Rīgas Tehniska universitāte

Datorzinātnes un Informācijas tehnoloģijas fakultāte

Pirmais praktiskais darbs mācību priekšmetā

„Datu bāzu vadības sistēmas”

Datu bāzes struktūras definēšana

Izstrādāja: Sergejs Terentjevs

2. kurss, 9.grupa

Apl.nr. 061RDB140

Pārbaudīja: lekt. V. Vinogradova

2006./2007. māc. g.

# ANOTĀCIJA

Laboratorijas darbā gaitā tika izstrādāta *Kancelejas Preces* datu bāze ar CASE modulēšanas rīka *SyBase PowerDesigner* palīdzību.

Laboratorijas darbā pārskatā ir aprakstītas konceptuālas prasības datu bāzei, speciālu prasību risinājumi, doti tabulu, saišu un atribūtu skaidrojumi, ir apskatīti datu bāzes projektēšanas un modulēšanas soļi, moduļa verifikācijas apraksts MS Access 2007 vidē. Laboratorijas darbā pārskats satur vienu pielikumu, kura uzradīts ģenerētais SQL skripts.

Datu bāzes modelis tika sastādīts *PowerDesigner* CASE rīkā un realizēts MS Access 2007 tukšajā datu bāzē uz *Intel Centrino Duo* tipā procesorā HP pārnēsājamā datora.

Laboratorijas darba pārskata apjoms: 20 lappuses izklāsta tekstā, tajā ietilpst 5 nodaļas un 7 apakšnodaļas, literatūras saraksts, 17. attēli, 6. tabulas un 1. pielikums

# SATURS

[ANOTĀCIJA 2](#_Toc191283702)

[SATURS 3](#_Toc191283703)

[Datu bāzes priekšmetiskais pamatojums 4](#_Toc191283704)

[ER Diagrammas projektēšana 5](#_Toc191283705)

[Konceptuāla modeļa veidošana un pārbaude 9](#_Toc191283706)

[Realitāšu izveidošana 9](#_Toc191283707)

[Realitāšu atribūtu definēšana 10](#_Toc191283708)

[Saišu definēšana 11](#_Toc191283709)

[Datu bāzes projektējuma realizēšana 13](#_Toc191283710)

[Fiziska modeļa iegūšana no konceptuāla modeļa 13](#_Toc191283711)

[Datu bāzes struktūras izveidošana 14](#_Toc191283712)

[Datu bāzes moduļa pārbaude MS Access vidē 15](#_Toc191283713)

[SECINĀJUMI 16](#_Toc191283714)

[LITERATŪRAS SARAKSTS 17](#_Toc191283715)

[PIELIKUMS 18](#_Toc191283716)

[Ģenerētais SQL skripts 18](#_Toc191283717)

# Datu bāzes priekšmetiskais pamatojums

Pieņemsim, man – izstrādātājam, ir atnācis pasūtījums ar lūgumu izstrādāt kādas kompānijas visam nodaļām kopīgi vienotu centralizētu kancelejas preču pasūtījumu informācijas sistēmu.

Pašlaik katra no kompānijas nodaļām patstāvīgi veic kancelejas preču pasūtījumus, kas apgrūtina kopējo kancelejas preču iegādei nepieciešamo izmaksu noskaidrošanu un šo izmaksu ierobežošanu, ka arī izdevīgāko noslēgto līgumu sameklēšanu.

Veicot lietotāju aptauju tika noskaidrotas šādas prasības sistēmai:

* Lietotājam (nodaļas darbiniekam) ir dotas tiesības veikt pasūtījumu un viņam ir katru reizi jāautorizējas sistēmā.
* Nodaļu menedžeris kontrolē izmaksas un apstiprina kancelejas preču pasūtījumus, viņam arī ir jāautorizējas sistēmā.
* Nodaļu menedžeris nodod pasūtījumu sarakstu piegādātājiem un vienojas par pasūtījuma piegādei nepieciešamo laiku. Lai būtu vieglāk kontrolēt pasūtījumus, precēm ir jābūt sadalītām kategorijas.
* Saņemot pasūtījumu, nodaļu menedžeris pārbauda preču daudzumu, atzīme, ka pasūtījums ir pienācis un veic preču sadalīšanu starp nodaļām.
* Viens darbinieks var veikt vairākus pasūtījumus.
* Vairāki pasūtījumi var saturēt vienu un to pašu pasūtījuma būtību (tanī paša datumā vairāki nodaļu darbinieki var pasūtīt vienādas pildspalvas un zīmuļus)
* Viena pasūtījumā var norādīt vairākas preces.
* Vairākas preces var attiecināt kādai noteiktai kategorijai.
* Vairākas preces var piegādāt vairāki pasūtītāji.

Informācijas sistēmai ir jāsatur datu bāze, kurai ir jāglabā dati par kompānijas personālu, darbinieku veiktajiem pasūtījumiem, precēm, piegādātājiem un kategorijām.

