Kapitola 4

Vkládání, aktualizace, mazání

Tématem několika předchozích kapitol byly základní techniky pokládání dotazů, které se všechny zaměřovaly na zisk dat z databáze. V kapitole čtvrté půjde o něco úplně jiného, jejím tématem jsou následující operace:

- Vkládání nových záznamů do databáze
- Aktualizace existujících záznamů
- Mazání záznamů, které již v databázi nechcete

Abyste mohli snáze najít přesně to, co potřebujete, jsou návody v této kapitole setříděné podle výše uvedených druhů operací: první část kapitoly se věnuje vkládání nových záznamů, následuje část věnovaná aktualizaci záznamů a nakonec část, která obsahuje návody pro mazání záznamů.

Vkládání záznamů je obvykle velmi přímočarou operací. Začíná jednoduchým problémem, jak do databáze vložit jeden nový řádek. Často je však výhodnější použít při vkládání nových řádků množinový přístup. V této kapitole proto také najdete návody pro vkládání více řádků najednou.

Podobně i aktualizace a mazání jsou nejprve představeny ve své nejjednodušší formě. Můžete aktualizovat či smazat pouze jeden záznam. Ale můžete také aktualizovat celé množiny záznamů najednou, a to velmi efektivně. Existuje také spousta užitečných způsobů mazání záznamů. Můžete například mazat řádky jedné tabulky v závislosti na tom, zda existují také v jiné tabulce či nikoliv.

V SQL je nově dokonce možné řádky najednou vkládat, aktualizovat i mazat. Možná vám to zatím nepřijde příliš užitečné, ale příkaz MERGE představuje velmi efektivní způsob synchronizace tabulky s externím zdrojem dat (například transportním souborem ze vzdáleného systému). Viz recept na straně 95.

Vložení nového záznamu

Problém

Chcete do tabulky vložit nový záznam. Chcete například vložit jeden nový záznam do tabulky EMP. Hodnota záznamu pro DEPTNO bude 50, pro DNAME PROGRAMMING a LOC BALTIMORE.

Řešení

Když chcete do tabulky vložit pouze jeden záznam, použijte příkaz INSERT spolu s klauzulí VALUES:

```
insert into dept (deptno,dname,loc)
values (50,'PROGRAMMING','BALTIMORE')
```

Pokud pracujete s DB2 nebo MySQL, máte možnost vložit více řádků najednou, když zadáte více parametrů klauzule VALUES najednou:

```
/* vložení více řádků */
insert into dept (deptno,dname,loc)
values (1,'A','B'),
       (2,'B','C')
```

Rozbor řešení

Příkaz INSERT vám umožňuje tvorbu nových řádků v tabulkách databáze, kterou spravujete. Syntaxe je shodná pro všechny databázové systémy.

Můžete také vynechat v příkazu INSERT výpis sloupců:

```
insert into dept
values (50,'PROGRAMMING','BALTIMORE')
```

Když sloupce nevypíšete, musíte vložit nové hodnoty záznamů do *všech* sloupců tabulky a dávat pozor na pořadí hodnot v klauzuli VALUES; musíte vložit hodnoty ve stejném pořadí, jaké vidíte, když zobrazíte tabulku prostřednictvím příkazu SELECT *.

Vkládání standardních hodnot

Problém

Tabulka může být definovaná takovým způsobem, že některé sloupce obsahují standardní, předem definované hodnoty. Chcete do tabulky vložit řádek obsahující standardní hodnoty záznamů, aniž byste museli tyto hodnoty specifikovat. Podívejte se na tuto tabulku:

```
create table D (id integer default 0)
```

Chcete do tabulky vložit nulu, aniž byste ji explicitně specifikovali v příkazu INSERT. Chcete prostě vložit do tabulky standardní hodnotu, ať už je jakákoliv.

Řešení

Všechny databázové systémy podporují klíčové slovo DEFAULT, jehož prostřednictvím můžete explicitně definovat standardní hodnotu záznamu v daném sloupci. Některé systémy pak nabízejí více způsobů řešení tohoto problému.

Tento příklad ilustruje použití klíčového slova DEFAULT:

```
insert into D values (default)
```

Můžete také explicitně specifikovat název sloupce, což musíte udělat vždy, když nevkládáte standardní hodnotu do všech sloupců tabulky:

```
insert into D (id) values (default)
```

Databáze Oracle 8*i* a dřívější verze nepodporují klíčové slovo DEFAULT. V těchto verzích není možné vložit do sloupce standardní hodnotu záznamu (až do verze Oracle 8*i*, Oracle 9*i* už to umožňuje).

MySQL vám umožňuje zadat prázdný seznam hodnot, pokud je nějaká standardní hodnota definovaná pro všechny sloupce:

```
insert into D values ()
```

V tomto případě se nastaví hodnoty záznamů ve všech sloupcích na hodnoty standardní.

Systémy PostgreSQL a MySQL podporují klauzuli DEFAULT VALUES:

```
insert into D default values
```

Když použijete tuto klauzuli, nastaví se hodnoty záznamů ve všech sloupcích na hodnoty standardní.

Rozbor řešení

Když použijete v seznamu hodnot klíčové slovo DEFAULT, vložíte do daného sloupce příslušnou standardní hodnotu záznamu definovanou při vzniku tabulky. Toto klíčové slovo lze použít pro všechny databázové systémy.

Uživatelé systémů MySQL, PostgreSQL a SQL Server mají v případě, že jsou standardní hodnoty záznamů definované pro všechny sloupce tabulky (jak je tomu zde u tabulky D), možnost zadat prázdný seznam hodnot klauzule VALUES (MySQL), nebo použít pro vytvoření nového řádku se standardními hodnotami záznamů klauzuli DEFAULT VALUES (PostgreSQL a SQL Server); jinak musíte specifikovat standardní hodnoty pro každý sloupec zvlášť.

