Poglavje 5

Podatkovna skladišča

Podatkovna baza in podatkovno skladišče

- Podobno, vendar ne enako!
- Podatkovna baza (PB oz. DB):
  - OLTP sistem (on-line transaction processing)
  - opisuje trenutno stanje
- Podatkovno skladišče (PS oz. DW):
  - OLAP sistem (on-line analytical processing)
  - opisuje zgodovino vsebin OLTP sistema (pogosto več OLTP sistemov)
- Oba pristopa tečeta na SUPB, vendar z različnimi prioritetami izvajanja (OLTP - hitro izvajanje transakcij, OLAP - hitra analiza)

Zakaj podatkovna skladišča?..

- Nosilci odločanja potrebujejo dostop do vseh podatkov v okviru poslovanja ne glede na platformo in fizično lokacijo
- Nosilci odločanja potrebujejo pregled ne le nad trenutnimi vrednostmi posameznih (zbirnih, sumariziranih) podatkov in kazalnikov, temveč pregled nad zgodovino vrednosti
- Nosilci odločanja potrebujejo tudi pregled nad trendi (rasti, padanja, stagnacija)

Zakaj podatkovna skladišča?

- Podatkovna skladišča omogočajo realizacijo podpore odločanju glede na predhodno opredeljene potrebe nosilcev odločanja in nuditi podporo morebitnim novim potrebam
- Podatkovna skladišča:
  - Hranijo usklajene podatke iz več različnih podatkovnih virov
  - Hranijo zgodovinske podatke
  - Hranijo povzete podatke (agregirane na različne načine)
- Podatkovna skladišča so idealen vir podatkov za podatkovno analitiko (podatkovno rudarjenje / strojno učenje)

Evolucija podatkovnih skladišč..

- V sedemdesetih so podjetja v svoje poslovanje začela uvajati aplikativne sisteme, ki so omogočali avtomatizacijo poslovnih in ostalih procesov na operativnem nivoju
- Posledica je bila akumulacija večjih in večjih količin podatkov v podatkovnih bazah transakcijskih sistemov
- Podjetja sedaj dajejo poudarek različnim načinom uporabe teh podatkov za podporo odločitvenim procesom z namenom pridobiti ali zadržati strateško prednost pred konkurenco

Evolucija podatkovnih skladišč..

- Nekateri transakcijski aplikativni sistemi imajo elemente, ki omogočajo podporo odločitvenim sistemom na operativnem in taktičnem nivoju (poročila, sumarni pregledi, grafi, …)
- Podjetja imajo velikokrat več transakcijskih aplikativnih sistemov, ki vsak na svoj način (včasih tudi kontradiktorno) obravnavajo iste podatke

Evolucija podatkovnih skladišč

- IS podjetij potrebujejo podatkovno skladišče, ki predstavlja arhiv podatkov, ki predstavlja vir znanja in omogoča enoten in hkrati uporabniku prilagojen integriran in konsolidiran pogled na (sumarizirane, zgodovinske, ..) podatke
- Podatkovna skladišča predstavljajo vir podatkov za različna orodja (OLAP orodja, Data Mining orodja, ..) in na ta način omogočajo podporo odločitvenim procesom v podjetjih

Kaj je podatkovno skladišče?

- Podatkovno skladišče je
  - entitetno usmerjena (subject-oriented),
  - integrirana (integrated),
  - časovno odvisna (time-variant) in
  - nespremenljiva (non-volatile)

  zbirka podatkov za namene podpore odločitvenim procesom

Entitetna usmerjenost

- Organizacija podatkovnega skladišča temelji na **glavnih entitetnih tipih** podjetja (npr. stranka, izdelek, regija, račun,…) in ne na funkcionalnih področjih oz. področjih, ki jih **pokrivajo posamezni transakcijski sistemi** (prodaja, nabava, …)

Integriranost

- Podatkovno skladišče integrira podatke iz več aplikativnih sistemov v okviru podjetja.
- Podatki iz različnih sistemov so večkrat med seboj nekonsistentni. Naloga podatkovnega skladišča je, da omogoči konsistenten in enoten pogled na podatke.

Časovna odvisnost

- Za podatek v podatkovnem skladišču je oz. mora biti poznan čas (DD:MM:YYYY HH:MIN) ali časovni interval njegove veljavnosti oz. čas, ko je bil prenesen v podatkovno skladišče
- Časovna odvisnost je večkrat prikazana v razširjenem časovnem formatu (poleg leta še npr. mesec, četrtletje, polletje), kar olajša grupiranje in povzemanje (sumarizacijo) podatkov (spomnite se na GROUP BY)
- Časovna odvisnost implicira kontinuiran zajem trenutnih stanj podatkov, kar omogoča opazovanje skozi čas (trendi, časovne vrste, modeliranje, …)