# ER Diagrammas projektēšana

Tātad informācijas sistēmai ir jāizveido piecas realitātes:

* Darbinieki - glabās informāciju par kompānijas personālu. Šai realitātei būs divas lomas:
  + Nodaļas darbinieks,
  + Nodaļu menedžeris.
* Pasūtījumi - glabāsies visu darbinieku veiktie pasūtījumi. Šai realitātei būs divas lomas:
  + Saņemts,
  + Nav piegādāts.
* Preces - glabās informācija par pasūtītām precēm.
* Kategorijas - glabās informācija par preču viediem.
* Piegādātāji - saturēs informāciju par piegādātājiem.

Nākama projektēšanas posmā ir jānoskaidro visas asociācijas starp realitātēm un jāizveido relācijas saites. Vadoties pēc prasībām (skat. 4.lpp.), būs jāizveido piecas relācijas saites.

Starp realitātēm *Darbinieki* un *Pasūtījumi* ir jāizveido saite *viens pret daudziem*, jo viens darbinieks var veikt vairākus pasūtījumus.

Starp realitātēm *Pasūtījumi* un *Preces* ir jāizveido saite *daudzi pret daudziem*, jo vairāki pasūtījumi var saturēt vairākas preces.

Starp realitātēm *Kategorijas* un *Preces* ir jāizveido saite *viens pret daudziem*, jo viena kategorijā var ietilpt vairākas preces.

Starp realitātēm *Preces* un *Piegādātāji* ir jāizveido saite *daudzi pret daudziem*, jo vairākas preces var piegādāt vairāki piegādātāji.

Realitātei *Darbinieki* ir paredzētas divas lomas – *Nodaļas darbinieks* un *Nodaļu menedžeris*. Šīs lomas ļauj informācijas sistēmai atpazīt lietotāju, kurš ir autorizējies sistēmā un piešķir šīm lietotājam atbilstošas tiesības. Piemēram, nodaļas darbinieks var veikt pasūtījumu, norādīt pasūtījuma datumu, preci, daudzumu, bet nevar piešķirt statusu pasūtījumam, noradīt piegādātāju u.tml.

Citai realitātei *Pasūtījumi* arī eksistē divas lomas – *Saņemts* un *Nav piegādāts*. Šīs lomas nosaka stāvoklī kādā atrodas kāds pasūtījums, t.i., pasūtījuma status. Ja pasūtījums ir saņemts nodaļu menedžeris piešķir šīm pasūtījumam *Saņemts* lomu.

Šeit gribētos pieminēt starp tabulu - *Pasūtījuma priekšmets,* kas tiks izveidota starp tabulām *Pasūtījumi* un *Preces* datu bāzes modulēšanas laikā, šādu starp tabulu veidošanos es varu pamatot ar to, ka parasti saites starp tabulām tiek uzdotas ar *viens pret daudziem* attiecību. Pirmajā attēla es esmu paradījis šo starp tabulu, jo pievienoju tai klāt atribūtus *Daudzums* un *PiegDatums*, kas, savukārt, ir nepieciešami, lai apvienotu datus par vairākām precēm vienādos pasūtījumos - vienā datumā vairāku nodaļu un pašas nodaļas darbinieki var pasūtīt vienādu preci, kas būtībā ir viens un tas pats pasūtījums, tādā veida es varu uzzināt cik kopīgi liels ir nepieciešamas preces daudzums u.tml.

Realitāšu shēma un saites starp tām ir paradītas 1.attēlā:

Pasūtījumi

Pasūtījuma priekšmets

Preces

Kategorijas

Piegādātāji

Darbinieki

saņemts

nav piegādāts

nodaļas darbinieks

Nodaļu menedžeris

atbilst

satur

veic pasūtījumu

ietilpst

piegādā

1.att. Realitāšu saišu diagramma

Turpmāka projektēšanas posmā ir jānoskaidro kāda tipa ierakstus glabās tabulas ( tabulas atribūti, to datu tips un apraksts ).

Tabula Darbinieki saturēs informāciju par kompānijas personālu. Šī informācija būs nepieciešama, lai noskaidrotu kādām darbiniekam pieder šīs pasūtījums, no kādas nodaļas tika veikts pasūtījums, kādas tiesības ir jāpiešķir sistēmā autorizētām lietotajām. Zemāk ir paradīta tabulas shēma un tas atribūtu apraksts (skat. tabulu 1).

|  |  |  |
| --- | --- | --- |
| **Atribūti** | **Apraksts** | **Datu tips** |
| *DarbID* | Unikāls darbinieka identifikators | *Number* |
| *LietotVards* | Reģistrētais lietotāja vārds | *Text* |
| *Parole* | Lietotāja parole | *Text* |
| *Vards* | Darbinieka vārds | *Text* |
| *Uzvards* | Darbinieka uzvārds | *Text* |
| *Nodala* | Nodaļā, kura strādā darbinieks | *Text* |
| *Menedzeris* | Vai lietotājs ir nodaļas darbinieks, vai menedžeris | *Boolean* |

1. Tabula - tabulas *Darbinieki* shēma.

Tabula *Pasūtījumi* glabās informāciju par visu darbinieku veiktajiem pasūtījumiem neatkarīgi no nodaļas. Tabulā 2 ir paradīta tabulas shēma un tas atribūtu apraksts.

