

SQL

Martina Cvinčková



AGENDA

- **Opakování**
- **Databázový jazyk SQL**
 - **Views - pohledy**
 - **CTEs - společné tabulkové výrazy**
 - **Užitečné funkce**
 - **Práce s řetězci**
 - **CASE**

SQL

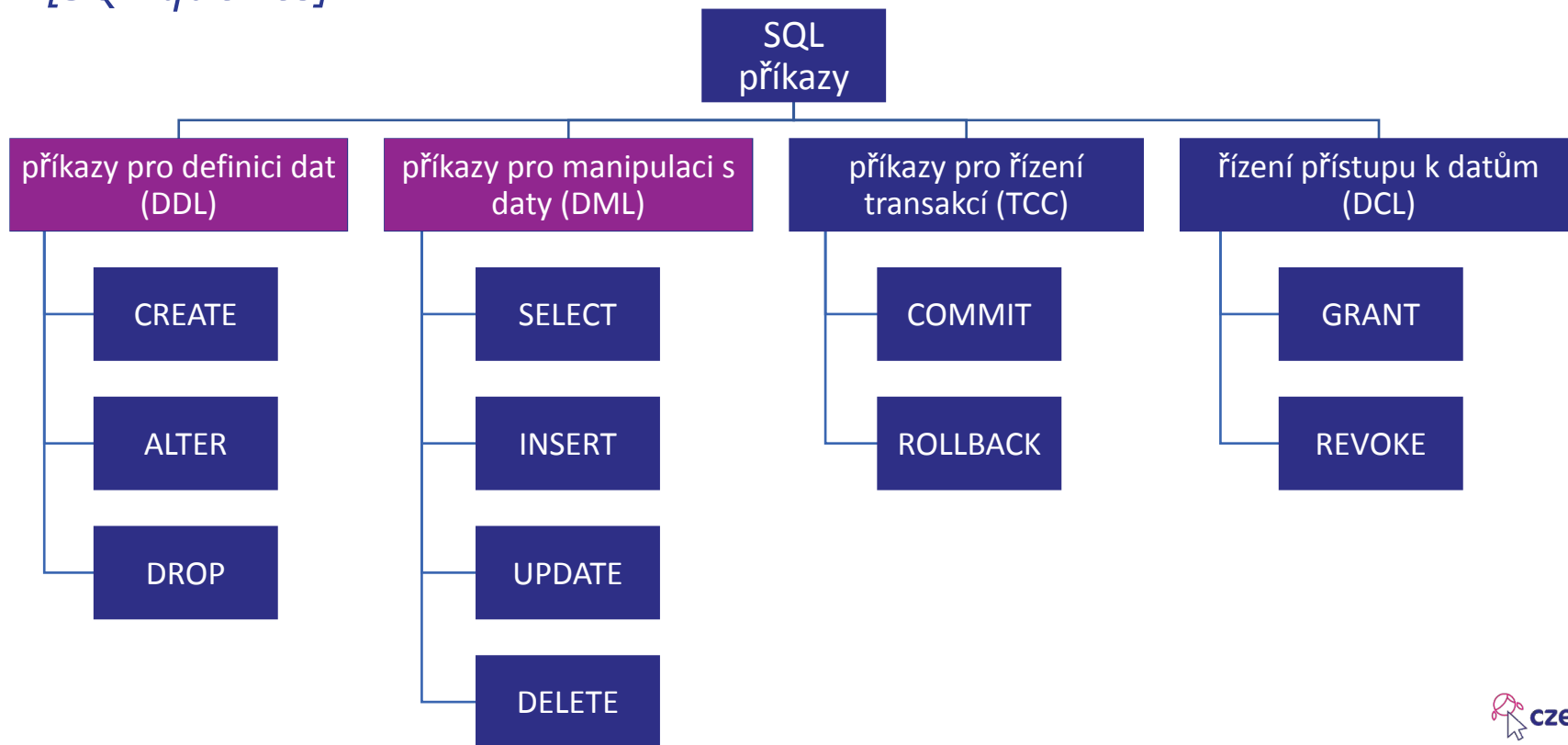
Opakování

Úrovně DB přístupu

- **READ ACCESS** (přístup pro čtení)
 - umožňuje uživatelům prohlížení existujících dat v databázi, ale neumožňuje jim provádět žádné změny v datovém obsahu nebo ve struktuře databáze.
- **WRITE ACCESS** (přístup pro zápis)
 - umožňují uživatelům provádět operace manipulace s daty, jako je vkládání nových dat (INSERT), aktualizace existujících dat (UPDATE) a mazání dat (DELETE)
- **ADMINISTRATIVE PRIVILEGES** (správcovská oprávnění)
 - umožňují provádět operace spojené se správou databáze. To zahrnuje definici nebo změnu struktury databáze pomocí DDL příkazů (CREATE, ALTER, DROP), správu uživatelů a jejich oprávnění a další úkoly související se správou databáze

SQL příkazy/dotazy

[SQL queries]



SQL

VIEWS - pohledy

VIEWS (pohledy)

- Uložený výsledek dotazu, který funguje jako virtuální tabulka
- Je definován pomocí příkazu SELECT, který získává data z jedné nebo více tabulek

VIEWS vs TABLEs (pohledy vs tabulky)

- Hlavní rozdíl mezi pohledem (view) a tabulkou (table) spočívá v tom, že tabulka ukládá data fyzicky, zatímco pohled data neuchovává; je to pouze uložený výsledek dotazu.
- S pohledy lze pracovat téměř stejně jako s tabulkami (DML příkazy vedoucí ke změně dat (INSERT, UPDATE, DELETE) mají určitá omezení)

