Rīgas Tehniska universitāte

Datorzinātnes un Informācijas tehnoloģijas fakultāte

Otrais praktiskais darbs mācību priekšmetā

„Lielas datu bāzes”

Relāciju objektu datu bāzes struktūra un objektu SQL vaicājumi

Izstrādāja: Sergejs Terentjevs

3. kurss, 4.grupa

Apl.nr. 061RDB140

Pārbaudīja: Prof. J. Eiduks

2007./2008. māc. g.

# ANOTĀCIJA

Praktiska darbā izpildes gaitā tika izstrādātas vairākas tabulas: iekļauta tabula, tabula ar kolekciju, tabula ar salikto objektu, tabula ar objektu kolonnu un objektu tabula. Tika realizēta saite viens pret daudziem starp bāzes tabulu un objektu eksemplāriem, izveidoti vairāki objektu tipi un kolekcijas.

Darba pārskatā ir apskatīts praktiskā darba izpildes process, apskatīta datu bāzes arhitektūra, relāciju objektu datu bāzes objekti, Objekta SQL paplašinājumi, balstoties uz piemēriem.

Praktiskais darbs tika izpildīts *Oracle 10g Express Edition* datu bāzes vadības sistēmā uz *Genuine Intel* tipā procesorā *Windows XP* operētājsistēmā.

Laboratorijas darba pārskata apjoms: 34 lappuses izklāsta tekstā, tajā ietilpst 13 nodaļas, 12 apakšnodaļas un literatūras saraksts.

# SATURS

[ANOTĀCIJA 2](#_Toc214338198)

[SATURS 3](#_Toc214338199)

[UZDEVUMA NOSTĀDNE 4](#_Toc214338200)

[IEVADS 5](#_Toc214338201)

[OBJEKTU TABULA 6](#_Toc214338202)

[OBJEKTU TABULAS IZVEIDOŠANA 6](#_Toc214338203)

[OBJEKTU TABULAS AIZPILDIŠANA 7](#_Toc214338204)

[OBJEKTU TABULAS DATU UN META IZGŪŠANA 8](#_Toc214338205)

[TABULA AR OBJEKTU KOLONNĀM 11](#_Toc214338206)

[TABULAS IZVEIDOŠANA 11](#_Toc214338207)

[TABULAS AIZPILDIŠANA 12](#_Toc214338208)

[TABULAS DATU UN META IZGŪŠANA 12](#_Toc214338209)

[TABULA AR OBJEKTU KOLONNU (SALIKTAIS OBJEKTS) 15](#_Toc214338210)

[TABULAS AR SALIKTO OBJEKTU IZVEIDOŠANA 15](#_Toc214338211)

[DATU IEVADĪŠANA TABULĀ AR SALIKTO OBJEKTU 16](#_Toc214338212)

[TABULAS AR SALIKTO OBJEKTU DATU UN META DATU IZGŪŠANA 16](#_Toc214338213)

[TABULA AR KOLEKCIJU 19](#_Toc214338214)

[TABULAS AR KOLEKCIJU IZVEIDOŠANA 19](#_Toc214338215)

[DATU IEVADĪŠANA TABULĀ 20](#_Toc214338216)

[TABULAS DATU UN META DATU IZGŪŠANA 20](#_Toc214338217)

[OSQL VAICĀJUMI 24](#_Toc214338218)

[SAITES VIENS PRET DAUDZIEM REALIZĀCIJA 28](#_Toc214338219)

[MANTOŠANAS PIEMĒRS 31](#_Toc214338220)

[SECINĀJUMI 33](#_Toc214338221)

[IZMANTOTA LITERATŪRA 34](#_Toc214338222)

# UZDEVUMA NOSTĀDNE

Izveidot objektu tabulu:

1. Paredzēt datu ievadīšanu objektu tabulā,
2. Izgūt objektu tabulas meta datus,
3. Izvadīt objektu vērtības un objektus.

Izveidot tabulu ar objektu kolonnām:

1. Paredzēt datu ievadīšanu objektu tabulas kolonnās,
2. Izgūt objektu tabulas meta datus,
3. Izvadīt objektu vērtības, objektus un atsevišķās tabulu kolonnu rindas ar objektiem.

Izveidot tabulu ar objektu kolonnu (saliktais objekts):

1. Paredzēt datu ievadīšanu objektu tabulas kolonnās,
2. Izgūt objektu meta datus,
3. Izvadīt objektu vērtības, objektus, objektu kolekciju un atsevišķas tabulu kolonnu rindas ar objektiem.

Izveidot tabulu ar kolekciju:

1. Paredzēt datu ievadīšanu tabula ar kolekciju,
2. Izgūt objekta tipa un tabulas meta datus,
3. Izvadīt objektu vērtības, objektus, objektu kolekciju un atsevišķas tabulu kolonnu rindas ar objektiem.

Realizēt OSQL operatoru vaicājumus (VALUE un TABLE).

Realizēt viens pret daudziem saiti starp objektu tabulām.

Izstrādāt mantošanas piemēru.

# IEVADS

Objektorientēta tehnoloģija ir mūsdienās vispopulārākā metodoloģija programmatūras izstrādei, projektēšanai un analīzei. Vairākas mūsdienu programmēšanas valodas tika plaši papildinātas dotas metodoloģijas atbalstām, tajās tika iestrādātas objektu aprakstīšanas un manipulēšanas iespējas.

Šādas valodas galvenokārt tiek risinātas objektu iegaumēšanas problēmas, lai šos objektus turpmāk varētu pielietot jebkuros citos programmu izsaukumos. Šāda koncepcija (objektu glabāšana un to turpmāka pielietošana) veicināja pētījumu veikšanu objektorientētas pieejas pielietošanai datu bāzēs. Var izšķirt divus šādas pielietošanas izstrādes virzienus: objektorientētas datu bāzes un relāciju objektu datu bāzes. Pirmais virziens paredz tīri objektorientētas datu bāzes un datu bāzes vadības sistēmas izstrādi, bet otrais paplašināta eksistējošas relāciju datu bāzes ar objektorientētas paradigmas iespējām.

Objektorientētais SQL ir otra virziena neatņemama sastāvdaļa, turpmākas nodaļās mēs apskatīsim vairākas objektorientētas pieejas pielietošanas iespējas relāciju objektu datu bāze Oracle.

# OBJEKTU TABULA

## OBJEKTU TABULAS IZVEIDOŠANA

Sākotnēji apskatīsim vienkāršu objektu pielietošanas piemēru relāciju datu bāzē, kurā izveidosim objektu tabulu. Objektu tabula ir tabulas speciālais veids, kuras katrā rinda ir objekts. Parasti šādas tabulas ir tikai viena kolonna ar objektiem.

Šīm nolūkam vispirms ir nepieciešams veikt objekta tipa definēšanu, kas tiek panākts ar komandu *CREATE TYPE <nosaukums> AS OBJECT*. Vienkārša objekta tipa definēšanas sintakse izskatās sekojoši:

CREATE [OR REPLACE] TYPE <TIPA\_NOSAUKUMS> AS | IS OBJECT (

<ATRIBŪTS> <DATU TIPS>,

<MEMBER |STATIC> <PROCEDURE |FUNCTION> <Nosaukums> (<Parametri>)

[RETURN <Datu tips> ]

)

<FINAL | NOT FINAL > <INSTANTIABLE | NOT INSTANTIABLE> , kur

*MEMBER* - parametrs attiecināms uz tipā deklarēto metodi (funkciju vai procedūru). Tā tiks asociēta ar noteiktu objektu ( objekta\_mainīgais.metode() ) ,

*STATIC* – parametrs attiecināms uz tipā deklarēto metodi. Tā tiks asociēta ar objektu tipu ( objekta\_tips.metode() ),

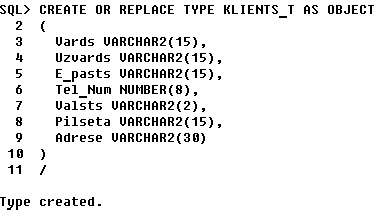
*RETURN* - metodes izpildes atgriežamais rezultāts,

*NOT* *FINAL* | *FINAL* – atslēgvārds norāda vai no dota objekta tipa varēs, ne varēs veidot apakš tipus (mantošanas nosacījums),

*NOT INSTANTIABLE* - atslēgvārds norāda, ka šīm tipam ne eksistē konstruktors, tāda veida neļaujot veidot šī tipa objektus (eksemplārus). Piekļūt pie dota tipa metodēm un argumentiem var vienīgi izmantojot apakš tipu objektus,

*INSTANTIABLE* - atslēgvārds norāda uz tipa objektu veidošanas iespējamību.

