ktu informatikos fakultetas 1922

DML

AGREGAVIMAS, GRUPAVIMAS,

INSERT, UPDATE, DELETE,

ktu informatikos fakultetas 1922

Agregavimo operatoriai ir grupavimas

SUM,

AVG,

MIN,

MAX,

COUNT

Agregavimo operatoriai

SUM, AVG, MIN, MAX, COUNT

FILMAI (pavadinimas, metai, trukmė, spalvotas, studija, prodiuseris)
ATLIKEJAI (aktoriaus_vard, vaidmuo, filmo_pav, filmo_metai)
PRODIUSERIAI (kodas, vardas, adresas, pelnas)

```
SELECT SUM(pelnas)
FROM prodiuseriai;
```

```
SELECT vardas, MIN (pelnas)
FROM prodiuseriai;
```

```
SELECT COUNT(*)
FROM atlikejai;
```

```
SELECT COUNT(aktoriaus_vard)
FROM atlikejai;
```

```
SELECT COUNT(DISTINCT aktoriaus_vard)
FROM atlikejai;
```

Visi agregavimo operatoriai taikomi konkrečiam lentelės stulpeliui, išskyrus COUNT operatorių.

SQL SELECT grupavimas

SELECT [ALL | DISTINCT] <atributai>

FROM <lentelės>

[WHERE <sąlyga>]

[GROUP BY <grupavimo sąlyga>]

[HAVING <*where*_sąlyga>]

[ORDER BY <rikiavimo sąlyga> [ASC | DESC]];

Grupavimas (1)

FILMAI (pavadinimas, metai, trukmė, spalvotas, studija, prodiuseris)

```
SELECT studija,
SUM(trukme)
FROM filmai
GROUP BY studija;
```

```
SELECT studija,metai, SUM(trukme)
FROM filmai
GROUP BY studija, metai;
```

```
SELECT vardas, SUM(pelnas)
FROM prodiuseriai GROUP BY vardas
```

```
SELECT vardas, adresas, SUM(pelnas)
FROM prodiuseriai GROUP BY vardas
```

```
SELECT studija
FROM filmai
GROUP BY studija;
```

```
SELECT DISTINCT studija
FROM filmai;
```

ANSI SQL 92

Iki šio standarto buvo galima grupuoti tik tuos laukus, kurie yra atrenkamųjų sąraše, o atrenkamųjų sąraše negali būti laukų, kurie negrupuojami.

Grupavimas (2)

FILMAI (pavadinimas, metai, trukme, spalvotas, studija, prodiuseris)
PRODIUSERIAI (vardas, adresas, kodas, pelnas)

```
SELECT vardas, SUM(trukme)
FROM prodiuseriai INNER JOIN filmai
ON prodiuseris = kodas
GROUP BY vardas;
```

```
SELECT vardas, SUM(trukme)
FROM prodiuseriai, filmai
WHERE prodiuseris = kodas
GROUP BY vardas;
```

```
SELECT vardas, SUM(trukme) AS trukme
FROM prodiuseriai INNER JOIN filmai
ON prodiuseris = kodas
GROUP BY vardas;
```

```
SELECT vardas, SUM(trukme) AS trukme
FROM prodiuseriai INNER JOIN filmai
ON prodiuseris = kodas
GROUP BY vardas
ORDER BY trukme
```

Raktinis žodis HAVING

FILMAI (pavadinimas, metai, trukme, spalvotas, studija, prodiuseris)
PRODIUSERIAI (vardas, adresas, kodas, pelnas)

```
SELECT vardas, SUM(trukme)
FROM prodiuseriai INNER JOIN filmai
ON prodiuseris = kodas
GROUP BY vardas
HAVING MIN(metai) < 1930;
```

```
SELECT
   uzsakymo_numeris,
   COUNT(prekes_id) AS prekiu_kiekis,
   SUM(prekes_kaina) AS viso_kaina
FROM
   uzsakymo_detales
GROUP BY uzsakymo_numeris
HAVING viso_kaina > 900;
```

Svarbu:

- Sąlyga HAVING dalyje taikoma tik įrašams papuolusiems į grupę.
- tik tie atributai, kurie yra FROM ir SELECT dalyje gali būti agreguoti HAVING dalyje
- tik tie atributai, kurie yra GROUP BY dalyje negali būti agreguoti HAVING dalyje.
- HAVING nenaudojamas be GROUP BY.