Proč a kdy používat VIEWS?

- Zjednodušení složitých dotazů
- Opakovatelnost kódu
- Abstrakce & příprava dat
 - uživatelé mohou pracovat interagovat s pohledem, který představuje zjednodušenou nebo přizpůsobenou reprezentaci dat
- Ovládání zabezpečení
 - Skrytí sloupců, které by někteří uživatelé neměli vidět.
 - Přidělování práv uživatelům (viditelnost pouze view místo tabulky).

CREATE & DROP VIEW

- Vytvoření view:

CREATE VIEW nazev_view **AS**

SELECT sloupec1,..., sloupecN **FROM** tabulka **WHERE** ;

- Smazání view:

DROP VIEW nazev_view;

VIEW příklad

```
CREATE VIEW StateManufacturerView AS
```

```
SELECT
```

```
    c.State,
```

```
    m.Manufacturer,
```

```
    SUM(s.Revenue) AS TotalRevenue
```

```
FROM
```

```
    Sales s
```

```
    JOIN Country c ON s.Zip = c.Zip
```

```
    JOIN Product p ON s.ProductID = p.ProductID
```

```
    JOIN Manufacturer m ON p.ManufacturerID = m.ManufacturerID
```

```
GROUP BY
```

```
    c.State,
```

```
    m.Manufacturer;
```

SQL

Common Table Expressions (CTEs) - společné tabulkové výrazy

Common Table Expressions

- Dočasné výsledky dotazu
- Nejsou uloženy jako objekty v databázi
- Nemohou být odkazovány mimo dotaz, ve kterém jsou definovány

Proč a kdy používat CTEs?

- Zjednodušení složitých dotazů
- Opakovatelnost kódu
- Rekurzivní dotazy
- Zvýšení výkonu dotazů
 - V některých případech mohou CTE zlepšit výkon dotazů tím, že umožní optimalizaci dotazu nebo sníží počet operací spojení.

