Rīgas Tehniska universitāte

Datorzinātnes un Informācijas tehnoloģijas fakultāte

Otrais praktiskais darbs mācību priekšmetā

„Datu bāzu vadības sistēmas”

Vaicājumu valoda SQL

Izstrādāja: Sergejs Terentjevs

2. kurss, 9.grupa

Apl.nr. 061RDB140

Pārbaudīja: lekt. V. Vinogradova

2006./2007. māc. g.

# ANOTĀCIJA

Laboratorijas darbā gaitā tika izstrādāti vairāki SQL valodas vaicājumi MS Access 2007 vidē.

Laboratorijas darbā pārskatā ir aprakstīti SQL loģiskie operatori, datu manipulēšanas, izgūšanas, grupēšanas, kārtošanas un nosacījumu SQL vaicājumi, tika definētas uzdevumu prasības un parādīti šo uzdevumu risinājumi.

Laboratorijas darba pārskata apjoms: 23 lappuses izklāsta tekstā, tajā ietilpst 6 nodaļas un 8 apakšnodaļas un 9 apakš apakšnodaļas , literatūras saraksts, 30. Attēli.

# 

# SATURS

[ANOTĀCIJA 2](#_Toc193521733)

[SATURS 3](#_Toc193521734)

[Uzdevuma nostādne 4](#_Toc193521735)

[Tabulu ievaddatu īss raksturojums 5](#_Toc193521736)

[Datu manipulēšanas vaicājumi 8](#_Toc193521737)

[Jauna raksta ievietošana esošā tabulā (*INSERT*) 8](#_Toc193521738)

[Lauku vērtības maiņa (*UPDATE*) 8](#_Toc193521739)

[Datu izgūšana no datu bāzes (SELECT) 10](#_Toc193521740)

[Vairākas saites tabulai pašai ar sevi 10](#_Toc193521741)

[Tabulu iekšēja saistīšana 11](#_Toc193521742)

[LIKE operators 11](#_Toc193521743)

[Grupēšana un kārtošana 12](#_Toc193521744)

[Ierakstu grupēšanas vaicājumi (GROUP BY) 12](#_Toc193521745)

[Ierakstu kārtošanas vaicājumi (ORDER BY) 13](#_Toc193521746)

[Grupu nosacījums (HAVING) 14](#_Toc193521747)

[Apakšvaicājumi 15](#_Toc193521748)

[Apakšvaicājums SELECT rindā 15](#_Toc193521749)

[Apakšvaicājums WHERE rindā 15](#_Toc193521750)

[Apakšvaicājums FROM rindā 16](#_Toc193521751)

[SQL loģiskie operatori 17](#_Toc193521752)

[Vaicājums ar EXISTS/NOT EXISTS 17](#_Toc193521753)

[Vaicājums ar SOME 18](#_Toc193521754)

[Vaicājums ar ALL 19](#_Toc193521755)

[Tabulu apvienošana (UNION) 20](#_Toc193521756)

[Šķērstabulas vaicājums (*Crosstab Query*) 21](#_Toc193521757)

[SECINĀJUMI 22](#_Toc193521758)

[IZMANTOTAS LITERATŪRAS SARAKSTS 23](#_Toc193521759)

# Uzdevuma nostādne

1. Par pamatu otram darbam tiek ņemta pirmajā darbā izveidota datu bāze.
2. Izveidot vismaz 20 piemērus SQL vaicājumiem ar vienu vai vairākām tabulām. Vaicājumu skaits ar vienu tabulu nedrīkst pārsniegt 5. Vaicājumu skaits ar vairāk ka divām tabulām – vismaz 2.
3. Obligāti piemēri vaicājumiem ar :

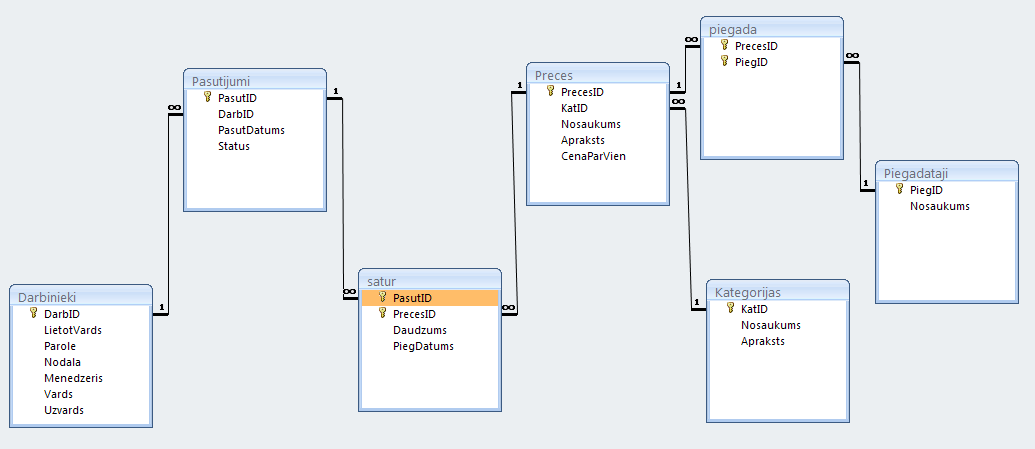
* Vienu un vairāku kritēriju noradīšanu,
* Grupēšanu,
* *HAVING*,
* Apakšvaicājumiem vismaz 3 dažādas rindas (*SELECT, FROM, WHERE, HAVING*),
* Salīdzināšanu ar kopu (*EXISTS*, *ANY*,....),
* Apvienošanu,
* Rezultātu transformēšanu.

