Aktivní databáze

Michal Ficek m.ficek@volny.cz
Tomáš Pop tomas.pop@seznam.cz
http://www.tydlin.cz

Obsah

- úvod
  - co jsou aktivní databáze, terminologie
- problémy aktivních databází
- vybrané modely
  - Starburst
  - Oracle
  - DB2
  - Chimera
  - SQL Server
- taxonomie aktivních databází
- literatura a odkazy

Úvod

- databáze
  - obraz skutečnosti
  - reálná data a na ně kladené požadavky
- omezení dat
  - integritní omezení
    - jednoduché, rychlé
    - ne vždy postačuje

Aktivní pravidla

- *aktivní pravidla (active rules)*
  - pro vyhodnocení složitých podmínek kladených na data (tzv. business rules)
  - kontrola na databázové úrovni
  - usnadnění práce – auditovatelnost, bezpečnost
- *triggery (triggers)*
  - v překladu „spoušť“, „kohoutek“
  - jiný název pro aktivní pravidla
  - v dalším textu mu budeme často dávat přednost

AKTIVNÍ PRAVIDLA = TRIGGERY

Zaměření referátu

- syntax
  - zápis triggeru v daném DB systému
- sémantika
  - kdy se pustí
  - jak proběhne
  - jak se navzájem volají
  - nekonečné cykly
  - …
- vybrané modely aktivních pravidel
  - historicky významné nebo prakticky používané
  - zajímavě implementované

Problémy s triggery

- standardizace
  - není
  - snaha by byla
    - od 80. let
    - v SQL-92 stále nejsou
- proprietární řešení výrobců DB systémů
  - rozdíly v syntaxi i sémantice
  - vazba aplikace na konkrétního výrobce
- technické problémy
  - nekonečné vzájemné volání triggerů (retriggering)
    - několik možných řešení
    - používají se všechny (od omezení po pštrosí hlavu)

Starburst

Starburst

- Starburst Active Rule System
  - DDL rozšíření projektu Starburst
  - jednoduchá syntaxe i sémantika

- *událost – podmínka – akce (Event-Condition-Action, ECA)*
  - princip fungování triggerů
  - „Když nastane událost a je splněna podmínka, vykonej akci“
    - zní to jednoduše
    - má to spousty háčků

událost – podmínka – akce

- událost
  - **INSERT, DELETE, UPDATE**
    - manipulační primitiva
- podmínka
  - libovolná SQL podmínka
- akce
  - libovolný SQL příkaz
    - **SELECT, INSERT**
    - **DELETE, UPDATE**
  - příkaz řízení transakce
    - **ROLLBACK WORK**

Syntaxe aktivních pravidel Starburst

```
CREATE RULE <jméno pravidla> ON <jméno
  tabulky>
WHEN <události>
[ IF <SQL podmínka> ]
THEN <SQL príkazy>
[ PRECEDES <seznam jmen pravidel> ]
[ FOLLOWS <seznam jmen pravidel> ]
```

Příklad vytvoření aktivního pravidla

```
CREATE RULE platy2 ON zamestnanci
WHEN INSERTED, DELETED, UPDATED
IF (SELECT avg(plat) FROM zamestnanci) > 100
THEN UPDATE zamestnanci SET plat = 0.9 *
  plat
FOLLOWS platy1
```

Doplnění syntaxe

- unikátní jméno pravidla
  - asociováno se specifickou tabulkou – cílem pravidla
- použití
  - pravidlo sleduje více událostí
  - stejná událost více pravidly
- příkazy PRECEDES a FOLLOW
  - použít lze pouze v době vzniku pravidla
  - určují *částečné uspořádání (partial order)*
  - vztah uspořádání musí být acyklický
- sdružování pravidel do skupin
- aktivace a deaktivace pravidel

Sémantika – terminologie

Pravidlo je:

