

Databázové systémy

Funkcie v SQL

Prečo potrebujeme funkcie?

- Databáza nie je len pasívne úložisko, vie dáta aj upravovať a počítať.
- **Zadanie:** Potrebujeme zoznam všetkých žijúcich klientov, ich vek v rokoch a meno v tvare meno a priezvisko usporiadané od najstaršieho.
- **Bez SQL funkcií:** Museli by ste stiahnuť celé tabuľky, v kóde aplikácie odfiltrovať zosnulých, ručne napísať cykly na spájanie textu, matematicky prepočítať dátum narodenia na vek a pole zotriediť.
- **S SQL funkciami:** Databáze pošlete jeden elegantný dopyt s funkciami **CONCAT**, **DATEDIFF**, **COALESCE** a klauzulou **ORDER BY**. Výsledok dostanete okamžite, očistený a pripravený na zobrazenie.
- **Presun logiky na správne miesto (Database vs. Application):**
 - Prečo ťahať milióny riadkov surových dát do aplikácie (Java, Python, C#) a spracovávať ich tam, keď ich dokáže databáza prepočítať priamo na serveri za zlomok sekundy?
 - S SQL funkciami sa **znižuje zaťaženie siete** a zrýchľuje odozva aplikácií.

Typy funkcií podľa spracovania riadkov

- **Skalárne (Single-Row)**

- Pracujú nad jedným riadkom a vrátia hodnotu pre každý riadok zvlášť
- Napr. úprava textu, formátovanie dátumu, ...
- Napr. **UPPER**(meno)

- **Agregačné (Multi-Row)**

- Zoberú celú skupinu riadkov a vypočítajú z nich jednu sumárnu hodnotu
- Napr. priemer, súčet
- Napr. **AVG**(vyska)

- **Window (Analytické)**

- Počítajú hodnoty nad skupinou riadkov, ale **nezlučujú riadky** -zachovávajú identitu každého záznamu (využívajú klauzulu **OVER**()).

- **Tabuľkové (Table-Valued) Tabuľkové (Table-Valued):** Špeciálne funkcie, ktoré namiesto jednej hodnoty vrátia celú výslednú tabuľku dát (používajú sa v klauzule FROM).

- Špeciálne funkcie, ktoré namiesto jednej hodnoty vrátia celú výslednú tabuľku dát (používajú sa v klauzule **FROM**).

Typy funkcií podľa účelu a dátového typu (Čo spracovávajú)

- **Textové**

- Manipulácia s textom (**CONCAT()**, **SUBSTRING()**, **LENGTH()**).

- **Numerické**

- Matematické operácie a zaokrúhľovanie (**ROUND()**, **ABS()**, **MOD()**).

- **Dátumové a časové**

- Práca s časom, priradovanie a formátovanie (**NOW()**, **DATEDIFF()**, **DATE_FORMAT()**).

Špeciálne funkcie

- **Systémové a logické**
 - Ošetrovanie chýb, konverzie a podmienky (**COALESCE()**, **CAST()**, **CASE ... WHEN**).
- **Priestorové (Spatial)**
 - Práca s GPS súradnicami a mapami (**ST_Distance()**).
- **Pološtruktúrované data**
 - Práca s formátmi **JSON** a **XML** (**JSON_EXTRACT()**).
- **Kryptografické**
 - Šifrovanie a kontrolné súčty pre bezpečnosť (**SHA256()**)
- ...

Pasca pre vývojárov: Každá databáza má vlastné funkcie

- **Prenositeľnosť kódu:** Prečo funkcia z MySQL nefunguje v Oracle?
- **Limity štandardizácie:** Kým základné klauzuly ([SELECT](#), [FROM](#), [WHERE](#)) sú v **ANSI SQL štandarde** takmer totožné, **skalárne funkcie sa naprieč DBMS výrazne líšia**.
- **Problém pri migrácii:** Ak napíšete aplikáciu využívajúcu špecifické funkcie pre MySQL, pri prechode na PostgreSQL alebo MS SQL budete musieť prepísať veľkú časť SQL dopytov.
- **MariaDB vs. MySQL:** MariaDB vznikla ako vetva (fork) MySQL, takže v oblasti funkcií si zachovávajú takmer 100 % kompatibilitu. Pri Oracle a MS SQL však narazíte na úplne iný svet.

Úloha (Problém)	MySQL / MariaDB	Microsoft SQL Server (T-SQL)	Oracle DB (PL/SQL)
Spájanie textu	CONCAT (meno, ' ', priezvisko)	meno + ' ' + priezvisko alebo CONCAT (meno, ' ', priezvisko)	meno ' ' priezvisko alebo CONCAT (meno, ' ', priezvisko)
Dĺžka reťazca	LENGTH (priezvisko)	LEN (priezvisko)	LENGTH (priezvisko)
Ošetrenie NULL	IFNULL (stĺpec, 'X') alebo COALESCE (stĺpec, 'X')	ISNULL (stĺpec, 'X') alebo COALESCE (stĺpec, 'X')	NVL (stĺpec, 'X') alebo COALESCE (stĺpec, 'X')
Aktuálny dátum	CURDATE () alebo NOW ()	GETDATE ()	SYSDATE
Formát dátumu	DATE_FORMAT (dat_nar, '%d.%m.%Y')	FORMAT (dat_nar, 'dd.MM.yyyy')	TO_CHAR (dat_nar, 'DD.MM.YYYY')

