SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ivana Đurđević

ODRŽAVANJE SKLADIŠTA PODATAKA

Diplomski rad

Voditelj rada:
prof.dr.sc. Robert Manger

Zagreb, 2017.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Sadržaj **iii**

Uvod **1**

1 Skladišta podataka **3**
1.1 Povijesni pregled i osnovni pojmovi 3
1.2 Definicija i glavne značajke skladišta podataka 6
1.3 Usporedba OLTP i OLAP sustava 8
1.4 Dva pristupa skladištenju 10
1.5 Višedimenzionalni model 12
1.6 Alati za fizičku realizaciju skladišta podataka 15
1.7 Nadzor i održavanje skladišta podataka 16

2 ETL proces **19**
2.1 Dohvaćanje podataka 20
2.2 Transformacija podataka 21
2.3 Učitavanje podataka u skladište 22
2.4 Koristiti ETL ili ELT? 23
2.5 ETL alati 24

3 Studijski primjer **29**
3.1 Shema transakcijske baze 29
3.2 Implementacija dimenzionalnog modela 30
3.3 Implementacija ETL procesa 32
3.4 Primjer upita nad skladištem podataka 38

Zaključak **41**

Bibliografija **43**

Uvod

Razvojem informacijskih tehnologija i mogućnosti spremanja velikih količina podataka na disk javila se potreba za učinkovitim, strukturiranim načinima spremanja podataka tako da bi se mogli koristiti za analizu. Mnoge tvrtke koriste sustav skladištenja podataka kako bi poboljšale svoje poslovne procese i otkrile neke uzorke u ponašanju svojih korisnika.
Skladištenje podataka je vrlo opširna tema, a mi ćemo se ovdje dotaknuti samo nekih osnovnih dijelova, bez ulaženja u velike detalje. Rad je podijeljen u tri poglavlja. U prvom je dan povijesni pregled nastanka i razvoja skladišta podataka. Zatim su navedeni osnovni pojmovi i glavne značajke skladišta. Nakon toga slijedi usporedba dva različita pristupa modeliranju skladišta podataka - Inmonovog i Kimballovog. Kimballov koristi višedimenzionalni model koji je detaljnije opisan u sljedećem potpoglavlju. Zatim su navedeni neki alati za fizičku realizaciju skladišta podataka. Na kraju je opisan način održavanja skladišta podataka nadziranjem nekih dijelova tako da bi se mogle optimizirati performanse.
Drugo poglavlje je posvećeno ETL procesu - najvažnijem dijelu pri oblikovanju skladišta podataka. Opisani su dijelovi ETL procesa - dohvaćanje (extract) podataka iz izvorišnih sustava, njihova transformacija (transform) i učitavanje (load) u skladište podataka. ETL proces možemo provesti koristeći neki od dostupnih komercijalnih ili open-source alata ili pisanjem vlastitih skripti. Na početku povijesnog razvoja skladištenja podataka koristio se ETL način integracije sustava (podaci se transformiraju prije učitavanja u skladište podataka). Danas se sve više koristi ELT pristup, tj. podaci se prvo učitaju u skladište, a zatim se tamo transformiraju.
U zadnjem poglavlju je dan studijski primjer oblikovanja skladišta podataka za maloprodajnu trgovinu. ETL proces je napravljen korištenjem open-source alata Talend Open Studio for Data Integration.

Poglavlje 1

Skladišta podataka

1.1 Povijesni pregled i osnovni pojmovi

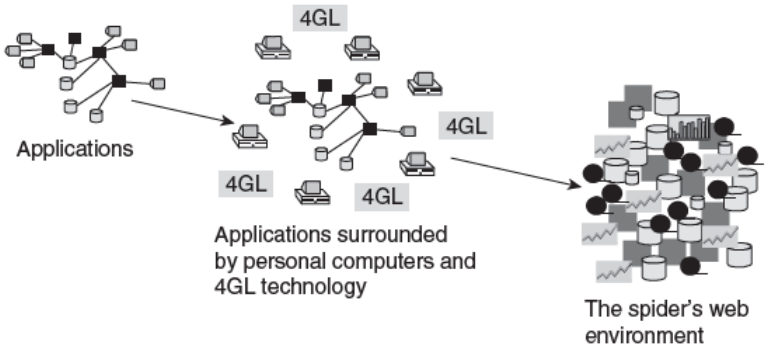
Bill Inmon, koji je poznat kao **"otac skladištenja podataka"**, u prvom poglavlju knjige *DW 2.0: The Architecture for the Next Generation of Data Warehousing* [14] navodi kratku povijest i razloge nastanka skladišta podataka.
U početku su za pohranu podataka postojali samo jednostavni mehanizmi koji su bili skupi i vrlo limitiranih mogućnosti poput bušenih kartica i papirnih vrpca. Pojavom magnetske vrpce došlo je do velikog napretka pri pohrani podataka - moglo se na jeftin način zapisivati velike količine podataka koji su se kasnije mogli i mijenjati. Najznačajniji napredak bio je izum pohrane podataka na disk. Kod takvog načina pohrane podacima se može pristupati direktno, mogu se zapisivati, mijenjati i brisati te im se može pristupati masovno. Uz pohranu podataka na disk, pojavio se i softver za upravljanje bazom podataka (eng. data base management system - DBMS) koji omogućava upravljanje pohranjenim podacima. Nakon što je omogućen direktan pristup podacima, pojavile su se razne online aplikacije koje mogu pristupati podacima brzo i konzistentno, kao što su npr. bankomat, aplikacije koje koriste službenici na šalterima u bankama, aplikacije za nadzor proizvodnje i sl. Pojavom osobnih računala i 4GL[1]tehnologije u korporacijama, krajnji korisnici su dobili direktan pristup podacima i više nisu ovisili o IT odjelu. Sada, kada su krajnji korisnici imali slobodan pristup podacima, otkrili su da im za donošenje dobrih odluka treba i nešto više od samog pristupa podacima. Primijetili su i razne probleme s podacima:

- nepotpuni podaci su beskorisni
- ako su podaci pogrešni, to je i gore nego da ih uopće nema jer pogrešni podaci mogu navesti na krivi put pri donošenju odluka

[1] Ideja 4GL(fourth generation) tehnologije bila je da se programiranje i razvoj sustava toliko pojednostavi da bilo tko može sudjelovati u tome.

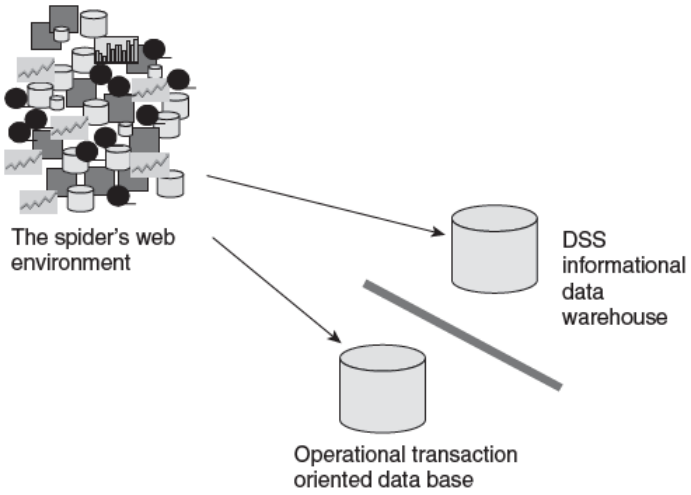
- podaci nisu uvijek pravovremeni, a trebali bi biti
- kad postoji više različitih verzija istih podataka, može doći do loših odluka ako se gleda pogrešna verzija
- podaci bez dokumentacije imaju upitnu vrijednost.

Na ovaj način došlo je do velikog kaosa u organizaciji informacija unutar korporacije i stvorilo se okruženje koje se naziva "paukova mreža".



Slika 1.1: Okruženje "paukova mreža" (slika preuzeta iz [14])

Ovakvo okruženje je postalo nemoguće za upotrebu i održavanje pa se javila potreba za razvijanjem drugačije arhitekture informacijskog sustava gdje je u središte postavljeno skladište podataka.
Ovo je bio opis razvoja sa stajališta tehnologije. Ali, možemo promotriti razloge razvoja skladišta i iz poslovne perspektive. Računala su se počela koristiti u poslovnom okruženju jer mogu obraditi velike količine podataka u malo vremena. U početku su se koristila za obračun plaća, izradu faktura, razne isplate i sl. Nakon toga je otkriveno da računala mogu spremati velike količine podataka pa su pojavom online baza podataka dobila ključno mjesto u transakcijskom poslovanju. Upotrebom online baza podataka otvorile su se nove mogućnosti poslovanja i procesiranje transakcija je bilo brže nego ikad prije. Tada su računala postala neizostavan dio poslovanja svake korporacije. Kasnije su se na temelju podataka koji cirkuliraju transakcijskim sustavima počele donositi poslovne odluke o daljnjim poslovnim potezima i strategijama. Tada je postalo jasno da postoje različiti tipovi baza podataka i iz okruženja "paukove mreže" je trebalo izgraditi novi informacijski sustav u kojem su jasno odvojene transakcijske baze od skladišta podataka.



Slika 1.2: Podjela na različite tipove baza podataka (slika preuzeta iz [14])

P. Williams u članku *A Short History of Data Warehousing* [18] daje vremenski pregled važnijih događaja u povijesti skladištenja podataka. U ranim 80-im godinama prošlog stoljeća javila se potreba za analizom dostupnih podataka. Otkriveno je da baze koje su efikasne u procesiranju transakcija nisu pogodne za potrebe analize i izrade izvještaja. Ono što je danas poznato pod pojmom skladištenja podataka počelo se razvijati krajem 80-ih godina iako je Inmon već krajem 70-ih smislio termin "skladište podataka" i počeo raspravljati o njihovim načelima. Početkom 90-ih godina, Inmon je osnovao svoju prvu tvrtku koja se bavila proizvodnjom alata za kreiranje i održavanje skladišta podataka te je 1992. godine izdao knjigu *Building the Data Warehouse* koja je i danas jedan od najvažnijih izvora informacija o oblikovanju skladišta podataka.
Drugo važno ime vezano uz povijesni razvoj skladišta podataka je **Ralph Kimball** koji je u svojoj knjizi *A Data Warehouse Toolkit*, izdanoj 1996., dao razne praktične primjere za analitičko modeliranje. Inmonov i Kimballov pristup oblikovanju skladišta podataka imaju brojne razlike o kojima će biti više riječi u potpoglavlju 1.4 i djelomično u potpoglavlju 1.5.

Osnovni pojmovi

Skladište podataka (eng. **data warehouse**) je baza podataka koja sadrži povijesne nepromjenjive podatke koji se prikupljaju i analiziraju kao pomoć pri donošenju poslovnih odluka. U sljedećem potpoglavlju ćemo dati dvije formalnije definicije skladišta podataka. **Skladištenje podataka** je proces integracije podataka o poslovanju neke organizacije u

jednu bazu podataka iz koje onda analitičari, menadžeri i donositelji odluka mogu raditi izvješća, postavljati upite i analizirati podatke.
Sustav skladištenja podataka je skup tehnologija i alata koji omogućavaju prikupljanje, integraciju i analizu podataka iz različitih izvora.
Tržnica podataka (eng. **data mart**) je skup podataka, dizajniran po istim principima kao i skladište podataka, koji poslužuje potrebe homogene grupe korisnika.
Meta podaci su podaci o podacima. Imaju puno važniju ulogu u okruženju skladišta podataka nego u operacijskom okruženju. Daju informacije o sadržaju skladišta podataka, kvaliteti podataka, korištenim algoritmima za sumiranje podataka, transformiranju i učitavanju podataka i dr.
ETL (Extract-Transform-Load) proces je proces u skladištenju podataka pomoću kojeg se podaci dohvaćaju iz izvorišnih sustava, transformiraju i učitavaju u skladište podataka.