|  |  |  |
| --- | --- | --- |
| **Atribūti** | **Apraksts** | **Datu tips** |
| *PasutID* | Unikāls pasūtījuma identifikators | *Number* |
| *DarbID* | Saite uz *Darbinieki* tabulu | *Number* |
| *PasutDatums* | Datums, kurā veikts pasūtījums | *Date* |
| *Status* | Pasūtījuma stāvoklis – vai ir saņemts, vai nav | *Boolean* |

2. Tabula - tabulas *Pasūtījumi* shēma.

Tabula *Pasūtījuma priekšmets* pildīs starp tabulas funkciju, atvieglojot kopīgi nepieciešamas vienas preces daudzuma un izmaksu noteikšanu visam kompānijas nodaļām. Tabulā 3 ir paradīta tabulas shēma un tas atribūtu apraksts.

|  |  |  |
| --- | --- | --- |
| **Atribūti** | **Apraksts** | **Datu tips** |
| *PasutID* | Saite uz *Pasūtījumi* tabulu | *Number* |
| *PrecesID* | Saite uz *Preces* tabulu | *Number* |
| *Daudzums* | Kopīgi nepieciešamas preces daudzums | *Integer* |
| *PiegDatums* | Vienota pasūtījuma piegādes datums | *Date* |

3. Tabula - tabulas *Pasūtījuma priekšmets* shēma.

Tabula *Preces* saturēs informāciju par lietotāja pieteiktajām precēm. Tabulā 4 ir paradīta tabulas shēma un tas atribūtu apraksts.

|  |  |  |
| --- | --- | --- |
| **Atribūti** | **Apraksts** | **Datu tips** |
| *PrecesID* | Unikāls preces identifikators | *Number* |
| *KatID* | Saite uz *Kategorijas* tabulu | *Number* |
| *Nosaukums* | Preces nosaukums | *Text* |
| *Apraksts* | Preces apraksts | *Text* |
| *CienaParVien* | Cena par vienību | *Float* |
| *PiegID* | Saite uz *Piegādātāji* tabulu | *Number* |

4. Tabula - tabulas *Preces* shēma.

Tabula *Kategorijas* nodrošinās preču klasificēšanu pēc to veidiem . Tabulā 5 ir paradīta tabulas shēma un tas atribūtu apraksts.

|  |  |  |
| --- | --- | --- |
| **Atribūti** | **Apraksts** | **Datu tips** |
| *KatID* | Unikāls kategorijas identifikators | *Number* |
| *Nosaukums* | Kategorijas nosaukums | *Text* |
| *Apraksts* | Kategorijas apraksts | *Text* |

5. Tabula - tabulas *Kategorijas* shēma.

Tabula *Piegādātāji* saturēs informāciju par piegādātājiem. Tabulā 6 ir paradīta tabulas shēma un tas atribūtu apraksts.

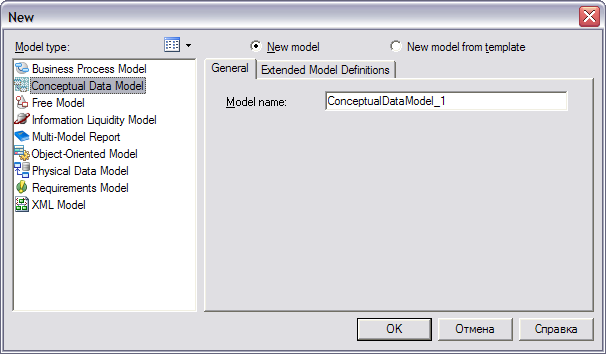
|  |  |  |
| --- | --- | --- |
| **Atribūti** | **Apraksts** | **Datu tips** |
| *PiegID* | Unikāls piegādātāja identifikators | *Number* |
| *Nosaukums* | Piegādātāja nosaukums | *Text* |

6. Tabula - tabulas *Piegādātāji* shēma.

# Konceptuāla modeļa veidošana un pārbaude

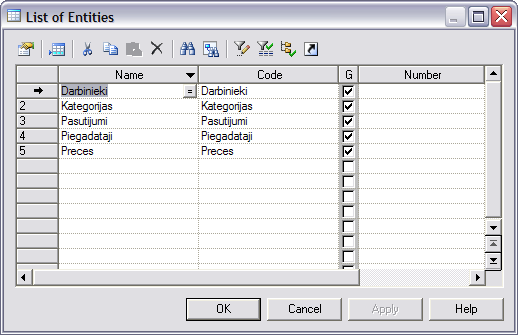
## Realitāšu izveidošana

Lai uzsāktu darbu ar *SyBase PowerDesigner* *12.1* versiju, vispirms ir jāizvēlas datu modelis, kurā tiks projektēta datu bāze. Tas ir izdarāms izvēlnē *File*, izpildot komandu *New ( File🡲New).* Rezultātā tiks atvērta forma *New*, kurā ir jāizvēlas *Conceptual Data Model* modulis ( skat. 2. att.).



1. Att. Formas *New* piedāvātais modeļu saraksts.

Pēc moduļa izvēles var ķerties pie realitāšu izveidošanas, viens no viediem, ka to izdarīt ir izvēlnē *Model* izvēlēties komandu *Entities*. Uz ekrāna parādīsies formā *List of Entities*, kurā pēc kārtas ir jāievada topošas datu bāzes tabulu nosaukumus (skat. 3. att.). Pēc ievadīšanas ir jānospiež pogu *OK*, lai apstiprinātu ievadīto tabulu izveidošanu.

