
KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI
www.inf.upol.cz

Jan Laštovička
https://www.inf.upol.cz/lide/jan-lastovicka
jan.lastovicka@upol.cz
17. listopadu 12, 771 46 Olomouc

Databázové systémy ◇ poznámky k přednášce

12. Pohledy

verze z 16. prosince 2024

1 Transakce

Každá změna databáze (hodnot relačních proměnných) se provádí v takzvaných **transakcích**.

Transakce splňují následující čtyři požadavky známé pod zkratkou **ACID**:

1. Atomicity. Provedou se všechny změny v transakci, nebo žádná.
2. Consistency. Transakce převádějí konzistentní stav databáze na konzistentní. Po provedení transakce musí být splněna všechna omezení databáze.
3. Isolation. Transakce nemůže být narušena jinou ve stejný čas probíhající transakcí.
4. Durability. Po dokončení transakce jsou změny v databázi trvalé.

Transakci skládající se z více změn začneme příkazem:

```
BEGIN ISOLATION LEVEL SERIALIZABLE;
```

Potvrzení složené transakce:

```
COMMIT;
```

Zrušení složené transakce:

```
ROLLBACK;
```

Například uvažujme relační proměnnou `actor` s charakteristickou vlastností „Herec `name` se narodil roku `born`.“ a hodnotou:

`name`	`born`
Keanu Reeves	1964
Laurence Fishburne	1961

založme transakci a přidejme entici do relace:

```
BEGIN ISOLATION LEVEL SERIALIZABLE;
INSERT INTO actor VALUES
  ( 'Gary Oldman', 1958 );
```

Změnu uvidíme:

```
# TABLE actor;

        name         | born
---------------------+------
 Keanu Reeves        | 1964
 Laurence Fishburne  | 1961
 Gary Oldman         | 1958
(3 rows)
```

Jiný uživatel však stále uvidí původní hodnotu proměnné.

Nyní se můžeme rozhodnout transakci zrušit a vrátit zpět její původní hodnotu, nebo ji potvrdit a tím učinit změnu viditelnou i pro jiné uživatele. Rozhodneme se pro potvrzení transakce:

```
ROLLBACK;
```

2 Virtuální relační proměnné

Relační proměnné dělíme na základní a virtuální. Základní relační proměnná přímo obsahuje svou hodnotu. Při zjišťování hodnoty stačí relaci z proměnné vyzvednout. Hodnota **virtuální relační proměnné** je definována relačním výrazem. Zjišťování její hodnoty vede k vyhodnocení relačního výrazu, který ji definuje.

Virtuální relační proměnnou (zkráceně **virtuální relaci** nebo jen **pohled**) deklarujeme příkazem:

```
CREATE VIEW relation AS expr;
```

kde *expr* je relační výraz. Schéma proměnné *relation* je rovno schématu relačního výrazu *expr*. Hodnota virtuální relace *relation* je rovna hodnotě výrazu *expr*.

Virtuální relaci můžeme zrušit příkazem:

```
DROP VIEW relation;
```

Uvažujme například relační proměnnou `movie` s charakteristickou vlastností „Film `title` byl vydaný roku `year`.“ a hodnotou:

name	born
The Matrix	1999
Dracula	1992

Vytvoříme virtuální relační proměnnou `movie_title` příkazem:

```
CREATE VIEW movie_title AS
SELECT DISTINCT title FROM movie;
```

Proměnná bude mít charakteristickou vlastnost „`title` je název filmu.“ Získáme hodnotu proměnné:

```
# TABLE movie_title;

   title
------------
 The Matrix
 Dracula
(2 rows)
```

Změna proměnné `movie` povede ke změně hodnoty `movie_title`:

```
# INSERT INTO movie VALUES
   ( 'American Beauty', 1999 );


# TABLE movie_title;

      title
-----------------
 American Beauty
 The Matrix
 Dracula
(3 rows)
```

Uživatel by neměl poznat, zda používá základní nebo virtuální relační proměnnou. Předchozí pravidlo se jmenuje **princip zaměnitelnosti**. Za život databáze se dokonce může základní proměnná v jistý okamžik stát virtuální. Základní relační proměnnou lze přímo měnit. Změní se relace, kterou uchovává. Změna virtuální relační proměnné by měla být také umožněna, pokud chceme dodržet princip zaměnitelnosti.