Nespremenljivost

- Podatki v podatkovnem skladišču niso podvrženi spremembam v realnem času s strani aplikacij, temveč se osvežujejo (iz transakcijskih sistemov in ostalih virov) z neko (smiselno) frekvenco
- Podatki se ob osveževanju **le dodajajo** v podatkovno skladišče (novi podatki)

- Izjemo predstavljajo strogi kontrolirani popravki, naknadne transformacije podatkov, kjer pa zagotovimo, da do ažurirnih anomalij sploh **ne more priti**

DW (OLAP) in DB (OLTP)

- Podatkovno skladišče:
  - DW – Data Warehouse
  - Optimizirano za analitično delo, pogosto spreminjanje poizvedb (ad-hoc, nepredvidljivo), agregacijo in sumarizacijo statičnih podatkov
  - OLAP – On Line Analytical Processing
- Običajna (transakcijska) podatkovna baza
  - DB – Data Base
  - Optimizirana za hitro, predvidljivo delovanje, pogosto spreminjanje podatkov
  - OLTP – On Line Transaction Processing
- Praviloma imamo za DW in OLTP različne podatkovne strežnike ali vsaj instance (odvisno od količine podatkov in obsega uporabe)

DW in OLTP..

OLTP systems	Data warehousing systems
Holds current data	Holds historical data
Stores detailed data	Stores detailed, lightly, and highly summarized data
Data is dynamic	Data is largely static
Repetitive processing	*Ad hoc*, unstructured, and heuristic processing
High level of transaction throughput	Medium to low level of transaction throughput
Predictable pattern of usage	Unpredictable pattern of usage
Transaction-driven	Analysis driven
Application-oriented	Subject-oriented
Supports day-to-day decisions	Supports strategic decisions
Serves large number of clerical/operational users	Serves relatively low number of managerial users

DW in OLTP..

- DW omogoča od enostavnih do zelo kompleksnih poizvedb ob uporabi različnih orodij in tehnologij

- Orodja, tehnologije, sistemi:
  - Reporting, poizvedovanje, razvojna orodja
  - Direktorski IS (EIS - Executive information systems)
  - OLAP orodja in aplikacije
  - Orodja za Odkrivanje zakonitosti v podatkih (Data mining tools)

DW in OLTP

- Kompleksne agregatne poizvedbe
  - **What was the total revenue for Scotland in the third quarter of 2004?**
  - What was the total revenue for property sales for each type of property in Great Britain in 2003?
  - What are the three most popular areas in each city for the renting of property in 2004 and how does this compare with the figures for the previous two years?
- Kompleksne analitične poizvedbe
  - **What is the monthly revenue for property sales at each branch office, compared with rolling 12-monthly prior figures?**
  - Which type of property sells for prices above the average selling price for properties in the main cities of Great Britain and how does this correlate to demographic data?
  - What is the relationship between the total annual revenue generated by each branch office and the total number of sales staff assigned to each branch office?
- Analitza hipotetičnih scenarijev (*if-then* analiza)
  - **What would be the effect on property sales in the different regions of Britain if legal costs went up by 3.5% and Government taxes went down by 1.5% for properties over £100,000?**

