

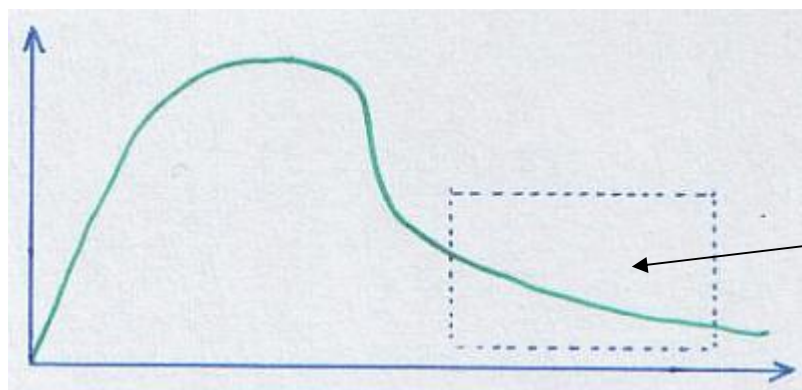
2005-10-11

## RELAČNÍ DATABÁZOVÉ SYSTÉMY

**Vznik relačního datového modelu:** 1969, E. F. Codd, laboratoře IBM, DB2

### Aspekty relačního datového modelu:

- Ü Hodnoty v tabulkách musí být atomické
- Ü Hodnoty musí být skalární – mají pouze jeden rozměr
- Ü Hodnoty ve sloupcích jsou prvky dotyčných domén
- Ü Práce s tabulkami využívá operací výrokové logiky
- Ü Primární klíče pro identifikaci řádek mezi sebou
- Ü Pro propojování tabulek navzájem – cizí klíče



**Křivka „humbuku“** – zájem, popularita, očekávání – zpočátku obrovské, např. očekávání úspor atd., pokud ale nebyli proškoleni zaměstnanci, přestalo se to vyplácet.

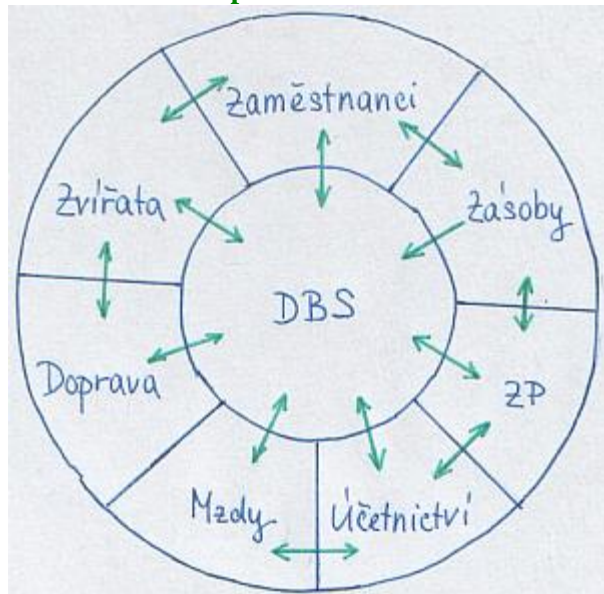
**Obdélník** – přešlo šílení uživatelů, software se ustálil, začal plnit naše požadavky, měli bychom si ho pořizovat v tomto čase.

- Ü V tabulkách lze definovat podmnožiny řádek (operace selekce) a podmnožiny sloupců (projekce).
- Ü Pro práci s více tabulkami – operace spojení.

### Základní pojmy relační databázové teorie:

- Ü Datové soubory chápány jako množiny – dvourozměrné tabulky – **relace**
- Ü Jednotlivé řádky – n-tice sledované problematiky
- Ü Jména sloupců – atributy (vlastnosti)
- Ü Obor přípustných hodnot v sloupcích – **doména**
- Ü Každá věta (**record**) – souhrn všech údajů o jednom objektu dané problémové domény
- Ü Sloupce tabulky reprezentují jednotlivé vlastnosti (atributy) objektů
- Ü Všechny záznamy v rámci 1 tabulky – stejná struktura
- Ü Záznam – množina položek (**pole** – fields) množina vlastností sledovaných objektů

### Sdílení dat mezi aplikacemi IS:



### Podmínky relačnosti tabulky:

- Ü Všechny hodnoty v tabulce musí být elementární – nedělitelné
- Ü Pozice sloupců je nevýznamná – pořadí lze měnit
- Ü Pozice řádků je nevýznamná – pořadí lze měnit
- Ü Sloupce musí být homogenní – obor hodnot tohoto sloupce musí být stejný
- Ü Každý sloupec musí být jednoznačně pojmenován
- Ü Každý sloupec musí být jednoznačně rozlišen – primární, sekundární klíč

