

OBSAH

DBS 1

- **T01 DB, Modely; SQL dopyty, WHERE**
- **T02 MySQL - dátové typy a funkcie num. a ret'.**
- **T03 Dátové funkcie; JOIN; Agregácia, GROUP BY**
- **T04-5-6 Dátové modely; Integrita; Návrh DB, ER diagramy**
- **T07 Systémové príkazy; Kaskádovité mazanie**
- **T08 Vnorené dopyty; CASE**
- **T09 Trojh. logika; Kvantif. a NOT; Množ. operácie**
- **T10 Dátové sklady DWH, dátové kocky, pivot**
- **T11 Data science, R, dplyr, NASA**
- **T12 Normalizácia, NF 1. Relačná algebra 1.**

DBS 2 - LS

- a) Relačná algebra 2, NF 2 - ETNF**
- b) Nové témy RDB v rámci Microsoft SQL Server – MSSQL**
- c) NoSQL - MongoDB**

1.Týždeň

DB, Modely, SQL dopyty, WHERE

- 1) DB
- 2) Modely DB
- 3) SQL
- 4) Príklady
- 5) WHERE klauzula <http://dev.mysql.com/doc/refman/5.5/en/select.html>

1) Databáza (DB)

Údaj vs. informácia

Údaj môže byť text, číslo, obrázok, zvuk, video, ...

Informácia je zaznamenaný a overený **údaj (správa)**, ktorý v rámci kontextu

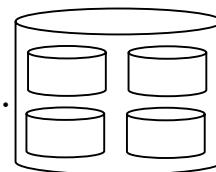
- vedie k zvýšeniu znalosti a zníženiu neistoty a
- ovplyvňuje rozhodovanie, správanie sa.

Nižšie uvedieme niektoré pojmy, ktoré súvisia s databázami.

- **Báza dát, databáza (DB)** – súhrn dát (údajov), ktoré podľa možnosti sú príbuzné a štruktúrované (papierové [kartotéky](#), “clay tables sumer”, banka d.)
- **Počítačová (digitálna) DB** – štruktúrovaná kolekcia **dát** a **metadát**, ktoré sú uložené v počítačovom systéme.
- **DB model** – štruktúra je realizovaná na základe DB modelu pomocou metadát.
 - **relačná algebra**: n-tice a atribúty
 - NoSQL, ...
- **DB server** – DB je uložená a s ňou súvisiace príkazy sú vykonané na DB serveri. Server je v prvom rade software až potom je hardware.
- **DBMS** (DB management system) – riadi ukladanie, editovanie, prezeranie dát a administráciu. Prístup k dátam je možný iba cez DBMS.

Databázový systém (Systém na správu databáz-DBMS) je softvérový systém, ktorý umožňuje efektívnu tvorbu, manipuláciu a správu databáz.

K **funkciám** DBMS patria:



- definície dát (vzťahy, závislosti, integrity, pohľady, atď.)
- manipulácia s dátami (pridávanie, aktualizácia, mazanie, vyhľadávanie, atď.)
- bezpečnosť dát a kontrola ich integrity
- podpora programovacieho jazyka.

Zložky DB-ho systému zahŕňajú:

- hardvér a operačný systém
- DB
- DBMS
- súvisiace softvérové systémy a / alebo aplikácie
- koncoví užívatelia.

Primárne **ciele** databázového systému sú:

- dostupnosť - jednoduchý a rýchly prístup k dátam
- spoľahlivosť - presnosť a konzistencia
- zníženie redundancie (nadbytočnosť, duplicita) a anomálií
- bezpečnosť a ochrana

Databázové systémy nepracujú izolovane – vyžadujú podporu operačného systému, ktorý riadi prístup k prostriedkom počítača. Okrem rôznych desktopových a webových aplikácií (informačných, expertných a pod.) potrebujú aj operačný systém, ktorý zabezpečuje ich beh.

Životný cyklus vývoja databázy:

- prieskum a analýza požiadaviek a špecifikácií
- modelovanie DB
- DB dizajn
- implementácia DB
- použitie DB
- správa DB.

Vo svete existujú rôzne DB systémy:

- Oracle Database, [MySQL](#) - Oracle Corporation
- [SQL Server](#) – MS
- AWS, Google Cloud, Azure
- DB2 - IBM; Ingres
- NoSQL - [MongoDB](#), Elasticsearch, ...

Teória Relačných DB bola vytvorená s [Edgar Frank Codd](#), 1970, IBM.

