Základy forenzních databází

Tereza Uhlíková

verze 2.0

Kdo jsem

Tereza Uhlíková

Ústav analytické chemie

skupina teoretické spektroskopie

místnost A277

https://web.vscht.cz/~uhlikovt/

tereza.uhlikova@vscht.cz

1. lekce

- Co je to databáze
- Proč, kdo, kdy, jak ...
- Něco málo z historie
- CAP problém

2. lekce

- Základy informatiky
- Software
  - informace
  - záznam informace
  - číselné soustavy
  - písmenné kódy
- Hardware
  - Alan Turing, John von Neumann a počítač
  - historie vývoje počítače & super počítač
  - procesor a datová uložiště
- Architektura databází

3. lekce

- Algoritmus
  - vlastnosti
  - zápis
  - struktura
- Datové typy

Arthur C. Clarke - Devět miliard božích jmen

4. lekce

- Výroková logika
- Databáze
  - DB a SŘBD
  - databázové modely
- Relační model
  - návrh tabulky

Aghata Christie - Hercules Poirot

5. lekce

- ERA model
- Klíče, integrita a kardinalita
- Normalizace databáze

6. lekce

- Datové struktury
- Ukládání
- Složitost
- Řazení
- Přenos dat

7. lekce

- Vyhledávání
  - sekvenční
  - binární
  - hashing
- Jak google pracuje
- Neuronové sítě a Strojové učení

8. lekce

- Proč kvantové počítače
- Hardware
- Qbit
- Základní principy - superpozice, provázanost, interference
- Využití, užitečné algoritmy pro vyhledávání
- Problémy

O čem budeme mluvit dnes

- Strukturované programovací jazyky
- Databázové jazyky
- Pár vět o SQL

Algoritmus

Algoritmus

Formálně zapsaný postup pro řešení daného problému.
Základní komponenty algoritmu:

Algoritmus

Formálně zapsaný postup pro řešení daného problému.
Základní komponenty algoritmu:

- posloupnost (sekvence) příkazů – kroky v daném pořadí
- větvení (podmínka) – výběr dalších prováděných kroků závisí na splnění/nesplnění nějaké podmínky
- cyklus – opakované provádění kroků

Vlastnosti:

Algoritmus

Formálně zapsaný postup pro řešení daného problému.
Základní komponenty algoritmu:

- posloupnost (sekvence) příkazů – kroky v daném pořadí
- větvení (podmínka) – výběr dalších prováděných kroků závisí na splnění/nesplnění nějaké podmínky
- cyklus – opakované provádění kroků

Vlastnosti:

- rezultativost (výsledkovost); elementárnost (jednoduchost); determinovanost (jednoznačnost); finitnost (konečnost)
- korektnost (správnost); univerzálnost (obecnost, hromadnost); efektivita (úspornost)

Strukturovaný přirozený jazyk

Napiš algoritmus pro výpočet obvodu kruhu.

Strukturovaný přirozený jazyk

Napiš algoritmus pro výpočet obvodu kruhu.

1. Tiskni: Zadej poloměr
2. Čti:r
3. Jestliže je $r < 0$, krok 7
4. $o = 2 \cdot \pi \cdot r$
5. Tiskni: Obvod je o
6. Konec
7. Tisk: Chyba: Poloměr je záporný
8. Konec

Strukturované programování

struktura všech programů může být vyjádřena pomocí čtyř prostředků (stavebních prvků):

- posloupnost instrukcí
- větvení (podmínka)
- cykly (opakování)
- moduly (funkce)

Kromě těchto strukturálních prvků potřebujeme doplnit několik dalších rysů, bez kterých by program ztrácel smysl:

Strukturované programování

struktura všech programů může být vyjádřena pomocí čtyř prostředků (stavebních prvků):

- posloupnost instrukcí
- větvení (podmínka)
- cykly (opakování)
- moduly (funkce)

Kromě těchto strukturálních prvků potřebujeme doplnit několik dalších rysů, bez kterých by program ztrácel smysl:

- data
- operace (sčítání, odčítání, porovnávání, atd.)
- schopnost vstupu a výstupu údajů (například výstup výsledků na displej)

Proměnné, výrazy a příkazy

proměnná

Proměnné, výrazy a příkazy

proměnná - symbolické jméno, které odkazuje na nějakou hodnotu v paměti. Hodnota proměnné se může měnit.

výraz

Proměnné, výrazy a příkazy

proměnná - symbolické jméno, které odkazuje na nějakou hodnotu v paměti. Hodnota proměnné se může měnit.

výraz - matematické znázornění toho, co se má udělat.
Infixová notace - běžný způsob zápisu matematických výrazů, ve kterém jsou operátory napsány mezi operandy, se kterými pracují (např. 3 + 4).
prefixová notace (+ 3 4), postfixová notace (3 4 +)

příkaz - zapíšeme-li za přiřazovací výraz v jazyce C,SQL středník, stane se z něj přiřazovací příkaz, např. i=i+j;

Způsob zpracování programu

Celý program se skládá z množiny funkcí, které obsahují a zpracovávají data a proměnné a jejich prostřednictvím vzájemně komunikují.

