TRANSACT-SQL

Učební texty

© ROMAN DANEL, 2017

## SQL přehled

**Příkazy pro manipulaci s daty** (DML – Data Manipulation Language):

SELECT, INSERT, UPDATE, DELETE

**Příkazy pro manipulaci s objekty** (Data Definition Language - DDL):

CREATE vytvoření objektu

ALTER změna (úprava) definice existujícího objektu

DROP zrušení objektu

GRANT změna (nastavení) přístupových práv

**Přehled syntaxe DML příkazů**

* **SELECT** [ **DISTINCT** ] **\*** | LIST OF COLUMNS, FUNCTIONS, CONSTANTS    **FROM** LIST OF TABLES OR VIEWS    [ **WHERE** CONDITION(S) ]    [ **ORDER BY** ORDERING COLUMN(S) [ **ASC** | **DESC** ] ]    [ **GROUP BY** GROUPING COLUMN(S) ]    [ **HAVING** CONDITION(S) ]
* **DELETE FROM** TABLE NAME  
     [ **WHERE** CONDITION(S) ]
* **INSERT INTO** TABLE NAME  
     [ **(**COLUMN LIST**)** ]  
     **VALUES** **(**VALUE LIST**)**
* **UPDATE** TABLE NAME  
     **SET** COLUMN NAME = VALUE  
     [ **WHERE** CONDITION ]

**Hodnota NULL**

NULL znamená prázdná (nevyplněná) hodnota. Je rozdíl mezi nulou nebo prázdným řetězcem a NULL hodnotou.

Při použití v SQL dotazech je nutné použít syntaxi „IS NULL“, „IS NOT NULL“.

**Predikáty**

* **BETWEEN ... AND** – vyhledá hodnoty v rozsahu
* **IN** – vyhledání v seznamu (v množině)
* **LIKE** – vyhledání podle masky (příklady: like ‘D%’, like ‘\_a\_%’)
* **JOIN**– spojení dvou tabulek

**Spojení tabulek přes JOIN**

* **Inner join -** vrátí pouze záznamy, pro které existuje záznam v druhé tabulce
* **Outer join -** vrátí všechny záznamy zprava (right) nebo zleva (left) bez ohledu na existenci záznamů v spojované tabulce

Příklad inner join:

SELECT column\_name(s)  
FROM table\_name1  
 INNER JOIN table\_name2   
 ON table\_name1.column\_name=table\_name2.column\_name

## Úvod do TRANSACT-SQL

Transact-SQL lze používat pro práci s SQL Serverem třemi způsoby:

1. Pomocí SQL Server Management Studia
2. Pomocí utility sqlcmd
3. Z aplikace, která se připojí na SQL Server

Deklarace (vytvoření) lokální proměnné pro uchování hodnot:

Declare @promenna datatype

Příklad:

Declare @vyska smallint

Proměnná se od jiných objektů odlišuje znakem „@“. Musí mít určen datový typ.

Přiřazená hodnoty do proměnné:

Select @vyska = 100

Set @vyska = 100

Přiřazení hodnoty do proměnné načtením ze sloupce z tabulky v databázi:

Select @vyska = VYSKA from SEZNAM

Globální proměnné – jsou definovány SQL Serverem; v názvu mají „@@“.

@@rowcount - počet řádků ovlivněných poslední SQL operací

@@error - kod chyby poslední SQL operace; je-li = 0, operace byla bez chyby

@@identity - vrací hodnotu identity u poslední operace

(pokud jde prováděna s tabulkou obsahující hodnotu identity, tj. automaticky generované číslo)

