35. CRUD, agregační funkce, množinové operace, vnořené dotazy

35.1. CRUD

Shrnuje čtyři základní operace nad záznamem v nějakém trvalém úložišti (např. v SQL databázi):

- Create (Insert)
- Retrieve / Read (Select)
- Update (Update)
- Delete (Delete)

35.2. Agregační funkce

Agregační funkce jsou v SQL statistické funkce, pomocí kterých systém řízení báze dat umožňuje seskupit vybrané řádky dotazu (získané příkazem SELECT) a spočítat nad nimi výsledek určité aritmetické nebo statistické funkce. Agregační funkce se v SQL používají s konstrukcí GROUP BY.

- AVG()
- SUM()
- COUNT()
- MIN()
- MAX()

35.2.1. Having:

Syntaktická konstrukce HAVING, za kterou následuje omezující podmínka, umožňuje omezit řádky, které se budou ve výsledku agregovat a u kterých se budou počítat výsledky agregačních funkcí. Na rozdíl od podmínek v klauzuli za WHERE, kde to není povoleno, dovoluje podmínka v klauzuli HAVING používat agregační funkce. Sloupce, které se objeví v agregačních funkcích za HAVING, musejí být uvedeny v sekci za GROUP BY. Návrh SQL používá dvě různé konstrukce pro omezující podmínky výběru podle výskytu či absence agregačních funkcí proto, že agregovaný výběr se od toho běžného liší (použitím rozdílných postupů, algoritmů, výkonem, atd.).

```
SELECT zeme, COUNT(*) AS Pocet
FROM zamestnanci
GROUP BY zeme
HAVING COUNT(*)>1
```

35.3. Množinové operace

35.3.1. Sjednocení

Množinové operace: Sjednocení relací

Definice (sjednocení relací, angl.: *union*)

Pro dvě relace $\mathcal{D}_1$ a $\mathcal{D}_2$ na relačním schématu $R \subseteq Y$ zavádíme

$$\mathcal{D}_1 \cup \mathcal{D}_2 = \{r \in \prod_{y \in R} D_y \,|\, r \in \mathcal{D}_1 \text{ a/nebo } r \in \mathcal{D}_2\}.$$

Relace $\mathcal{D}_1 \cup \mathcal{D}_2$ na schématu R se nazývá **sjednocení relací** $\mathcal{D}_1$ a $\mathcal{D}_2$.

Tutorial D:

$\langle relační\text{-}výraz_1 \rangle$ `UNION` $\langle relační\text{-}výraz_2 \rangle$

`UNION {`$\langle relační\text{-}výraz_1 \rangle$`,` $\langle relační\text{-}výraz_2 \rangle$`, ...}`

SQL:

`SELECT * FROM` $\langle jméno_1 \rangle$
`UNION`
`SELECT * FROM` $\langle jméno_2 \rangle$

35.3.2. Průnik

Množinové operace: Průnik relací

Definice (průnik relací, angl.: *intersection*)

Pro dvě relace $\mathcal{D}_1$ a $\mathcal{D}_2$ na relačním schématu $R \subseteq Y$ zavádíme

$$\mathcal{D}_1 \cap \mathcal{D}_2 = \{r \in \prod_{y \in R} D_y \,|\, r \in \mathcal{D}_1 \text{ a zároveň } r \in \mathcal{D}_2\}.$$

Relace $\mathcal{D}_1 \cap \mathcal{D}_2$ na schématu R se nazývá **průnik relací** $\mathcal{D}_1$ a $\mathcal{D}_2$.

Tutorial D:

$\langle relační\text{-}výraz_1 \rangle$ `INTERSECT` $\langle relační\text{-}výraz_2 \rangle$

`INTERSECT {`$\langle relační\text{-}výraz_1 \rangle$`,` $\langle relační\text{-}výraz_2 \rangle$`, ...}`

SQL:

`SELECT * FROM` $\langle jméno_1 \rangle$
`INTERSECT`
`SELECT * FROM` $\langle jméno_2 \rangle$

35.3.3. Rozdíl

Množinové operace: Rozdíl relací

Definice (rozdíl relací, angl.: *difference/minus*)

Pro dvě relace $\mathcal{D}_1$ a $\mathcal{D}_2$ na relačním schématu $R \subseteq Y$ zavádíme

$$\mathcal{D}_1 \setminus \mathcal{D}_2 = \{r \in \prod_{y \in R} D_y \,|\, r \in \mathcal{D}_1 \text{ a zároveň } r \notin \mathcal{D}_2\}.$$

Relace $\mathcal{D}_1 \setminus \mathcal{D}_2$ na schématu R se nazývá **rozdíl relací** $\mathcal{D}_1$ a $\mathcal{D}_2$.