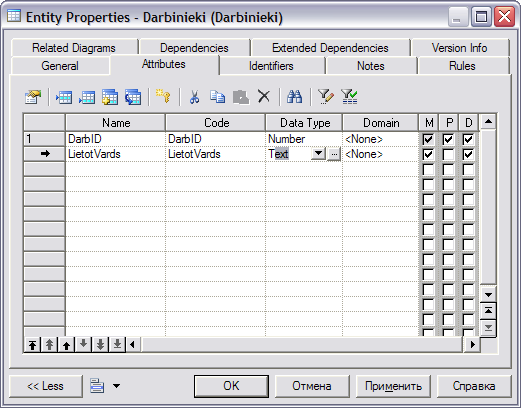
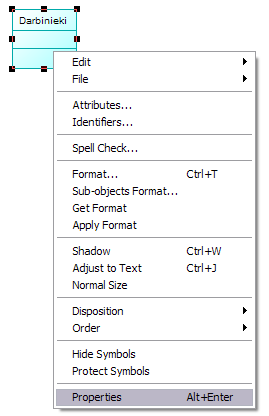


1. Att. Tabulu definēšana.

Tagad mēs esam ieguvuši tabulas, bet esošam tabulām nav atribūtu. Lai definētu atribūtus ir jāpariet uz nākamo solī.

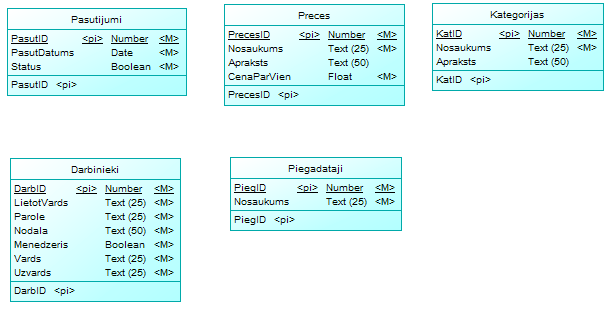
## Realitāšu atribūtu definēšana

Nākamais solis ir atribūtu definēšana tabulām, lai to izdarītu ir jāiezīmē attiecīgu tabulu un jānospiež peles laba poga, tad atvērtajā sarakstā ir jāizvēlas komandu *Properties* ( skat. 4.att.). Atvērsies forma *Entity Properties*, kurā ir jāizvēlas *Atributes* izvēlni (skat. 5. att.). Šeit atbilstošas kolonnas ir jānorāda atribūtu nosaukumi, primārie atribūtu nosaukumi, atribūtu datu tipi. Lielu uzmanību ir jāpievērš kolonnām ar nosaukumu M, P, D. Dotie apzīmējumi nosaka to vai norādītais atribūts būs unikālais identifikators ( P ), tiks radīts tabulā ( D ) un vai dotā atribūta vērtība būs obligāta ( M ), piemēram, tabulā *Preces* pretī atribūtam *Apraksts* es nelikšu ķeksīti pie M, tādējādi lietotājam nebūs obligāti katrai precei jāraksta apraksts.



1. Att. Komandu saraksts 5. Att. Forma *Entity Properties* atribūtu definēšanai.

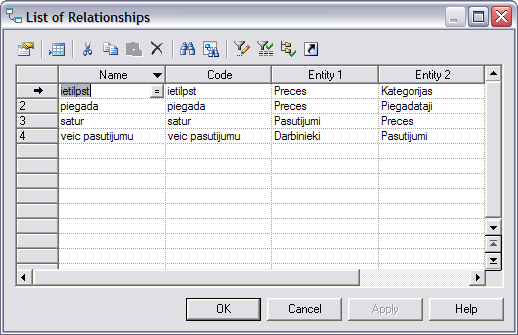
Vēlams formas *Entity Properties* izvēlnē *Identifiers* atkārtoti norādīt atvērtas tabulas unikālo identifikatoru, mana gadījumā atribūtu *DarbID*. Pēc atribūtu definēšanas modelis izskatās šādi (6. att.):