1. Datu manipulācijas un definēšanas vaicājumu piemēri,
2. Aprakstā iekļaut vaicājuma izpildei izmantotas tabulas, vaicājumu piemērus ar iegūtajiem rezultātiem ( vēlamā norāde uz izmantotu MS Access vaicājumu veidu ),
3. Pievienot darba elektrisko versiju.

# Tabulu ievaddatu īss raksturojums

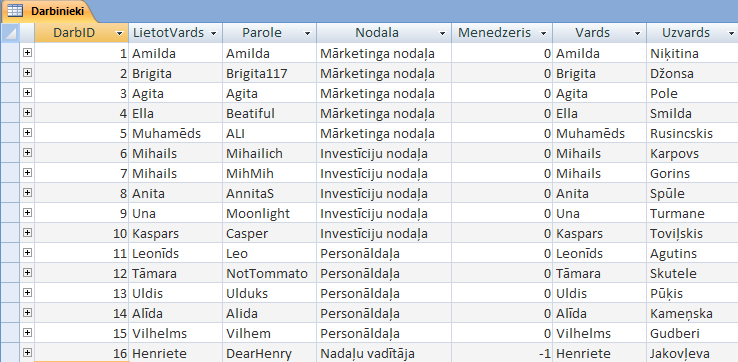
Iepriekšēja uzdevuma tika izstrādāta „Kancelejas preces” datu bāze, kura, savukārt, ir pamats kopīgi vienotai centralizētai kancelejas preču pasūtījumu informācijas sistēmai, kuru man bija jāizstrādā kādas kompānijas vajadzībām. Bez minētas sistēmas katra no kompānijas nodaļām patstāvīgi veica kancelejas preču pasūtījumus, kas apgrūtināja kopējo izmaksu noskaidrošanu un šo izmaksu ierobežošanu, ka arī izdevīgāko piegādātāju noteikšanu.

Datu bāzes projektēšanas gaitā tika izveidotas 7 tabulas: darbinieki, pasūtījumi, satur, preces, kategorijas, piegādā un piegādātāji - ar saitēm starp tam (skat. att. 1).



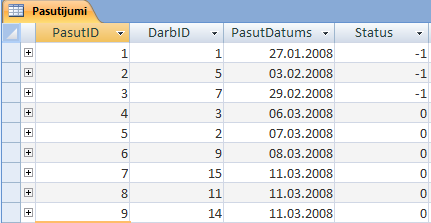
1. att. Datu bāzes tabulas un saites starp tām.

Tabula *Darbinieki* satur informāciju par kompānijas personālu. Šī informācija ir nepieciešama, lai atlasītu informāciju par darbinieku, kuram pieder kāds pasūtījums, cik pasūtījumi kopā ir veikti kādā konkrētā nodaļā u.tml. (skat. att. 2).



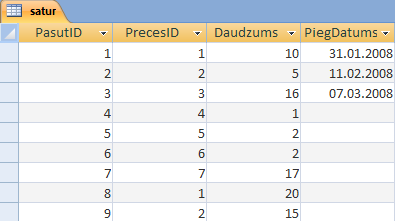
1. att. Tabula *Darbinieki*.

Tabula *Pasūtījumi* satur informāciju par visu darbinieku veiktajiem pasūtījumiem neatkarīgi no nodaļas. Šeit tiek fiksēts pasūtījuma datums, pasūtītājs un pasūtījuma status – saņemts vai nē (skat. 3 att.). Šī tabula tiks izmantota, lai atlasītu nepieciešamo informāciju par pasūtījumiem.



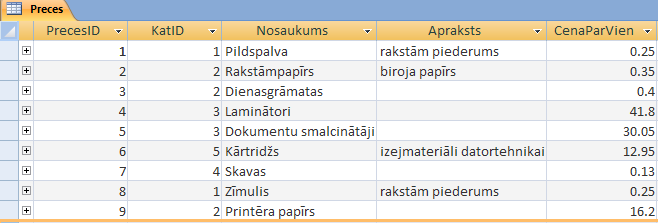
1. att. Tabula *Pasūtījumi*.

Tabula *satur* iekļauj sevī informāciju par pasūtījumiem un šajos pasūtījumos veikto preci. Šī tabulā tiks izmantota, lai izsekotu preces kopējo daudzumu vairākos pasūtījumos, ja tāda prece tiek pasūtīta vairakkārt, kādas preces ir piegādātas u.tml. (skat. att. 3).



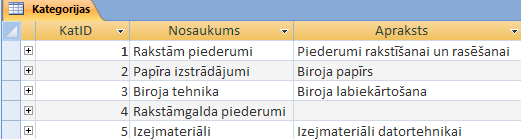
1. att. Tabula *satur*.

Tabula *Preces* satur informāciju par precēm. Šī tabulā tiks lietota, lai izgūtu informāciju par pasūtījumos pasūtīto preci, izmaksām, kategorijām, kuras ietilpst šīs preces u.c. (skat. att. 5).



1. att. Tabula *Preces*.

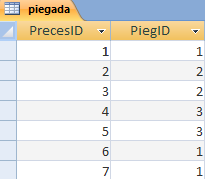
Tabula *Kategorijas* satur informāciju par kategorijām, kas būs nepieciešama, lai veiktu preču klasificēšanu pēc to veidiem (skat. att. 6).

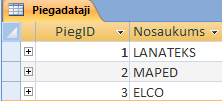


1. att. Tabula *Kategorijas*.

