3. harjutustund (18.09)

Kava

Denormaliseerimine (eelmise tunni jätk)

o Dimensioonitabeli lähteandmete päringu koostamine
Faktitabeli koostamine

Iseseisvad harjutused

Harjutused

Harjutus 3.1 – dimensioonitabeli koostamine

Soovime koostada toote (kaup) dimensiooni, mille abil saaks analüüsida müügitehinguid kategooriate, alamkategooriate ja mudelite lõikes.

A. Koostada SQL päring, mis kogub kokku dimensiooni moodustamiseks vajalikud lähteandmed. Salvestada päring oma andmebaasis vaatesse **vwDimProductETL_Nimi_Perekonnanimi**
   - Vihjeks: tooteandmed leiab tabelist **AdventureWorks2012.Production.Product**, kategooriad ja alamkategooriad vastavalt **Production.ProductSubcategory** ja **Production.ProductCategory**.
   - Seoste tuvstamiseks kasutada andmebaasi diagrammi.

Lahenduskäik:

Tootedimensioonis peaks kindlasti olema tootekood ning nimetus. Denormaliseerimise tulemusena soovime sinna lisada ka kategooriate nimetuse ning mudelinime.

Alustame päringust tootetabelile. Märgime, et tabelile on antud alias **P**, kuna sama nimega väli (**Name**) asub mitmes tabelis, seega tuleb väljad kvalifitseerida.

```
SELECT P.ProductID, P.Name
FROM AdventureWorks.Production.Product AS P
```

Andmebaasi diagrammilt leiame, et mudelite kirjeldused on tabelis Production.ProductModel. Ühendame tabelid *LEFT OUTER JOIN*-iga, kuna lähteandmebaasis leidub tooteid, millel võib olla mudel määramata.

```
SELECT P.ProductID, P.Name, M.Name
FROM AdventureWorks.Production.Product AS P
      LEFT JOIN AdventureWorks.Production.ProductModel M
            ON M.ProductModelID = P.ProductModelID
```

Oluline on märkida ka, et *FULL OUTER JOIN* ei oma siinkohal mõtet – lähteandmebaasi reeglitest on teada, et müügitellimusi ei saa luua ilma toodet määramata, seega ei saa ühtlasi müüa "abstraktset tootemudelit", millega pole seotud ühtegi konkreetset toodet.

Lisame päringusse alamkategooria...

```
SELECT P.ProductID, P.Name, M.Name, SC.Name
FROM AdventureWorks.Production.Product AS P
      LEFT JOIN AdventureWorks.Production.ProductModel M
            ON M.ProductModelID = P.ProductModelID
      LEFT JOIN AdventureWorks.Production.ProductSubcategory SC
            ON SC.ProductSubcategoryID = P.ProductSubcategoryID
```

... ja peakategooria:

```
SELECT P.ProductID, P.Name, M.Name, SC.Name, C.Name
FROM AdventureWorks.Production.Product AS P
      LEFT JOIN AdventureWorks.Production.ProductModel M
            ON M.ProductModelID = P.ProductModelID
      LEFT JOIN AdventureWorks.Production.ProductSubcategory SC
            ON SC.ProductSubcategoryID = P.ProductSubcategoryID
      LEFT JOIN AdventureWorks.Production.ProductCategory C
            ON C.ProductCategoryID = SC.ProductCategoryID
```

Enne päringu vaatesse salvestamist peame andma igale väljale unikaalse nime. Kuna loome andmeaida tarbeks dimensioonitabelit, peaks väljade nimed andma aimu nende tähendusest (nn isedokumenteeriv e. *self-documenting* andmebaasiskeem).

```
SELECT P.ProductID AS ProductCode
      , P.Name AS ProductName
      , M.Name AS ModelName
      , C.Name AS CategoryName
      , SC.Name AS SubCategoryName
FROM AdventureWorks.Production.Product AS P
      LEFT JOIN AdventureWorks.Production.ProductModel M
            ON M.ProductModelID = P.ProductModelID
      LEFT JOIN AdventureWorks.Production.ProductSubcategory SC
            ON SC.ProductSubcategoryID = P.ProductSubcategoryID
      LEFT JOIN AdventureWorks.Production.ProductCategory C
            ON C.ProductCategoryID = SC.ProductCategoryID
```