1. Att. Realitātes bez saitēm.

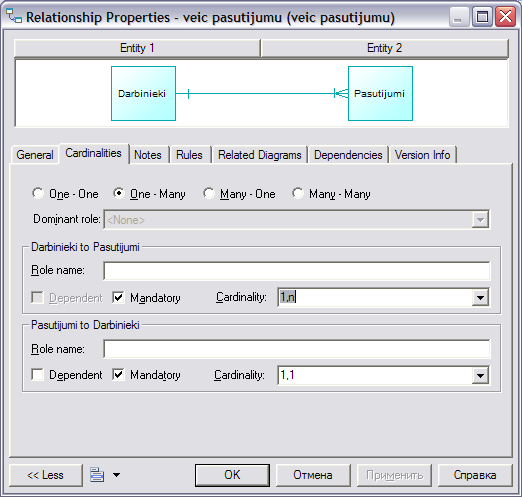
## Saišu definēšana

Turpmākajā projektēšanas solī ir jāveic tabulu savienošanu atbilstoši nosacījumiem, kuri tika izvirzīti 5. lpp. Lai izveidotu saites starp tabulām, izvēlnē *Model* ir jāizvēlas komandu *Relationships*. Rezultātā tiks atvērta forma *List of Relations*, kurā kolonnā *Name* ir jānorāda saišu nosaukumi, kolonnā *Code* saišu primārie nosaukumi, savukārt, kolonnās *Entity 1* un *Entity 2* ir jānorāda tabulas, kuras saistīs izveidota saite (skat. 7. att.). Svarīga ir secība, kāda tabulas tiek savienotas, t.i., no vienas tabulas unikāla identifikatora ir jāvelk saiti uz citas tabulas atribūtu – šeit *Entity 1* ir jānorāda tabula, no kuras saite tiks vilkta, bet *Entity 2* ir jānorāda tabula, kurai tiks pievienota šī saite.



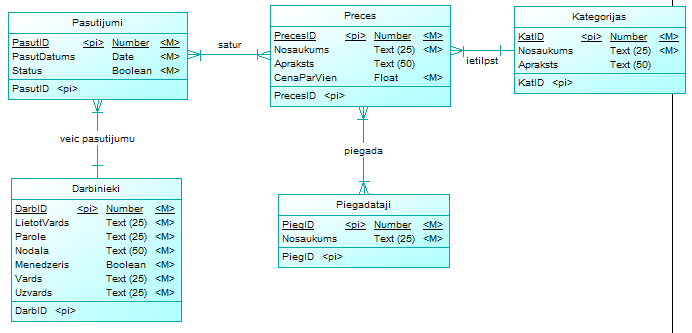
1. Att. Saišu definēšana.

Turpmāk ir jānorāda saišu attiecības. Lai to izdarītu, vispirms ir jāiezīmē attiecīga saitē un jānospiež peles labo poga, atvērtajā sarakstā jāizvēlas *Properties* komandu. Pēc minēto darbību izpildes, atvērsies forma *Relationship Properties*, šeit ir jāizvēlas *Cardinalities* ieliktni ( skat. 8. att.).



1. Att. Saišu attiecību definēšana.

Šinī ieliktnī lietotājam ir dota iespēja norādīt saites attiecības - *viens pret vienu*, *daudzi pret daudziem*, *viens pret daudziem* un *daudzi pret vienu*. Kad ir veikta izvēle, tad ir ļoti svarīgi veikt saites pareizo apzīmējumu, tas ir izdarāms *Cordinality* ievades laukā. Astotajā attēlā saitei no *Pasūtījumi* uz *Darbinieki*, *Cordinality* ievades laukā, es norādīju simbolisku apzīmēju 1, kas izsaka sakarību, ka tikai viens nevis daudzi kopā uzreiz darbinieki veic pasūtījumu. Savukārt, saitei *Darbinieki* uz *Pasūtījumi*, *Cordinality* ievades laukā, es noradīju 1..N apzīmējumu, kas nozīme, ka varbūt vairāki vai arī kopuma tikai viens pasūtījums, ko ir veicis kāds viens lietotājs. Tāda veida tiek izveidotas pārējas saites, kuru attiecības un apzīmējumi atbilst 5. lpp. definētam prasībām un 1. att. paradītajai realitāšu saišu diagrammai. Pēc saišu definēšanas kopēja diagramma izskatās šādi (skat. 9. att.):



1. Att. Konceptuālais modelis ar saitēm.

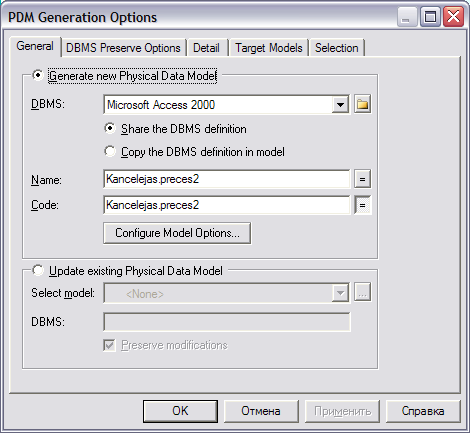
Šeit gribētos piebilst ka turpmāk, izveidojot fizisko modelī starp tabulām *Pasūtījumi* un *Preces*, ka arī *Preces* un *Piegādātāji* izveidosies starp tabulas, jo šo tabulu saitēm ir definētā *daudzi pret daudziem* attiecībā. Vienai tādai tabulai – *Pasūtījuma priekšmets (satur)* – tiks pievienoti divi papildus atribūti *Daudzums* un *PiegDatums*, tas būs lietderīgi vienas visas nodaļas pieteiktas preces daudzuma noteikšanai un tāda kopēja pasūtījuma nodošanai piegādātājiem, ka arī kopīga pasūtījuma saņemšanas datuma atzīmējumam (skat. 11. Att.).

Pirms pariet uz nākošo soli, vēlams izpildīt konceptuāla modeļa pārbaudi - *Tools🡲Check Model.* Ja moduļa pārbaudes laikā netiek izvadīts nekāds kļūdas paziņojums, tad var ķerties pie fiziska moduļa iegūšanas, citādi nepieciešams izlabot kļūdas.

