AKTIVNÍ DATABÁZE

- Triggery

Úvod

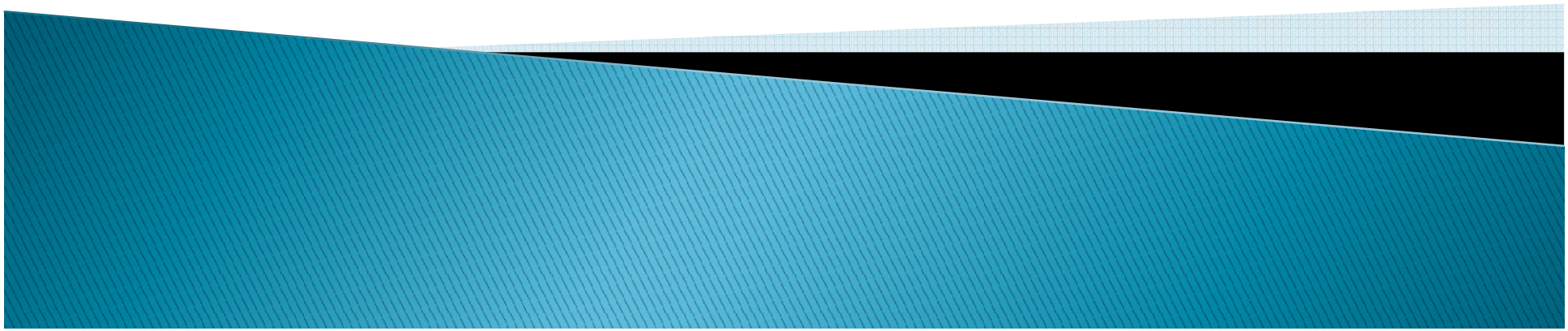
- databáze
  - obraz skutečnosti
  - reálná data a na ně kladené požadavky
- omezení dat
  - integritní omezení
    - jednoduché, rychlé
    - ne vždy postačuje

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

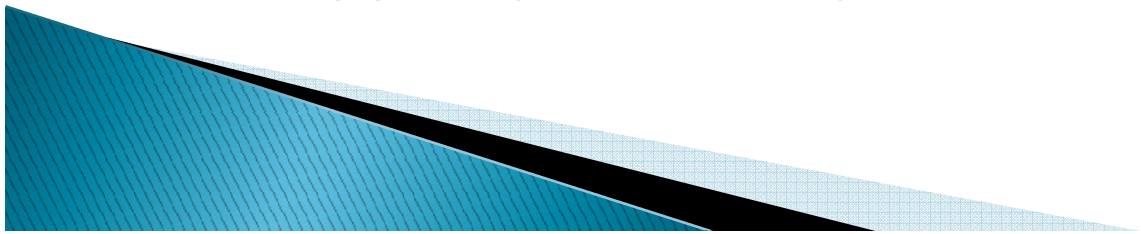
AKTIVNÍ PRAVIDLA = TRIGGERY

Úvod – historický vývoj



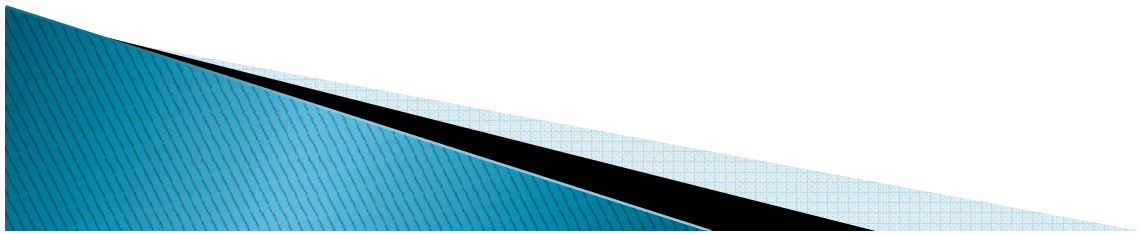
Jinak řečeno ……

- Trigger („spoušť“) je “procedura”, která se spustí při výskytu nějaké sledované události.
- V relačních databázích *trigger = aktivní pravidlo*
- Konec 80. let
  - první snahy o formální definici
- SQL92
  - triggery neobsahuje
  - nedostatky ve standardizačních dokumentech
- SQL1999
  - triggery již obsahuje



Starburst

- IBM, Almaden Research Center
  - Starburst Active Rule System
- Získalo popularitu
  - Jednoduchá syntaxe a sémantika
    - Množinově orientovaná
- Pravidla založena na ECA-paradigmatu (*Event-Condition-Action*)