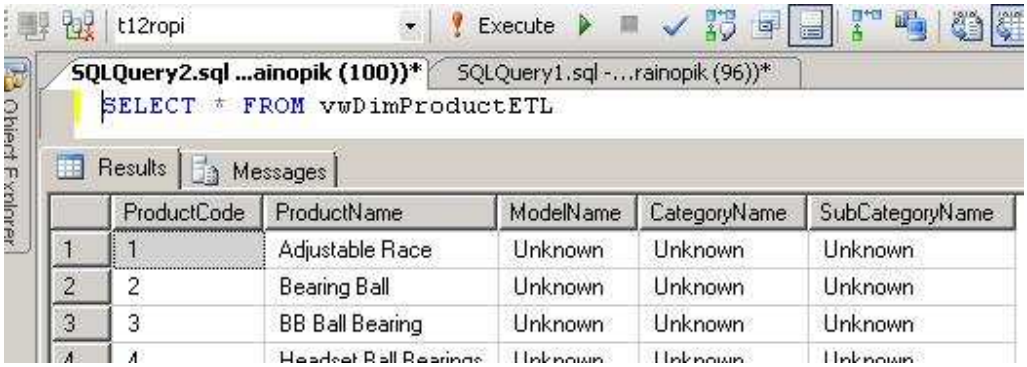
Andmeaita lisatud andmed peaksid olema kontrollitud ja puhastatud. Käivitades päringu, näeme, et paljudel toodetel on mudeli- ja kategooriate nimetused määramata. Dimensiooni koostamisel tuleb mõelda, mida tähendab ärilises vaates *ModelName = NULL*: kas tegemist on poolikult täidetud kaubakaardiga ERP-süsteemis, antud tootel ei olegi võimalik mudelit defineerida (nt. mutrid jas seibid) ning teisendada NULL väärtus ärikasutajatele ühtselt arusaadavaks, eelnevalt kokkulepitud tekstiks. Antud ettevõttes kehtib esimene juht – jalgrataste varuosadel (seibid, poldid) ei ole võimalik mudelit defineerida , mistõttu markeerime NULL väärtused tekstiga *Unknown*.

```
SELECT P.ProductID AS ProductCode
      , P.Name AS ProductName
      , COALESCE(M.Name, 'Unknown') AS ModelName
      , COALESCE(C.Name, 'Unknown') AS CategoryName
      , COALESCE(SC.Name, 'Unknown') AS SubCategoryName
FROM AdventureWorks.Production.Product AS P
      LEFT JOIN AdventureWorks.Production.ProductModel M
            ON M.ProductModelID = P.ProductModelID
      LEFT JOIN AdventureWorks.Production.ProductSubcategory SC
            ON SC.ProductSubcategoryID = P.ProductSubcategoryID
      LEFT JOIN AdventureWorks.Production.ProductCategory C
            ON C.ProductCategoryID = SC.ProductCategoryID
```

Kui eelnev on valmis, salvestame päringu vaateks (NB! Kontrollige, kas töötate ikka AdventureWorks2012 admebaasis)...

```
CREATE VIEW vwDimProductETL_Nimi_Perekonnanimi AS
SELECT ...
```

... ning testime, kas vaade toimib:



B. Luua andmebaasis tabel **DimProduct_Nimi_Perekonnanimi**, mis hakkab hoidma dimensiooni andmeid.

Lahenduskäik:
Käivitame SSMS objektipuust oma andmebaasi peal *Tables* *New Table ...*

mis avab tabelidisaineri. Defineerime disainerivaates dimensioonitabeli veerud:

IDBI\SQLDIMENSIONA....DimProduct_Eduard* | SQLQuery6.sql -

Column Name	Data Type	Allow Nulls
ProductCode	nchar(10)	☐
ProductName	nchar(10)	☑
ModelName	nchar(10)	☑
CategoryName	nchar(10)	☑
SubCategoryName	nchar(10)	☑
		☐

Andmetüüpide määramiseks peame otsima lähteandmebaasist (**AdventureWorks2012**) väljade pikkused. Seda on kõige mugavam teha, kasutades meta-andmete vaadet INFORMATION_SCHEMA:

```
SELECT *
FROM AdventureWorks.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = 'Production' AND TABLE_NAME = 'Product'
```

Defineerime igale veerule andmetüübid:

IDBI\SQLDIMENSIONA...2012 - dbo.Table_1* | SQLQuery6.sql - I

Column Name	Data Type	Allow Nulls
ProductCode	int	☐
ProductName	nvarchar(50)	☑
ModelName	nvarchar(50)	☑
CategoryName	nvarchar(50)	☑
SubCategoryName	nvarchar(50)	☑
		☐

Määrame primaarvõtmeks **ProductCode** (väljal parem hiir -> *Set Primary Key*) ning keelame nimetuste veergudes NULL-väärtused, kuna eelneva andmete puhastamise tulemusena ei saa ega tohi neid tabelisse tekkida:

IDBI\SQLDIMENSIONA...2012 - dbo.Table_1* | SQLQuery6.sql - I

Column Name	Data Type	Allow Nulls
ProductCode	int	☐
ProductName	nvarchar(50)	☑
ModelName	nvarchar(50)	☑
CategoryName	nvarchar(50)	☑
SubCategoryName	nvarchar(50)	☑
		☐

Salvestame tabeli nimega **DimProduct_Nimi_Perekonnanimi**

C. Täita dimensioonitabel käsitsi:

```
DELETE FROM DimProduct
INSERT DimProduct SELECT /* väljade nimed siia*/ FROM vwDimProductETL
```

Lahenduskäik:

Loome oma esimese "ETL programmi", mis kustutab dimensioonitabelist vanad andmed ning täidab selle värsketega. Laisa andmeaida koostaja päring näeks välja järgmine:

```
DELETE FROM DimProduct;
INSERT INTO DimProduct SELECT * FROM vwDimProductETL;
```

Selle kasutamise eelduseks on, et väljade järjekord langeb vaates **vwDimProductETL_Nimi_Perekonnanimi** ja sihttabelis **DimProduct_Nimi_Perekonnanimi** üksüheselt kokku. Kui arenduse käigus peaks dimensioonitabelisse veerge lisanduma, siis päring enam ei käivitu.

Korrektses lahenduses on väljanimed ilmutatud kujul loetletud:

```
DELETE FROM DimProduct;

INSERT INTO DimProduct(ProductCode, ProductName, ModelName, CategoryName,
SubCategoryName)
SELECT ProductCode, ProductName, ModelName, CategoryName, SubCategoryName
FROM vwDimProductETL;
```

Kui päringu käivitamisel tekkis vigu, siis reeglina lähteandmete vaade ei vasta dimensioonitabeli nõuetele:

- Tootekoodi unikaalsus ei ole tagatud (*Violation of PRIMARY KEY constraint 'PK_DimProduct'. Cannot insert duplicate key in object 'dbo.DimProduct'…)*

  Lahendus: Kontrollida päringuga, kas dimensioonitabelis on primaarvõti (nt. **ProductID**) unikaalne ning eemaldada topelt-kirjed. Tõenäoliselt on põhjuseks vigaselt kirjutatud *JOIN*-tingimus.

```
SELECT ProductCode, COUNT(*)
FROM vwDimProductETL
GROUP BY ProductCode
HAVING COUNT(*) > 1
```