Konečná množina podmínek nad schématem S se nazývá **omezení** nad S. Například $\{\texttt{born} \leq \texttt{died}, 0 \leq \texttt{born}, 0 \leq \texttt{died}\}$ je omezení nad $\{\texttt{born}, \texttt{died}\}$. Řekneme, že entice t nad S **splňuje** omezení C nad S, jestliže splňuje všechny podmínky v C. Například entice

$$\{\langle \texttt{name}, \texttt{'Federico Fellini'}\rangle, \langle \texttt{born}, 1920\rangle, \langle \texttt{died}, 1993\rangle\}$$

splňuje omezení $\{\texttt{born} \leq \texttt{died}, 0 \leq \texttt{born}, 0 \leq \texttt{died}\}$. Relace nad S pak **splňuje** omezení C, jestliže ho splňuje každá entice v relaci.

Základní relační proměnné ***relation*** nad S můžeme přiřadit omezení. Hodnota proměnné, pak musí splňovat její omezení. Omezení C relační proměnné ***relation*** nad S je slabší než její charakteristická vlastnost V v tom smyslu, že jestliže relace nad S má vlastnost V, pak musí splňovat C.

Například pokud bychom měli proměnnou `corpse` nad $\{\texttt{name}, \texttt{born}, \texttt{died}\}$ určenou charakteristickou vlastností: „Člověk jménem `name` se narodil v roce `born` a umřel v roce `died`.“, pak proměnné můžeme přiřadit omezení $\{\texttt{born} \leq \texttt{died}, 0 \leq \texttt{born}, 0 \leq \texttt{died}\}$.

Omezení $\{\theta_1, \ldots, \theta_n\}$ nad S základní relace ***relation*** nad S můžeme uvést při jejím vytvoření jako sadu omezení základní relace:

```
CHECK (θ1),
⋮
CHECK (θn)
```

Například:

```
CREATE TABLE corpse (
    name text,
    born integer,
    died integer,
    CHECK (born <= died),
    CHECK (0 <= born),
    CHECK (0 <= died)
);
```

deklaruje omezení $\{\texttt{born} \leq \texttt{died}, 0 \leq \texttt{born}, 0 \leq \texttt{died}\}$ základní relace `corpse`.

Pokud θ je podmínka nad S a $h : S \to S'$ přejmenování atributů, pak $h(\theta)$ označíme podmínku, kde každý atribut $y \in S$ nahradíme za $h(y)$. Například pro θ rovno $0 \leq \texttt{born}$ a h rovno $\{\langle \texttt{born}, \texttt{person\_born} \rangle\}$ je $h(\theta)$ rovno $0 \leq \texttt{person\_born}$.

Ze znalosti omezení pro základní relační proměnné, můžeme vypočítat omezení hodnot relačních výrazů. Vezměme dva výrazy E_1 a E_2 nad S_1 a S_2 s omezeními C_1 a C_2, pak

- $E_1 \cup E_2$ má omezení $\{\theta_1 \vee \theta_2 \mid \theta_1 \in C_1 \text{ a } \theta_2 \in C_2\}$,
- $E_1 \cap E_2$ má omezení $C_1 \cup C_2$,
- $E_1 - E_2$ má omezení C_1,
- $\sigma_\theta(E_1)$ má omezení $C_1 \cup \{\theta\}$,
- $\pi_S(E_1)$ má omezení $\{\theta \in C_1 \mid \theta \text{ je podmínka nad } S\}$,

- $E_1 \bowtie E_2$ má omezení $C_1 \cup C_2$,
- $\rho_h(E_1)$ má omezení $\{h(\theta) \mid \theta \in C_1\}$

Podle předchozích pravidel můžeme například odvodit, že výraz

$$\pi_{\{\texttt{name},\texttt{born}\}}(\sigma_{\texttt{born}<2000}(\texttt{corpse}))$$

má omezení $\{0 \leq \texttt{born}, \texttt{born} < 2000\}$

Virtuální relační proměnné přiřadíme omezení shodné s omezením relačního výrazu, který určuje hodnotu virtuální relační proměnné.