**GO** – ukončí dávku, lokální proměnné přestanou existovat.

## Větvení – IF, CASE

**IF podmínka**

begin

blok příkazů

end

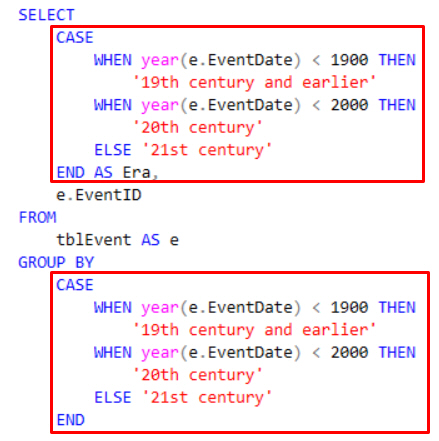
**ELSE**

begin

blok příkazů

end

Příklad s CASE:



Příklad č. 2

UPDATE FOODS

SET RATING = CASE

WHEN FAT < 1

THEN ‘very low fat’

WHEN FAT < 5

THEN ‘low fat’

WHEN FAT < 20

THEN ‘moderate fat’

WHEN FAT < 50

THEN ‘high fat’

ELSE ‘heart attack’

END ;

Příklad s CASE č. 3

SELECT ProductNumber,

Category =

CASE ProductLine

WHEN 'R' THEN 'Road'

WHEN 'M' THEN 'Mountain'

WHEN 'T' THEN 'Touring'

WHEN 'S' THEN 'Other sale items'

ELSE 'Not for sale'

END,

Name

FROM Product

## ŘÍZENÍ TOKU

### Cyklus while

While podmínka

Begin

blok příkazů

[break] -- předčasné ukončení

end

### Return

Ukončí proceduru a může vracet numerickou hodnotu

V těle procedury: Return @retval

Procedura se pak volá příkazem: Exec @var1 = proc\_name

### Waitfor

Waitfor delay ‚00:00:10‘

Begin

…

End

Waitfor time ‚19:00:00‘

Begin

…

end

## ULOŽENÉ PROCEDURY

Pojmenovaný kód v jazyce SQL, uložený v databázi. Může, ale nemusí vracet hodnotu.

CREATE PROCEDURE dbo.sp\_student\_result @student\_surname varchar(50)

AS

SELECT first\_name, last\_name , subject\_name, exam\_result

FROM student

LEFT JOIN results ON student.id\_student = results.id\_student

RIGHT JOIN subjects ON subjects.id\_subject = results.id\_subject

WHERE

student.last\_name = @student\_lastname

GO

Procedura se spustí v konzoli příkazem:

execute sp\_student\_result ‚NOVAK‘

Poznámka – místo excecute stačí napsat exec.

Kde najdeme v Management Studiu uložené procedury?

Stored procedures in Management Studio:

'database\_name'

Programmability

Stored Procedures

## FUNKCE

Příklad **skalární funkce** (vrací jednu hodnotu):

alter function typ\_obce(@typ smallint = null)

RETURNS varchar(20)

as

begin

return case @typ

when 1 then 'vesnice'

when 2 then 'městys'

when 3 then 'město'

when 4 then 'statutární město'

else 'neurčeno'

end

end

Funkce jsou uloženy v databázi.

Použití v rámci příkazů SQL.

Příklad použití funkce:

Select obec\_nazev, obec\_poc\_obyv, dbo.typ\_obce(obec\_typ)

from OBCE

**Systémové funkce** – definované SQL Serverem.

Přehled funkcí SQL Serveru:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/functions>

DATABASEPROPERTY(), USER\_NAME, USER, HOST\_ID(), HOST\_NAME(), APP\_NAME()…

select object\_id('OBCE')

select object\_name(245575913)

Příklad:

IF OBJECT\_ID(‘tabulka’) IS NOT NULL

begin

DROP TABLE tabulka;

end

Alternativa k výše uvedenému příkladu:

IF EXISTS (SELECT \* FROM sys.tables

WHERE name =‘tabulka’)

begin

DROP TABLE tabulka;

end

### Příklady matematických funkcí

* Round(x, y) – zaokrouhlení na y desetinných míst
* FLOOR(x) – zaokrouhlení dolů
* CEILING(x) – zaokrouhlení nahoru
* RAND([seed]) – generování náhodného čísla
* Goniometrické funkce
* Pi()
* Mocnění – SQUARE(x), SQRT(x), POWER(x), EXP(x), LOG10(x), LOG(x)
* MOD