Tātad definēsim *Klients\_t* objekta tipu, kurām paredzēsim septiņus atribūtus: klienta vārds, uzvārds, e-pasts, telefona numurs, valsts, pilsēta un adrese. Dotie atribūti nosacīti raksturo klientu kā tādu, līdz ar to mēs varēsim attiecināt doto tipu uz jebkura tipa klientu. Bet jā būs nepieciešams veikt detalizētākas informācijas apkopojumu par klientu, mēs varam izveidot apakšklases jeb apakštipa objektus, izmantojot mantošanas mehānismu, piemēram, izveidot korporatīva klienta objekta tipu, kurš mantos klienta objekta tipa atribūtus un saturēs jaunus atribūtus, kā kompānijas nosaukums, atlaides priekš kompānijas u.tml. Klienta objekta tipa definējums ir paradīts nākama lappuse.



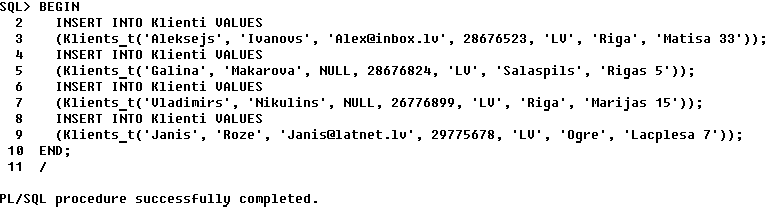
Tagad varam izveidot tabulu Klienti, kurā glabās mūsu objektu atribūtu vērtības. Šīm nolūkam, veicām šādu pierakstu:



Jāpiebilst, kā atslēgvārds *OF* norāda jaunizveidotai tabulai glabāt mūsu objekta tipa eksemplārus, kuriem ir tāds pats nosaukums kā objektu tipam - Klients\_t.

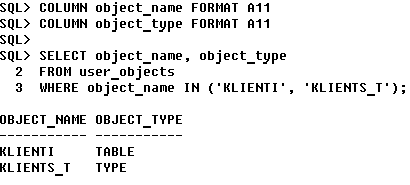
## OBJEKTU TABULAS AIZPILDIŠANA

Piereģistrēsim četrus jaunus klientus, kuriem ir katrām savas īpašības: vārds, uzvārds, telefons, e-pasts u.t.t. (katrs klients ir indivīds - tipa eksemplārs). Lai piereģistrētu klientu ir jānorāda eksemplāra nosaukumu Klients\_t un tikai tad varam aizpildīt dota eksemplāra atribūtu vērtības. Klientu piereģistrēšanu veiksim ar sekojošo PL/SQL procedūras izpildi:

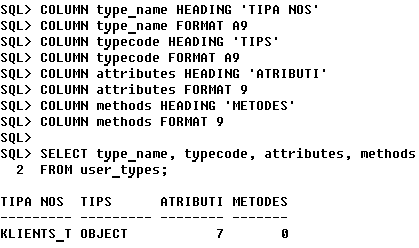


## OBJEKTU TABULAS DATU UN META IZGŪŠANA

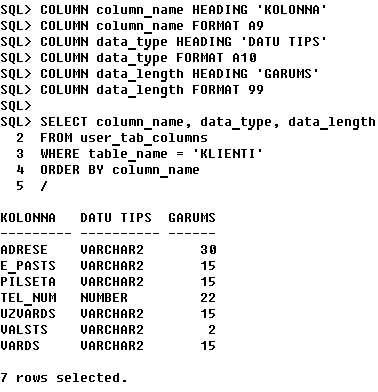
Meta datu vaicājumu izmantošana, ļauj uzskatāmi attēlot informāciju par datu bāzes objektiem (tabulām, skatiem, tipiem u.c.), to struktūrām. Tās ir īpaši lietderīgi administratoriem, piemēram, lai apskatītu manis izveidotus datu bāzes objektus kāda no tabulu telpām u.tml. Tātad vispirms izstrādāsim vaicājumu, kurš ļaus pārliecināties par manis izveidoto shēmu objektu eksistenci. Šīm nolūkam izvadīsim objektu nosaukumus un tipus, izmantojot *user\_objects* lietotāja skata vārdnīcu un atslēgvārdus *object\_name*, *object\_type*. Dotais vaicājums un tā izvades rezultāts izskatās sekojoši:



Spriežot pēc vaicājuma izvades rezultātā manā tabulu telpā eksistē Klienti tabula un Klients\_t tips, kuru mēs apskatīsim sīkāk, izgūstot tā meta datus ar *user\_types* vārdnīcas skata un atslēgvārdu *type\_name* (tipa nosaukums), *typecode* (tips), *attributes* (atribūtu skaits), *methods* (metožu skaits) palīdzību. Vaicājums un tā rezultāts izskatās šādi:

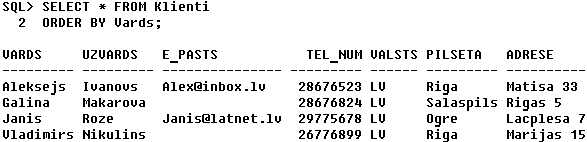


Vaicājums uzskatāmi parada kā objektu tipā Klients\_t tika deklarēti septiņi atribūti un neviena metode. Kā nākamo apskatīsim pašas tabulas Klienti meta datus (*user\_tab\_columns* datu vārdnīcas skats, kur *user* atslēgvārds visiem vārdnīcas skatiem, kur nepieciešams aplūkot informāciju tikai lietotāja shēmas objektiem; atslēgvārdi *column\_name* - kolonnas nosaukums, *data\_type* - kolonnu vērtību datu tipi un *data\_length* - vērtību maksimāli pieļaujamais izmērs). Vaicājums un tā izpildes rezultāts izskatās šādi:



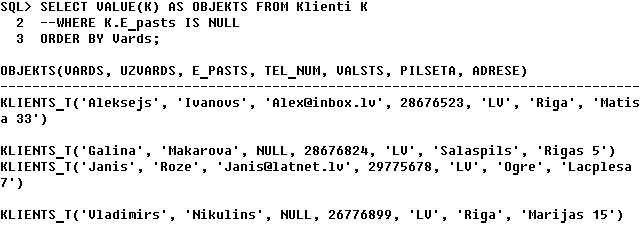
Spriežot pēc vaicājuma izvades, var redzēt ka tabula satur visus septiņus objektu tipā definētos atribūtus, to datu tipus un to izmēri ir atbilstoši (tāda veida tabula reprezentē objektus).

Izgūt datus šādai objektu tabulai relāciju objektu datu bāzes var pēc divām pieejam: konstruēt relāciju datu bāzes SQL datu izgūšanas vaicājumus vai arī izmantot relāciju objektorientēto SQL pieeju datu izgūšanai. Realizēsim abus šādus vaicājumus, kuros izvadīsim objektu tabulas datus. Vispirms izpildīsim relāciju SQL datu izgūšanas vaicājumu, tā izpildes rezultāts izskatās sekojoši:

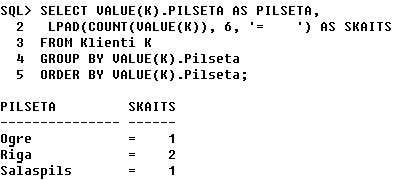


Pēc vaicājuma izpildes mēs esam ieguvuši objekta atribūtu vērtības, kas liek secināt kā Oracle datu bāzes šai objektu tabulai objekta atribūtus vērtības ieglabā tieši tabulas kolonnas. Tātad šinī gadījumā mēs iegūstam vienkāršo tabulas struktūru un lai operētu ar datiem, mēs varam konstruēt parastus relāciju SQL vaicājumus (nav nepieciešama relāciju objektu pieeja).