Tabula *Piegādātāji* satur informāciju par piegādātājiem, kuri piegādā dotajai kompānijai darbinieku pasūtītas preces (skat. att. 7). Šie dati tiks izmantoti, lai atlasītu kādas preces un to daudzums ir vēl jāpiegādā kādam no piegādātajiem u.tml.

Tabula *piegādā* iekļauj sevī informāciju par precēm, kuras ir jāpiegādā kādam piegādātājam (skat. att. 8).

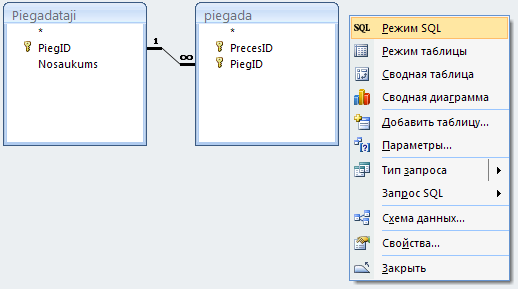
****

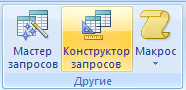


1. att. Tabula *Piegādātāji*. 8. att. Tabula *piegādā*.

Turpmākas sadaļas apskatīsim vairākus SQL vaicājumus, kuri ļauj atlasīt nepieciešamu informāciju no visam tabulām, kas, savukārt ir ļoti lietderīgi atskaišu veidošanai, ka arī datu manipulēšanas, transformējamos un apvienojuma vaicājumus.

Lai veiktu vaicājumu konstruēšanu Access 2007 rīkjoslā *INSERT* ir jāizvēlas *Create Queries* komandpoga (skat. att. 9), rezultāta atvērsies *Design View* skats (skat. att. 10), kurā ir jāizvēlas *SQL View* režīms (skat. att. 10).





1. att. Komandpoga *Create Queries*. 10. att. Skats *Design View*.

# Datu manipulēšanas vaicājumi

Datu manipulēšanas vaicājumi paredz datu ievadi, koriģēšanu un dzēšanu no tabulām. Pie manipulēšanas vaicājumiem tiek pieskaitīti šāda tipa vaicājumi:

* INSERT- paredz jauno rakstu ievietošanu datu bāzes esošās tabulas;
* UPDATE - paredz datu atjaunošanu un labošanu esošās tabulas;
* DELETE- paredz datu dzēšanu tabulas*.*

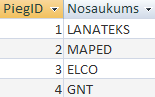
## Jauna ieraksta ievietošana esošā tabulā (*INSERT*)

Lai ievietotu jaunu ierakstu esošā tabulā ir jāizmanto INSERT vaicājums, norādot attiecīgo tabulu un datus, kurus mēs vēlamies ievietot (INSERT VALUES). Šeit gribētos piebilst, ka esoša tabulā var tikt ievietoti noklusēti dati (INSERT DEFAULT VALUES) vai arī dati, kuri tiek atlasīti SELECT vaicājuma izpildes rezultāta (INSERT SELECT).

**Vaicājums.** Ievietot tabulā *Piegādātāji* jauno piegādātāju GNT.

INSERT INTO Piegadataji (PiegID, Nosaukums)

VALUES ('4', "GNT");



1. att. *INSERT* vaicājuma izpildes rezultāts.

Kā redzams tabulā Piegādātāji tika pievienots jauns ieraksts (11. att.). Lai ieraksts tiktu pievienots tieši šai tabulai, vaicājumā aiz INTO komandas jānorāda šīs tabulas nosaukums un iekavās tabulas atribūtus, kuros tiks ievietotas jaunas ierakstu vērtības, kuras, savukārt, tika noradītas aiz komandas VALUES iekavās.

## Lauku vērtības maiņa (*UPDATE*)

Bieži vien ir nepieciešams veikt kādas izmaiņas, kuras parasti ir saistītas ar lauku vērtību maiņu. Šīm nolūkam paredzēta UPDATE komanda, kuras sintakse ir sekojoša:

UPDATE tabulas nosaukums [AS sinonīms]

SET atribūta(kolonnas) nosaukums = izteiksme

[WHERE nosacījums];

**Vaicājums.** Atjaunot tabulu *Kategorijas*, kategorijai „rakstāmgalda piederumi” pievienot aprakstu „Komforta paaugstināšanai”.

UPDATE Kategorijas

SET Kategorijas.Apraksts = "Komforta paaugstināšanai"

WHERE Kategorijas.KatID=4;

Vaicājuma izpildes rezultāta minētajai kategorijai (unikālais identifikators ir 4) tika pievienots apraksts „Komforta paaugstināšanai”. Jā šīm vaicājumiem netiktu uzradīts nosacījums, tad visām dotas tabulas kategorijām tiktu pievienots šāds apraksts.

# Datu izgūšana no datu bāzes (SELECT)

Lai veiktu datu atlasīšanu no datu bāzes tabulām, vaicājumam ir jāsatur komandu SELECT, tāda veida ziņojot DBVS, ka tai ir jāatrod un jāparāda noteiktu informāciju. Vienkārša SELECT vaicājuma sintakse ir šāda:

SELECT {\*| [DISTINCT] izteiksmju saraksts}

FROM tabulu saraksts

[WHERE nosacījums ]

[GROUP BY izteiksmju saraksts grupēšanai]

[HAVING grupas nosacījums]

[ORDER BY izteiksmju saraksts kārtošanu]

Zemākas sadaļas tiks aplūkoti vairāki vaicājuma piemēri, kuros tabula tiks sasaistīta gan ar paša ar sevi, gan arī ar vairākām citām tabulām, izmantojot iekšējo saistīšanu (JOIN) un saistīšanu pēc vienādības (D.ID = P.ID). Tiks apskatīta grupēšana, kārtošana, SQL loģiskie operatori un apakšvaicājumi.