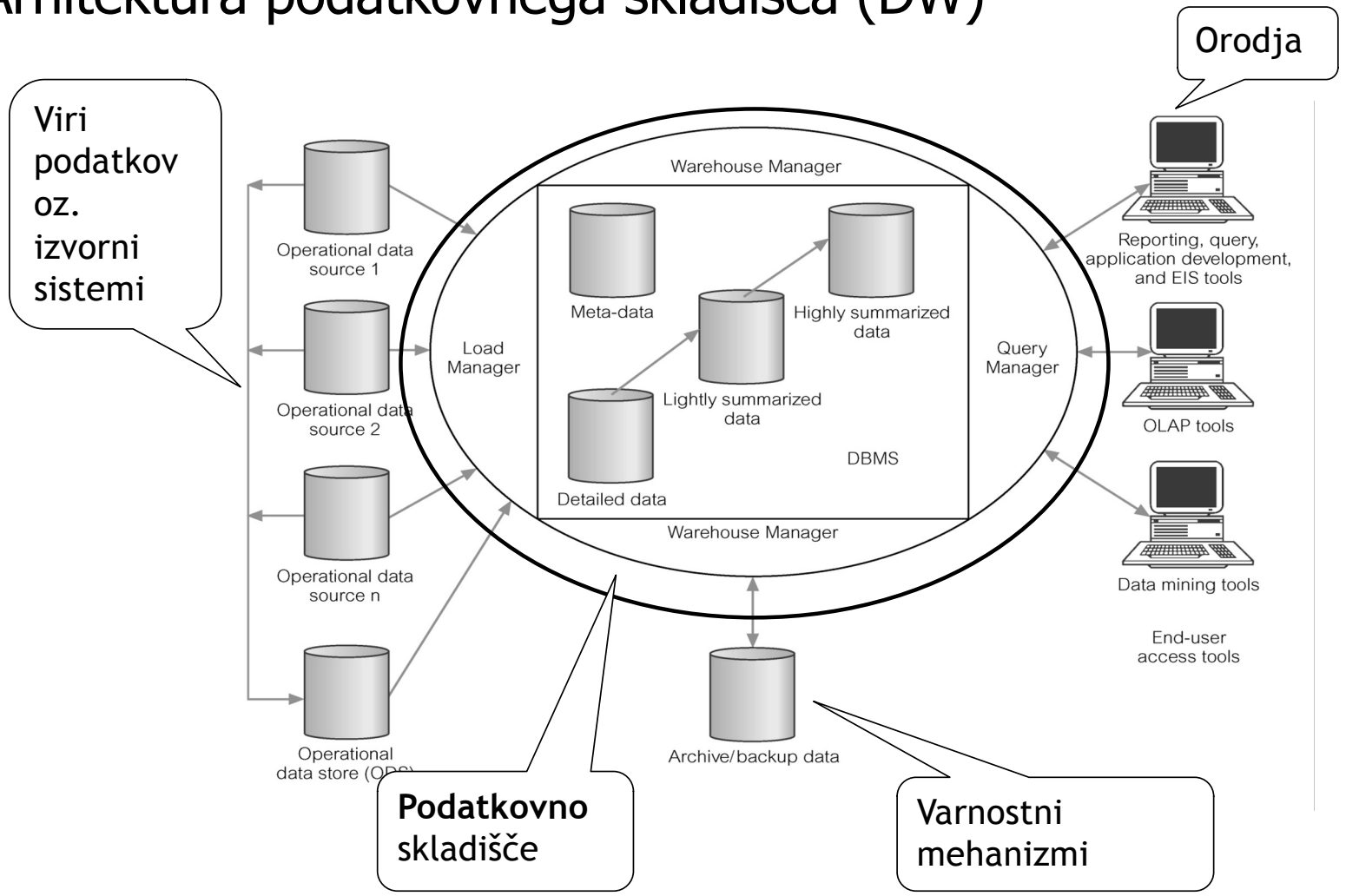
Problemi podatkovnih skladišč..

- Izgradnja in vzpostavitev DW je praviloma zapleten, drag in dolgotrajen projekt
- Podcenjevanje potrebnega časa in resursov za polnjenje podatkovnih skladišč
  - Poleg polnjenja tudi primarno pridobivanje in čiščenje podatkov v večini primerov zahtevata veliko časa
- Skriti problemi transakcijskih in drugih sistemov, ki predstavljajo podatkovni vir (izvorni sistemi)
- V izvornih sistemih se nekateri pomembni podatki niso zajemali
- Končni uporabniki
  - večja zahtevnost s strani končnih uporabnikov (v primerjavi z izvornimi sistemi)
  - nerealne zahteve in pričakovanja