Grupavimas, agregavimas ir tuščios reikšmės

- Reikšmė NULL agregate ignoruojama. COUNT(*) duos įrašų skaičių lentelėje, bet COUNT (A) duos įrašų, kuriuose stulpelis A nėra tuščias, skaičių.

- NULL reikšmė normaliai traktuojama grupavimo atributuose.

ktu informatikos fakultetas 1922

SQL DML

INSERT,

UPDATE,

DELETE

SQL DML

- **Manipuliavimo duomenimis kalba** (Data manipulation language)

Apima viską, ką galima atlikti su:

- SELECT
- INSERT
- UPDATE
- DELETE

INSERT

INSERT INTO <lentelės vardas>(stulpelis1, stulpelis2...)

VALUES (reikšmė1, reikšmė2,...);

- Lentelės stulpelių eiliškumas gali būti pateikiamas pagal jūsų poreikius;
- Stulpelių ir reikšmių eiliškumas turi sutapti.

```
INSERT INTO darbuotojai (tabelio_nr,vardas,pavarde,fk_nuomos_biuras)
VALUES (,8945‘,‘Timon‘,‘Thomas‘,45);
```

INSERT

INSERT INTO <lentelės vardas>(stulpelis1, stulpelis2...)

VALUES (reikšmė1, reikšmė2,...),

(reikšmė1, reikšmė2,...),

.....;

```
INSERT INTO darbuotojai (tabelio_nr,vardas,pavarde,fk_nuomos_biuras)
 VALUES ("8945","Timon","Thomas",45),
("9981","Charity","Gomez",59),
("11114","Hop","Rhodes",54);
```

```
INSERT INTO klientai (asmens_kodas,vardas,pavarde,gimimo_data,telefonas,epastas)
VALUES
(‚42430878668‘,‘Sara‘,‘Wise‘,‘1969/12/31‘,‘370-856-90639‘,‘rutrum@isqueornare.org‘)
```

INSERT

INSERT INTO <lentelės vardas>

VALUES (reikšmė1, reikšmė2,...);

INSERT INTO <lentelės vardas>

VALUES (reikšmė1, reikšmė2,...),

(reikšmė1, reikšmė2,...),

.....;

- Jei reikšmes pateikiate visiems lentelės stulpeliams ir išlaikote eiliškumą, tada stulpelių vardų nurodyti nereikia;
- Jei turite AUTOINCREMENT stulpelius, jiems reikšmių pateikti nereikia.

INSERT INTO darbuotojai VALUES ("8945","Timon","Thomas",45);

INSERT

```
INSERT INTO lentelė1

SELECT stulpelis1, stulpelis2 from lentelė2 ;
```

```
INSERT INTO lentelė1

SELECT * from lentelė2 ;
```

```
INSERT INTO klientai (asmens_kodas, vardas, pavarde)
SELECT tabelio_nr, vardas, pavardė FROM darbuotojai
where tabelio_nr =‘8945‘;
```

INSERT ON DUPLICATE KEY

INSERT INTO <lentelės vardas>(stulpelių sąrašas)
VALUES(reikšmių sąrašas)
ON DUPLICATE KEY UPDATE stulpelis1 = nauja_reikšmė1, stulpelis2 = nauja_reikšmė2, …;

- Jei įterpimo metu pažeista pirminio rakto ar kita unikalumo taisyklė, ON DUPLICATE KEY reikšmė gali būti atnaujinamos pagal pateiktą logiką.

INSERT INTO darbuotojai (tabelio_nr,vardas,pavarde,fk_nuomos_biuras)
VALUES (‘8945‘, ‘Timon‘, ‘Thomas‘,45)
ON DUPLICATE KEY tabelio_NR= CONCAT (tabelio_nr, ‘B‘)

LAST_INSERT_ID function

SELECT LAST_INSERT_ID();

- Grąžina paskutinio įterpto įrašo ID, kurį sukūrė DB variklis, jei laukas yra AUTOINCREMENT.
- Jei vienos INSERT komandos metų įterpėte daug eilučių, bus grąžintas pirmos įterptos eilutės ID.

```
INSERT INTO markes (pavadinimas) VALUES
('Abarth'),
('Aixam'),
('Alfa Romeo'),
('Audi'),
('BMW'),
('Citroen');
```

