**SQL injekcijos(angl. SQL**

**injection)**

Naudojant paruoštas uzklausas ir MySQLi, nereikia naudoti

mysqli\_real\_escape\_string ir t.t. Už mus viską tai padaro PHP.

Object oriented style

string mysqli::real\_escape\_string ( string $escapestr )

Procedural style

string mysqli\_real\_escape\_string ( mysqli $link , string $escapestr )

**ĮTERPTINĖS UŽKLAUSOS**

**PVZ**

users.php:

<?php

// Prisijungimas prie db.

$id = $\_GET['id'];

$query = “SELECT \* FROM users WHERE UserId = '$id'”;

$rez = mysql\_query($query);

// galutinis rezultatas

?>

users.php?=1 , tai rezultate pamatysime vartotojo su id=1 duomenis.

O tarkim users.php?=1' or 1=1″;– rezultate to pamatysime visų vartotojų

duomenis.

**ĮTERPTINĖS UŽKLAUSOS**

• SQL-Injekcija yra viena iš daugelio web skript'ų atakų. Tai vienas

paprasčiausių ir populiariausių web atakų metodų. Šis

pažeidžiamumas dažniausiai sutinkamas PHP ir kituose web

skirptuose.

• Pagrindinė bėda, jei web aplikacija parametrus perduoda iš

nepatikimų šaltinių (pavyzdžiui formos) deda tiesiai į SQL skriptą,

nepatikrinus SQL raktinių žodžių, simbolių, backslashai arba kabutės.

• Tokio modifikuoto skripto pagalba „blogasis lankytojas“ gali

‘patobulinti’ jūsų

SQL užklausą ir sukurti duomenis su neteisingais duomenimis, arba

perimti iš duomenų bazės slaptus duomenis arba sugadinti jūsų

duomenis.

**SQL apribojimų tipai:**

• NOT NULL – nurodo, kad lentelės stulpeliuose negali būti saugomos NULL reikšmės;

• UNIQUE – užtikrina, kad kiekviena stulpelio ar jų rinkinio reikšmė turėtų unikalią

reikšmę;

• PRIMARY KEY – t.y. NOT NULL ir UNIQUE apribojimo derinys, šis apribojimas

užtikrina, kad stulpelis arba stulpelių derinys užtikrintų unikalią reikšmę, pagal kurią

galima identifikuoti konkretų duomenų bazės lentelės įrašą;

• FOREIGN KEY – užtikrina duomenų bazės lentelių tarpusavio reikšmių integralumą,

t.y. tikrina, kad susijusių lentelių stulpelių reikšmės tarpusavyje derėtų;

• CHECK – užtikrina, kad stulpelio reikšmė tenkintų konkrečias sąlygas;

• DEFAULT – nusako stulpelio reikšmę pagal nutylėjimą.

**Apribojimai**

SQL apribojimai naudojami aprašyti lentelių duomenų kontrolės taisykles.

Jei įvyksta prieštara tarp duomenų DML užklausų ir apribojimų, duomenų modifikavimo užklausos

atšaukiamos.

Apribojimai gali būti sukurti:

•kuriant duomenų bazės lentelę (CREATE TABLE);

•arba modifikuojant (ALTER table).

CREATE TABLE <Lentelės vardas>

(

<1\_stuleplio\_vardas> <duomenų tipas>[(dydis)] [apribojimas]

<2\_stuleplio\_vardas> <duomenų tipas>[(dydis)] [apribojimas]

<3\_stuleplio\_vardas> <duomenų tipas>[(dydis)] [apribojimas]

....

);

**Indekso kūrimas**

CREATE [ UNIQUE ] [ CLUSTERED |

NONCLUSTERED ] INDEX index\_name ON

<object> ( column [ ASC | DESC ] [ ,...n ] )

DROP INDEX index\_name

**Grupiniai ir paprasti indeksai**

Paprasti indeksai naudojami:

kai užklauso gražina nedidels duomenų aibes;

kai naudojamos tikslios užklausos (where dalyje nurodomos konkrečios duomenų reikšmės);

kai indeksas apima keletą ir daugiau stulpelių, pagal kuriuos atliekamas duomenų filtravimas;

abu indeksų tipai tinkamai, kai stulpelių duomenys pasižymi dideliu unikalumu;

paprastų indeksų naudojimas patartinas, išorinių raktų stulpeliams, kai yra santykis vienas:vienu;

