1.T DB modely, tabuľky, SQL dopyty

1) DB
2) Modely DB
2) SQL
3) Príklady

1) 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, ktoré podľa možnosti sú príbuzné a štrukturované (papierové kartotéky, "clay tables sumer", banka d)
- Počítačová 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
- DB server – DB je uložená a s ňou súvisiace príkazy sú vykonané na DB servri Server je v prvom rade software az potom je PC.
- DBMS (DB management/riadiaci systém) – riadi ukladanie, editovanie, prezeranie dát a administráciu. Prístup k dátam je možné iba cez DBMS.

Vo svete existujú rôzne DB systémy:

- DB2 - IBM,
- Oracle Database - Oracle Corporation
- SQL Server – MS
- Ingres, MySQL, …

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%

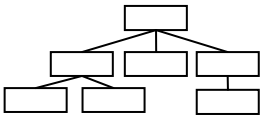
DB (relačná) obsahuje

- Tabuľky dát – stĺpce a riadky (atribúty a záznamy) (.MDF, .NDF)
  - Primárny a cudzí kľúč (integritné obmedzenie)
- Tabuľky indexov a iné objekty, ako Views, uložené procedúry
- Transaction log súbor (.LDF)

Po vykonaní SQL príkazu (DML):

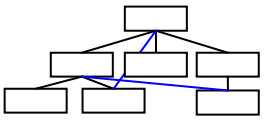
- v pamäti sa modifikuje/ú stránka/y dát
- do LDF sa kopíruje kód dopytu
- neskoršie sa uloží/ia zmenená/é stránka/y do DB na disk

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 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 PC
  - *ľahší* prístup k dátam

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 - 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 po transakcii (výpadok prúdu, ...)

Pod **transakciou** rozumie práve také operácie, ktoré spĺňajú ACID vlastnosti.

CAP (Consistency, Availability, Partition tolerance) veta Brewer-a hlási, že v distribuovaných počítačových systémoch nie je možné zabezpečiť súbežné garantovanie všetkých troch požiadaviek:
- **konzistencia** (v danom čase všetky uzly vidia ten istý údaj)
- **dostupnosť** (spätné potvrdenie požiadavky⇔ request)
- **tolerančná partícia** (aj keď došlo k strate dát, systém beží ďalej)

Relačný DB model (RDB)
V súčastnosti najviac sú rozšírené relačné databázy, ktoré charakterizujú:
- **n-tice** a **atribúty**, **relácia, 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 - hlavička (**schéma**) + telo => tabuľka
  - vzťah / relationship medzi tabuľkami pomocou kľúčov
- funkčná závislosť, normalizácia (odstránenie duplicít a zabezpečenie integrity)

Teória RDB bola vytvorená s Edgar Frank Codd, 1970.

C1	C2
1	a
2	b
1	a

- takáto tabuľka v RA nie je dovolená, v SQL áno

Rozlišujeme **tri dátové modely** pri príprave a návrhu DB s rôznymi úrovňami abstrakcie
- konceptuálny model – sémantika (význam)
- logický model – usporiadať dáta do logických štruktúr – tabuľky; doména
- fizycký model – typy atribút, indexy (tieto rozhodnutia ovplyvňujú ukladanie)

a **štyri štruktúry** (≠ modely) v DB:
- konceptuálna štr.
- logická štr. – tabuľky
- interná / fyzická štr. – stránky (dát a indexov), 8 KB stránky/pages
- 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
SQL na rozdiel od **imperatívnych** jazykov, ako napr. C, Java, C#, 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:

- agregáciu SUM, MEAN
- filtráciu WHERE
- usporiadanie ORDER BY
- zoskupenie GROUP BY
- spojenie JOIN
- vnorené dopyty SELECT … SELECT …

Jazyk SQL bol štandardizovaný organizáciami

- ANSI (American National Standards Institute)
- ISO (International Standards Organization)

niekoľkokrát
SQL1986, SQL1989
SQL1992, SQL1999
SQL2003

T-SQL (Transact-SQL) je rozšírenie štandardu jazyka SQL (MS dialekt SQL)
http://msdn.microsoft.com/en-us/library/ms166026(SQL.90).aspx