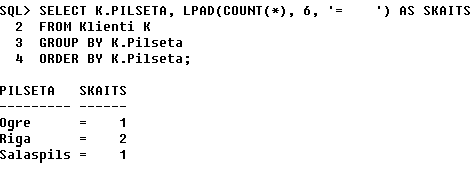
Izpildīsim relāciju OSQL vaicājumu, tāda paša rezultāta iegūšanai, izmantojot funkciju *VALUE*, kura ļauj tieši atsaukties uz objektu eksemplāriem un to atribūtu vērtībām. Vaicājums un tā rezultāts izskatās šādi:



Šoreiz vaicājuma izpildes rezultātā mēs iegūstam objektus un to vērtības, tāda veidā uzskatāmi attēlot klientus kā indivīdus. OSQL vaicājumus var izmatot nepieciešamo datu iegūšanai, piemēram, atsaucoties tieši uz objektu atribūtu vērtībām var saskaitīt kāds ir kopējais reģistrēto klientu skaits pa pilsētām. Dotais vaicājums un tā izpildes rezultāts ir šāds:



Relāciju SQL vaicājums, šāda uzdevuma izpildei, un tā izpildes rezultāts ir šāds:



# TABULA AR OBJEKTU KOLONNĀM

## TABULAS IZVEIDOŠANA

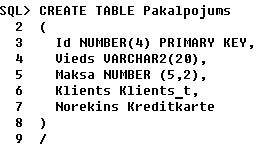
Dotajā sadaļā izveidosim tabulu ar objektu kolonnām. Tabula ar objektu kolonnām ir tabulas speciālais veids, kur viena vai vairākas kolonnās satur objektus, bet pārējas kolonnas satur parastus ierakstus.

Izstrādāsim tabulu ar nosaukumu Pakalpojums, kas saturēs piecas kolonnas, no kurām trīs glabās lietotāja ievadīto informāciju par pakalpojumu veidu, maksu par attiecīgo pakalpojumu un pakalpojuma identifikatoru, bet pārējas divas tabulas kolonnas saturēs objektus - Klients\_t un Kredītkarte - šī informācija būs nepieciešama, lai redzētu kāds klients ir izmantojis kādu no pakalpojumiem un kāda veidā viņš ir norēķinājies par sniegto pakalpojuma veidu u.tml..

Ņemot vēra, ka objektu tips ar nosaukumu Klients\_t mums ir jau izveidots, pašas tabulas veidošanai nepieciešams vienīgi definēt kredītkartes objektu tipu. Minētais tips saturēs tikai divus atribūtus: kredītkartes tips un tās numurs. Kredītkartes objekta tipa definējums izskatās sekojoši:



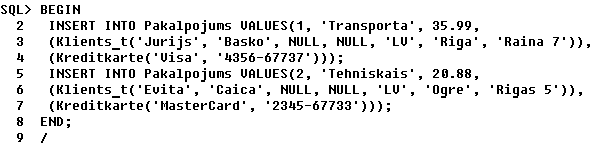
Nākamais solis ir izveidot tabulu Pakalpojums, šīm nolūkam veicām šādu pierakstu:



Pēc pieraksta var redzēt, ka tabulas kolonnām Klients un Norēķins parasta datu tipu vietā tiek norādīts objekta tips, tāda veidā objekti kā tādi un to atribūti tiks tieši attēloti relācijas tabulas ietvaros. Lai atsauktos uz kāda no objekta atribūtu vērtībām mēs izmatosim punktu sintaksi jeb elementu attiecības (angl. *nested relation*), bet atsaukties tieša veidā uz objektu ar *Value* funkcijas palīdzību mēs nevarēsim, jo, piemēram, lai piekļūtu objektam Klients\_t mums ir nepieciešams griezties tieši pie Klients tabulas kolonnas laukā un tas ir jau relāciju struktūras ietvaros.

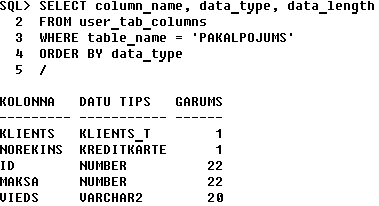
## TABULAS AIZPILDIŠANA

Reģistrēsim divus pakalpojumus, norādot sniegta pakalpojuma unikālo vērtību, tā veidu un maksu, bez tām reģistrēsim informāciju par klientiem, kuri ir izmantojoši šo pakalpojumu un norēķinu veidu, kādu ir izmantojis klients (parasti, tā ir kredītkarte, attiecīgi ievadot par to informāciju). Mūsu gadījumā klients un kredītkarte ir indivīdi un lai mēs varētu aizpildīt šo objektu atribūtu vērtības, mums ir nepieciešams noradīt eksemplāru nosaukumus, bet parējās tabulas kolonnas vērtības tiek aizpildītas bez īpašiem gadījumiem. Datu ievadi veiksim ar sekojošo PL/SQL procedūras izpildi:



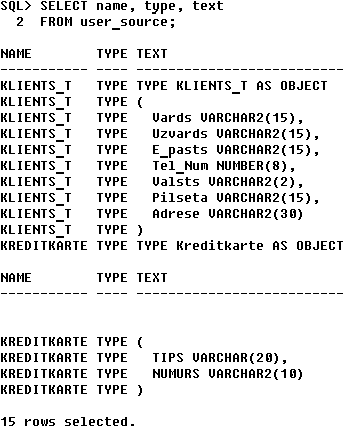
## TABULAS DATU UN META IZGŪŠANA

Pēc datu ievades tabulā mani interesē pašas tabulas struktūra un tās meta dati, šī uzdevuma izpildei izmantošu jau iepriekš apskatītu *user\_tab\_columns* datu vārdnīcas skatu ar mums zināmiem atslēgvārdiem. Vaicājums un tā izpildes rezultāts izskatās šādi:

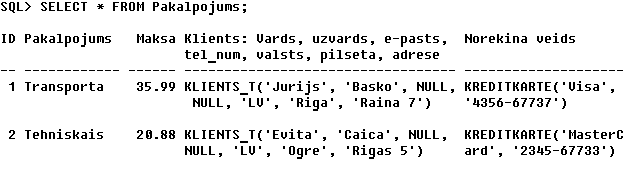


Pēc vaicājuma izpildes rezultātā, var redzēt ka tabula satur atbilstošo struktūru kādu mēs esam definējuši, bet mulsina kolonnu klients un norēķins lauku izmēri, kas ir viens. Šajā kontekstā viens nozīme, ka objekta kolonnu izmērs, teorētiski, ir viens baits (radītāju izmērs).

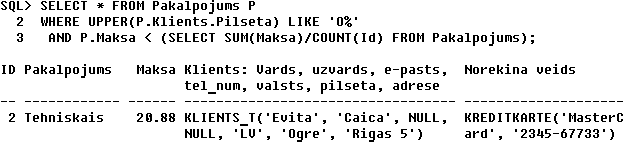
Otrkārt, es vēlētos apskatīties definēta tipa vai pat tipu struktūru, t.i., atribūtus un metodes. Atšķirība no iepriekšēja reizes, kur šādam uzdevumam tika izmantots *user\_types* vārdnīcas skats, kas ļāva virspusēji aplūkot tipus, šoreiz es izmantošu *user\_source* datu vārdnīcas skatu. Izmantojot šo skatu es varu iegūt tipu definējuma pierakstu, tādējādi redzēt visus atribūtus, metodes u.tml., izmantojot *text* atslēgvārdu un papildus izmantošu *name* - datu bāzes objekta nosaukums, *type* – objekta tips, atslēgvārdus. Vaicājums un tā izpildes rezultāts izskatās šādi:



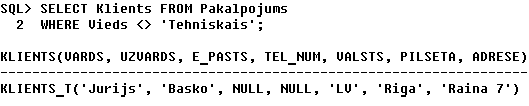
Sākotnēji tika pieminēts, ka šinī tabulā objekti kā tādi un to atribūti tiks tieši attēloti relācijas tabulas ietvaros, lai par to pārliecinātos, izgūsim visus tabulas datus ar šādu parastu relāciju SQL vaicājumu:



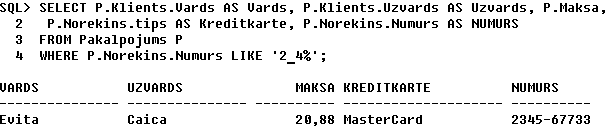
Kā redzams, tabula satur kredītkartes un klienta\_t tipu eksemplārus un kā jau tika piemīnēts, lai atsauktos uz kāda no objekta atribūtu vērtībām mēs izmatosim punktu sintaksi, t.i., tabulas\_apzīmējums.kolonna.atribūts u.tml. Uzskatamāk tas ir paradīts nākama vaicājuma. Vaicājums un tā izpildes rezultāts ir šāds:



Vaicājumā mēs atlasām tabulas rindas vērtības, pēc nosacījumiem, kā klienta maksai par pakalpojumu ir jābūt mazākai par vidējo klientu maksu un klientam ir jānāk no pilsētas, kuras nosaukuma pirmais burts ir „O”. Tādējādi, lai grieztos pie Klients\_t objekta atribūta Pilsēta vērtībām, es izmantoju relācijas objektu pieeju, t.i., izmantojot tabulas apzīmējumu, griežos pie tabulas kolonnas Klients (netieša veida griežos pie Klients\_t objekta) un atlasu kolonnas lauku Pilsēta vērtības pēc nosacījuma (atsaucos uz objektu izmantojot relācijas struktūras). Protams var griezties pie tabulas laukiem nelietojot tabulas apzīmējumu, ka tas ir paradīts apakšvaicājumā, kurā es iegūstu maksas vidēju vērtību, vai arī nākamā piemēra, kur es izgūstu pašu objektu.