- *spuštěné (triggered)*
  - pokud nastane jím sledovaná událost
  - ostatní pravidla nespuštěná
  - spuštěné neznamená „vykonává se akce“, „vyhodnocuje se podmínka“
- *bráno v úvahu (considered)*
  - podmínka pravidla je vyhodnocena
- *provedeno (executed)*
  - příslušná akce je vykonána
  - vykonání je odložené
    - ve chvíli provedení příkazu **COMMIT WORK**
    - explicitně voláním **PROCESS RULE**

Sémantika – spuštění pravidel

- pravidlo je spuštěno
  - poté co nastane událost
  - pokud událost sleduje více pravidel, pak tvoří *konfliktní množinu (conflict set)*
- algoritmus vyhodnocení pravidel

  DOKUD *je množina spuštěných pravidel M neprázdná* {
  vyber pravidlo R s nejvyšší prioritou z M a označ jako nespuštěné
  POKUD *je podmínka R splněna* {
  proveď akci R
  }
  }
- jednoznačně opakovatelné
  - díky úplnému uspořádání

Sémantika - cykly

- mohou nastat
  - trigger T1 způsobí akci která znovu spustí trigger T1…
- *konečný stav (quiescent state)*
  - je určen prázdnou konfliktní množinou
- zajistit konečnost konečnost vyvolávání triggerů je na autorovi pravidel ☠

Sémantika – detaily

- *stavové přechody (state transitions)*
  - transformace jednoho stavu databáze do druhého
  - vykonání SQL příkazů transakcemi
- *vloženo, vymazáno, změněno (inserted, deleted, updated)*
  - množiny popisující přechody
  - plní se n-ticemi změněnými SQL příkazy
  - představují všechny změny, které povedou ze stavu S1 do stavu S2
- *čistý efekt (net effect)*
  - znamená, že se každá n-tice změněných dat objeví právě v jedné z množin vloženo, vymazáno, změněno
  - např. vložení a následné vymazání n-tice má nulový efekt
  - insert a následný update má čistý efekt insert nové hodnoty

Sémantika - detaily

- probíhá algoritmus vyhodnocování pravidel
- pravidlo je spuštěno
  - pokud je množina operací pravidlem sledovaných neprázdná
    - vztaženo k aktuálnímu přechodovému stavu
    - přechodový stav se mění s vykonáváním pravidel
  - akce spuštěného pravidel vyústí ve změnu množin
  - znovu se utvoří konfliktní množina
- končí se prázdnou konfliktní množinou
  - konečným stavem

Oracle

Oracle – triggery

- podpora se vyvíjí
  - v dřívějších verzích četná omezení
    - událost sleduje pouze jeden trigger
    - nebylo možné ovlivňovat pořadí spouštění

Dva typy triggerů:

- *řádkové (row-level)*
  - událostí je změna každého jednotlivého řádku
- *příkazové (statement-level)*
  - událostí je příkaz provádějící změny

Syntaxe Oracle triggerů

```
CREATE OR REPLACE TRIGGER <jméno triggeru>
[FOLLOWS <schéma.jméno triggeru>]
[<ENABLE | DISABLE>]
{BEFORE | AFTER} <událost> [OR <událost> [ OR <událost> ]]
[OF <seznam sloupců>]
ON <jméno tabulky>
REFERENCING NEW AS <název> OLD AS <název> PARENT AS
   <název>
[[ FOR EACH ROW ] WHEN (<podmínka>) ]]
DECLARE
 <definice proměnných>
BEGIN
  <PL/SQL kód triggeru>
EXCEPTION
  <vyjímky>
END <jméno triggeru>;
```

Syntaxe Oracle triggerů - detaily

- *událost*
  - manipulační primitiva
    - **INSERT, DELETE, UPDATE**
- *podmínka*
  - libovolná SQL podmínka
  - pouze pro řádkové triggery
- *akce*
  - libovolný PL/SQL kód
    - velmi silné!
    - nesmí obsahovat DDL příkazy

