROM R WHERE A > 10)
  - multimnožina celých čísel z hodnot ve sloupci tabulky

MULTISET - operátory a funkce

/* *multi* představuje multimnožinu */

- CARDINALITY(*multi*)
  - vrací počet prvků v *multi*
- SET(multi)
  - vrací obsah z *multi* bez duplicit prvků
- ELEMENT(*multi*)
  - kardinalita této multimnožiny musí být 1
  - vrací odpovídající jediný prvek (singleton)

MULTISET - operátory a funkce

- UNNEST(*multi*) AS *jméno*
  - "odhnízdí" *multi* a vrací tyto prvky v řádcích virtuální tabulky *jméno*

  UNNEST MULTISET (2, 3, 5, 7) AS P

P
2
3
5
7

MULTISET – operátory a funkce

- *multi1* MULTISET *op* [*kvantifikátor*] *multi2*

  op — UNION, EXCEPT, INTERSECT

  kvantifikátor — ALL nebo DISTINCT

Pz.: podobné množinovým operátorům

UNION, EXCEPT a INTERSECT

Pz.: implicitně se uvažuje

MULTISET – operátory a funkce

```
SELECT A MULTISET INTERSECT
  DISTINCT B
FROM R
WHERE CARDINALITY(B) > 50;
```

MULTISET – operátory a funkce

Nové agregační funkce pro multimnožiny:

Předpoklad: skupina daná GROUP BY nebo sloupcem

- COLLECT — transformuje hodnoty ve skupině do multimnožiny
- FUSION — vytváří *sjednocení* multimnožin ve skupině — počet duplicit dané hodnoty ve výsledku je součtem počtů duplicit v multimnožinách řádků daných skupin
- INTERSECTION — dělá *průnik* multimnožin ve skupině — počet duplicit dané hodnoty ve výsledku je minimem z počtů duplicit v multimnožinách řádků daných skupin

MULTISET – operátory a funkce

```
CREATE TABLE Logins(
    Id_sezení INT NOT NULL PRIMARY KEY,
    Úspěšné BOOLEAN NOT NULL,
    Id_uživatel INT,
    Pokusy ROW(VARCHAR(128), VARCHAR (128))
        MULTISET);
```

username, password

```
SELECT Id_uživatel, COLLECT (Id_sezení) AS S_Ids,
    FUSION(Pokusy) AS Všechny_pokusy,
    INTERSECTION(Pokusy) AS Společné_pokusy
FROM Logins
WHERE Úspěšné
GROUP BY Id_uživatel;
```

MULTISET – operátory a funkce

Část Logins pro uživatele s Id = 8 a jeho úspěšné pokusy

Logins:

Id_sezení	Id_uživatel	Pokusy
a	8	multiset[(1,x),(2,y)]
b	8	multiset[(1,x)]
c	8	multiset[(1,x),(3,h)]

Výsledek (1 řádek pro uživatele s Id = 8)

S_Ids	Všechny_pokusy	Společné_pokusy
multiset[a,b,c]	multiset[(1,x), (1,x), (1,x), (2,y), (3,h)]	multiset[(1,x)]

Nové predikáty

- Porovnávací (multimnožinové!) operátory = a <>
- [NOT] MEMBER [OF]
- [NOT] SUBMULTISET [OF]
- IS [NOT] A SET

predikát MEMBER

h [NOT] MEMBER [OF] multi

- *h* porovnatelná s typem prvků v *multi*
- jestliže je *multi* prázdná, vrací FALSE
- jestliže je *h* rovna nějakému prvku z *multi*, vrací TRUE
- jestliže nějaký prvek z *multi* je NULL, vrací UNKNOWN

predikát SUBMULTISET

multi1 [NOT] SUBMULTISET [OF] multi2

…a typy prvků multimnožin musí být porovnatelné

- relace „být podmultimnožinou“
- jestliže $|multi1| = |multi2|$ a jestliže každá hodnota v *multi1* má korespondující hodnotu v *multi2*, pak vrací TRUE

predikát SET

multi IS [NOT] A SET

- *multi* musí být multimnožina
- jestliže neexistují žádné duplicitní hodnoty v *multi*, vrací TRUE
- maximálně 1 NULL hodnota v množině

MERGE

- Kombinace INSERT a UPDATE do jednoho příkazu
- Řádky vstupní (referenční) tabulky jsou rozděleny do dvou skupin podle predikátu P:

  insert source table *(IST)*, jestliže je na nich hodnota P FALSE nebo UNKNOWN a

  update source table *(UST)*, jestliže je P vyhodnotí jako TRUE.

MERGE

- IST se vkládá do cílové tabulky R.
- každý řádek v R, který se shoduje s nějakým řádkem v UST, je v R aktualizován.
  - Je chybou, jestliže ta shoda nastává u více řádků cílové tabulky.
- Syntaxe: pomocí složek MATCHED a NOT MATCHED (každá nejvýše jednou, v libovolném pořadí)

MERGE

```
MERGE INTO tabulka [AS jméno]
   USING referenční tabulka
   ON podmínka
   WHEN MATCHED THEN
                SET sloupec = hodnota;

MERGE INTO tabulka [AS jméno]
   USING referenční tabulka
   ON podmínka
   WHEN NOT MATCHED THEN
                INSERT [(seznam_sloupců)]
                VALUES (seznam_hodnot);
```

