

Databáze

Michal Krátký

Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

2026/2027

Obsah přednášky

1. Předmět Databáze
2. Úvod
3. Konceptuální model
4. Relační datový model
5. SQL

Michal Krátký

- ▶ kancelář: 5.044
- ▶ e-mail: m.kratky@upol.cz
- ▶ web: <https://www.inf.upol.cz/lide/michal-kratky/>

Zápočet

- ▶ **Test na SQL select:** 3 příklady, pro alespoň minimální počet bodů je nutné napsat kompletní řešení dvou příkladů.
- ▶ **Datová analýza pro vlastní vybraný informační systém:** zadání, relační datový model, skript pro create table a skript pro inicializaci databáze.

Proč se bavit o databázových systémech?

- ▶ Proč se bavit o databázových systémech?
 - ▶ Téměř každý informační systém (například informační systém banky nebo univerzity) potřebuje ukládat a dotazovat data, k čemuž používá nějaký databázový systém (například **Microsoft SQL Server**, **Oracle** nebo **PostgreSQL**).
 - ▶ Počet aktivních instalací takových databázových systémů je tedy v desítkách až stovkách milionů.
 - ▶ Téměř každý chytrý telefon nebo webový prohlížeč používá pro uložení a dotazování databázový systém **SQLite**.
 - ▶ Ten má tedy přes 10 miliard aktivních instalací.
- ▶ Datovou vrstvu, která se stará o komunikaci s databázovým systémem, takového informačního systému musí někdo **navrhnout, implementovat nebo spravovat**.
 - ▶ A tento člověk musí mít netriviální znalosti o uložení a dotazování dat, případně správě databázového systému.

Databázový systém

- ▶ **Databázový systém** je aplikace pro ukládání a dotazování dat (např. PostgreSQL nebo SQL Server).
 - ▶ **Databáze** jsou data spravovaná databázovým systémem.
- ▶ Databázový systém používá nějaký **datový model**, který určuje, jakým způsobem jsou data uložena a dotazována (např. relační datový model nebo dokumentový model).
 - ▶ Uživatel přistupuje k databázi pomocí **dotazovacího jazyka** (např. SQL – Structured Query Language).

Historie databázových systémů 1/3

1. **Herman Hollerith** sestrojil pro sčítání lidu v USA v roce 1890 elektromechanické stroje, **tabelátor** a **třídič (sorter)**, které zpracovávaly děrné štítky. Jeden děrný štítek odpovídal jednomu záznamu a obsahoval hodnoty atributů. Hollerithem založená firma se později stala základem společnosti IBM.



2. V 50. letech 20. století vznikly proprietární aplikace využívající záznam dat na **magnetické pásky**, což vyžadovalo sekvenční přístup k datům.

Historie databázových systémů 2/3

3. Na přelomu 60. a 70. let 20. století vznikly databázový systém **IMS**¹ a standard **CODASYL**, které byly založeny na hierarchickém resp. síťovém datovém modelu.

Hlavní vlastnosti:

- ▶ Použití magnetických disků s přímým přístupem (vs. sekvenční přístup u magnetických pásek).
- ▶ IMS a CODASYL nepoužívaly vysokoúrovňový dotazovací jazyk, ale **procedurální přístup**, kdy programátor psal dotaz jako kód, který postupně navigoval záznamy. Změna dat tedy často znamenala i kompletní přepsání takového programu.

¹Původně vytvořený společností IBM pro projekt Apollo v roce 1968. ▶

Historie databázových systémů 3/3

4. **E. F. Codd** z IBM uvedl **relační datový model** v článku: E. F. Codd: *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM, 1970, 13(6), s. 377–387. DOI: 10.1145/362384.362389.

Relační datový model přinesl **deklarativní dotazovací jazyk** Structured Query Language (SQL), ve kterém uživatel píše, co hledá, nikoli jak to hledá (na rozdíl od procedurálního přístupu). Relační databázové systémy se používají dodnes s mnoha vylepšeními a rozšířeními.