Šinī vaicājumā es atlasu objektus, kuriem sniegtais pakalpojums nav tehniskais, vēršoties tieši pie relāciju tabulas kolonnas un lietot tabulas apzīmējumu dotajā gadījumā nav īpašas vajadzības. Nākamā vaicājuma atlasīsim datus par klientu, to kredītkarti un maksu, kura kredītkartes numurs sākas ar ciparu 2 un trešais numurs ir cipars 4. Vaicājums un tā rezultāts ir šāds:



Dotajā vaicājuma, lai izgūtu objektu vērtības nācās obligāti lietot elementu saistības jeb punktu sintaksi.

# TABULA AR OBJEKTU KOLONNU (SALIKTAIS OBJEKTS)

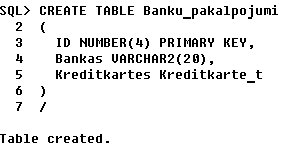
## TABULAS AR SALIKTO OBJEKTU IZVEIDOŠANA

Šajā sadaļā iepazīsimies ar tabulu, kuras kolonnā saturēs saliktu objektu, t.i., pēc būtības dota tabula līdzināsies iepriekš apskatītai, vienīgi tabulas kolonnā saturēs tādu objektu, kurā ietilps cits objekts.

Šīm nolūkam vispirms izveidosim saliktu objektu. Ņemot vērā, ka uz dotu momentu mums ir jau izveidoti divi objektu tipi, mēs varam paņemt vienu no objektu tipiem un papildināt to citā objekta tipā. Piemēram, paņemsim objektu tipu Kredītkarte, kurš pats par sevi satur kredītkartes numura un tipu atribūtus, bet man papildus ir nepieciešami tādi kredītkartes atribūti kā likme un limits, tādēļ izveidošu jaunu kredītkartes objekta tipu Kreditkarte\_t, kurš saturēs visus man vajadzīgus atribūtus, ka arī jau esošo kredītkartes objekta tipu ar tā atribūtiem. Šāda objekta tipa definējums izskatās šādi:

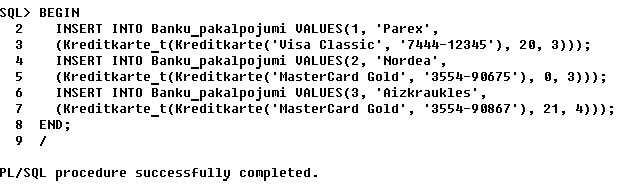


Sekojoši izveidosim Banku pakalpojumu tabulu, kuras uzdevums būs glabāt banku piedāvātos pakalpojumus (mūsu piemērā mēs aprobežosimies tikai ar kredītkartēm). Tātad tabula saturēs kolonnas Bankas - banku nosaukumu glabāšanai, ID - pakalpojuma identifikatora glabāšanai un Kredītkartes - salikta objekta (kredītkartes) un tā vērtību glabāšanai. Tabulas izveidošanas pieraksts:



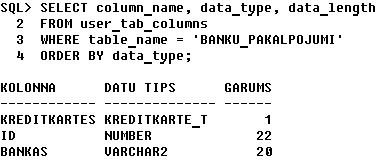
## DATU IEVADĪŠANA TABULĀ AR SALIKTO OBJEKTU

Aizpildīsim tabulu ar trim ierakstiem, kuros paredzēsim banku pakalpojumu unikālo identifikatoru, banku nosaukumu un paša pakalpojuma īpašību noradīšanu. Lai veiktu pakalpojuma, kas ir kredītkartes, atribūtu vērtību aizpildīšanu, līdzīgi kā iepriekšējos gadījumos, tiem vispirms ir jāpiekļūst, norādot attiecīga tipa eksemplāra nosaukumu. Taču šinī gadījumā mums ir jānorāda Kredītkartes tipa eksemplārs iekš cita tipa atribūtiem, tieši tur, kur dotais eksemplārs tika definēts. Datu ievadi veiksim ar sekojošo PL/SQL procedūras izpildi:



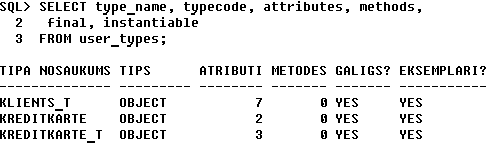
## TABULAS AR SALIKTO OBJEKTU DATU UN META DATU IZGŪŠANA

Pēc datu ievades tabulā rīkošos ka iepriekšēja gadījuma - vispirms apskatīšu pašas tabulas meta datus. Vaicājums un tā izpildes rezultāts izskatās šādi:



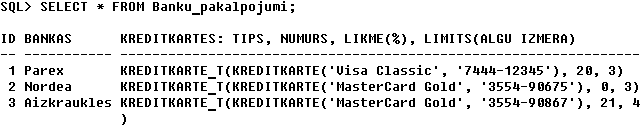
Redzam, ka tabulas struktūru ir atbilstoša tai, kādu mēs esam definējuši. Iepriekš es nebiju izskaidrojis, kāpēc kolonnai ar salikto objektu izmērs ir tikai viens baits. Lieta tāda kā mūsu pirmajai apskatītai objektu tabulai, Oracle datu bāzes vadības sistēma visus objektu atribūtus ieglabā tieši objektu tabulas atsevišķajās kolonnās. Bet dotajām tabulām (tabulas ar objektu kolonnām) Oracle datu bāzes vadības sistēma izveido slēptas kolonnas priekš katra objekta tipa atribūta, kuros parasti tiek glabāti sistēmas ģenerētie objektu atribūtu identifikatori (OID), tādējādi tie kalpo līdzīgi radītājiem uz atribūtiem (no šejienes ir nepieciešama punktu sintakse jeb elementu saistības hierarhija). Manuprāt, šīs izmērs ir rezervēts priekš rādītājiem.

Nākamais objekts, kuram es vēlētos aplūkot meta datus ir jaunizveidotais objektu tips kopā ar pārējiem tipiem. Šīm nolūkam izpildu sekojošu vaicājumu:

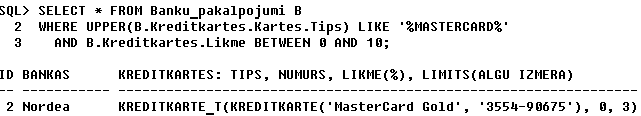


Tātad var redzēt, ka jaunizveidotais objektu tips Kredītkarte-t satur trīs mūsu definētos atribūtus, tam nav nevienas metodes, tās ir galīgais (*final*) - apakštipi nevar mantot dota tipa atribūtus un tām ir iespējams veidot objektu eksemplārus jeb instances (*instantiable*).

Secīgi aplūkosim tabulas datus, objektus un to atribūtu vērtības. Vispirms konstruēsim parastu relāciju SQL vaicājumu tabulas datu izgūšanai. Vaicājuma rezultāts ir šāds:

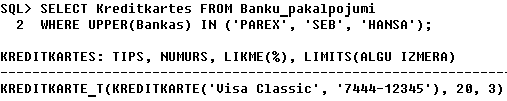


Attiecīgi var redzēt, ka tabula satur trīs kolonnas, no kurām viena satur saliktu objektu. Līdzīgi iepriekšējā sadaļā apskatītai tabulai ar objektu kolonnām, arī dotai tabulai, lai iegūtu kāda objekta atribūta vērtību, ir nepieciešams lietot elementu saistības hierarhiju (punktu sintaksi), vienīgi ar tādu atšķirību, ka šoreiz ir elementu hierarhija daudz plašāka. Piemēram, izstrādāsim vaicājumu, kurā es vēlētos izgūt visu informāciju, kas ir saistīta ar MasterCard kredītkartēm un kurām procentu likme ir no diapazonā 0 - 10. Vaicājums un tā rezultāts ir šāds:

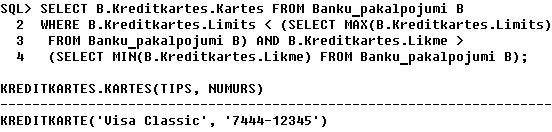


Kā redzams vaicājums izvada tabulas kolonnu rindas vērtības un lai panāktu šādu rezultātu es biju spiests ar tabulas apzīmējumu griezties pie attiecīgas tabulas kolonnas - objekta, un tad pie noteikta kolonnas laukā - objekta atribūta un jā bija nepieciešams griezos no tā uz kādu cita objekta atribūtu, piemēram, B.Kreditkartes.Kartes.Tips.

