Základní přehled SQL příkazů

SELECT

Základní použití

Příkaz SELECT slouží k získání dat z tabulky nebo pohledu v požadované podobě.

Získání všech řádků a sloupců z tabulky

```
SELECT * FROM Person.Contact
```

Vrací všechny všechny řádky a sloupce (*) z tabulky Person.Contact (Person je schéma do kterého tabulka Contact patří, je-li tabulka v jiném schématu než dbo, je třeba uvádět v jejím názvu toto schéma)

Získání všech řádků a vybraných sloupců z tabulky

```
SELECT FirstName, LastName FROM Person.Contact
```

Vrací všechny řádky z tabulky, ale jen zvolné sloupce

Aplikace filtru na vracené řádky

```
SELECT * FROM Person.Contact WHERE Title = 'Mr.'
```

Vrací pouze řádky, kde hodnota sloupce Title je rovna řetězci 'Mr.' Podmínky lze seskupovat logickými operátory (AND, OR, NOT)

```
SELECT * FROM Person.Contact WHERE Title = 'Mr.' AND Suffix = 'Jr.'
```

Práce s řetězci v podmínce

Pokud potřebujeme použít zástupné symboly v řetězci, musíme použít klíčové slovo LIKE

```
SELECT * FROM Person.Contact WHERE Title NOT LIKE 'M%'
```

Dotaz vrací všechny řádky z dané tabulky, kde ve sloupci Title není (NOT) řetězec začínající písmenem M, které je následované 0-n libovolnými symboly.

% - 0..n libolných symbolů

_ - 1 libolný symbol

[abc] - množina symbolů (aplikován 1x)

[^abc] -všechny symboly, kromě určených (aplikován 1x)

```
SELECT * FROM Person.Contact WHERE ContactID < 10
```

Vrací řádky, kde je ContactID < 10

Podmínka na test hodnoty NULL

```
SELECT * FROM Sales.SalesOrderHeader WHERE CreditCardID IS NULL
```

Na hodnotu NULL se v podmínce musíme dotazovat klíčovým slovem IS

Poddotazy

Existenční poddotaz

```
SELECT * FROM SalesLT.Customer AS C
WHERE EXISTS (SELECT * FROM SalesLT.CustomerAddress AS CA WHERE
CA.CustomerID = C.CustomerID AND CA.AddressType = 'Shipping')
```

Vrací ty řádky z tabulky Customer, pro které se povedlo najít záznam v CustomerAddress vyhovující podmínce poddotazu (Zákazník má definovanou Shipping adresu). Pro každý řádek z Customer je zavolán a vyhodnocen poddotaz, pokud je podmínka splněna, je tento řádek zhrnut do výsledku, jinak ne

Práce s více tabulkami - JOIN

INNER JOIN

```
SELECT SOH.SalesOrderNumber, CC.CardType FROM Sales.SalesOrderHeader AS SOH
INNER JOIN Sales.CreditCard AS CC ON SOH.CreditCardID = CC.CreditCardID
WHERE CC.CardType = 'Vista'
```

Za příkazem FROM a tabulkou následují příkaz INNER JOIN, který do našeho dotazu zahrne data i z další tabulky Sales.CreditCard. Pokud bychom nespecifikovali které sloupce chceme vybrat, tak by došlo k tomu, že za sloupce z tabulky SalesOrderHeader by byly přidány sloupce z tabulky CreditCard. Řádky jsou spojeny přes stejnou hodnotu vybraného sloupce (CreditCardID).

Protože se jedná o INNER JOIN, výsledkem budou jen ty záznamy které se povedlo spojit (nebudou uvedeny záznamy z tabulky SalesOrderHeader, ke kterým se nepovedlo najít kreditní kartu).

LEFT JOIN

```
SELECT SOH.SalesOrderNumber, CC.CardType FROM Sales.SalesOrderHeader AS SOH
LEFT JOIN Sales.CreditCard AS CC ON SOH.CreditCardID = CC.CreditCardID
```

*Výskedkem tohoto dotazu budou **všechny** záznamy z tabulky SalesOrderHeader ikdyž nebude nalezena pro ně platební karta (rozdíl oproti INNER). V tom případě budou mít sloupce z připojené tabulky hodnotu null.*

Aliasy

```
SELECT * FROM SalesLT.Customer AS C
WHERE EXISTS (SELECT * FROM SalesLT.CustomerAddress AS CA WHERE
CA.CustomerID = C.CustomerID AND CA.AddressType = 'Shipping')
```

*U tohoto dotazu byla tabulka SalesLT.Customer pojmenována dočasně jako C a CustomerAddress jako CA. U poddotazů a při použití JOIN by mohlo dojít k následujícímu problému: Pracujeme se sloupcem, který se vyskytuje v obou tabulkách (CustomerID zmanená CustomerID z Customer nebo z CustomerAddress? Chyba.) a proto je třeba uvádět kromě názvu sloupce i tabulku. Abychom nemuseli uvádět celý název tabulky, můžeme jí pomocí **AS** přiřadit nejaký kratší název.*

Příkaz AS můžeme použít i k přejmenování sloupce ve výsledku dotazu

```
SELECT FirstName AS [Jméno], LastName AS [Příjmení] FROM Person.Contact
```

Seskupování záznamů a agregační funkce

Při zpracování záznamů můžeme chtít je seskupit do skupin podle hodnot ve vybraných sloupcích. Toho docílíme použitím GROUP BY

```
SELECT CardType, COUNT(*) AS CardCount FROM Sales.CreditCard
GROUP BY CardType
```

Tento dotaz vybere záznamy z tabulky CreditCard a provede jejich seskupení podle hodnot ve sloupci CardType. Tím docílíme toho, že se vrátí pouze např. 4 řádky reprezentující každý typ karty a přidáním sloupce COUNT(*) získáme počet řádků v dané skupině (počet karet daného typu).