Kompilátor

input		proces		output
zdrojvý kód	→	Kompilátor	→	Spustitelný program
Data	→	Spustitelný program	→	Output programu

Interpret

input		proces		output
zdrojvý kód	→	Interpretr	→	Output programu
Data	→	→↑		

Kompilátor x Interpret

kompilované jazyky (C, Pascal, vizual Basic):

1. napíšu (třeba v text. editoru) zdrojový text
2. překladače ho přeloží do strojového kódu, uloží do souboru (.exe)
3. přeložený soubor jde spustit na jakémkoli počítači

Kompilátor x Interpret

kompilované jazyky (C, Pascal, vizual Basic):

1. napíšu (třeba v text. editoru) zdrojový text
2. překladače ho přeloží do strojového kódu, uloží do souboru (.exe)
3. přeložený soubor jde spustit na jakémkoli počítači

interpretované jazyky (Basic, Python, Java, SQL, TCL, C#):

1. napíšu (třeba v text. editoru) zdrojový text
2. soubor uložím (v souladu s konvencí s příponou podle jazyka .py)
3. poklikáním na tento spustím interpret (Python.exe), který soubor načte, analyzuje a provede

Generace programovacích jazyků

1. Strojový kód (1100101) - rychlý a efektivní, bez překladače x těžko srozumitelný

Generace programovacích jazyků

1. Strojový kód (1100101) - rychlý a efektivní, bez překladače x těžko srozumitelný
2. Assembler (Assembler) - rychlý, snadno změnitelný x závisí na hardwaru, architektuře

Generace programovacích jazyků

1. Strojový kód (1100101) - rychlý a efektivní, bez překladače x těžko srozumitelný
2. Assembler (Assembler) - rychlý, snadno změnitelný x závisí na hardwaru, architektuře
3. Procedurální (C, Fortran, Algol) - strukturované, anglická slova, méně řádků x potřeba compilátor

Generace programovacích jazyků

1. Strojový kód (1100101) - rychlý a efektivní, bez překladače x těžko srozumitelný
2. Assembler (Assembler) - rychlý, snadno změnitelný x závisí na hardwaru, architektuře
3. Procedurální (C, Fortran, Algol) - strukturované, anglická slova, méně řádků x potřeba compilátor
4. Neprocedurální (SQL) - deklarativní, dobře pochopitelné, rychle zvládne každý x velká spotřeba paměti, méně flexibilní

Generace programovacích jazyků

1. Strojový kód (1100101) - rychlý a efektivní, bez překladače x těžko srozumitelný
2. Assembler (Assembler) - rychlý, snadno změnitelný x závisí na hardwaru, architektuře
3. Procedurální (C, Fortran, Algol) - strukturované, anglická slova, méně řádků x potřeba compilátor
4. Neprocedurální (SQL) - deklarativní, dobře pochopitelné, rychle zvládne každý x velká spotřeba paměti, méně flexibilní
5. AI (Prolog, LISP) - stroj dělá rozhodnutí, snadné k naučení i užití x dlouhý a komplexní kód, potřeba mnoho paměti

Vyšší programovací jazyky

- Procedurální (strukturované) – C, Fortran, Algol, Cobol, Basic, PL/1, dbase, clipper, FoxPro,...
  - Objektově orientované – SmallTalk, C++, Java
- Neprocedurální (deklarativní) - SQL - popis cíle, ale přesný algoritmus je záležitostí překladače
  - Funkcionální – LISP, APL, Haskell, Erlang
  - Logické – Prolog, Absys, Planner
- Značkové – HTML, XML (1996)
- Skriptovací – shell, JavaScript, Perl, Python, ...

CASE sensitivita – rozlišuje velká a malá písmena v názvech proměnných, příkazů, funkcí...

Pseudojazyk

– využíváme konstruktů nějakého programovacího jazyka, ale příkazy píšeme česky, často „volnou“ formou

1. Tiskni: Zadej poloměr
2. Cti: r
3. if r $<$ 0 then Tiskni: Chyba: Poloměr je záporný
4. else
5. begin
6. o $= 2 \cdot \pi \cdot r$
7. Tiskni: Obvod je o
8. end

Jazyk C

```
#include <stdio.h>
#include <stdlib.h>
int main()
{float r,o;
  printf("Zadej polomer: ");
  scanf("%f",&r);
  if(r<0) printf("Chyba: Polomer je zaporny");
  else{
    o = 2*M_PI*r;
    printf("Obvod je %f",o);
  }
  return 0;
}
```

Python

```
import math
vstup = raw_input("Zadejte polomer: ")
r = float(vstup)
O = 2 * math.pi *r
print "Výsledek je:", O
```

jazyk LISP, Prolog

výpočet faktoriálu

```
(defun fact (n)
    (if (= n 0)
    1
     (* n (fact (- n 1)))))
```

```
factorial(0,1).
factorial(N,M) :-
N>0,
N1 is N-1,
factorial(N1,M1),
M is N*M1.
```

“Hello world” v mnoha jazycích
http://helloworldcollection.de/

Jazyky pro práci s daty

R - jazyk a prostředí pro statistickou analýzu, existuje grafické rozhraní, interpretovaný, propojitelný s C a TEXem
r-project.cz/about.html

Jazyky pro práci s daty

R - jazyk a prostředí pro statistickou analýzu, existuje grafické rozhraní, interpretovaný, propojitelný s C a TEXem
r-project.cz/about.html

Python - interpretovaný, skriptovací, jednoduchý, otevřený, spousty knihoven (data manilupace - pandas a NumPy; vizualizace - Matplotlib a seaborn; testování hypotéz a modelování - SciPy, scikit-learn, statsmodels)