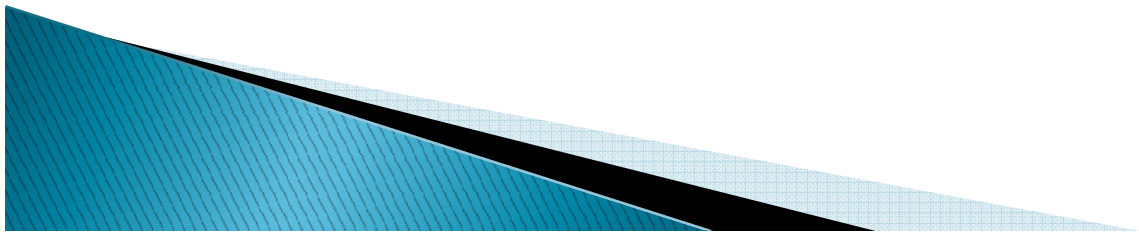


ECA-paradigma

- Událost (Event)
  - SQL-příkazy pro manipulaci s daty (*INSERT, DELETE, UPDATE*)
- Podmínka (Condition)
  - booleovský predikát nad stavem databáze, vyjádřen pomocí SQL
- Akce (Action)
  - provádí libovolné SQL dotazy (například *SELECT, INSERT, DELETE, UPDATE*)
  - navíc mohou obsahovat příkazy pro manipulaci s aktivními pravidly a transakční instrukci *ROLLBACK WORK*

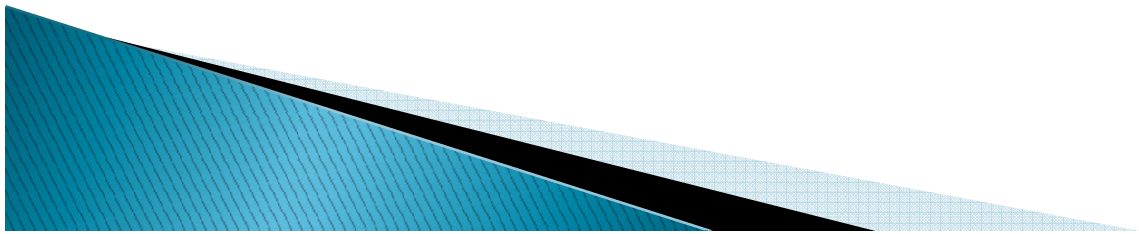
Sémantika aktivních pravidel

- Jednoduchá a intuitivní
- Když nastane **Událost**, pokud je splněna **Podmínka**, proveď **Akci**.
- Říkáme, že pravidlo je:
  - **spuštěno** (triggered) – pokud nastane příslušná **Událost**
  - **vyhodnoceno** (considered) – po vyhodnocení dané **Podmínky**
  - **vykonáno** (executed) – po provedení jeho **Akce**



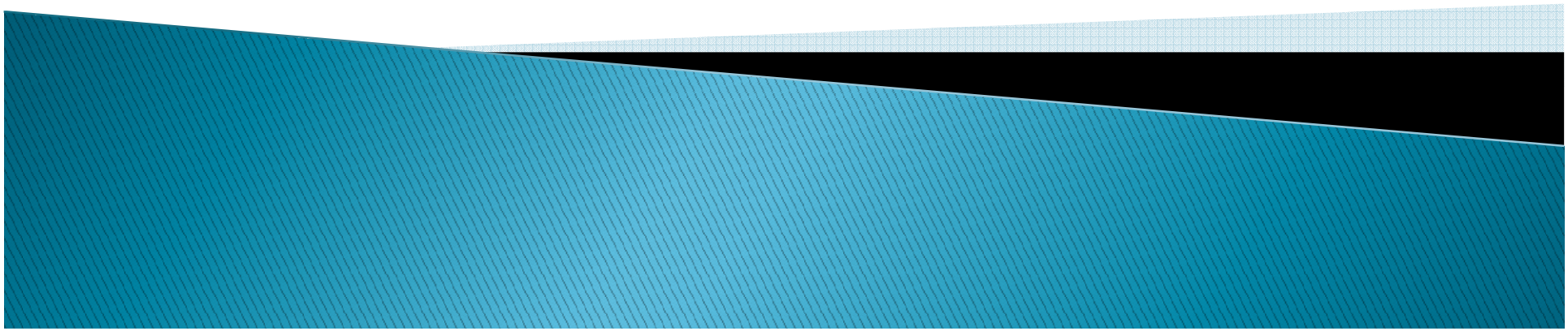
Vlastnosti aktivních pravidel (triggerů)

- Jsou přidané do schématu databáze a jsou sdílené všemi aplikacemi.
- Mohou být dynamicky aktivovány a deaktivovány každou transakcí.
- Mohou tvořit skupiny.
- Každé pravidlo ve Starburstu má jedinečné jméno a je spojeno s jednou určitou tabulkou, zvanou *rule's target*.
- Každé aktivní pravidlo může sledovat více *Událostí*, tzv. r*ule's triggering operations*.
- Jeden SQL příkaz může být sledován více pravidly.
  - Pořadí pravidel je určeno na základě jejich částečného uspořádání.