5. **NoSQL databázové systémy** – po roce 2005 vznikly databázové systémy založené často na jiných datových modelech, například klíč–hodnota, dokumentový nebo grafový datový model.

Konceptuální model

- ▶ Na začátku návrhu modelu pro danou aplikaci máme zadání, většinou tvořené **zákazníkem**, ve kterém je popsáno, co má být v daném modelu evidováno.
- ▶ Analytik přepíše zadání do **konceptuálního modelu** (lineárního zápisu nebo grafické reprezentace), který obsahuje jednotlivé skupiny objektů reálného světa, tzv. **entitní typy**, a jejich vlastnosti, tzv. **atributy**. Mluvíme o **datové analýze**.
- ▶ Konceptuální model je **nezávislý na konkrétním datovém modelu** (například relačním datovém modelu) nebo **konkrétním databázovém systému** (například PostgreSQL).

Příklad: zadání a lineární zápis konceptuálního modelu

- ▶ **Zadání:** V informačním systému budeme evidovat jednotlivé uživatele. Pro každého uživatele budeme evidovat jedinečné číslo, jméno, příjmení, bydliště (ulici, číslo popisné, město) a email.
- ▶ Základním nástrojem **konceptuálního modelu** je **lineární zápis**, v tomto případě jsem identifikovali jediný **entitní typ** a jeho **atributy**:

```
User(idu, fname, lname, street, street_no, city,  
email)
```

- ▶ **Pravidlo syntaxe lineárního zápisu:** název entitního typu má první znak velký, na rozdíl od názvu atributu, který má první znak malý.

Entitní typ vs entita

- ▶ Mějme **entitní typ** User:
 User(idu, fname, lname, street, street_no, city, email)
- ▶ Pro entitní typ User máme například tyto **entity**, tedy konkrétní uživatele s konkrétními hodnotami atributů:
 - (1, 'Jan', 'Sobota', '17. listopadu', 15, 'Olomouc', 'jan.sobota01@upol.cz')
 - (2, 'Yveta', 'Švecová', 'Hněvotínská', 217, 'Olomouc', 'yveta.svecova01@upol.cz')
 - (3, 'Petr', 'Švec', 'Hněvotínská', 217, 'Olomouc', 'petr.svec02@upol.cz')
- ▶ **Entita** tedy představuje **konkrétní výskyt** (také instanci) entitního typu s vlastními **hodnotami atributů**.
- ▶ **Poznámka:** Ačkoli v konceptuálním modelu neuvažujeme datové typy atributů, při zápisu hodnot atributů entit píšeme řetězce do apostrofů (jednoduchých uvozovek) a čísla bez apostrofů.

Klíč (také kandidátní klíč)

- ▶ Atribut, jehož hodnota identifikuje entitu mezi ostatními entitami daného entitního typu, nazýváme **klíč** (také **kandidátní klíč**).
- ▶ V lineárním zápisu označujeme klíč **podtržením**:
`User(idu, fname, lname, street, street_no, city, email)`
- ▶ Všimněme si, že klíč `idu` je **syntetický**, v doméně reálného světa nemá žádný význam, zatímco atribut `email` je **přirozený klíč**.

Konceptuální model - ER diagram

- ▶ V praxi často používáme grafickou reprezentaci konceptuálního modelu, **Entity-Relationship Diagram (ERD)**, který představil Peter Chen v roce 1976 v článku:

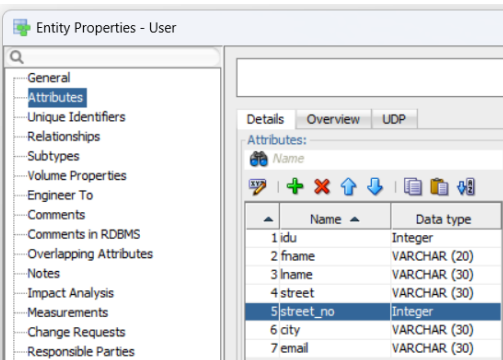
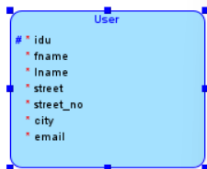
Chen, Peter Pin-Shan. *The entity-relationship model—toward a unified view of data*. ACM Transactions on Database Systems (TODS), 1976, 1(1), 9–36.2