Ve výsledcích se mohou objevit jen ty sloupce z tabulky CreditCard, podle kterých byla data seskupena, jinak dojde k chybě.

Další agregační funkce:

SUM(*název sloupce*) - sečte hodnoty v daném slupci pro každou skupinu

MIN(název slupce) - vrací minimální hodnotu

MAX(název slupce) - vrací maximální hodnotu

AVG(název slupce) - vrací vypočítanou průměrnou hodnotu v daném sloupci pro danou skupinu záznamů

Filtrování seskupených záznamů

Pro filtrování seskupených záznamů slouží klíčové slovo HAVING

```
SELECT CardType, COUNT(*) AS CardCount FROM Sales.CreditCard
GROUP BY CardType
HAVING COUNT(*) > 4800
```

Dotaz provede výpočet počtu karet podle jednotlivých typů a následně vrátí jen ty řádky, kde je počet karet větší než 4800.

Rozdíl mezi HAVING a WHERE:

WHERE se aplikuje na řádky před seskupováním

HAVING se aplikuje na seskupené záznamy

Řazení výsledků

```
SELECT CardType, COUNT(*) AS CardCount FROM Sales.CreditCard
GROUP BY CardType
HAVING COUNT(*) > 4800
ORDER BY CardType
```

Dotaz vrátí výsledky seřazené vzestupně podle hodnot ve sloupci CardType. Je možno definovat více sloupců:

```
SELECT * FROM Person.Contact
ORDER BY FirstName, LastName DESC
```

Dále je možno za sloupcem specifikovat, jestli budeme řadit vzestupně nebo sestupně (defaultně vzestupně).

Fulltextové dotazy

Aby fungovalo fulltextové vyhledávání, je třeba vytvořit Fulltextový katalog a Fulltextový index nad danými sloupci.

FREETEXT

```
SELECT DocumentID, DocumentSummary
FROM Production.Document
WHERE FREETEXT (DocumentSummary, 'change payment')
```

CONTAINS

```
SELECT DocumentID, DocumentSummary
FROM Production.Document
WHERE CONTAINS (DocumentSummary, '"replacing" or "pedals"')
```

Hledání podle blízkosti slov:

```
SELECT DocumentID, DocumentSummary
FROM Production.Document
WHERE CONTAINS (DocumentSummary, 'replacing NEAR pedals')
```

Vyhledávání s rankovanými výsledky (vyšší rank = vyšší relevance):

```
SELECT f.RANK, DocumentID, DocumentSummary
FROM Production.Document d
INNER JOIN FREETEXTTABLE(Production.Document, DocumentSummary,  'bicycle
seat') f
ON d.DocumentID = [KEY]
ORDER BY RANK DESC
```

Modifikace dat

Vkládání dat

```
INSERT INTO Sales.CreditCard VALUES ('Maestro','5464654654',12,2009)
```

Do specifikované tabulky vložíme hodnoty uvedené za VALUES. Pořadí se bere ze schématu tabulky a jsou vynechány vypočítané sloupce a sloupce s IDENTITY.

```
INSERT INTO Sales.CreditCard (CardType, CardNumber) VALUES
('Maestro','5464654654')
```

Možnost specifikovat pořadí sloupců a případně možnost některé vynechat.

Modifikace dat

```
UPDATE Sales.CreditCard SET CardNumber = '45646546' WHERE CardNumber =
'45646547'
```

Ve vybrané tabulce aktualizujeme záznamy splňující podmínku (pokud není podmínka, aktualizace se provede nad všemi záznamy!)

Odstranění dat

```
DELETE FROM Sales.CreditCard WHERE CardNumber = '45646547'
```

Odstraníme záznamy splňující podmínku (není-li podmínka, vymažou se všechny záznamy)

Programátorské objekty

Pohledy

```
CREATE VIEW Sales.vCreditCardUsage
AS
SELECT CardType, COUNT(*) AS CardCount FROM Sales.CreditCard
GROUP BY CardType
```

*Vytvoří pohled s názvem Sales.vCreditCardUsage, který bude vracet záznamy jako dotaz specifikovaný za slovem **AS***

Uložené procedury

```
CREATE PROCEDURE Sales.uspCreditCardUsage
AS
BEGIN

SELECT CardType, COUNT(*) AS CardCount FROM Sales.CreditCard
GROUP BY CardType

END
```

Vytvoří uloženou proceduru, která po spuštění vykoná daný kód (dotaz) mezi BEGIN a END

```
CREATE PROCEDURE Sales.uspCreditCardUsageForCardType
      @CardType nvarchar(50)
AS
BEGIN

SELECT CardType, COUNT(*) AS CardCount FROM Sales.CreditCard
GROUP BY CardType
HAVING CardType = @CardType

END
```

Ukázka vytvoření parametrizované uložené procedury