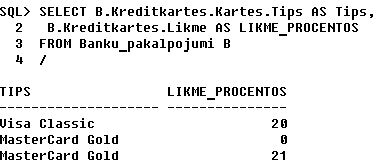
Lai izgūtu saliktu objektu es varu pielietot parastu vaicājumu, kurš atlasīs kolonnas lauka vērtības. Piemēra pēc atlasīsim visas kredītkartes, kādas ir Parex, SEB un Hansa bankas.



Lai izgūtu objektu iekš cita objekta, izmantosim jau mums zināmo punktu sintaksi. Piemēram, izvadīsim kredītkartes, kuru procentu likme ir nemazāka par vismazāko procentu likmi kredītkartēm un kuru darbu algu izmēra limits nav lielāks vai vienāds ar maksimālo darba algas izmēra limitu kredītkartēm.



Kā pēdējo izstrādāsim vaicājumu, kurš atlasīs noteiktas objektu vērtības - mūsu gadījumā objekta Kredītkarte atribūta Tips vērtību un objekta Kredītkarte\_t atribūta Likme vērtību.



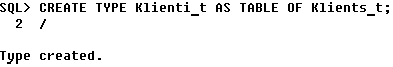
# TABULA AR KOLEKCIJU

## TABULAS AR KOLEKCIJU IZVEIDOŠANA

Datu kolekcija ir speciāls tips, kurš pats no sevis attēlo sakārtotu vai nesakārtotu, neierobežota viena tipa elementu daudzumu. Kolekcijas ļauj veidot datu struktūras, kuras tiek realizētas kā ieliktas (iekļautas) viens otrā tabulas. Ieliktas tabulas elementi glabājas atsevišķā glabājamā tabulā, kura satur kolonnu, kas identificē vecākas tabulas rindu vai objektu, pie kura pieder katrs elements.

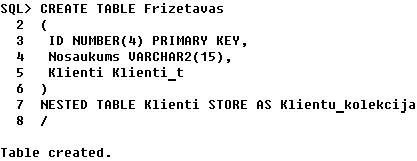
Šinī sadaļā mēs praktiski izveidosim klientu objektu kolekciju, kuru iekļausim mūsu jaunizveidotajā tabulā ar nosaukumu Frizētava. Šādi mēs varam izveidot klientu sarakstus frizētavām, kas ir uzskatāmi ērti.

Tā kā klienta objekta tips Klients\_t mums jau ir izveidots, mēs varam uzreiz veidot abstrakto datu struktūru - tabulas objektu tipu, kurš sevī saturēs neierobežotu objekta tipa Klients\_t eksemplāru skaitu. Šīm nolūkam veiksim šādu pierakstu:



Šajā objektu kolekcijas tipā, līdzīgi iepriekš apskatītai objektu tabulai, gadījumā, jā būs nepieciešams atsaukties uz pašu objektu un to atribūtiem, mums varēsim lietot funkciju *VALUE*.

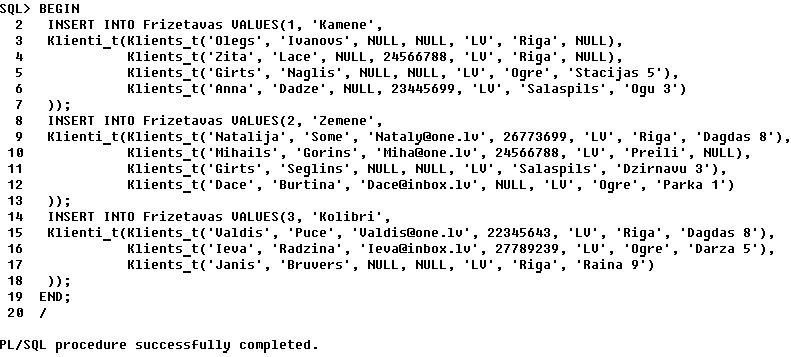
Tagad varam ķerties pie pašas tabulas ar kolekciju veidošanas. Tabulā paredzēsim kolonnas ID - frizētavas unikālo identifikatoru glabāšanai, Nosaukums - frizētavu nosaukumu glabāšanai un Klienti - saturēs klientu kolekciju jeb sarakstu atbilstoši frizētavām.



Pēc pašas tabulas definējuma, obligāti ir jānorāda ka dota tabula saturēs kolekciju, kas tiek izdarīts noradot kolonnas nosaukumu, kurā tiks saistīta ar iekļautu tabulu, mūsu gadījuma tāda kolonna ir Klienti. Šī kolonna glabās saites uz citu tabulu – iekļautu tabulu, kurai mēs piešķiram nosaukumu Klientu kolekcija, savukārt šī jau glabās saites (oid) uz objektu kolekcijām, tāda veida tā kalpos kā starp tabula starp objektu kolekcijām un tabulu.

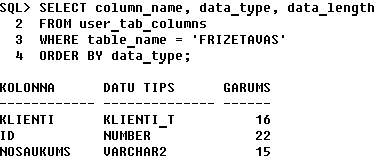
## DATU IEVADĪŠANA TABULĀ

Reģistrēsim trīs frizētavas: „Kamene”, „Zemene” un „Kolibri”, bez tam katrai frizētavai sastādīsim visbiežāk apkalpojamo klientu sarakstu jeb kolekciju. Šīm nolūkam ievadām datus par frizētavām divas tabulas kolonnas, bet lai izveidotu klientu sarakstu attiecīgajai frizētavai, vispirms norādām tabulu tipa eksemplāra nosaukumu - Klienti\_t un kurām sastādām klientu kā objektu sarakstu. Lai veiktu klienta informācijas ievadīšanu, ir jāpiekļūst pie objekta atribūtiem un tas ir izdarāms, norādot klienta tipa eksemplāra nosaukumu - Klients\_t. Datu ievadi veiksim ar sekojošo PL/SQL procedūras izpildi:



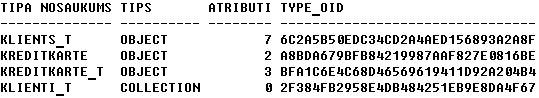
## TABULAS DATU UN META DATU IZGŪŠANA

Līdzīgi visiem iepriekšējiem gadījumiem, arī šoreiz vispirms apskatīšos tabulas meta datus ar šāda vaicājuma palīdzību:



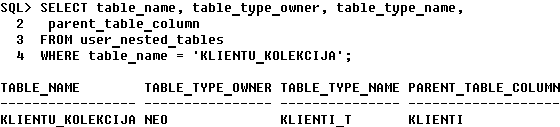
Nevar nepamanīt, ka kolekcijas kolonnas lauku izmērs ir 16 baiti, secīgi var secināt, ka datu bāzes vadības sistēma rezerve 16 baitus priekš objektu kolekcijas rādītājiem, kuri rāda uz objektu atribūtiem ar starp tabulas palīdzību.

Nākamais, kas mūs varētu interesēt ir definēta objekta tabulas tipa meta dati. Izpildīsim vaicājumu, kurš izgūs attiecīgas kolekcijas meta datus, izmantojot *user\_types* datu vārdnīcas skatu:



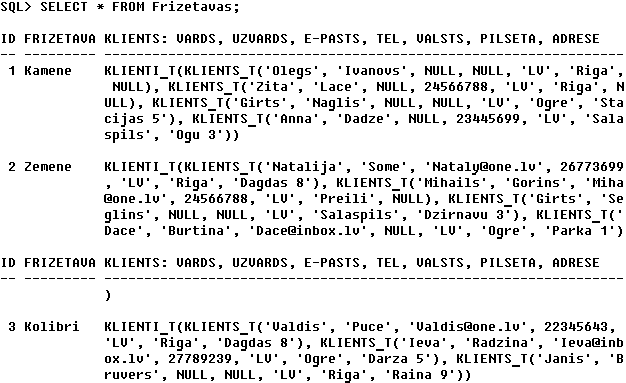
Redzams, ka Klienti\_t objektu tabulas tips jeb kolekcija satur rādītājus un atsaucēs uz objekta atribūtiem.