MySQL

- 1995, MySQL Ab (aktiebolag = stock company), Michael “Monty” **Widenius**, David Axmark, Upsala
- 1996, MySQL
- 2008, Sun Microsystems
- 2009, Oracle

Časová os DB systémov a kľúčové udalosti (2006–2024)

- 2006–2009 — dominancia relačných DB, nástup open-source
- 2009–2014 — NoSQL & Big Data éra
- 2010–2020 — cloud
- 2020–2024 — cloud-first, open-source rast, AI & špecializované DB
- 2023–2024 — nástup nových trendov: vektorové databázy, grafové DB

Rank			DBMS	Database Model	Score		
Sep 2017	Aug 2017	Sep 2016			Sep 2017	Aug 2017	Sep 2016
1.	1.	1.	Oracle	Relational DBMS	1359.09	-8.78	-66.47
2.	2.	2.	MySQL	Relational DBMS	1312.61	-27.69	-41.41
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1212.54	-12.93	+0.99
4.	4.	4.	PostgreSQL	Relational DBMS	372.36	+2.60	+56.01
5.	5.	5.	MongoDB	Document store	332.73	+2.24	+16.74
6.	6.	6.	DB2	Relational DBMS	198.34	+0.87	+17.15
7.	7.	8.	Microsoft Access	Relational DBMS	128.81	+1.78	+5.50
8.	8.	7.	Cassandra	Wide column store	126.20	-0.52	-4.29
9.	9.	10.	Redis	Key-value store	120.41	-1.49	+12.61
10.	10.	11.	Elasticsearch	Search engine	120.00	+2.35	+23.52



Ukazovatele databázového trhu

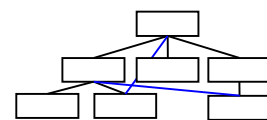
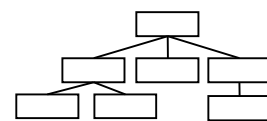
Celkový stav trhu DB v 2006		Stav DB na Windows serveroch v 2008	
Oracle	44.6 %	MS	51.4%
IBM	21.4 %	Oracle	28.8%
MS	16.8 %	IBM	9.8%

2022	2024
≈ 78 % relačné DB	≈ 61-62 % relačné DB
≈ 21 % nerelačné	≈ 38-39 % nerelačné

2) Modely DB

Z historického hľadiska rozlišujeme:

- Súborový model (slabá štruktúra - oddelovače)
- Hierarchický DB model – strom, predstavuje dáta ako množinu relácií, tabuľky sú prepojené ako rodič & potomok väzbou typu **1:n (1vstup/rodič a n výstupov/potomkov)**
- Sieťový DB model – predstavuje dáta ako množinu entít a párových vzťahov medzi nimi, väzba typu **m:n**
- **RDB – relačný** DB je v súčasnosti najviac rozšírený model
- Objektový DB model – trojrozmerná štruktúra, rýchly prístup k prvkom, dedičnosť, dáta+metódy
- XML – semištrukturovaný model dát



Každý model má mať svoj **dopytovací jazyk**: RDB – SQL, XML - XQuery

- NoSQL = Not only SQL
- NoSQL databáza
 - umožňuje ukladanie a rýchle načítanie dát na základe key-value
 - jeho konzistenčný model má slabšie požiadavky na konzistenciu dát (dáta sú spoľahlivé bez protirečení a výsledok operácie je predvídateľný)
 - podporuje horizontálne škálovanie, distribuované ukladanie dát na viacerých serveroch

ACID a CAP transakčné modely

Pre charakterizáciu DB uvedieme dve skupiny vlastností, ACID a CAP, ktoré *by mali byť* v databázových systémoch garantované. Prvú relačné databázy garantujú, druhú v distribuovaných počítačových systémoch nie je možné garantovať.

Množina vlastností ACID zabezpečuje spoľahlivé vykonanie databázových operácií.

ACID (Atomicity, Consistency, Isolation, Durability)

A – prevod dát/**transakcia** je buď úplná alebo žiadna - "all or nothing"

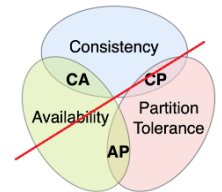
C – stav po transakcii nenarušuje zadefinované pravidlá

I – konkurentné vykonanie transakcie končí stavom, ako keby transakcia bola vykonaná sériovo

D – trvanlivosť stavu (výpadok prúdu, ...)

Databázová transakcia je množina operácií čítania z a zápisu do databázy. Pod **transakciou** rozumieme práve také operácie prevodu dát, ktoré spĺňajú ACID.