Vložení standardní hodnoty záznamu do sloupce v tabulce, ve které jsou smíšené standardní i nestandardní hodnoty záznamů, je jednoduché: prostě tento sloupec vynecháte ze seznamu sloupců příkazu INSERT; použití klíčového slova DEFAULT není nutné. Řekněme, že tabulka D má ještě jeden sloupec, pro který není definovaná standardní hodnota záznamu:

```
create table D (id integer default 0, foo varchar(10))
```

Do sloupce ID vložíte standardní hodnotu záznamu jednoduše tak, že na seznam sloupců příkazu INSERT dáte pouze sloupec FOO:

```
insert into D (name) values ('Bar')
```

Výsledkem provedení tohoto příkazu bude řádka, ve které má sloupec ID hodnotu záznamu 0 a FOO BAR. Ve sloupci ID tedy bude standardní hodnota záznamu, protože jste žádnou jinou nespecifikovali.

Přepsání standardní hodnoty záznamu na prázdnou hodnotu záznamu

Problém

Chcete hodnotu záznamu ve sloupci, pro který je definovaná standardní hodnota záznamu, přepsat na hodnotu prázdnou. Podívejte se na tuto tabulku:

```
create table D (id integer default 0, foo VARCHAR(10))
```

Chcete vložit řádek s prázdnou hodnotou záznamu ve sloupci ID.

Řešení

Můžete explicitně specifikovat prázdnou hodnotu záznamu na seznamu hodnot záznamů:

```
insert into d (id, foo) values (null, 'Brighten')
```

Rozbor řešení

Ne každý si uvědomuje, že je možné explicitně specifikovat prázdnou hodnotu na seznamu hodnot příkazu INSERT. Když nechcete pro daný sloupec specifikovat žádnou hodnotu záznamu, obvykle ho vynecháte ze seznamu sloupců a hodnot:

```
insert into d (foo) values ('Brighten')
```

Zde pro sloupec ID nespecifikujete žádnou hodnotu záznamu. Někdo z vás si může myslet, že když sloupec vynechá, hodnota jeho záznamu zůstane prázdná. To však není pravda, protože při vzniku tabulka byla nastavena jeho standardní hodnota, takže výsledkem tohoto příkazu INSERT bude nastavení hodnoty sloupce ID na nulu (standardní hodnotu). Když specifikujete hodnotu záznamu v daném sloupci jako prázdnou, hodnota záznamu se nastaví na prázdnou navzdory tomu, že je definovaná hodnota standardní.

Kopírování řádků z jedné tabulky do druhé

Problém

Chcete prostřednictvím dotazu kopírovat řádky z jedné tabulky do druhé. Dotaz může být jednoduchý nebo složený, ale hlavním cílem je vložit jeho výsledek do jiné tabulky. Například chcete vložit řádky tabulky DEPT do tabulky DEPT_EAST. Tabulka DEPT_EAST již existuje, má stejnou strukturu (má stejné sloupce a datové typy) a je prázdná.

Řešení

Použijte příkaz INSERT a dotaz, abyste získali řádky, které chcete:

```
1 insert into dept_east (deptno,dname,loc)
2 select deptno,dname,loc
3   from dept
4  where loc in ( 'NEW YORK','BOSTON' )
```

Rozbor řešení

Za příkaz INSERT jednoduše vložte dotaz, který vrací požadované řádky. V případě, že chcete vložit do cílové tabulky všechny řádky tabulky zdrojové, vynechejte v dotazu klauzuli WHERE. Jako při normálním vkládání nemusíte explicitně specifikovat, do kterých sloupců vkládáte. Když to však neuděláte, budete vkládat hodnoty záznamů do *všech* sloupců cílové tabulky a musíte dávat pozor na pořadí hodnot v příkazu SELECT, viz recept *Vložení nového záznamu*.

Kopírování vlastností tabulky

Problém

Chcete vytvořit tabulku, která bude obsahovat stejnou množinu sloupců, jako tabulka, která již existuje. Například chcete vytvořit kopii tabulky DEPT a pojmenovat ji jako DEPT_2. Nechcete kopírovat řádky, pouze strukturu sloupců.

Řešení

DB2

Použijte klauzuli LIKE spolu s příkazem pro tvorbu nové tabulky CREATE TABLE:

```
create table dept_2 like dept
```

Oracle, MySQL a PostgreSQL

Použijte příkaz CREATE TABLE spolu s vnořeným dotazem, který nevrací žádné řádky:

```
1 create table dept_2
2 as
```

```
3 select *
4   from dept
5  where 1 = 0
```

SQL Server

Použijte klauzuli INTO spolu s vnořeným dotazem, který nevrací žádné řádky:

```
1 select *
2   into dept_2
3   from dept
4  where 1 = 0
```

Rozbor řešení

DB2

Příkaz systému DB2 CREATE TABLE spolu s klauzulí LIKE vám umožňuje použít jednu tabulku jako vzor pro tvorbu jiné tabulky. Jednoduše uveďte jméno vzorové tabulky za klauzulí LIKE.

Oracle, MySQL a PostgreSQL

Když vytváříte tabulku výše uvedeným způsobem (CREATE TABLE, AS, SELECT) a nepoužijete neplatnou podmínku v klauzuli WHERE, přenesou se do nové tabulky všechny řádky z tabulky původní. Ve výše uvedeném řešení podmínka 1 = 0 v klauzuli WHERE způsobí, že dotaz nevrátí žádné řádky. Výsledkem je prázdná tabulka, která má sloupce uvedené v příkazu SELECT.