- ▶ Pro kreslení ERD budeme používat Oracle SQL Developer Data Modeler², který nepoužívá **Chenovu notaci** (ta je také poměrně komplikovaná), ale vlastní, tzv. **Barkerovu notaci**, která vznikla v Oracle v 80. letech 20. století.
 - ▶ Barkerova notace je blíže relačnímu datovému modelu a umožňuje tak snadnější převod konceptuálního modelu na relační model.

²<https://www.oracle.com/database/sqldeveloper/technologies/sql-data-modeler/>

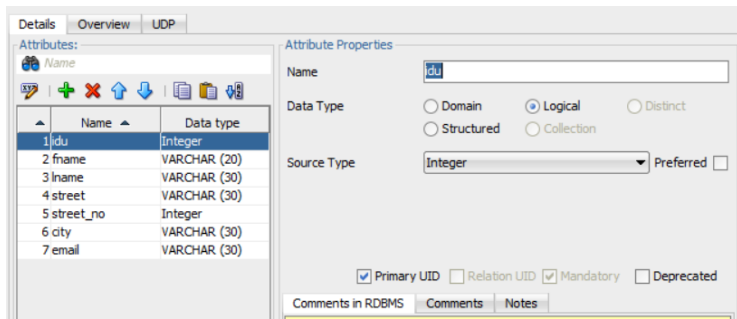
Vlastnosti ERD v Oracle Data Modeler 1/4

- ▶ Vlastnosti ERD v Oracle Data Modeler, které neodpovídají konceptuálnímu modelu, ale jsou blíže relačnímu modelu:
 - ▶ U atributů definujeme datové typy: Integer pro celá čísla nebo VARCHAR(*n*) pro řetězce maximální délky *n*.



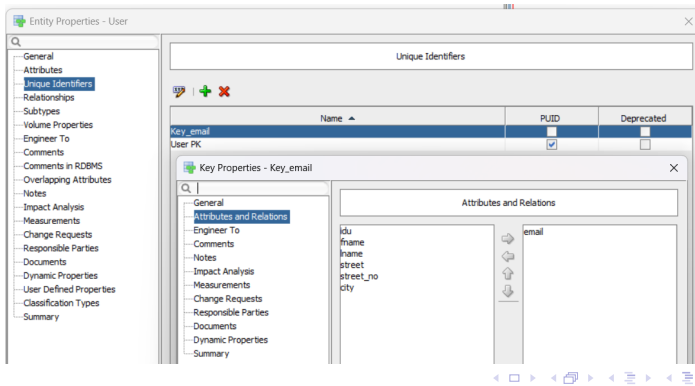
Vlastnosti ERD v Oracle Data Modeler 2/4

- ▶ Vlastnosti ERD v Oracle Data Modeler, které neodpovídají konceptuálnímu modelu, ale jsou blíže relačnímu modelu:
 - ▶ Jeden z klíčů je označen jako **primární identifikátor (Primary UID)**. Později, v relačním datovém modelu, jej budeme označovat jako **primární klíč (primary key)**.



Vlastnosti ERD v Oracle Data Modeler 3/4

- ▶ Vlastnosti ERD v Oracle Data Modeler, které neodpovídají konceptuálnímu modelu, ale jsou blíže relačnímu modelu:
 - ▶ Ostatní klíče kromě primárního identifikátoru, musíme zařadit do seznamu **Unique Identifiers**. Později, v relačním datovém modelu, budou takové atributy označeny jako **jedinečné (unique)**.