Príklady

-- MySQL / MariaDB

```
SELECT
    CONCAT(meno, ' ', priezvisko) AS cele_meno
FROM osoba;
```

-- Microsoft SQL Server (T-SQL)

```
SELECT
    meno + ' ' + priezvisko AS cele_meno
FROM osoba;
```

-- Oracle DB (PL/SQL)

```
SELECT
    meno || ' ' || priezvisko AS cele_meno
FROM osoba;
```

Príklady

SELECT

meno,

priezvisko,

vyska,

ROUND(vyska, 0) AS zaokruhlene_klasicky,

FLOOR(vyska) AS zaokruhlene_nadol,

CEILING(vyska) AS zaokruhlene_nahor,

ROUND(vaha / ((vyska/100) * (vyska/100)), 1) AS bmi_index

FROM

osoba;

Formátovanie dátumu a času

- **DATE_FORMAT(dátum, šablóna)** (v MS SQL funkcia **FORMAT()**) – transformuje dátum na text podľa zadanej masky.
- **Základné formátovacie špecifikátory (MySQL štandard):**
 - %d – deň v mesiaci číselne (01 až 31)
 - %m – mesiac číselne (01 až 12)
 - %Y – štvorciferný rok (napr. 1918)
 - %W – názov dňa v týždni (napr. Monday / Pondelok podľa lokalizácie)
 - %M – slovný názov mesiaca (napr. May)

Príklady

SELECT

meno,
priezvisko,
dat_nar AS povodny_datum,
DATE_FORMAT(dat_nar, '%d.%m.%Y') AS slovensky_format,
DATE_FORMAT(dat_nar, '%W') AS den_v_tyzdni

FROM

osoba

Code	Description	Results
%a	Abbreviated weekday name	(Sun...Sat)
%d	Day of the month (numeric)	(00...31)
%D	Day of the month with English suffix	(1st, 2nd, 3rd, etc.)
%e	Day of the month (numeric)	(0...31)
%j	Day of the year	(001...366)
%w, %W	Day of the week	(0=Sunday...6=Saturday)
%u	Week, where Monday is the first day of the week	(0...52)
%U	Week, where Sunday is the first day of the week	(0...52)
%v	Week, where Monday is the first day of the week; used with `%x`	(1...53)
%V	Week, where Sunday is the first day of the week; used with `%X`	(1...53)
%b	Abbreviated month name	(Jan...Dec)
%c	Month (numeric)	(1...12)
%m	Month (numeric)	(01...12)
%M	Month name	(January...December)
%y	Year (numeric, two digits)	(yy)
%Y	Year (numeric, four digits)	(yyyy)
%f	Microseconds (numeric)	(000000...999999)
%s	Seconds	(00...59)
%S	Seconds	(00...59)
%i	Minutes (numeric)	(00...59)
%h	Hour	(01...12)
%H	Hour	(00...23)
%l	Hour	(01...12)
%k	Hour	(0...23)
%I	Hour	(1...12)
%p	AM or PM	AM or PM
%r	Time, 12-hour	(hh:mm:ss [AP]M)
%T	Time, 24-hour	(hh:mm:ss)

Agregačné funkcie

- Na rozdiel od skalárnych funkcií, tieto zbalia viacero riadkov do jedného čísla.
- **COUNT()** – spočíta riadky alebo vyplnené hodnoty.
- **SUM()** – vypočíta celkový súčet.
- **AVG()** – vypočíta aritmetický priemer.
- **MIN()** / **MAX()** – nájde extrémny (najmenšiu a najväčšiu hodnotu).

Agregačné funkcie

- Na rozdiel od skalárnych funkcií, tieto zbalia viacero riadkov do jedného čísla.
- **COUNT()** – spočíta riadky alebo vyplnené hodnoty.
- **SUM()** – vypočíta celkový súčet.
- **AVG()** – vypočíta aritmetický priemer.
- **MIN()** / **MAX()** – nájde extrémy (najmenšiu a najväčšiu hodnotu).

Agregačné funkcie

- SELECT
COUNT(*) AS celkovo_ludi,
AVG(vyska) AS priemerna_vyska,
MAX(vaha) AS najvyssia_vaha,
MIN(dat_nar) AS najstarsi_dátum_narodenia
FROM Osoba;

Rozdiel medzi COUNT(*) a COUNT(stĺpec)

- **COUNT(*)**: Spočíta **všetky riadky** bez ohľadu na to, čo v nich je (počíta aj riadky plné NULL hodnôt).
- **COUNT(názov_stĺpca)**: Spočíta iba riadky, kde je tento konkrétny stĺpec **vyplnený** (ignoruje NULL).
- **SELECT**
COUNT(*) AS pocet_vsetkych_riadkov, -- Vráti: 23
COUNT(dat_smrti) AS pocet_vyplnenych_dat -- Vráti: 5
FROM Osoba;

Kombinácia funkcií: GROUP BY a HAVING

- Agregáčné funkcie najčastejšie kombinujeme s GROUP BY na výpočet štatistík pre kategórie (napr. podľa pohlavia).
- Ak chceme výsledné agregované skupiny filtrovať, používame **HAVING**.

Agregačné funkcie

- SELECT
 pohlavie,
 COUNT(*) AS pocet_ludi,
 AVG(vyska) AS priemerna_vyska
FROM osoba
GROUP BY pohlavie
HAVING AVG(vyska) > 160.0;