TRIGGER

Základní pojmy
Problémy
Použití



Základní pojmy

- 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

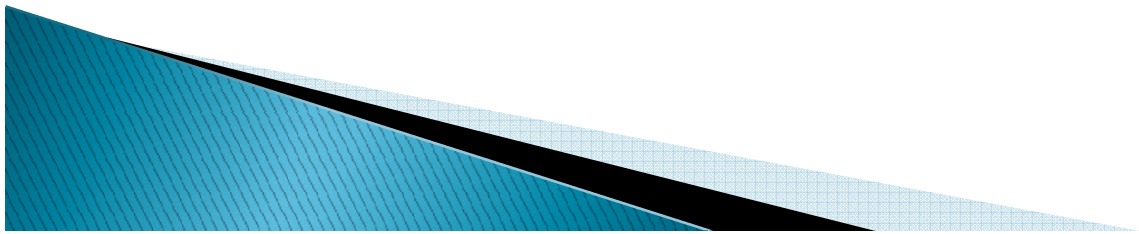
- standardizace
  - není
  - snaha by byla
    - od 80. let
    - v normě SQL-92 nejsou standardně uvedeny
- SQL 1999 – základní podmínky realizace, ale …

Problémy s triggery

- 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 všechna

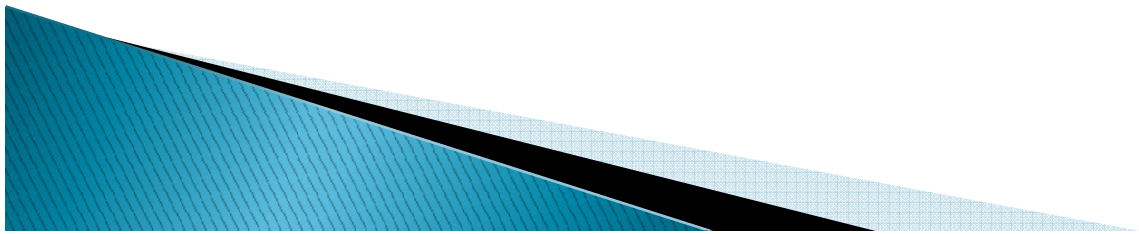
Spouště (triggery)

- Jedná se o PL/SQL objekty spouštěné vyvoláním příslušné události v DB
- Vyvolání může způsobit DML událost, DDL operace nebo speciální DB událost
- Vyvolat trigger můžeme před nebo po provedení operace
- Existuje také možnost vyvolání místo příslušné operace
- Je možné omezit vyvolání podmínkou



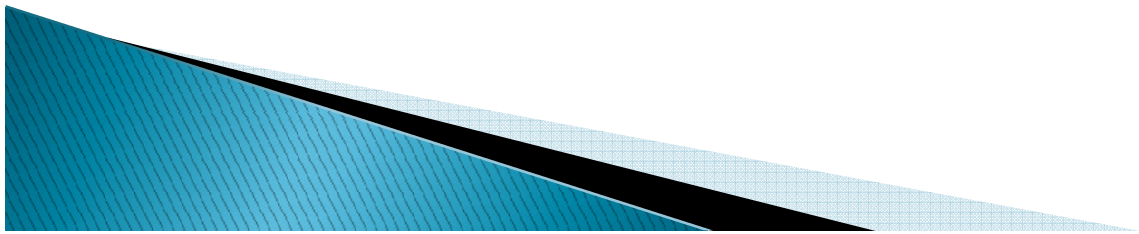
Existující typy DML triggerů

- DML aktivované triggery:
  - při rušení řádků (DELETE)
  - při vkládání řádků (INSERT)
  - při modifikaci určitých sloupců (UPDATE OF)
- Způsob vyvolání triggeru:
  - jednou při celé operaci
  - pro každý řádek (FOR EACH ROW) vstupující do zpracování operace
- Možnost kombinací operací (slučování OR)



- Zápis DML triggeru:

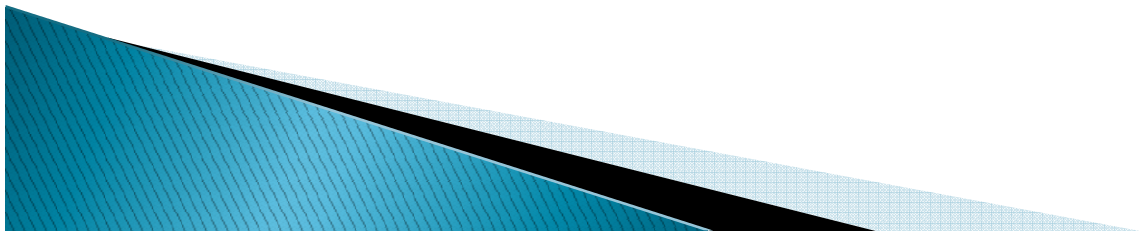
```
CREATE OR REPLACE TRIGGER jméno
BEFORE | AFTER | INSTEAD OF
DELETE | INSERT | UPDATE OF cols
ON  tabulka
[ způsob odkazování ]
[ FOR EACH ROW ]
[ WHEN ( podmínka ) ]
AS pl/sql kód
```