Syntaxe Oracle triggerů - detaily

- *predikáty*
  - dostupné jsou **INSERTING, DELETING, UPDATING**

- *reference*
  - staré a nové hodnoty
    - **OLD, NEW**
  - lze je přejmenovat
  - pouze pro řádkové triggery

Sémanitka Oracle triggerů

- spouštění
  - probíhá okamžitě při události
  - nelze spustit explicitně (uživatelským příkazem)
- vnořené spouštění triggerů
  - činností triggeru dojde ke spuštění jiného triggeru nebo sebe sama
  - probíhající se přeruší, uloží se jeho kontext a provádí se jiný
  - omezena maximální hloubka zanoření
    - 32, poté je vyvolána vyjímka
- vyjímky nebo chyby
  - všechny změny původní SQL operace a následné změny provedené triggery odrolovány
  - Oracle podporuje částečný rollback (oproti transakčnímu)

Sémanitka Oracle triggerů

- spouštění
  - probíhá okamžitě při události
  - nelze spustit explicitně (uživatelským příkazem)
- vnořené spouštění triggerů
  - činností triggeru dojde ke spuštění jiného triggeru nebo sebe sama
  - probíhající se přeruší, uloží se jeho kontext a provádí se jiný
  - omezena maximální hloubka zanoření
    - 32, poté je vyvolána vyjímka
- vyjímky nebo chyby
  - všechny změny původní SQL operace a následné změny provedené triggery odrolovány
  - Oracle podporuje částečný rollback (oproti transakčnímu)

Sémanitka Oracle triggerů

- řazení (klauzule FOLLOWS)
  - podporováno nedlouho
  - ve verzi 11.1
  - zajištěno předcházení triggerů
  - úplné uspořádání určené vznikem triggeru
    - novější bude spuštěn dříve

Sémanitka Oracle triggerů

- algoritmus prokládání SQL příkazu triggery během jeho vykonávání

  - *Proveď **statement-level before triggery***
  - *Pro každý řádek v tabulce*
    - *Proveď **row-level before triggery***
    - *proveď změnu řádku (a kontroly integrity)*
    - *Proveď **row-level after triggery***
  - *Proveď kontrou integrity na úrovni příkazu*
  - *Proveď **statement-level after triggery***

Triggery v DB2

DB2

- triggery pro DB2 Common Server
  - navrženy IBM, vychází ze zkušenosti se systémem Starburst
  - zaměřeno na
    - jednoduchost syntaxe
    - jednoznačnost sémantiky
    - kontrola integritních omezení
- dva typy triggerů
  - řádkové
  - sloupcové
- každý trigger monitoruje pouze jednu událost
  - ale více triggerů může monitorovat stejnou událost
  - hlavní změna oproti Starburst

Syntaxe DB2 triggerů

```
CREATE TRIGGER <jméno triggeru>
[ NO CASCADE BEFORE | AFTER ]
{ INSERT | DELETE | UPDATE [OF COLUMN <jméno
  sloupce>,... ] }
ON <jméno tabulky>
[ FOR EACH ROW MODE DB2SQL | FOR EACH STATEMENT ]
[ REFERENCING [OLD AS  <označení původních hodnot> ]
  [ NEW AS <označení nových hodnot> ]
[ OLD_TABLE AS  <označení původní tabulky> ]
  [ NEW_TABLE AS <označení nové tabulky > ] ]
WHEN  <podmínka>
BEGIN ATOMIC
<povolené SQL příkazy viz dále>
END
```

Syntaxe DB2 triggerů - detaily

- *událost*
  - manipulační primitivum
    - **INSERT, DELETE, UPDATE**
- *podmínka*
  - libovolná SQL podmínka
- *akce*
  - !! závisí na konkrétním typu triggeru
  - SQL příkaz
    - ne DDL
    - ne transakční
- *přechodové hodnoty*
  - pro řádkové triggery
    - **OLD, NEW**
  - pro příkazové triggery
    - **OLD_TABLE, NEW_TABLE**