- Lähteandmete vaates leidub puhastamata kirjeid (*Cannot insert the value NULL into column 'SubCategoryName', table 't12ropi.dbo.DimProduct'; column does not allow nulls. INSERT fails.*)

  Lahendus: otsida NULL-väärtustega kirjed, tuvastada puuduvate andmete tekkepõhjus ning lisada lähteandmete vaatesse teisendusreegel, mis elimineerib NULL-väärtused.

```
SELECT * FROM vwDimProductETL WHERE SubCategoryName IS NULL
```

D. Kontrollida, kas dimensioonitabelis leiduvad kõik tooted. Kontrollimiseks saab kasutada erinevaid võtteid:

- Valida pisteliselt lähtetabelist (**AdventureWorks2012.Production.Product**) paar tootekoodi ning kontrollida nende olemasolu dimensioonitabelis.
- Loendada kirjed lähte- ja sihttabelis

```
SELECT COUNT(*) FROM AdventureWorks2012.Production.Product
```

- Arvutada igale kirjele CHECKSUM() funktsiooniga kontrollsumma ning liita need CHECKSUM_AGG() funktsiooniga kokku:

```
SELECT CHECKSUM_AGG( CHECKSUM( P.ProductID, P.Name) )
FROM AdventureWorks2012.Production.Product P
```

Harjutus 3.2 – faktitabeli koostamine

Soovime analüüsida kaupade müüki järgnevalt:

- Kui suur on käive erinevate tootekategooriate lõikes, millised on kõige suurema läbimüügika kategooriad?
- Millised on kõige kasumlikumad tooted ning milliste toodete müük (või ebasoodsad kampaaniad) tuleks ära lõpetada?

Küsimustele vastuse leidmiseks loome müügiprotsessi kirjeldava faktitabeli:

A. Koostada SQL päring, mis paneb kokku müügitellimuste päiste (**Sales.SalesOrderHeader**) ning ridade (**Sales.SalesOrderDetail**) andmed ning tagastab tulemuses:
   - Dimensioonidele viitavad välisvõtmed: tootekood, kliendikood, kuupäev
   - Arvulised mõõdikud: tellimuste arv, müüdud kogus, käive, müügikulu (müük omahinnas e. kogus * omahind)

Lahenduskäik:

Müügifakti tabeli aluseks on müügitellimused ja müügitellimuste read. Koostame päringu, mis ühendab müügitellimuse päised ridadega ning kuvab tulemuses dimensioonidele viitavad välisvõtmed.

```
SELECT H.ShipDate AS SalesDateCode
      , H.CustomerID AS CustomerCode
      , D.ProductID AS ProductCode
FROM AdventureWorks.Sales.SalesOrderHeader H
      INNER JOIN AdventureWorks2012.Sales.SalesOrderDetail D
            ON D.SalesOrderID = H.SalesOrderID
```

NB! Tehingutabelites on tavaliselt märkimisväärne kogus kirjeid, mistõttu *SELECT ** päring võib mõjuda andmbaasile koormavalt. Kui hakkate sarnast tegevust katsetama mõne "päris" andmbaasi peal, lisage kindlasti päringu kirjutamise ajaks kirjete arvu piirav filter (tavaliselt *LIMIT n*, MSSQLis *TOP n*). Filtri eemaldame siis, kui päring valmis.