1.2 Definicija i glavne značajke skladišta podataka

Najpoznatiju definiciju skladišta podataka dao je Bill Inmon u svojoj knjizi *Building the Data Warehouse* [13]. Prema toj definiciji, **skladište podataka** je:

- tematski orijentirani
- integrirani
- vremenski ovisni
- postojani

skup podataka koji služi kao potpora u procesu odlučivanja.
Marko Banek s Fakulteta elektrotehnike i računarstva je u svom magistarskom radu [11] dao detaljna objašnjenja za ova četiri svojstva. U nastavku navodimo kratka objašnjenja za svako od navedenih svojstava.
Tematska orijentiranost. Skladište podataka je tematski orijentirano jer je vezano uz aktivnosti od ključnog značaja za poduzeće. Transakcijska baza automatizira poslovne procese, dok je skladište orijentirano na one procese koji su zanimljivi za analizu.
Integriranost. Podaci u skladištu podataka se najčešće dobivaju iz više različitih izvora. U svakom izvorišnom sustavu mogu biti u različitom formatu, ali u skladištu moraju imati jedinstven format kako bi se omogućila njihova konzistentnost.
Vremenska različitost. Transakcijske baze sadrže podatke koji su točni u trenutku pristupa - kasnije se podaci mijenjaju ili brišu, a na njihovo mjesto dolaze nove vrijednosti. Za razliku od njih, skladište podataka predstavlja skup vremenskih snimaka, od kojih je svaki bio aktualan u određenom trenutku.

Postojanost. Nakon što su podaci uneseni u skladište, oni se ne mijenjaju ako je unos napravljen ispravno. Uz operaciju čitanja postoji samo operacija inicijalnog učitavanja, koja se obavlja periodički. Ne postoje druge operacije kao i kod transakcijskih baza, kao što su unošenje, izmjene, brisanja. Kako je podatke nemoguće mijenjati ili brisati, nestaje opasnost od nekonzistentnosti podataka. Time se gubi potreba za pohranom podataka u vidu relacijskih schema viskog stupnja normaliziranosti i otvara mogućnost korištenja drugačijeg, višedimenzionalnog, modela podataka.

Ralph Kimball u knjizi *The Data Warehouse ETL Toolkit* [16] daje drugu definiciju skladišta podataka:
Skladište podataka je sustav koji dohvaća, čisti, prilagođava i dostavlja izvorišne podatke u višedimenzionalno spremište podataka, a zatim pruža podršku i implementira mogućnost postavljanja upita i analize u svrhu donošenja poslovnih odluka.
Također, navodi neke najčešće zablude u vezi skladišta podataka. Skladište podataka nije:

- **Proizvod.** Skladište podataka se ne može kupiti. Sastoji se od mnogo različitih komponenti, a ne postoji jedinstven proizvod koji bi mogao ispuniti sve zadatke potrebne za izgradnju skladišta podataka (analiza sustava, manipulacija i čišćenje podataka, transport podataka, modeliranje skladišta, pristup podacima u skladištu).

- **Jezik.** Ne možemo naučiti kodirati skladište podataka kao što možemo naučiti neki programski jezik. Skladište se sastoji od više komponenti i svaka od njih zahtijeva znanje jednog ili više programskih jezika.

- **Projekt.** Pravilno razvijeno skladište podataka nije samo jedan projekt nego se sastoji od više manjih projekata (i svaki od tih projekata se može sastojati od više faza razvoja). Uglavnom se razvoj svake tržnice podataka smatra zasebnim projektom sa vlastitim budžetom i vremenom razvoja. Skladište podataka na razini cijelog poduzeća se razvija i raste završetkom svakog projekta za pojedinu tržnicu podataka.

- **Model podataka.** Model podataka sam za sebe ne čini skladište podataka. Mora postojati i neki ETL proces da bi se skladište napunilo podacima. Čak je i najbolje dizajnirani model podataka potpuno beskoristan ako se u njemu ne nalaze nikakvi podaci.

- **Kopija transakcijskog sustava.** Pogrešno je misliti da je za izgradnju skladišta podataka dovoljno samo kopirati oparacijsku bazu u zaseban sustav za izvještavanje. Treba napraviti još neke dodatne operacije nad podacima koje selimo u skladište (restrukturiranje, čišćenje, prilagodba i sl.).

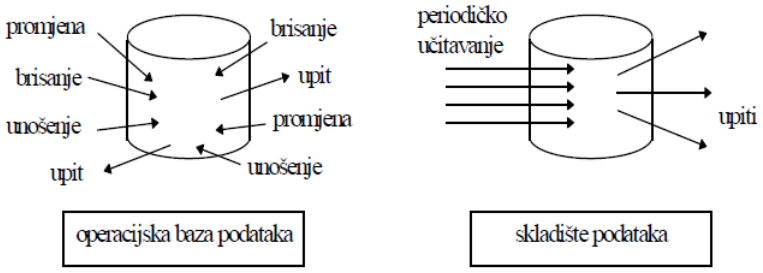
1.3 Usporedba OLTP i OLAP sustava

IT sustave možemo podijeliti na transakcijske (OLTP) i analitičke (OLAP). Generalno gledano, OLTP sustavi opskrbljuju skladišta podataka novim podacima, a OLAP sustavi pomažu pri analizi podataka.
OLTP sustav (Online Transaction Processing) je sustav koji se bavi procesiranjem transakcija i spremanjem podataka u operacijsku (transakcijsku) bazu. U takvim sustavima, podaci se vrlo često mijenjaju pa operacije nad bazom trebaju biti brze i efikasne. Za te baze je karakteristično da su normalizirane (3NF) jer upravo normaliziranost podataka omogućava brzinu i efikasnost operacija za pisanje po bazi.
Klasičan primjer OLTP sustava je bankarski sustav za transakcije. Puno korisnika može istovremeno izvršavati transakcije, a sustav osigurava da se svaka od tih akcija završi efikasno i u potpunosti i da podaci u bazi budu ispravni i konzistentni.
OLAP sustav (Online Analitical Processing) je sustav u kojem je fokus stavljen na analizu podataka. Karakteristika ovih sustava su analitičke mogućnosti, pri čemu se koriste višedimenzionalni podaci. Upotrebljavaju se u poslovnoj inteligenciji, za izradu raznih izvještaja koji pomažu u donošenju poslovnih odluka.
Postoji nekoliko tipova OLAP sustava: MOLAP (Multidimensional OLAP - podaci su spremljeni u višedimenzionalnu kocku), ROLAP (Relational OLAP - podaci su spremljeni u relacijsku bazu podataka) i HOLAP (Hybrid OLAP - kombinacija prethodna dva).
Kao primjer OLAP sustava možemo navesti bolnicu koja čuva podatke o pacijentima unazad 20 godina. Recimo da netko iz uprave želi detaljan izvještaj o najčešćim bolestima i uspješnim načinima liječenja. Tada primijenimo OLAP operacije na skladište podataka s povijesnim podacima i kroz kompleksne upite dobijemo tražene rezultate.

OLTP - OLAP usporedba

Za OLTP sustave je karakterističan velik broj kratkih transakcija (INSERT, UPDATE, DELETE). Naglasak je stavljen na brzu obradu upita, očuvanje integriteta podataka i na efektivnost sustava koja se mjeri prema broju obrađenih transakcija u sekundi.
Za OLAP sustave je karakterističan malen broj transakcija, baza se samo povremeno (periodički) puni novim podacima. Najčešće korištena operacija je SELECT. Upiti su vrlo često kompleksni i uključuju razne agregacije podataka pa su zato i spori. Efektivnost OLAP sustava se mjeri prema vremenu odziva.
Da bi se što više smanjio negativan utjecaj analitičke obrade na rad transakcijskih sustava, a i zbog brojnih razlika tih dvaju sustava, nužno ih je logički i fizički odvojiti. Ključni razlog fizičkog odvajanja skladišta na zasebno računalo u odnosu na sve transakcijske baze podataka je opasnost narušavanja dostupnosti transakcijskih baza. Ako bi se skladište podataka postavilo na isto računalo kao i transakcijska baza, značajni dio procesorske moći trošio bi se na obradu upita nad skladištem podataka koji zahtijevaju puno veći broj pristupa bazi i

znatno su dugotrajniji.



Slika 1.3: OLTP - OLAP operacije (slika preuzeta iz [4])

U sljedećoj tablici navodimo glavne razlike OLTP i OLAP sustava.

	OLTP	OLAP
Podaci	trenutni podaci	povijesni podaci i to samo oni koji su bitni za analizu
Izvor podataka	originalni izvor podataka	podaci dolaze iz raznih OLTP baza
Svrha	kontrola i provođenje poslovnih procesa	analiza podataka - pomoć pri donošenju poslovnih odluka
Unos podataka	nove podatke i promjene na starima uglavnom unose korisnici	periodički dugotrajni automatizirani unosi
Upiti	standardizirani i jednostavni upiti, uglavnom vraćaju mali broj zapisa	kompleksni upiti koji vrlo često uključuju agregacije
Brzina	vrlo brzo izvršavanje upita	ovisi o količini podataka; periodički unosi i kompleksni upiti mogu potrajati i satima
Prostor	može biti malen ako se povijesni podaci arhiviraju	puno veći zbog agregacijskih struktura i povijesnih podataka
Shema	normalizirana shema; puno tablica i veza među njima	star ili snowflake shema; manji broj tablica koje nisu normalizirane
Korisnici	krajnji korisnici poslovnih aplikacija	analitičari, menadžeri

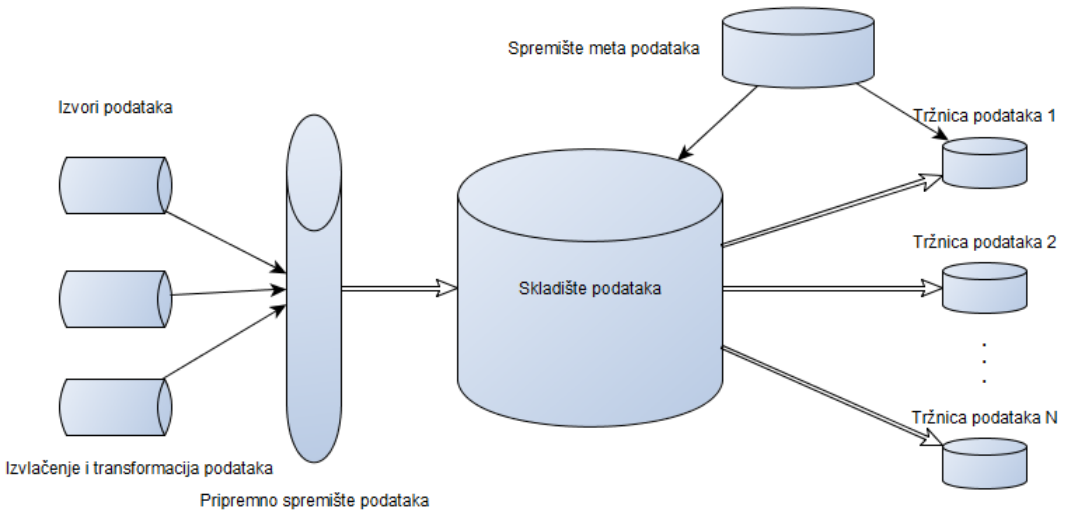
Tablica 1.1: Usporedba OLTP i OLAP sustava

1.4 Dva pristupa skladištenju

Kad se neko poduzeće odluči na implementaciju skladišta podataka, na izbor ima vrlo velik broj softverskih alata i različitih pristupa razvoju skladišta. Da bi napravili dobar odabir, ključno je razumijevanje dva glavna modela za razvoj skladišta podataka - Inmonovog i Kimballovog. Mary Breslin u članku *Battle of the Giants* [12] daje detaljne opise oba modela i navodi glavne sličnosti i razlike među njima.

Inmonov model

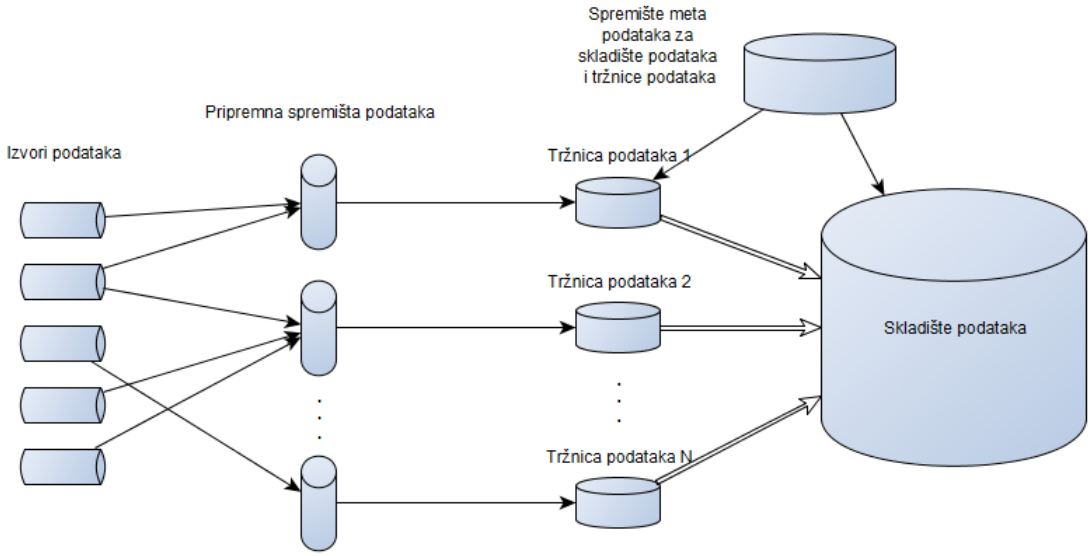
Ovaj model zastupa pristup razvoju **"od vrha prema dolje"** (eng. **"top-down"**). Inmon skladište podataka vidi kao sastavni dio tzv. tvornice podataka o korporaciji (eng. Corporate Information Factory - CIF), što, između ostalog, podrazumijeva da su skladište podataka i transakcijske baze dio jedne veće cjeline. Ovakvo shvaćanje objašnjava zašto Inmonovo skladište podataka mora poštivati većinom iste standarde kao i transakcijski sustavi. U skladištu podataka prevladava normalizirani model podataka, a za pojedine teme, tj. grupe pitanja koje će krajnji korisnici često postavljati, kreiraju se manji dimenzijski modeli.