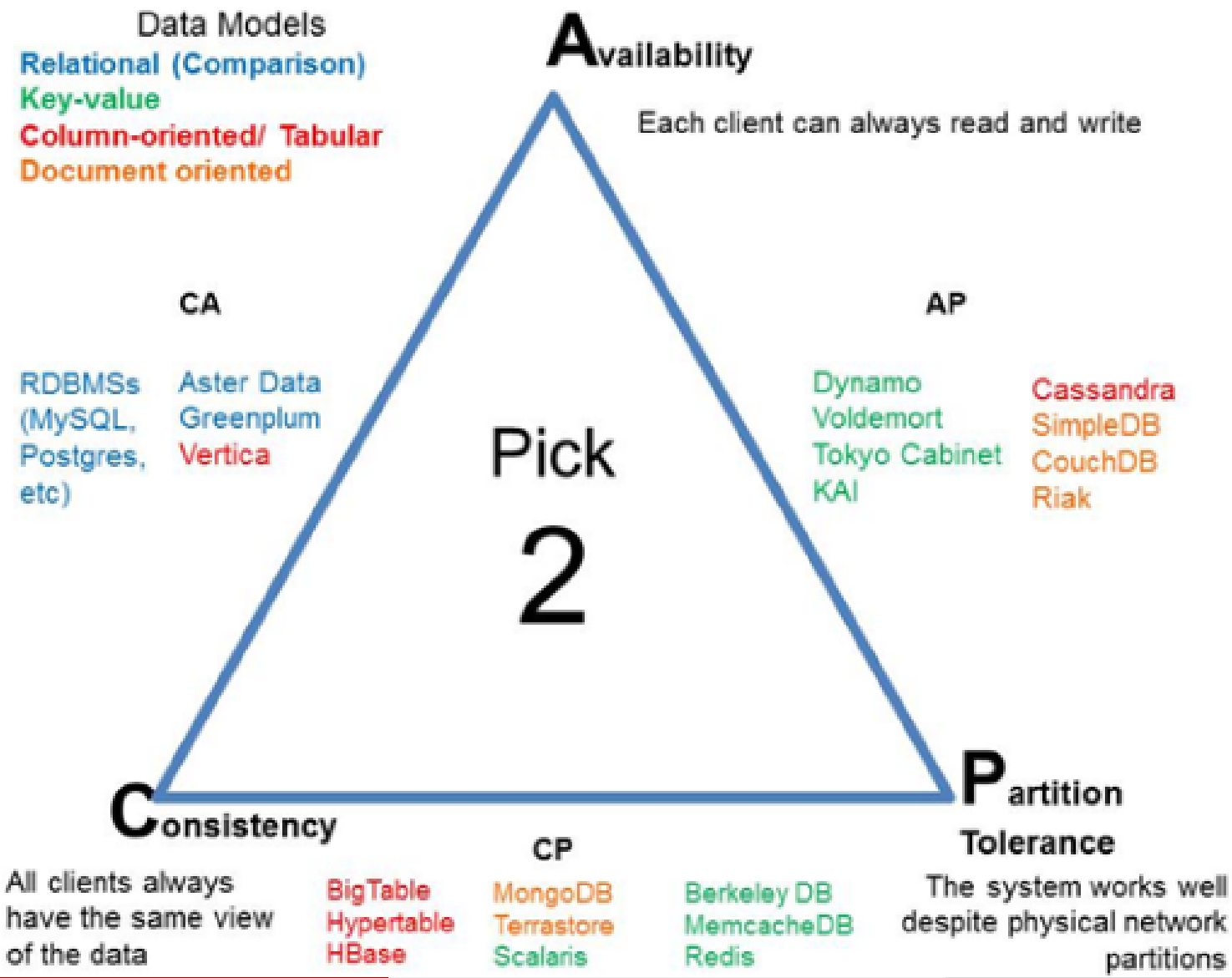
Jazyky pro práci s daty

R - jazyk a prostředí pro statistickou analýzu, existuje grafické rozhraní, interpretovaný, propojitelný s C a TEXem
r-project.cz/about.html

Python - interpretovaný, skriptovací, jednoduchý, otevřený, spousty knihoven (data manilupace - pandas a NumPy; vizualizace - Matplotlib a seaborn; testování hypotéz a modelování - SciPy, scikit-learn, statsmodels)

SQL - neprocedurální, deklarativní, zjednodušené anglické věty, pro práci s relačními databázemi, vstupní brána do světa správy dat

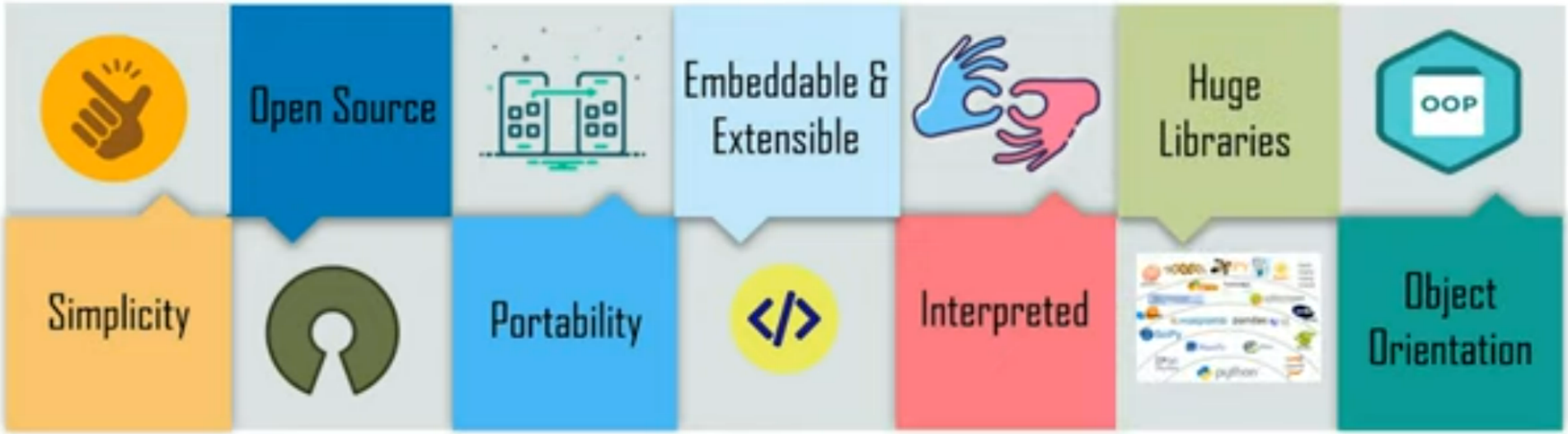
Jazyky v SŘBD



Popularita ŘSBD

https://db-engines.com/en/ranking

Python


Open Source
Embeddable & Extensible
Huge Libraries
OOP
Simplicity
Portability
Interpreted
Object Orientation

