**„EU peníze školám“**

**Projekt DIGIT – digitalizace výuky na ISŠTE Sokolov**

**reg. č. CZ.1.07/1.5.00/34.0496**

|  |  |
| --- | --- |
| **III/2 Inovace a zkvalitnění výuky prostřednictvím ICT** | **VY\_32\_INOVACE\_1\_6\_12** |
| Název vzdělávacího materiálu | Základy SQL – jednoduché dotazy |
| Jméno autora | Mgr. Miloslav MLÁDEK |
| Tématická oblast | Databázové systémy |
| Vzdělávací obor | 68-46-M/01 Veřejnosprávní činnost |
| Předmět | Informační a komunikační technologie |
| Ročník | 2., 3. |
| Rozvíjené klíčové kompetence | Kompetence k učení   * posoudí vlastní pokrok a určí překážky či problémy bránící učení; * kriticky zhodnotí výsledky;   Kompetence k řešení problémů   * osvědčené postupy aplikuje při řešení obdobných problémových situací; * sleduje vlastní pokrok při zdolávání problémů. |
| Průřezové téma | Informační a komunikační technologie |
| Časový harmonogram | 1 vyučovací hodina |
| Použitá literatura a zdroje | Databáze, modely dat, relační algebra, SŘBD, SQL, normální formy [online]. [cit. 2013-02-12]. Dostupné z: http://is.muni.cz/th/143327/fi\_b/bc\_prace\_vitxx.txt  Entity-relationship model. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-02-12]. Dostupné z: http://cs.wikipedia.org/wiki/Entity-relationship\_model  SKŘIVAN, Jaromír. SQL - tvorba tabulek. In: [online]. [cit. 2013-02-12]. Dostupné z: http://interval.cz/clanky/sql-tvorba-tabulek/ |
| Pomůcky a prostředky | pc, internet, sw |
| Anotace | Databázové systémy, základy jazyka SQL – vazby mezi tabulkami. |
| Způsob využití výukového materiálu ve výuce | Výklad, pracovní list |
| Datum (období) vytvoření vzdělávacího materiálu | 2/2013 |

*Tento výukový materiál je plně v souladu s Autorským zákonem (jsou zde dodržována všechna autorská práva). Pokud není uvedeno jinak, autorem textů   
a obrázků je Mgr. Miloslav Mládek.*

**Jednoduché dotazy**

Pro získávání jakýchkoliv informací slouží jediný příkaz SELECT, který je ze všech příkazů jazyka SQL nejsložitější, má nejbohatší strukturu a nejvíce možností.

Pro potřebu příkladů na procvičení příkazu SELECT, si vytvoříme novou databázi. Vyzkoušíme databázový model evidence knihovny, který se bude skládat z pěti tabulek. Budeme evidovat informace o knihách a jejich výtiscích, nakladatelstvích a autorech. Tabulky jsou popsané v následujícím přehledu (pozn.: v databázovém světě se pro pořadové číslo velmi často používá označení ‚id‘ – jako identifikátor).

**Přehled tabulek evidence knihovny:**

Tabulka AUTORI:

CREATE TABLE `autori` (

`id\_autor` int(10) unsigned NOT NULL auto\_increment,

`jmeno` varchar(60) NOT NULL,

`prijmeni` varchar(60) NOT NULL,

`rok\_narozeni` int(4) default NULL,

PRIMARY KEY (`id\_autor`))

Tabulka KNIHY:

CREATE TABLE `knihy` (

`id\_kniha` int(10) unsigned NOT NULL auto\_increment,

`nazev` varchar(60) NOT NULL,

`id\_autor` int(10) unsigned default NULL,

`rok\_vydani` int(4) default NULL,

PRIMARY KEY (`id\_kniha`))

Tabulka VYTISKY:

CREATE TABLE `vytisky` (

`id\_vytisk` int(10) unsigned NOT NULL default '0',

`id\_kniha` int(10) unsigned default NULL,

`vypujcena` enum('a','n') NOT NULL default 'n',

PRIMARY KEY (`id\_vytisk`))

**Syntaxe příkazu SELECT**

**Kompletní syntaxe vypadá následovně:**

SELECT seznam sloupců

FROM seznam tabulek

[WHERE restrikce]

[GROUP BY výrazy pro seskupení]

[HAVING doplňující podmínky pro skupinu]

[UNION 'select']

[INTERSECT 'select']

[MINUS 'select']