Slika 1.4: Inmonov model

Kimballov model

Kimballov model zastupa pristup **"od dna prema gore"** (eng. **"bottom-up"**) pri čemu koristi dimenzionalno modeliranje koje je jedinstveno za skladište podataka (takvo što se ne koristi kod transakcijskih baza podataka). Predlaže da se za svaki veći poslovni proces kreira jedna tržnica podataka, a zatim se sve tržnice poveže u skladište podataka.



Slika 1.5: Kimballov model

Sličnosti

Dvije najvažnije sličnosti između ova dva modela su:

- korištenje podataka s vremenskim biljegom (eng. timestamp)
- ETL proces.

Vremenski atribut je među najvažnijim karakteristikama podataka u skladištu podataka. Pomoću njega korisnici mogu provoditi razne vremenske analize, kao što je npr. usporedba prodaje određenog proizvoda ove godine u odnosu na prošlu godinu. Iako imaju malo drugačiji pristup kod skladištenja ovog atributa, i Inmon i Kimball mu posvećuju

puno pažnje pa onda, bez obzira koji pristup se koristi, korisnici mogu postavljati upite prema danu, mjesecu, godini, u ovisnosti o tome je li bio vikend ili praznik i slično.
Korištenje ETL procesa je ključno za svako skladište podataka jer osigurava očuvanje integriteta podataka unutar skladišta. U oba modela, podaci se dohvaćaju iz transakcijskih baza, transformiraju i učitavaju u skladište podataka u Inmonovom modelu, odnosno u tržnice podataka u Kimballovom.

Razlike

Između ova dva modela postoje brojne razlike, a najvažnije su one u područjima metodologije razvoja, modeliranja podataka i arhitekture skladišta podataka.

Razlike u razvojnim metodologijama i arhitekturi

Kao što je već prije spomenuto, Inmonov model ima pristup razvoju "od vrha prema dolje", dok Kimballov ima pristup "od dna prema gore". Inmonov model razvoja skladišta je vrlo kompleksan i namijenjen IT stručnjacima. Izgradnja sustava je dugotrajna i skupa. Za razliku od njega, Kimballov pristup je puno jednostavniji i omogućava i krajnjim korisnicima da sudjeluju u izgradnji skladišta. Izgradnja sustava je brza, ali postoji veća mogućnost zalihosti i nekonzistentnosti unutar sustava pa zato problem integracije dolazi u središte pozornosti.

Razlike u modeliranju podataka

Kod modeliranja podataka, Inmonov pristup je temeljen na podacima, a Kimballov na procesima. Inmon koristi tradicionalne metode modeliranja podataka (model entitet-veza) za koje je potrebno tehničko predznanje pa glavnu ulogu u modeliranju podataka imaju IT stručnjaci. S druge strane, Kimball modeliranje podataka definira na temelju interakcije podataka u različitim poslovnim procesima. Takav pristup ide dobro uz dimenzionalni model podataka jer se upravo na temelju procesa odlučuje koje će se činjenice i dimenzije koristiti u skladištu podataka. Alati za dimenzionalno modeliranje omogućavaju i krajnjim korisnicima da sudjeluju u procesu modeliranja podataka.

1.5 Višedimenzionalni model

Višedimenzionalni model pruža mogućnost izgradnje skladišta podataka u kojem je jasno određen "početni" element za analizu. Također, omogućava istodobno postavljanje nekih parametara poslovanja kao međusobno posve nezavisnih te su podaci modelirani tako da su pogodni za sumiranje.

Činjenica (eng. **fact**) predstavlja središnju točku interesa u procesu analize i donošenja poslovne odluke. **Dimenzije** (eng. **dimensions**) su atributi koji pružaju kontekst potreban za proučavanje činjenica. Glavne funkcije dimenzija su filtriranje i grupiranje. Najčešće korištene dimenzije su vrijeme, ljudi (klijenti, kupci, pacijenti,...), lokacije i sl. **Mjere** (eng. **measures**) su atributi (najčešće numerički) s neprekidnim skupom vrijednosti koji opisuju činjenicu. Skladište podataka sadrži ogroman broj zapisa pa upravo zbrajanjem numeričkih podataka dobivamo vrijednosti zanimljive za analizu. Mjere mogu biti:

1. **zbrojive** - mogu se zbrajati po svim dimenzijama. Npr. ukupna zarada u skladištu podataka za maloprodajni lanac.

2. **poluzbrojive** - mjere koje se mogu zbrajati po nekim, ali ne svim dimenzijama. Npr. stanje bankovnog računa - ne može se zbrajati po vremenskoj dimenziji, ali ako klijent ima više računa u toj banci, može se izračunati zbroj stanja na svim računima za tog klijenta.

3. **ne-zbrojive** - mjere koje se ne mogu zbrajati niti po jednoj dimenziji. Npr. stanje na zalihi - ne može se zbrajati po vremenskoj dimenziji, a ne može se niti zbrajati stanje na zalihi za različite proizvode.

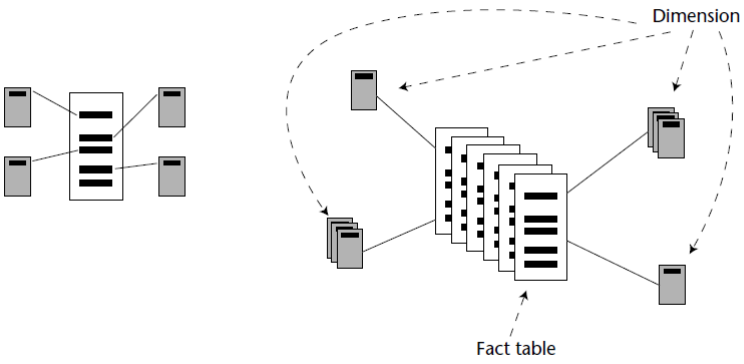
Sve dimenzije su međusobno nezavisne pa se mogu prikazati kao n-dimenzionalni koordinatni sustav, pri čemu su vrijednosti na osima diskretne. Takve prikaze nazivamo **OLAP kockama**.

Slika 1.6: OLAP kocka (slika preuzeta sa [5])

U primjeru sa slike, svaka n-torka (mala kockica) prikazuje prodaju određenog proizvoda na određen dan u određenoj regiji. Ovakva n-torka naziva se **primarnim događajem** (eng. **primary event**). Unutar svake kockice pohranjene su vrijednosti mjera za tu n-torku.
Dimenzijsku hijerarhiju čini niz međusobno funkcijski ovisnih dimenzijskih atributa unutar jedne dimenzije koji činjenicu opisuju na različitoj razini zrnatosti. Na primjer,

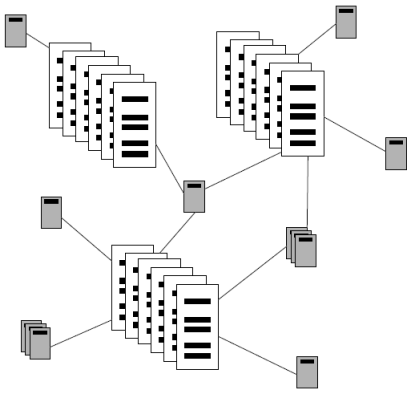
u dimeziji proizvod najdetaljniju razinu određuje upravo atribut proizvod, dok se zbrajanje može izvršiti s obzirom na atribute tip proizvoda (tj. potkategorija) i kategorija proizvoda. Zbrajanje vrijednosti mjera s obzirom na vrijednost nekog atributa u hijerarhiji na višoj razini od osnovne naziva se **agregacijom** po tom atributu. Primjerice, agregacijom prodaje po mjesecima međusobno se zbrajaju svi dnevni prihodi od prodaje nekog proizvoda u istom mjesecu u godini. Na taj način se primarni događaji (tj. kockice osnovne veličine) stapaju u veću kocku koju nazivamo **sekundarni događaj**.
Opisni atributi unutar dimenzije su atributi pomoću kojih nije moguće izvršiti bilo kakvu agregaciju.

Prema Kimballovom članku *A Dimensional Modeling Manifesto* [15], dimenzionalno modeliranje je tehnika oblikovanja čiji je cilj prezentirati podatke na standardiziran, intuitivan način koji omogućava brz pristup podacima. Svaki dimenzionalni model se sastoji od jedne tablice sa višedijelnim ključem koju nazivamo **tablica činjenica** i više manjih tablica koje nazivamo **dimenzijskim tablicama**. Svaka dimenzijska tablica ima jednodijelni primarni ključ koji odgovara točno jednoj komponenti višedijelnog ključa u tablici činjenica. Ovakvu karakterističnu strukturu nazivamo **zvjezdani spoj** (eng. **star join**).



Slika 1.7: Zvjezdani spoj (slika preuzeta iz [13])

U zvjezdanom spoju postoji samo jedna tablica činjenica. Međutim, možemo napraviti dizajn u kojem je nekoliko tablica činjenica međusobno povezano na temelju jedne ili više zajedničkih dimenzija. Takvu strukturu nazivamo **strukturom snježne pahulje** (eng. **snowflake schema**).



Slika 1.8: Struktura snježne pahulje (slika preuzeta iz [13])

Model entitet-veza

Model entitet-veza (eng. entity-relation model - ER) je tehnika oblikovanja baze podataka kojoj je cilj ukloniti zalihost podataka. Koristi se kod transakcijskih baza podataka i, kao što je već ranije spomenuto, kod Inmonovog pristupa oblikovanju skladišta podataka.
Prema Kimballu [15], problemi korištenja ovog modela pri oblikovanju skladišta podataka su sljedeći:

- krajnji korisnici imaju problema sa razumijevanjem i korištenjem ER modela zbog njegove složenosti
- ER model je optimiziran za upisivanje novih zapisa i mijenjanje starih, ali ne i za postavljanje upita koji bi mogli biti korisni pri analizi poslovnih procesa
- ER modeliranje je u suprotnosti sa osnovnim načelima skladištenja podataka, tj. dohvaćanje podataka nije intuitivno i brzo kod ER modela.

1.6 Alati za fizičku realizaciju skladišta podataka

Prilikom oblikovanja skladišta podataka javlja se pitanje gdje ćemo fizički smjestiti te podatke. Odgovor na ovo pitanje ovisi o budžetu, količini podataka i željenim performansama.
Skladišta podataka koja nemaju jako velike količine podataka mogu se realizirati koristeći klasične alate za baze podataka poput MySQL-a, Oracle-a, SQL Servera, PostgreSQL-a,...
Prema članku *Which data warehouse should you use?* [10], to su ona skladišta koja sadrže

manje od 10-100 milijuna redaka, a jednom kad trajanje jednostavnijih upita bude nekoliko minuta ili veće, trebalo bi prijeći na neku specijaliziranu bazu za skladišta podataka. Glavne razlike između alata za analitičke baze podataka u odnosu na klasične baze su mogućnost paralelnog rada i različit format podataka. Na izbor imamo velik broj open-source i komercijalnih analitičkih baza.
Open-source baze su besplatne, skalabilne i najčešće temeljene na PostgreSQL-u, a to su npr. Citrus i Greenplum. Što se tiče komercijalnih rješenja, Amazon i Microsoft nude svoje proizvode Redshift, odnosno Azure. Njih načešće uzimaju oni korisnici koji već koriste neke proizvode za skladišta podataka od tih pružatelja usluga. Osim tih rješenja, na tržištu su i Teradata, Oracle, Vertica te mnogi drugi. Problem kod komercijalnih rješenja je to što su vrlo skupa i komplicirana za postavljanje i administraciju. Ali, s druge strane, imaju iskustvo sa složenim implementacijama, skalabilna su, visoko optimizirana i pružaju uslugu podrške korisnicima.