SQL Server

Když použijete pro zkopírování tabulky klauzuli INTO a nepoužijete neplatnou podmínku v klauzuli WHERE, použijí se k naplnění nové tabulky všechny řádky vašeho dotazu. Ve výše uvedeném řešení podmínka 1 = 0 v klauzuli WHERE způsobí, že dotaz nevrátí žádné řádky. Výsledkem je prázdná tabulka, která má sloupce uvedené v příkazu SELECT.

Vkládání řádků do více tabulek najednou

Problém

Chcete vložit řádky, které vrací nějaký dotaz, do více cílových tabulek najednou. Například chcete řádky tabulky DEPT vložit do tabulek DEPT_EAST, DEPT_WEST a DEPT_MID. Všechny tři tabulky mají identickou strukturu (stejné sloupce a datové typy) jako tabulka DEPT a jsou prázdné.

Řešení

Řešením je vložení výsledků dotazu do cílových tabulek. Rozdíl mezi řešením v receptu *Kopírování řádků z jedné tabulky do druhé* a řešením tohoto problému spočívá v tom, že chcete kopírovat do více cílových tabulek najednou.

Oracle

Použijte buď příkaz INSERT ALL, nebo INSERT FIRST. Kromě použití výrazů ALL a FIRST mají oba příkazy stejnou syntaxi. V následujícím dotazu se využívá příkaz INSERT ALL, aby byly brány v potaz všechny možné cílové tabulky:

```
1   insert all
2     when loc in ('NEW YORK','BOSTON') then
```

```
3       into dept_east (deptno,dname,loc) values (deptno,dname,loc)
4     when loc = 'CHICAGO' then
5       into dept_mid (deptno,dname,loc) values (deptno,dname,loc)
6     else
7       into dept_west (deptno,dname,loc) values (deptno,dname,loc)
8     select deptno,dname,loc
9       from dept
```

DB2

Vkládejte do řádkového pohledu, který provádí sjednocení tabulek, do nichž vkládáte. Zároveň musíte použít omezení, která zaručí, že se každá řádka vloží do správné tabulky:

```
create table dept_east
( deptno integer,
  dname  varchar(10),
  loc    varchar(10) check (loc in ('NEW YORK','BOSTON')))

create table dept_mid
( deptno integer,
  dname  varchar(10),
  loc    varchar(10) check (loc = 'CHICAGO'))

create table dept_west
( deptno integer,
  dname  varchar(10),
  loc    varchar(10) check (loc = 'DALLAS'))

1  insert into (
2    select * from dept_west union all
3    select * from dept_east union all
4    select * from dept_mid
5  ) select * from dept
```

MySQL, PostgreSQL a SQL Server

Tyto systémy nepodporovaly vkládání řádků do více tabulek najednou v době, kdy byla kniha napsána.

Rozbor Řešení

Oracle

Vkládání řádků do více tabulek najednou funguje v tomto systému na základě klauzule WHEN-THEN-ELSE, prostřednictvím které se vyhodnotí řádky vnořeného příkazu SELECT a jsou vloženy do správné tabulky. V tomto příkladu získáme stejné výsledky, když použijeme příkaz INSERT ALL, jako když použijeme příkaz INSERT FIRST, ale INSERT FIRST přeruší cyklus WHEN-THEN-ELSE ve chvíli, kdy narazí na podmínku, která má hodnotu TRUE; INSERT ALL naproti tomu vyhodnotí všechny podmínky nezávisle na tom, jestli v průběhu předchozího vyhodnocování narazil na podmínku, jež má hodnotu TRUE, takže ho můžete použít k vložení stejného řádku do více než jedné tabulky.

DB2

Řešení pro DB2 není úplně konvenční. Vyžaduje, že tabulky do kterých vkládáte, mají nastavená taková omezení, aby se každá řádka, jež je výsledkem dotazu, vložila do tabulky, do které patří. Řešení funguje tak, že se řádky vkládají do pohledu, který je sjednocením tabulek. Pokud nejsou

omezující podmínky pro tabulky vypsané v příkazu INSERT jedinečné (platí pro více tabulek najednou), příkaz INSERT nebude vědět, kam řádek vložit, a selže.

MySQL, PostgreSQL a SQL Server

Tyto systémy nepodporovaly vkládání řádků do více tabulek najednou prostřednictvím jednoho příkazu v době, kdy byla kniha napsána.

Omezení vkládání do určitých sloupců

Problém

Nechcete, aby uživatelé nebo nějaká pochybná aplikace, mohli vkládat data do určitých sloupců. Například chcete umožnit nějakému programu, aby mohl vkládat data do tabulky EMP, ale pouze do sloupců EMPNO, ENAME a JOB.

Řešení

Vytvořte pohled, který bude obsahovat pouze ty sloupce, k nimž chcete umožnit přístup. Pak donuťte všechny příkazy INSERT, aby vkládaly do tohoto pohledu.

Vytvořte pohled, který bude obsahovat tři sloupce tabulky EMP, třeba takto:

```
create view new_emps as
select empno, ename, job
  from emp
```

Umožněte přístup do tohoto pohledu pouze těm programům a uživatelům, kterým nechcete dovolit zapisovat do jiných sloupců, než do těchto tří. Neumožňujte jim přistupovat k celé tabulce EMP. Uživatelé tak budou moci vytvářet záznamy v tabulce EMP prostřednictvím záznamů vytvořených v pohledu NEW_EMP, ale nebudou moci zapisovat data do jiných sloupců, než do těch, které jsou v tomto pohledu.