Vlastnosti ERD v Oracle Data Modeler 4/4

- ▶ Vlastnosti ERD v Oracle Data Modeler, které neodpovídají konceptuálnímu modelu, ale jsou blíže relačnímu modelu:
 - ▶ Pokud má být hodnota atributu **povinná**, označujeme jej jako **Mandatory**. Později, v relačním datovém modelu, bude takový atribut označen jako **not null**.

Entity Properties - User

Attributes

Details Overview UDP

Attributes:

Name	Data type
1 idu	Integer
2 fname	VARCHAR (20)
3 lname	VARCHAR (30)
4 street	VARCHAR (30)
5 street_no	Integer
6 city	VARCHAR (30)
7 email	VARCHAR (30)

Attribute Properties

Name:

Data Type: ☐ Domain ☒ Logical ☐ Distinct

☐ Structured ☐ Collection

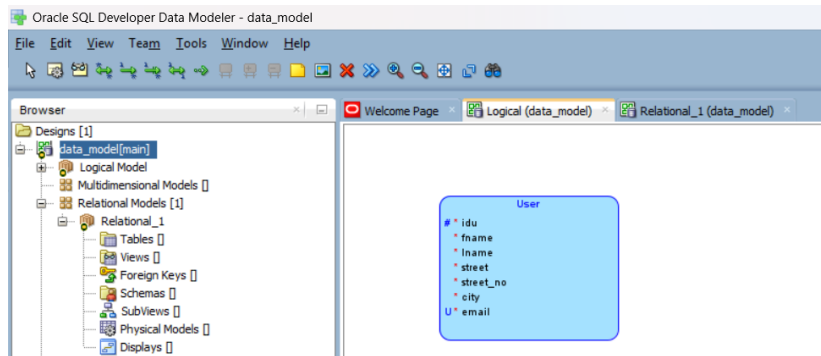
Source Type: Preferred ☐

Size:

Units:

☐ Primary UID ☐ Relation UID ☒ Mandatory ☐ Deprecated

Konceptuální model - ERD



Označení atributů entitního typu:

- ▶ primární identifikátor: #
- ▶ povinný atribut: *
- ▶ klíč: U

Relační datový model

- ▶ Entitní typ je v relačním datovém modelu reprezentován **relačním schématem**: množinou **atributů** a jejich **domén**³:

$$RS = \{A_1 : D_1, A_2 : D_2, \dots, A_k : D_k\}.$$
- ▶ Entity jsou pak prvky **relace** R relačního schématu RS :

$$R \subseteq D_1 \times D_2 \times \dots \times D_k.$$
- ▶ Abstraktním dotazovacím jazykem relačního datového modelu je **relační algebra**.
- ▶ Implementace relačního datového modelu používají **tabulky** namísto relací, prvek relace pak označujeme jako **záznam**. Tabulky jsou multimnožiny – na rozdíl od relací (množin) mohou obsahovat **duplicitní záznamy**.
 - ▶ Dotazovací jazyk relačního datového modelu **SQL** pracuje s tabulkami, nikoli s relacemi.

³Doména je množina všech možných hodnot atributu a určuje jeho datový typ atributu.

SQL – Structured Query Language

- ▶ Historie jazyka SQL začíná v 80. letech 20. století, postupně bylo uvedeno několik specifikací, **SQL92** představuje základní specifikaci.
 - ▶ **Relační databázové systémy** (RDBS) pak implementují jednotlivé specifikace SQL.
- ▶ SQL se skládá, mimo jiné, z těchto částí:
 - ▶ **Jazyk pro definici data – JDD** (Data definition language – DDL): vytvoření tabulky dle schématu (`create table`), změna schématu tabulky (`alter table`), zrušení tabulky (`drop table`) apod.
 - ▶ **Jazyk pro manipulaci s daty – JMD** (Data manipulation language – DML): vložení, aktualizace a smazání záznamu (`insert`, `update`, `delete`).
 - ▶ **Jazyk pro dotazování dat** (Data query language – DQL): `select`.