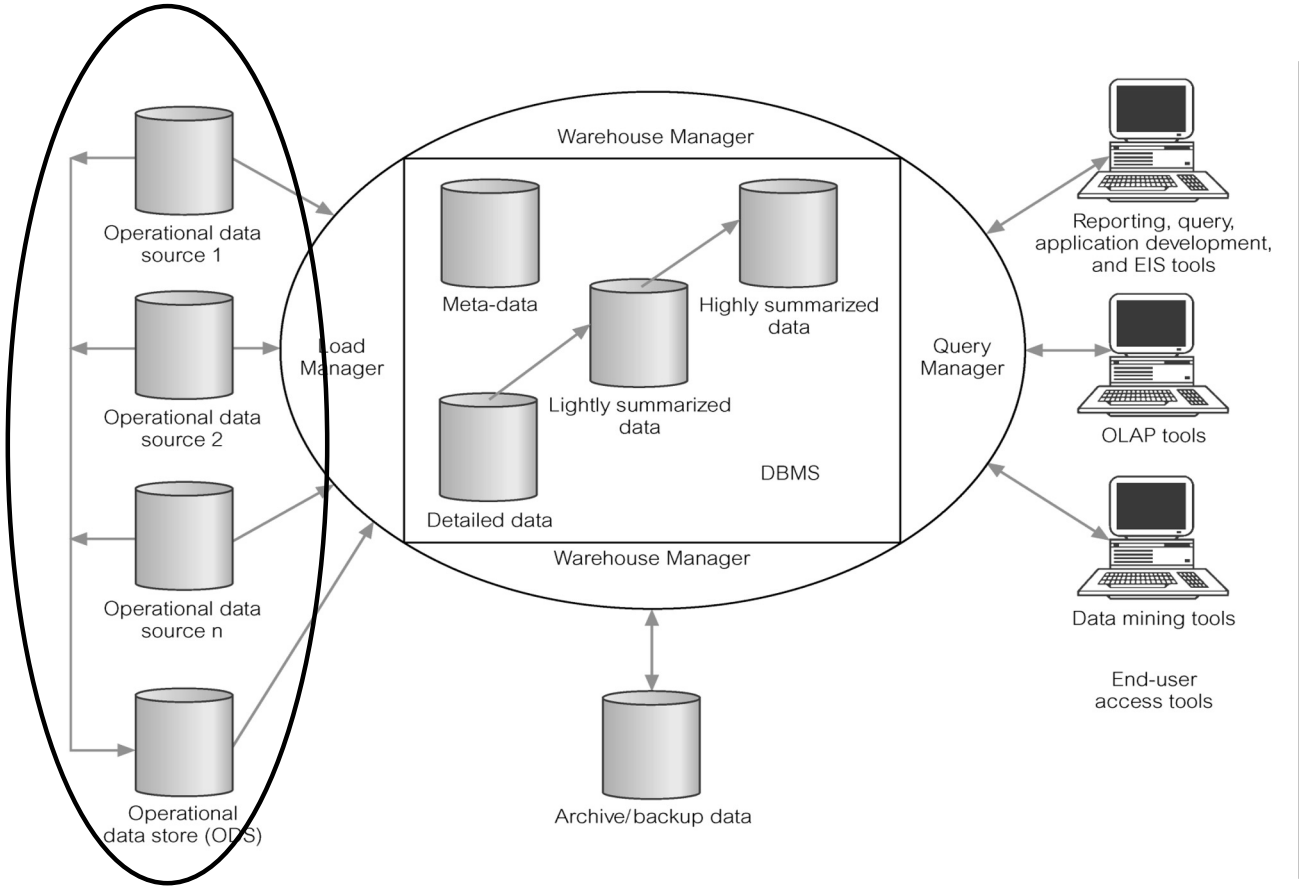
Problemi podatkovnih skladišč

- Homogenizacija podatkov in problemi pri integraciji
  - Načrtovalec DW ima včasih probleme pri združevanju podatkov iz različnih izvornih sistemov in pri tem lahko prihaja do napak
  - Skriti problemi z viri podatkov
  - Neobstoječi potrebni podatki
- Ogromna poraba diskovnega prostora
- Lastništvo podatkov
- Zahtevno vzdrževanje in zahteve uporabnikov

Arhitektura podatkovnega skladišča (DW)



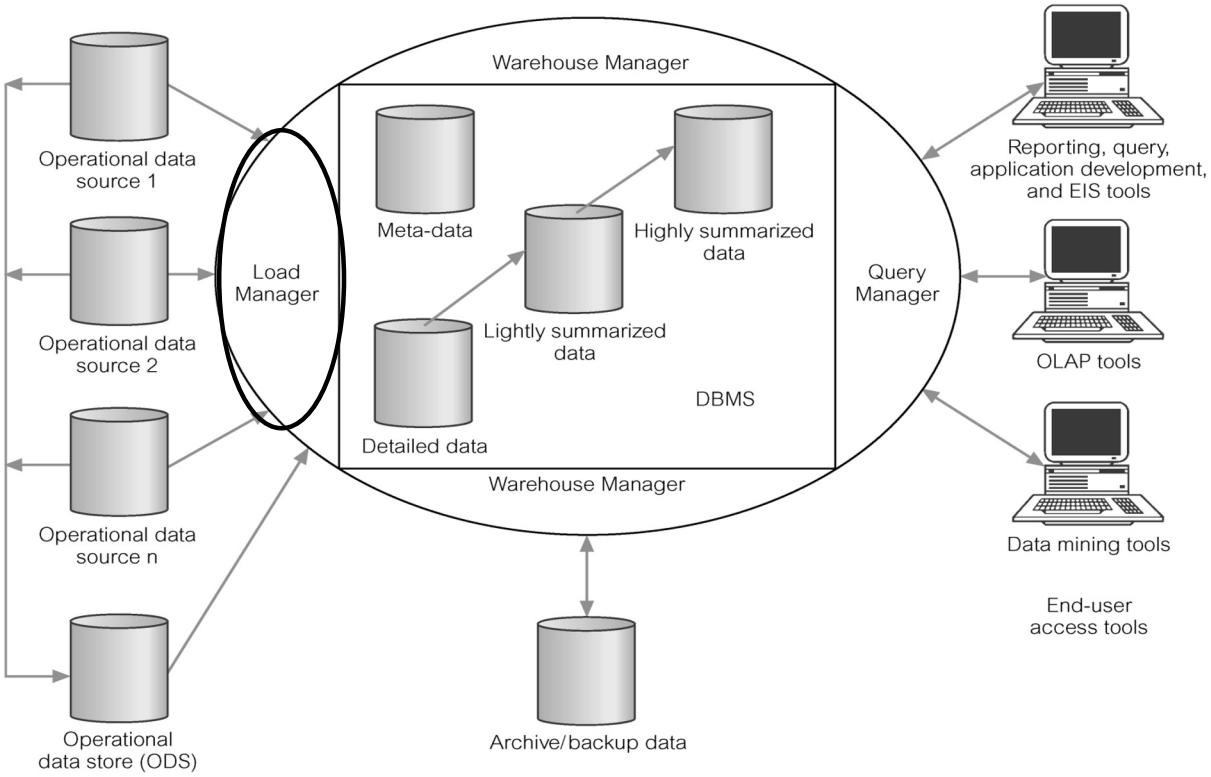
Arhitektura DW (izvorni sistemi in repozitorij)



Arhitektura DW..