1.7 Nadzor i održavanje skladišta podataka

Jednom kad izgradimo skladište podataka, moramo ga održavati. Glavna komponenta održavanja skladišta je optimiziranje performansi, a temelji se na nadzoru skladišta podataka. Inmon u svojoj knjizi [13] opisuje kako nadzirati skladište, tj. na što sve trebamo obratiti pažnju.
Dvije su komponente koje treba redovito nadzirati: podaci koji se nalaze u skladištu podataka i njihovo korištenje. Nadzor podataka je nužan za efektivno upravljanje skladištem. Dobivamo sljedeće korisne informacije:

- gdje se događa rast i kojom brzinom
- koji podaci se koriste
- koliko vremena treba za izvršavanje određenih naredbi
- tko su uopće korisnici skladišta podataka
- koji dijelovi skladišta se koriste
- kada se koristi skladište
- koliko često se koristi skladište.

Ako arhitekt skladišta podataka ne zna odgovore na ova pitanja, ne može efektivno upravljati skladištem i raditi na njegovom daljnjem razvoju i održavanju. Pogledajmo, na primjer, zašto nam je bitno znati točno koji se podaci koriste u skladištu. Jedno od glavnih

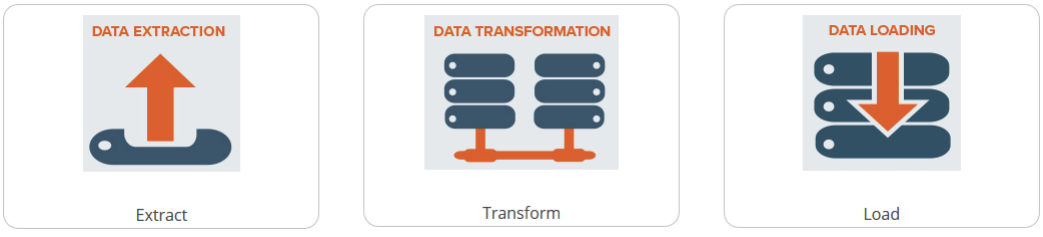
obilježja skladišta je njegov kontinuirani rast. Velike količine povijesnih podataka se periodički dodaju u skladište, isto tako i sumarni podaci te se kreiraju i novi streamovi za dohvaćanje podataka. Prostor potreban za spremište podataka i tehnologije za procesiranje su vrlo skupi pa se javlja pitanje "Trebaju li nam svi ti podaci koji se čuvaju u skladištu ili možemo bez nekih od njih?".
Sve dok arhitekt skladišta ne provodi nadzor korištenja podataka, nema drugog izbora nego stalno kupovati nove resurse za spremanje podataka, dodatne procesore i sl. Ali, ako nadzire aktivnost i korištenje pojedinih podataka u skladištu, onda može odrediti koji podaci se ne koriste. U tom slučaju, ako je moguće, može prebaciti te podatke koji se ne koriste na neki jeftiniji medij za spremanje podataka.
Vrijeme odziva je puno drugačije u okruženju skladišta podataka nego u OLTP sustavima. U OLTP sustavima je nužna brzina izvršavanja upita, ako je izvršavanje sporo, može doći do zastoja u poslovnim procesima, a to može jako štetiti poduzeću. S druge strane, kod skladišta podataka brzina izvršavanja nije toliko kritična. Tu se vrijeme izvršavanja mjeri u minutama i satima, a ponekad i danima. Ali, bez obzira na to što nije važno da je vrijeme odziva vrlo brzo, ne znači da ono uopće nije bitno. Kod korištenja skladišta podataka za analizu, krajnji korisnik iterativno dolazi do rezultata. Dakle, ako je vrijeme koje je potrebno da dobije tražene podatke iz skladišta manje, tada će korisnik biti produktivniji jer će ranije moći prijeći na sljedeći korak analize podataka.
Jedan od problema kod mjerenja vremena odziva u OLAP sustavima je to što postoje različite interpretacije što točno mjerimo. U OLTP sustavima je jasno što mjerimo - korisnik postavi upit pa mjerimo vrijeme koje je potrebno da on dobije odgovor na taj upit. Međutim, kod OLAP sustava je problem u tome što oni na postavljeni upit vraćaju veliku količinu podataka pa nije jasno mjeri li se vrijeme odziva do trenutka kad podaci počnu pristizati ili prestajemo mjeriti tek kad stignu svi traženi podaci, ili možda nešto treće. Upravo iz tog razloga, sustav za nadzor skladišta podataka bi trebao omogućiti različite interpretacije mjerenja vremena.
Još jedan od problema kod nadzora skladišta je gdje postaviti sustav za mjerenje. Jedna mogućnost je postaviti ga na računala krajnjih korisnika i u tom slučaju je utjecaj na performanse cijelog sustava minimalan. Ali, problem kod takvog pristupa je činjenica da bi tada trebalo zasebno administrirati svako korisničko računalo što je gotovo nemoguće u sustavima gdje je jako velik broj korisnika. Alternativna mogućnost je nadzirati OLAP sustav na razini servera. Nakon što je upit formuliran i poslan na server koji upravlja skladištem podataka, nadzor aktivnosti može početi. U ovom slučaju je mnogo jednostavnija administracija sustava za nadziranje, ali postoji mogućnost smanjenja performansi na razini cijelog sustava jer sustav za nadzor koristi resurse servera. Dakle, kod odluke gdje ćemo postaviti sustav za nadzor, treba postići kompromis između lakoće administracije i minimizacije utjecaja na performanse cjelokupnog sustava.
Među najvažnijim namjenama sustava za nadzor je mogućnost usporedbe današnjih rezul-

tata s prosječnim danom. Kad se dogode neki neuobičajeni uvjeti sustava, možemo se pitati po čemu se današnji dan razlikuje od prosječnog. Ali, da bi mogli napraviti takvu usporedbu, moramo imati profil prosječnog dana koji sadrži bitne informacije koje opisuju jedan dan u OLAP sustavu. Jednom kad izmjerimo rezultate trenutačnog dana, možemo ih usporediti sa profilom prosječnog dana. S obzirom na to da se prosječni dan mijenja s vremenom, moramo periodički pratiti te promjene.

Poglavlje 2

ETL proces

Skladište podataka se mora redovno puniti novim podacima da bi moglo služiti svojoj svrsi olakšavanja poslovne analize. Glavni izazov kod skladišta podataka je kako integrirati i spojiti velike količine podataka iz raznih sustava. Proces dohvata podataka iz izvorišnih sustava i njihovo spremanje u skladište podataka je poznato pod nazivom ETL, što je skraćeno za extract (dohvaćanje), transform (transformacija) i load (učitavanje, punjenje). Naziv ETL je možda malo zavaravajući jer se nigdje ne spominje faza transporta i zvuči kao da su ove tri navedene faze u potpunosti odvojene, iako se u praksi preklapaju. Metodologija i zadaci ETL-a su dobro poznati i u upotrebi godinama, te nisu ograničeni samo na skladišta podataka nego se koriste i u drugim područjima IT-a (kad treba povezivati podatke iz raznih aplikacija, sustava i sl.).
ETL sustav je osnova svakog skladišta podataka jer kvaliteta skladišta najviše ovisi upravo o kvaliteti implementacije ETL sustava. Iako je implementacija ETL sustava "pozadinska" aktivnost koja nije pretjerano vidljiva krajnjim korisnicima, prema Kimballu [16], ona oduzima 70% resursa potrebnih za izgradnju i održavanje tipičnog skladišta podataka. Neki "graditelji" skladišta podataka se odlučuju koristiti već gotove ETL alate, a neki koriste svoje vlastito razvijene ETL procese.



Slika 2.1: Extract - Transform - Load (slika preuzeta sa [2])

2.1 Dohvaćanje podataka

Dohvaćanje podataka iz izvorišnog sustava, najčešće transakcijske baze, je prvi korak ETL procesa. Oblikovanje procesa dohvaćanja podataka je često vremenski najzahtjevniji dio ETL procesa, a samim time i skladišta podataka. Izvorišni sustavi mogu biti vrlo kompleksni i slabo dokumentirani pa nam je u tom slučaju teško odrediti koje podatke trebamo dohvatiti. Dohvat podataka se vrši periodički da bi nam skladište podataka moglo služiti kao povijesni pregled svih promjena na podacima. Prilikom dohvata podataka za skladište ne bi smjeli modificirati izvorišni sustav niti prilagođavati njegove performanse ili dostupnost.
Tvrtka Oracle na svojim stranicama [6] ima korisničku dokumentaciju u kojoj navode različite metode dohvata podataka.

Metode dohvata podataka

Odabir metode dohvata podataka ovisi najviše o izvorišnom sustavu i poslovnim potrebama skladišta podataka kojeg oblikujemo. U nastavku navodimo logičke i fizičke metode dohvata.

Logičke metode dohvata

Postoje dva tipa logičkog dohvata:

- **potpuni dohvat**
- **postupni dohvat**.

Kod potpunog dohvata, podaci se u cijelosti dohvaćaju iz izvorišnog sustava. U ovom slučaju dobivamo sve podatke trenutno dostupne u izvorišnom sustavu pa nema potrebe u izvorišnim tablicama pratiti promjene pomoću vremenskog biljega ili nečeg sličnog.
Kod postupnog dohvata, dohvaćamo samo one podatke koji su se promijenili nakon nekog točno određenog događaja u prošlosti. Taj događaj može biti datum zadnjeg dohvata podataka ili neki kompliciraniji poslovni događaj kao npr. zadnji dan fiskalnog perioda. Dakle, potrebna nam je informacija o tome koji su se sve podaci promijenili, a možemo je dobiti ako u izvorišnim tablicama postoji stupac koji sadrži vremenski biljeg ili u izvorišnom sustavu postoji još neka dodatna tablica koja služi za praćenje promjena. Ako u izvorišnom sustavu ne postoje takve informacije, potrebno mu je dodati jednu od navedene dvije logike dohvata.
Mnoga skladišta podataka ne koriste ovakav pristup nego dohvate sve podatke sa izvorišnog sustava i onda ih uspoređuju sa prethodnim dohvatom da bi otkrili koji su se podaci promijenili. Na taj način nema nikakvih potencijalnih promjena na izvorišnom sustavu,

ali zato može doći do velikog opterećenja skladišta podataka, posebno ako su jako velike količine podataka u pitanju.

Fizičke metode dohvata

U ovisnosti o odabranoj logičkoj metodi dohvata i mogućnostima izvorišnog sustava, podaci se mogu fizički dohvatiti na dva načina:

- **online dohvat**
- **offline dohvat**.

Kod online dohvata, podaci se dohvaćaju direktno iz izvorišnog sustava. Prilikom procesa dohvaćanja podataka, spajamo se direktno na izvorišni sustav da bi dohvatili izvorišne tablice ili na neki međusustav koji pohranjuje potrebne podatke u unaprijed određenom formatu, kao što su transakcijski logovi ili tablice promjena. Taj međusustav se također nalazi na izvorišnom sustavu, samo što ne dohvaćamo originalne izvorišne tablice nego ove koje se nalaze na međusustavu.
Može se dogoditi da nemamo direktan pristup izvorišnom sustavu i u tom slučaju koristimo offline dohvat. Podaci su spremljeni izvan izvorišnog sustava i kreirani nekom ekstrakcijskom rutinom. Uglavnom su pohranjeni u nekom unaprijed definiranom formatu u neku običnu tekstualnu datoteku. Tada su nam za daljnje procesiranje potrebne još neke dodatne informacije.

2.2 Transformacija podataka

Transformacija podataka je dio ETL procesa koji podrazumijeva čišćenje i agregaciju podataka tako da bi se podaci mogli analizirati. Postoje dva različita pristupa transformaciji:

- klasičan - nakon što se podaci dohvate i stave u pripremno spremište podataka, na njima se naprave potrebne transformacije pa se nakon toga učitaju u skladište podataka
- ELT pristup - podaci se dohvate i učitaju u skladište podataka pa se onda tamo provode transformacije.