MERGE

Zásoby(zboží, popis, množství)
Dovoz(zboží, popis, množství, cena)

```
MERGE INTO Zásoby AS Z
   USING (SELECT zboží, popis, množství
                FROM Dovoz) AS D
   ON (Z.zboží = D.zboží)
WHEN MATCHED THEN
   UPDATE SET množství = Z.množství + D.množství
WHEN NOT MATCHED THEN
   INSERT (zboží, popis, množství)
   VALUES (D.zboží, D.popis, D.množství);
```

TABLESAMPLE

- Nový rys pro zpracování typu OLAP
- Dovoluje vyhodnocování agregací na vzorcích odvozených z databázových dat
- Dovoluje rychlejší ladění aplikace, když je db velká
- Dvě formy vzorkování: BERNOULLI a SYSTEM

TABLESAMPLE

```
TABLESAMPLE {BERNOULLI | SYSTEM}
(počet%) [REPEATABLE(počet_op)]
```

- BERNOULLI: tabulka vzorku obsahuje přibližně počet% řádků původní tabulky; pravděpodobnost daného řádku původní tabulky vyskytujícího se v tabulce vzorku je počet%, nezávisle na každém dalším řádku.
- SYSTEM: tabulka vzorku obsahuje přibližně počet % řádků původní tabulky; pravděpodobnost daného řádku původní tabulky vyskytujícího se v tabulce může záviset na řádcích již vložených do vzorku.
- REPEATABLE: opakované volání v počtu počet_op generuje stejný vzorek pro stejný zdroj.

TABLESAMPLE

D.: Najdi přibližný odhad celkové mzdy zaměstnanců každého oddělení

```
SELECT odd, SUM(mzda) * 10
FROM Zaměstnanci TABLESAMPLE
   BERNOULLI (10)
REPEATABLE (5)
GROUP BY odd
```

Generátory posloupností

- mechanismus pro automatické generování posloupnosti hodnot
- dvě možnosti: explicitně pomocí CREATE, implicitně s identitním sloupcem

Parametry:

- typ dat (přesný numerický)
- počáteční hodnota
- přírůstek (kladný: rostoucí, záporný: klesající, impl. = 1)
- MINVALUE a MAXVALUE
- možnost cyklu (po dosažení MAXVALUE se řada opakuje)
- externí (explicitní objekt schématu) nebo interní (část dalšího objektu schématu, např. sloupce)

Externí generátory posloupností

```
CREATE SEQUENCE jméno_p AS type
    START WITH hodnota
    INCREMENT BY hodnota
    MAXVALUE hodnota
    CYCLE;
```

} volby pro generátor posloupnosti

- možnosti:
  - NO CYCLE
  - NO MAXVALUE, MINVALUE, NO MINVALUE
  - pořadí složek může být různé

Generátory posloupností

- každý generátor posloupnosti má “běžnou základní hodnotu” (Z) — na počátku nastavenou na počáteční hodnotu
- generování následující hodnoty:

  NEXT VALUE FOR jméno_p
- vrací Z + *N* **přírůstek* pro nějaké $N \geq 0$
  - jestliže je spočítaná hodnota > MAXVALUE (nebo < MINVALUE) a NO CYCLE, pak nastává výjimečný stav
  - jinak, spočítej novou hodnotu pro nějaké *N*.

Generátory posloupností

Př.: máme definován generátor posloupnosti sekvgen a relaci

Objednávky(č_o, zboží, množství)

```
INSERT INTO Objednávky
    VALUES (NEXT VALUE FOR sekvgen, ‘řemen’, 2
   );

UPDATE Objednávky
    SET č_o = NEXT VALUE FOR sekvgen;
```

Generátory posloupností

- Při změně generátoru lze nastavit počáteční hodnotu, maximum nebo minimum, přírůstek, volbu cyklu

```
ALTER SEQUENCE jméno_p
RESTART WITH nový_základ;
```

- Zrušení generátoru

```
DROP SEQUENCE jméno_p;
```

Interní generátory posloupnosti

- GENERATED ALWAYS nebo GENERATED BY DEFAULT
  - ALWAYS — pak není dovolen UPDATE na sloupec; INSERT požaduje OVERRIDING SYSTEM VALUE
  - BY DEFAULT — dovolen INSERT nebo UPDATE sloupce; jestliže sloupec není specifikován na INSERT, pak je jeho hodnota automaticky generována

Identitní sloupce

- identitní sloupec: mechanismus dovolující automatickou populaci klíčů tabulky (jeden sloupec)

```
CREATE TABLE Zaměstnanci (
  z_id    INTEGER GENERATED ALWAYS AS
                IDENTITY (START WITH 100
                          INCREMENT 1
                          MINVALUE 100
                          NO MAXVALUE
                          NO CYCLE),
  mzda DECIMAL(7,2), ...,
);
```

Generované sloupce

- původní sloupce základních tabulek: základní sloupce
- generovaný sloupec má hodnoty vypočítané z hodnot 0-více základních sloupců ve stejném řádku

```
CREATE TABLE Zaměstnanci (
z_ID          INTEGER,
odd string(6)
mzda          DECIMAL(7,2),
osobní_p      DECIMAL(7,2),
celková_m GENERATED ALWAYS AS (mzda+osobní_p),
uživatel      GENERATED ALWAYS AS
                        (CURRENT_USER ));
```

- hodnoty generovaných sloupců se počítají automaticky při INSERTu do tabulky.

Závěr

- Uvedená rozšíření podporují tvorbu analytických funkcí v SQL, tj. jsou použitelná pro OLAP a dnes pro Big Analytics.