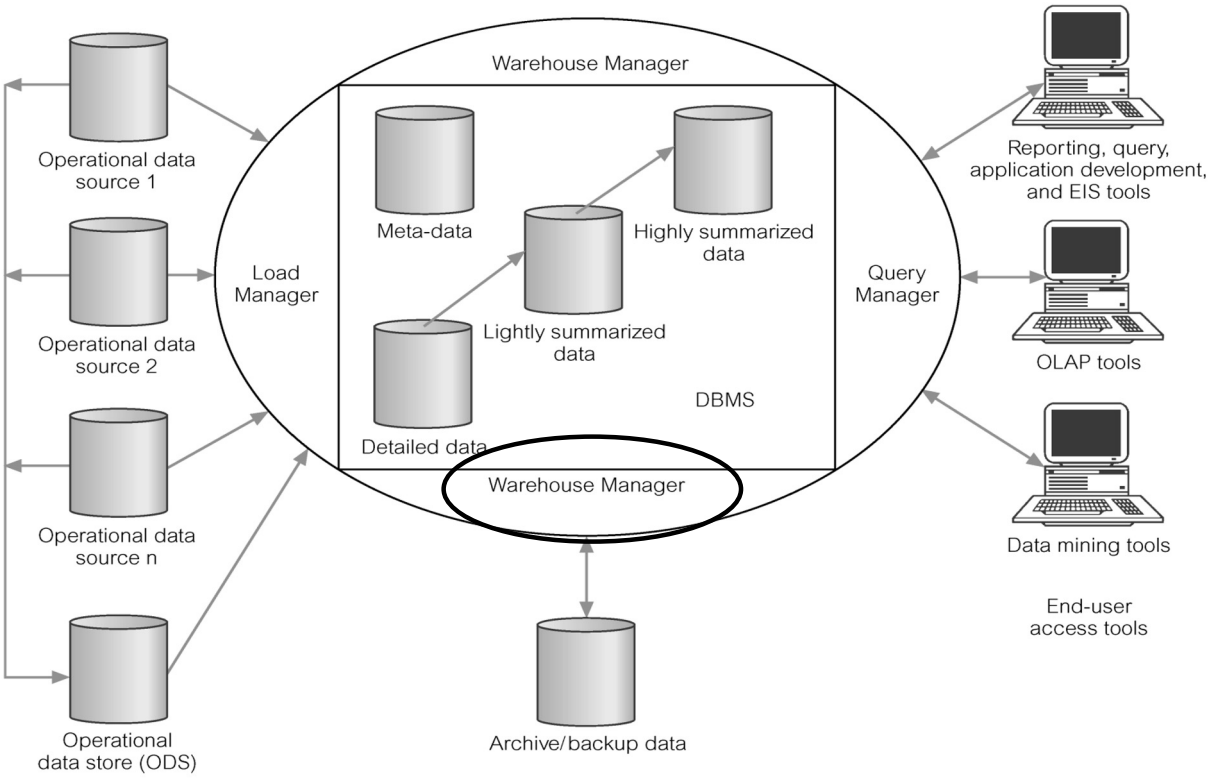
- Izvorni sistemi (Operational Data Sources):
  - Podatkovne baze transakcijskih aplikativnih sistemov
  - Nerelacijske podatkovne baze (starejše ali moderne)
  - Internetni viri
  - Ostali viri
- Repozitorij (Operational Data Store):
  - Repozitorij podatkov podatkovnega skladišča
  - Večkrat ima vlogo vmesnega člena pri prenosu podatkov v DW
  - Vzpostavljen predvsem v primerih, ko so med izvornimi sistemi stari sistemi, do katerih DW ne more direktno dostopati
  - Uporabnikom nudi poznano okolje relacijskega sistema tudi na ne-relacijskih podatkih!