## Vairākas saites tabulai pašai ar sevi

**Vaicājums.** Izvadīt darbinieku sarakstu pa nodaļām.

SELECT DISTINCT a.Nodala AS Nodala, a.Vards AS Darbinieki

FROM Darbinieki AS A, Darbinieki AS B

WHERE a.Nodala=b.Nodala And a.Vards<>b.Vards;



1. att. Kompānijas personāla saraksts.

Vaicājumā viena un tā pati tabula ir apzīmēta ar dažādiem apzīmējumiem un izveidota saite starp vienu to pašu tabula, izmantojot vienādības saistīšanu, rezultātā kad tiek atlasīts viens darbinieks, tas katru reizi tiek salīdzināts ar sevi pašu un vairākiem citiem darbiniekiem. Darbinieku dublēšanos novērš DISTINCT atslēgvārds.

## Tabulu iekšēja saistīšana

Šāda saistīšana notiek aiz komandas FROM norādot JOIN (INNER, LEFT OUTER, RIGHT OUTER) atslēgvārdu. Šeit ir ļoti svarīga secība pēc kuras tabulas tiek savienotas un obligāti jālieto iekavas. Dotus savienojumus var precizēt, izmantojot ON vai USING operatorus. Sintakse šādam savienojumam izskatās sekojoši:

tabula [INNER | {FULL | LEFT | RIGHT } [OUTER]] JOIN tabula {ON vienādības izteiksme | USING (atribūtu saraksts)}

**Vaicājums.** Pāradīt pasūtīto preču sarakstu, daudzumu, pasūtītājus un pasūtīšanas datumu.

SELECT Preces.Nosaukums AS Preces, Darbinieki.Vards AS Vārds, Darbinieki.Uzvards AS uzvārds, darbinieki.Nodala AS Nodaļa,

satur.Daudzums AS Daudzums, Pasutijumi.PasutDatums AS Pasūtīšanas\_datums

FROM Darbinieki

INNER JOIN (Pasutijumi INNER JOIN (Preces INNER JOIN satur ON Preces.PrecesID=satur.PrecesID)ON Pasutijumi.PasutID=satur.PasutID) ON Darbinieki.DarbID=Pasutijumi.DarbID

ORDER BY Preces.Nosaukums;



1. att. Pasūtīto preču saraksts.

Vispirms tiek savienotas tabulas *Preces* un *Satur*, tad jau esošām savienojumam tiek pievienota vēl viena *Pasūtījumi* tabula, veidojot savienojumu pēc *PasutID* primāras atslēgas ar *Satur* tabulu, tad papildus tiek pievienota *Darbinieki* tabula. Rezultāta iegūstam vienu kopīgo tabulu, kurā satur visus sasaistīto tabulu ierakstus.

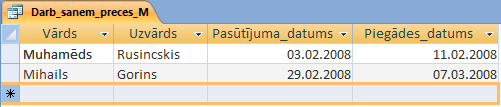
## LIKE operators

**Vaicājums.** Pāradīt nodaļu darbiniekus, kuri ir saņēmuši preces un kuru vārds sākas ar burtu „M”.

SELECT Darbinieki.Vards AS Vārds, Darbinieki.Uzvards AS Uzvārds, Pasutijumi.PasutDatums AS Pasūtījuma\_datums, satur.PiegDatums AS Piegādes\_datums

FROM (Darbinieki INNER JOIN Pasutijumi ON Darbinieki.DarbID=Pasutijumi.DarbID) INNER JOIN satur ON Pasutijumi.PasutID=satur.PasutID

WHERE (Pasutijumi.Status=True) And (Darbinieki.Menedzeris=0) And (Darbinieki.Vards Like "M" & "\*");



1. att. Darbinieku saraksts, kuru vārds sākas uz „M” burta un kuri ir saņēmuši pasūtījumus.

Lai atrisinātu šādu uzdevumu tika izmantots operators LIKE, kurš nosaka ka tiks atlasīti tikai tie darbinieki kuru vārds sākas ar burtu „M”.

## Grupēšana un kārtošana

Bieži vien veidojot atskaites ir nepieciešamība atlasīt informācija tieši par konkrētiem datiem, nevis par visu tabulu, veikt šo datu kārtošanu un uzdod papildus nosacījumus grupām. Šādas operācijas nodrošina trīs operatori HAVING, GROUP BY un ORDER BY, kuri tiks apskatīti zemāk šajā nodaļā.

### Ierakstu grupēšanas vaicājumi (GROUP BY)

Operators GROUP BY ļauj grupēt tabulu ierakstus un pielietot agregāt funkcijas, piemēram, COUNT (\*), izveidotajām grupām. Operatora sintakse izskatās šādi :

GROUP BY izteiksme [, izteiksme, ......]

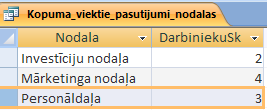
**Vaicājums.** Noteikt cik kopumā darbinieki nodaļas ir veikuši pasūtījumus (bez nod. vadītāja).