CAP (Consistency, Availability, Partition tolerance) – veta Brewer-a hlási, že v distribuovaných počítačových systémoch nie je možné zabezpečiť súbežné garantovanie všetkých troch požiadaviek:



- **konzistencia** - dôslednosť (v danom čase všetky uzly vidia ten istý údaj) – jediná aktuálna kópia dát

- **dostupnosť** (spätné potvrdenie požiadavky ⇔ request) – vysoká dostupnosť dát

- **tolerančná partícia** (aj keď došlo k strate dát, systém beží ďalej)

Dokonalá dôslednosť a dostupnosť v prípade partície je zriedkavá. [Pozri aj ďalšiu charakterizáciu ⇒ PACELC = CAP + Else Latency Consistency](#)

Hnutie NoSQL je o voľbe, ktorá sa zameriava predovšetkým na **dostupnosť** až potom na konzistenciu. Pre NoSQL databázy sa používa aj BASE transakčný model.

Pozri aj ďalšiu charakterizáciu ⇒ PACELC = CAP + Else Latency Consistency

Relačný DB model (RDB)

Ešte stále najviac sú rozšírené relačné databázy (2024: 61% relačné, 38% nerelačné DB), ktoré charakterizujú:

- **n-tice** a **atribúty**, **relácia** – **tabuľka**, **množina**, **domény** a **ohraničenia**,

dvoj/trojhodnotová logika, **NULL** nie je relačný prvok, chýbajúca informácia

- priamy prístup ku každej tabuľke, tabuľky môžu byť prepojené

- jazyky: **relačná algebra**, relačný kalkulus, **SQL**

- relácia / relation vs. hlavička (**schéma**) ⇒ tabuľka je inštancia rel. schémy

- vzťah / relationship medzi tabuľkami pomocou **klúčov**

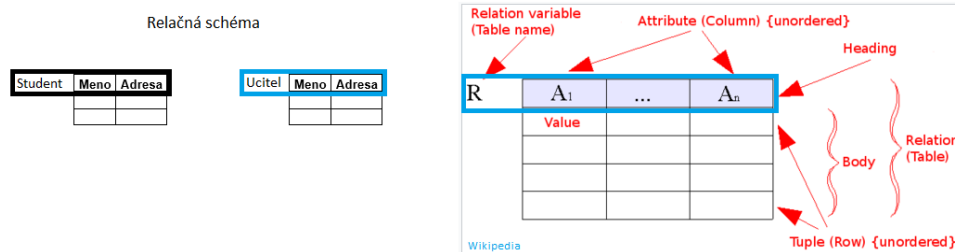
- funkčná závislosť, normalizácia (odstránenie duplicít, anomálií a zabezpečenie integrity)

Základnými pojmami relačnej teórie, **relácia**, **n-tica**, **atribút** v SQL DB zodpovedajú **tabuľka**, **riadok**, **stĺpec**. Codd používal pojem doména, v súčasnosti v computer science sa používa skôr typ.

Relácia¹_r

- i. u Codd je množina n-tíc, teda **podmnožina** karteziánskeho súčinu $A_1 \times A_2 \times \dots \times A_n$.
- ii. je usporiadaný pár (H, T), kde
 - o H je hlavička **r**, **atribúty** H sú atribúty **r** a
 - o T je množina n-tíc (tuples, telo **r**), ktoré všetky majú hlavičku H, **n-tice** T zodpovedajú n-tice **r**.
- iii. je **inštancia** relačnej schémy, pozri 4. týždeň.

Teda **hlavička** A je množina atribútov, **telo** T je množina n-tíc, zodpovedajúcich tej istej hlavičke. Pomenovaná hlavička s obmedzeniami na A je **relačná schéma R**



C.J. Date narába s pojmom **relvar** (relačná premenná), ktorá je premenná, obsahujúca relačnú hodnotu, teda inštanciu relácie.

Zdôrazňujeme, že:

relácia ~ tabuľka (relácie sú reprezentované tabuľkami)

relácia != tabuľka - každá relácia môže byť reprezentovaná ako tabuľka, ale nie každá tabuľka zodpovedá nejakej relácii.

Základná požiadavka Codd, teda relačnej teórie DB, na reláciu:

každý riadok v tabuľke obsahuje hodnotu pre každý stĺpec.

C1	C2
1	a
2	b
1	

C1	C2
1	a
2	b
1	a

- takéto tabuľky v relačnej algebre (RA) nie sú dovolené ($\rightarrow$ null, jedinečnosť riadkov), v SQL áno.

¹ Pripomínáme, že binárna **relácia** R medzi dvomi množinami A a B je **podmnožina** karteziánskeho súčinu $A \times B$, teda $R \subseteq A \times B$. Prvok R sa označuje (a,b), resp. aRb (a je v relácii s b, a súvisí s b).