Nižšie uvedieme (pre zaujímavosť) syntax príkazu CREATE TABLE.

```
CREATE TABLE
    [ database_name . [ schema_name ] . | schema_name . ] table_name
        ( { <column_definition> | <computed_column_definition> }
        [ <table_constraint> ] [ ,...n ] )
    [ ON { partition_scheme_name ( partition_column_name ) | filegroup
        | "default" } ]
    [ { TEXTIMAGE_ON { filegroup | "default" } ]
[ ; ]

<column_definition> ::=
column_name <data_type>
    [ COLLATE collation_name ]
    [ NULL | NOT NULL ]
    [
        [ CONSTRAINT constraint_name ] DEFAULT constant_expression ]
      | [ IDENTITY [ ( seed , increment ) ] [ NOT FOR REPLICATION ]
    ]
    [ ROWGUIDCOL ] [ <column_constraint> [ ...n ] ]

<data type> ::=
[ type_schema_name . ] type_name
    [ ( precision [ , scale ] | max |
        [ { CONTENT | DOCUMENT } ] xml_schema_collection ) ]

<column_constraint> ::=
[ CONSTRAINT constraint_name ]
{     { PRIMARY KEY | UNIQUE }
        [ CLUSTERED | NONCLUSTERED ]
        [
            WITH FILLFACTOR = fillfactor
          | WITH ( < index_option > [ , ...n ] )
        ]
        [ ON { partition_scheme_name ( partition_column_name )
            | filegroup | "default" } ]
  | [ FOREIGN KEY ]
        REFERENCES [ schema_name . ] referenced_table_name [ ( ref_column ) ]
        [ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
        [ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
        [ NOT FOR REPLICATION ]
  | CHECK [ NOT FOR REPLICATION ] ( logical_expression )
}

<computed_column_definition> ::=
column_name AS computed_column_expression
[ PERSISTED [ NOT NULL ] ]
[
    [ CONSTRAINT constraint_name ]
    { PRIMARY KEY | UNIQUE }
        [ CLUSTERED | NONCLUSTERED ]
        [
            WITH FILLFACTOR = fillfactor
          | WITH ( <index_option> [ , ...n ] )
        ]
        [ ON { partition_scheme_name ( partition_column_name )
            | filegroup | "default" } ]
    | [ FOREIGN KEY ]
        REFERENCES referenced_table_name [ ( ref_column ) ]
        [ ON DELETE { NO ACTION | CASCADE } ]
        [ ON UPDATE { NO ACTION } ]
        [ NOT FOR REPLICATION ]
    | CHECK [ NOT FOR REPLICATION ] ( logical_expression )
]

< table_constraint > ::=
[ CONSTRAINT constraint_name ]
{
    { PRIMARY KEY | UNIQUE }
        [ CLUSTERED | NONCLUSTERED ]
                (column [ ASC | DESC ] [ ,...n ] )
        [
            WITH FILLFACTOR = fillfactor
           |WITH ( <index_option> [ , ...n ] )
        ]
        [ ON { partition_scheme_name (partition_column_name)
            | filegroup | "default" } ]
    | FOREIGN KEY
                ( column [ ,...n ] )
        REFERENCES referenced_table_name [ ( ref_column [ ,...n ] ) ]
        [ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
        [ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
        [ NOT FOR REPLICATION ]
    | CHECK [ NOT FOR REPLICATION ] ( logical_expression )
}

<index_option> ::=
{
    PAD_INDEX = { ON | OFF }
  | FILLFACTOR = fillfactor
  | IGNORE_DUP_KEY = { ON | OFF }
  | STATISTICS_NORECOMPUTE = { ON | OFF }
  | ALLOW_ROW_LOCKS = { ON | OFF}
  | ALLOW_PAGE_LOCKS ={ ON | OFF}
}
```

Transact-SQL Syntax Conventions (Transact-SQL)http://msdn.microsoft.com/en-us/library/ms177563(SQL.90).aspx http://msdn.microsoft.com/en-US/library/ms174979(v=sql.90).aspx