<https://docs.microsoft.com/en-us/sql/t-sql/functions/mathematical-functions-transact-sql>

### Příklady řetězcových funkcí

* Txt1 + txt2
* LEN(txt)
* LOWER(txt)
* UPPER(txt)
* SUBSTRING(txt, start, length)
* REPLACE(Where, What, Using)
* REVERSE(txt)
* CHARINDEX(what, where, start)
* STR(num) -- konverze čísla na řetězec
* LTRIM(x), RTRIM(x) – odstranění prázdných znaků
* REPLACE(what, times)
  + - Příklad: REPLACE(‚a‘, 5) Výsledek: aaaaa

<https://docs.microsoft.com/en-us/sql/t-sql/functions/string-functions-transact-sql>

### Příklady datumových funkcí

* Getdate() – vrací aktuální datum a čas
* DATENAME(datepart, date\_str)
* DATEPART(datepart, date\_str) -- vrací číslo
* DAY(date), MONTH(date), YEAR(date)
* DATEDIFF(datepart, date1, date2) – rozdíl dvou datumů
* DATEADD(datepart, number, date) – přidání časové jednotky k datumu
* ISDATE(str) -- vrací 1, pokud lze řetězec převést na datum

<https://docs.microsoft.com/en-us/sql/t-sql/functions/date-and-time-data-types-and-functions-transact-sql>

### Konverze datových typů

CONVERT

CAST

### Funkce pro administraci SQL Serveru

* GRANT, DENY
* KILL
* RECONFIGURE – změny konfigurace učiní trvalé
* SHUTDOWN – vypnutí serveru
* BACKUP/RESTORE
* BULK INSERT – nahrává data ze souboru
* CHECKPOINT
* DBCC CHECK… - kontrola chyb v databázích
* DBCC SHRINKDATABASE

### Systémové procedury

* Sp\_dboption – volby platné pro celou db
* Sp\_lock – informace o zámcích
* Sp\_dbremove
* Sp\_monitor – informace o výkonu
* Sp\_tableoption – volby na úrovni tabulky

Dalším typem funkcí jsou funkce tabulkové (**table-valued)**. Na rozdíl od skalárních funkcí vracejí řádky z tabulky – recordset. Lze je použít v SQL příkazech v klauzuli FROM jako zdroj dat.

### Agregační funkce

* **SUM** Total of the values in a field.
* **AVG** Average of the values in a field.
* **MIN** Lowest value in a field.
* **MAX** Highest value in a field.
* **COUNT** Number of values in a field, not counting Null (blank) values

S agregačními funkcemi lze v dotazech použít klauzuli GROUP BY a filtr HAVING.

# Integrita

* **Entitní** – jedinečná identifikace každé entity - pomocí primárního klíče (PK)
* **Referenční** – pomocí cizího klíče (FK) – integrita vazeb

**Primární klíč** – definuje se nad jedním nebo více sloupci tabulky. Hlavním účelem primárního klíče je zajistit jednoznačnost záznamů v tabulce.

Data primárního klíče nesmí obsahovat duplicitu nebo hodnotu Null.

**Cizí klíč** se definuje na sloupci tabulky, která je navázána na primární tabulku. Pomocí něho jsou realizovány vztahy v relační databázi.

Cizí klíč se odkazuje na sloupec primární tabulky. Do sloupce s definovaným cizím klíčem nelze vložit hodnotu, která není vložena v nadřízeném sloupci primární tabulky.

**Referenční integrita**

1. **Restrict** – z nadřízené tabulky nelze záznam smazat, je-li odkaz v tabulce podřízené
2. **Cascade** – po výmazu z nadřízené automaticky smaže i odkazy v podřízené tabulce
3. **Set null** – po výmazu v nadřízené dostanou odkazy v podřízené hodnotu NULL
4. **Set default**

Integrita zajišťuje správnost vztahu mezi daty.