Rozbor řešení

Když vkládáte do jednoduchého pohledu, jako v tomto příkladu, váš databázový server vloží data, která jste do pohledu vložili, do příslušné tabulky. Například následující záznam:

```
insert into new_emps
   (empno ename, job)
   values (1, 'Jonathan', 'Editor')
```

vloží server do tabulky tímto způsobem:

```
insert into emp
   (empno ename, job)
   values (1, 'Jonathan', 'Editor')
```

Je také možné, i když zřejmě ne tak užitečné, vkládat do řádkového pohledu (v současné době tento způsob řešení podporuje pouze Oracle):

```
insert into
  (select empno, ename, job
     from emp)
values (1, 'Jonathan', 'Editor')
```

Vkládání do pohledů je samostatným tématem. Jeho pravidla jsou velmi komplikovaná pro všechny pohledy, kromě těch nejjednodušších. V případě, že chcete techniku vkládání do pohledů využívat, musíte se plně seznámit s pravidly jejího používání pro váš databázový systém.

Modifikace záznamů v tabulce

Problém

Chcete modifikovat hodnoty záznamů všech řádků tabulky, nebo jen některých z nich. Chcete například zvýšit plat všem zaměstnancům pro DEPTNO 20 o 10%. Následující výsledná množina obsahuje hodnoty záznamů ve sloupcích DEPTNO, ENAME a SAL všech zaměstnanců pro DEPTNO 20:

```
select deptno,ename,sal
  from emp
 where deptno = 20
 order by 1,3

DEPTNO ENAME             SAL
------ ---------- ----------
    20 SMITH             800
    20 ADAMS            1100
    20 JONES            2975
    20 SCOTT            3000
    20 FORD             3000
```

Všem zaměstnancům chcete zvýšit plat o 10 %.

Řešení

Když chcete modifikovat existující řádky tabulky, použijte příkaz UPDATE. Například:

```
1 update emp
2    set sal = sal*1.10
3  where deptno = 20
```

Rozbor řešení

Použijte příkaz UPDATE spolu s klauzulí WHERE, abyste specifikovali, které řádky chcete modifikovat; když klauzuli WHERE vynecháte, budete modifikovat všechny řádky. Výraz SAL*1.10 v tomto řešení způsobí, že se všechny platy zvýší o 10 %.

Když se připravujete na hromadnou modifikaci, hodí se prohlédnout si její výsledek. Můžete to udělat tak, že použijete příkaz SELECT, který bude obsahovat výrazy, jež chcete dát do klauzulí SET. Následující příkaz SELECT zobrazí výsledky navýšení platu všech zaměstnanců o 10 %:

```
select deptno,
       ename,
       sal      as orig_sal,
       sal*.10  as amt_to_add,
       sal*1.10 as new_sal
  from emp
 where deptno=20
 order by 1,5

DEPTNO ENAME  ORIG_SAL AMT_TO_ADD  NEW_SAL
------ ------ -------- ----------  -------
    20 SMITH       800         80      880
    20 ADAMS      1100        110     1210
    20 JONES      2975        298     3273
    20 SCOTT      3000        300     3300
    20 FORD       3000        300     3300
```

Navýšení platů lze sledovat ve dvou sloupcích: v jednom z nich je navýšení oproti původnímu platu, v druhém pak nová výše platu.

Aktualizace řádků tabulky za předpokladu, že tyto řádky existují v jiné tabulce

Problém

Chcete aktualizovat řádky tabulky v případě, že tyto řádky existují i v jiné tabulce. Například pokud pro daného zaměstnance existuje řádek v tabulce EMP_BONUS, chcete zvýšit hodnotu záznamu pro plat tohoto zaměstnance v tabulce EMP o 20 %. Tato výsledná množina obsahuje data, která jsou v tabulce EMP_BONUS:

```
select empno, ename
  from emp_bonus

     EMPNO ENAME
---------- ---------
      7369 SMITH
      7900 JAMES
      7934 MILLER
```

Řešení

Použijte vnořený dotaz v klauzuli WHERE příkazu UPDATE, abyste nalezli zaměstnance tabulky EMP, kteří jsou zároveň v tabulce EMP_BONUS. Příkaz UPDATE tak bude pracovat pouze s těmito řádky a umožní vám zvýšit plat daných zaměstnanců o 20 %:

```
1 update emp
2    set sal=sal*1.20
3  where empno in ( select empno from emp_bonus )
```

Rozbor řešení

Výsledky vnořeného dotazu představují řádky tabulky EMP, které se budou aktualizovat. Klauzule IN zjišťuje, zda se hodnoty záznamů ve sloupci EMPNO tabulky EMP nacházejí ve výsledcích vnořeného dotazu. Pokud ano, odpovídající hodnoty záznamů ve sloupci SAL se upraví.

Alternativně můžete místo klauzule IN použít klauzuli EXISTS:

```
update emp
   set sal = sal*1.20
 where exists ( select null
                  from emp_bonus
                 where emp.empno=emp_bonus.empno )
```

Možná jste překvapeni, že na seznamu příkazu SELECT klauzule EXISTS vnořeného dotazu vidíte prázdnou hodnotu záznamu. Nemusíte se však ničeho obávat, protože prázdná hodnota záznamu aktualizaci řádků neovlivňuje. Podle názoru autora knihy naopak zlepšuje čitelnost, protože je lépe vidět, že naproti řešení, které využívá vnořený dotaz s klauzulí IN, aktualizaci řídí (tedy rozhoduje o tom, jaké řádky se budou aktualizovat) klauzule WHERE vnořeného dotazu a ne hodnoty na seznamu příkazu SELECT.