Abu indeksų tipai vienodai gerai veikia ir reikalingi group by dalies stulpeliams;

Grupiniai indeksai naudojami:

kai užklauso gražina dideles duomenų aibes;

kai duomenų filtravimo kriterijai užduodami diapazonais, aibėmis, apytiklėmis reikšmėmis operatorius

(>,<, like ir .t.t);

grupinių indeksų naudojimas rekomenduojamas, išorinių raktų stulpeliams, kai yra santykis vienas:daug;

Indeksas pasitarnauja, jei naudojami užklausose order by;

**Užklausų optimizatorius**

Užklausų optimizatorius (query optimaizer) – jis įvertimas visus galimus

užklausos vykdymo variantus bandydamas aptikti tinkamiausią.

Pagrindinis veikimo principas:

Pirmiausia jis įvertina atrenkamų duomenų struktūrą;

Analizuoja ar atrenkami duomenys turi indeksus;

Įvertina tinkamiausius indeksus ir sudaro užklausos vykdymo

planą;

Jei nustatoma, kad užklausa gražina ženklią dalį duomenų iš

galimo aibės, tai tam tikrom lentelėm užklausų optimizatorius gali

ir nepanaudoti indekso.

**Indeksų tipai**

Grupiniai indeksai – duomenys yra saugomi surūšiuoti pagal indeksą;

Paprastas indeksas – suformuojamas surūšiuotų nuorodų sąrašas, pagal

kuri greitai randami reikiami duomenys;

Unikalūs indeksai;

Full text indeksas - pilnos tekstinės paieškos indeksai;

Erdvinių duomenų indeksai;

XML indeksas – paieškai xml duomenų tipo laukuose.

**Indeksai**

Padidina užklausų našumą (select,update,delete);

Indeksai yra kuriami stulpeliams;

Indeksai kuriame pirminiams raktams arba esant unikalumo

apribojimams;

**Schemos surišimas (shemabinding)**

Teikia pranešimus apie bandomus redaguoti lentelių struktūrą.

Susiejami atitinkami virtualios lentelės ir fizinės lentelės stulpeliai ir

juos redaguojant gaunamas perspėjimas.

CREATE VIEW [dbo].[v\_stud\_paz] with schemabinding AS

SELECT ... FROM ...

**Padalintos virtualios lentelės**

Kai užklausa gražina daug duomenų;

Užklausos vykdymas vyksta ilgai;

Duomenų apdorojimui naudojamas paskirstytas

sprendimas;

**Indeksuotos virtualios lentelės**

Kai užklausa gražina daug duomenų naudojant

agregavimo funkcijas;

Duonmenų bazėje išsaugomas lentelių

apjungimo ir agregavimo rezultatas;

CREATE VIEW virtualios lentelės\_vardas

[(atributas [,…n])] with schemabinding

Create unique clustered index on <poschemes

vardas> (< lauko pavadinimas>)

Pastabos: virtuali lentelė gali būti naudojama

modifikavimui , jei:

◦select\_sakinys neturi agregatinių funkcijų Select

sąraše, o taip pat neturi TOP, GROUP BY, UNION, arba

DISTINCT operatorių. Kol reikšmės

nemodifikuojamos, agreguotos funkcijos gali būti

naudojamos subužklausose FROM sakinyje.

◦select\_sakinys neturi išvestinių laukų Select sąraše.

Išvestiniai laukai sudaryti naudojant funkcijas,

sudėties, atimties ar pan. operacijas.

◦FROM operatorius nurodo bent vieną lentelę, o

išrenkami duomenys nėra vien tik ne iš lentelės

išvedamos reikšmės ar ne lentelių pagrindu

suformuojami rezultatai.

SELECT sakinyje galima naudoti:

◦funkcijas;

◦keletą SELECT sakinių, atskirtų UNION operatoriumi.