Funkcia je taká binárna relácia medzi dvomi množinami A a B, pre ktoré platí, že každý prvok A je v relácii iba s jedným prvkom B: $\forall x \in A \exists! y \in B : xRy$.

Rozlišujeme **tri dátové modely** pri príprave a návrhu DB s rôznymi úrovňami abstrakcie

1. konceptuálny model – sémantika (význam)
2. logický model – usporiadať dáta do logických štruktúr – tabuľky; doména
3. fyzický model – typy atribút, indexy (tieto rozhodnutia ovplyvňujú ukladanie)

a **štyri štruktúry** ($\neq$ modely) v DB:

1. konceptuálna štr.
2. logická štr. – tabuľky
3. interná / fyzická štr. – stránky (dát a indexov), 8 KB stránky/pages
4. externá / užívateľská štr. – pohľady

3) SQL

Komunikácia s DB sa uskutočňuje príkazmi jazyka **SQL** (Structured Query Language)

- neprocedurálny, deklaratívny prístup k dátam, D. Chamberlin, 1974 – SEQL

Na rozdiel od **imperatívnych** (píkazových [stav]) jazykov, ako napr. C, Java a C#, SQL je **deklaratívny** jazyk, kde dôraz nie je na presnom predpise príkazov ale na logickej formulácii úlohy.

V rámci SQL rozlišujeme DDL, **DML** (Data Definition & **Manipulation** Language) a **dopyt** (**query** – prezeranie / prehľadávanie)

V jazyku definície dát (**DDL**) môžeme vytvárať, zmeniť a odstrániť databázy; vytvárať, zmeniť a odstrániť tabuľky, definovať stĺpce tabuľky, indexy a vykonať ďalšie kroky, ktoré majú vplyv na štruktúru databázy.

V jazyku na manipuláciu s dátami (**DML**) môžeme pridávať, upravovať a mazať riadky, záznamy a inak manipulovať obsah databázy.

Pomocou dopytov (**SELECT**) prehľadávame obsah tabuliek.

DDL DB / Table	DML Údaje v tabuľke	Dopyt Obsah tabuliek
- CREATE - ALTER - DROP	- INSERT - UPDATE , MERGE - DELETE , ...	SELECT

V SQL príkaze SELECT môžeme používať:

- | | |
|---------------------------------|-----------------------|
| - filtráciu hor. a vert. | WHERE a SELECT |
| - usporiadanie | ORDER BY |
| - agregáciu | SUM, MEAN |
| - zoskupenie | GROUP BY |
| - spojenie | JOIN |
| - vnorené dopyty | SELECT ... SELECT ... |

Jazyk SQL bol štandardizovaný organizáciami

- ANSI (American National Standards Institute)
- ISO (International Standards Organization) <http://www.iso.org/iso/home/standards.htm>

niekoľkokrát

SQL1986, SQL1989

SQL1992, SQL1999

SQL2003, 2008, ...

Zdroje

- Nižšie nájdete (pre zaujímavosť) syntax príkazu CREATE TABLE
<http://dev.mysql.com/doc/refman/5.7/en/create-table.html> <https://docs.microsoft.com/en-us/sql/t-sql/statements/create-table-transact-sql>
- Info [ChatGPT](#)

4) Príklady

<http://www.mysql.com/>

<http://dev.mysql.com/doc/refman/5.7/en/>

Nižšie uvedieme niekoľko ilustračných SQL príkazov v MySQL.

###

0)

###

SELECT **USER()**; # **MySQL**

SELECT 1+1; #Komentár - **MySQL**

SELECT 1*1; -- Komentár s medzerou

SHOW databases;

USE world;

SHOW TABLES FROM world;

SELECT * from city;

###

1) CREATE DATABASE

###

CREATE DATABASE DBmaz;

CREATE DATABASE DBmaz; # NO

DROP DATABASE DBMaz;

CREATE DATABASE DBmaz;

CREATE DATABASE **IF NOT EXISTS** DBmaz;

DROP DATABASE **IF EXISTS** DbMaz;

CREATE DATABASE **IF NOT EXISTS** DbMaz;

```

####
#### 2a) CREATE TABLE
####
USE DBmaz;
CREATE TABLE TabMaz (id int, meno varchar(20));
# CREATE TABLE TabMaz (id int, meno varchar(20)); # NO
DROP TABLE IF EXISTS TabMaz;
CREATE TABLE IF NOT EXISTS TabMaz (id int, meno varchar(20));

####
#### 02b): INSERT pomocou VALUES
####
INSERT TabMaz VALUES(1,'Ja');
INSERT INTO TabMaz(id, meno) VALUES(2,'Ty'), (3,'On');
INSERT TabMaz VALUES
(4,'Ona'),
(5,'Ono');
SELECT * FROM TabMaz;

SELECT DATABASE();           -- current DB
SHOW VARIABLES LIKE "%version%"; -- 8.0.17

-- select db_name(); select @@SERVERNAME; -- SQL Serves