# Datu bāzes projektējuma realizēšana

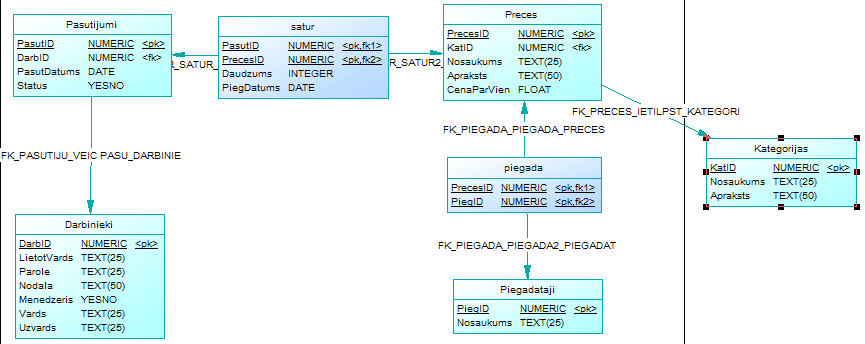
## Fiziska modeļa iegūšana no konceptuāla modeļa

Lai iegūtu datu fizisko modeli, izvēlnē *Tools* ir jāizpilda komanda *Generate Physical Data Model*. Komandas izpildīšanas rezultāta tiks atvērta *PDM Generation Options* forma, kurā vispirms ir jānorāda tas, ka tiks veidots jauns datu fiziskais modelis - ieklikšķinot aplītī pretī norādei *Generate new Physical Data Model* (skat. 10.att.).Atbilstoši mana uzdevuma nostādnei, ir jāizvēlas Microsoft Access 2000 tipa datu bāzes vadības sistēmu, ka rezultātā tiks izstrādāts modulis, ko atbalsta Access ( atbilstoši atribūtu datu tipi u.tml.). Vēlams formas izvēlne *Selection* pārbaudīt vai tiks ņemtas vērā visas definētas realitātes. Pēc minēto operāciju virknes izpildes ir jānospiež poga *OK*, fiziska moduļa izveidošanas apstiprināšanai.



1. Att. Fiziska moduļa iegūšanas forma.

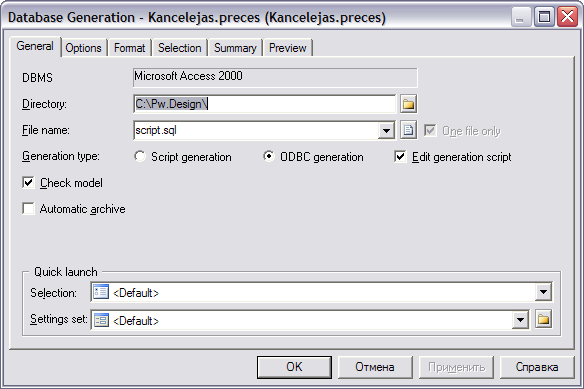
Iegūtas datu bāzes fiziskais modelis izskatās šādi (skat. att. 11):



1. Att. Iegūtais datu bāzes fiziskais modelis.

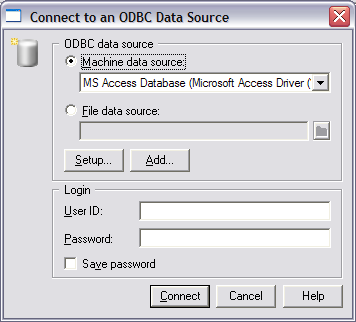
## Datu bāzes struktūras izveidošana

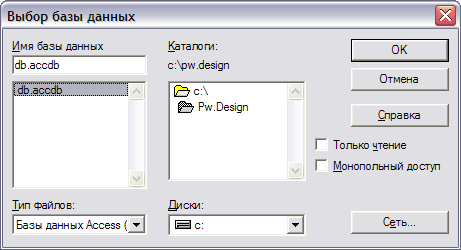
Tagad mēs esam ieguvuši pilnvērtīgo datu bāzes modeli. Nākamais solis ir ieeksportēt esošo moduli MS Access jaunizveidotajā datu bāze ( *Blank database*). Lai to panāktu, izvēlnē *Database* ir jāizvēlas komandu *Database Generation*, rezultātā atvērsies *Database Generation* formā (skat. 12.att.). Šajā formā ir jānorāda direktoriju, kurā tiks saglabāts ģenerētais SQL skripts un jāizvēlas *ODBC generation*, datu bāzes modeļa ģenerēšanai uzreiz *ODBC* objektā. Turpmāk jānospiež poga *OK*.



1. Att. Datu bāzes modeļa ģenerēšanas forma.

Pēc pogas apstiprināt nospiešanas parādās *Connect to an ODBC Data Source* formā, kurā lietotājs var noradīt jau esošo tukšo Access datu bāzi, moduļa ieeksportēšanai uzreiz tajā ( skat. 13. un 14. Att.).