WITH CHECK OPTION

užtikrina, kad visos duomenų modifikacijos,

atliekamos su virtualia lentele, tenkintų Select sakinio

kriterijus. Kai duomenys modifikuojami per virtualią

lentelę, tai WITH CHECK OPTION garantuoja, kad

eilutės duomenys pasiliks matomi ir atlikus pakeitimą.

select\_sakinys

SELECT sakinys virtualiai lentelei aprašyti. Ji gali

susidėti iš daugiau nei vienos lentelės arba kitų

viertualių lentelių. Virtualią lentelę sudarančioms

lentelėms ir kitoms virtualioms lentelės turi būti

nustatytos atitinkamos vartotojo teisės.

Nors SELECT sakinys gali būti bet kokio sudėtingumo,

tačiau yra tam tikri apribojimai.

Negalima naudoti:

◦ ORDER BY, COMPUTE BY operatorių;

◦INTO raktinio žodžio;

◦laikinos lentelės nuorodos

Pastaba:

Naudojant ALTER VIEW su CREATE VIEW sukurta

virtuali lentelė gali būti pakeista, neįtakojant susietų

procedūrų ir trigerių, tačiau pakeičiant ankstesnius

atributų vardus kitais, pakeičiant apribojimo sąlygas

WITH ENCRYPTION

Užkoduoja sisteminėje lentelėje esantį tekstą su

CREATE VIEW išraiška

AS

virtualios lentelės apibrėžimas

**atributas**

◦atributo vardą būtina suteikti tik tuomet, kai jis

yra aritmetinės išraiškos rezultatas, funkcija ar

konstanta, kai du atributai turi vienodus vardus

(Join atveju), kai atributui virtualioje lentelėje

norime suteikti kitą vardą, nei jis buvo DB

lentelėje.

Atributų vardus galima nurodyti ir Select -sakinyje

Jei atributų vardų nenurodome, tai jie paveldimi iš

Select\_sakinio.

**Virtualių lentelių tipai**

Standartinės virtualios lentelės;

Indeksuotos (indexed) poschemos; (Sukuriamas fizinis indeksas, ir

virtuali lentelė saugoma bazėje)

Padalintos virtualio lentelės (partitioned) (Paskirtymas tarp duomenų

bazių, serverių)

◦ Lokalios;

◦ Paskirstytos;

**MySQL virtulių lentelių**

**apribojimai**

• Virtualioms lentelėms, negalima sukurti indeksų;

• Iki Mysql 5.7.7 versijos nebuvo galim naudoti užklausų

užklausose, kai konstruojamos virtualios lentelės;

• MySql nepalaiko materializuotų virtualių lentelių;

• Tik be join virtualios lentelės gali būti redaguojamo

MySQL virtuli redaguojama

lentelė negali turėti

• MIN, MAX, SUM, AVG, COUNT;

• DISTINCT;

• GROUP BY ;

• HAVING;

• UNION arba UNION ALL;

• Left join or outer join;

• Nuorodas į neredaguojamas lenteles.

**Virtulių lentelių minusai**

• Našumas lėtokas ypač jei virtualios lentelės

konstruojamos kitų virtualių lentelių pagrindu;

• Lentelių priklausomybių palaikymas, jei pasikeičia DB

struktūra turėsite pertvarkyti ir virtualias lenteles

**Kodėl verta naudoti virtualias**

**lenteles**

• Padeda supaprastinti sudėtingas užklausas arba paslėpti

DB schemos sudėtingumą;

• Suteikia priemones naudotojams apriboti duomenų

pasiekimą konkretiems naudotojams;

• Suteikia dar vieną saugumo sluoksnį, ypač kai kalbama

apie duomenų redagavimą;

• Suteikia galimybę iškart turėti išvestinius (agreguotus)

laukus;

• Virtualios lentelės suteikia galimybę sistemų

suderinamumui realizuoti.

**Virtualios lentelės**

Virtualios lentelės (angl. view) - tai DB loginė lentelė, atvaizduojanti vienos arba kelių DB

lentelių duomenis norimu pjūvių. virtualios lentelės užtikrina vartotojų teisių apribojimą,

suteikdamos galimybę naudotis tik virtualioje lentelėje apibrėžtais duomenimis.