Sémantika DB2 triggerů

- BEFORE triggery
  - ke kontrole chyb, integritních omezení
  - podmínka je vyhodnocena ke stavu databáze před událostí
  - nemohou měnit databázi použitím **UPDATE, INSERT, DELETE** → nemohou aktivovat jiné triggery
  - mohou vyvolávat chyby
- AFTER triggery
  - k implementaci aplikační logiky
  - spuštěny po změně dat
  - stav tabulky T před událostí lze zjistit
    - (T **MINUS NEW_TABLE**) **UNION OLD_TABLE**

Sémantika DB2 triggerů

- více triggerů může monitorovat stejnou událost
  - úplné uspořádání
  - klíčem je čas vytvoření
- vykonávání řádkové a příkazové triggerů se prolíná
  - na rozdíl od Oracle (pořadí typů triggerů určeno)
  - umožňují akce řádkových triggerů, které mají více příkazů

Sémantika DB2 triggerů - detaily

- algoritmus pro situaci
  „příkaz **P** v akci **A** způsobí událost **U**“

  - *Pozastav akci **A**, ulož její kontext*
  - *Spočti hodnoty **NEW** a **OLD** vzhledem k události **U***
  - *Proveď všecny **before-triggery** pro událost **U** (mohou měnit **NEW**)*
  - *Propaguj obsah **NEW** do databáze, stav DB odpovídá stavu po události **U***
  - *Proveď všechny **after-triggery** pro událost **U**. Obsahují-li akci **B** která aktivuje jiný trigger, volej rekurzivně tuto proceduju pro akci **B**.*
  - *Vyzvedni akci **A** ze zásobníku a pokračuj*

Sémantika DB2 triggerů – ošetření chyb

- chybu způsobí příkaz
  - měl teprve trigger spustit
  - odroluje se do stavu před prováděním příkazu
- chyba při propagaci **NEW** do databáze
  - bod 4 předchozího algoritmu poruší
    - referenční omezení
    - integritní omezení
    - omezení pohledů
  - nastává řada *kompenzačních akcí*
    - např. nastavení hodnoty na NULL
    - mohou vyvolat další řadu triggerů

Sémantika DB2 triggerů – ošetření chyb

- chyba při propagaci **NEW** do databáze
  - *Aplikuj hodnoty **NEW**. Pro každé porušené integritní omezení naplánuj kompenzační akci $\mathbf{A_i}$, která je napraví*
    - *spočti hodnoty **OLD** a **NEW** relativně k akci $\mathbf{A_i}$*
    - *spusť **before-triggery** relativně k akci $\mathbf{A_i}$ (možná změna **NEW**)*
    - *propaguje nové hodnoty **NEW** do databáze*
    - *vlož všechny **after-triggery** do fronty uspaných triggerů*

DB2 triggery – poznámka

Podobné propojení kontroly integrity a spouštění triggerů bylo prosazeno do standardu ***SQL3***

Chimera

Chimera - terminologie

- objektově orientovaný jazyk
- *třídy (object class)*
  - základ definice dat
- *atributy třídy*
  - určují jeho stav
  - definovány konstruktory
    - **record**
    - **set**
    - **list**
- třídy hierarchicky uspořádány
- triggery jsou určovány pro třídu (*targetted triggers)*

Chimera – příklady objektů

```
define object class Zamestnanec
attributes
  Jméno: string,
  Plat: integer
end;

define object class Oddelení
attributes
  Jméno: string,
  Zamestnanci: set-of(Zamestnanec)
end;
```

Chimera – deklarativní výrazy

- *atomické termy*
  - proměnné, konstanty
  - X (proměnná), 5 (konstanta)

- *složené termy*
  - z atomických pomocí konstruktorů,
  - za pomocí funkcí Chimery (selektory, operátory, atributy)
  - X.jméno, set-of(Zaměstnanec)