[ORDER BY dle čeho třídit]

Uvedená definice je rekurzivní z toho důvodu, že pomocí klíčových slov UNION, INTERSECT a MINUS, můžeme dělat sjednocení vrácených sloupců z uvedených dotazů, nebo průnik či doplněk. V dnešním díle se budeme o příkazu SELECT bavit na zjednodušené úrovni, jejíž syntaxe by mohla vypadat takto:

SELECT seznam sloupců

FROM seznam tabulek

[WHERE restrikce]

[ORDER BY dle čeho třídit]

**SELECT seznam sloupců**

Nejprve si řekneme, co znamená pojem projekce. Projekci si lze představit jako zobrazení konkrétní položky (sloupce). Jednotlivé řádky tabulky nám mohou vystupovat jako n-tice hodnot, např. v tabulce AUTOŘI jeden řádek je pětice různých hodnot (id, jméno, příjmení, titul, prostřední). Kdybychom se v tabulce zajímali pouze o tituly, udělali bychom projekci na složku titul, zapisujeme P[titul]. Kdyby nás zajímalo jméno a příjmení, udělali bychom projekci P[jméno, příjmení].

Jako seznam sloupců uvedeme právě projektované sloupce, tj. sloupce, jejichž hodnoty nás zajímají. Obecně zde můžeme uvést výčet sloupců z více tabulek. Pokud by se některé názvy sloupců v různých tabulkách opakovaly, pak je potřeba název sloupce uvést ve tvaru jméno\_tabulky­.jméno\_sloupce. Jména sloupců oddělujeme čárkou.

**FROM seznam tabulek**

V této části uvedeme tabulky, ze kterých chceme data získávat. Všechny sloupce, zadané za klíčovým slovem SELECT, musí patřit do zde uvedených tabulek. Názvy tabulek oddělujeme také čárkou. Zatím budeme pracovat s jednou tabulkou.

**Jednoduchý výpis tabulky**

Pro výpis obsahu tabulky použijeme příkaz SELECT v jeho nejjednodušší podobě. Následující příklady provedou výpis základních údajů z tabulek:

SELECT id\_kniha, nazev

FROM knihy

SELECT \*

FROM autori

První SELECT zobrazí obsah sloupců id\_kniha a nazev tabulky KNIHY. Druhý příkaz zobrazí obsah celé tabulky AUTORI. Znak hvězdička nahrazuje názvy všech sloupců v tabulce a to ve stejném pořadí, jak jsou sloupce v tabulce vytvořeny.

**Restrikce**

S tímto pojmem jsme se již několikrát setkali při aktualizaci dat v tabulkách – v příkazech UPDATE a DELETE jsme uváděli část WHERE s podmínkami, které nám říkaly, na které řádky se má příkaz aplikovat. Právě ono omezování řádků pomocí podmínek se nazývá restrikce.

**WHERE restrikce**

Podmínka se skládá z kombinace logických výrazů a logických spojek AND, OR a NOT a kulatých závorek. Logické výrazy můžeme sestavovat z názvů sloupců, z relačních operátorů a speciálních klíčových slov. V jednotlivých částech logického výrazu mohou vystupovat aritmetické výrazy se základními matematickými operátory a dále řetězcové výrazy s operátory pro řetězce. Přehled základních operátorů ukazuje následující tabulka:

a>b, a<b, a>=b;, a<=b, a=b

IS NULL, IS NOT NULL

LIKE 'K%', LIKE 'L\_\_\_\_', LIKE '\_b%k'

AND, OR, NOT,

(, )

Každý databázový systém má svá různá rozšíření, toto je úplný základ, který by měl většinou dodržovat každý systém. Zastavím se na chvíli u některých operátorů. Klíčové slovo IS NULL se používá pro zapsání podmínky, že nějaká položka (sloupec) má nedefinovanou hodnotu (uživatel ji nezadal). IS NOT NULL slouží pro opačný případ, kdy se ptáme, zdali položka má definovanou hodnotu.