Způsob odkazování

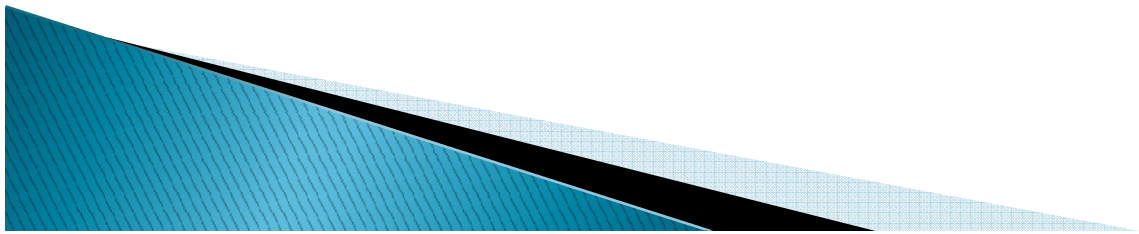
- Definuje, jak budou přístupné původní a nové záznamy (vstupující do DML operací)
- Implicitně :*new*, :*old*, :*parent*
- Existuje klauzule

```
REFERENCING
    [ OLD AS jméno ]
    [ NEW AS jméno ]
    [ PARENT AS jméno ]
```



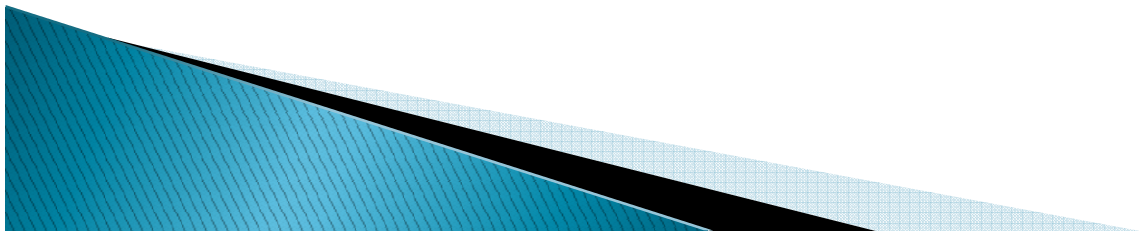
Způsob vyvolávání triggerů

- Pro BEFORE a AFTER je trigger chápán jako tzv. statement trigger a vyvolán je pouze jedenkrát (není-li klauzulí FOR EACH ROW explicitně stanoveno jinak)
- V případě INSTEAD OF triggeru je trigger implicitně chápán jako řádkový trigger, protože zde statement trigger nemá prakticky žádný význam



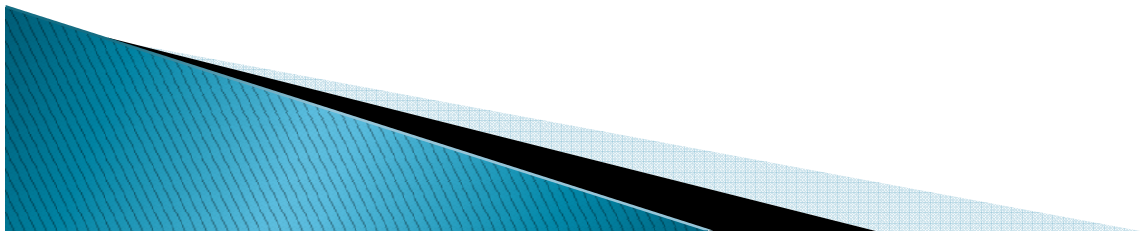
DDL triggery

- Jsou vyvolány po provedení DDL příkazu
- Mohou být BEFORE, AFTER
- Mohou být omezeny podmínkou (WHEN)
- Definují se dvěma způsoby:
  - *jméno události* ON DATABASE
  - *jméno události* ON *jméno schématu*
- Existuje řada definovaných událostí, např. CREATE, ALTER, DROP, RENAME, GRANT, COMMENT, AUDIT, DDL



Triggery databázových událostí

- Pracují stejně jako DDL triggery, pouze je jiná množina povolených událostí
- Typicky se jedná o události zásadních událostí v celé databázové instanci, např. `STARTUP`, `SHUTDOWN`, `LOGON`, `LOGOFF`, `SERVERERROR`, `SUSPEND` apod.
- Uvnitř DDL a databázových triggerů nelze provádět jiné DDL operace
- Velmi specifické použití

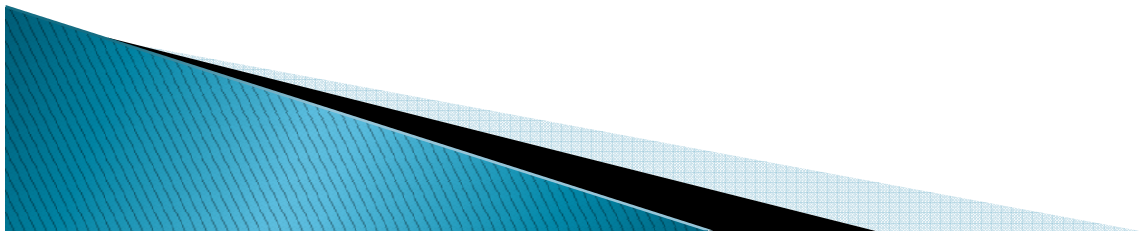


Omezení triggerů

- BEFORE a AFTER triggery nelze specifikovat nad pohledy
- V BEFORE triggerech není možné zapisovat do `:old` záznamů
- V AFTER triggerech nelze zapisovat ani do `:old`, ani do `:new` záznamů
- `INSTEAD OF` triggery pracují jen s pohledy, mohou číst `:old` i `:new`, ale nemohou zapisovat ani do jednoho
- Nelze kombinovat `INSTEAD OF` a `UPDATE`
- Nelze definovat trigger nad LOB atributem