Chimera – deklarativní výrazy

- *atomické formule*
  - predikátový symbol a seznam parametrů
  - integer(X)
    - zavádí proměnnou typu integer
  - X.jméno='John'
    - porovnání

- složené formule
  - konjunkce a disjunkce atomických formulí
  - Zaměstnanec(X), Oddělení(Y)
  - Y.jméno=hračky, not(X in Y.Zaměstnanci)

Chimera – procedurální výrazy

- *create()*
  - **create(**Zaměstnanci**, ['**John**',** 4000**])**
- *select()*
  - **select(**X **where** Zaměstnanci**(**X**),**X**.**jméno**='**John**')**
- *delete()*
  - **delete(**Zaměstnanci**,** X**)**
- *modify()*
  - **modify(**Zaměstnanci**.**plat**,** X**,** X**.**plat * 1.01**)**

Chimera – triggery

- nejmenší jednotky, kterých se může trigger zachytit jsou

  transakční řádky (transaction lines)

  – zřetězení procedurálních výrazů do řádek
  – oddělena čárkou, ukončena středníkem
  – platnost proměnných je v rámci jedné řádky

Chimera – syntaxe triggerů

```
define <volba> trigger <jméno triggeru>
[ for <jméno třídy> ]
events <události>
condition <formule>
actions <proceduralni výraz>
[ { before | after } <jména triggerů>]
end
```

Chimera – syntaxe triggerů

- *volba*
  - consumption-opt
    - event-consuming
    - event-preserving
  - execution-opt
    - deferred
    - immediate
  - standardně nastaveno deferred a consuming
- *události*
  - **create, display, modify, deisplay, generalize, specialize**

Chimera – sémantika triggerů

- *odložené (deffered) triggery*
  - volány příkazy **commit** nebo **savepoint**
  - okamžité triggery spuštěné akcí odloženého triggeru se zařadí do fronty s odloženými triggery
- *okamžité (immediate) triggery*
  - volány po dokončení transakční řádky
- pořadí
  - **before** a **after** určují částečné uspořádání
  - úplné uspořádání určeno systémem
    - čas vytvoření triggeru

Chimera – sémantika více triggerů

- množina spuštěných triggerů
  - stejně jako u Starburstu
- výběr z množiny

DOKUD *je množina spuštěných neprázdná* {
  vyber jeden trigger nejvyšší v pořadí
  POKUD *je podmínka splněna* {
    vykonej akci triggeru
  }
}

Triggery v SQL Server

SQL Server, triggery

- DML triggery
  - kontrola složitých omezení na data
  - dodatečná validace dat
  - změny v jiné tabulce
  - automatické zaznamenávání auditovatelných událostí
- DDL triggery
  - jako datový trigger, ale na systémových tabulkách
  - zejména jako prevence a záznam změn ve struktuře databáze
  - audity, bezpečnostní management
    - např. salámový útok změnami uložených procedur

Syntaxe DML triggerů

```
CREATE TRIGGER jméno_triggeru
ON {table|view}
[WITH ENCRYPTION]
{
{{FOR {AFTER|INSTEAD OF} {[INSERT] [,]
  [UPDATE] [,] [DELETE]}
AS
[{IF [UPDATE (sloupec)
[{AND|OR} UPDATE (sloupec)] ] ]
COLUMNS_UPDATE()]
sql_kód}}
```

Sémantika DML triggerů

- AFTER trigger
  - událost (provede se) → **INSTEAD OF** triggery → omezení dat (constraints) → **AFTER** trigger spustí
  - nelze použít na kontrolu vkládaných dat!