Aktualizace hodnot záznamů řádku tabulky podle hodnot záznamů řádku jiné tabulky

Problém

Chcete aktualizovat řádky jedné tabulky hodnotami záznamů z jiné tabulky. Například máte tabulku, která se jmenuje NEW_SAL, a obsahuje záznamy o nové výši platů určitých zaměstnanců. Tabulka obsahuje tyto sloupce:

```
select *
  from new_sal

DEPTNO        SAL
------ ----------
    10       4000
```

Sloupec DEPTNO je pro tabulku NEW_SAL primárním klíčem. Chcete aktualizovat výši platů a provizí ze zisku určitých zaměstnanců v tabulce EMP podle hodnot záznamů v tabulce NEW_SAL, pokud jsou hodnoty záznamů v EMP.DEPTNO a NEW_SAL.DEPTNO shodné, tak, že se hodnota záznamů ve sloupci EMP.SAL nastaví podle hodnoty záznamů ve sloupci NEW_SAL.SAL a hodnota záznamů ve sloupci EMP.COMM se nastaví na 50 % hodnoty záznamů ve sloupci NEW_SAL.SAL. V tabulce EMP jsou tyto řádky:

```
select deptno,ename,sal,comm
  from emp
 order by 1

DEPTNO ENAME             SAL       COMM
------ ---------- ---------- ----------
    10 CLARK            2450
    10 KING             5000
    10 MILLER           1300
    20 SMITH             800
    20 ADAMS            1100
    20 FORD             3000
    20 SCOTT            3000
    20 JONES            2975
    30 ALLEN            1600        300
    30 BLAKE            2850
    30 MARTIN           1250       1400
    30 JAMES             950
    30 TURNER           1500          0
    30 WARD             1250        500
```

Řešení

Abyste nalezli a vrátili nové hodnoty záznamů ve sloupci COMM pro příkaz UPDATE, vytvořte spojení mezi tabulkami NEW_SAL a EMP. Aktualizace tohoto typu se běžně provádějí prostřednictvím korelovaného vnořeného dotazu. Jiným způsobem řešení je tvorba pohledu (obyčejného nebo řádkového, v závislosti na tom, jaké jsou možnosti vašeho databázového systému) a jeho následná aktualizace.

DB2 a MySQL

Pro nastavení nových hodnot ve sloupcích SAL a COMM tabulky EMP použijte korelovaný vnořený dotaz. Korelovaný vnořený dotaz použijte také k identifikaci řádků tabulky EMP, které se mají aktualizovat:

```
update emp e set (e.sal,e.comm) = (select ns.sal, ns.sal/2
                                     from new_sal ns
                                    where ns.deptno=e.deptno)
 where exists ( select null
                  from new_sal ns
                 where ns.deptno = e.deptno )
```

Oracle

Způsob řešení pro DB2 funguje i v Oraclu, ale alternativně můžete také aktualizovat řádkový pohled:

```
update (
 select e.sal as emp_sal, e.comm as emp_comm,
        ns.sal as ns_sal, ns.sal/2 as ns_comm
   from emp e, new_sal ns
  where e.deptno = ns.deptno
) set emp_sal = ns_sal, emp_comm = ns_comm
```

PostgreSQL

Způsob řešení pro DB2 funguje i v PostgreSQL, ale alternativně můžete (což je velmi výhodné) provést spojení přímo v příkazu UPDATE:

```
update emp
   set sal = ns.sal,
       comm = ns.sal/2
  from new_sal ns
 where ns.deptno = emp.deptno
```

SQL Server

Způsob řešení pro DB2 funguje i v SQL Server, ale alternativně můžete (podobně jako v PostgreSQL) provést spojení přímo v příkazu UPDATE:

```
update e
   set e.sal  = ns.sal,
       e.comm = ns.sal/2
  from emp e,
       new_sal ns
 where ns.deptno = e.deptno
```

Rozbor řešení

Před samotným rozborem řešení bude vhodné zmínit něco velmi důležitého ohledně aktualizací, které využívají k získání aktuálních hodnot záznamů dotazy. Klauzule WHERE ve vnořeném dotazu korelované aktualizace není totožná s klauzulí WHERE tabulky, která se aktualizuje. Když se podíváte na příkaz UPDATE v sekci Problém, spojení mezi tabulkami EMP a NEW_SAL založené na sloupci DEPTNO je provedeno a vrací řádky pro klauzuli SET příkazu UPDATE. Pro zaměstnance DEPTNO 10 se vrací platné hodnoty záznamů, protože ve sloupci DEPTNO tabulky NEW_SAL jsou odpovídající hodnoty záznamů. Ale co zaměstnanci jiných oddělení? V tabulce NEW_SAL žádná jiná oddělení nejsou, takže hodnota záznamů ve sloupci SAL a COMM pro tyto zaměstnance bude prázdná. Pokud neomezíte počet řádků výsledné množiny prostřednictvím klauzulí LIMIT

nebo TOP, či jakýmkoliv jiným mechanismem, který podporuje váš databázový systém, jedinou možností pro omezení počtu řádků získaných z tabulky je v SQL klauzule WHERE. Abyste aktualizaci provedli správným způsobem, použijte klauzuli WHERE na tabulku, kterou aktualizujete spolu s klauzulí WHERE v korelovaném vnořeném dotazu.