O sličnostima i razlikama između ova dva pristupa transformaciji bit će više riječi u potpoglavlju 2.4.
Prema stranici *etldatabase.com*[1], najčešći tipovi transformacija su:

- čišćenje - osiguravanje konzistentnosti formata podataka, ujednačavanje oznaka (npr. ako se negdje koristi ‘Male’, a negdje ‘M’, onda ih treba uskladiti tako da je oznaka jedinstvena)
- micanje duplikata - identifikacija i brisanje duplih zapisa
- utvrđivanje odnosa ključeva među tablicama
- filtriranje - odabir samo određenih redaka ili stupaca
- povezivanje podataka iz različitih izvora
- dijeljenje jednog stupca u više njih
- validacija podataka
- sumacija podataka.

2.3 Učitavanje podataka u skladište

Dvije su osnovne metode za učitavanje podataka u skladište:

- potpuno učitavanje - koristimo prvi put kad učitavamo podatke u skladište podataka
- postupno učitavanje - podaci se učitavaju periodički; pohranjuje se datum zadnje promjene pa se onda u skladište učitavaju samo podaci poslije tog datuma.

Postupno učitavanje zahtijeva manje vremena, ali je puno kompliciranije s logičke strane. Tri glavna problema su:

- redoslijed - dolaze velike količine podataka, a kako postoji mogućnost paralelnog rada, može se dogoditi da podaci neće biti procesuirani u redosllijedu u kojem su stigli
- promjena sheme - shema se mijenja ako brišemo ili dodajemo neke atribute ili šaljemo krive tipove podataka, u tom slučaju podaci mogu postati nekonzistentni
- nadzor- s obzirom na to da podaci dolaze iz različitih izvora, pogreške su neizbježne - bitno ih je što prije uočiti.

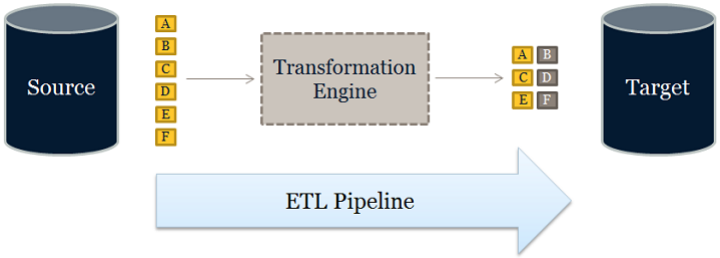
Svaki od ovih problema može za rezultat imati krive podatke, a oporavak od ovakvih problema je prilično zahtjevan.

2.4 Koristiti ETL ili ELT?

ELT je pojam koji se koristi za integracijski proces kod kojeg se podaci prvo učitavaju u odredišni sustav i tek onda transformiraju. Na samom početku povijesnog razvoja skladišta podataka koristio se ETL pristup, ali je danas, s pojavom novih, bržih i moćnijih tehnologija, omogućen i ELT način skladištenja podataka.

ETL način procesiranja

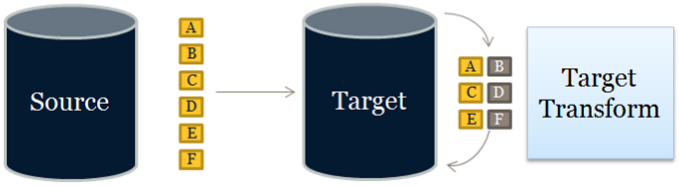
ETL proces bi trebao funkcionirati poput cjevovoda, tj. stalno bi trebali teći podaci kroz njega bez ikakvih smetnji. Za razliku od pravih, fizičkih cjevovoda, ETL alati imaju mogućnost proširivanja da bi mogli primiti veće količine podataka. Ali, ipak, imaju i oni ograničenja pa im u nekom trenutku može ponestati memorije.
Mnogi ETL alati podržavaju mogućnost paralelnog procesiranja - kreiraju više tokova podataka i rasporede podatke po njima. Ovakav način rada jako ubrzava proces, ali treba paziti da su podaci takvi da se lako mogu razdvojiti na različite tokove bez da to razdvajanje ima utjecaja na krajnji rezultat.
Dizajniranje kvalitetnih ETL procesa može biti zahtjevno, ali se na kraju trud ipak isplati.



Slika 2.2: ETL način procesiranja (slika preuzeta sa [17])

ELT način procesiranja

Specifičnosti ELT razvoja mogu varirati u odnosu na platformu na kojoj se nalaze. Na primjer, Hadoop klasteri mogu razbiti problem na manje dijelove i onda ih distribuirati među velikim brojem strojeva za daljnju obradu. Tada će problem biti brže riješen zbog paralelizma. Neki sustavi imaju dovoljno resursa za izvršavanje bilo koje transformacije, dok neki drugi zahtijevaju pažljivo planiranje i dizajn.



Slika 2.3: ELT način procesiranja (slika preuzeta sa [17])

Odabir pristupa

Odabir pristupa ovisi o prioritetima. U svakom slučaju, bolje je imati što manje podataka koje trebamo prenositi među sustavima. ELT nema dio zadužen za transformaciju nego se on nalazi na odredišnom sustavu, gdje možda služi još i za neke druge poslove. ETL može biti ekonomičniji jer su nam za ELT potrebni vrlo moćni odredišni sustavi.
Predost ELT pristupa je ta što minimizira trajanje procesiranja na izvorišnom sustavu, a kako je izvorišni sustav vrlo često transakcijski produkcijski sustav, brzina rada čini razliku krajnjem korisniku tog produkcijskog sustava. Loše strane ovog pristupa su dodatni korak u procesiranju jer trebamo koristiti pomoćne tablice i zauzimanje veće količine memorije na disku.

2.5 ETL alati

Postoje tri opcije kod odabira ETL alata:

(a) kupiti neki komercijalni alat

(b) koristiti open-source alat

(c) pisati vlastite skripte.

Ako se odlučimo na pisanje vlastitih ETL skripti, one se mogu pisati u većini programskih jezika, pri čemu su Python i SQL najpopularniji. Što se tiče već gotovih alata, na tržištu postoji mnogo komercijalnih i open-source alata od raznih proizvođača.

Komercijalni ETL alati

ETL alati štede vrijeme i novac tako što eliminiraju potrebu za "ručnim kodiranjem" pri razvoju novih skladišta podataka. Također se koriste za olakšavanje rada administratorima

baza podataka koji povezuju različite grane baza i integriraju ili mijenjaju postojeće baze podataka.
Neki od najpopularnijih komercijalnih ETL alata su:

- Alooma
- IBM InfoSphere DataStage
- Informatica PowerCenter
- Microsoft SQL Server Integration Services
- Oracle Warehouse Builder i Oracle Data Integrator

Open-source ETL alati

Na tržištu postoji i mnogo open-source ETL alata, njihovo glavo obilježje je, naravno, to što su besplatni za korištenje, za razliku od komercijalnih alata s licencom.
Prema stranici *etltools.net* [3], na njihovo korištenje se najčešće odlučuju pripadnici jedne od ovih skupina:

- nezavisni dobavljači softvera koji traže ugrađenu integraciju podataka - smanjuju se troškovi i ta ušteda se onda prenosi i na kupce njihovih proizvoda; mogućnosti integracije, migracije i transformacije su uključene kao ugrađena komponenta; krajnji proizvod zauzima manje memorije u usporedbi sa komercijanim rješenjima
- sistemski integratori koji traže jeftin integracijski alat - open-source ETL alati im omogućavaju da brže i kvalitetnije isporuče svoja integracijska rješenja
- male i srednje velike kompanije sa malim budžetima koje nemaju pretjerano kompleksne zahtjeve pa su im dovoljni alati koje nude open-source davatelji usluga.

Neki od najpoznatijih open-source ETL alata su:

- Apache Kafka
- clover.ETL
- Pentaho Data Integration
- Talend Open Studio.

Koristiti ETL alat ili sve samostalno kodirati?

Ukratko, odgovor je "Ovisi". Kimball i Caserta u svojoj knjizi [16] navode glavne prednosti za svaki od ta dva pristupa.

Prednosti korištenja ETL alata:

- jednostavniji, brži i jeftiniji razvoj. Cijena alata će se isplatiti na dovoljno velikim i kompliciranim projektima.
- ETL alate mogu koristiti i ljudi tehničke struke koji imaju široka poslovna znanja, ali nisu profesionalni programeri
- mnogi ETL alati imaju integrirane repozitorije meta podataka koji mogu sinkronizirati meta podatke među različitim izvorišnim sustavima, odredišnim bazama podataka i drugim BI alatima
- mnogi ETL alati automatski generiraju meta podatke u svakom koraku procesa i time potiču konzistentnu metodologiju koje se svi programeri moraju držati
- mnogi alati imaju ugrađene funkcije koje pomažu pri kreiranju dokumentacije i lakšem nadzoru promjena. Također, trebali bi pružati podršku za upravljanje međuovisnostima pojedinih dijelova sustava i mogućnost rukovanja pogreškama (eng. error handling)
- ETL alati imaju konektore za većinu izvorišnih i odredišnih sustava pa bi trebali biti u mogućnosti obraditi raznovrsne kompleksne konverzije tipova podataka
- uglavnom nude mogućnost enkripcije i kompresije podataka
- većina alata ima vrlo dobre performanse čak i za prilično velike skupove podataka
- ETL alati mogu upravljati kompleksnim balansiranjem opterećenja među serverima čime je riješen problem potencijalnog deadlock-a servera
- automatski provjeravaju koje su posljedice na procese i aplikacije ukoliko pokušamo promijeniti shemu
- ETL alati se mogu proširivati dodavanjem novih funkcionalnosti.

Prednosti samostalnog kodiranja:

- alati za testiranje (eng. unit testing) su dostupni kod ručno kodiranih sustava - npr. JUnit (za testiranje Java programa)

- tehnike objektno-orijentiranog programiranja mogu pomoći pri održavanju konzistentnosti transformacija podataka
- može se izravnije upravljati meta podacima, iako, treba samostalno kreirati sva sučelja za meta podatke
- pristup temeljen na postojećem ETL alatu može biti ograničavajući s obzirom na sposobnosti dobavljača i njihov skriptni jezik, za razliku od ručnog kodiranja, gdje sami možemo odabrati u kojem jeziku ćemo programirati (iako, većina ETL alata ima dodatne module kojima pruža korisnicima mogućnost kodiranja u nekom od standardnih jezika)
- ručno-kodirani ETL sustavi pružaju neograničenu fleksibilnost. Vrlo često, jedinstveni pristup ili odabir programskog jezika mogu učiniti veliku razliku.

Kod izgradnje skladišta podataka dolazi do vrlo velikih početnih troškova. Treba kupiti servere: barem jedan server za bazu, server za OLAP i ETL server. Zatim, treba platiti razne licence, konzultante i pokriti mnoge druge troškove pokretanja novog projekta. Svi ti troškovi su obavezni, ali vrlo često se pokušava smanjiti troškove tako što se ne uloži u dobar ETL alat. Moguće je implementirati skladište podataka bez ETL alata, ali ipak, preporuča se njegova kupnja jer, dugoročno gledano, ETL alat smanjuje cijenu izgradnje i održavanja skladišta podataka.
Još neke prednosti korištenja gotovih ETL alata:

- "definiraj jednom, iskoristi više puta" - neka pravila i strukturirane rutine možemo ponovo koristiti čime se omogućava konzistentnost skladišta podataka
- analiza utjecaja - možemo odrediti na koje tablice, stupce i procese utječu predložene promjene
- repozitorij meta podataka - jednostavno kreiranje, održavanje i objavljivanje podataka; nasljeđivanje poslovnih definicija iz alata za modeliranje podataka
- mogućnost dinamičkog ažuriranja tablica
- upravljanje batch procesiranjem - mogućnost uvjetnog učitavanja, statistike učitavanja, automatske e-mail notifikacije
- jednostavnije povezivanje sa raznim kompleksnim sustavima, kao što je npr. SAP
- paralelno procesiranje
- iskustvo dobavljača.