Tutorial D:

$\langle relační\text{-}výraz_1 \rangle$ MINUS $\langle relační\text{-}výraz_2 \rangle$

SQL:

SELECT * FROM $\langle jméno_1 \rangle$
EXCEPT
SELECT * FROM $\langle jméno_2 \rangle$

35.4. Vnořené dotazy

Vnořený dotaz není nic jiného, než příkaz SELECT vnořený do jiného příkazu SELECT. Vnořené dotazy využijeme tam, kde potřebujeme nejprve zjistit nějakou informaci a v závislosti na ní zjistit pak informace další. Typickým příkladem by mohl být následující dotaz. Vypište seznam všech zaměstnanců, kteří mají nadprůměrný plat. Pomocí klasických SQL dotazů, které zatím známe, tento výsledek obdržíme ve dvou krocích.

Nejprve si zjistíme, jaký je průměrný plat ve firmě (použijeme agregační funkci AVG):

```
SELECT
AVG(plat)
FROM platy
```

Tento dotaz nám vrátí např. číslo 15600. Toto číslo si zapamatujeme a použijeme jej v konstrukci následujícího dotazu:

```
SELECT jméno, příjmení
FROM platy
WHERE plat > 15600
```

Jistě uznáte, byť je tento příklad docela jednoduchý, že by někdy bylo pohodlnější, kdybychom oba kroky mohli spojit do jednoho, tj. napsali bychom jeden příkaz, který by nám potřebný seznam zaměstnanců vrátil.

Za jednoduchý vnořený dotaz budeme považovat vnořený příkaz SELECT vracející nám jednu hodnotu. Můžeme psát restrikce s využitím právě té hodnoty, kterou nám vrátí vnořený SELECT. Vnořený příkaz se píše do závorek a v dotazu je uveden v části WHERE. Výše uvedený dotaz bychom zapsali následovně:

```
SELECT jméno, příjmení
FROM platy
WHERE plat > (SELECT AVG(plat) FROM platy)
```

V restrikci výše uvedeného příkazu máme podmínku plat > vypočtená_hodnota. Vypočtená_hodnota se nám vyhodnotí vnořeným SELECTem na jednu hodnotu a máme tedy podmínku zapsanou v pořádku a dotaz je korektní. Pro úplnost uvádím, že vnořený SELECT se vyhodnotí pouze jednou, hodnotu si SQL server “zapamatuje” a s ní pak porovnává všechny ostatní hodnoty v tabulce.

35. SQL, DDL, integritní omezení, DML, typy spojení, vložené dotazy (Bartoš)

SQL

SQL je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. SQL je zkratka anglických slov **Structured Query Language** (strukturovaný dotazovací jazyk).

Neprocedurální jazyk – říkáme co chceme, ne jak to provést

Interakční a hostitelská verze

.. Interakční – příkazy lze zadávat přímo v režimu on-line
.. Hostitelská – SQL je součástí hostitelského jazyka, např. C, Java, PHP, ASP,…

SQL příkazy se dělí na čtyři základní skupiny:

- Příkazy pro manipulaci s daty (SELECT, INSERT, UPDATE, DELETE, …)
- Příkazy pro definici dat (CREATE, ALTER, DROP, …)
- Příkazy pro řízení přístupových práv (GRANT, REVOKE)
- Příkazy pro řízení transakcí (START TRANSACTION, COMMIT, ROLLBACK)
- Ostatní nebo speciální příkazy

Datové typy

– Integer – celé číslo se znaménkem;
– Decimal(p,q) – číslo s p ciframi a desetinou čárkou na q pozici zprava;
– Float – reálné číslo;
– Char(n) – řetězec znaků délky n;
– Varchar(n) – řetězec znaků max. délky n;
– Standartní: Numeric, Real, Double precision, Smallint;
– Nestandarní: Date,Money, geometric types