Konzistence – stejná hodnota dat ve všech jejich výskytech.

# Transakce

Co je to transakce?

Skupina příkazů, které převedou databázi z jednoho konzistentního stavu do druhého.

Databázové transakce musí splňovat tzv. vlastnosti **ACID**:

*A Atomicity* – atomicita (nedělitelnost transakce)

*C Consistency* - konzistence (není porušeno integritní omezení)

*I Isolation* - izolovanost (ostatní nevidí, dokud není ukončena)

*D Durability* - trvalost (změny, které se provedou potvrzenou transakcí, jsou v databázi trvalé a nemohou být ztraceny)

Transakce v databázových systémech je tedy **skupina databázových operací, která je provedena buď jako celek, nebo není provedena vůbec.**

SQL:

* Zahájení transakce – BEGIN TRANSACTION
* Ukončení transakce - END
  + - **COMMIT** – potvrzení transakce
    - **ROLLBACK** – zrušení transakce, návrat do původního stavu

Příklad transakce v jazyce SQL:

Begin transaction

delete from ORDERS\_ITEM

where order\_id = 1;

if @@error <> 0 then

delete from ORDERS where order\_id=1;

else

rollback;

End

**Dva přístupy ke zpracování transakce**

* **Pesimistické** – provedené změny jsou zapisovány do dočasných objektů a teprve po potvrzení se stanou platnými
* **Optimistické** – předpokládá se, že transakce se nebude vracet a změny jsou zapisovány do tabulek a do logu jsou zapisovány informace pro případný rollback

# Standardy pro psaní kódu

<https://blog.sqlauthority.com/2008/09/23/sql-server-coding-standards-guidelines-part-1/>

<https://blog.sqlauthority.com/2008/09/24/sql-server-coding-standards-guidelines-part-2/>

# Trigger

Trigger je speciální typ uložené procedury, který je spojen s databázovými objekty obsahující data a který se automaticky spustí, je-li splněna určitá podmínka.

Trigger může být spuštěn při operacích update, insert, delete.

Trigger je součástí transakce příkazu, který ho spustil.

Kde najdu trigger v Microsoft Management studiu?

Databases

test

dbo.SEZNAM

Triggers

test\_trg

Jestliže chci trigger editovat v Management Studiu: right mouse button menu -> Modify

Příklad triggeru:

CREATE TRIGGER [dbo].[test\_trg]

ON [dbo].[STUDENT]

AFTER INSERT

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for trigger here

insert into LOG (LOG\_DATE,LOG\_TEXT, LOG\_ID\_USER) values

(GETDATE(), 'STUDENT - record insert', '')

END

# Pohled (View)

Pohled je virtuální tabulka, vytvořená nad fyzickou tabulkou (nebo skupinou tabulek) pomocí dotazu typu Select. Dále se chová jako fyzická tabulka.

Cílem je zjednodušit práci s dotazy pro aplikace, automatizovat opakující se dotazy a zrychlit přístup na data.

Příklad skriptu pro vytvoření pohledu:

create view EXAM\_RESULT\_VIEW

as

select STUDENT\_NAME, SURNAME, SUBJECT\_NAME, EXAM\_RESULT\_POINTS, EXAM\_RESULT

from STUDENT

left join RESULT on STUDENT.ID\_STUDENT = RESULT.ID\_STUDENT

right join SUBJECTS on SUBJECTS.ID\_SUBJECT = RESULT.ID\_SUBJECT

# Dynamické SQL

Jako dynamické SQL označujeme situaci, kdy SQL dotaz je dynamicky sestaven podle určitých podmínek nebo zadaných parametrů těsně před vykonáním. Dopředu tedy není dán tvar příkazu.

Příklad:

Dotaz na data v tabulce, kde jméno tabulky je předáno jako parametr @tabulka:

Declare @SQLprikaz varchar(100)

Set @SQL prikaz = ‘select \* from ‘ + @tabulka

EXECUTE sp\_executesql @SQLprikaz