Dvě zásadní omezení

- Nelze použít transakce, pokud je zpracovávána jiná transakce (tedy prakticky nelze použít transakce vůbec)
- Není možné sledovat (ani modifikovat) data v tabulce, která způsobila vyvolání DML triggeru – toto omezení je často velice nepříjemné
- Jediné známé řešení: zrcadlení tabulek

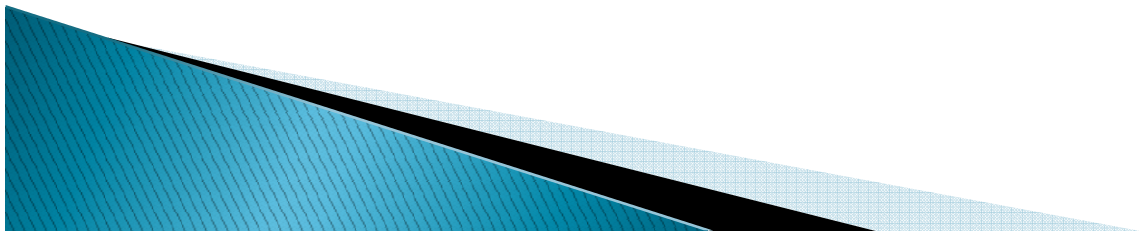


Rozlišení operace v triggeru

- Trigger může být volán různými operacemi (např. `INSERT OR DELETE`)
- V průběhu triggeru je třeba rozlišit, která operace se provádí
- Existují logické proměnné `INSERTING`, `DELETING` a `UPDATING` použitelné v rozhodování
- Pozn. Je třeba užívat řízení výjimek, protože chyba v triggeru ukončí obvykle celou nadřazenou transakci

Emulace `AUTOINCREMENT`

- Některé SQL databázové systémy používají modifikátor typu `AUTOINCREMENT` pro definici číslování primárních klíčů
- Je možné toto chování emulovat umístěním `BEFORE INSERT` triggeru, který vyčte novou hodnotu ze sekvence a modifikuje `:new.`*`id`* na tuto hodnotu



Starburst

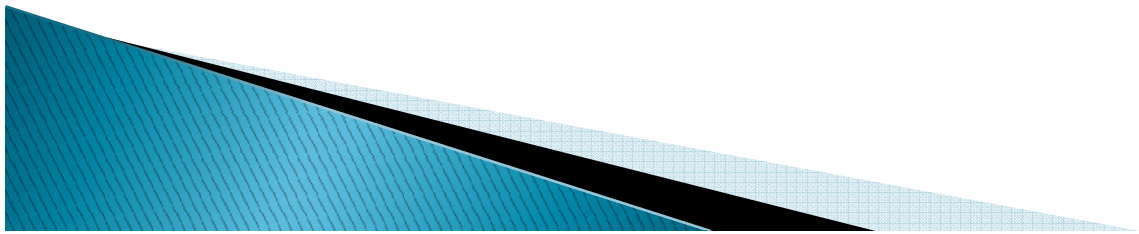
- Starburst Active Rule System
  - DDL rozšíření projektu Starburst
  - jednoduchá syntaxe i sémantika
- *událost – podmínka – akce (Event-Condition-Action, ECA)*
  - princip fungování triggerů
  - „Když nastane událost a je splněna podmínka, vykonej akci“
    - zní to jednoduše
    - má to spousty háčků

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

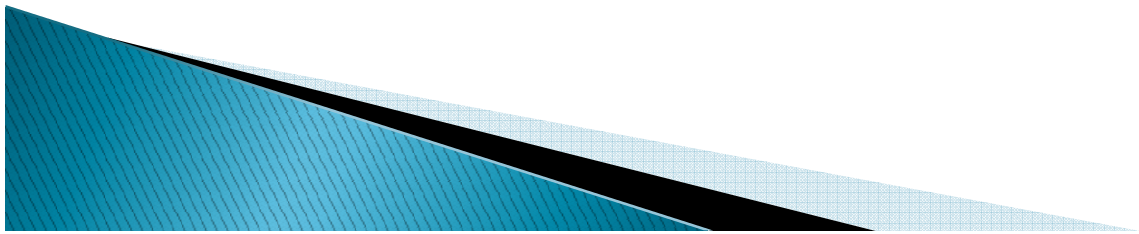
Syntaxe aktivních pravidel Starburst

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
```



Příklad I. – pokračování