PostgreSQL

- ▶ SQL se budeme učit v databázovém systému PostgreSQL⁴.
- ▶ Projekt POSTGRES založil v roce 1986 profesor Michael Stonebraker z Kalifornské univerzity v Berkeley.
- ▶ V roce 1996, po opuštění akademické půdy, byl projekt přejmenován na PostgreSQL.
- ▶ Dnes je PostgreSQL největším open-source databázovým systémem s odhadovaným počtem desítek až stovek miliónů aktivních instalací.



⁴<https://www.postgresql.org/>

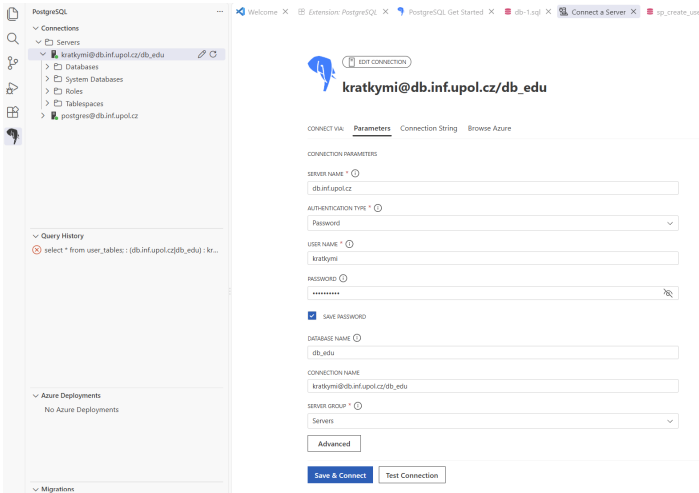
Proč zrovna PostgreSQL?

- PostgreSQL je jeden z nejpobulárnějších databázových systémů, viz hodnocení popularity na <https://db-engines.com/en/ranking>

Rank	Rank			DBMS	Database Model	Score		
	Sep 2026	Aug 2026	Sep 2025			Sep 2026	Aug 2026	Sep 2025
1.	1.	1.		Oracle	Relational, Multi-model 	1122.27	-1.17	-48.36
2.	2.	2.		MySQL	Relational, Multi-model 	844.82	+2.50	-46.95
				Microsoft SQL Server	Relational, Multi-model 	698.69	+4.13	-18.63
3.	3.	3.		PostgreSQL	Relational, Multi-model 	683.26	-1.32	+26.08
4.	4.	4.		MongoDB	Document, Multi-model 	381.31	-3.57	+0.81
5.	5.	5.		Snowflake	Relational	211.61	-4.28	+21.42
6.	6.	6.		Databricks	Multi-model 	168.34	+2.53	+44.27
7.	7.	↑9.		Redis	Key-value, Multi-model 	158.06	+1.55	+12.89
8.	8.	↓7.		IBM Db2	Relational, Multi-model 	110.57	-1.50	-13.61
9.	9.	↓8.		SQLite	Relational	97.05	+1.11	-10.83
10.	↑11.	↑11.						

Klient PostgreSQL

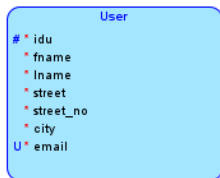
Pro zasílání příkazů na PostgreSQL budeme používat rozšíření PostgreSQL (autora Microsoft) pro Visual Studio Code⁵.



⁵<https://code.visualstudio.com/download>

Příkaz JDD: create table 1/5

Příkaz create table jazyka pro definici dat SQL pro ERD obsahující jediný **entitní typ** User:



```
create table "User"
(
    idu int primary key,
    fname varchar(20) not null,
    lname varchar(30) not null,
    street varchar(30) not null,
    street_no int not null,
    city varchar(30) not null,
    email varchar(30) unique not null
);
```

Spuštění příkazu (zaslání příkazu na serveru a jeho spuštění) provedeme, ve VS Code + rozšíření PostgreSQL, označením a stiskem Shift + Enter.