Vēl viens svarīgais objekts, kuru meta datu aplūkošana spētu dod plašāku priekšstatu par tabulu ar kolekciju ir Klientu\_kolekcijas starp tabula. Šo meta datu izgūšanai ir jāizmanto *user\_nested\_tables* datu vārdnīcas skats ar šādiem atslēgvārdiem: *table\_name* - starp tabulas nosaukums, *table\_type\_owner* - tabulas objekta tipa īpašnieks, *table\_type\_name* - tabulas objekta tipa nosaukums, *parent\_table\_column* - vecākas tabulas kolonnas nosaukums, kurā satur objektu identifikatorus (OID), kuri saistās ar šo tabulu. Vaicājums un tā izpildes rezultāts ir šāds:



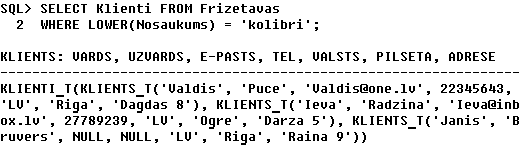
Izgūt datus šāda tipa tabulai (tieši objektu kolekcijas datus) var apvienojot divu pieeju vaicājumus: relāciju SQL vaicājums, izmantojot elementu saistības hierarhiju (punktu sintaksi), kopā ar objektorientēto SQL, izmantojot *TABLE* konstruktoru, kurš ļaus kolekciju vērtības formēt vēlama tabulas struktūrā, kopā ar *VALUE* funkciju. Secīgi aplūkosim vairākus piemērus, lietojot abas pieejas kopā.

Vispirms konstruēsim parastu relāciju SQL vaicājumu, kurš izvadīs visas tabulas Frizētavas ierakstus. Vaicājums un tā izpildes rezultāts ir paradīts nākamā lappusē.

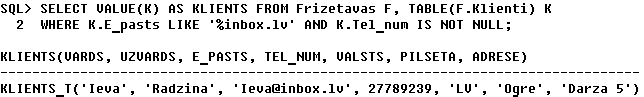


Vaicājuma izpildes rezultāts parāda, iespējams ne visai salasāmi, klientu kolekciju katrai atsevišķai frizētavai; attiecīgi ir var redzēt, kā frizētai Kamene ir četri visbiežāk apkalpojamākie klienti un atbilstoši kopumā ir savākta kaut kāda informācija par šiem klientiem, līdzīgi frizētavai Zemene ir arī četri klienti, bet Kolibri ir tikai trīs.

Nākamā vaicājumā atlasīsim tikai frizētavas Kolibri klientu kolekciju, šīm nolūkam ir izstrādāts parasts relāciju SQL vaicājums, kurā nav nepieciešamības lietot punktu sintaksi. Vaicājums izskatās šādi:

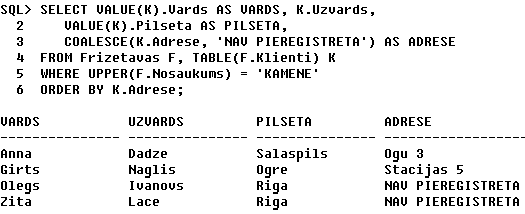


Pieņemsim, ka mums ir radusies nepieciešamība kontaktēties ar kādu no klientiem neatkarība no frizētavām. Tādējādi no klientu kolekcijām, mēs būsim ieinteresēti atrast tādu klientu, kuram ir piereģistrēts gan telefona numurs, gan e-pasts (vēlams inbox.lv). Dotais OSQL vaicājums un atrastais klients ir paradīti nākamā lappusē.



Izstrādājot šo vaicājumu nācās lietot divas pieejas kopā, jo mēs nevaram piekļūt pie eksemplāra Klients\_t atribūtu vērtībām, izmantojot vienīgi elementu saistības hierarhiju, jo Klients\_t tips nav definēts kā tabulas kolonas atribūts un pretēji, mēs nevaram atsaukties uz Klienti\_t tipu, izmantojot funkciju *VALUE*, jo šīs tips tabulā ir definēts kā tabulas kolonas atribūts. Tāpēc sākotnēji, izmatojot elementu saistības, griežamies pie tabulas objektu tipa un tā kolekciju nododam *TABLE* konstruktoram, kurš no kolekciju objektu atribūtu vērtībām formēs jaunu tabulu, no kuras mēs atlasīsim klienta eksemplāru, pielietojot funkcija *VALUE*.

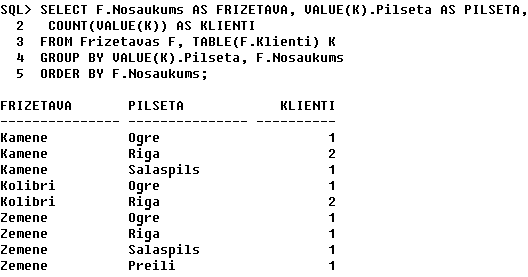
Jā mums būs nepieciešams atlasīt kolekciju objektu atribūtu vērtības, mēs konstruēsim līdzīgo vaicājumu. Piemēram, atlasīsim frizētavas Kamene klientu vārdus, uzvārdus, adresi un pilsētu, kurā viņi dzīvo. Gadījumā jā kādam klientam nebūs piereģistrētas adreses, tad izmantojot funkciju *COALESCE* nulles lauku vērtību vietā izvadīsim paziņojumu „nav piereģistrēta”. Vaicājums un tā rezultāts ir šāds:



# OSQL VAICĀJUMI

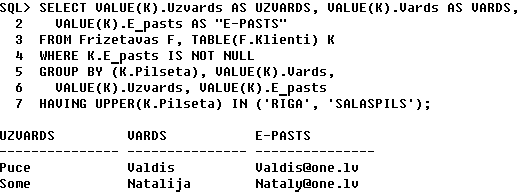
Šajā nodaļā apskatīsim dažus OSQL vaicājumus, izmantojot mums jau pazīstamo funkciju *VALUE* un tabulas konstruktoru *TABLE*, pie tām iepazīsimies ar dažiem jauniem OSQL papildus operatoriem, kuri tiek pielietoti operācijām ar kolekcijām. Tātad par pamata tabulu ņemsim iepriekšēja nodaļā izveidotu tabulu Frizētavas.

Sākotnēja vaicājumā sagrupēsim un saskaitīsim klientus frizētavām vadoties pēc pilsētām, šādi uzzināsim cik kopā klientu no tādam pilsētām kā Rīga, Salaspils un Ogre ir katrai frizētavai. Vaicājums un tā rezultāts ir šāds:

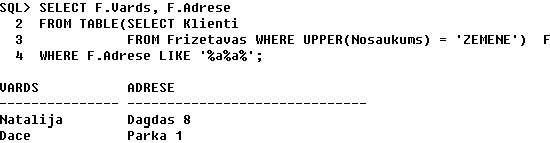


Vadoties pēc izvades, redzam, ka frizētavai Kamene ir tikai viens bieži apkalpojamais klients no Ogres, divi no Rīgas un viens no Salaspils u.t.t.. Pats vaicājums tika konstruēts, izmantojot divas pieejas - elementu saistību un objektu SQL (sīkāk par to ir minēts iepriekšēja nodaļā, kur tika konstruēti līdzīgie vaicājumi).

Pieņemsim, mums ir jāatrod klienti no Rīgas vai Salaspils pilsētām, kuriem ir reģistrēti e-pasti, izmantojot kurus, mēs varēsim kontaktēties ar tiem. Šīm nolūkam, sastādīsim šādu vaicājumu:

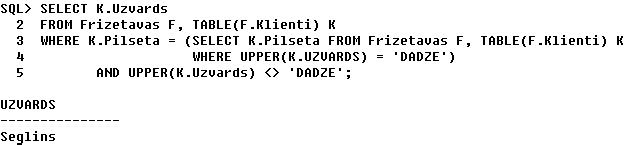


Nākamā vaicājumā atradīsim klientus no frizētavas Zemene darbinieku kolekcijas, kuru adrese kopumā satur nemazāk kā divus a burtus.



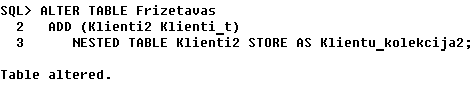
Kā redzams tika sameklēti divi tādi darbinieki. Šoreiz *TABLE* formā tabulas struktūru, no apakš vaicājumā atlasītas kolekcijas.