```
CREATE RULE RegulacePlatu ON
      Zaměstnanci
WHEN INSERTED, DELETED,
      UPDATED (Plat)
IF    (SELECT AVG(Plat) FROM
      Zaměstnanci) > 100
THEN UPDATE Zaměstnanci
            SET Plat = 0.9 * Plat
```

Zaměstnanec	Plat
Božena	90
Jan	90
Josef	110

- Průměrný plat zaměstnance je 97.
- Uvažujme transakci, která přidá záznamy

Bořivoj	150
Oldřich	120

- Vložení spustí pravidlo **Regulace platu.**
- Nový průměrný plat je 112
  - ⇒ podmínka je splněna
  - ⇒ provede se akce

Příklad I. – pokračování

- Nový stav databáze po provedení pravidla:

Zaměstnanec	Plat
Božena	81
Jan	81
Josef	99
Bořivoj	135
Oldřich	108

- Operace UPDATE v akci pravidla způsobí, že se pravidlo spustí znovu.

- Průměrný plat je nyní 101
  - ⇒ podmínka je splněna
  - ⇒ provede se akce
- Nový stav databáze:

Zaměstnanec	Plat
Božena	73
Jan	73
Josef	89
Bořivoj	121
Oldřich	97

Příklad I. – dokončení

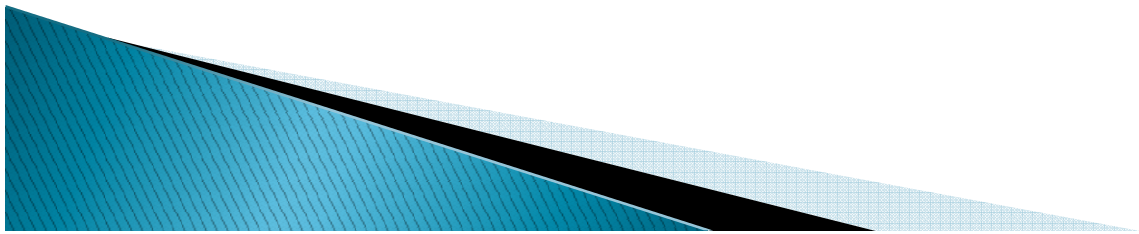
- Pravidlo je opět spuštěno díky operaci UPDATE.
- Pravidlo je vyhodnoceno, ale již se neprovede
  - Průměrný plat je teď 91.
- Algoritmus provádění aktivních pravidel končí.

- Nebezpečí „zacyklení“ v případě špatně definovaných pravidel.

Příklad II.

- Uvažujme databázi jako na začátku Příkladu I.
- K databázi přidáme nové aktivní pravidlo *VysocePlacení,*
  - Pravidlo vkládá do pohledu *VysocePlaceníZaměstnanci* (VPZ) ty nově přidané zaměstnance, kteří mají plat vyšší než 100.