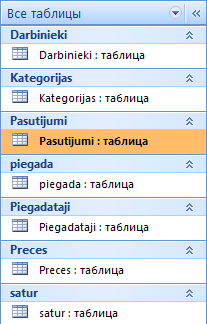


1. Att. ODBC objekta noradīšana. 14. Att. Access datu bāzes noradīšana.

Pēc pogas *Connect* nospiešanas, uz ekrānā tiks izvadīts SQL skripts un tiks piedāvātas koriģēšanas iespējas. Ja paradītais skripts apmierina lietotāju, tad tas var droši spiest *Run* pogu, ka rezultāta notiks savienojums ar MS Access datu bāzi fiziska modeļa ieeksportēšanai ar SQL skripta palīdzību.

## Datu bāzes moduļa pārbaude MS Access vidē

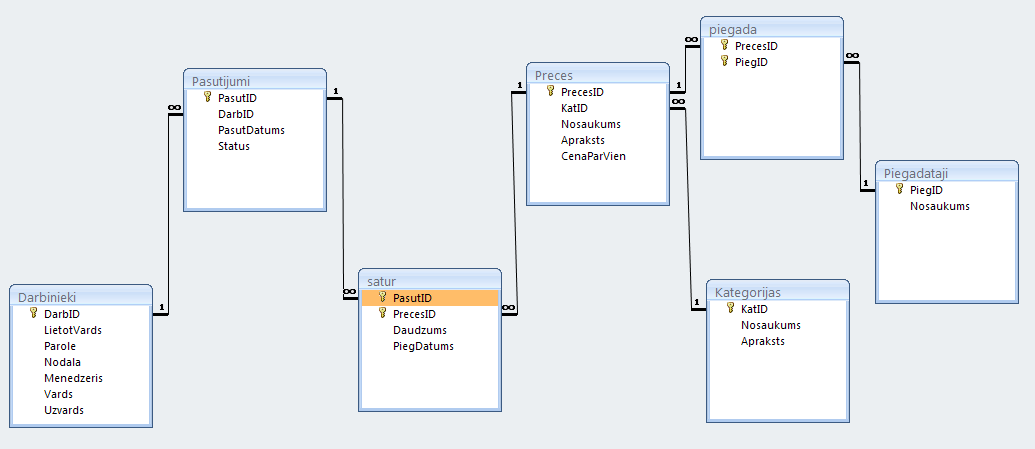
Pēdējais solis ir pārbaudīt tabulu saišu diagrammu MS Access vidē. Pēc MS Access atvēršanas var pamanīt ka kreisajā pusē ir paradītas visas tabulas, kuras tika izstrādātas datu bāzes projektēšanas gaitā, tapāt ir svarīgi pārbaudīt atribūtu datu tipu atbilstību (skat. 15. Att.). Viss ir sakritīs, tad atliek vienīgi veikt tabulas saišu diagrammas pārbaudi. Lai to izdarīti MS Access 2007 vidē ir jāizvēlas *Relationships* komandpoga ( skat. 16. Att.):





1. Att. Datu bāzes *Kancelejas Preces* tabulas. 16. Att. *Relationships* komandpoga.

Pēc komandpogas nospiešanas, tiek izvadīta datu bāzes tabulu saišu diagramma, kas pilnīgi atbilst manis izstrādātā fiziskā modeļa realitāšu saišu diagrammai (skat. 17. un 11. att.). Tātad varu secināt, ka esmu realizējis nepieciešamas prasības datu bāzei un tagad var ķerties pie pašas informācijas sistēmas izstrādes.



1. Att. Datu bāzes *Kancelejas Preces* tabulu saišu diagramma.

# SECINĀJUMI

Tika izstrādāts laboratorijas darbs, kurā primārais uzdevums bija iepazīties ar datu bāzes modelēšanas koncepcijām un apgūt pirmos pamatus datu bāzes projektēšanā un izstrādāšana. Šajā laboratorijas darbā tika izmantots *SyBase PowerDesigner* modulēšanas CASE rīks, kas pats par sevī sniedza saprašanu par CASE rīku nozīmi, ka arī ļāva mainīt pierastus uzskatus par datu bāzes projektēšanas būtību.

Uzdevums bija interesants, tanī bija nepieciešamība formulēt prasības datu bāzei un patstāvīgi veikt uzdevuma nostādnes analīzi. Tādējādi ļaujot studentam iegūt pirmās iemaņas izstrādātāja un sistēmas analītiķa lomā, kas turpmāk noderēs strādājot ar klientiem.

Par būtisku labumu uzskatu, ka šīs laboratorijas darbs ļāva izmainīt pierastus priekšstatus par datu bāzes projektēšanas būtību, t.i., saišu definēšanu realitātēm - biju uzskatījis, ka saitēm starp tabulām pārsvara tiek definētas *viens pret daudziem* vai *daudzi pret vienu* attieksmes.

Par neērtību var uzskatīt, ka *PowerDesigner* CASE rīkā, veidojot konceptuālu moduli realitātēm nav nepieciešams definēt atribūtus citu tabulu saistīšanai, t.i., lai veiktu tabulu sasaistīšanu ir jāvelk saite no unikāla identifikatora līdz tabulai, kuru es vēlos piesaistīt, turpmāk, izveidojot fizisko modeli, šai piesaistītājai tabulai pašai tiek izveidots šāds unikāls atribūts. Par otru būtisku neērtību var uzskatīt to, ka tabulām, kuras ir sasaistītas ar *daudzi pret daudziem* saiti, izveidojoties starp tabulai, tā satur abu tabulu unikālos identifikatorus, bet pašas šinīs tabulas nav citas tabulas unikāla identifikatora, kas mazliet apgrūtina tādas saites loģisko saprašanu.