Poglavlje 3

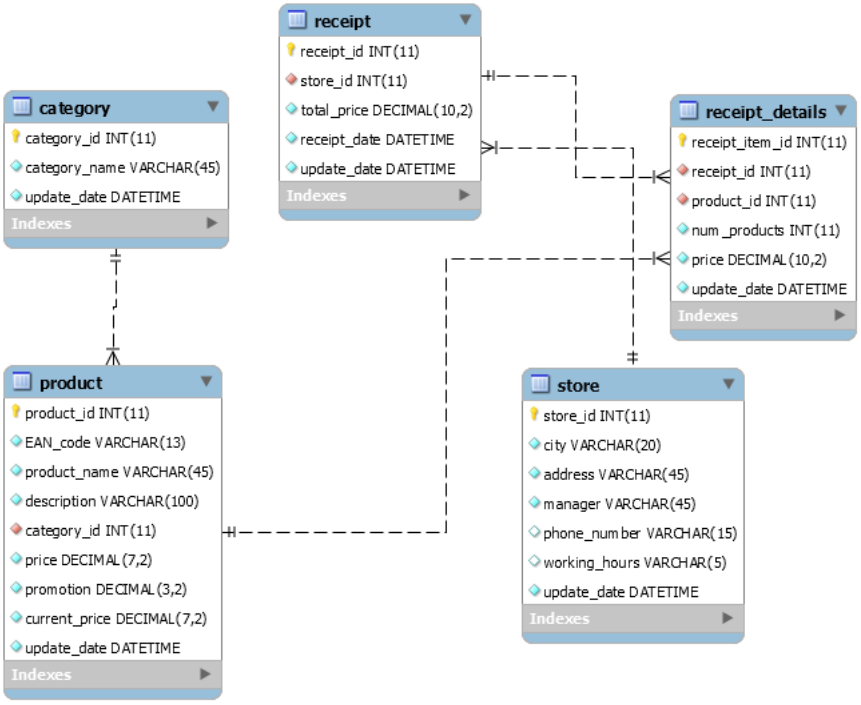
Studijski primjer

U ovom poglavlju ćemo demonstrirati kako dizajnirati i implementirati skladište podataka primjenom ETL procesa na primjeru maloprodajne trgovine. Koristimo MySQL bazu za izvorišne tablice te također i za skladište podataka koje konstruiramo iz njih. Kako je ovo samo studijski primjer koji ne sadrži velike količine podataka, možemo čuvati transakcijsku i analitičku bazu na istom serveru. Napravljene su dvije sheme oltp i olap - unutar prve su spremljene transakcijske tablice, a unutar druge ćemo kreirati dimenzijske tablice i tablicu činjenica. Za ETL proces koristimo alat Talend Open Studio for Data Integration.

3.1 Shema transakcijske baze

Pretpostavimo da imamo transakcijske podatke za trgovinu za koju trebamo napraviti skladište podataka. Implementiramo samo one tablice koje će nam biti potrebne pri dizajnu dimenzionalnog modela analitičke baze. Na slici 3.1 možemo vidjeti shemu naše transakcijske baze. Imamo pet tablica:

- `product` - sadrži podatke o proizvodima
- `category` - sadrži informacije o kategorijama proizvoda
- `store` - sadrži informacije o prodavaonicama
- `receipt` - račun koji sadrži informacije poput ukupnog iznosa računa te prodavaonice u kojoj je plaćen
- `receipt_details` - tablica sa detaljima računa poput proizvoda koji je kupljen i u kojoj količini.



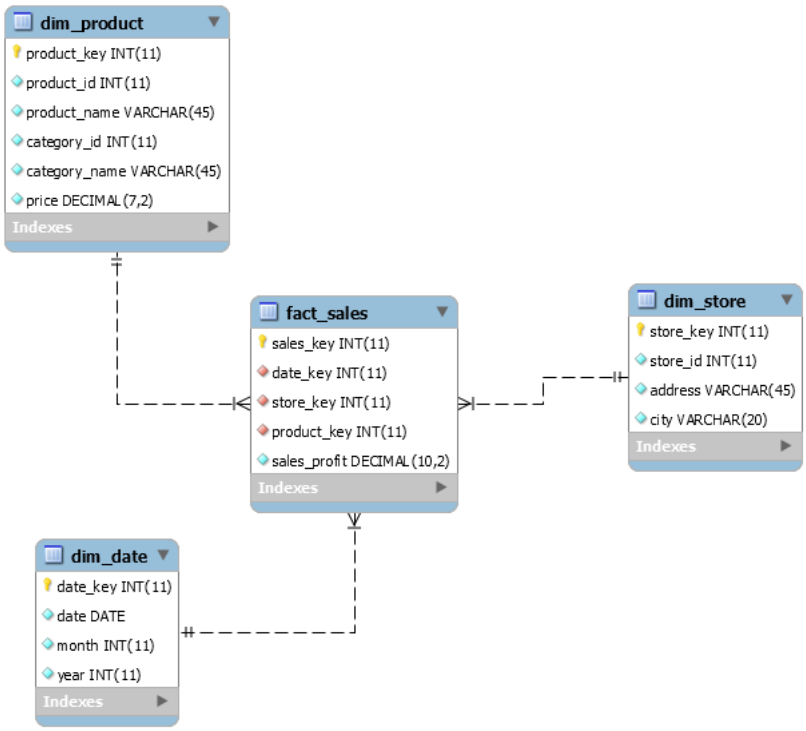
Slika 3.1: Shema transakcijske baze

Na slici vidimo odnose među tablicama, iscrtkane linije povezuju jednu tablicu s drugom koja među svojim atributima sadrži ključ prve tablice kao strani ključ. Npr. ključ `category_id` iz tablice `category` je strani ključ (označen crvenim rombom) u tablici `product`. Plavi i bijeli rombovi označavaju atribute, s time da oni koji su označeni plavom bojom ne smiju sadržavati `NULL` vrijednosti, a bijeli smiju.

3.2 Implementacija dimenzionalnog modela

Kod odabira dimenzija i činjenica, prvo moramo znati na koja pitanja želimo znati odgovore. U našem primjeru nas zanima odgovor na pitanje "Kolika je ukupna zarada od prodaje?". Dakle, činjenica koju promatramo je prodaja pa kreiramo tablicu `fact_sales`, a njena mjera je ukupna zarada, tj. atribut `sales_profit`.
Zaradu možemo promatrati samo za jedan proizvod, za neku kategoriju proizvoda ili za sve proizvode pa kao jednu od dimenzija uzimamo proizvod, tj. kreiramo dimenzijsku tablicu `dim_product` sa odgovarajućim atributima. Također, kreiramo dimenzijske tablice `dim_store` i `dim_date` jer nas zanima kolika je zarada od prodaje u određenim

prodavaonicama ili gradovima (`dim_store`), odnosno u jednom danu, mjesecu ili godini (`dim_date`). Na slici 3.2 je prikazana shema našeg skladišta u skladu s dimenzionalnim modelom.



Slika 3.2: Shema skladišta podataka u skladu s dimenzionalnim modelom

Kreiranje vremenske dimenzije

Prva i najbitnija dimenzija koju ćemo kreirati je vremenska dimenzija koja se koristi u većini, ako ne i u svim skladištima podataka. U ovom slučaju, za ključ `date_key` nećemo koristiti funkciju `auto_increment` nego ćemo za ključ uzimati string oblika "yyyyMMdd" koji će jednoznačno označavati datum "yyyy-MM-dd". Na primjer, za datum 12.08.2017. u tablici imamo `date_key = "20170812", date = "2017-08-12", month = 8, year = 2017`. U potpoglavlju 3.3 objasnit ćemo na koji način punimo tablicu.

Kreiranje ostalih dimenzija

U ostalim dimenzijskim tablicama za ključeve (`product_key` i `store_key`) koristimo funkciju `auto_increment`, a one također sadrže kao atribut i ključ iz transakcijske baze

(`product_id`, odnosno `store_id`). Za tablicu `dim_product` koristimo dvije tablice iz transakcijske baze, `product` i `category`. Tablicu `dim_store` popunjavamo koristeći samo transakcijsku tablicu `store`.

Kreiranje tablice činjenica

Tablica za činjenicu prodaja `fact_sales` sadrži višestruki ključ kojim je povezana sa svim dimenzijskim tablicama. Osim toga, sadrži atribut `sales_profit` koji je mjera za ukupnu prodaju. Jedan redak u tablici predstavlja zaradu za određen proizvod na određen datum u određenoj prodavaonici.

3.3 Implementacija ETL procesa

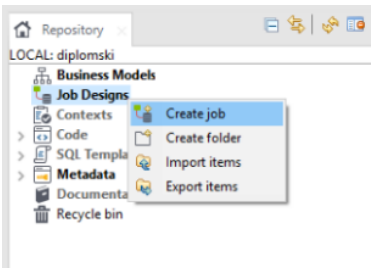
Sad kad smo kreirali tablice, trebamo ih napuniti podacima korištenjem ETL procesa. Prvo trebamo procesirati dimenzije, a tek onda činjenice jer dimenzije daju kontekst činjenicama. Različite komponente skladišta podataka možemo procesirati odvojeno pa nam to pruža vrlo veliku fleksibilnost. Koristit ćemo alat Talend Open Studio for Data Integration (TOS) [9]. Na internetu postoji mnogo priručnika za korištenje TOS-a, a za potrebe izrade ovog studijskog primjera korišteni su materijali dostupni na stranici Talend by Example [8] i priručnik za kreiranje skladišta podataka koji se nalazi u arhivi online tečajeva O'Reilly škole tehnologija [7].

Talend Open Studio for Data Integration

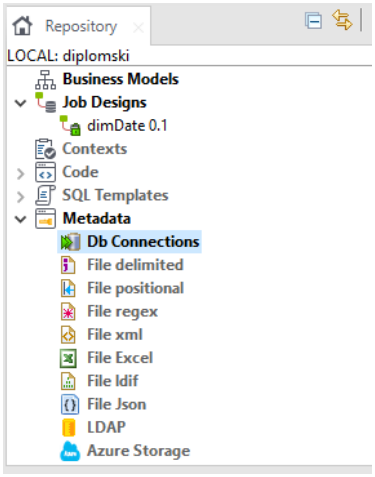
TOS je open-source ETL alat baziran na Javi. Postoji i komercijalna verzija koja ima još neke dodatne mogućnosti.
Sučelje TOS-a se sastoji od četiri osnovna dijela:

- "platno" - zauzima najveći dio sučelja, na njemu slažemo komponente i povezujemo ih u tokove podataka. Postoje dvije kartice - na jednoj možemo vidjeti vizualni prikaz, a na drugoj izvorni kod u Javi koji se automatski generira
- paleta - nalazi se s desne strane, sadrži komponente koje možemo dodati
- repozitorij - nalazi se s lijeve strane, tamo možemo pristupiti već postojećim poslovima, kreirati nove, možemo spremati razne meta podatke i još mnogo toga
- postavke - nalaze se na dnu sučelja ispod platna, postoje četiri kartice, od kojih su najbitnije zadnje dvije - na predzadnjoj postavljamo parametre za svaku dodanu komponentu, a na zadnjoj možemo pokrenuti posao.

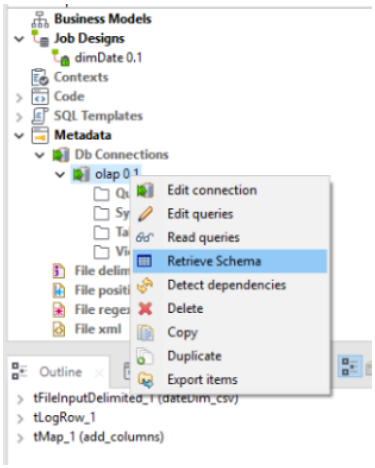
Unutar projekta kreiramo **poslove** (eng. **job**) koji se mogu sastojati od jednog ili više **tokova podataka** (eng. **data flow**). Najjednostavniji oblik toka se sastoji od jedne ili više ulaznih komponenti, zatim komponente koja služi za transformaciju podataka i izlazne komponente.
Što se tiče meta podataka, u repozitorij možemo spremati razne informacije, kao što su npr. podaci za konekciju na bazu, informacije o rasporedu datoteka, opisi tablica dohvaćenih iz baze i sl.