SELECT Darbinieki.Nodala, Count(Darbinieki.DarbID) AS DarbiniekuSk

FROM Darbinieki INNER JOIN Pasutijumi ON Darbinieki.DarbID=Pasutijumi.DarbID

WHERE (((Darbinieki.Menedzeris)=0))

GROUP BY Darbinieki.Nodala;



1. att. Veikto pasūtījumu daudzums nodaļas.

Vispirms vaicājums atlasa nodaļas un pasūtījumus. Tad izpilda nodaļu grupēšanu un veic pasūtījumu saskaitīšanu katrai nodaļai. Dotajā piemērā grupēšana notiek tikai pēc nodaļu vērtībām.

**Vaicājums.** Noteiktcik pasūtījumus ir jāpiegādā katram piegādātājiem, paradīt preču kopējo daudzumu šajos pasūtījumos.

SELECT DISTINCT Piegadataji.Nosaukums, COUNT(satur.PasutID) AS Pasut\_skaits, SUM(satur.Daudzums) AS Preces\_daudzums

FROM Piegadataji

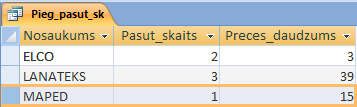
INNER JOIN ((Preces INNER JOIN satur ON Preces.PrecesID=satur.PrecesID)

INNER JOIN piegada ON Preces.PrecesID=piegada.PrecesID)

ON Piegadataji.PiegID=piegada.PiegID

WHERE satur.PiegDatums IS NULL

GROUP BY Piegadataji.Nosaukums;



1. att. Pasūtījumu skaits, kurus ir jāpiegādā piegādātājiem.

Šinī piemēra katrām piegādātājam bez pasūtījumu saskaitīšanas tiek arī izpildīta preču summēšana.

### Ierakstu kārtošanas vaicājumi (ORDER BY)

Pēc vaicājuma izpildes tabulas ieraksti netiek sakārtoti dilstoša vai augoša secība. Lai veiktu ierakstu kārtošanu var izmantot ORDER BY operatoru. Operatora sintakse ir šāda:

ORDER BY kārtošanas specifikācija [, kārtošanas specifikācija, ......]

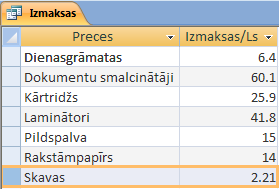
**Vaicājums.** Noteikt izmaksas katrai pasūtītai precei.

SELECT DISTINCT Preces.Nosaukums AS Preces, (Sum(satur.Daudzums)\*Sum(Preces.CenaParVien)) AS [Izmaksas/Ls]

FROM Preces INNER JOIN satur ON Preces.PrecesID=satur.PrecesID

GROUP BY Preces.Nosaukums

ORDER BY Preces.Nosaukums;



1. att. Kopējas izmaksas katrai precei.

Vaicājums aprēķina preču izmaksas – reizinot pasūtītas preces daudzumu ar preces cenu, tad sakārto preču grupas datus augoša secība.

### Grupu nosacījums (HAVING)

Operators HAVING tiek izmantots kopa ar GROUP BY operatoru un ļauj formulēt nosacījumus grupām papildus informācijas atlasīšanai.

**Vaicājums.** Atrasts kategorijas, kuras ietilpst vairāk kā viena prece.

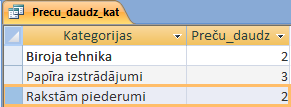
SELECT Kategorijas.Nosaukums AS Kategorijas, COUNT(Preces.PrecesID) AS Preču\_daudz

FROM Kategorijas, Preces

WHERE Kategorijas.KatID=Preces.KatID

GROUP BY Kategorijas.Nosaukums

HAVING COUNT(Preces.PrecesID)>1;



1. att. Kategoriju saraksts, kuros ietilpst vairāk ka viena prece.

Lai izpildīto šo uzdevumu, atlasām kategorijas un preces no tabulām *Kategorijas* un *Preces*. Nākamais solis ir sagrupēt kategorijas un pakārtot preču kopskaitu atbilstošai kategorijai. Tagad kad mēs esam ieguvuši informāciju par katru kategorijas grupu, varam definēt nosacījumu (HAVING), lai atlasītu tikai tas grupas, kuras preču skaits ir lielāks par viens.

**Vaicājums.** Atrasts kopējo preču daudzumu visos pasūtījumos, paradīt tikai tas preces, kuras tika pasūtītas vairakkārt (piemēram, pildspalvas tika pasūtītas gan personāldaļa, gan Marketinga nodaļā).

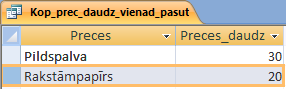
SELECT Preces.Nosaukums AS Preces, Sum(satur.Daudzums) AS Preces\_daudz

FROM satur, Preces

WHERE satur.PrecesID=Preces.PrecesID

GROUP BY Preces.Nosaukums

HAVING COUNT(satur.PrecesID)>1;



1. att. Daudzuma noteikšana precēm.

Piemērā dotais vaicājums ir līdzīgs augstāk apskatītam piemēram, vienīgi šoreiz ir iegūta preču kopsummā vienādos pasūtījumos.

**Vaicājums.** Atrasts datumus, kuros ir veikts nemazāk ka viens pasūtījums.

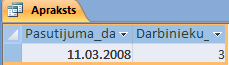
SELECT DISTINCT Pasutijumi.PasutDatums AS Pasutijuma\_datums,

COUNT(Pasutijumi.DarbID) AS Darbinieku\_pasut

FROM Pasutijumi

GROUP BY Pasutijumi.PasutDatums

HAVING COUNT(Pasutijumi.DarbID)<>1;



1. att. Datums, kura ir veikts nemazāk ka viens pasūtījums.