Pieņemsim, mēs zinām, ka frizētavu kolekcijā ir divi klienti no Salaspils un mums ir zināms vienā tāda klienta uzvārds - Dadze, bet otra klienta uzvārdu mēs nezinām. Uzzināt viņa uzvārdu ir mūsu nākamais uzdevums. Šīm nolūkam, sastādīsim šādu vaicājumu:

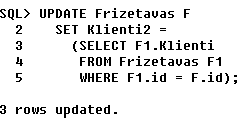


Tātad dotais vaicājums balstās uz loģisko izpratni, kuras būtība ir tāda - mēs atlasam tādu klientu, kurš ir no tādas pašas pilsētas ka klients Dadze, pats šīs klients nav Dadze, rezultāta uzzinām kā dota klienta uzvārds ir Segliņš.

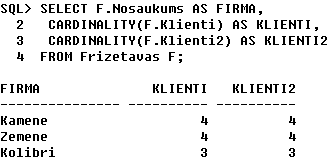
Objektorientētais SQL paredz vairākas operatorus, kuri ļauj operēt ar kolekciju. Lai mūsu turpmākos piemēros mēs varētu šos operatorus pielietot, vispirms ir nepieciešams papildināt pašu tabulu Frizētavas ar jaunu kolonnu Klienti2, kurā saturēs kolekciju. Līdz ar to izveidosim vēl vienu ielikto tabulu, ar kuru būs saistīti vecākas tabulas kolonnas laiki, savukārt, šī tabula būs saistīta ar objektu atribūtiem (oid). Tās viss tiks izdarīts ar *ALTER TABLE* komandas vaicājumu.



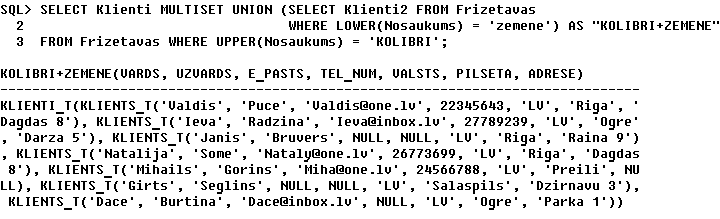
Nākamais solis ir šīs kolonnas aizpildīšana ar objektu kolekciju. Ērtības labam, mēs aizpildīsim šo kolonnu ar kolonnas Klienti kolekciju, izmantojot *UPDATE* datu modifikācijas vaicājumu, kas ir parādīts nākamā lappusē.



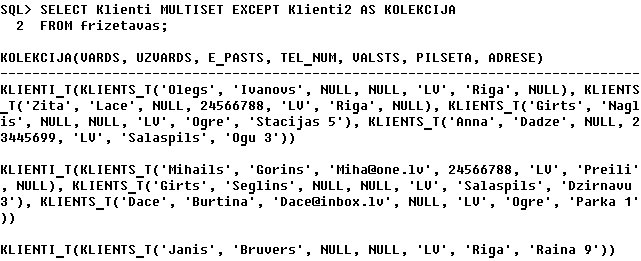
Funkcija *CARDINALITY* (dažreiz tiek dēvēts arī par operatoru) atgriež masīva vai kolekcijas elementu skaitu. Piemēram, pielietojot šo funkciju, mēs varam uzzināt cik katrai frizētai kopā ir piereģistrēti klienti.



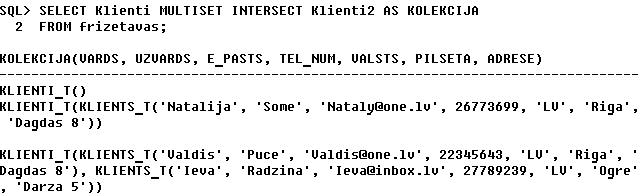
Operators *MULTISET UNION* atgriež abu iekļautu tabulu elementus apvienotus kopā. Piemēram, apvienosim iekļautas tabulas kolonnas objektus, kuri pieder frizētavai Kolibri ar otras iekļautas tabulas kolonnas objektiem, kuri pieder frizētavai Zemene. Rezultātā iegūstam vienu lielu kolekciju.



Operators *MULTISET EXCEPT* analizē divu iekļautu tabulu kolekcijas un atgriež kolekciju ar elementiem, kuri ir pirmās iekļautas tabulas kolekcijā, bet nav otras tabulas kolekcijā. Iegūtas kolekcijas, pielietojot praktiski šo operatoru, ir paradītas nākama lapaspusē.



Eksistē pretējs iepriekš aplūkotajām operatoram operators *MULTISET INTERSECT*, kurš salīdzinot divu iekļautu tabulu kolekcijas un atgriež kolekciju, kurā ir abas šīs tabulas. Pielietojot praktiski šo operatoru, tika iegūtas šādas kolekcijas:



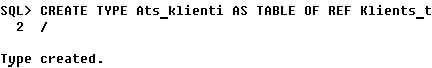
Bez tām eksistē arī citi OSQL operatori, kurus mēs visus praktiski neapskatīsimies, tāda veida uzreiz ķeroties pie nākamā uzdevuma.

# SAITES VIENS PRET DAUDZIEM REALIZĀCIJA

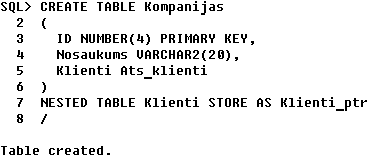
Viena no Oracle datu bāzes lieliskajām iespējām ir iespēja tieši saistīt objektus sava starpā ar atsaucēm, kas ir pretēji tām ka tie tiek saistīti relāciju datu bāzes, kurās katru reizi vēršoties pie objekta, tam ir jāpārsaistās.

Mūsu piemērā mēs izveidosim nelielu tabulu Kompānijas, kuras uzdevums būs glabāt katras kompānijas apkalpojamo klientu sarakstu. Šīm nolūkam mēs izmantosim jau eksistējošu klienta objekta tipu Klients\_t un objektu tabulu, kurā glabā eksemplārus un kuru fiziskas adreses mēs izmantosim, lai tos tieši piesaistītu mūsu jaunizveidotajai Kompānijas tabulai. Lai veiktu šādu sasaisti mums būs nepieciešams atsauču objektu tabulas tips, kas ir tās pamats objektu sasaistei tieši ar atsaucēm. Papildus izmantosim ielikto tabulu, tā ļaus mums piesaistīt bāzes tabulai objektu grupas, ka arī objektus, kuru vērtības atkārtojas (dzīves situācija viens objekts var izmantot vairākus servisus).

Tātad vispirms veidojām atsauču (atslēgvārds *REF*) objektu tabulas tipu. Šīm nolūkam veiksim šādu pierakstu:

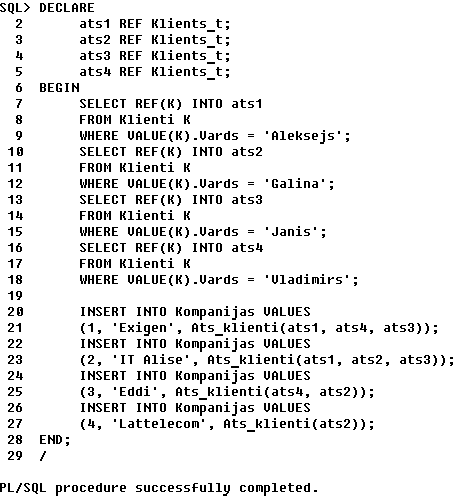


Tagad varam veidot pašu tabulu. Tabulā definējam kolonnas ID - kompānijas unikālo identifikatoru glabāšanai, Nosaukums - firmu nosaukumu glabāšanai un kolonnu Klienti, kurai noradām atsauču objektu tipu. Tabulas definējums izskatās šādi:



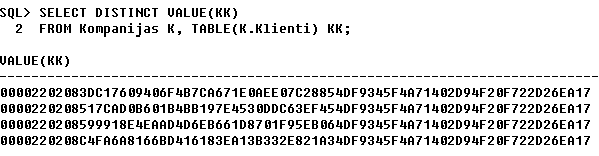
Ar frāzi NESTED TABLE Klienti STORE AS Klienti\_ptr, mēs izveidojam ieliktu tabulu, kura no dota brīža tiek saistīta ar tabulas Klienti kolonnu un Klients\_t objektiem.