#	Pavadinimas	Tipas	Palyginimas	Atributai	Null	Nutylint	Papildomai
1	pavadinimas	varchar(20)	utf8_unicode_ci		Ne	*Jokio*	
2	id	int(11)			Ne	*Jokio*	AUTO_INCREMENT

UPDATE

```
UPDATE [LOW_PRIORITY] [IGNORE] <lentelės vardas>
SET
   stulpelis1 = reikšmė,
   stuleplis2 = reikšmė2,
   ...
WHERE
   [sąlyga];
```

- [LOW_PRIORITY] - veikia [MyISAM, MERGE, MEMORY] DB varikliuose. Duomenų atnaujinimas atidedamas, kol neliks nė vieno prisijungimo prie DB, kuris skaito lentelės duomenis ;
- [IGNORE] – vykdys atnaujinimą net esant klaidoms, klaidas išsaukusios eilutės nebus atnaujintos.

UPDATE

```
UPDATE nuomos_biurai SET pavadinimas='Avis' WHERE id >=0 and id <
11;
```

```
UPDATE `nuomos_biurai` SET `fk_aukstesnis_padalinys`=1 WHERE pavadinimas= 'Avis';
```

```
UPDATE `nuomos_biurai` SET `fk_aukstesnis_padalinys`=1
```

UPDATE per susietas lenteles

```sql
CREATE TABLE orders (

    order_id INT PRIMARY KEY,
    customer_name VARCHAR(100),
    order_date DATETIME,
    total_orders INT
);
INSERT INTO  orders
SELECT 1, 'Jack', '2020-02-03', 4 UNION
ALL
SELECT 2, 'Rose', '2020-01-09', 19;

CREATE TABLE order_details (
    order_detail_id INT  PRIMARY KEY,
    order_id VARCHAR(100),
    item VARCHAR(100)
);

INSERT INTO  order_details
SELECT 1, 1, 'laptop' UNION ALL
SELECT 2, 2, 'mouse';
```

```sql
--MySQL
UPDATE orders o
INNER JOIN order_details od
  ON o.order_id = od.order_id
SET o.total_orders = 7
    ,item= 'pendrive'
WHERE o.order_id = 1
  AND order_detail_id = 1;

--MS SQL
  UPDATE o
SET total_orders = 7
FROM orders o
INNER JOIN order_details od
    ON o.order_id = od.order_id
WHERE customer_name = 'Jack';
```

UPDATE per susietas lenteles

```
UPDATE darbuotojas
       INNER JOIN
   premijos ON darbuotojas.premijos_id = premijos.ID
SET
   atlyginimas = atlyginimas + atlyginimas * premijos.premijos_dydis;
```

```
UPDATE `aiksteles` SET `pavadinimas` = CONCAT( CONCAT( (
SELECT miestai.pavadinimas
FROM miestai
WHERE miestai.id = aiksteles.fk_miestas
), '_' ) , CEIL( RAND( ) *90 ) )
```

UPDATE per susietas lenteles

```
 UPDATE[atr_NMIP00_Barelis]
   SET
     [atr_NMIP00_Barelis].[c_BrAdresas_AtsIkiKelio] =
t1.[c_BrAdresas_AtsIkiKelio],
[atr_NMIP00_Barelis].[c_BrAdresas_AtsIkiGriovio]=
t1.[c_BrAdresas_AtsIkiGriovio]
  from (SELECT apskaitinisKodas, matavimoMetai,
[c_BrAdresas_AtsIkiKelio],[c_BrAdresas_AtsIkiGriovio] FROM
[atr_NMIP00_Barelis] where matavimoMetai = '2017' and egzPaskirtis =
'p') t1
 WHERE [atr_NMIP00_Barelis].apskaitinisKodas
 = t1.apskaitinisKodas
   AND [atr_NMIP00_Barelis].matavimoMetai = '2022'
```