CREATE

[ALGORITHM = {MERGE | TEMPTABLE | UNDEFINED}]

VIEW [database\_name].[view\_name]

AS

[SELECT statement]

CREATE VIEW virtualios lentelės\_vardas [(atributas [,…n])]

[WITH ENCRYPTION]

AS

select\_sakinys

[WITH CHECK OPTION]

**TRUNCATE TABLE**

TRUNCATE TABLE nuomos\_biurai;

 Išvalo visus lentelės įrašus ir nunulina

AUTOINCREMENT identifikatorių;

 Jei lentelė turi sąsają su „vaikine“ lentelę ir jos įrašas,

lentelės valymas nevykdonas

**DELETE**

DELETE FROM lentelė

[WHERE sąlyga]

2016-04-20 DB 10 PASKAITA T. DANIKAUSKAS 22

DELETE FROM nuomos\_biurai WHERE id >=0

and id < 11;

ROW\_COUNT()

Sisteminė funkcija, kuri grąžiną informaciją, kiek įrašų buvo paveikta atliekant

INSERT, UPDATE, DATELE, SELECT arba REPLACE užklausą.

**REPLACE**

REPLACE <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;

**UPDATE**

UPDATE [LOW\_PRIORITY] [IGNORE] <lentelės vardas>

SET

stulpelis1 = reikšmė,

stuleplis2 = reikšmė2,

...

WHERE

[sąlyga];

2016-04-20 DB 10 PASKAITA T. DANIKAUSKAS 18

[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.

**INSERT**

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

VALUES (reikšmė1, reikšmė2,...);

2016-04-20 DB 10 PASKAITA T. DANIKAUSKAS 12

Lentelės stulpelių eiliškumas gali būti pateikiamas pagal jūsų poreikius;

Stulpelių ir reikšmių eiliškumas turi sutapti.

**SQL DML**

◦ Manipuliavimo duomenimis kalba (Data manipulation

language)

Apima viską, ką galima atlikti su:

◦ SELECT

◦ INSERT

◦ UPDATE

◦ DELETE

**RIGHT JOIN**

Lentelių apjungimas atrenka visus yra iš prijungiamos

lentelės ir tik susijusios iš pasirinktos.

SELECT

T1.c1, T1.c2, T2.c1, T2.c2

FROM

T1

RIGTH JOIN

T2 ON T1.c1 = T2.c1;

**LEFT JOIN**

Antras pagal dažnumą naudojamas apjungimo variantas.

Lentelių apjungimas atrenka visus įrašus iš pasirinktos

lentelės ir tik susijusiu iš prijungiamos lentelės.

SELECT

T1.c1, T1.c2, T2.c1, T2.c2

FROM

T1

LEFT JOIN

T2 ON T1.c1 = T2.c1;

**INNER JOIN**

Dažniausia naudojimas lentelių apjungimas.

Lentelių apjungimas atrenka tik susijusius lentelių įrašus.

SELECT stulpeliai

FROM t1

INNER JOIN t2 ON join\_condition1

INNER JOIN t3 ON join\_condition2

...

WHERE where\_conditions;

**SQL SELECT sakinys**

SELECT [ALL | DISTINCT] <atributai>

FROM <lentelės>

[WHERE <sąlyga>]

[GROUP BY <grupavimo sąlyga>]

[HAVING <where\_sąlyga>]

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

CREATE TABLE TableName

{(columnName dataType [NOT NULL] [UNIQUE]

[DEFAULT defaultOption][,...]}

[PRIMARY KEY (listOfColumns),]

{[UNIQUE (listOfColumns),] […,]}

{[FOREIGN KEY (listOfFKColumns)

REFERENCES ParentTableName [(listOfCKColumns)],

[ON UPDATE referentialAction]

[ON DELETE referentialAction ]] [,…]}

SQL DDL

•SQL sakinį sudaro rezervuoti žodžiai ir varto-tojo

apibrėžti žodžiai.

•Rezervuoti žodžiai: pastovi SQL dalis. Jie turi

būti parašyti tiksliai. Perkėlimai į kitą eilutę

negalimi.

•Vartotojo sukurti žodžiai: tai įvairių DB

objektų vardai, pvz., lentelių, stulpelių,

parodymų [views].