## Apakšvaicājumi

SQL paredz viena vaicājuma ievietošanu otrā, tāda veida viens vaicājums var izmantot datus, kurus izpilda otrs vaicājums. Parasti šāds apakšvaicājums tiek ievietots aiz WHERE, SELECT, FROM, HAVING, IN, ka arī EXISTS, ANY, SOME, ALL loģiskiem operatoriem.

### Apakšvaicājums SELECT rindā

Šāda apakšvaicājuma atgrieztais rezultāts tiek ievietots rezultējoša tabulā, kuru formē vaicājums. Gribētos piebilst, ka dotais apakšvaicājums var atgriezt tikai vienu vērtību, piemēram, ja likt tam atrast maksimālo vai minimālo vērtību, tad atgrieztais rezultāts būs kļūda.

**Vaicājums.** Parādīt informāciju par nodaļu vadītāju.

SELECT Darbinieki.Vards AS Vārds, Darbinieki.Uzvards AS Uzvārds,

Darbinieki.LietotVards AS LOGIN,

(SELECT D.Nodala

FROM Darbinieki D

WHERE D.DarbID = Darbinieki.DarbID )AS Nodaļa

FROM Darbinieki

WHERE Darbinieki.Menedzeris=True;



1. att. Informācija par nodaļas vadītāju.

Atlasām darbinieku vārdus, uzvārdus un lietotāj vārdus no tabulas *Darbinieki*, tad liekam apakšvaicājumam atgriezt informāciju par nodaļām, apzīmējot to pašu tabulu ar citu apzīmējumu. Tad WHERE nosacījumā noradām, ka ir nepieciešams atlasīt informāciju tikai par nodaļas vadītāju.

### Apakšvaicājums WHERE rindā

Apakšvaicājums WHERE rindā ( arī HAVING ) tiek iestarpināts ar IN, EXISTS, ALL, SOME, ANY vai arī ar operāciju ( = , <>, <, <=, >, >= ) operatoru palīdzību.

**Vaicājums.** Izvadīt informāciju par darbiniekiem, kuri gaida preces.

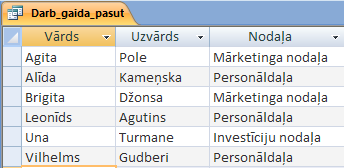
SELECT Darbinieki.Vards AS Vārds, Darbinieki.Uzvards AS Uzvārds, Darbinieki.Nodala AS Nodaļa

FROM Darbinieki

WHERE (((Darbinieki.DarbID)=(SELECT Pasutijumi.DarbID FROM Pasutijumi

WHERE Darbinieki.DarbID = Pasutijumi.DarbID AND Pasutijumi.Status = False)))

ORDER BY Darbinieki.Vards;



1. att. Atskaite par darbiniekiem, kuri gaida preces.

Apakšvaicājums no tabulas *Pasūtījumi* atlasa tikai tos darbiniekus, kuri nav saņēmuši preces. Savukārt, pats vaicājums no tabulas *Darbinieki* atlasa tikai informāciju par tiem darbiniekiem, kurus WHERE nosacījumā ir atgriezis apakšvaicājums.

### Apakšvaicājums FROM rindā

**Vaicājums.** Izvadīt pilno informāciju par pasūtījumiem – preces, darbinieki, pasūtīšanas datums, piegādātāji, kurus ir vēl jāpiegādā .

SELECT Preces.Nosaukums AS Prece, Satur.Daudzums, Darbinieki.Vards AS Vārds, Darbinieki.Uzvards AS Uzvārds, Piegadataji.Nosaukums AS Piegādātāji, Pasutijumi.PasutDatums AS Pasūtīšanas\_datums

FROM (SELECT \*

FROM Piegadataji, piegada, Preces, satur, Pasutijumi,Darbinieki

WHERE Piegadataji.PiegID=piegada.PiegID And

piegada.PrecesID=Preces.PrecesID And

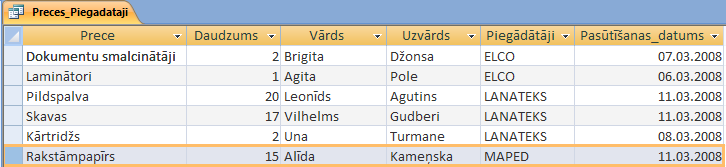
Preces.PrecesID=satur.PrecesID And

satur.PasutID=Pasutijumi.PasutID And

Pasutijumi.DarbID=Darbinieki.DarbID) AS [%$##@\_Alias]

WHERE (((satur.PiegDatums) Is Null))

ORDER BY Piegadataji.Nosaukums;



1. att. Atskaite par darbiniekiem, kuri gaida preces.

Apakšvaicājums FROM rinda veic vairāku tabulu saistīšanu, ka rezultātā tiks atlasīta visa informācija no tekošam tabulām. No dotas informācijas galvenais vaicājums izgūs tikai to informāciju, kuru mēs vēlāmies redzēt šajā atskaitē.

## SQL loģiskie operatori

### Vaicājums ar EXISTS/NOT EXISTS

SQL vaicājuma valodā operators EXISTS pārbauda vai viņam adresēta argumenta vērtība eksistē. Jā tas tā ir, tad EXISTS atgriež īstenības (*true*) vērtību, citādi melu (*false*) vērtību. Operators NOT EXISTS darbojas pretēji EXISTS loģikai. EXISTS operators var tikt izmantots kopa ar WHERE un HAVING.