Arhitektura DW (upravljalec prenosa podatkov - Load Manager)



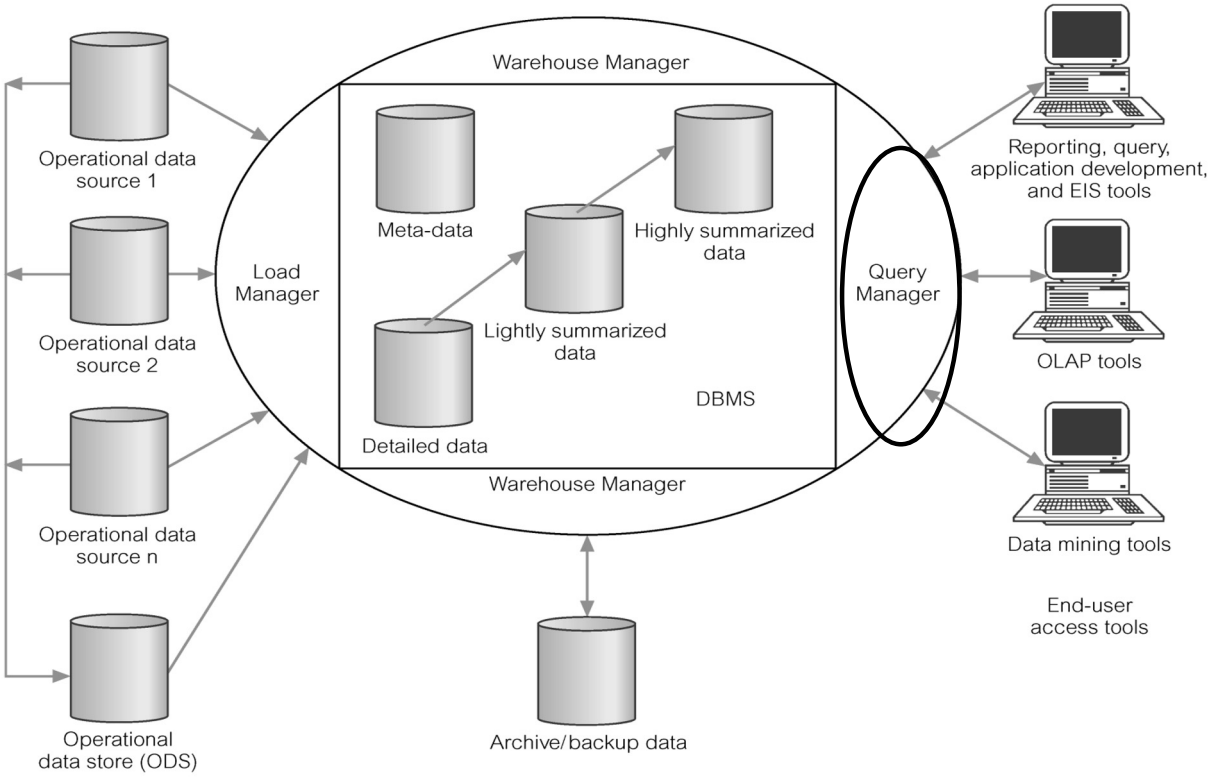
- Izvaja vse operacije vezane ekstrakcijo podatkov in prenos podatkov v DW: transformacije in čiščenje podatkov
  - **Process ETL: Extraction, Transformation, Loading**
  - Podatke prenaša bodisi direktno iz izvornih sistemov, bodisi preko repozitorija

Arhitektura DW (upravljalec DW - Warehouse Manager)