Duomenų struktūrų aprašymo kalba (Data Definition

Language)

**SQL injekcijos(angl. SQL**

**injection)**

Naudojant paruoštas uzklausas ir MySQLi, nereikia naudoti

mysqli\_real\_escape\_string ir t.t. Už mus viską tai padaro PHP.

Object oriented style

string mysqli::real\_escape\_string ( string $escapestr )

Procedural style

string mysqli\_real\_escape\_string ( mysqli $link , string $escapestr )

**ĮTERPTINĖS UŽKLAUSOS**

**PVZ**

users.php:

<?php

// Prisijungimas prie db.

$id = $\_GET['id'];

$query = “SELECT \* FROM users WHERE UserId = '$id'”;

$rez = mysql\_query($query);

// galutinis rezultatas

?>

users.php?=1 , tai rezultate pamatysime vartotojo su id=1 duomenis.

O tarkim users.php?=1' or 1=1″;– rezultate to pamatysime visų vartotojų

duomenis.

**ĮTERPTINĖS UŽKLAUSOS**

• SQL-Injekcija yra viena iš daugelio web skript'ų atakų. Tai vienas

paprasčiausių ir populiariausių web atakų metodų. Šis

pažeidžiamumas dažniausiai sutinkamas PHP ir kituose web

skirptuose.

• Pagrindinė bėda, jei web aplikacija parametrus perduoda iš

nepatikimų šaltinių (pavyzdžiui formos) deda tiesiai į SQL skriptą,

nepatikrinus SQL raktinių žodžių, simbolių, backslashai arba kabutės.

• Tokio modifikuoto skripto pagalba „blogasis lankytojas“ gali

‘patobulinti’ jūsų

SQL užklausą ir sukurti duomenis su neteisingais duomenimis, arba

perimti iš duomenų bazės slaptus duomenis arba sugadinti jūsų

duomenis.

**SQL apribojimų tipai:**

• NOT NULL – nurodo, kad lentelės stulpeliuose negali būti saugomos NULL reikšmės;

• UNIQUE – užtikrina, kad kiekviena stulpelio ar jų rinkinio reikšmė turėtų unikalią

reikšmę;

• PRIMARY KEY – t.y. NOT NULL ir UNIQUE apribojimo derinys, šis apribojimas

užtikrina, kad stulpelis arba stulpelių derinys užtikrintų unikalią reikšmę, pagal kurią

galima identifikuoti konkretų duomenų bazės lentelės įrašą;

• FOREIGN KEY – užtikrina duomenų bazės lentelių tarpusavio reikšmių integralumą,

t.y. tikrina, kad susijusių lentelių stulpelių reikšmės tarpusavyje derėtų;

• CHECK – užtikrina, kad stulpelio reikšmė tenkintų konkrečias sąlygas;

• DEFAULT – nusako stulpelio reikšmę pagal nutylėjimą.

**Apribojimai**

SQL apribojimai naudojami aprašyti lentelių duomenų kontrolės taisykles.

Jei įvyksta prieštara tarp duomenų DML užklausų ir apribojimų, duomenų modifikavimo užklausos

atšaukiamos.

Apribojimai gali būti sukurti:

•kuriant duomenų bazės lentelę (CREATE TABLE);

•arba modifikuojant (ALTER table).

CREATE TABLE <Lentelės vardas>

(

<1\_stuleplio\_vardas> <duomenų tipas>[(dydis)] [apribojimas]

<2\_stuleplio\_vardas> <duomenų tipas>[(dydis)] [apribojimas]

<3\_stuleplio\_vardas> <duomenų tipas>[(dydis)] [apribojimas]

....

);

**Indekso kūrimas**

CREATE [ UNIQUE ] [ CLUSTERED |

NONCLUSTERED ] INDEX index\_name ON

<object> ( column [ ASC | DESC ] [ ,...n ] )

DROP INDEX index\_name

**Grupiniai ir paprasti indeksai**

Paprasti indeksai naudojami:

kai užklauso gražina nedidels duomenų aibes;

kai naudojamos tikslios užklausos (where dalyje nurodomos konkrečios duomenų reikšmės);

kai indeksas apima keletą ir daugiau stulpelių, pagal kuriuos atliekamas duomenų filtravimas;

abu indeksų tipai tinkamai, kai stulpelių duomenys pasižymi dideliu unikalumu;

paprastų indeksų naudojimas patartinas, išorinių raktų stulpeliams, kai yra santykis vienas:vienu;