Python - Pandas

Pandas - open-source knihovna pro Python, specializovaná pro analýzu dat a jejich manipulaci.
https://data-flair.training/blogs/python-pandas-tutorial/

Pár vět o SQL

- Deklarativní programovací jazyk
- Dotazovací jazyk - umožňuje ovládat databázi prostřednictvím příkazů – dotazů, za pomoci vyhledávacích operátorů
- Programovací jazyk, který se používá na operace s daty v relačních databázových SŘBD
- Příkazem SQL říkáme databázovému stroji, co chceme dělat (např. načíst data za určitých podmínek), ale neříkáme mu, jak má tento příkaz provést

SQL příkazy

čtyři základní skupiny:

- definice dat (DDL) - CREATE, ALTER, DROP, ... vytvářet, upravovat a mazat objekty databáze

SQL příkazy

čtyři základní skupiny:

- definice dat (DDL) - CREATE, ALTER, DROP, ... vytvářet, upravovat a mazat objekty databáze
- manipulace s daty (DML) - SELECT, INSERT, UPDATE, DELETE, ... získávat, ukládat a mazat data v databázi

SQL příkazy

čtyři základní skupiny:

- definice dat (DDL) - CREATE, ALTER, DROP, ... vytvářet, upravovat a mazat objekty databáze
- manipulace s daty (DML) - SELECT, INSERT, UPDATE, DELETE, ... získávat, ukládat a mazat data v databázi
- správa uživatelských rolí a práv (DCL) - GRANT, REVOKE, ... přidělení, odebrání oprávnění

SQL příkazy

čtyři základní skupiny:

- definice dat (DDL) - CREATE, ALTER, DROP, ... vytvářet, upravovat a mazat objekty databáze
- manipulace s daty (DML) - SELECT, INSERT, UPDATE, DELETE, ... získávat, ukládat a mazat data v databázi
- správa uživatelských rolí a práv (DCL) - GRANT, REVOKE, ... přidělení, odebrání oprávnění
- řízení transakcí (GTL) - START TRANSACTION, COMMIT, ROLLBACK ... hromadné provedení několika příkazů

Pár vět o SQL

Zásady pojmenovávání

- začínat písmenem
- dlouhé 1 až 30 znaků
- obsahovat jen A - Z, a - z, 0 - 9, _ (podtržítko), $, a # (Nejsou! Case Sensitivní)
- nesmí být kopií jména dalšího objektu vlastněného stejným uživatelem
- název entity by měl odpovídat obsahu

Nejpoužívanější příkazy

CREATE DATABASE

Nejpoužívanější příkazy

CREATE DATABASE- vytváří novou databázi
ALTER DATABASE

Nejpoužívanější příkazy

CREATE DATABASE- vytváří novou databázi
ALTER DATABASE - upravuje databázi
CREATE TABLE

Nejpoužívanější příkazy

CREATE DATABASE- vytváří novou databázi
ALTER DATABASE - upravuje databázi
CREATE TABLE- vytváří novou tabulku
ALTER TABLE

Nejpoužívanější příkazy

CREATE DATABASE- vytváří novou databázi
ALTER DATABASE - upravuje databázi
CREATE TABLE- vytváří novou tabulku
ALTER TABLE - upravuje tabulku
UPDATE

Nejpoužívanější příkazy

CREATE DATABASE- vytváří novou databázi
ALTER DATABASE - upravuje databázi
CREATE TABLE- vytváří novou tabulku
ALTER TABLE - upravuje tabulku
UPDATE - upravuje data v databázi/tabulce
DELETE

Nejpoužívanější příkazy

CREATE DATABASE- vytváří novou databázi
ALTER DATABASE - upravuje databázi
CREATE TABLE- vytváří novou tabulku
ALTER TABLE - upravuje tabulku
UPDATE - upravuje data v databázi/tabulce
DELETE - maže data z databáze/tabulce
INSERT INTO

Nejpoužívanější příkazy

CREATE DATABASE- vytváří novou databázi
ALTER DATABASE - upravuje databázi
CREATE TABLE- vytváří novou tabulku
ALTER TABLE - upravuje tabulku
UPDATE - upravuje data v databázi/tabulce
DELETE - maže data z databáze/tabulce
INSERT INTO - vkládá nová data do databáze/tabulky
DROP TABLE

Nejpoužívanější příkazy

CREATE DATABASE- vytváří novou databázi
ALTER DATABASE - upravuje databázi
CREATE TABLE- vytváří novou tabulku
ALTER TABLE - upravuje tabulku
UPDATE - upravuje data v databázi/tabulce
DELETE - maže data z databáze/tabulce
INSERT INTO - vkládá nová data do databáze/tabulky
DROP TABLE - maže tabulku