DDL

- DDL - Data Definition Language; je to jazyk pro definici dat; jde o součást SQL
- pomocí jeho příkazů lze vytvářet databázové objekty (pohledy, tabulky, indexy, jmenné prostory, databáze, uložené procedury a funkce, triggery, uživatele) a upravovat jejich strukturu
- příkazy s klíčovými slovy CREATE, ALTER a DROP
- těmito příkazy se neovlivňují data, ale úložiště, do kterých se data ukládají
- druhým důležitým jazykem je DML (Data Modeling Language)

Jazyk SQL se skládá z několika příkazů rozdělených do dvou základních skupin: příkazy pro definici datových struktur (Data Definition Language) a příkazy pro práci s obsahem tabulek (Data Modeling Language).
DDL má deklarativní charakter a umožňuje nám vytvářet, modifikovat a rušit databázové objekty: tabulky, pohledy, indexy, jmenné prostory, tabulkové prostory, databáze, uložené procedury a funkce, triggery a uživatele. Objekty vytváříme příkazem CREATE, rušíme příkazem DROP, přidáváme přístupová práva příkazem GRANT nebo naopak příkazem REVOKE přístupová práva odebíráme. Vlastnosti objektů měníme příkazem ALTER. Každý SQL příkaz se skládá z několika volitelných částí; musíme ale respektovat logické vazby mezi těmito částmi a musíme tyto části zapisovat ve správném pořadí např. klauzule WHERE je za klauzulí FROM a před klauzulemi GROUP BY a ORDER BY.

```
CREATE TABLE dite (
  id SERIAL PRIMARY KEY,
  rodic INTEGER NOT NULL REFERENCES rodic(id),
  jmeno VARCHAR(15) NOT NULL CHECK jmeno <> '',
  prijmeni VARCHAR(15) NOT NULL CHECK prijmeni <> ''
);

ALTER TABLE dite ADD COLUMN vek INTEGER NOT NULL CHECK vek > 0;
GRANT SELECT ON TABLE dite TO public;
```

Integritní omezení

CHECK - kontrola vstupních dat
DEFAULT - výchozí hodnota
FOREIGN KEY - definice cizího klíče
PRIMARY KEY - definice primárního klíče
NOT NULL -sloupec nesmí mít hodnotu NULL
UNIQUE-všechny hodnoty ve sloupci/ích musí být unikátní, NULL je přípustné

NULL hodnota

.. Predpoklad že vždy známe hodnotu atributu je velmi silný a v reálném svete nereálný
.. Hodnoty atributu mohou být neznámé (nevíme do vložit)
.. SQL na toto pamatuje hodnotou NULL
.. NULL použije tehdy, pokud je hodnota atributu:
  – Neznámá
  – Neaplikovatelná
.. NULL není 0 ani mezera!!
.. NOT NULL IO nepripouští ani vložení NULL hodnoty do sloupce

CHECK

.. Jedná se integritní omezení na atributu
.. SQL92 = umožnuje definovat CHECK na více atributech, ci dokonce pro více tabulek
.. CHECK specifikuje Boolean podmínku (TRUE, FALSE, nebo neznámé), která se aplikuje na všechny hodnoty do atributu vložené
.. Pokud FALSE, pak je dotaz odmítnut
.. Lze specifikovat i vícenásobný CHECK pro každý atribut
.. JEDNÁ SE O DOMÉNOVÉ OMEZENÍ

```
CREATE TABLE my_table(
int_column INTEGER,
char_column VARCHAR(55),
pocet INTEGER NOT NULL,
PRIMARY KEY(int_column),
CHECK (pocet >=1 AND pocet<=10))
```

UNIQUE

- UNIQUE zajistí, že se nebudou vyskytovat žádné duplikáty v atributu, který není součástí primárního klíče
- UNIQUE je vhodné použít tam, kde chce mít atribut/y bez duplicit, které nejsou součástí

primárního klíče.
- Na rozdíl od PRIMARY KEY, UNIQUE dovolí vložit NULL do hodnoty atributu
- Pouze jedna NULL hodnota je však dovolena v rámci atributu!!!

PRIMARY KEY
.. Definuje atribut/atributy, které tvorí primární klíc
.. Pouze jeden pro tabulku
.. Nelze do takového atributu/u vložit NULL hodnotu
.. Když je specifikován PRIMARY KEY, SQL Server 2005 Database Engine zajistí unikátnost dat tím, že založí unique index pro atribut/y primárního klíce.
.. Index zároven urychluje prístup k atributum primárního klíce
.. Pokud je PRIMARY KEY definován pro více atributů, je možné, aby se hodnota opakovala v jednom z nich, ale kombinace musí být unikátní.