Abu indeksų tipai vienodai gerai veikia ir reikalingi group by dalies stulpeliams;

Grupiniai indeksai naudojami:

kai užklauso gražina dideles duomenų aibes;

kai duomenų filtravimo kriterijai užduodami diapazonais, aibėmis, apytiklėmis reikšmėmis operatorius

(>,<, like ir .t.t);

grupinių indeksų naudojimas rekomenduojamas, išorinių raktų stulpeliams, kai yra santykis vienas:daug;

Indeksas pasitarnauja, jei naudojami užklausose order by;

**Užklausų optimizatorius**

Užklausų optimizatorius (query optimaizer) – jis įvertimas visus galimus

užklausos vykdymo variantus bandydamas aptikti tinkamiausią.

Pagrindinis veikimo principas:

Pirmiausia jis įvertina atrenkamų duomenų struktūrą;

Analizuoja ar atrenkami duomenys turi indeksus;

Įvertina tinkamiausius indeksus ir sudaro užklausos vykdymo

planą;

Jei nustatoma, kad užklausa gražina ženklią dalį duomenų iš

galimo aibės, tai tam tikrom lentelėm užklausų optimizatorius gali

ir nepanaudoti indekso.

**Indeksų tipai**

Grupiniai indeksai – duomenys yra saugomi surūšiuoti pagal indeksą;

Paprastas indeksas – suformuojamas surūšiuotų nuorodų sąrašas, pagal

kuri greitai randami reikiami duomenys;

Unikalūs indeksai;

Full text indeksas - pilnos tekstinės paieškos indeksai;

Erdvinių duomenų indeksai;

XML indeksas – paieškai xml duomenų tipo laukuose.

**Indeksai**

Padidina užklausų našumą (select,update,delete);

Indeksai yra kuriami stulpeliams;

Indeksai kuriame pirminiams raktams arba esant unikalumo

apribojimams;

**Schemos surišimas (shemabinding)**

Teikia pranešimus apie bandomus redaguoti lentelių struktūrą.

Susiejami atitinkami virtualios lentelės ir fizinės lentelės stulpeliai ir

juos redaguojant gaunamas perspėjimas.

CREATE VIEW [dbo].[v\_stud\_paz] with schemabinding AS

SELECT ... FROM ...

**Padalintos virtualios lentelės**

Kai užklausa gražina daug duomenų;

Užklausos vykdymas vyksta ilgai;

Duomenų apdorojimui naudojamas paskirstytas

sprendimas;

**Indeksuotos virtualios lentelės**

Kai užklausa gražina daug duomenų naudojant

agregavimo funkcijas;

Duonmenų bazėje išsaugomas lentelių

apjungimo ir agregavimo rezultatas;

CREATE VIEW virtualios lentelės\_vardas

[(atributas [,…n])] with schemabinding

Create unique clustered index on <poschemes

vardas> (< lauko pavadinimas>)

Pastabos: virtuali lentelė gali būti naudojama

modifikavimui , jei:

◦select\_sakinys neturi agregatinių funkcijų Select

sąraše, o taip pat neturi TOP, GROUP BY, UNION, arba

DISTINCT operatorių. Kol reikšmės

nemodifikuojamos, agreguotos funkcijos gali būti

naudojamos subužklausose FROM sakinyje.

◦select\_sakinys neturi išvestinių laukų Select sąraše.

Išvestiniai laukai sudaryti naudojant funkcijas,

sudėties, atimties ar pan. operacijas.

◦FROM operatorius nurodo bent vieną lentelę, o

išrenkami duomenys nėra vien tik ne iš lentelės

išvedamos reikšmės ar ne lentelių pagrindu

suformuojami rezultatai.

SELECT sakinyje galima naudoti:

◦funkcijas;

◦keletą SELECT sakinių, atskirtų UNION operatoriumi.