Operátor LIKE slouží pro porovnávání řetězců a navíc nabízí možnost jakýchsi jednoduchých regulárních výrazů. LIKE ‚K%‘ znamená řetězec, který začíná na písmeno ‚K‘ (znak ‚%‘ nahrazuje libovolný počet znaků, i nula). LIKE ‚L\_\_\_\_‘ je zápis řetězce, který začíná na písmeno ‚L‘ a má právě 5 znaků (znak ‚\_‘ nahrazuje libovolný jeden znak). Příklad LIKE ‚\_b%k‘ je jen kombinací dvou předchozích. Říká, že řetězec obsahuje na druhém místě písmeno ‚b‘, následované libovolným počtem znaků, které pak končí písmenem ‚k‘. Následuje pár příkladů s užitím restrikce:

**Příklady**

Všichni autoři, kteří se narodili mezi roky 1900 a 2000 včetně:

SELECT jmeno, prijjmeni

FROM autori

WHERE rok\_narozeni >= 1900

AND rok\_narozeni <= 2000

Příjmení všech autorů, jejichž křestní jména začínají na písmeno ‚N‘:

SELECT příjmení

FROM autoři

WHERE jméno LIKE 'N%'

**ORDER BY dle čeho třídit**

Uvedení klíčového slova ORDER BY slouží pro třídění výstupu. Třídění výstupních řádků podle hodnot určitého sloupce nebo sloupců je velmi užitečné a často využívané. Sloupce, podle kterých chceme třídit, uvedeme jako seznam za ORDER BY. Jednotlivé sloupce od sebe oddělíme opět čárkou. Základní třídění probíhá lexikograficky u řetězcových položek. Třídění dle čísel, data apod. probíhá podle našeho očekávání. Pokud byste v řetězcovém třídění zaznamenali nějaké anomálie typu, že písmeno ‚ch‘ se bere jako dvojici písmen ‚c‘ a ‚h‘, nebo že by systém špatně zařazoval slova, která by začínala na nějaké diakritické písmeno, bude problém v lokálním nastavení vašeho databázového systému. Systému je nutné říct, že vaše prostředí je české.

**Příklad**

Chceme seznam všech autorů s jejich jménem, příjmením a rokem narození, jejichž příjmení začíná na písmeno 'C' nebo 'D', tříděno podle příjmení (když mají stejné příjmení, tak ještě podle jména):

SELECT jmeno, prameni, rok\_narozeni

FROM autori

WHERE prameni LIKE 'C'

OR prameni LIKE 'D'

ORDER BY prijmeni, jmeno

Nejčastěji jsou hodnoty tříděny vzestupně, od nejnižší po nejvyšší. Pokud bychom chtěli provést třídění sestupně (inverzní), použijeme třídění v následujícím tvaru:

ORDER BY dle čeho třídit DESC

Použití modifikátoru DESC by mělo fungovat ve většině databázových systémů. Kromě klíčového slova DESC existuje i ASC, které je pro základní vzestupné třídění, jeho uvedení je však nepovinné a zpravidla se tedy ani neuvádí.

**Pracovní list**

V prostředí aplikace adminer (ke stažení na http://www.adminer.org/cs/) zapište SQL příkaz, kterým vytvoříte tabulky AUTORI, KNIHY a VYTISKY jejichž struktura je uvedena ve výkladu.

Naplňte tabulky daty (můžete použít soubor autori.sql).

1. Zapište příkaz SQL, kterým vypíšete z tabulky *autori* všechny autory, jejichž křestní jméno je 'Jan'. Ve výpisu zobrazte sloupce j*meno*, *prijmeni* a *rok\_narozeni*. Setřiďte výstup podle příjmení a roku narození.
2. Zapište SQL příkaz, kterým vypíšete z tabulky *knihy* všechny knihy, jejichž název začíná na písmeno 'L'. Ve výpisu bude pouze sloupec *nazev.* Výstup setřiďte podle abecedy sestupně. Vyzkoušejte tento SQL příkaz pro jiné počáteční písmeno názvu knihy.
3. Zapište SQL příkaz, kterým vypíšete z tabulky *knihy* všechny knihy, jejichž název obsahuje 7 znaků. Ve výpisu bude pouze sloupec *nazev.* Výstup setřiďte podle abecedy vzestupně. Vyzkoušejte tento SQL příkaz pro jiný počet znaků v názvu knihy (například 3, 6, 8, 9 a další hodnoty).
4. Zapište SQL příkaz, kterým vypíšete z tabulky *knihy* všechny knihy, jejichž název obsahuje kdekoliv v názvu řetězec znaků 'na'. Ve výpisu bude pouze sloupec *nazev.* Výstup setřiďte podle abecedy sestupně. Vyzkoušejte tento SQL příkaz pro jiné kombinace řetězců znaků ('oku', 'ok', 'čer', 'če').