**Vaicājums.** Izvadīt darbinieku sarakstu nodaļas, kuri nav veikuši pasūtījumus (neskaitot nodaļu vadītāju).

SELECT Darbinieki.Nodala, Darbinieki.Vards AS Vārds, Darbinieki.Uzvards AS Uzvārds

FROM Darbinieki

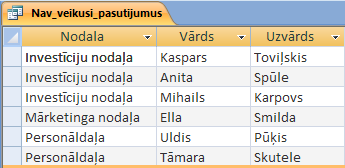
WHERE (Darbinieki.Menedzeris = 0) AND

(NOT EXISTS (SELECT \*

FROM Pasutijumi

WHERE Pasutijumi.DarbID = Darbinieki.DarbID))

ORDER BY Darbinieki.Nodala;



1. att. Darbinieki, kuri nav veikuši pasūtījumus.

Apakšvaicājums atlasa darbiniekus, kuri ir veikuši pasūtījumus, šo atlasīto informāciju pārbauda operators NOT EXISTS un atgriež galvenajām vaicājumam melu vērtību, tādejādi norādot, ka ir jāizgūst tieši tos darbiniekus, kuri nav veikuši pasūtījumus.

**Vaicājums.** Atrast kategoriju, kurā tajā ietilpstošo preču kopsummā ir lielāka par 20 Ls.

SELECT DISTINCT K.Nosaukums AS Kategorija, SUM(P.CenaParVien) AS Precu\_cenas\_kopsumma

FROM Preces AS P, Kategorijas AS K

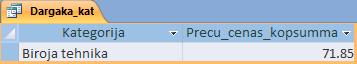
WHERE K.KatID = P.KatID

GROUP BY K.KatID, K.Nosaukums

HAVING EXISTS ( SELECT \*

FROM Preces P2

WHERE K.KatID = P2.KatID AND P2.CenaParVien > 20.0 );



1. att. Dārgāka kategorija .

Tabulā *Preces* eksistē preces, kuru cena ir lielāka par 20 Ls, tāpēc operators EXISTS pieņem īstenības vērtības. Rezultējoša tabula saturēs tikai tos ierakstus, kuru cenas ir lielākas par 20 Ls.

### Vaicājums ar SOME

Loģiskais operators SOME un ANY veic kreisā operanda vērtības salīdzināšanu ar laba operanda vairāku vērtību salīdzināšanu. Šāda salīdzināšana notiek ar operāciju (=, <>, >, u.c.) starpību. Jā salīdzināmais operands ir patiess kaut vienai no vairākām vērtībām, tad rezultāts būs patiesība, citādi meli.

**Vaicājums.** Atrast preces, kuru cenas ir mazākas kaut vai par vienu no precēm, kuras cena ir 12.95 Ls.

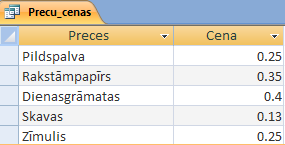
SELECT P.Nosaukums AS Preces, P.CenaParVien AS Cena

FROM Preces AS P

WHERE P.CenaParVien < SOME (SELECT P2.CenaParVien

FROM Preces P2

WHERE P2.CenaParVien = 12.95);



1. att. Preces, kuru cena nepārsniedz 12.95 Ls .

Tabulā *Preces* eksistē vairākas preces, kuru cenas ir mazākas par 12.95 Ls un eksistē tāda prece, kuras cena ir 12.95 Ls, tāpēc operators SOME pieņem īstenības vērtības. Rezultātā galvenais vaicājums var izvadīt tas preces, kuru cena ir mazākas par norādīto uzdevumā cenu.

### Vaicājums ar ALL

Atšķirība no SOME un ANY loģiskiem operatoriem ALL pieņem patiesības vērtību tikai tad, kad laba operanda vairākas vērtības atbilst kreisa operanda vērtībai pamatojoties uz salīdzināšanas operāciju.

**Vaicājums.** Atrast kategoriju, kurā ietilpst preces, kurās nodaļu darbinieki visvairāk pasūta.

SELECT Kategorijas.Nosaukums

FROM Kategorijas

WHERE Kategorijas.KatID IN

(

SELECT Preces.KatID FROM Preces, satur

WHERE Preces.PrecesID = satur.PrecesID

GROUP BY Preces.KatID

HAVING COUNT (Preces.PrecesID) > ALL

(

SELECT COUNT (satur.PrecesID)

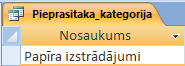
FROM satur, Preces

WHERE Preces.PrecesID = satur.PrecesID

GROUP BY satur.PrecesID

)

);



1. att. Pieprasītāka kategorija.

Tabulā *Kategorijas* tiek atlasītas tikai tas kategorijas, kuras izpildās operatora IN apakšvaicājumā ietvaros. Savukārt, šajā apakšvaicājumā kategorijas tiek sagrupētas pa grupām un tam tiek pakārtots preču skaits, kuras ietilpst šajās kategorijas. Turpmāk izmantojot ALL loģisko operatoru tiek salīdzināts preču skaits kategorijas ar preču skaitu, kuras ir pasūtītas, loģiski iegūstam „papīra izstrādājumi” kategoriju, jo kategorijas asortiments ir lielāks par citu preču asortimentu un tas tiek vairāk pasūtītas. Tad IN apakšvaicājumā izpildās tikai viena kategorija, kuru turpmāk izvada galvenais vaicājums.