DB2 a MySQL

Abyste zajistili, že nedojde k aktualizaci každého řádku tabulky EMP, nezapomeňte zahrnout korelovaný vnořený dotaz do klauzule WHERE příkazu UPDATE. Vytvoření spojení (korelovaný vnořený dotaz) v klauzuli SET nestačí. Když použijete klauzuli WHERE v příkazu UPDATE, zajistíte, že dojde pouze k aktualizaci takových řádků tabulky EMP, které mají stejnou hodnotu záznamu ve sloupci DEPTNO, jako řádky tabulky NEW_SAL. To platí pro všechny databázové systémy.

Oracle

V řešení pro Oracle, které využívá pro aktualizaci pohled spolu se spojením, používáte spojení založená na podmínce rovnosti (equi-join) abyste určili, jaké řádky se budou aktualizovat. Když položíte dotaz samostatně, můžete potvrdit, které to budou. Abyste byli schopni tento typ aktualizace správně používat, musíte nejprve pochopit koncept zachování klíče. Sloupec DEPTNO tabulky NEW_SAL je jejím primárním klíčem, takže hodnoty záznamů v něm jsou v rámci této tabulky jedinečné. Když ale spojujete tabulky EMP a NEW_SAL, NEW_SAL.DEPTNO již nemá ve výsledné množině jedinečné hodnoty záznamů:

```
select e.empno, e.deptno e_dept, ns.sal, ns.deptno ns_deptno
  from emp e, new_sal ns
 where e.deptno = ns.deptno

EMPNO     E_DEPT        SAL  NS_DEPTNO
----- ---------- ---------- ----------
 7782         10       4000         10
 7839         10       4000         10
 7934         10       4000         10
```

Abyste umožnili Oraclu aktualizovat toto spojení, musí být v jedné z tabulek zachován klíč, takže pokud nejsou hodnoty daného sloupce, který je klíčem, jedinečné v rámci výsledné množiny, musí být jedinečné alespoň v rámci tabulky, z níž pocházejí. V tomto případě je primárním klíčem tabulky NEW_SAL sloupec DEPTNO, takže hodnoty jeho záznamů jsou v této tabulce jedinečné. Protože jsou v této tabulce jedinečné, mohou se objevit ve výsledné množině vícekrát a současně se zachovat jako klíč, a zajistit tak úspěšný průběh a dokončení aktualizace.

PostgreSQL a SQL Server

Syntaxe obou databázových systémů je mírně odlišná, ale princip řešení je stejný. Možnost provést spojení přímo v příkazu UPDATE je velmi výhodná. Jakmile jednou určíte, která tabulka se bude aktualizovat (tabulka, jež následuje za klíčovým slovem UPDATE), nemůže být sporu o tom, ve které tabulce dojde k modifikaci řádků. Navíc, protože používáte pro aktualizaci spojení (klauzule WHERE), můžete se vyhnout některým nástrahám při zápisu korelovaného vnořeného dotazu; konkrétně řečeno, když v něm zapomenete provést spojení, je jasné, že to bude problém.

Slučování záznamů

Problém

Chcete vkládat do tabulky nové záznamy, aktualizovat záznamy stávající nebo záznamy mazat, v závislosti na tom, zda řádky existují (pokud řádek v tabulce existuje, chcete řádek aktualizovat, pokud ne, chcete ho vložit, a pokud po aktualizaci nesplňuje nějakou podmínku, chcete ho smazat). Například chcete upravit tabulku EMP_COMMISSION následujícím způsobem:

- Pokud nějaký zaměstnanec z tabulky EMP_COMMISSION existuje také v tabulce EMP, chcete aktualizovat výši jeho provize ze zisku na hodnotu 1 000.
- Každého zaměstnance, jehož provize ze zisku jste aktualizovali na hodnotu 1 000, a jehož plat je menší než 2 000, chcete z tabulky EMP_COMMISSION vymazat.
- V ostatních případech chcete vložit do tabulky EMP_COMMISSION hodnoty záznamů ve sloupcích EMPNO, ENAME a DEPTNO tabulky EMP.

V podstatě jde o to, že chcete provést buď příkaz UPDATE, nebo INSERT, v závislosti na tom, zda stejný řádek existuje v obou tabulkách či nikoliv. Následně chcete provést pro daný řádek příkaz DELETE v případě, že hodnota výše provize ze zisku je po provedení příkazu UPDATE příliš vysoká s ohledem na hodnotu záznamu ve sloupci SAL tohoto řádku.

Tabulky EMP a EMP_COMMISSION obsahují tyto řádky:

```
select deptno,empno,ename,comm
  from emp
 order by 1

DEPTNO      EMPNO ENAME        COMM
------ ---------- ------ ----------
    10       7782  CLARK
    10       7839  KING
    10       7934  MILLER
    20       7369  SMITH
    20       7876  ADAMS
    20       7902  FORD
    20       7788  SCOTT
    20       7566  JONES
    30       7499  ALLEN         300
    30       7698  BLAKE
    30       7654  MARTIN       1400
    30       7900  JAMES
    30       7844  TURNER          0
    30       7521  WARD          500

select deptno,empno,ename,comm
  from emp_commission
 order by 1

    DEPTNO      EMPNO ENAME            COMM
---------- ---------- ---------- ----------
        10       7782 CLARK
        10       7839 KING
        10       7934 MILLER
```

Řešení

Jediným systémem, který v současné době disponuje příkazem navrženým za účelem řešení této úlohy, je Oracle. Tímto příkazem je příkaz MERGE, který provádí buď UPDATE, nebo INSERT, podle toho, jak je potřeba. Například:

```
1  merge into emp_commission ec
2  using (select * from emp) emp
3     on (ec.empno=emp.empno)
4   when matched then
5        update set ec.comm = 1000
6        delete where (sal < 2000)
7   when not matched then
8        insert (ec.empno,ec.ename,ec.deptno,ec.comm)
9        values (emp.empno,emp.ename,emp.deptno,emp.comm)
```