WITH CHECK OPTION

užtikrina, kad visos duomenų modifikacijos,

atliekamos su virtualia lentele, tenkintų Select sakinio

kriterijus. Kai duomenys modifikuojami per virtualią

lentelę, tai WITH CHECK OPTION garantuoja, kad

eilutės duomenys pasiliks matomi ir atlikus pakeitimą.

select\_sakinys

SELECT sakinys virtualiai lentelei aprašyti. Ji gali

susidėti iš daugiau nei vienos lentelės arba kitų

viertualių lentelių. Virtualią lentelę sudarančioms

lentelėms ir kitoms virtualioms lentelės turi būti

nustatytos atitinkamos vartotojo teisės.

Nors SELECT sakinys gali būti bet kokio sudėtingumo,

tačiau yra tam tikri apribojimai.

Negalima naudoti:

◦ ORDER BY, COMPUTE BY operatorių;

◦INTO raktinio žodžio;

◦laikinos lentelės nuorodos

Pastaba:

Naudojant ALTER VIEW su CREATE VIEW sukurta

virtuali lentelė gali būti pakeista, neįtakojant susietų

procedūrų ir trigerių, tačiau pakeičiant ankstesnius

atributų vardus kitais, pakeičiant apribojimo sąlygas

WITH ENCRYPTION

Užkoduoja sisteminėje lentelėje esantį tekstą su

CREATE VIEW išraiška

AS

virtualios lentelės apibrėžimas

**atributas**

◦atributo vardą būtina suteikti tik tuomet, kai jis

yra aritmetinės išraiškos rezultatas, funkcija ar

konstanta, kai du atributai turi vienodus vardus

(Join atveju), kai atributui virtualioje lentelėje

norime suteikti kitą vardą, nei jis buvo DB

lentelėje.

Atributų vardus galima nurodyti ir Select -sakinyje

Jei atributų vardų nenurodome, tai jie paveldimi iš

Select\_sakinio.

**Virtualių lentelių tipai**

Standartinės virtualios lentelės;

Indeksuotos (indexed) poschemos; (Sukuriamas fizinis indeksas, ir

virtuali lentelė saugoma bazėje)

Padalintos virtualio lentelės (partitioned) (Paskirtymas tarp duomenų

bazių, serverių)

◦ Lokalios;

◦ Paskirstytos;

**MySQL virtulių lentelių**

**apribojimai**

• Virtualioms lentelėms, negalima sukurti indeksų;

• Iki Mysql 5.7.7 versijos nebuvo galim naudoti užklausų

užklausose, kai konstruojamos virtualios lentelės;

• MySql nepalaiko materializuotų virtualių lentelių;

• Tik be join virtualios lentelės gali būti redaguojamo

MySQL virtuli redaguojama

lentelė negali turėti

• MIN, MAX, SUM, AVG, COUNT;

• DISTINCT;

• GROUP BY ;

• HAVING;

• UNION arba UNION ALL;

• Left join or outer join;

• Nuorodas į neredaguojamas lenteles.

**Virtulių lentelių minusai**

• Našumas lėtokas ypač jei virtualios lentelės

konstruojamos kitų virtualių lentelių pagrindu;

• Lentelių priklausomybių palaikymas, jei pasikeičia DB

struktūra turėsite pertvarkyti ir virtualias lenteles

**Kodėl verta naudoti virtualias**

**lenteles**

• Padeda supaprastinti sudėtingas užklausas arba paslėpti

DB schemos sudėtingumą;

• Suteikia priemones naudotojams apriboti duomenų

pasiekimą konkretiems naudotojams;

• Suteikia dar vieną saugumo sluoksnį, ypač kai kalbama

apie duomenų redagavimą;

• Suteikia galimybę iškart turėti išvestinius (agreguotus)

laukus;

• Virtualios lentelės suteikia galimybę sistemų

suderinamumui realizuoti.

**Virtualios lentelės**

Virtualios lentelės (angl. view) - tai DB loginė lentelė, atvaizduojanti vienos arba kelių DB

lentelių duomenis norimu pjūvių. virtualios lentelės užtikrina vartotojų teisių apribojimą,

suteikdamos galimybę naudotis tik virtualioje lentelėje apibrėžtais duomenimis.