Pēdējais un viss būtiskākais, kas mums atliek, ir veikt pašu objektu tiešo sasaisti ar tabulu pēc adresēm. Šīm nolūkam sastādām nākamā lappuse paradītu PL/SQL procedūru.

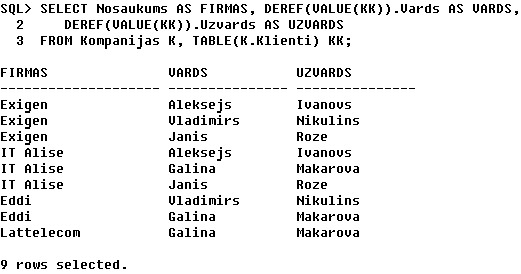


Var redzēt, ka sākotnēji tiek deklarēti četri atsauces objekta tipa mainīgie, kuriem procedūras izpildes blokā ietvaros tiek piešķirtas klientu objektu fiziskas adreses, t.i., izmantojot funkciju *REF*, kura atgriež objekta fizisko adresi, tā ar komandu *INTO* tiek piešķirtā atsauces mainīgajām, kur pati adrese ir tāda objekta adrese, kura vārda atribūta vērtība atbilst nosacījumā noradītai. Pēdējā blokā daļā mēs aizpildām tabulu ar vērtībām, kolonnā ar kolekciju norādot objektu adreses ar mainīgo palīdzību.

Apskatot tabulas objektu kolekciju, redzam ka objektu un to vērtību vietā tiek glabātas adreses, kuras rāda tieša veida rāda uz objektiem.



Lai piekļūtu pašiem objektiem pēc to adresēm ir jāizmanto funkciju *DEREF*, kura ir reversa funkcijai *REF*. Piemēra pēc izvadīsim klientus, kādi ir reģistrētajām kompānijām, vaicājums un tā izpildes rezultāts ir šāds:

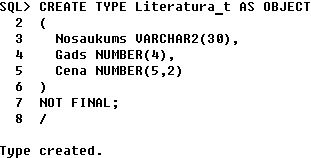


Acīm redzami, var secināt kā saite viens pret daudziem tiek realizēta ar atsauču un ieliktas tabulas palīdzību.

# MANTOŠANAS PIEMĒRS

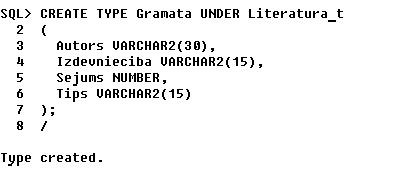
Mantošanas mehānisms ļauj veidot jaunus datu tipus ar datu un metožu mantošanu no citiem tipiem. Tās dod iespēju dažās kopīgas īpašības, kuras raksturīgas vairākiem datu tipiem, apvienot vienā kāda kopīga priekš viņiem datu tipā. Mantošanas hierarhija varbūt daudzlīmeņu.

Realizēsim mantošanas piemēru par literatūras avotiem un to mantošanas hierarhiju. Vispirms izveidosim Literatūra\_t supertipu, kurš saturēs atribūtus Nosaukums - literatūras avota nosaukums, Gads - izdošanas vai publicēšanas gads, Cena - cena par literatūras avotu. Minētie atribūti piemīt vairākiem, man zināmiem, literatūras avotiem. Tipa definēšanas pieraksts ir sekojošs:

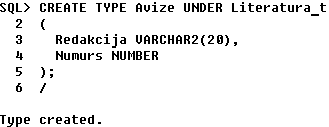


Iepriekšējas nodaļas tipa piemīnēts atslēgvārds *FINAL*, kurš pēc noklusējuma tiek piešķirts visiem jaunizveidotajiem objektu tipiem. *FINAL* nosaka mantošanas nosacījumu, tas neļauj no šī tipa veidot apakštipus. Mums ir jāpiešķir *NOT FINAL* vērtību.

Definēsim grāmatas apakštipu, kurš mantos supertipa Literatūra\_t atribūtus (atslēgvārds *UNDER*). Bez mantotam īpašībām, grāmatai ir sev raksturīgas īpašības, tādas ir autors, izdevniecība, sējums un tips - pasaka, romāns, detektīvs u.tml..Grāmatas tipa definējums ir šāds:



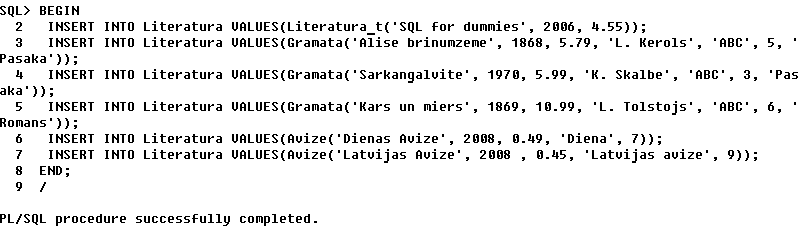
Otrkārt, izveidosim avīzes apakštipu, kurš arī mantos bāzes tipa īpašības un saturēs sev raksturīgas īpašības: redakcija un avīzes iznākšanas kārtas numurs. Avīzes tipa definējums ir paradīts nākama lappusē.



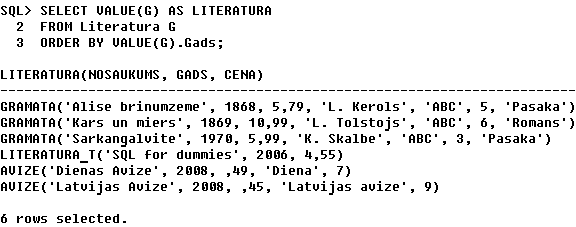
Nākamais solis, ir paredzēt tabulu, kurā glabāsies literatūras avotu eksemplāri, šīm nolūkam izveidosim objektu tabulu.



Tagad, kad viss ir gatavs, var veikt objektu tabulas aizpildīšanu. Norādot datus par kādu literatūras avotu ir jāievēro kāda tipa objekts tās ir, piemēram, ja tā ir avīze, tad ir jāgriežas pie Avīzes tipa objekta, savukārt, jā tas ir kāds cits literatūras avots, kuru nevar attiecināt nevienam apakštipam, tad mēs griezīsimies pie bāzes tipa objekta. Datu ievadīšanas PL\SQL procedūra ir šāda:



Izvadot objektu tabulas saturu, var redzēt, ka visi objektu tipi satur trīs kopējus atribūtus no bāzes klases un sev raksturīgos atribūtus.



# SECINĀJUMI

Praktiskā darbā galvenais mērķis bija iepazīties ar relāciju objektu datu bāzes struktūru un objektorientēta SQL paplašinājumiem. Darba gaitā tika iegūtas vairākas zināšanas un praktiskās iemaņas objektorientētas metodoloģijas pieejas izmantošanai datu bāzēs dažādu problēmu risināšanai.

Tika izprasta objektu tipu un eksemplāru saistība ar relāciju struktūrām - saites un objektu identifikatori, objektu uzglabāšanas īpatnības tabulas kolonnas, objektu kolekciju organizēšana izmantojot iekļautas tabulas un operēšana ar to - OSQL paplašinājumi, konstruktori, atsauces u. tml. Darbs šķita interesants, ar to kā ļāva salīdzināt objektorientētas pieejas pielietojumu gan datu bāzes, gan programmatūras izstrādē. Būtiski vairāk pretrunās sagādāja nevis klāšu bet tipu lietošana, nevar atstāt vienaldzīgu tādu objektorientēto paradigmu nodrošināšana datu bāzes ka iekapsulēšana, tipu konstruktori u.t.t.. Viss interesantāk bija saprast, ka tieši objekti tiek saistīti ar relāciju datu bāzes struktūrām, pat rādās interese pastrādāt ar pilnīgi objektorientēto datu bāzi.

Runājot, par praktiska darbā sastaptajām grūtībām, vairāk laikā tika patērēts tieši dokumentācijai izstrādei, jo prasīja papildus laiku visu neskaidrību noskaidrošana. Darbs izrādījās apjomīgs.

# IZMANTOTA LITERATŪRA

1. Jānis Eiduks. Lekciju konspekti studiju priekšmetā „Lielās datu bāzes” - Rīga, RTU, 2007.
2. Ян Абрамсон, Майкл С. Эбби, Майкл Кори. «Oracle 10g первое знакомства» - Москва, «Лори», 2007.
3. Филипп Андон, Валерий Резничкою „Язык запросов SQL учебный курс” - Москва, „Питер”, 2006.

.