Příkaz JDD: create table 2/5 – zápis příkazů SQL

User	
#	* idu
*	* fname
*	* lname
*	* street
*	* street_no
*	* city
U	* email

```
create table "User"
(
  idu int primary key,
  fname varchar(20) not null,
  lname varchar(30) not null,
  street varchar(30) not null,
  street_no int not null,
  city varchar(30) not null,
  email varchar(30) unique not null
);
```

Často se můžeme setkat se zápisem **SQL příkazů velkými znaky** (**CREATE TABLE** vs **create table**), jedná se o pozůstatek černobílých terminálů, kdy nebyla jiná možnost zvýraznění syntaxe. Na přednáškách a cvičení budou pro SQL příkazy použity malé znaky: je to praktické a dnešní zvýraznění syntaxe je založeno na barvách, případně jiných řezech písma.

Příkaz JDD: create table 3/5

User	
#	* idu
	* fname
	* lname
	* street
	* street_no
	* city
U	* email

```
create table "User"
(
  idu int primary key,
  fname varchar(20) not null,
  lname varchar(30) not null,
  street varchar(30) not null,
  street_no int not null,
  city varchar(30) not null,
  email varchar(30) unique not null
);
```

- **Název tabulky** by měl být User, nicméně v některých DBS se jedná o vyhrazené slovo a musíme tedy použít **"User"**.

Příkaz JDD: create table 4/5

User	
#	* idu
*	* fname
*	* lname
*	* street
*	* street_no
*	* city
U	* email

```
create table "User"  
(  
    idu int primary key,  
    fname varchar(20) not null,  
    lname varchar(30) not null,  
    street varchar(30) not null,  
    street_no int not null,  
    city varchar(30) not null,  
    email varchar(30) unique not null  
);
```

Definice atributu obsahuje:

- ▶ **název atributu**: např. idu nebo fname,
- ▶ **datový typ**, případně jeho délku: např. int nebo varchar(30),
- ▶ **integritní omezení**: např. primary key nebo not null.

Vybrané datové typy SQL

► Celočíselné datové typy:

- integer (zkráceně int): nejčastěji⁶ je v databázových systémech implementován jako 32bitové znaménkové číslo s rozsahem $[-2\,147\,483\,648; 2\,147\,483\,647]$, tedy `int32_t` z C++.
- `smallint`/`bigint`: nejčastěji jsou implementovány jako `int16_t` resp. `int64_t`.

► Řetězce:

- `varchar(n)`: řetězec proměnné délky n znaků. Řetězcové hodnoty v SQL zapisujeme do apostrofů, např. `'Jan'`.
- `char(n)`: řetězec pevné délky n znaků.

► Datové typy pro reálná čísla:

- `real` / `double precision` (někdy také `float` / `double`): implementace nejčastěji odpovídá datovým typům `float` resp. `double` z C++.

⁶Závisí na konkrétní implementaci.

Integritní omezení atributů tabulky

- ▶ **Integritní omezení (IO)** atributů tabulky představuje omezení hodnot atributů, které vyplývá z reálného světa.
- ▶ Rozlišujeme, mimo jiné, tato **integritní omezení (IO)**:
 - ▶ **Doménová**: určují povolené hodnoty atributů, např. `not null` znamená, že hodnota atributu nemůže být prázdná (`null`⁷). Implicitní IO je `null`, tedy hodnota atributu může být prázdná.
 - ▶ **Entitní**: zajišťují jednoznačnou identifikaci záznamu v tabulce (v tabulce nemůže existovat jiný záznam se stejnou hodnotou takové atributu):
 - ▶ **primary key** – jeden z klíčů entitního typu z konceptuálního modelu je vybrán, v relačním datovém modelu, jako primární klíč. Hodnota tohoto atributu **nemůže** být `null`.
 - ▶ **unique** – ostatní klíče jsou označeny jako jedinečné. Hodnota tohoto atributu **může** být `null`.