- INSTEAD OF trigger
  - událost (neprovede se) → akce **INSTEAD OF** triggeru → omezení dat
    - OK – akce **AFTER** triggeru
    - KO – **INSTEAD OF** akce je odrolována → **AFTER** trigger se neprovede

Sémantika DML triggerů

- INSTEAD OF trigger na tabulce
  - nespouští se rekurzivně (došlo by k vyvolání stejného triggeru)
  - akce triggeru se provede jako by tabulka již tento trigger neměla
  - následuje opět ošetření dat a vykonání **AFTER** triggerů
- INSTEAD OF trigger na pohledu
  - opět není rekurzivní
  - akce triggeru je vztažena k základním tabulkám pod pohledem
- INSTEAD OF trigger přínosy
  - umožňuje upravovatelné (*updatable*) pohledy
  - hodí se na správu pohledů vytvořených z více tabulek
  - umožní úspěch dávky příkazů při zamítnutí části dávky

Sémantika DML triggerů

- granularita sledování změn
  - na úrovni příkazů
  - neexistuje řádková alternativa triggerů
  - při změně více řádků mají logické tabulky také více řádků
- vnoření triggerů
  - nastaveno na maximální hloubku 32 triggerů

Sémantika DML triggerů

- přechodové hodnoty
  - uloženy v logických systémových tabulkách **INSERTED** a **DELETED**
  - změna (update) hodnot je implicitní
  - trigger na tabulce
    - logické tabulky mají stejnou strukturu jako tabulka na níž byl trigger definovaný
  - trigger na pohledu
    - logické hodnoty mají strukturu seznamu sloupců v příkazu **SELECT** definovaném na pohledu

Sémantika DML triggerů - rekurze

- nepřímá rekurze
  - T1 spustí TR1, upraví T2, spustí TR2, upraví T1…
    - omezeno maximální hloubkou zanoření
    - vypnutí - lze nastavit hloubka zanoření na 0
- přímá rekurze
  - T1 spustí TR1, změní T1, spustí TR1…
    - pouze při zapnutém **RECURSIVE_TRIGGERS** nastavení

DDL triggery

- reagují na události spojené se správou databáze
  - trigger může příslušet jedné nebo více akcím
  - příklady událostí:
    - CREATE_TABLE
    - DROP_VIEW
    - CREATE_SYNONYM
    - CREATE_FUNCTION
    - ALTER_FUNCTION
    - CREATE_PROCEDURE
    - DROP_PROCEDURE
    - DROP_TRIGGER
    - CREATE_EVENT_NOTIFICATION
    - DROP_EVENT_NOTIFICATION
    - ALTER_INDEX
    - …….. (je jich opravdu hodně)

DDL triggery

- serverové události je možné zachytit také
  - CREATE_LOGIN
  - ALTER_LOGIN
  - DROP_LOGIN
  - CREATE_HTTP_ENDPOINT
  - DROP_HTTP_ENDPOINT
  - GRANT_SERVER_ACCESS
  - DENY_SERVER_ACCESS
  - REVOKE_SERVER_ACCESS
  - CREATE_CERT
  - ALTER_CERT
  - DROP_CERT

  (to jsou všechny)
  - pozn.: není možné vytvořit trigger aktivní při serverové i databázové události

syntaxe DDL triggerů

```
CREATE TRIGGER jméno_triggeru
ON {ALL SERVER|DATABASE}
[WITH ENCRYPTION]
{
{{FOR |AFTER } {typ_události,…}
AS
sql_kód}}
```

DDL triggery

- k dispozici nejsou logické tabulky
  - nelze využít hodnot v **INSERTED** a **DELETED**

- lze použít funkci **EVENTDATA()**
  - vrací XML datový typ
  - obsah závisí na typu události

příklad XML dokumentu pro událost na databázové úrovni

```
<SQLInstance>
  <PostTime>datum_a_čas</PostTime>
  <SPID>SQLServer_proces_ID</SPID>
  <ComputerName>name</ComputerName>
  <DatabaseName>jméno</DatabaseName>
  <UserName>jméno</UserName>
  <LoginName>jméno</LoginName>
</SQLInstance>
```