SELECT

Nejpoužívanější příkazy

CREATE DATABASE- vytváří novou databázi
ALTER DATABASE - upravuje databázi
CREATE TABLE- vytváří novou tabulku
ALTER TABLE - upravuje tabulku
UPDATE - upravuje data v databázi/tabulce
DELETE - maže data z databáze/tabulce
INSERT INTO - vkládá nová data do databáze/tabulky
DROP TABLE - maže tabulku

SELECT - výběr dat z databáze

SELECT - Získání záznamů z tabulky

Relační databáze - data uložena v formě záznamů v tabulce

Syntax:

```
SELECT výčet výrazů a atributů FROM tabulka;
```

SELECT - Získání záznamů z tabulky

Relační databáze - data uložena v formě záznamů v tabulce

Syntax:

```
SELECT výčet výrazů a atributů FROM tabulka;
```

Pro všechny atributy - použít výrazu *

Příklad: vypište všechny záznamy z tabulky budova

```
SELECT * FROM budova;
```

SELECT - Získání záznamů z tabulky

Relační databáze - data uložena v formě záznamů v tabulce

Syntax:

```
SELECT výčet výrazů a atributů FROM tabulka;
```

Pro všechny atributy - použít výrazu *

Příklad: vypište všechny záznamy z tabulky budova

```
SELECT * FROM budova;
```

Vždy vypisujme pouze potřebné atributy (čas přenosu, paměť)

```
SELECT adresa, rok_stavby FROM budova;
```

WHERE - podmínka na zobrazený záznam

Mnohdy vyžadujeme vypsat pouze záznamy splňující nějakou logickou podmínku

Syntax:

```
SELECT ...  FROM tabulka WHERE podmínka;
```

WHERE - podmínka na zobrazený záznam

Mnohdy vyžadujeme vypsat pouze záznamy splňující nějakou logickou podmínku

Syntax:

```
SELECT ...  FROM tabulka WHERE podmínka;
```

Příklad: vypište záznamy z tabulky budova, jejichž id je menší než 3

```
SELECT adresa, rok_stavby
    FROM budova
    WHERE id_budova < 3;
```

Podmínku lze komponovat z více výrazů pomocí logických spojek `AND`, `OR` a `NOT`

WHERE - podmínka na zobrazený záznam

Příklad: Vypište budovy, které byly postaveny před rokem 2010 v ulici “Na bojišti”

WHERE - podmínka na zobrazený záznam

Příklad: Vypište budovy, které byly postaveny před rokem 2010 v ulici "Na bojišti"

```
SELECT adresa, rok_stavby
    FROM budova
    WHERE rok_stavby < 2010 AND adresa='Na bojišti';
```

ORDER BY - řazení záznamů

Pořadí záznamů v databázi je "náhodné", můžeme je řadit

Syntax:

```
SELECT ...  FROM tabulka
ORDER BY výraz {DESC|ASC} [, výraz {DESC,ASC}];
```

Příklad: seřaďte budovy podle roku zahájení stavby

ORDER BY - řazení záznamů

Pořadí záznamů v databázi je "náhodné", můžeme je řadit

Syntax:

```
SELECT ...  FROM tabulka
ORDER BY výraz {DESC|ASC} [, výraz {DESC,ASC}];
```

Příklad: seřaďte budovy podle roku zahájení stavby

```
SELECT město, ulice, rok_stavby
    FROM budova
    ORDER BY rok_stavby;
```

řazení:

- vzestupně - nic nebo ASC
- sestupně - DESC

LIMIT OFFSET - stránkování záznamů

Zobrazování mnoho záznamů najednou může být pomalé, proto jej omezujeme na n záznamů

Syntax:

```
SELECT ...  FROM tabulka LIMIT n;
```

Příklad: zobrazte první 3 záznamy z tabulky budova

```
SELECT *
    FROM budova
    LIMIT 3;
```

LIMIT OFFSET - stránkování záznamů

Zobrazování mnoho záznamů najednou může být pomalé, proto jej omezujeme na n záznamů

Syntax:

```
SELECT ...  FROM tabulka LIMIT n;
```

Příklad: zobrazte první 3 záznamy z tabulky budova

```
SELECT *
    FROM budova
    LIMIT 3;
```

ke stránkování slouží klíčové slovo OFFSET od kterého záznamu se bude zobrazovaných n záznamů počítat

Příklad: zobrazte 2 až (2+3) záznam

```
SELECT *
    FROM budova
    LIMIT 3 OFFSET 2;
```

LIKE - jen část textu

Vyhledá textové hodnoty podle části textu.

Syntax:

```
SELECT ...  FROM tabulka WHERE podmínka LIKE '% nebo _';
```

Příklad: zobrazte jména ulic začínajících s tabulky budova

```
SELECT ulice
    FROM budova
    LIKE 's%';
```

LIKE - jen část textu

Vyhledá textové hodnoty podle části textu.

Syntax:

```
SELECT ...  FROM tabulka WHERE podmínka LIKE '% nebo
_';
```

Příklad: zobrazte jména ulic začínajících s tabulky budova

```
SELECT ulice
    FROM budova
    LIKE 's%';
```

Příklad: zobrazte jména ulic obsahující jako druhé písmeno ů s tabulky budova

```
SELECT ulice
    FROM budova
    LIKE '_ů%';
```

JOIN - výběr z více tabulek

Data rozprostřena mezi různé tabulky

Syntax:

```
SELECT tabulka1.atribut1, tabulka1.atribut2,
tabulka1.atribut1, ...
        FROM tabulka1
        JOIN tabulka2
    ON tabulka1.atribut1_id=tabulka2.atribut1_fk
```

