# Syntaxe SQL

## Jazyk SQL

Zkratka SQL znamená Structured Query Language (strukturovaný dotazovací jazyk). Také se někdy nazývá „sequel“. Tento jazyk je neprocedurální, množinově zaměřený. Tzn. Že SQL zpracovává data jako množiny (skupiny). SQL je de facto standardním jazykem pro získávání a manipulaci dat z relačních databází. Dovoluje např. upravovat strukturu databáze, zabezpečení, oprávnění uživatelů, tvořit dotazy apod.

Mezi jeho implementace řadíme MySQL, Oracle, Sybase, Microsoft SQL Server apod.

## Dotazy v jazyce SQL

Každý příkaz (nebo série příkazů, které vykonáváme jako jeden) je nazýván dotaz (SqueryL). V závislosti na tom, jakého typu tento dotaz je (tedy jaký příkaz používáme) rozlišujeme typy dotazů na: • příkazy pro návrh a administraci databáze,   
• příkazy pro manipulaci s daty. Způsobu zápisu příkazů se říká SYNTAX.

# Příkazy pro manipulaci s daty

## Příkazy SELECT a FROM

**Syntax:**

**SELECT jmena\_sloupců**

Příkaz SELECT nám dovoluje určit, které ze sloupců v tabulce chceme zobrazit.

Nepracuje samostatně. Je nutné ještě příkaz rozšířit o udání toho, z které tabulky chceme

informace zobrazit.

**Syntax:**

**SELECT jmeno\_sloupce FROM jmeno\_tabulky**

Takto zapsaný příkaz už je korektní. Můžeme ale také použít zástupné znaky:

**SELECT \* FROM Knihy**

Výsledkem vykonání tohoto dotazu bude zobrazení všech údajů se všemi sloupci   
  
Podmínky v dotazech SELECT

Chceme-li vyhledat jen určitou část dat z databáze (ne všechny záznamy), je nutné

omezit výběr nějakou podmínkou. Jedním z příkazů, který dovoluje do dotazu typu **SELECT**

zavést podmínku je klauzule **WHERE.**

**Syntax:**

**SELECT jmeno\_sloupce FROM jmeno\_tabulky WHERE omezující\_podmínka**

Toto nám dovolí ovlivňovat výpis dat v závislosti na omezující podmínce. Příklad:

**SELECT \* FROM Knihy WHERE JmenoKnihy=‘Umění ve 20. Století‘**

Operátor **LIKE**

Pokud potřebujeme ovlivnit záznamy, které mají odpovídat nějakému vzoru (nikoliv

přesnému výrazu), použijeme příkaz LIKE:

**Syntax:**

**SELECT \* FROM Knihy WHERE JmenoKnihy LIKE ‘%ostrov%‘**

Tento dotaz vyhledá všechny záznamy z tabulky Knihy, kteří mají v názvu slovo

ostrov.

# Logické a množinové operátory

Chceme-li kupříkladu vyhledat záznamy, které něco nesplňují, lze použít tzv. logické operátory.

**Syntax:**

**SELECT \* FROM Knihy WHERE JmenoKnihy NOT LIKE ‘%ostrov%‘**

Tento dotaz vyhledá všechny záznamy z tabulky Knihy, kteří nemají v názvu slovo

ostrov.

Hlavní logické operátory jsou: **AND**, **OR**, **NOT**.

Množinové operátory známe tyto:

**UNION**: výsledky budou bez duplicitních řádků,

**INTERSECT**: výsledkem jsou jen ty řádky, které jsou v obou dotazech,

**MINUS**: vrátí ty řádky, které jsou v prvním dotazu a nejsou v druhém.

**Syntax:**

**SELECT \* FROM tabulka WHERE podmínka   
UNION  
SELECT \* FROM tabulka WHERE podmínka2**

Operátor **IN** a **BETWEEN**

Tyto příkazy obvykle jen zkracují zápisy pomocí jiných operátorů. Jsou tedy

nahraditelné.

**Syntax:**

**SELECT \* FROM tabulka WHERE jmeno\_sloupce IN (602,145,458,154)**

Tento dotaz se omezí na ty řádky, které mají ve sloupci jmeno\_sloupce hodnoty

602,145…

**Syntax:**

**SELECT \* FROM tabulka WHERE cena BETWEEN 100 AND 1000**

Tento dotaz se omezí na ty řádky, které mají ve sloupci cena hodnoty mezi 100 a

1000.

**Klauzule ORDER BY**

Pokud potřebujeme výsledky nějak seřadit. Kupříkladu pokud chceme seřadit seznam

účtů podle zůstatků:

**Syntax:**

**SELECT \* FROM Ucty ORDER BY Zustatek**

Tento dotaz by z tabulky Ucty vyhledat všechny záznamy a seřadí je podle zůstatku

(jeden ze sloupců).

**Sestupnost řídíme pomocí příkazů DESC a ASC.**

## Klauzule GROUP BY a agregační funkce

Pokud chceme výsledek dotazu spojovat (agregovat) v jiný výsledek, použijeme

klauzuli **GROUP** **BY**. Příklad:

**Syntax:**

**SELECT Platce, SUM(Zaplaceni) FROM Zakaznici**

Tento dotaz vyhledá a seskupí záznamy podle pole Platce. Každý plátce bude

zobrazen jednou a poli SUM(Zaplaceni) bude zobrazen součet ze všech záznamů od stejného

plátce. **Další agregační funkce jsou AVG (průměr), MIN, MAX, COUNT apod**.

**Klauzule HAVING**

Pokud chceme předcházející příklad upravit tak, aby byl zobrazen jen takový záznam,

u kterého je součet všech plateb větších než 10000, pak požijeme jen tento příklad:

**SELECT Platce, SUM(Zaplaceni) FROM Zakaznici HAVING SUM(Zaplaceni) < 10000**

# Vytvoření databáze

Jazyk SQL umožňuje i vytvořit databázi. Tento příkaz ovšem nefunguje v programu

Microsoft Access.

**Syntax:**

**CREATE DATABASE Zákazníci**

Tento příkaz vytvoří databázi Zákazníci.

## Tvorba tabulek

Pro vytvoření tabulky v databázi použijeme příkaz:

**CREATE TABLE název**

**{**

**Sloupec1 DatovýTyp [NOT NULL],**

**Sloupec1 DatovýTyp [NOT NULL]**

**…**

**}**

Datové typy máme (viz. literatura) např, CHAR (alfanumerická data), DATE (datum a

čas), LONG (an řetězce s délkou až 2Gb) apod.(např. http://msdn.microsoft.com/enus/library/ms714540%28v=vs.85%29.aspx)

Hodnota NULL (nebo NOT NULL) dovoluje (nebo

nedovoluje) ponechat při zadávání pole prázdné.

**Změna struktury tabulky – ALTER TABLE**

**Příkaz ALTER TABLE dovoluje měnit strukturu tabulky i po jejím vytvoření**. Například,

chceme-li změnit pojmenování pole (sloupce):

**ALTER TABLE tabulka CHANGE stare\_pojmenovani\_slouce nove\_pojmenovani\_slouce**

## Smazání tabulky a databáze – DROP TABLE a DROP DATABASE

**Syntax:**

**DROP TABLE Zakaznici**

**DROP DATABASE Zakaznici**

Těmito příkazy smažeme tabulku resp. celou databázi.