DEFAULT
- Pomocí DEFAULT lze zadat výchozí hodnotu atributu
- Pokud tedy uživatel nezadaná hodnotu bude takto definovaný atribut mít hodnotu definovanou pomocí DEFAULT

FOREIGN KEY
Pomocí FOREIGN KEY definujeme vazbu mezi dvěma entitami - referenční integritu

Referenční integrita
- Referenční integrita definuje logické vazby (vztahy) mezi tabulkami
- Jedná se o „provázání“ dvou tabulek pomocí odkazu z vedlejší tabulky do tabulky hlavní

DML
DML příkazy slouží k manipulaci s daty v databázi. ANSI SQL:1999 definije příkazy INSERT, UPDATE, DELETE, a SELECT. Tato základní sada je ještě v ANSI SQL:2003 rozšířena o příkaz MERGE (není podporován PostgreSQL).
- Příkaz INSERT se používá pro přidávání nových záznamů do tabulky
- Příkaz UPDATE se používá pro aktualizaci existujících záznamů v tabulce
- Příkaz DELETE se používá pro odstranění záznamů z tabulky
- Příkaz SELECT se používá pro zobrazení a hledání záznamů v tabulce

```
INSERT INTO dite(rodic,jmeno, prijmeni,vek)
 VALUES(1,'Jindřich','Suchý',12);
UPDATE dite SET vek = 13 WHERE id = 2;
DELETE FROM dite WHERE vek > 26;
SELECT * FROM dite WHERE rodic = 1;
```

Nejčastěji používaným příkazem DML příkazem je nejspíš příkaz SELECT. Skládá se s následujících povinných a nepovinných klauzulí (při zápisu je nezbytné respektovat pořadí klauzulí):

```
SELECT <seznam zobrazovaných sloupců>
FROM <zdroj nebo zdroje tabulek>
WHERE <výběrová kritéria>
[GROUP BY <kritéria pro sdružování záznamů>]
```

```
[HAVING <kritéria pro filtrování sdružených záznamů>]
[ORDER BY <způsob řazení dat>]
```

Příkladem může být dotaz, který zobrazí všechny rodiče, které mají více než jedno dítě:

```
SELECT rodic.* FROM rodic
  WHERE id IN (SELECT rodic FROM dite GROUP BY rodic HAVING count(*) >
1)
```

Typy spojení

- PROJEKCE - `SELECT jmeno FROM osoby`
- KARTÉZKÝ SOUCIN - `SELECT * FROM osoby, byty`
- SELEKCE - `SELECT jmeno FROM osoby WHERE jmeno=”jirik”`
- SPOJENÍ - `SELECT jmeno FROM osoby, byty WHERE osoby.rc = byty.rc`

Vnitřní spojení
.. INNER JOIN
.. Je to výchozí hodnota pro JOIN
.. Ve výsledku budou jen ty řádky, které mají spojované tabulky společné

Vnější spojení
.. OUTER JOIN
.. Spojované tabulky mohou mít i různé hodnoty v řádcích.
.. Kde hodnota nebyla dolní se automatický NULL

LEFT OUTER JOIN
- T1 LEFT OUTER JOIN T2
- Provede se vnitřní spojení
- Pak se vezmou řádky z tabulky „vlevo“ (tabulka T1) a do odpovídajících atributů tabulky pravé (T2) se doplní NULL

RIGHT OUTER JOIN
- T1 RIGHT OUTER JOIN T2
- Provede se vnitřní spojení
- Pak se vezmou řádky z tabulky „vpravo“ (T2) a do odpovídajících atributů tabulky „levé“ (T1) se doplní NULL

FULL OUTER JOIN
- Provede se vnitřní spojení
- Pak se provede LEFT
- Následně RIGHT

Vložené dotazy

```
SELECT jmeno FROM Zakaznici
WHERE rc IN (
SELECT rc FROM Rezervace WHERE
Rezervace.jmeno_f=
(SELECT F.jmeno FROM Filmy WHERE
F.reziser=‘Sverak’)
```

```
);
```