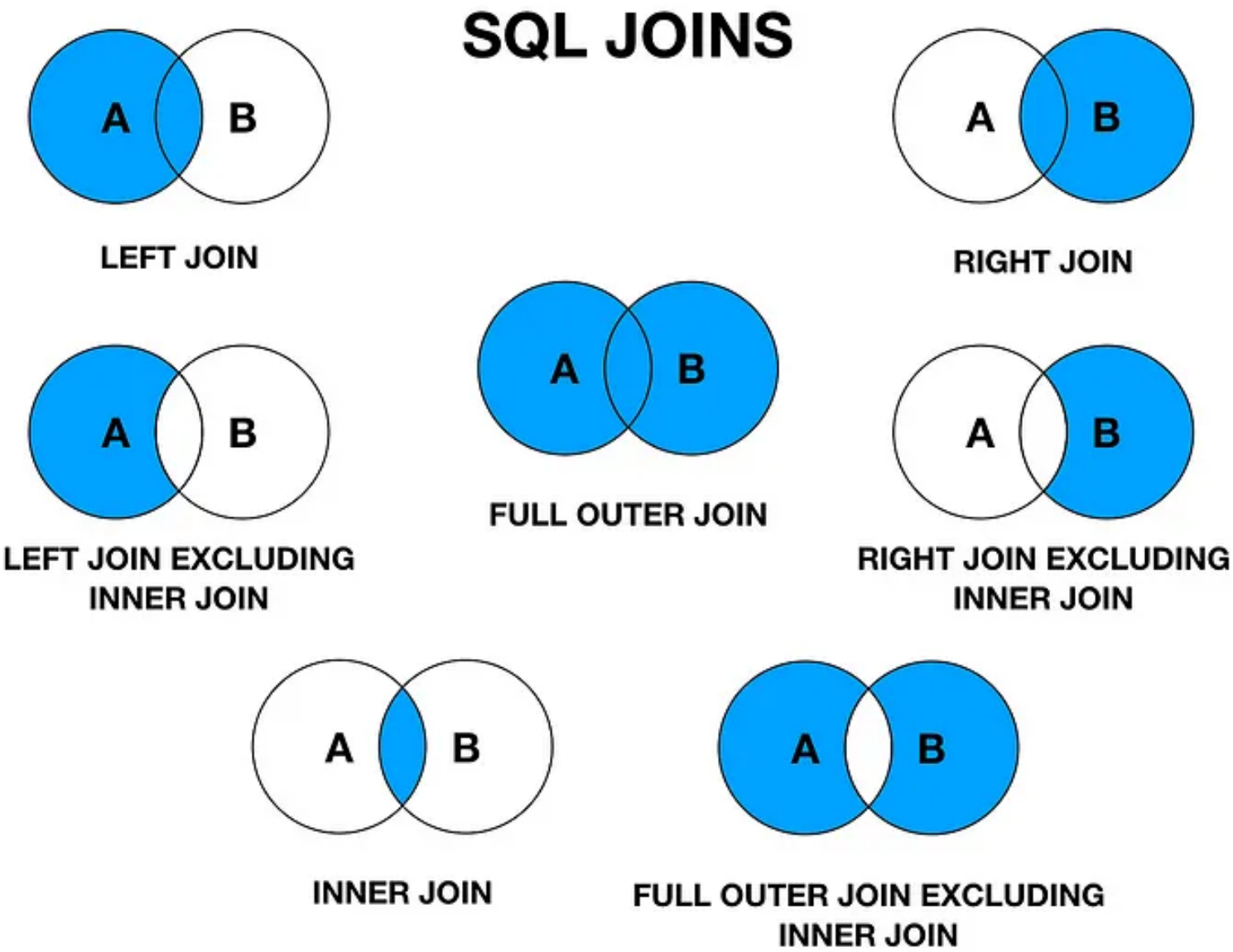
tabulka1: První tabulka
tabulka2: Druhá tabulka
společný sloupec: primární a k němu odpovídající cizí klíč