Common Table Expressions

- Použití CTE:

WITH *alias_CTE* **AS**

(**SELECT** * **FROM** table)

SELECT * **FROM** *alias_CTE*;

CTE příklad

```
WITH RevenueUnitsByState AS
```

```
(
```

```
    SELECT
```

```
        c.State,
```

```
        SUM(s.Revenue) AS TotalRevenue,
```

```
        SUM(s.Units) AS TotalUnits
```

```
    FROM
```

```
        Sales s JOIN Country c ON s.Zip = c.Zip
```

```
    GROUP BY
```

```
        c.State
```

```
)
```

```
SELECT
```

```
    state,
```

```
    TotalRevenue / TotalUnits AS RevenuePerUnit
```

```
FROM
```

```
    RevenueUnitsByState
```

```
GROUP BY
```

```
    state
```

Samostatná práce 1

VIEWs & CTEs

SQL

Užitečné funkce -

SQL

Užitečné funkce - práce s řetězci

LIKE

... WHERE column **LIKE** '%n%';

- Název produktu začíná na 'Abbas RS', obsahuje 'RS' nebo končí na '03'.

```
SELECT
```

```
    *
```

```
FROM
```

```
    Product
```

```
WHERE
```

```
    Product LIKE 'Abbas RS%' -- OR '%RS%' OR '%03';
```

Často používané funkce

Funkce	Syntaxe	Popis
<u>SUBSTR</u>	substr(řetězec, začátek, délka)	Získá část řetězce, jež začíná na zadané pozici a má zadanou délku.
<u>TRIM</u>	trim(řetězec, [znak])	Vrátí kopii řetězce, ze kterého bylo odstraněn specifický charakter ze začátku a konce řetězce.
<u>LTRIM</u>	ltrim(řetězec,[znak])	Vrátí kopii řetězce, ze kterého bylo odstraněn specifický charakter ze začátku řetězce.
<u>RTRIM</u>	rtrim(řetězec,[znak])	Vrátí kopii řetězce, ze kterého bylo odstraněn specifický charakter z konce řetězce.
<u>LENGTH</u>	length(řetězec)	Vrátí počet znaků v řetězci.
<u>REPLACE</u>	replace(řetězec,pattern,replacement)	Vrátí kopii řetězce, kdy každý výskyt podřetězce je nahrazen jiným podřetězcem.
<u>UPPER</u>	upper(řetězec)	Vrátí kopii řetězce, kdy všechny jeho znaky jsou nyní velká písmena.
<u>LOWER</u>	lower(řetězec)	Vrátí kopii řetězce, kdy všechny jeho znaky jsou nyní malá písmena.
<u>CONCAT</u>	concat(řetězec1, řetězec2, ... řetězecN)	Spojí dva nebo více textových řetězců dohromady
<u>INSTR</u>	instr(řetězec, podřetězec);	Najde podřetězec v řetězci a vrátí číslo pozice, kde začíná jeho první výskyt.

Práce s řetězci - příklady

- Vyberte názvy měst bez přípon státu a země.

Práce s řetězci - příklady

- Vyberte města bez přípon státu a země.

```
SELECT  
    RTRIM(city, CONCAT(state, ', ', country)) AS trimmedCity  
FROM  
    country
```

Práce s řetězci - příklady

- Vyberte názvy a sériová čísla produktů - sériové číslo je dvojčíslí za znakem '-'.

Práce s řetězcí - příklady

- Vyberte názvy a sériová čísla produktů - sériové číslo je dvojčíslí za znakem '-'.

```
SELECT
    product,
    SUBSTR(product, INSTR(product, '-') + 1) AS SerialNumber
FROM
    product;
```


SQL

Užitečné funkce - CASE

CASE

- Funkce, jež vrací výsledky v závislosti na hodnotě či podmínce
- Dva typy:
 - Jednoduchý příkaz CASE
 - Hledaný příkaz CASE

Jednoduchý CASE *[Simple CASE statement]*

- Přímě porováváme sloupec se specifickými hodnotami

CASE sloupec

WHEN hodnota1 **THEN** výsledek1

WHEN hodnota2 **THEN** výsledek2

...