```
CREATE RULE VysocePlacení ON Zaměstnanci
WHEN INSERTED
IF EXISTS (SELECT * FROM INSERTED WHERE Plat > 100)
THEN  INSERT INTO VPZ (SELECT * FROM INSERTED
                                        WHERE Plat > 100)
FOLLOWS RegulacePlatu
```



Příklad II. – pokračování

- Uvažujme nyní znovu přidání Bořivoje a Oldřicha do databáze.

Zaměstnanec	Plat
Božena	90
Jan	90
Josef	110

Bořivoj	150
Oldřich	120

- Operace INSERT spustí obě pravidla.
- Algoritmus zpracovávání pravidel vybere nejprve pravidlo *RegulacePlatu.*
- Pravidlo *RegulacePlatu* se provede díky rekurzi dvakrát.

Příklad II. – dokončení

- Tabulka Zaměstnanci se dostane do stavu jako v Příkladě I.

Zaměstnanec	Plat
Božena	73
Jan	73
Josef	89
Bořivoj	121
Oldřich	97

- Nyní je pravidlo *RegulacePlatu* „*nespuštěno*“ a pravidlo *VysocePlacení* je „*spuštěno*“.
- Pravidlo pokládá za vloženou tuto dočasnou tabulku:

Bořivoj	121
Oldřich	97

- Pouze řádek (Bořivoj,121) je vložen do *VPZ*.

Příklad II. – dokončení

- Tabulka Zaměstnanci se dostane do stavu jako v Příkladě I.

Zaměstnanec	Plat
Božena	73
Jan	73
Josef	89
Bořivoj	121
Oldřich	97

- Nyní je pravidlo *RegulacePlatu* „*nespuštěno*“ a pravidlo *VysocePlacení* je „*spuštěno*“.
- Pravidlo pokládá za vloženou tuto dočasnou tabulku:

Bořivoj	121
Oldřich	97

- Pouze řádek (Bořivoj,121) je vložen do *VPZ*.

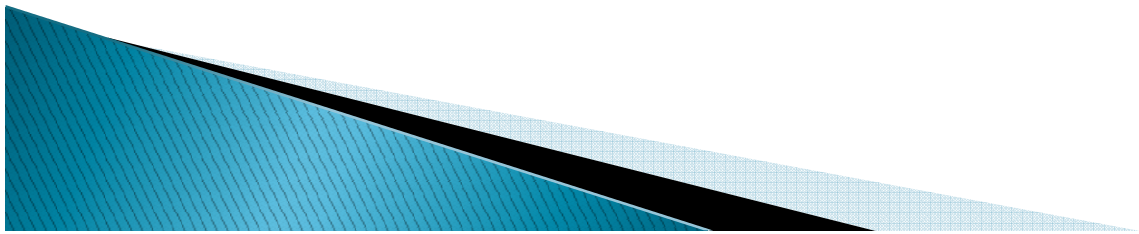
Doplnění syntaxe

- 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

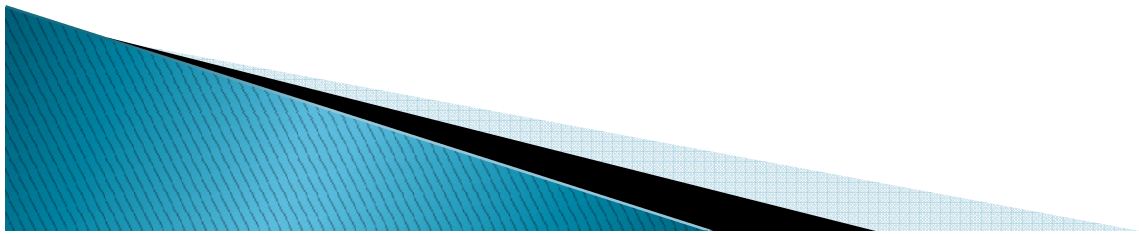
Pravidlo je:

- *spuštěné (triggered)*
  - pokud nastane jím sledovaná událost
  - ostatní pravidla nespuštěná
  - spuštěné neznamená „vykonává se akce“, „vyhodnocuje se podmínka“



Sémantika – terminologie

- *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)*

Sémantika – spuštění pravidel

- algoritmus vyhodnocení pravidel

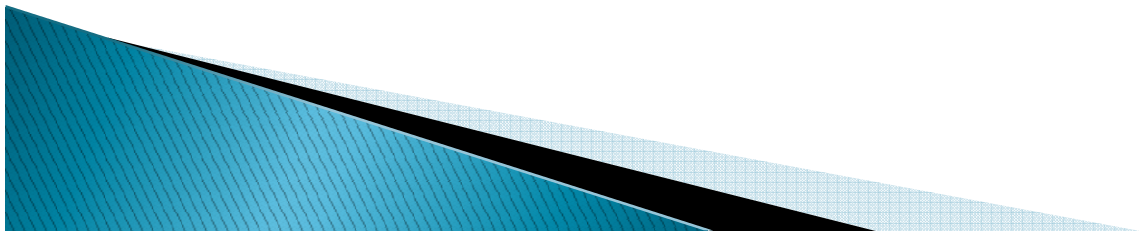
  DOKUD je množina spuštěných pravidel M neprázdná
  {
  vyber pravidlo R s nejvyšší prioritou z M a označ jako nespuštěné
  POKUD je podmínka R splněna {
  proveď akci R
  }
  }
- jednoznačně opakovatelné
  - díky úplnému uspořádání

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



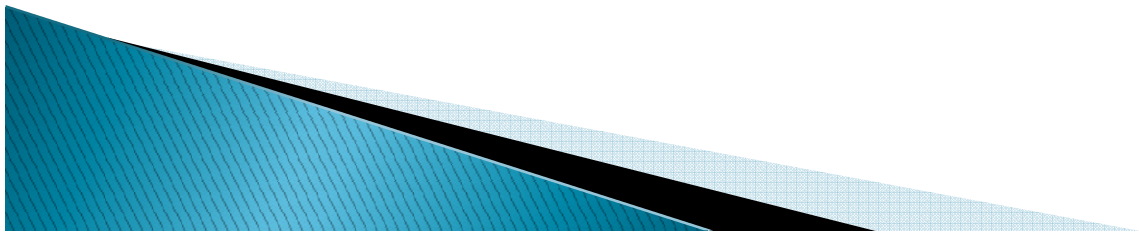
Sémantika – detaily

- *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

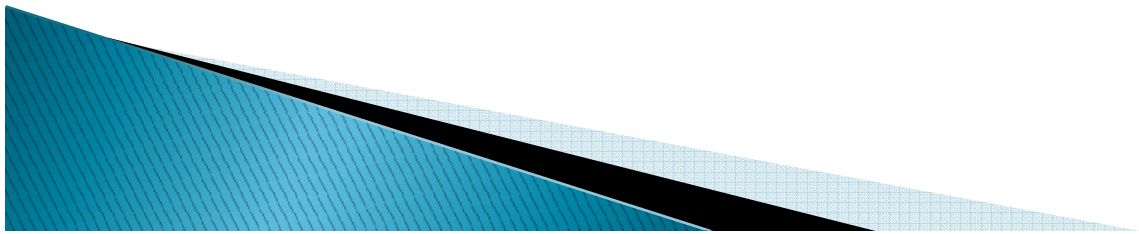
- probíhá algoritmus vyhodnocování pravidel
- pravidlo je spuštěno
  - pokud je množina operací pravidlem sledovaných neprázdná
    - vztaženo k aktuálnímu přechodovému stavu
    - přechodový stav se mění s vykonáváním pravidel
  - akce spuštěného pravidel vyústí ve změnu množin
  - znovu se utvoří konfliktní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

Oracle



Oracle – triggery

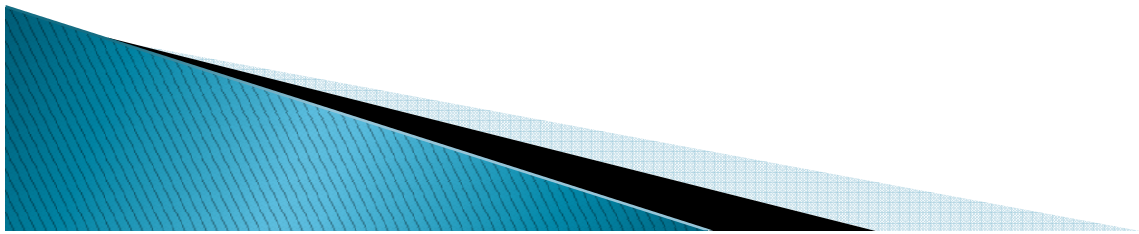
- podpora se vyvíjí
  - v dřívějších verzích četná omezení
    - událost sleduje pouze jeden trigger
    - nebylo možné ovlivňovat pořadí spouštění



Oracle – triggery

Dva typy triggerů:

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



Statement Triggers

Inserting, Deleting a Updating - může být použito ke zjištění, která událost nastane