⁷**Pozor:** v případě čísel: `0` $\neq$ `null`, v případě řetězců: prázdný řetězec `"` není `null`.

Integritní omezení unique bez not null

- ▶ Pokud je integritní omezení atributu **unique** bez **not null**, můžeme vložit více záznamů s hodnotou null?
 - ▶ **Na první pohled ne**, protože druhý záznam s hodnotou null porušuje podmínku jedinečnosti hodnoty atributu v tabulce.
 - ▶ **V SQL ale** výsledek porovnání `null = null` není `true`, ale neznámá hodnota (`unknown`), protože SQL nepoužívá dvouhodnotovou logiku, ale **tříhodnotovou logiku**.
 - ▶ Vložením dalších záznamů s hodnotou atributu null tedy neporušujeme podmínku jedinečnosti.
 - ▶ **Poznámka:** ne všechny databázové systémy dodržují v tomto specifikaci SQL, např. Microsoft SQL Server, kde u atributu `unique` můžeme vložit pouze jeden záznam s hodnotou atributu null.

Příkaz JDD: create table 5/5

User	
#	* idu
	* fname
	* lname
	* street
	* street_no
	* city
U	* email

```
create table "User"
(
  idu int primary key,
  fname varchar(20) not null,
  lname varchar(30) not null,
  street varchar(30) not null,
  street_no int not null,
  city varchar(30) not null,
  email varchar(30) unique not null
);
```

Proč je jako **primární klíč vybrán** syntetický klíč idu namísto přirozeného klíče email? Později si ukážeme, že je výhodnější volit číselný datový typ s menší velikostí, než je řetězec o maximální délce 30 znaků (int vs varchar(30)).

Dotazování dat: select

- ▶ Tabulky dotazujeme pomocí příkazu **select**, základní syntaxe je jednoduchá, reálné dotazy, ale mohou mít desítky řádků:

```
select <projekce> from <table_name> where
    <selekce>;
```

- ▶ **Dotaz:** všechny záznamy z tabulky User⁸:

```
select * from "User"; -- výsledek je prázdný
```

- ▶ **Dotaz:** počet záznamů z tabulky User:

```
select count(*) from "User"; -- 0
```

- ▶ Stejný dotaz, ale sloupec s počtem záznamů pojmenujeme pocet:

```
select count(*) as pocet from "User";
```

	pocet	↕
1	0	

⁸ Jednořádková poznámka v SQL je --

Příkaz JMD: insert

- ▶ Příkazem jazyka pro modifikaci dat **insert** vložíme záznam do tabulky:

```
insert into "User"(idu, fname, lname, street,  
    street_no, city, email)  
values (1, 'Jan', 'Sobota', '17. listopadu',  
    15, 'Olomouc', 'jan.sobota01@upol.cz');
```

- ▶ Pokud dodržíme pořadí atributů z příkazu **create table**⁹, můžeme jednoduše napsat:

```
insert into "User" values (2, 'Yveta',  
    'Švecová', 'Hněvotínská', 215, 'Olomouc',  
    'yveta.svecova01@upol.cz');
```

⁹create table tedy neodpovídá přesně pojmu relační schéma, jelikož to je definováno jako množina, zatímco v případě create table jde o posloupnost s definovaným pořadím atributů.

Příkaz JMD: insert a integritní omezení (primary key)

- Databázový systém hlídá **splnění integritních omezení** definovaných **pro danou tabulku**, například jedinečnost hodnoty primárního klíče pomocí primary key:

```
insert into "User" values (1, 'Jan', 'Sobota',  
    '17. listopadu', 15, 'Olomouc',  
    'jan.sobota@upol.cz');  
-- záznam vložen
```

```
insert into "User" values (1, 'Yveta',  
    'Švecová', 'Hněvotínská', 215, 'Olomouc',  
    'yveta.svecova@upol.cz');  
-- duplicate key value violates unique  
    constraint "User_pkey"  
-- DETAIL: Key (idu)=(1) already exists.
```