Slika 3.3: Kreiranje posla



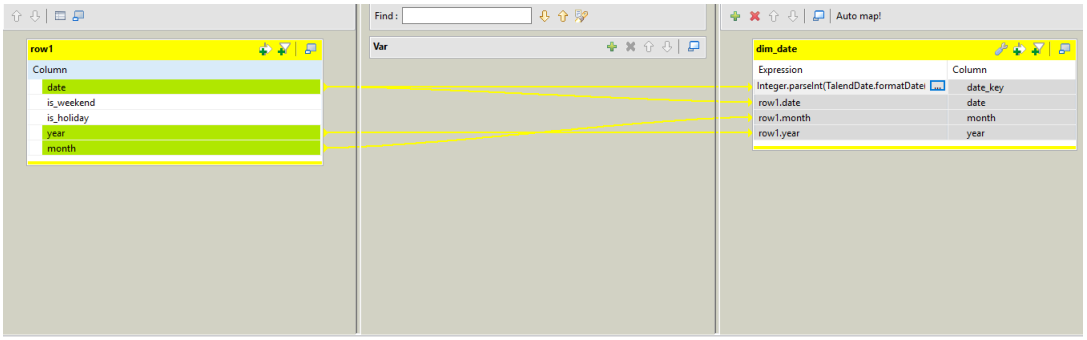
Slika 3.4: Repozitorij meta podataka



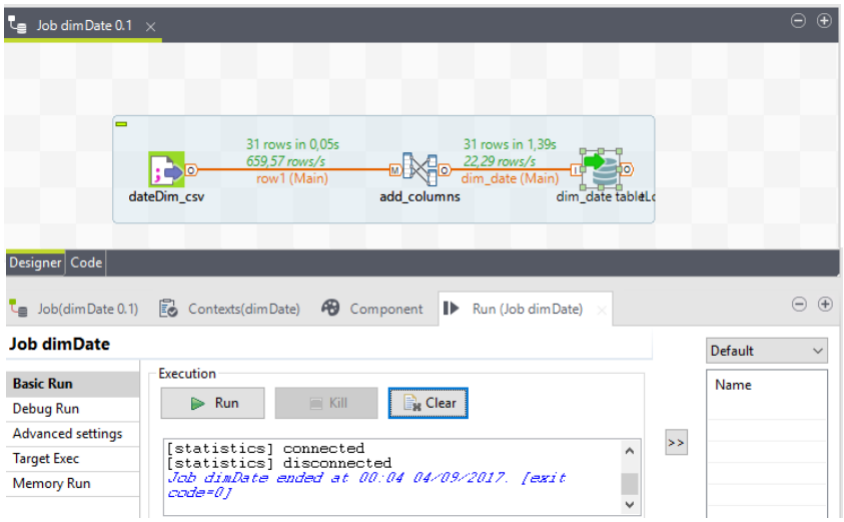
Slika 3.5: Dohvaćanje sheme

ETL posao za vremensku dimenziju

Vremensku dimenziju je dovoljno procesirati samo jednom i zato ćemo za nju kreirati odvojen posao. Podatke ćemo učitavati iz .csv datoteke u kojoj se nalaze zapisi u obliku *date,is_weekend,is_holiday,year,month*. To ćemo napraviti korištenjem komponente **tFileInputDelimited** iz palete. U tablici `dim_date` nam ne trebaju svi ti podaci, a oni koji nam trebaju su poredani drugačijim redoslijedom pa ćemo ih pomoću komponente **tMap** transformirati i pravilno preslikati (slika 3.6). Sljedeće što trebamo napraviti je učitati te podatke u stvarnu tablicu `dim_date` koja se nalazi na bazi (u olap shemi). Prvo trebamo u repozitoriju meta podataka spremiti informacije za konekciju na bazu (tip baze, login, password, server, port,...) i zatim dohvatiti shemu. Nakon toga trebamo staviti komponentu **tMysqlOutput** na "platno", spojiti je sa prethodnom komponentom i postaviti sve parametre kako treba (za detalje pogledati bilo koji od već spomenutih materijala za rad s Talendom). Na kraju trebamo pokrenuti naš posao i tada smo gotovi sa ETL procesom za vremensku dimenziju (slika 3.7).



Slika 3.6: Transformacija



Slika 3.7: ETL za vremensku dimenziju

ETL posao za periodičko punjenje skladišta podataka

Preostale dimenzije i tablica činjenica bi se trebale periodički ažurirati. Možemo kreirati odvojeni posao za svaku od njih, ali možemo ih i sve staviti u isti posao. Naš studijski primjer ćemo raditi na drugi način.

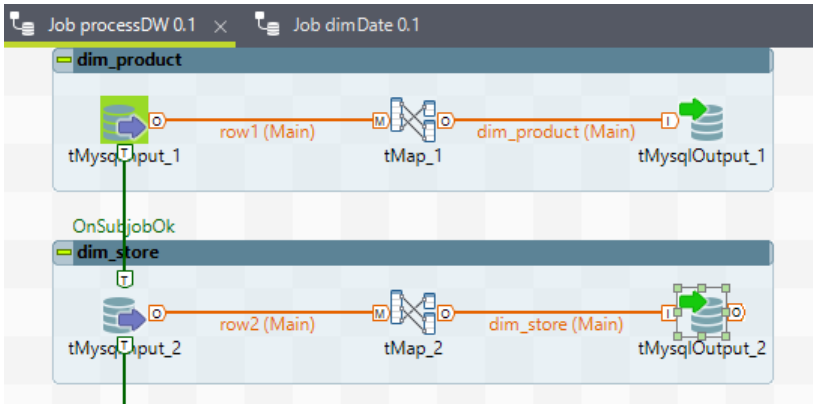
ETL za preostale dimenzije

Podatke ćemo dohvaćati iz baze iz oltp sheme pa onda prvo kreiramo konekciju na tu shemu u bazi i spremimo je u repozitorij meta podataka tako da je možemo i kasnije koristiti.
Prvo radimo ETL za dimenziju proizvod. Podatke možemo dohvatiti pomoću **tMysqlInput**

komponente. U postavkama u polje query upisujemo SQL upit kojim dohvaćamo željene podatke, u našem slučaju upit glasi ovako:

```
"SELECT p.product_id, p.product_name, c.category_id, c.category_name,
p.current_price as price
FROM oltp.product p
JOIN oltp.category c ON (p.category_id = c.category_id);".
```

Zatim dodajemo komponentu **tMap** u kojoj ispravno povežemo sve atribute u tablicama i napravimo potrebne transformacije ako ih ima. Vrlo je bitno da se podudaraju tipovi podataka u obje tablice da ne bi došlo do greške. Također, kod definiranja sheme tablice koju popunjavamo, moramo navesti ispravan ključ i poredati sve atribute onim redoslijedom kojim su poredani u tablici `dim_product` na bazi. Na kraju dodamo **tMysqlOutput** komponentu i u postavkama u polje Action on data stavimo Update or insert. Prema ključu zadanom u prethodnom koraku, TOS će znati postoji li već redak s tim ključem u tablici i na temelju toga poduzeti odgovarajuću akciju.
ETL proces za `dim_store` se kreira na sličan način kao i za `dim_product` pa nećemo navoditi korake. TOS može izvršavati potposlove paralelno, ali u ovom slučaju mi to ne želimo da nam se slučajno ne bi dogodilo da se tablica činjenica popuni prije nego što sve dimenzije završe sa punjenjem. Da bi to spriječili, dodat ćemo okidače (triggere) `OnSubjobOk` koji će osigurati da se potposlovi izvršavaju jedan za drugim, a ne paralelno (slika 3.8).



Slika 3.8: Okidač `OnSubjobOk`

ETL za tablicu činjenica

Činjenicu o prodaji ćemo procesirati u tri koraka:

1. prebacit ćemo potrebne podatke sa izvorišnog sustava u pomoćnu tablicu `stage_fact_sales`

2. kreirati indekse na tablici `stage_fact_sales`

3. učitati podatke u tablicu `fact_sales` korištenjem ELT komponenti.

Dakle, prvo trebamo provesti ETL proces nad tablicom `stage_fact_sales`. To radimo na isti način kao i kod dimenzijskih tablica. Tablice iz transakcijske baze iz kojih dohvaćamo podatke su `receipt` i `receipt_details` pa upit koji moramo upisati u polje query u komponentu **tMysqlInput** glasi:

```
"SELECT r.receipt_id, r.store_id, rd.product_id, rd.price,
r.receipt_date FROM receipt r
JOIN receipt_details rd ON (r.receipt_id = rd.receipt_id);".
```

Kako tablica `stage_fact_sales` nije kreirana na bazi, TOS ju može sam kreirati ako mu u komponenti **tMysqlOutput** u polju Action on table označimo Drop table if exists and create.
Sada se prebacimo na MySQL bazu u olap shemu gdje se kreirala ova tablica. Trebamo napisati proceduru koja kreira indekse:

```
CREATE PROCEDURE ‘etl_pre_fact_sales‘()
BEGIN
  ALTER TABLE stage_fact_sales add index(receipt_id);
  ALTER TABLE stage_fact_sales add index(store_id);
  ALTER TABLE stage_fact_sales add index(product_id);
  ALTER TABLE stage_fact_sales add index(receipt_date);
  TRUNCATE TABLE fact_sales;
END
```

Vratimo se u TOS i pozovemo proceduru `etl_pre_fact_sales()` pomoću komponente **tMysqlRow** - u postavkama komponente u polje query stavimo

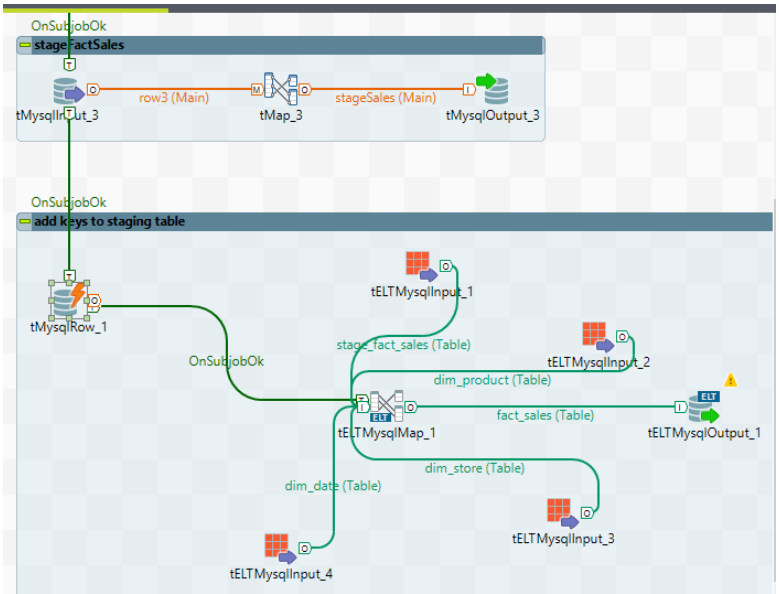
```
"call etl_pre_fact_sales();".
```

Ova procedura se ne može izvršiti prije nego tablicu `stage_fact_sales` popunimo sa podacima pa i ovdje moramo dodati okidač `OnSubjobOk` (na komponentu **tMysqlInput** u toku podataka za tablicu `stage_fact_sales`). Trebamo još pokrenuti posao da bi se tablica kreirala i nakon što se posao uspješno završi trebamo ažurirati shemu u repozitoriju

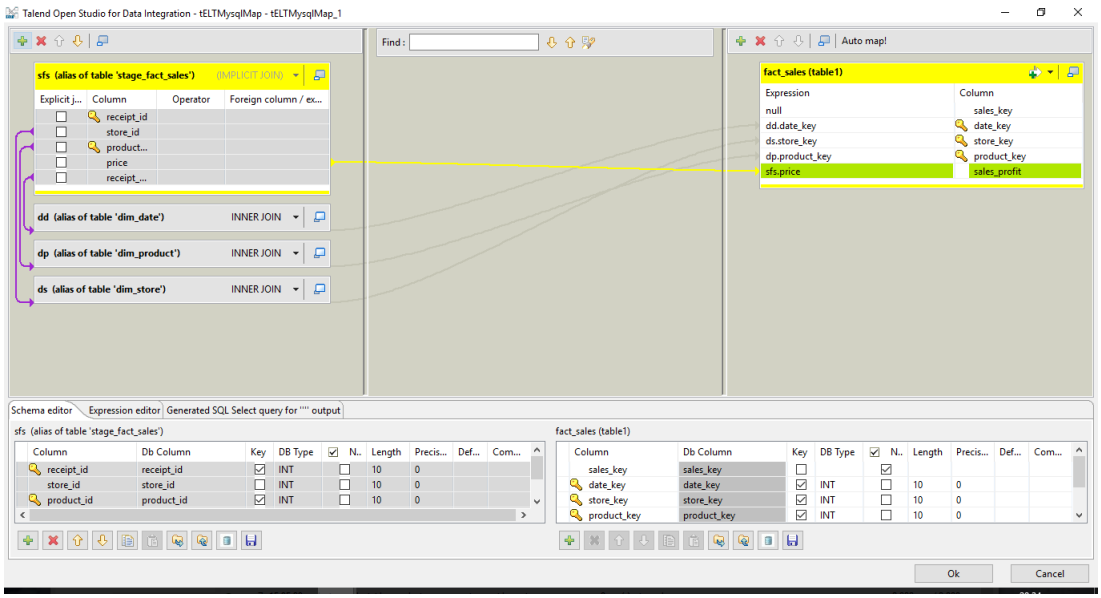
meta podataka jer će nam trebati u sljedećem koraku.
Sada je sve spremno za preslikavanje. Na platno trebamo dodati komponentu **tELTMysqlMap** te četiri komponente **tELTMysqlInput** po jednu za svaku izvorišnu tablicu (`stage_fact_sales, dim_product, dim_store, dim_date`) i još jednu izlaznu komponentu **tELTMysqlOutput** (slika 3.9). Zatim dvaput kliknemo na komponentu **tELTMysqlMap** i otvorit će nam se prozor za kreiranje preslikavanja. Trebamo spojti sve izvorišne tablice na sljedeći način:

```
"SELECT
null, dd.date_key , ds.store_key , dp.product_key , sfs.price
FROM stage_fact_sales sfs
INNER JOIN dim_date dd ON( dd.date = date(sfs.receipt_date) )
INNER JOIN dim_product dp ON( dp.product_id = sfs.product_id )
INNER JOIN dim_store ds ON( ds.store_id = sfs.store_id )".
```

Ovaj kod će se sam generirati u tabu Generated SQL query SELECT nakon što na odgovarajući način dodamo i pospajamo sve tablice. Zatim trebamo mapirati podatke iz izvorišnih tablica u odredišnu tablicu. Nakon što sve posložimo, mapiranje treba izgledati kao na slici 3.10. Sada još samo pokrenemo posao i podaci će se učitati u skladište podataka u tablicu `fact_sales`.