Spojování tabulek

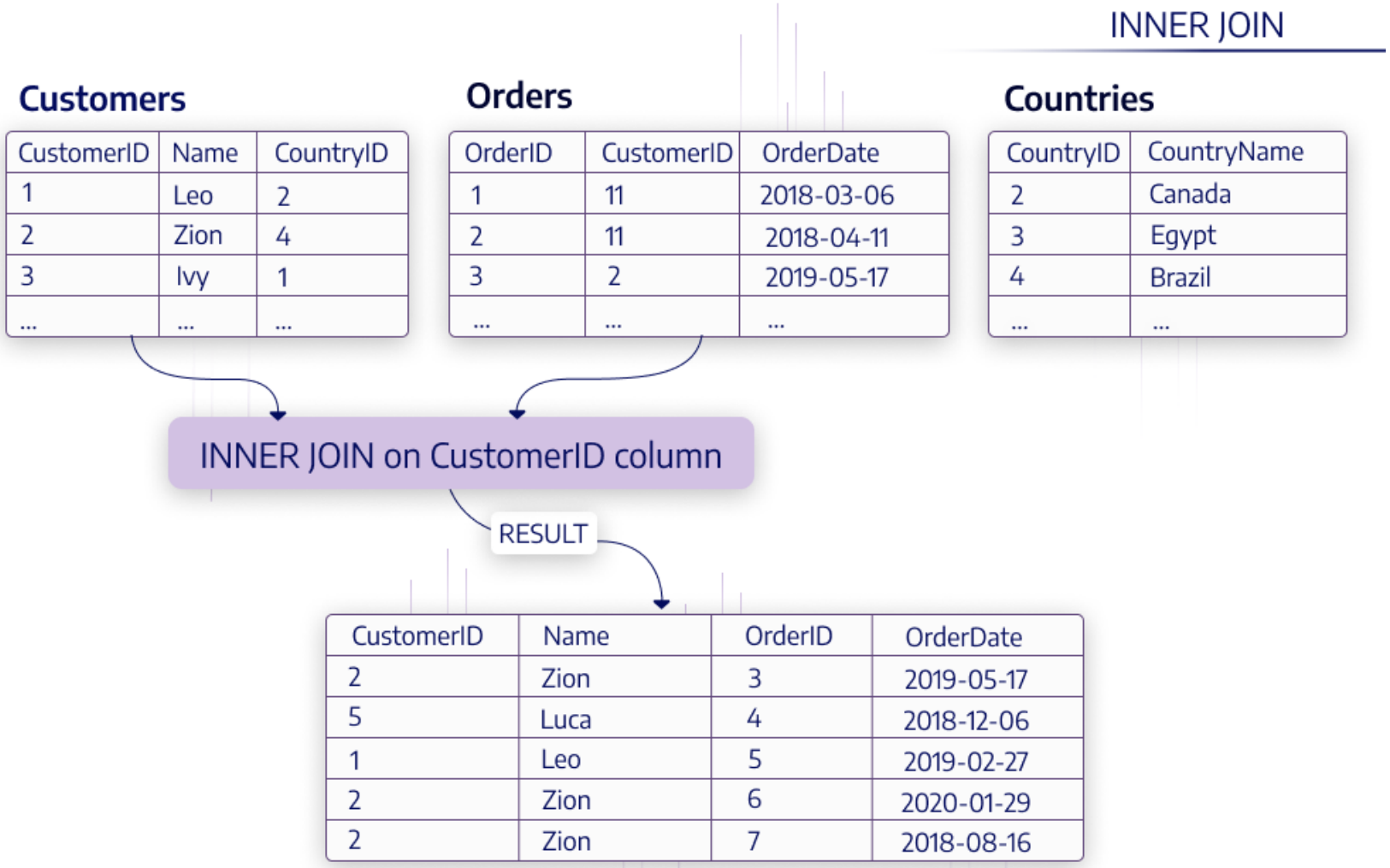
ID_student	jmeno	obor_fk
42	Tereza	12
43	Lenka	12
44	Jakub	14
45	Jan	14
46	Tomas	null

ID_obor	nazev
12	forenzni
13	chemie
14	fyzikalni
15	molekulova

JOIN - Vennovy diagramy



JOIN



*customerdemo database

INNER JOIN - Vnitřní spojování tabulek

Příklad: vyber všechny studenty, co studují fyzikální chemii

```
SELECT student.jmeno, student.ID_student, obor.nazev
    FROM student
    INNER JOIN obor
    ON student.obor_fk=obor.ID_obor
    WHERE obor.nazev="fyzikalni";
```

INNER JOIN - Vnitřní spojování tabulek

Příklad: vyber všechny studenty, co studují fyzikální chemii

```
SELECT student.jmeno, student.ID_student, obor.nazev
    FROM student
    INNER JOIN obor
    ON student.obor_fk=obor.ID_obor
    WHERE obor.nazev="fyzikalni";
```

student.jmeno	student.ID_student	obor.nazev
Jakub	44	fyzikalni
Jan	45	fyzikalni

LEFT JOIN

Tabulka určující počet uvedený záznamů je levá (student)

```
SELECT *
     FROM student
     LEFT JOIN obor
ON student.obor_fk=obor.ID_obor;
```

LEFT JOIN

Tabulka určující počet uvedený záznamů je levá (student)

```
SELECT *
     FROM student
     LEFT JOIN obor
ON student.obor_fk=obor.ID_obor;
```

ID_student	jmeno	obor_fk	ID_obor	nazev
42	Tereza	12	12	forenzni
43	Lenka	12	12	forenzni
44	Jakub	14	14	fyzikalni
45	Jan	14	14	fyzikalni
46	Tomas	null	null	null

LEFT JOIN

```
SELECT *
FROM student, obor
WHERE student.obor_fk=obor.ID_obor;
```

RIGHT JOIN

Tabulka určující počet uvedený záznamů je pravá (obor)

```
SELECT *
     FROM student
     RIGHT JOIN obor
ON student.obor_fk=obor.ID_obor;
```

RIGHT JOIN

Tabulka určující počet uvedený záznamů je pravá (obor)

```
SELECT *
     FROM student
     RIGHT JOIN obor
ON student.obor_fk=obor.ID_obor;
```