# Tabulu apvienošana (UNION)

Agrāk apskatītie SQL vaicājuma varianti (SELECT) atbilst tikai dažām relāciju algebras operācijām – projekcijai, selekcijai un savienošanai. Bet SQL vaicājuma valoda ietver arī apvienošanas (UNION), dažādības (EXCEPT) u.c. operācijas.

Operators UNION apvieno divu tabulu ierakstus vienā tabulā (tabulām jāsatur vienādus ierakstus), dzēšot dublikātus. Ir iespējami daži izņēmumi, kad ir iespējams veikt vienas tabulas apvienošanu citā, apskatīsim tādu piemēru:

**Vaicājums.** Secinotpēc piegādātāju atgādātājam mantām, katram piegādātājam ir jāpievieno klāt apraksts.

SELECT Piegadataji.Nosaukums AS Piegādātājs, " Iekārtu pieg. " AS Apraksts

FROM Piegadataji INNER JOIN piegada ON Piegadataji.PiegID = piegada.PiegID

WHERE piegada.PrecesID = 4 AND 5

UNION

SELECT Piegadataji.Nosaukums, " Mantu pieg. "

FROM Piegadataji INNER JOIN piegada ON Piegadataji.PiegID = piegada.PiegID

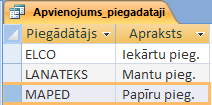
WHERE piegada.PrecesID = 1 AND 6 AND 7

UNION

SELECT Piegadataji.Nosaukums, " Papīru pieg. "

FROM Piegadataji INNER JOIN piegada ON Piegadataji.PiegID = piegada.PiegID

WHERE piegada.PrecesID = 2 AND 3;



1. Att. Vaicājuma atlasīta informācija.

Vaicājums atlasa piegādātājus balstoties uz nosacījumu kādas preces piegādā piegādātāji, t.i., piemēram jā piegādātājs piegādā dienasgrāmatas vai rakstāmpapīru, tam tiek pievienots apraksts „Papīru piegādātājs”, tad jau esošai tabulai tiks pievienots cits piegādātājs ar savu unikālo aprakstu u.tml.

# Šķērstabulas vaicājums (*Crosstab Query*)

Dotais vaicājums ļauj atlasīt vairākus datus no vienas tabulas tāda veida, ka dati tiek pakārtoti viens otrām šķērstabulas veidā, kas ir ļoti lietderīgi informācijas detalizētai apskatīšanai.

**Vaicājums.** Izvadīt darbinieku sarakstu pa nodaļām ( kāds darbinieks pieder kādai nodaļai).

TRANSFORM COUNT(Nodala)

SELECT Uzvards AS Uzvārds

FROM Darbinieki

WHERE Menedzeris=False

GROUP BY Uzvards

PIVOT Nodala;



1. Att. Transformējamais vaicājums par darbiniekiem.

Vaicājums vispirms atlasa darbinieku uzvārdus, tad atlasa informācija par nodaļām un pakārto tas pretī attiecīgajam darbiniekam. Šādu informācijas pakārtošanu izpilda *Transform* un *PIVOT* komandas.

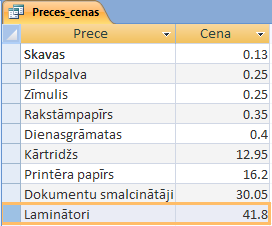
**Vaicājums.** Paradīt preču cenas.

TRANSFORM CenaParVien

SELECT Nosaukums AS Prece

FROM Preces

GROUP BY CenaParVien, Nosaukums

 PIVOT "Cena";

1. Att. Transformējamais vaicājums par precēm.

# SECINĀJUMI

Tika izstrādāts laboratorijas darbs, kurā primārais uzdevums bija iepazīties un apgūt SQL vaicājuma valodu. Uzdevums bija interesants, tanī bija nepieciešamība patstāvīgi veikt uzdevuma formulēšanu un to atrisināšanu, pielietojot gan tradicionālas, gan netradicionālas pieejas metodes, t.i., piemēram, lai veiktu darbinieku atlasīšanu attiecīgas nodaļas varēja izmantot vienkāršu vaicājumu, bet izmantojot tabulas saistīšanu ar sevi dotais uzdevums izrādījās interesantāks.

Par labumu uzskatu, ka dotais darbs ļāva paplašināt savas zināšanas SQL vaicājuma valodā, piemēram, uzzināju par šķērstabulu vaicājumiem, kas pats par sevi ir ļoti ērts veids, ka panākt detalizēto datu atskaites izveidošanu u.tml., un iepazinos ar vairāku izstrādātāju sintakses īpatnībām, t.i., dažas SQL realizācijas simbols (;) tiek uzskatīts par vaicājuma beigu pazīmi un tas ir obligāts, piemēram, kompānijas Oracle SQL \* PLUS versija nevar izpildīt vaicājumu kamēr netiek ierakstīts šāds simbols, savukārt, MS SQL šāds simbols nav obligāts.

Darba gaitā mēģināju izpildīt gan visas prasības, gan izveidot vaicājumus, kuri būtu lietderīgi kādas kompānijas vajadzībām, kura lietotu esošo datu bāzi.

# IZMANTOTAS LITERATŪRAS SARAKSTS

1. Viktorija Vinogradova. Lekciju konspekti studiju priekšmetā „Datu bāzes vadības sistēma” – Rīga, RTU, 2007.
2. „Язык запросов SQL учебный курс”, Филипп Андон, Валерий Резничко, „Питер”, 2006.