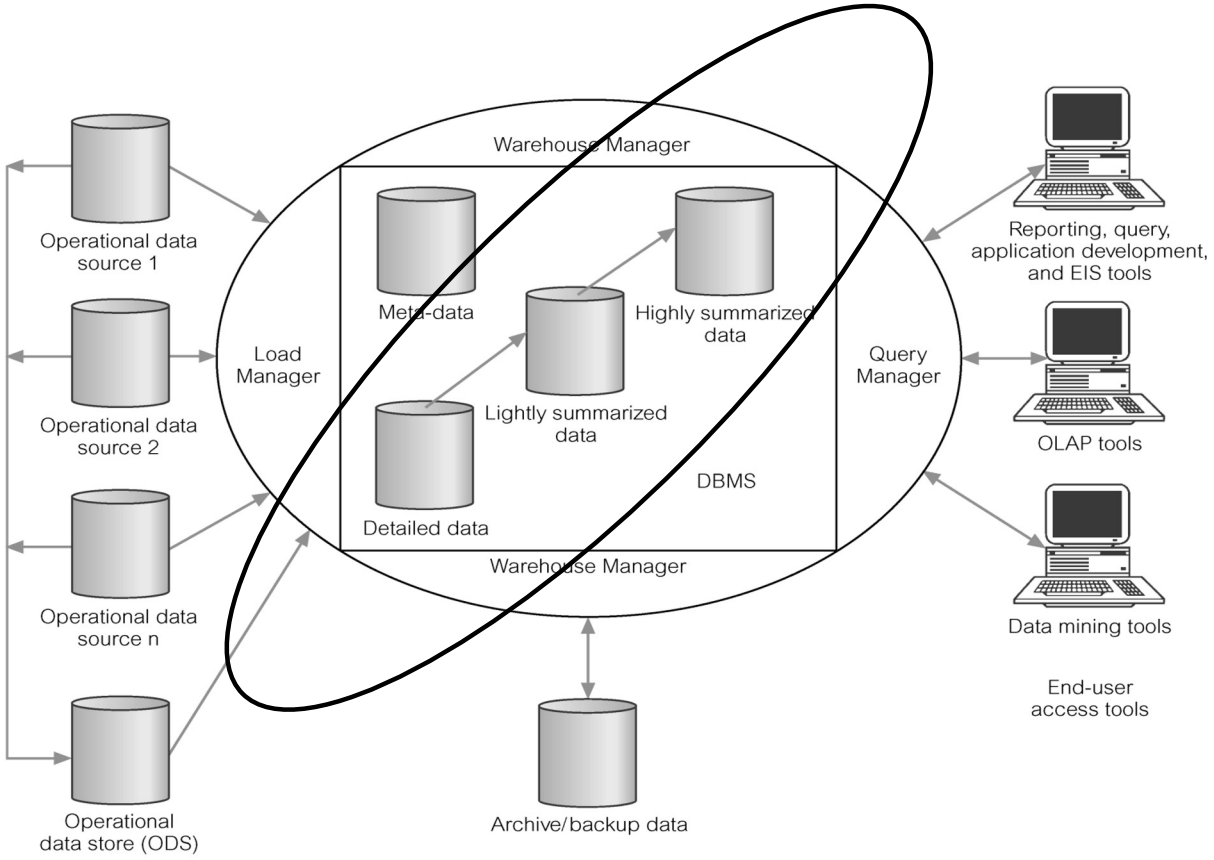
- Upravlja s podatki v DW in izvaja operacije nad podatki:
  - Analizira podatke za zagotavljanje konsistentnosti
  - Transformira podatke iz izvornih sistemov in jih združuje (integrira)
  - Kreira indekse in poglede nad tabelami (avtomatsko in ročno)
  - Generira (odkriva in kreira) denormalizacije (če je potrebno)
  - Generira agregacije (če je potrebno)
  - Izvaja arhiviranje podatkov in izdelovanja varnostnih kopij
  - Skrbi za in upošteva uporabniške profile poizvedovanja

Arhitektura DW (upravljalec poizvedb - Query Manager)



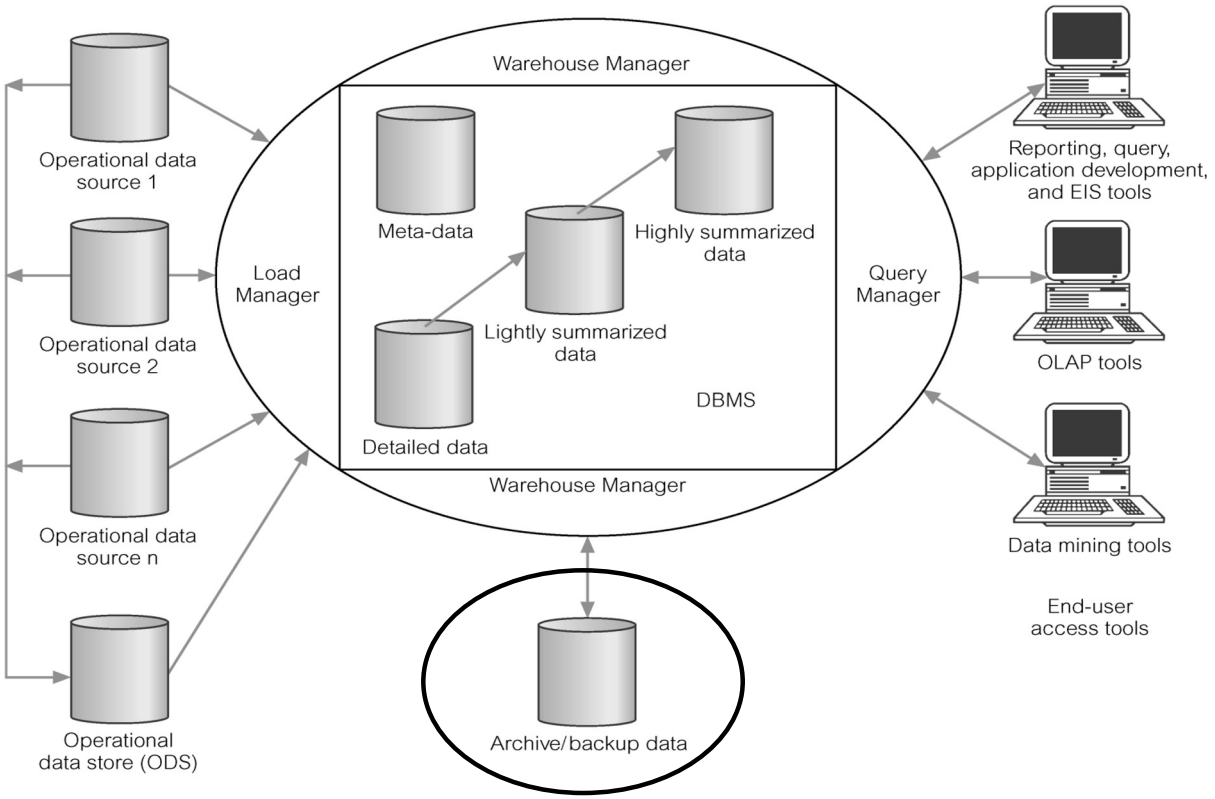
- Izvaja vse **operacije vezane na poizvedbe** uporabnikov
- Izdeluje **plane** izvajanja poizvedb in **urnike** za izvajanje poizvedb
- Izdeluje **profile poizvedb**, kar omogoča Upravljalcu DW, da določi, katere **indekse** in **agregacije** potrebuje