Základní relační proměnné můžou určovat **výchozí hodnoty** atributů ve svém schématu. Přesněji každá základní relační proměnná ***relation*** typu S má určené výchozí hodnoty enticí nad $S' \subseteq S$. Například základní relační proměnná `movie` nad $\{\texttt{title}, \texttt{watch\_count}\}$ s charakteristickou vlastností „Film `title` byl shlédnut `watch_count` krát.“ by mohla mít výchozí hodnoty $\{\langle \texttt{watch\_count}, 0\rangle\}$.

Atributové omezení:

```
DEFAULT value
```

deklaruje výchozí hodnotu ***value*** pro daný atribut. Například:

```
CREATE TABLE movie (
    title text,
    watch_count integer DEFAULT 0
);
```

deklaruje proměnnou `movie` s výchozími hodnotami $\{\langle \texttt{watch\_count}, 0\rangle\}$.

Tento tvar příkazu INSERT:

```
INSERT INTO relation ( y1, ..., ym ) VALUES
  tuple1,
  ⋮
  tuplen;
```

vloží entice do relace, kde uživatel zadá hodnoty jen pro vyjmenované atributy. Pro zbylé atributy se použije jejich výchozí hodnota. Například:

```
INSERT INTO movie ( title ) VALUES ( 'The Matrix' );
```

vloží do `movie` entici $\{\langle \texttt{title}, \texttt{'The Matrix'}\rangle, \langle \texttt{watch\_count}, 0\rangle\}$.

Podobně jako u omezení, lze výchozí hodnoty definovat pro libovolný relační výraz následujícími pravidly. Vezměme dva výrazy E_1 a E_2 nad S_1 a S_2 s výchozími hodnotami d_1 a d_2 nad S_1' a S_2', pak

- $E_1 \cup E_2$ má výchozí hodnoty $d_1(d_2(S_2' - S_1'))$,
- $E_1 \cap E_2$ má výchozí hodnoty $d_1(d_2(S_2' - S_1'))$,
- $E_1 - E_2$ má výchozí hodnoty d_1,
- $\sigma_\theta(E_1)$ má výchozí hodnoty d_1,
- $\pi_S(E_1)$ má výchozí hodnoty $d_1(S_1' \cap S)$,
- $E_1 \bowtie E_2$ má výchozí hodnoty $d_1(d_2(S_2' - S_1'))$,
- $\rho_h(E_1)$ má výchozí hodnoty $\rho_h(d_1)$.

Například výraz $\rho_{\texttt{count} \leftarrow \texttt{watch\_count}}(\sigma_{\texttt{title='The Matrix'}}(\texttt{movie}))$ má výchozí hodnoty $\{\langle \texttt{count}, 0 \rangle\}$.

Zlaté pravidlo změny hodnoty relační proměnné zní:

> Po změně hodnoty relační proměnné musí její hodnota splňovat její omezení.

Vezměme libovolný relační výraz E s omezením C. Níže definujeme přidání entice t do výrazu E, odebrání entice t z výrazu E a změnu entice t_1 na entici t_2 ve výrazu E. Přidávaná entice t a změněná entice t_2 musí splňovat omezení C. Odebíraná entice t a měněná entice t_1 musí být v hodnotě výrazu E.

Pokud je relační výraz E základní relační proměnnou, přidání, odebrání a změna entice se provede přímočaře na hodnotě proměnné. Konkrétně vezměme základní relační proměnnou ***relation*** s hodnotou R. Přidání entice t do výrazu ***relation*** nastaví hodnotu proměnné ***relation*** na $R \cup \{t\}$, odebrání entice t z výrazu ***relation*** nastaví hodnotu proměnné ***relation*** na $R - \{t\}$ a změna entice t_1 na t_2 nejprve povede k odebrání entice t_1 z výrazu ***relation*** a poté přidání entice t_2 do výrazu ***relation***.

Pokud je E virtuální relační proměnná, pak se změna provede na relačním výrazu, který určuje hodnotu virtuální relační proměnné. Následuje rozbor změn pro jednotlivé relační operace. Níže použité relační výrazy E_1 a E_2 jsou nad S_1 a S_2 a mají omezení C_1 a C_2.