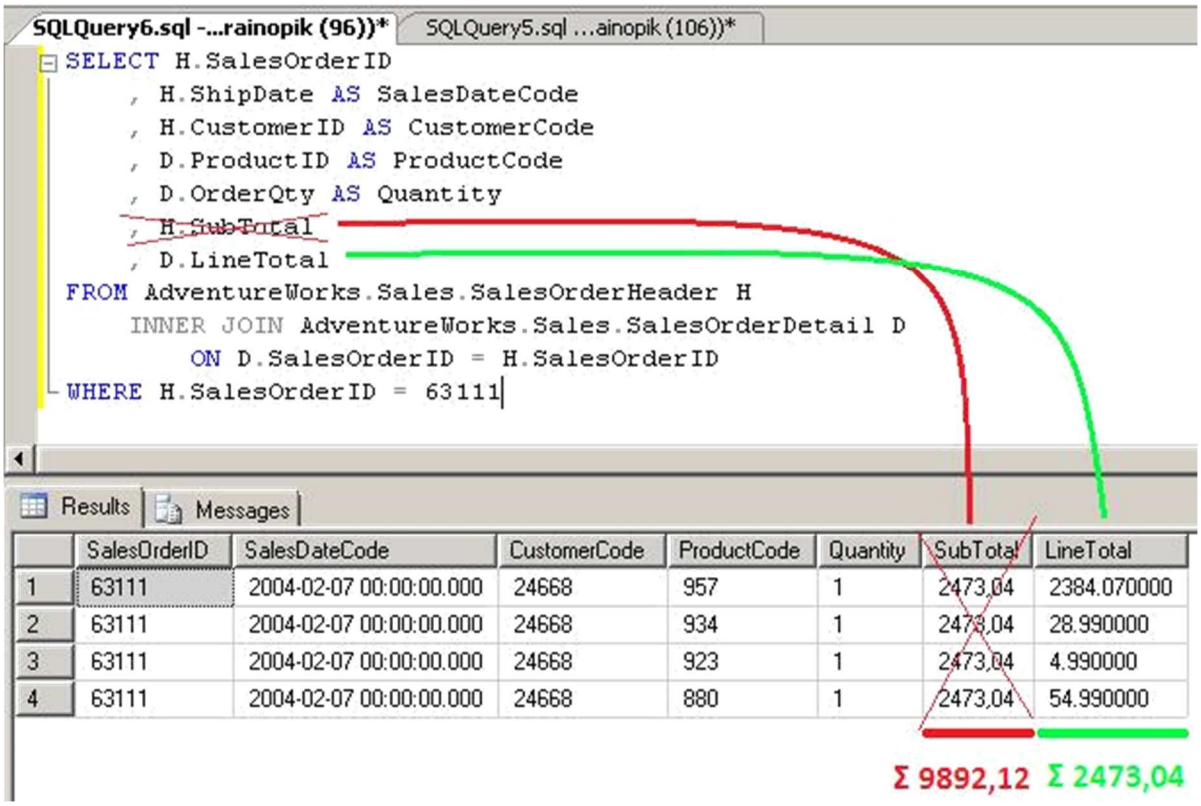
```
SELECT TOP 100 /* säästame dba head tuju ja närvirakke :) */
        H.ShipDate AS SalesDateCode
        ...
```

Hakkame päringusse mõõdikuid lisama. Kogusega on lihtne, kuna selle saab müügitellimuse ridadelt:

```
SELECT H.ShipDate AS SalesDateCode
      , H.CustomerID AS CustomerCode
      , D.ProductID AS ProductCode
      , D.OrderQty AS Quantity
FROM AdventureWorks.Sales.SalesOrderHeader H
      INNER JOIN AdventureWorks.Sales.SalesOrderDetail D
            ON D.SalesOrderID = H.SalesOrderID
```

Müügikäibe saamiseks tuleb lugeda andmebaasi dokumentatsiooni. Leiame, et iga tellimuse rea summa on: **SalesOrderDetail.LineTotal = OrderQty * UnitPrice** ning reasummad kokku peaksid andma tellimuse summa ilma maksudeta: **SalesOrderHeader.SubTotal = SUM(SalesOrderDetail.LineTotal)**.

Oluline on leida see väärtus ridadelt (**SalesOrderDetail.LineTotal**), sest kui ühel tellimusel on nt. 4 kirjet, summeeritaks päisest võetud kogusumma (**SalesOrderHeader.SubTotal**) neljakordselt.