4) Príklady

Nižšie uvedieme niekoľko ilustračných príkladov, ktoré môžete vyskúšať v SQL Server Management Studio.

```
-- a
CREATE DATABASE DBMaz;
CREATE DATABASE DBMaz;-- chyba

-- b)
DROP DATABASE  DBMaz;
DROP DATABASE  DBMaz;  -- chyba

-- c)
CREATE DATABASE DBMaz;
USE DBMaz;
DROP DATABASE  DBMaz;  -- chyba

-- d)OK
USE master;
GO
IF EXISTS (SELECT * FROM sysdatabases WHERE NAME = 'DBMaz')
-- IF DB_ID (N'DBMaz') IS NOT NULL
    DROP DATABASE DBMaz;
GO

-- e)OK
USE DBMaz;
GO

-- f)OK
IF OBJECT_ID ('TabHaha') IS NOT NULL
    DROP TABLE TabHaha;
GO

CREATE TABLE TabHaha
(
  id    INT  NOT NULL PRIMARY KEY,
  hah   VARCHAR(20), juj   DATETIME);
GO

-- g)OK
INSERT INTO TabHaha VALUES (1, 'Vesmirne objekty' , '1937.6.1' );
INSERT  TabHaha VALUES (2, 'Tajomstva prirody', '1967.5.12');
GO
SELECT * FROM TabHaha;
```

Danú tabuľku môžeme vytvoriť nasledujúcimi príkazmi T-SQL:

meno	priezvisko	pohlavie	dátum narodenia	ročník	priemer
Ján	Hraško	muž	12. 7. 1987	1	1,83
Ružena	Šípová	žena	1. 2. 1984	1	1,22
Aladár	Baba	muž	22. 1. 1980	2	2,03
Ferdinand	Mravec	muž	3. 3. 1984	3	1,00
Ján	Polienko	muž	14. 4. 1982	5	2,28
Juraj	Truľo	muž	16. 7. 1979	1	3,00
Jana	Botková	žena	21. 9. 1977	4	1,50
Dana	Botková	žena	21. 9. 1977	4	1,40
Ján	Hlúpy	muž	nezistený	2	3,00
Aladár	Miazga	muž	22. 12. 1987	3	2,06
Mikuláš	Myšiak	muž	6. 6. 1983	5	1,66
Donald	Káčer	muž	7. 10. 1982	5	1,83
Jozef	Námorník	muž	23. 9. 1981	2	2,90

```
USE [master];
GO

IF DB_ID (N'UPJS') IS NOT NULL     -- IF  EXISTS (SELECT name FROM sys.databases WHERE name =
N'UPJS')
BEGIN
  RAISERROR('- DB UPJS Drop ...',0,1)
  DROP DATABASE UPJS
END
GO

RAISERROR('- DB UPJS Create ....',0,1)
CREATE DATABASE UPJS;
GO -- nutne

RAISERROR('- DB UPJS Created.',0,1)

USE UPJS
GO

IF OBJECT_ID (N'dbo.student') IS NOT NULL
BEGIN
  RAISERROR('- Drop Tab student.',0,1)
  DROP TABLE dbo.student
END
GO

CREATE TABLE student -- no primary key
(
      meno        VARCHAR(10),
      priezvisko  VARCHAR(15),
      pohlavie    CHAR(1),
      datum_narodenia DATETIME,
      rocnik            INT,
      priemer     DEC(3,2)
)

INSERT student VALUES('Jan',        'Hrasko',   'm', '1987.7.12', 1, 1.83)
INSERT student VALUES('Ruzena',     'Sipova',   'z', '1984.2.1',  1, 1.22)
INSERT student VALUES('Aladar',     'Baba',     'm', '1980.1.22', 2, 2.03)
INSERT student VALUES('Ferdinand',  'Mravec',   'm', '1984.3.3',  3, 1.00)
INSERT student VALUES('Jan',        'Polienko', 'm', '1982.4.14', 3, 2.28)
INSERT student VALUES('Juraj',      'Trulo',    'm', '1979.7.16', 1, 3.00)
INSERT student VALUES('Jana',       'Botkova',  'z', '1977.9.21', 4, 1.50)
INSERT student VALUES('Dana',       'Botkova',  'z', '1977.9.21', 4, 1.40)
INSERT student VALUES('Jan',        'Hlupy',    'm',  NULL,       2, 3.00)
INSERT student VALUES('Aladar',     'Miazga',   'm', '1987.12.22',3, 2.06)
INSERT student VALUES('Mikulas',    'Mysiak',   'm', '1983.6.6',  5, 1.66)
INSERT student VALUES('Donald',     'Kacer',    'm', '1982.10.7', 5, 1.83)
INSERT student VALUES('Jozef',      'Namornik', 'm', '1981.9.23', 2, 2.90)

SELECT * FROM student
      ORDER BY 3, priezvisko  -- 3=pohlavie
```