Taxonomie konceptů aktivních databází

Základní pojmy

- *události (events)*
  - změna stavu databáze
  - časové události
  - externí, definované aplikací
- *podmínky (conditions)*
  - databázový predikát
  - databázový dotaz
- *akce (actions)*
  - libovolná manipulace s daty
    - transakční příkazy
    - pravidla zpracování
    - externí procedury

Vyhodnocení triggeru

- *okamžité (immediate)*
  - **před** událostí
  - **po** událost
  - **namísto** události
- *odložené (deferred)*
  - na konci transakce (odstartované příkazem **COMMIT WORK**)
  - po uživatelském příkazu
  - následkem uživatelského příkazu (např. **PROCESS RULES**)
- *oddělené (detached*)
  - v kontextu samostatné transakce vypuštěné z počáteční transakce poté, co nastala událost
    - možné kauzální závislosti počáteční a oddělené transakce

Vykonání akce

- *okamžité (immediate)*
  - následuje ihned po vyhodnocení podmínky
- *odložené (deferred)*
  - akce je odsunuta na konec transakce
  - akci vyvolá uživatelský příkaz
- *oddělené (detached)*
  - probíhá v kontextu samostatné transakce vypuštěné z počáteční transakce ihned po vyhodnocení podmínky
    - možné kauzální závislosti počáteční a oddělené transakce

Úroveň granularity sledování změn

- *na úrovni instancí (instance level)*
  - událostí je změna řádku tabulky
  - nebo změna jednotlivých objektů v třídách (v případě objektově orientovaných databází)
  - přechodové hodnoty postihují pouze jednu n-tici nebo objekt
    - proměnné **OLD** a **NEW**
- *na úrovni příkazů (statment level)*
  - událostí je příkaz provádějící manipulaci s daty
  - přechodové hodnoty jsou společné
    - shromážděné v tabulkách **INSERTED** a **DELETED**
    - explicitní změna dat – tabulky **OLD-UPDATED** a **NEW-UPDATED**
    - implicitní změna dat – tabulky **DELETED** a **INSERTED**

Aktivace více triggerů

- *konfliktní množina (conflict set)*
  - skupina aktivních pravidel, která mohou být aktivována současně
  - je zapotřebí metoda, která určí pořadí v konfliktní množině

- výběr dalšího pravidla
  - po každém vyhodnocení podmínky a případném vykonání příkazů nějakého triggeru
  - seznam všech aktivovaných triggerů a provádí se jeden po druhém

Výběr triggeru z konfliktní množiny

Priority

- *úplné uspořádání*
  - pravidlo je svázáno s číselnou prioritou
- *částečné uspořádání*
  - pravidla obsahují číselnou nebo relativní prioritu
  - soulad úplného systémového uspořádání a uživatelsky definované priority udržuje
    - systém
    - nedeterministický výběr z nejvyšších priorit
- *bez priorit*
  - systémově definované úplné uspořádání
  - nedeterminismus u všech pravidel

Další vlastnosti triggerů

- *opakovatelnost*
  - transakce1 = transakce2 → výsledek1 = výsledek2
  - stejná posloupnost vykonávaných příkazů
- *aktivovace* a *dektivovace*
  - velmi nebezpečné kvůli integritě databáze
  - jsou součástí autorizační politiky databáze
    - změny provádí administrátor
    - nebo pověřený uživatel (např. explicitním **GRANT PRIVILEGE**)
- *seskupování*

Odkazy a literatura

- http://www.unife.it/ing/informazione/sistemi-informativi/alle
- http://www.dbazine.com/sql/sql-articles/dewson1
- http://dev.mysql.com/doc/refman/5.0/en/triggers.html
- http://www.tar.hu/sqlbible/sqlbible0109.html
- http://www.cs.duke.edu/~junyang/courses/cps196.3-2002-fall/notes/sql.html