Slika 3.9: Tok podataka za tablicu `fact_sales`



Slika 3.10: Preslikavanje za tablicu `fact_sales`

3.4 Primjer upita nad skladištem podataka

Recimo da nas zanima ukupna zarada po kategoriji proizvoda na dan 16.08.2017. Tada će nam biti potrebni podaci iz tri tablice - iz tablice činjenica `fact_sales` (jer se tamo nalaze podaci o zaradi) te dimenzijskih tablica `dim_product` (jer se tamo nalaze podaci o kategoriji proizvoda) i `dim_date`. Sadržaj tablica vidimo na slikama 3.11, 3.12 i 3.13.
Da bi dobili tražene informacije, upit koji trebamo izvršiti je sljedeći:

```
SELECT dp.category_id, dp.category_name, dd.date, SUM(fs.sales_profit)
FROM fact_sales fs
INNER JOIN dim_product dp ON (fs.product_key = dp.product_key)
INNER JOIN dim_date dd ON (fs.date_key = dd.date_key)
WHERE dd.date_key = 20170816
GROUP BY dp.category_id;
```

Tada dobivamo traženi rezultat koji vidimo na slici 3.14.
Isti ovaj rezultat mogli smo dobiti i iz transakcijske baze, ali kao što smo već ranije spomenuli, potrebno je razdvojiti transakcijske baze od analitičkih da ovakvi analitički upiti nad velikim količinama podataka ne bi ometali (usporavali) rad transakcijske baze.

sales_key	date_key	store_key	product_key	sales_profit
1	20170816	1	1	20.00
2	20170816	1	3	15.79
3	20170816	1	7	9.80
4	20170816	3	4	22.50
5	20170816	3	5	14.20
6	20170817	4	2	6.50
7	20170817	4	6	55.59
8	20170817	3	3	15.79
9	20170817	3	4	11.25

Slika 3.11: Tablica `fact_sales`

product_key	product_id	product_name	category_id	category_name	price
1	1	banana	1	hrana i piće	10.00
2	2	jabuka	1	hrana i piće	6.50
3	3	četkica za zube	3	higijenski pribor	15.79
4	4	šalica	4	posuđe	11.25
5	5	sok Juicy	1	hrana i piće	14.20
6	6	set tanjura	4	posuđe	55.59
7	7	vreće za smeće	2	kućanske potrepštine	9.80
8	8	šampon za kosu	3	higijenski pribor	16.29

Slika 3.12: Tablica `dim_product`

date_key	date	month	year
20170809	2017-08-09	8	2017
20170810	2017-08-10	8	2017
20170811	2017-08-11	8	2017
20170812	2017-08-12	8	2017
20170813	2017-08-13	8	2017
20170814	2017-08-14	8	2017
20170815	2017-08-15	8	2017
20170816	2017-08-16	8	2017
20170817	2017-08-17	8	2017
20170818	2017-08-18	8	2017
20170819	2017-08-19	8	2017

dim_date 13 ×

Slika 3.13: Tablica `dim_date`

category_id	category_name	date	sum(fs.sales_profit)
1	hrana i piće	2017-08-16	34.20
2	kućanske potrepštine	2017-08-16	9.80
3	higijenski pribor	2017-08-16	15.79
4	posuđe	2017-08-16	22.50

Slika 3.14: Rezultat upita

Zaključak

Oblikovanje skladišta podataka je vrlo opširna tema koja obuhvaća modeliranje podataka na temelju korisničkih zahtjeva i dostupnih podataka iz transakcijskih baza, zatim odabir raznih alata koje ćemo koristiti te oblikovanje ETL procesa kojim prebacujemo potrebne podatke u skladište. Na izbor imamo velik broj alata, što onih za fizičku realizaciju skladišta, što onih za implementaciju ETL procesa. Nakon realizacije skladišta i inicijalnog punjenja podacima, skladište treba nadzirati i održavati.

Tema koje se nismo dotakli u ovome radu je poslovna inteligencija (eng. business intelligence - BI) koja je usko vezana uz našu temu. Područje kojim se bavi poslovna inteligencija je upravo analiza podataka iz skladišta korištenjem BI alata. Još neke povezane teme su big data i rudarenje podataka. Svi ti podaci koji se trebaju analizirati moraju biti spremljeni negdje, u neko skladište podataka.

Kako se danas generira više podataka nego ikad prije, a taj trend rasta će se vjerojatno i nastaviti, potrebe za skladištenjem podataka neće ponestati. Svi mi na dnevnoj razini generiramo velike količine podataka, a da toga nismo ni svjesni.

Danas većina tvrtki i organizacija koristi neki oblik skladištenja i analiziranja podataka, a one koje ne koriste će ih vjerojatno u nekom trenutku u budućnosti početi koristiti jer će im inače biti teško održati konkurentnost na tržištu.

Bibliografija

[1] *ETL Database - ETL Transform*, http://etldatabase.com/etl-transform/, datum pristupa: 2017-09-05.

[2] *ETL Database*, http://etldatabase.com/, datum pristupa: 2017-08-05.

[3] *ETL Tools - free tools*, https://www.etltools.net/free-etl-tools.html, datum pristupa: 2017-08-09.

[4] *FER - Oblikovanje skladišta podataka*, https://www.fer.unizg.hr/_download/repository/OSP1_Skocir.pdf, datum pristupa: 2017-08-21.

[5] *Microsoft TechNet - About OLAP cubes*, https://technet.microsoft.com/en-us/library/hh916536(v=sc.12).aspx, datum pristupa: 2017-08-25.

[6] *Oracle Database Data Warehousing Guide - Part III Data Movement/ETL*, https://docs.oracle.com/database/121/DWHSG/part4.htm#DWHSG8270, datum pristupa: 2017-09-02.

[7] *O'Reilly School of Technology - DBA 3: Creating a Data Warehouse*, http://archive.oreilly.com/oreillyschool/courses/dba3/DBA%203%20Creating%20a%20Data%20Warehouse%20v1.pdf.

[8] *Talend by Example*, https://www.talendbyexample.com/.

[9] *Talend Open Studio for Data Integration*, https://www.talend.com/products/data-integration/.

[10] S. Al-Sakran, *Which data warehouse should you use?*, (2016), http://www.metabase.com/blog/which-data-warehouse-to-use.

[11] M. Banek, *Oblikovanje skladišta podataka iz polustrukturiranih izvora*, Magistarska radnja, Fakultet elektrotehnike i računarstva, 2005, https://bib.irb.hr/datoteka/199596.Marko-Banek-magistarski-rad.pdf.

[12] M. Breslin, *Battle of the Giants*, (2004), http://olap.it/Articoli/Battle%20of%20the%20giants%20-%20comparing%20Kimball%20and%20Inmon.pdf.

[13] W. H. Inmon, *Building the Data Warehouse, Fourth Edition*, Wiley Publishing, Inc., 2005.

[14] W. H. Inmon, D. Strauss i G. Neushloss, *DW 2.0: The Architecture for the Next Generation of Data Warehousing*, pogl. 1, str. 1–6, Morgan Kaufmann, 2008, dostupno na: http://booksite.elsevier.com/samplechapters/9780123743190/Sample_Chapters/02~Chapter_1.pdf.

[15] R. Kimball, *A Dimensional Modeling Manifesto*, (1997), http://www.kimballgroup.com/1997/08/a-dimensional-modeling-manifesto/.

[16] R. Kimball i J. Caserta, *The Data Warehouse ETL Toolkit*, Wiley Publishing, Inc., 2004.

[17] G. Speare, *ETL vs. ELT - What's the big difference?*, (2015), https://www.ironsidegroup.com/2015/03/01/etl-vs-elt-whats-the-big-difference/.

[18] P. Williams, *A Short History of Data Warehousing*, (2012), http://www.dataversity.net/a-short-history-of-data-warehousing/.

Sažetak

Skladište podataka je baza podataka koja sadrži povijesne nepromjenjive podatke koji se prikupljaju i analiziraju kao pomoć pri donošenju poslovnih odluka. U radu je dan pregled osnovnih pojmova i razloga nastanka i korištenja skladišta podataka. Također, opisane su razlike u odnosu na klasične, transakcijske baze podataka.
Bill Inmon i Ralph Kimball imaju vrlo velik značaj u području oblikovanja skladišta podataka. Inmon je poznat kao "otac skladištenja podataka", a Kimball je tvorac dimenzionalnog modeliranja. Njih dvojica imaju različite pristupe oblikovanju skladišta, Inmon se zalaže za pristup razvoju "od vrha prema dolje", dok Kimball zastupa pristup "od dna prema gore".
U radu je dan pregled alata za fizičku realizaciju skladišta podataka te su opisani razlozi i načini nadzora i održavanja skladišta.
ETL (Extract-Transform-Load) proces je proces koji označava dohvaćanje podataka, njihovu transformaciju i učitavanje u skladište podataka. Predstavlja najznačajniji dio svakog skladišta podataka. Što se tiče alata, postoje brojni komercijalni i open-source ETL alati, a također možemo i samostalno kodirati ETL proces u proizvoljnom programskom jeziku.
Na kraju rada dajemo opis studijskog primjera u kojem demonstriramo kako dizajnirati i implementirati dimenzionalno skladište. Za implementaciju ETL procesa koristimo alat Talend Open Studio.

Summary

Data warehouse is a database used to collect large amounts of historical data which is then analyzed and used to make better business decisions. This paper gives an overview of basic terms and reasons for the creation and use of data warehouses. Also, we describe the differences between data warehouses and transactional databases.
Bill Inmon and Ralph Kimball have made a huge impact in data warehouse modeling techniques. Inmon is known as "father of data warehousing" and Kimball is known for his dimensional modeling technique.The two of them have different approaches to data warehousing, Inmon advocates a "top-down" approach, whereas Kimball suggests a "bottom-up" approach.
This paper gives an overview of tools used for physical realization of data warehouses. Also, it describes the reasons for monitoring and maintaining a data warehouse and how to do it.
ETL is a process of extracting data from source systems, transforming them and loading them into the data warehouse. It is the most important part of building a data warehouse. There are many commercial and open-source ETL tools but there is also the option of hand coding the whole ETL process.
At the end of this paper, we give an example in which we demonstrate how to model and implement a data warehouse based on a dimensional model. Our ETL tool of choice is Talend Open Studio.

Životopis

Rođena sam 18. kolovoza 1990. godine u Zagrebu te sam u istom gradu pohađala osnovnu i srednju školu. Nakon završene V. gimnazije, upisala sam Prirodoslovno-matematički fakultet, Matematički odsjek Sveučilišta u Zagrebu. Poslije završenog preddiplomskog studija, upisala sam smjer Računarstvo i matematika na istom fakultetu. Sada već gotovo godinu dana radim u mStart-u u Službi podrške Oracle Retail sustavima gdje zajedno s kolegama radim na održavanju, razvoju i integraciji poslovnih sustava koji se koriste u maloprodajnim i veleprodajnim lancima trgovina.