CREATE

[ALGORITHM = {MERGE | TEMPTABLE | UNDEFINED}]

VIEW [database\_name].[view\_name]

AS

[SELECT statement]

CREATE VIEW virtualios lentelės\_vardas [(atributas [,…n])]

[WITH ENCRYPTION]

AS

select\_sakinys

[WITH CHECK OPTION]

**TRUNCATE TABLE**

TRUNCATE TABLE nuomos\_biurai;

 Išvalo visus lentelės įrašus ir nunulina

AUTOINCREMENT identifikatorių;

 Jei lentelė turi sąsają su „vaikine“ lentelę ir jos įrašas,

lentelės valymas nevykdonas

**DELETE**

DELETE FROM lentelė

[WHERE sąlyga]

2016-04-20 DB 10 PASKAITA T. DANIKAUSKAS 22

DELETE FROM nuomos\_biurai WHERE id >=0

and id < 11;

ROW\_COUNT()

Sisteminė funkcija, kuri grąžiną informaciją, kiek įrašų buvo paveikta atliekant

INSERT, UPDATE, DATELE, SELECT arba REPLACE užklausą.

**REPLACE**

REPLACE <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;

**UPDATE**

UPDATE [LOW\_PRIORITY] [IGNORE] <lentelės vardas>

SET

stulpelis1 = reikšmė,

stuleplis2 = reikšmė2,

...

WHERE

[sąlyga];

2016-04-20 DB 10 PASKAITA T. DANIKAUSKAS 18

[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.

**INSERT**

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

VALUES (reikšmė1, reikšmė2,...);

2016-04-20 DB 10 PASKAITA T. DANIKAUSKAS 12

Lentelės stulpelių eiliškumas gali būti pateikiamas pagal jūsų poreikius;

Stulpelių ir reikšmių eiliškumas turi sutapti.

**SQL DML**

◦ Manipuliavimo duomenimis kalba (Data manipulation

language)

Apima viską, ką galima atlikti su:

◦ SELECT

◦ INSERT

◦ UPDATE

◦ DELETE

**RIGHT JOIN**

Lentelių apjungimas atrenka visus yra iš prijungiamos

lentelės ir tik susijusios iš pasirinktos.

SELECT

T1.c1, T1.c2, T2.c1, T2.c2

FROM

T1

RIGTH JOIN

T2 ON T1.c1 = T2.c1;

**LEFT JOIN**

Antras pagal dažnumą naudojamas apjungimo variantas.

Lentelių apjungimas atrenka visus įrašus iš pasirinktos

lentelės ir tik susijusiu iš prijungiamos lentelės.

SELECT

T1.c1, T1.c2, T2.c1, T2.c2

FROM

T1

LEFT JOIN

T2 ON T1.c1 = T2.c1;

**INNER JOIN**

Dažniausia naudojimas lentelių apjungimas.

Lentelių apjungimas atrenka tik susijusius lentelių įrašus.

SELECT stulpeliai

FROM t1

INNER JOIN t2 ON join\_condition1

INNER JOIN t3 ON join\_condition2

...

WHERE where\_conditions;

**SQL SELECT sakinys**

SELECT [ALL | DISTINCT] <atributai>

FROM <lentelės>

[WHERE <sąlyga>]

[GROUP BY <grupavimo sąlyga>]

[HAVING <where\_sąlyga>]

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

CREATE TABLE TableName

{(columnName dataType [NOT NULL] [UNIQUE]

[DEFAULT defaultOption][,...]}

[PRIMARY KEY (listOfColumns),]

{[UNIQUE (listOfColumns),] […,]}

{[FOREIGN KEY (listOfFKColumns)

REFERENCES ParentTableName [(listOfCKColumns)],

[ON UPDATE referentialAction]

[ON DELETE referentialAction ]] [,…]}

SQL DDL

•SQL sakinį sudaro rezervuoti žodžiai ir varto-tojo

apibrėžti žodžiai.

•Rezervuoti žodžiai: pastovi SQL dalis. Jie turi

būti parašyti tiksliai. Perkėlimai į kitą eilutę

negalimi.

•Vartotojo sukurti žodžiai: tai įvairių DB

objektų vardai, pvz., lentelių, stulpelių,

parodymų [views].

Duomenų struktūrų aprašymo kalba (Data Definition

Language)