Uzskatu, ka izveidoto datu bāzi var uzlabot papildinot tabulas vēl ar papildus atribūtiem, lai nodrošinātu precīzākas informācijas izgūšanu no datu bāzes un atvieglotu SQL vaicājuma pieprasījumus.

# LITERATŪRAS SARAKSTS

1. Viktorija Vinogradova. Lekciju konspekti studiju priekšmetā „Datu bāzes vadības sistēma” – Rīga, RTU, 2007.
2. Wikipedia. <http://en.wikipedia.org/wiki/PowerDesigner>
3. SyBase books. <http://sybooks.sybase.com/nav/detail.do?docset=1415>

# PIELIKUMS

## Ģenerētais SQL skripts

/\*==============================================================\*/

/\* DBMS name: Microsoft Access 2000 \*/

/\* Created on: 19.02.2008 20:33:47 \*/

/\*==============================================================\*/

alter table Pasutijumi

drop constraint "FK\_PASUTIJU\_VEIC PASU\_DARBINIE";

alter table Preces

drop constraint FK\_PRECES\_IETILPST\_KATEGORI;

alter table piegada

drop constraint FK\_PIEGADA\_PIEGADA\_PRECES;

alter table piegada

drop constraint FK\_PIEGADA\_PIEGADA2\_PIEGADAT;

alter table satur

drop constraint FK\_SATUR\_SATUR\_PASUTIJU;

alter table satur

drop constraint FK\_SATUR\_SATUR2\_PRECES;

drop table Darbinieki;

drop table Kategorijas;

drop table Pasutijumi;

drop table Piegadataji;

drop table Preces;

drop table piegada;

drop table satur;

/\*==============================================================\*/

/\* Table: Darbinieki \*/

/\*==============================================================\*/

create table Darbinieki

(

DarbID NUMERIC not null,

LietotVards TEXT(25) not null,

Parole TEXT(25) not null,

Nodala TEXT(50) not null,

Menedzeris YESNO not null,

Vards TEXT(25) not null,

Uzvards TEXT(25) not null,

constraint PK\_DARBINIEKI primary key (DarbID)

);

/\*==============================================================\*/

/\* Table: Kategorijas \*/

/\*==============================================================\*/

create table Kategorijas

(

KatID NUMERIC not null,

Nosaukums TEXT(25) not null,

Apraksts TEXT(50),

constraint PK\_KATEGORIJAS primary key (KatID)

);

/\*==============================================================\*/

/\* Table: Pasutijumi \*/

/\*==============================================================\*/

create table Pasutijumi

(

PasutID NUMERIC not null,

DarbID NUMERIC not null,

PasutDatums DATE not null,

Status YESNO not null,

constraint PK\_PASUTIJUMI primary key (PasutID)

);

/\*==============================================================\*/

/\* Table: Piegadataji \*/

/\*==============================================================\*/

create table Piegadataji

(

PiegID NUMERIC not null,

Nosaukums TEXT(25) not null,

constraint PK\_PIEGADATAJI primary key (PiegID)

);

/\*==============================================================\*/

/\* Table: Preces \*/

/\*==============================================================\*/

create table Preces

(

PrecesID NUMERIC not null,

KatID NUMERIC not null,

Nosaukums TEXT(25) not null,

Apraksts TEXT(50),

CenaParVien FLOAT not null,

constraint PK\_PRECES primary key (PrecesID)

);

/\*==============================================================\*/

/\* Table: piegada \*/

/\*==============================================================\*/

create table piegada

(

PrecesID NUMERIC not null,

PiegID NUMERIC not null,

constraint PK\_PIEGADA primary key (PrecesID, PiegID)

);

/\*==============================================================\*/

/\* Table: satur \*/

/\*==============================================================\*/

create table satur

(

PasutID NUMERIC not null,

PrecesID NUMERIC not null,

Daudzums INTEGER not null,

PiegDatums DATE not null,

constraint PK\_SATUR primary key (PasutID, PrecesID)

);

alter table Pasutijumi

add constraint "FK\_PASUTIJU\_VEIC PASU\_DARBINIE" foreign key (DarbID)

references Darbinieki (DarbID);

alter table Preces

add constraint FK\_PRECES\_IETILPST\_KATEGORI foreign key (KatID)

references Kategorijas (KatID);

alter table piegada

add constraint FK\_PIEGADA\_PIEGADA\_PRECES foreign key (PrecesID)

references Preces (PrecesID);

alter table piegada

add constraint FK\_PIEGADA\_PIEGADA2\_PIEGADAT foreign key (PiegID)

references Piegadataji (PiegID);

alter table satur

add constraint FK\_SATUR\_SATUR\_PASUTIJU foreign key (PasutID)

references Pasutijumi (PasutID);

alter table satur

add constraint FK\_SATUR\_SATUR2\_PRECES foreign key (PrecesID)

references Preces (PrecesID);