Rozbor řešení

Spojení na třetím řádku tohoto řešení určuje, které řádky již v tabulce existují a budou aktualizovány. Spojení je mezi tabulkou EMP_COMMISSION (která má alias EC) a vnořeným dotazem (jenž má alias EMP). Pokud je spojení platné, považují se řádky za shodné a provede se příkaz UPDATE v klauzuli WHEN MATCHED. Jinak není nalezen žádný shodný řádek a provede se příkaz INSERT v klauzuli WHEN NOT MATCHED. Takže se do tabulky EMP_COMMISSION vloží řádky tabulky EMP, jejichž hodnota záznamu ve sloupci EMPNO neodpovídá hodnotě záznamu ve sloupci EMPNO řádků tabulky EMP_COMMISSION. Ze všech zaměstnanců tabulky EMP by se měla hodnota výše provize ze zisku aktualizovat v tabulce EMP_COMMISSION pouze u těch, kteří náleží do DEPTNO 10, zatímco zbytek zaměstnanců z tabulky EMP by se měl do tabulky EMP_COMMISSION pouze vložit. Navíc se z tabulky EMP_COMMISSION smaže řádek zaměstnance MILLER, který je sice v DEPTNO 10, ale jehož plat je menší než 2000.

Smazání všech záznamů tabulky

Problém

Chcete z tabulky vymazat všechny záznamy.

Řešení

Když chcete z tabulky mazat záznamy, použijte příkaz DELETE. Když chcete například smazat všechny záznamy tabulky EMP, postupujte takto:

```
delete from emp
```

Rozbor řešení

Když použijete příkaz DELETE bez klauzule WHERE, smažete všechny řádky dané tabulky.

Mazání specifických záznamů

Problém

Chcete z tabulky smazat záznamy, které splňují určitá kritéria.

Řešení

Použijte příkaz DELETE spolu s klauzulí WHERE, jejímž prostřednictvím specifikujete řádky, které chcete smazat. Když například chcete smazat všechny zaměstnance pro DEPTNO 10, postupujte takto:

```
delete from emp where deptno = 10
```

Rozbor řešení

Když použijete příkaz DELETE s klauzulí WHERE, můžete místo všech řádků tabulky mazat pouze jejich podmnožiny.

Smazání jednoho záznamu

Problém

Chcete z tabulky smazat pouze jeden záznam.

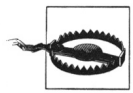
Řešení

Jde o zvláštní případ mazání specifických záznamů. Nejdůležitější je, aby kritérium, které rozhoduje o tom, jaký záznam se smaže, bylo co nejpřesnější a došlo tedy ke smazání opravdu pouze jednoho záznamu. Často budete potřebovat mazat záznamy podle primárního klíče. Když například chcete smazat zaměstnance CLARK (EMPNO 7782), postupujte následujícím způsobem:

```
delete from emp where empno = 7782
```

Rozbor řešení

Při mazání je vždy nejdůležitější přesně určit, které řádky chcete smazat a výsledek příkazu DELETE vždy odpovídá způsobu, jakým definujete podmínky v klauzuli WHERE. Když vynecháte klauzuli WHERE, smažete celou tabulku. Když správně definujete podmínky v klauzuli WHERE, můžete mazat množiny záznamů, nebo pouze jediný z nich. Když mažete jednotlivý záznam, obvykle ho identifikujete podle primárního klíče tabulky, nebo podle některé z jeho jedinečných hodnot.



V případě, že je kritérium pro smazání záznamu nastaveno podle primárního nebo jedinečného klíče, můžete si být jistí výsledkem (protože váš databázový systém nedovolí, aby dva řádky obsahovaly stejné hodnoty ve sloupci primárního klíče, nebo v jiném sloupci s jedinečnými hodnotami záznamů). Pokud máte toto kritérium nastaveno jiným způsobem, měli byste se nejprve ujistit, že se opravdu smažou pouze ty řádky, které chcete smazat.

Mazání záznamů narušujících referenční integritu

Problém

Potřebujete z tabulky odstranit záznamy, které odkazují k neexistujícím záznamům v jiné tabulce. Příklad: někteří zaměstnanci jsou zařazeni do oddělení, jež neexistují. Chcete tyto zaměstnance z tabulky odstranit.

Řešení

Pro otestování platnosti hodnot záznamů ve sloupci DEPTNO použijte výraz NOT EXISTS spolu s vnořeným dotazem:

```
delete from emp
 where not exists (
   select * from dept
    where dept.deptno = emp.deptno
)
```

Alternativně můžete v dotazu využít výraz NOT IN:

```
delete from emp
where deptno not in (select deptno from dept)
```

Rozbor řešení

Výsledek mazání skutečně vždy závisí na tom, zda správně určíte, co chcete mazat: nejdůležitější je napsat klauzuli WHERE takovým způsobem, aby správně určila záznamy, které se mají smazat.

Řešení s výrazem NOT EXISTS využívá korelovaný vnořený dotaz pro ověření existence takové hodnoty záznamu ve sloupci DEPTNO tabulky DEPT, která odpovídá dané hodnotě záznamu ve sloupci DEPTNO tabulky EMP. Pokud taková hodnota záznamu existuje, záznam v tabulce EMP se zachová. Pokud ne, smaže se. Tímto způsobem se vyhodnotí každý záznam tabulky EMP.