```
CREATE OR REPLACE TRIGGER trig_testTable
AFTER INSERT or UPDATE ON Personnel
BEGIN
If Inserting
   Then INSERT into testTable values ('insert done', SYSDATE) ;
Else
    INSERT into testTable values ('update done', SYSDATE) ;
End If;
END;
```

Test_trigger_1 tests this trigger

Příkazové triggery jsou vhodné pro INSERT, neboť zahrnují pouze jeden řádek, DELETE a UPDATE často pracují s mnoha řádky najednou

Row trigger

- Pracuje potencionálně na více řádcích

```
CREATE OR REPLACE TRIGGER trig_test
AFTER UPDATE OF SNUM ON PERSONNEL
FOR EACH ROW
BEGIN
   null;                    -- write operations here
END;
```

Řádkový trigger se spouští pro každou řádku ovlivněnou DML operací

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

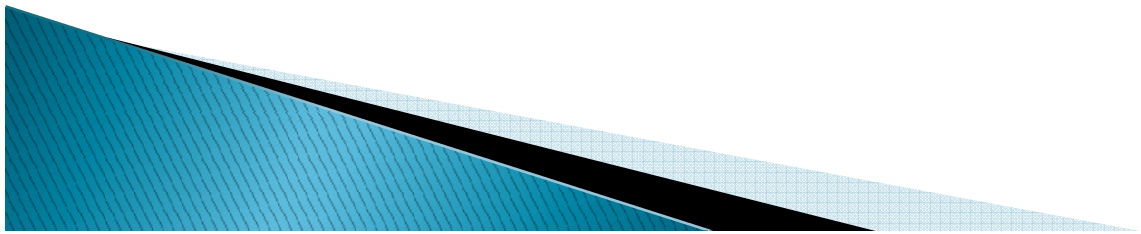
- *predikáty*
  - dostupné jsou **INSERTING, DELETING, UPDATING**
- *reference*
  - staré a nové hodnoty
    - **OLD, NEW**
  - lze je přejmenovat
  - pouze pro řádkové triggery

Sémantika 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

Sémantika Oracle triggerů

- vyjímky nebo chyby
  - všechny změny původní SQL operace a následné změny provedené triggery odrolovány
  - Oracle podporuje částečný rollback (oproti transakčnímu)



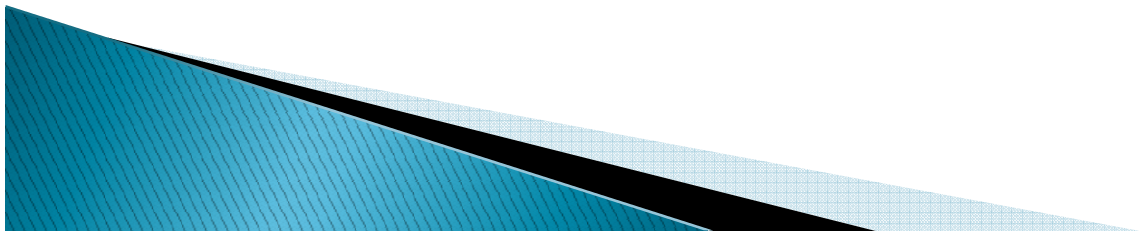
Sémantika Oracle triggerů

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

Sémantika 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***

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/allegati/23-triggers_in_sql_server.pdf
- 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