Faktitabeli päring, mis sisaldab ka müügikäivet, näeb välja järgmine.

```
SELECT H.ShipDate AS SalesDateCode
      , H.CustomerID AS CustomerCode
      , D.ProductID AS ProductCode
      , D.OrderQty AS Quantity
      , D.LineTotal AS SalesAmount
FROM AdventureWorks2012.Sales.SalesOrderHeader H
      INNER JOIN AdventureWorks2012.Sales.SalesOrderDetail D
            ON D.SalesOrderID = H.SalesOrderID
```

Kaubakulu (müük omahinnas) leidmine on keerulisem, kuna lähteandmebaasi müügitellimustel ega –ridadel ei salvestata müügi toimumise hetkel kehtinud omahinda.

Andmebaasidiagrammi uurimisel leiame, et tootetabeli (**Production.Product**) väljalt **StandardCost** leiab omahinna. Kiire lahendusena saame päringu külge ühendada tootetabeli ning arvutada kaubakulu kui müüdud kogus * tootekaardi omahind:

```
SELECT H.ShipDate AS SalesDateCode
      , H.CustomerID AS CustomerCode
      , D.ProductID AS ProductCode
      , D.OrderQty AS Quantity
      , D.LineTotal AS SalesAmount
      , D.OrderQty * P.StandardCost AS CostAmount
FROM AdventureWorks2012.Sales.SalesOrderHeader H
      INNER JOIN AdventureWorks2012.Sales.SalesOrderDetail D
            ON D.SalesOrderID = H.SalesOrderID
      INNER JOIN AdventureWorks2012.Production.Product P
            ON D.ProductID = P.ProductID
```

On ilmne, et pakutud lahendus küll töötab, aga sellisel arvutusreeglil on omad puudused:

- Toote ühiku omahind kipub ajas muutuma, tootekaardile märgitud omahind tähistab aga hetkel kehtivat väärtust. Kui me arvutame 2 aastat vanadele tehingutele omahinna täna, saaksime kaubakuluks nt. $200, samas käivitades seda päringut aasta tagasi, oli tulemuseks $150.

Probleemi lahendaks see, kui hakkame andmeaita igapäevaselt salvestama omahinna muutuste ajalugu.

- Omahinnaarvutus tuleb eelnevalt kokku leppida äripoole esindajatega. See tähendab, et kui käivitame oma faktitabeli peal päringu ...

```
SELECT SUM(CostAmount) FROM FactSales
WHERE SalesDateCode >= '2001-07-01' AND SalesDateCode < '2001-08-01'
```

... peab tulemus andma sama numbri, mis on pearaamatus kaupade müügikulu konto vastava perioodi käive.

Korrektse lahenduse saamiseks tuleb otsida lähteandmebaasist tabel, mis kajastab lao koguselist ja väärtuselist liikumist. AdventureWorks baasis on selliseks tabeliks **Production.TransactionHistory**, mis sisaldab viimase aasta laokandeid ning **TransactionHistoryArchive**, kuhu on arhiveeritud varasemate aastate kanded. Andmebaasi dokumentatsioonist leiame, kannete tabelisse luuakse iga **SalesOrderDetail** kohta eraldi kirje.

Müügitellimuste laokanded saame kätte järgmise päringuga:

```
SELECT ReferenceOrderID, ProductID, ActualCost
FROM AdventureWorks2012.Production.TransactionHistory
WHERE TransactionType = 'S'
UNION
SELECT ReferenceOrderID, ProductID, ActualCost
FROM AdventureWorks2012.Production.TransactionHistoryArchive
WHERE TransactionType = 'S'
```