Řešení s klauzulí IN využívá vnořený dotaz pro vrácení seznamu platných čísel oddělení. Následně se podle tohoto seznamu zkontrolují všechny hodnoty záznamů ve sloupci DEPTNO tabulky EMP. Když je nalezen takový záznam tabulky EMP, pro který je hodnota záznamu ve sloupci DEPTNO jiná, než odpovídající hodnota záznamu na seznamu, smaže se.

Mazání duplikátů

Problém

Chcete z tabulky odstranit duplikáty. Prohlédněte si tuto tabulku:

```
create table dupes (id integer, name varchar(10))
insert into dupes values (1, 'NAPOLEON')
insert into dupes values (2, 'DYNAMITE')
insert into dupes values (3, 'DYNAMITE')
insert into dupes values (4, 'SHE SELLS')
insert into dupes values (5, 'SEA SHELLS')
insert into dupes values (6, 'SEA SHELLS')
insert into dupes values (7, 'SEA SHELLS')


select * from dupes order by 1

         ID NAME
----------- ----------
          1 NAPOLEON
          2 DYNAMITE
          3 DYNAMITE
          4 SHE SELLS
          5 SEA SHELLS
          6 SEA SHELLS
          7 SEA SHELLS
```

Z každé skupiny duplikovaných jmen, jako například SEA SHELLS, chcete v tabulce zachovat pouze jedno z nich a ostatní smazat. V případě SEA SHELLS je vám jedno, kolik těchto záznamů smažete, ale nakonec chcete mít pouze jeden záznam jména SEA SHELLS.

Řešení

Použijte vnořený dotaz spolu s agregační funkcí, jako je například funkce MIN, abyste rozhodli, které záznamy v tabulce zachováte (v tomto případě záznamy s nejmenší hodnotou ve sloupci ID):

```
1  delete from dupes
2   where id not in ( select min(id)
3                       from dupes
4                      group by name )
```

Rozbor řešení

Když chcete mazat duplikáty, je třeba je nejprve nějak definovat. V rámci tohoto příkladu se za duplikáty považují takové řádky, které mají stejnou hodnotu záznamu ve sloupci NAME. Když je jasná jejich definice, je potřeba najít nějaký jiný sloupec, který vám pomůže určit záznamy, jež se mají zachovat. Nejlepší je, když je tento sloupec zároveň primárním klíčem tabulky. V tomto příkladu se využívá sloupec ID, protože žádný záznam nemá v tomto sloupci stejnou hodnotu.

Klíčem k řešení tohoto problému je setřídění záznamů, které jsou duplikáty (mají stejnou hodnotu ve sloupci NAME), a následné použití agregační funkce pro vybrání pouze jediné klíčové hodnoty záznamu, jenž se má zachovat. Vnořený dotaz v sekci Řešení vrací nejmenší hodnotu záznamu ve sloupci ID každého záznamu označující řádek, který chcete zachovat:

```
select min(id)
  from dupes
 group by name

    MIN(ID)
-----------
          2
          1
          5
          4
```

Příkaz DELETE pak smaže všechny řádky tabulky, které mají jinou hodnotu záznamu ve sloupci ID (v tomto případě ID 3, 6 a 7). Pokud máte potíže pochopit, jak tento mechanismus pracuje, zavolejte vnořený dotaz se sloupcem NAME v příkazu SELECT:

```
select name, min(id)
  from dupes
 group by name

NAME          MIN(ID)
---------- ----------
DYNAMITE           2
NAPOLEON           1
SEA SHELLS         5
SHE SELLS          4
```

Řádky, které vrací vnořený dotaz, jsou řádky, jež se mají zachovat. Výraz NOT IN v příkazu DELETE způsobí, že se všechny ostatní záznamy smažou.

Mazání záznamů, na které se odkazují záznamy jiné tabulky

Problém

Chcete smazat z jedné tabulky záznamy, ke kterým se odkazují záznamy z jiné tabulky. Prohlédněte si tuto tabulku, DEPT_ACCIDENTS, která obsahuje jeden řádek pro každou nehodu, jež se stala v továrně při výrobě. V každém řádku je hodnota označující číslo oddělení, ve kterém k nehodě došlo, a typ nehody.

```
create table dept_accidents
( deptno         integer,
  accident_name  varchar(20) )

insert into dept_accidents values (10,'BROKEN FOOT')
insert into dept_accidents values (10,'FLESH WOUND')
insert into dept_accidents values (20,'FIRE')
insert into dept_accidents values (20,'FIRE')
insert into dept_accidents values (20,'FLOOD')
insert into dept_accidents values (30,'BRUISED GLUTE')

select * from dept_accidents

    DEPTNO ACCIDENT_NAME
---------- --------------------
        10 BROKEN FOOT
        10 FLESH WOUND
        20 FIRE
        20 FIRE
        20 FLOOD
        30 BRUISED GLUTE
```

Chcete smazat z tabulky EMP záznamy zaměstnanců pracujících pro oddělení, ve kterých se staly tři nebo více nehod.

Řešení

Pro nalezení oddělení, ve kterých se staly tři nebo více nehod, použijte agregační funkci COUNT. Pak smažte všechny zaměstnance pracující v těchto odděleních:

```
1 delete from emp
2  where deptno in ( select deptno
3                      from dept_accidents
4                     group by deptno
5                    having count(*) >= 3 )
```

Rozbor řešení

Vnořený dotaz určí, ve kterých odděleních se staly tři nebo více nehod:

```
select deptno
  from dept_accidents
 group by deptno
having count(*) >= 3
```

```
    DEPTNO
----------
        20
```

Příkaz DELETE pak smaže všechny zaměstnance pracující pro oddělení, která vrátil vnořený dotaz (v tomto případě pouze zaměstnance oddělení 20).