- Případ kdy $E = E_1 \cup E_2$.
  - Přidání. Pokud entice t splňuje omezení C_1, pak přidáme entici t do výrazu E_1. Pokud entice t splňuje omezení C_2, pak přidáme entici t do výrazu E_2.
  - Odebrání. Pokud entice t je v hodnotě výrazu E_1, pa ji odebereme z E_1. Poté pokud entice t je v hodnotě výrazu E_2, pak ji odebereme z E_2.
  - Změna. Odebereme entici t_1 z výrazu E a poté přidáme entici t_2 do výrazu E.

- Případ kdy $E = E_1 \cap E_2$.
  - Přidání. Přidáme entici t od výrazu E_1 a poté do výrazu E_2.
  - Odebrání. Odebereme entici t z výrazu E_1. Poté pokud je entice t v hodnotě výrazu E_2, pak ji odebereme z E_2.
  - Změna. Odebereme entici t_1 z výrazu E a poté přidáme entici t_2 do výrazu E.
- Případ kdy $E = E_1 - E_2$.
  - Přidání. Pokud se entice t nelézá v hodnotě výrazu E_2, pak ji odebereme z výrazu E_2. Přidáme entici t do výrazu E_1. Zkontrolujeme, zda se entice t nenalézá v hodnotě výrazu E_2. Pokud se zde nalézá, přidání nelze provést.
  - Odebrání. Odebereme entici t z výrazu E_1.
  - Změna. Odebereme entici t_1 z výrazu E a poté přidáme entici t_2 do výrazu E.
- Případ kdy $E = \sigma_\theta(E_1)$.
  - Přidání. Přidáme entici t do výrazu E_1.
  - Odebrání. Odebereme entici t z výrazu E_1.
  - Změna. Změníme entici t_1 na entici t_2 ve výrazu E_1.
- Případ kdy $E = \pi_S(E_1)$.
  - Přidání. Nechť entice t' nad S' jsou výchozí hodnoty výrazu E_1. Pokud $S_1 - S \not\subseteq S'$, pak přidání nelze provést. Jinak přidáme $t(t'(S_1 - S))$ do výrazu E_1.
  - Odebrání. Odebereme z výrazu E_1 každou entici t_1 v hodnotě výrazu E_1, která splňuje $t_1(S) = t(S)$.
  - Změna. Pro každou entici t v hodnotě výrazu R_1 splňující $t(S) = t_1$ provedeme změnu entice t na $t_2(t(S_1 - S))$ ve výrazu E_1.
- Případ kdy $E = E_1 \bowtie E_2$.
  - Přidání. Přidáme entici $t(R_1)$ do výrazu E_1 a poté entici $t(R_2)$ do výrazu E_2.
  - Odebrání. Odebereme entici $t(R_1)$ z výrazu E_1. Poté, pokud je entice $t(R_2)$ v hodnotě výrazu E_2, pak ji odebereme z E_2.
  - Změna. Odebereme entici t_1 z výrazu E a poté přidáme entici t_2 do výrazu E.
- Případ kdy $E = \rho_h(E_1)$. Níže používáme entici $t \circ h^{-1}$ nad S_1 definovanou tak, že $(t \circ h^{-1})(y) = t(h^{-1}(y))$ pro každé $y \in S_1$.[1]

[1] Je možné, že znáte definici kompozice funkcí, kde se argumenty zadávají v opačném pořadí. Výraz $r \circ h^{-1}$ čteme r *po* h^{-1}.

– Přidání. Přidáme entici $t \circ h^{-1}$ do výrazu E_1.
– Odebrání. Odebereme entici $t \circ h^{-1}$ z výrazu E_1.
– Změna. Odebereme entici t_1 z výrazu E a poté přidáme entici t_2 do výrazu E.

Vraťme se k dříve definované proměnné `movie` a uvažujme její hodnotu:

`title`	`watch_count`
The Matrix	5
Ford v Ferrari	2

Relační výraz $\pi_{\{\texttt{title}\}}(\texttt{movie})$ má hodnotu:

`title`
The Matrix
Ford v Ferrari

Změna entice $\{\langle \texttt{title}, \texttt{'Ford v Ferrari'}\rangle\}$ na $\{\langle \texttt{title}, \texttt{'Le Mans '66'}\rangle\}$ ve výrazu $\pi_{\{\texttt{title}\}}(\texttt{movie})$ povede na změnu hodnoty proměnné `movie`:

`title`	`watch_count`
The Matrix	5
Le Mans '66	2

Relační výraz $\pi_{\{\texttt{title}\}}(\texttt{movie})$ má změnou požadovanou hodnotu:

`title`
The Matrix
Le Mans '66

Přidání entice $\{\langle \texttt{title}, \texttt{'Inland Empire'}\rangle\}$ do téhož výrazu způsobí změnu hodnoty proměnné `movie`:

`title`	`watch_count`
The Matrix	5
Le Mans '66	2
Inland Empire	0

Opět došlo k požadované změně hodnoty výrazu:

`title`
The Matrix
Le Mans '66
Inland Empire

3 Aktualizovatelnost pohledů v SQL

Aktualizovatelnost relační proměnné znamená, zda můžeme změnit její hodnotu pomocí příkazů INSERT, DELETE a UPDATE.

Základní relace jsou vždy aktualizovatelné. Virtuální relace je aktualizovatelná, pokud pro relační výraz, který určuje její hodnotu, současně platí:

1. Musí se jednat o SELECT výraz.
2. V SELECT výrazu se mohou vyskytovat pouze klauzule SELECT, FROM a WHERE.
3. Ve FROM klauzuli musí být jediná aktualizovatelná relační proměnná.
4. SELECT klauzule nesmí obsahovat klíčové slovo DISTINCT.

Pokud SELECT výraz realizuje projekci relace na nadklíč, tak klíčové slovo DISTINCT můžeme bezpečně vynechat.

Aktualizovatelné relační proměnné můžou mít **aktualizovatelné** jen některé **atributy**. Všechny atributy základní relace jsou aktualizovatelné. Atribut aktualizovatelné virtuální relace je aktualizovatelný, jestliže jeho hodnota je v SELECT výrazu určena skalárním výrazem tvořeným pouze aktualizovatelným atributem.

Aktualizovatelné pohledy je potřeba zakládat příkazem:

```
CREATE VIEW relation AS expr WITH CHECK OPTION;
```

aby systém kontroloval splnění omezení.

Například:

```
CREATE VIEW actor_age AS
SELECT name,
       2022 - born AS age
FROM   actor
```

pohled má aktualizovatelný pouze atribut `name`, za předpokladu, že `actor` je aktualizovatelná relační proměnná s aktualizovatelnými atributy `name` a `age`.

Otázky a úkoly na cvičení

1. Vytvořte ve sdílené databázi transakci, ve které uděláte nějakou změnu, a zkontrolujte, že spolužáci změnu uvidí, až po potvrzení transakce.

2. Vytvořte základní relační proměnnou `cast` s charakteristickou vlastností „Herec `name` hrál ve filmu `title`.“ a základní relační proměnnou `movie` s charakteristickou vlastností „Film `title` byl natočen v roce `year`.“ Nastavte hodnoty proměnných. Vytvořte virtuální relační proměnnou `movie_cast` s charakteristickou vlastností „Herec `name` hrál ve filmu `title` z roku `year`.“

3. Dále vytvořte virtuální proměnnou `actor_active` s charakteristickou vlastností „Herec `name` hrál ve filmu z roku `year`.“ Použijte virtuální relační proměnnou z předchozího úkolu.

4. Předefinujte proměnnou `movie` tak, aby atribut `year` měl rozumná omezení na své hodnoty.

5. Odvoďte omezení pro následující výraz:

$$\sigma_{\texttt{name='Keanu Reeves'}}(\texttt{cast}) \bowtie \texttt{movie}$$

6. Definujte výchozí hodnotu pro atribut `year` proměnné `movie` rovnou aktuálnímu roku.

7. Vložte entici do proměnné `movie` tak, aby se na rok vydání filmu použila výchozí hodnota.

8. Odvoďte výchozí hodnoty výrazu:

$$\rho_{\texttt{year\_movie}\leftarrow\texttt{year}}(\pi_{\{\texttt{year}\}}(\texttt{movie}))$$

9. Zkuste podle pravidel provést odebrání entice z výrazu:

$$\sigma_{\texttt{year}=1999}(\texttt{actor\_active}).$$

10. Je možné do předchozího výrazu přidat entici?

11. Vytvořte aktualizovatelný pohled s charakteristickou vlastností „Film `title` byl natočen v roce 2000.“ a zkuste změnit jeho hodnotu.