Příkaz JMD: insert a integritní omezení (unique)

- ... nebo jedinečnost hodnoty atributu pomocí unique:

```
insert into "User" values (1, 'Jan', 'Sobota',  
    '17. listopadu', 15, 'Olomouc',  
    'jan.sobota01@upol.cz');
```

-- *záznam vložen*

```
insert into "User" values (2, 'Yveta',  
    'Švecová', 'Hněvotínská', 215, 'Olomouc',  
    'jan.sobota01@upol.cz');
```

-- *duplicate key value violates unique
constraint "User_email_key"*

-- *DETAIL: Key (email)=(jan.sobota01@upol.cz)
already exists.*

Příkaz JMD: insert a integritní omezení (not null)

- Databázový systém hlídá **splnění integritních omezení** definovaných pro **daný atribut**, například vynucení povinné hodnoty atributu pomocí not null:

```
insert into "User"(idu, fname, lname, street,
    street_no, city)
values (1, 'Jan', 'Sobota', '17. listopadu', 15,
    'Olomouc');
-- null value in column "email" of relation "User"
-- violates not-null constraint
-- DETAIL: Failing row contains (1, Jan, Sobota,
-- 17. listopadu, 15, Olomouc, null).
```

```
insert into "User" values (2, 'Yveta', 'Švecová',
    'Hněvotínská', 215, 'Olomouc', null);
-- null value in column "email" of relation "User"
-- violates not-null constraint
-- DETAIL: Failing row contains (2, Yveta, Švecová,
-- Hněvotínská, 215, Olomouc, null).
```

Dotazování tabulky

► Stejné dotazy:

```
select * from "User";
-- výsledek:
-- idu fname lname      street      street_no city      email
-- 1   Jan   Sobota  17. listopadu 15          Olomouc
--      jan.sobota01@upol.cz
-- 2   Yveta Švecová Hněvotínská  215         Olomouc
--      yveta.svecova01@upol.cz
```

```
select count(*) from "User"; -- count: 2
```

► Dotaz: záznam z tabulky User s idu=1 (použijeme selekci v klauzuli where na idu=1):

```
select * from "User" where idu=1;
-- výsledek:
-- idu fname  lname      street      street_no      city
--      email
-- 1   Jan    Sobota  17. listopadu 15          Olomouc
--      jan.sobota01@upol.cz
```

Příkaz JMD: update

- ▶ Příkazem jazyka pro modifikaci dat **update** změníme hodnotu atributu.
- ▶ **Aktualizace:** pro záznam s idu=2 změníme hodnotu atributu street_no na 217¹⁰:

```
update "User" set street_no=217 where idu=2;
```

- ▶ **Dotaz:**

```
select * from "User" where idu=2;
-- výsledek:
-- idu  fname  lname  street  street_no
--      city   email
-- 2      Yveta  Švecová Hněvotínská      217
--      Olomouc yveta.svecova01@upol.cz

-- původní hodnota byla 215
```

¹⁰Všimněme si **selekce** v klauzuli where, podobně jako u příkazu select.

Příkaz JMD: delete

- ▶ Příkazem jazyka pro modifikaci dat **delete** smažeme záznam z tabulky.

- ▶ **Mazání:** záznamu s idu=1 z tabulky "User"¹¹:

```
delete from "User" where idu=1;
```

- ▶ **Dotaz:**

```
select count(*) from "User"; -- 1
```

- ▶ **Mazání:** všech záznamů z tabulky "User":

```
delete from "User"
```

- ▶ **Dotaz:**

```
select count(*) from "User"; -- 0
```

¹¹Všimněme si **selekce** v klauzuli where, podobně jako u příkazu select.

Reference



H. Garcia-Molina, J.D. Ullman, J. Widom. Database Systems: The Complete Book, Prentice Hall, ISBN 0-13-031995-3, 2002.