[**ELSE** výsledek_jinak]

END

CASE

- Zobrazte číslo a název měsíce, vytvořte nový sloupec s ročním obdobím.

CASE

- Zobrazte číslo a název měsíce, vytvořte nový sloupec s ročním obdobím.

```
SELECT
```

```
    DISTINCT monthNo,
```

```
    monthName,
```

```
    CASE monthNo
```

```
        WHEN 1 THEN 'Zima' WHEN 2 THEN 'Zima' WHEN 3 THEN 'Jaro'
```

```
        WHEN 4 THEN 'Jaro' WHEN 5 THEN 'Jaro' WHEN 6 THEN 'Léto'
```

```
        WHEN 7 THEN 'Léto' WHEN 8 THEN 'Léto' WHEN 9 THEN 'Podzim'
```

```
        WHEN 10 THEN 'Podzim' WHEN 11 THEN 'Podzim' WHEN 12 THEN 'Zima'
```

```
    END AS rocni_obdobi
```

```
FROM
```

```
    date;
```

CASE - příklad

- Vybere názvy produktů z tabulky "product" a přiřadíte jim cenovou kategorii podle následujících kritérií:
 - "HIGH" – pokud je cena produktu vyšší nebo rovna 1000
 - "MEDIUM" – pokud je cena je vyšší nebo rovna 500, ale nižší než 1000
 - "LOW" – pokud je cena je nižší než 500

CASE - příklad

- Vybere názvy produktů z tabulky "product" a přiřadíte jim cenovou kategorii podle následujících kritérií:
 - "HIGH" – pokud je cena produktu vyšší nebo rovna 1000
 - "MEDIUM" – pokud je cena je vyšší nebo rovna 500, ale nižší než 1000
 - "LOW" – pokud je cena je nižší než 500

```
SELECT
```

```
    product,
```

```
    CASE
```

```
        WHEN priceNew >= 1000 THEN 'HIGH'
```

```
        WHEN priceNew >= 500 THEN 'MEDIUM'
```

```
        ELSE 'LOW'
```

```
    END AS price_category
```

```
FROM
```

```
    product
```

Hledaný CASE *[Searched CASE statement]*

- Klíčové slovo WHEN je následované podmínkou

CASE

WHEN podmínka Ano/Ne **THEN** výsledek1

WHEN podmínka Ano/Ne **THEN** výsledek2

...

[**ELSE** výsledek_jinak]

END

CASE

- Zobrazte číslo a název měsíce, vytvořte nový sloupec s ročním obdobím (pomocí podmínky).

CASE

- Zobrazte číslo a název měsíce, vytvořte nový sloupec s ročním obdobím (pomocí podmínky).

```
SELECT
```

```
    DISTINCT monthNo,
```

```
    monthName,
```

```
    CASE
```

```
        WHEN monthNo IN (1, 2, 12) THEN 'Zima'
```

```
        WHEN monthNo IN (3, 4, 5) THEN 'Jaro'
```

```
        WHEN monthNo IN (6, 7, 8) THEN 'Léto'
```

```
        WHEN monthNo IN (9, 10, 11) THEN 'Podzim'
```

```
    END AS roční_období
```

```
FROM
```

```
    date;
```

Samostatná práce 2

Řetězce & CASE

SQL část je za námi 😊

SQL "best practices"

- <https://data36.com/sql-best-practices-data-analysts/>
- <https://365datascience.com/tutorials/sql-tutorials/sql-best-practices/>
- <https://www.metabase.com/learn/building-analytics/sql-templates/sql-best-practices>

JAK POKRAČOVAT DÁL S SQL?

- Czechitas kurzy - SQL 2 a SQL 3
- mystery.knightlab.com
- datacamp.com
- hackerrank.com
- sqlclimber.com
- codewars.com