### Relační algebra:

- Operace **PROJEKCE** – výběr požadovaných sloupců, výsledkem je relace o p-sloupcích vzniklá z původní o n-sloupcích,  $p < n$
- Operace **SELEKCE** – výběr požadovaných řádků, výsledkem je relace o r-řádcích vzniklá z původní o m-řádcích,  $r < m$
- Operace **spojení JOIN** (na rovnost, nerovnost, vnější) – spojením dvou relací vzniká třetí, která obsahuje všechny kombinace vyhovující zadané podmínce

### Coddova pravidla pro relační model:

- Pravidlo **SŘBD** (systém řízení báze dat) – data spravována pouze pomocí relačních operací
- Pravidlo **informační** – data reprezentována na logické úrovni jako hodnoty relačních tabulek
- Pravidlo **přístupu** – každý údaj logicky dosažitelný pomocí kombinace názvu tabulky, sloupce a hodnoty primárního klíče
- Pravidlo **zpracovatelnosti neznámých hodnot** – ke každé neznámé hodnotě lze dojít prostřednictvím jiných známých hodnot
- Pravidlo **relačního katalogu** – popis celé databáze je na logické úrovni reprezentován jako relační systémový katalog
- Pravidlo **pro jazyk** – pro komunikaci se SŘBD
  - **Definici dat** – DDL
  - **Integritní omezení** – DCL
  - **Manipulaci s daty** – DML
- Pravidlo **pohledů** – SŘBD musí umožňovat konstrukci pohledů
- Pravidlo **operací** – všechny relační operace pracují s tabulkami jako s celky
- Pravidlo **fyzické a logické nezávislosti dat**
- Pravidlo **nezávislosti dat na integritních omezeních** – výsledky operací musí být nezávislé na změnách IO

### Dotazovací jazyk SQL:

- Příkazy **pro definici dat** – definice tabulky v SQL
  - **CREATE TABLE**
  - **ALTER TABLE**
  - **DROP TABLE**
- **CREATE TABLE:**  
CREATE TABLE <jméno-tabulky> (<jméno-sloupce> <datový-typ>  
[NOT NULL]  
[,<jméno-sloupce> <datový-typ> [NOT NULL]]...]  
**NOT NULL** u popisu sloupce se zabezpečí, že všechna pole v daném sloupci nesmí obsahovat hodnotu NULL

### Nápověda v SŘBD Oracle:

```
SQL> help STARTUP
STARTUP
-----
Starts an Oracle instance with several options, including mounting, and opening a database.
STARTUP [FORCE] [RESTRICT] [PFILE=filename] [MOUNT [dbname]
| OPEN [open_options] [dbname] | NOMOUNT]
where open_options has the following syntax:
READ {ONLY | WRITE [RECOVER]} | RECOVER
SQL>
```

### Praktické příklady:

#### Vytvoření relace STUDENT:

STUDENT (CIND,JMENO,BYDLISTE,DATNAR,STIP)

```
SQL> CREATE TABLE student
(cind CHAR (6) NOT NULL,
jmeno VARCHAR (12),
bydliste VARCHAR (10),
datnar DATE,
stip NUMBER (4));
```

Tabulka vytvořena.  
SQL>

#### Naplnění relace STUDENT daty:

SQL> **INSERT INTO student**  
**VALUES ('111/33','Karel Novák','Praha 1','10.1.1980',3500);**  
 1 řádka vložena.  
 SQL> **INSERT INTO student**  
**VALUES ('222/01','Jiří Dvořák','Praha 9','24.7.1984',0);**  
 1 řádka vložena.  
 SQL> atd.

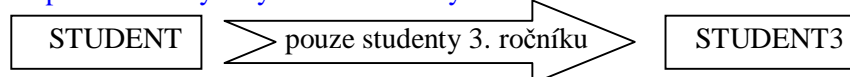
#### Výpis obsahu relace STUDENT:

SQL> **SELECT \* FROM student;**

CIND	JMENO	BYDLISTE	DATNAR	STIP
111/99	Karel Novák	Praha 1	10.01.80	3500
222/01	Jiří Dvořák	Praha 9	24.07.84	0
333/01	Karel Franěk	Kolín 13	12.03.85	2500
444/02	Josef Havlát	Praha 1	15.04.85	2000

SQL>

#### Naplnění tabulky daty ze staré tabulky:



SQL> **INSERT INTO studnet3**  
**SELECT \* FROM student**  
**WHERE rocnik='3';**  
 2 řádek vytvořeno.  
 SQL> **SELECT \* FROM student3;**

CIND	JMENO	BYDLISTE	DATNAR	STIP	ROCNIK
111/99	Karel Novák	Praha 1	10.01.80	3500	3
333/01	Karel Franěk	Kolín 13	12.03.85	2500	3

SQL>

#### Doplnění položky ROCNIK do relace STUDENT:

SQL> **ALTER TABLE student ADD rocnik CHAR(1);**  
 Tabulka změněna.  
 SQL> **DESCRIBE student;**