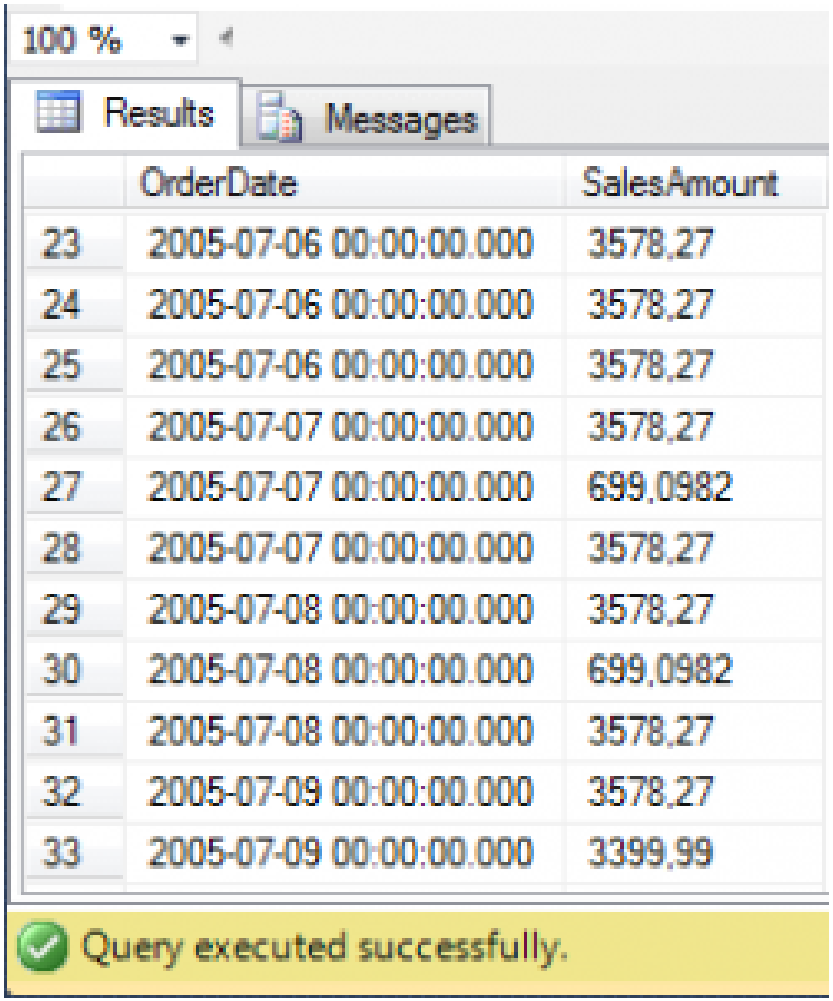
ID_student	jmeno	obor_fk	ID_obor	nazev
42	Tereza	12	12	forenzni
43	Lenka	12	12	forenzni
44	Jakub	14	14	fyzikalni
45	Jan	14	14	fyzikalni
null	null	null	13	chemie
null	null	null	15	molekulova

Agregace

SQL umožňuje vyhodnotit i agregační funkce

AVG() – Vrátí průměrnou hodnotu ze souboru dat
COUNT() – Vrátí počet řádků (hodnot)
MAX() – Vrátí maximální (nejvyšší) hodnotu
MIN() – Vrátí minimální (nejmenší) hodnotu
SUM() – Vrátí součet hodnot

Agregace


100 %

Results | Messages

	OrderDate	SalesAmount
23	2005-07-06 00:00:00.000	3578,27
24	2005-07-06 00:00:00.000	3578,27
25	2005-07-06 00:00:00.000	3578,27
26	2005-07-07 00:00:00.000	3578,27
27	2005-07-07 00:00:00.000	699,0982
28	2005-07-07 00:00:00.000	3578,27
29	2005-07-08 00:00:00.000	3578,27
30	2005-07-08 00:00:00.000	699,0982
31	2005-07-08 00:00:00.000	3578,27
32	2005-07-09 00:00:00.000	3578,27
33	2005-07-09 00:00:00.000	3399,99

Query executed successfully.

Agregace

```
SELECT
        CAST(OrderDate AS DATE)     AS  Datum_Objednavky
        SUM(SalesAmount)            AS  Soucet_trzeb
        AVG(SalesAmount)            AS  Prumer_trzeb
        COUNT(SalesAmount)          AS  Pocet_trzeb
        MAX(SalesAmount)            AS  Maximalni_trzba
        MIN(SalesAmount)            AS  Minimalni_trzba
FROM    AdventureWorksDW2014.dbo.FactInternetSales
GROUP BY  OrderDate
ORDER BY  OrderDate;
```

Agregace - výsledek

Results | Messages

	Datum_Objednavky	Soucet_trzeb	Prumer_trzeb	Pocet_trzeb	Maximalni_trzba	Minimalni_trzba
1	2010-12-29	14477,3382	2895,4676	5	3578,27	699,0982
2	2010-12-30	13931,52	3482,88	4	3578,27	3374,99
3	2010-12-31	15012,1782	3002,4356	5	3578,27	699,0982
4	2011-01-01	7156,54	3578,27	2	3578,27	3578,27
5	2011-01-02	15012,1782	3002,4356	5	3578,27	699,0982
6	2011-01-03	14313,08	3578,27	4	3578,27	3578,27
7	2011-01-04	7855,6382	2618,546	3	3578,27	699,0982
8	2011-01-05	7855,6382	2618,546	3	3578,27	699,0982
9	2011-01-06	20909,78	3484,9633	6	3578,27	3374,99
10	2011-01-07	10556,53	3518,8433	3	3578,27	3399,99
11	2011-01-08	14313,08	3578,27	4	3578,27	3578,27
12	2011-01-09	14134,80	3533,70	4	3578,27	3399,99
13	2011-01-10	7156,54	3578,27	2	3578,27	3578,27
14	2011-01-11	25047,89	3578,27	7	3578,27	3578,27
15	2011-01-12	11230,6282	2807,657	4	3578,27	699,0982

Query executed successfully.

Na závěr

“nic neprogramovat, pouzit MS Access nebo LibreOffice Base. Naucit se to je jednodussi a usetrite firme penize. Je hloupost v dnesni dobe neco takoveho psat rucne.”
https://forum.root.cz/index.php?topic=11634.0

Skripta, stránky a materiály

https://cw.fel.cvut.cz/b202/_media/courses/a5m33izs/sql-select.pdf
https://cw.fel.cvut.cz/old/_media/courses/x33mis/2-dotazovaci-jazyk-sqlv2.pdf...
https://cw.fel.cvut.cz/
https://en.wikipedia.org/wiki/Join_(SQL)#Right_outer_join
https://wikisofia.cz/wiki/SQL
https://janzednicek.cz/sql-agregacni-funkce/
https://www.itnetwork.cz/ms-sql/