Täiendame oma müügifaktide päringut, lisades sellele laokandeid tagastava alampäringu:

```
SELECT H.ShipDate AS SalesDateCode
      , H.CustomerID AS CustomerCode
      , D.ProductID AS ProductCode
      , D.OrderQty AS Quantity
      , D.LineTotal AS SalesAmount
      , T.ActualCost AS CostAmount
FROM AdventureWorks.Sales.SalesOrderHeader H
      INNER JOIN AdventureWorks.Sales.SalesOrderDetail D
            ON D.SalesOrderID = H.SalesOrderID
      LEFT JOIN (
            SELECT ReferenceOrderID, ProductID, ActualCost
            FROM AdventureWorks.Production.TransactionHistory
            WHERE TransactionType = 'S'
            UNION
            SELECT ReferenceOrderID, ProductID, ActualCost
            FROM AdventureWorks.Production.TransactionHistoryArchive
            WHERE TransactionType = 'S'
      ) T ON T.ReferenceOrderID = D.SalesOrderID
            AND T.ProductID = D.ProductID
```

Meenutame dimensionaalse modelleerimise loengust, et faktitabelis peaks iga mõõtmise, st. dimensioonide kombinatsiooni kohta olema üks kirje. Grupeerime kirjed välisvõtmete järgi kokku ja summeerime mõõdikud:

```
SELECT H.ShipDate AS SalesDateCode
      , H.CustomerID AS CustomerCode
      , D.ProductID AS ProductCode
      , SUM(D.OrderQty) AS Quantity
      , SUM(D.LineTotal) AS SalesAmount
      , SUM(T.ActualCost) AS CostAmount
FROM AdventureWorks.Sales.SalesOrderHeader H
      INNER JOIN AdventureWorks.Sales.SalesOrderDetail D
            ON D.SalesOrderID = H.SalesOrderID
      LEFT JOIN (
            SELECT ReferenceOrderID, ProductID, ActualCost
```

```
            FROM AdventureWorks.Production.TransactionHistory
            WHERE TransactionType = 'S'
            UNION
            SELECT ReferenceOrderID, ProductID, ActualCost
            FROM AdventureWorks.Production.TransactionHistoryArchive
            WHERE TransactionType = 'S'
      ) T ON T.ReferenceOrderID = D.SalesOrderID
            AND T.ProductID = D.ProductID
GROUP BY H.ShipDate, H.CustomerID, D.ProductID
```

Nüüdseks on faktide päring valmis – salvestame selle vaateks:

```
CREATE VIEW vwFactSalesETL_Nimi_PerekonnanimiS AS
SELECT ...
```

B. Luua tabel **FactSales_Nimi_Perekonnanimi**, mis hakkab andmeid talletama.

Lahenduskäik:

Tabeli loomine on sarnane eelnevaga – kasutame INFORMATION_SCHEMA vaateid, et tuvastada lähtetabelite andmetüübid. Märgime, et faktitabelis on primaarvõtmeks dimensioonide kombinatsioon – kasutame nn. komposiitvõtit.

ELBRUS.t12ropi - dbo.Table_1*

Column Name	Data Type	Allow Nulls
SalesDateCode	datetime	☐
CustomerCode	int	☐
ProductCode	int	☐
Quantity	int	☑
SalesAmount	numeric(38, 6)	☑
CostAmount	numeric(38, 6)	☑

Salvestame tabeli nimega **FactSales Nimi_Perekonnanimi**

☐ Täita faktitabel järgneva skriptiga:

```
DELETE FROM FactSales Nimi_Perekonnanimi;
INSERT FactSales_Nimi_Perekonnanimi SELECT /* väljade nimed siia*/ FROM
vwFactSalesETL_Nimi_Perekonnanimi;
```

☐ Kontrollida, kas faktitabelis on tehingud õigesti kajastatud:

- Kas käive e. tellimuste summa läheb kokku lähteandmebaasiga?

```
SELECT SUM(SalesAmount) FROM vwFactSalesETL _Nimi_Perekonnanimi
```

NB! Tuleb eristada tellimuse päises ja ridadel toodud summasid. Tellimuses võib olla summasid, mis ei ole kaupadega kirjeldatud (nt. lisakulud)

```
SELECT SUM(D.LineTotal)  FROM AdventureWorks2012.Sales.SalesOrderDetail D
```