Název	Nezadáno? Typ
CIND	NOT NULL CHAR(6)
JMENO	VARCHAR2(12)
BYDLISTE	VARCHAR2(10)
DATNAR	DATE
STIP	NUMBER(4)
ROCNIK	CHAR(1)

SQL>

#### Oprava dat v položce STIPENDIUM v relaci STUDENT:

SQL> **UPDATE student SET stip=1000**  
**WHERE cind='222/01';**  
 1 řádka aktualizována.  
 SQL> **SELECT \* FROM student;**

CIND	JMENO	BYDLISTE	DATNAR	STIP
------	-------	----------	--------	------

-----	-----	-----	-----	-----
111/99	Karel Novák	Praha 1	10.01.80	3500
222/01	Jiří Dvořák	Praha 9	24.07.84	1000
333/01	Karel Franěk	Kolín 13	12.03.85	2500
444/02	Josef Havlát	Praha 1	15.04.85	2000

SQL>

#### Oprava dat v položce STIPENDIUM v relaci STUDENT:

UPDATE <jméno-tabulky> SET <jméno-sloupce> = <hodnota>, [<jméno-sloupce> = <hodnota>] ...  
WHERE <výběrová-podmínka>

SQL> UPDATE student SET stip=stip\*2  
WHERE ROCNIK>='3';

2 řádek aktualizováno.

SQL> SELECT \* FROM student;

CIND	JMENO	BYDLISTE	DATNAR	STIP
-----	-----	-----	-----	-----
111/99	Karel Novák	Praha 1	10.01.80	7000
222/01	Jiří Dvořák	Praha 9	24.07.84	0
333/01	Karel Franěk	Kolín 13	12.03.85	5000
444/02	Josef Havlát	Praha 1	15.04.85	2000

SQL>

#### Přejmenování relace STUDENT na STUDENTI:

RENAME <staré jméno tabulky> TO <nové jméno tabulky>]

SQL> RENAME student TO studenti;

Tabulka přejmenována.

SQL> SELECT \* FROM studenti;

CIND	JMENO	BYDLISTE	DATNAR	STIP
-----	-----	-----	-----	-----
111/99	Karel Novák	Praha 1	10.01.80	3500
222/01	Jiří Dvořák	Praha 9	24.07.84	0
333/01	Karel Franěk	Kolín 13	12.03.85	2500
444/02	Josef Havlát	Praha 1	15.04.85	2000

SQL>

#### Přejmenování sloupce STIP na STIPENDIUM:

1. Nejprve je nutno **přidat nový sloupec** s požadovaným jménem a datovým typem shodným s původním sloupcem.

SQL> ALTER TABLE student ADD stipendium NUMBER(4);

Tabulka změněna.

SQL> DESC student;

Název	Nezadáno? Typ
-----	-----
CIND	NOT NULL CHAR(6)
JMENO	VARCHAR2(12)
BYDLISTE	VARCHAR2(10)
DATNAR	DATE
STIP	NUMBER(4)
ROCNIK	CHAR(1)
STIPENDIUM	NUMBER(4)

SQL>

2. Nyní je třeba **zkopírovat hodnoty** z původního sloupce do nového sloupce za všechny řádky.

SQL> UPDATE student SET stipendium=stip;

4 řádek aktualizováno.

SQL> SELECT \* FROM student;

CIND	JMENO	BYDLISTE	DATNAR	STIP	ROCNIK	STIPENDIUM
-----	-----	-----	-----	-----	-----	-----
111/99	Karel Novák	Praha 1	10.01.80	7000	3	7000
222/01	Jiří Dvořák	Praha 9	24.07.84	0	2	0
333/01	Karel Franěk	Kolín 13	12.03.85	5000	3	5000
444/02	Josef Havlát	Praha 1	15.04.85	2000	4	2000

111/99	Karel Novák	Praha 1	10.01.80	7000	3	7000
222/01	Jiří Dvořák	Praha 9	24.07.84	0	2	0
333/01	Karel Franěk	Kolín 13	12.03.85	5000	3	5000
444/02	Josef Havlát	Praha 1	15.04.85	2000	1	2000
SQL>						

#### Zrušení záznamu v relaci STUDENT:

DELETE FROM <jméno-tabulky> [WHERE <výběrová-podmínka>]

Vymaže z uvedené tabulky všechny řádky splňující podmínku zadanou za klíčovým slovem WHERE. Není-li tato podmínka uvedena, budou z dotyčné tabulky vymazány všechny její záznamy.

SQL> **DELETE FROM student WHERE cind='444/02';**

1 řádka vymazána.

SQL> **SELECT \* FROM student;**

CIND	JMENO	BYDLISTE	DATNAR	STIP
-----	-----	-----	-----	-----
111/99	Karel Novák	Praha 1	10.01.80	3500
222/01	Jiří Dvořák	Praha 9	24.07.84	0
333/01	Karel Franěk	Kolín 13	12.03.85	2500
SQL>				