```

5) WHERE klauzula <http://dev.mysql.com/doc/refman/5.5/en/select.html>

```

SELECT [DISTINCT] zoznam
FROM
zoznam/zdroj_tabuliek
[WHERE podmienka]
[GROUP BY zoznam
[HAVING podmienka] ]
[ORDER BY]

```

WHERE klauzula príkazu SELECT špecifikuje
vyhl'adávaciu podmienku / [filter](#)
 pre riadky, ktoré dopyt má vrátiť.

Syntax:

WHERE <vyhl'adávacia podmienka> [**WITH ROLLUP**]

kde vyhl'adávacia podmienka je kombináciou niekoľkých predikátov spájaných logickými operátormi AND, OR a NOT.

Predikát používa atribúty ako premenné a nadobúda hodnoty T, F (v SQL plus NULL) na n-ticiach.

WHERE, podobne ako HAVING, je **filter**: určuje sériu vyhľadávacích podmienok a do výsledku sa zaraďujú iba tie riadky, ktoré spĺňajú dané podmienky.

Vyhľadávacia podmienka môže obsahovať:

- porovnávacie operátory ako: =, >, ...
- intervaly: [NOT] BETWEEN
- zoznamy: [NOT] IN
- zhodu schém: [NOT] LIKE
- IS NULL, IS NOT NULL
- = ALL, > ALL, <= ALL, ... ANY
- kombináciu týchto podmienok vďaka OR, AND, NOT

ALL, ANY<=>SOME, EXISTS `SELECT * FROM T WHERE i+10 >= ALL (SELECT * FROM U);`

ALL a ANY pre každý riadok porovnávajú skalárnu hodnotu so zoznamom alebo množinou hodnôt jediného stĺpca vnoreného dopytu - (SELECT ako vonkajší dopyt môže obsahovať ďalšie SELECTy, ako poddopyty ⇔ vnorené dopyty, pozri 8.týždeň))

ALL / **ANY** vráti TRUE, ak dané porovnanie je
TRUE pre **všetky** dvojice / **aspoň** pre jednu dvojicu.

Syntax:

... WHERE LavStr > **ALL** (VnorDop) ...

teda skalárna veličina LavStr VonkDopytu (ľavá strana porovnania) sa porovnáva s **každou** vrátenou hodnotou VnorDopytu

EXISTS vráti TRUE, ak poddopyt vráti **aspoň** jeden riadok.

Príklady

```
### A) WHERE IN, BETWEEN
###
```

```
DROP TABLE IF EXISTS T;
CREATE TABLE T(i INT);
INSERT INTO T VALUES (1), (2), (6), (7);
```

```
SELECT * FROM T WHERE i IN (1,3,6);           # 1 / 6
SELECT * FROM T WHERE i BETWEEN 3 AND 7;      # 6 / 7  <=>:
SELECT * FROM T WHERE 3<=i AND i <= 7;        # 6 / 7
```

```
### B) ALL, ANY + prvé poddopyty: pozri podrobnejšie Tyzdne 8 a9.
###
```

```
DROP TABLE IF EXISTS T;
CREATE TABLE T(i INT, j INT, k INT);
INSERT INTO T VALUES
```

```
(1, 10, 5),
(2, NULL, 4),
(3, 50, 3),
(4, 50, 2),
(5, 10, 1),
(6, 50, 2);
```

```
ALTER TABLE T DROP COLUMN k; # podrobnejšie Tyzden 6
DROP TABLE IF EXISTS U;
```

Tab. T		Tab. U
i	j	i
1	10	11
2	NULL	12
3	50	13
4	50	14
5	10	
6	50	

```
CREATE TABLE U(i INT);
INSERT INTO U VALUES(11),(12),(13),(14);

SELECT * FROM T WHERE i+10 IN (SELECT * FROM U);    -- vnoreny dopyt
SELECT * FROM T WHERE i+10 >= ANY (SELECT * FROM U);
SELECT * FROM T WHERE i+10 >= ALL (SELECT * FROM U);
SELECT * FROM T WHERE i+10 <= ALL (SELECT * FROM U);
```

Výsledky:

i	j	i	j	i	j	i	j
1	10	1	10	4	50	1	10
2	NULL	2	NULL	5	10		
3	50	3	50	6	50		
4	50	4	50				
		5	10				
		6	50				