Tabelite ühendamisel jälgida, et päise andmeid ei summeeritaks mitmekordselt:

```
SELECT SUM(D.LineTotal), SUM(H.SubTotal) FROM
AdventureWorks.Sales.SalesOrderHeader H
      INNER JOIN AdventureWorks2012.Sales.SalesOrderDetail
          D ON H.SalesOrderID = D.SalesOrderID
```

Iseseisev töö

Harjutus 3.3 – müügifakti täiendamine (1 p)

Lisada faktitabelisse mõõdik: tellimuste arv. Kontrollida, kas tellimuste koguhulk läheb kokku lähteandmebaasiga?

```
SELECT SUM(OrderCount) FROM vwFactSalesETL
SELECT COUNT(*) FROM AdventureWorks2012.Sales.SalesOrderHeader
```

Harjutus 3.4 – andmeaida kasutamine (1p)

Koostada SQL päring, mis leiab H2 alguses toodud kahele küsimusele vastused.

Harjutus 3.5 – dimensiooni ja fakti moodustamine (1p)

Soovime analüüsida kaupade sisseostmist järgnevates aspektides:

- Kui suur on tarnija tarnekindlus –mitu protsenti tellitud kaupadest jõuab reaalselt kohale?
- Kui suur on praagi osakaal saabunud saadetistes – mitu protsenti toodetest on DOA (*dead on arrival*)
- Kas saamata kaupade osakaal ning praagi osakaal sõltub tarnijast (halb partner)? Toote mudelist (keeruline toode / halb kvaliteedikontroll)? Üksikust tootest?

Koostada dimensionaalne andmemudel, mille pealt on võimalik nendele küsimustele vastuseid otsida:

- Luua hankija dimensioonitabel **DimVendor_Nimi_Perekonnanimi**, mis koondab endas hankija üldandmed (nimi, staatused), aadressiandmed (tänav, linn, regioon, riik) ning kontaktisik.
- Luua ostuprotsessi kirjeldav faktitabel **FactPurchases**, milles on:
  - Kolm dimensioonidele viitavat välisvõtit: toode, hankija ning kuupäev
  - Mõõdikud: tellitud kogus, saadud kogus, saadud praak, ostu summa.

Tulemuses peaks olema:

- SQL vaated **vwDimVendorETL_Nimi_Perekonnanimi**, mille abil saab dimensioonitabelit täita ning **vwFactPurchasesETL_Nimi_Perekonnanimi**, mis annab andmed faktitabeli jaoks.
- Andmeaida tabelid **DimVendor** ja **FactPurchases** (tootedimensioon **DimProduct** on juba eelnevalt loodud).
- SQL skript, mille abil saab tabeleid täita (*DELETE FROM … ; INSERT INTO … SELECT FROM …*).

Mudeli testimiseks koostage päringud, mis leiavad vastused järgmistele küsimustele:

- Kui suur on tootemudelite lõikes praagi osakaal. Reastada tulemus selliselt, et kõige veaohtlikumad tootemudelid oleksid eespool.
- Kui suur on hankijate lõikes saamata kaupade osakaal. Reastada tulemus, et kõige problemaatilisemad tarnijad oleksid eespool.

Harjutus 3.6 – nuputamiseks

Kuidas kajastada müügifakti tabelis summasid, mis ei ole toodete peal kajastatud? Vt. tellimus nr. 50256 – kaupade summa on $86,52, aga tellimuse netosumma $93,73. Andmeaidas peaks müügifakt kajastama kogu müüki tervikuna.

PS: Harjutuste kontrollimiseks palun saatke emaili õpetajale.

Materjalid

- AdventureWorks Business Scenarios. http://msdn.microsoft.com/en-us/library/ms124825%28v=sql.100%29.aspx
- AdventureWorks Data Dictionary. http://msdn.microsoft.com/en-us/library/ms124438%28v=sql.100%29.aspx

Kõik viited on seisuga 17.09.2012