UPDATE per susietas lenteles

```
 UPDATE   atr_NMIP00_Barelis__Sektorius__Medis
SET
  atr_NMIP00_Barelis__Sektorius__Medis.c_MedisBendra_Atstumas  = t1.c_MedisBendra_Atstumas,
  atr_NMIP00_Barelis__Sektorius__Medis.c_MedisBendra_Azimutas = t1.c_MedisBendra_Azimutas
       from   atr_NMIP00_Barelis__Sektorius__Medis RIGHT OUTER JOIN  atr_NMIP00_Barelis__Sektorius ON
atr_NMIP00_Barelis__Sektorius__Medis.tevoId =  atr_NMIP00_Barelis__Sektorius.atr_NMIP00_Barelis__Sektorius_id
RIGHT OUTER JOIN
                        atr_NMIP00_Barelis ON  atr_NMIP00_Barelis__Sektorius.tevoId =
atr_NMIP00_Barelis.atr_NMIP00_Barelis_id,
(SELECT         atr_NMIP00_Barelis.apskaitinisKodas,  atr_NMIP00_Barelis.matavimoMetai,
atr_NMIP00_Barelis__Sektorius__Medis.apskaitinisNr,  atr_NMIP00_Barelis.egzPaskirtis,
 atr_NMIP00_Barelis__Sektorius__Medis.egzPaskirtis AS Expr1,
atr_NMIP00_Barelis__Sektorius__Medis.c_MedisBendra_Atstumas,
atr_NMIP00_Barelis__Sektorius__Medis.c_MedisBendra_Azimutas
FROM            atr_NMIP00_Barelis__Sektorius__Medis RIGHT OUTER JOIN atr_NMIP00_Barelis__Sektorius ON
atr_NMIP00_Barelis__Sektorius__Medis.tevoId =  atr_NMIP00_Barelis__Sektorius.atr_NMIP00_Barelis__Sektorius_id
RIGHT OUTER JOIN atr_NMIP00_Barelis ON  atr_NMIP00_Barelis__Sektorius.tevoId =
atr_NMIP00_Barelis.atr_NMIP00_Barelis_id
WHERE          atr_NMIP00_Barelis.matavimoMetai = '2017' and  atr_NMIP00_Barelis.egzPaskirtis = 'p') t1

WHERE [nmip00].[atr_NMIP00_Barelis].apskaitinisKodas
 = t1.apskaitinisKodas and  atr_NMIP00_Barelis__Sektorius__Medis.apskaitinisNr = t1.apskaitinisNr
   AND [nmip00].[atr_NMIP00_Barelis].matavimoMetai = '2022'
```

REPLACE

REPLACE INTO <lentelės vardas>(stulpelis1, stulpelis2...)
VALUES (reikšmė1, reikšmė2,...);

- REPLACE veikia, kaip INSERT, tik jei aptinkamas duomenų įrašas su tokia pat PRIMARY KEY arba UNIQUE indekso reikšme , tada sena eilutė pašalinama, o nauja įterpiama;
- REPLACE neveiks, jei neturės pirminio rakto arba unikalaus indekso;

REPLACE INTO darbuotojai (tabelio_nr,vardas,pavarde,fk_nuomos_biuras)
VALUES (‘8945‘, ‘Timon‘, ‘Thomas‘,45)

DELETE

DELETE FROM lentelė

[WHERE sąlyga]

```
DELETE FROM  nuomos_biurai WHERE id >=0
and id < 11;
```

ROW_COUNT()

- Sisteminė funkcija, kuri grąžiną informaciją, kiek įrašų buvo paveikta atliekant INSERT, UPDATE, DETELE arba REPLACE užklausą.

DELETE per susietas lenteles

```
DELETE darbuotojai,
  nuomos_biurai
FROM darbuotojai INNEr JOIN nuomos_biurai  ON
darbuotojai.fk_nuomos_biuras = nuomos_biuras.ID WHERE
 nuomos_biuras.ID = 1
```

- Pašalina tik tuos įšašus iš darbuotojai lentelės, kurie susiję su nuomos biuriu, kurio ID=1 ir patį biuro įrašą.

TRUNCATE TABLE

TRUNCATE TABLE nuomos_biurai;

- Išvalo visus lentelės įrašus ir nustato į pirminę reikšmę AUTOINCREMENT identifikatorių;
- Jei lentelė turi sąsają su „vaikine“ lentelę ir jos įrašais, lentelės valymas nevykdomas.