Arhitektura DW (trije nivoji agregiranosti podatkov)



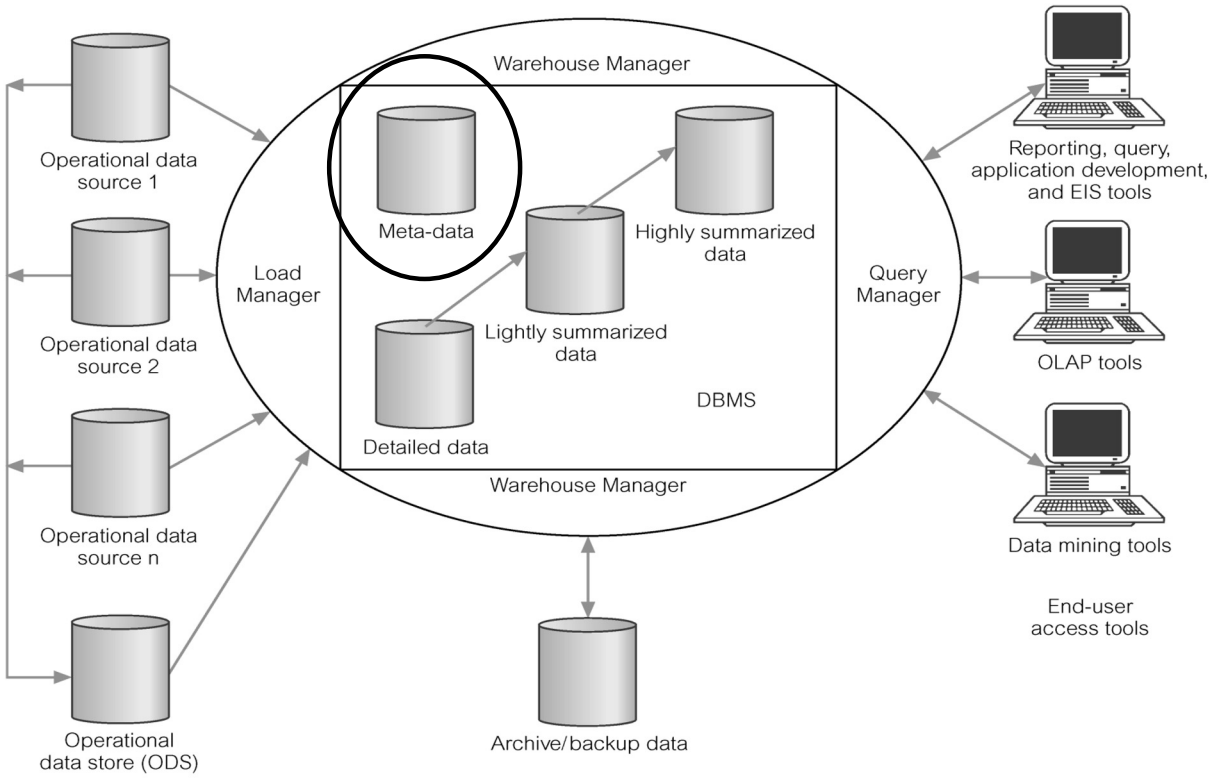
- **Podrobni** oz. transakcijski podatki (Detailed Data), ki predstavljajo vir za višja dva nivoja. Običajno ti podatki niso neposredno dostopni, ampak so vidni le skozi agregirane podatke.
- **Delno** in **visoko** agregirani podatki (Lightly and Highly Summarized Data), ki jih generira Upravljalec DW.
  - Cilj agregiranih podatkov je pospešitev izvajanja poizvedb.
  - Ta del DW se spreminja glede na spremembe profilov poizvedb.

Arhitektura DW (arhivski podatki, varnostne kopije)



- Za arhiviranje/hranjenje podrobnih in agregiranih podatkov
- Pomembno je izdelati tudi varnostno kopijo agregiranih podatkov

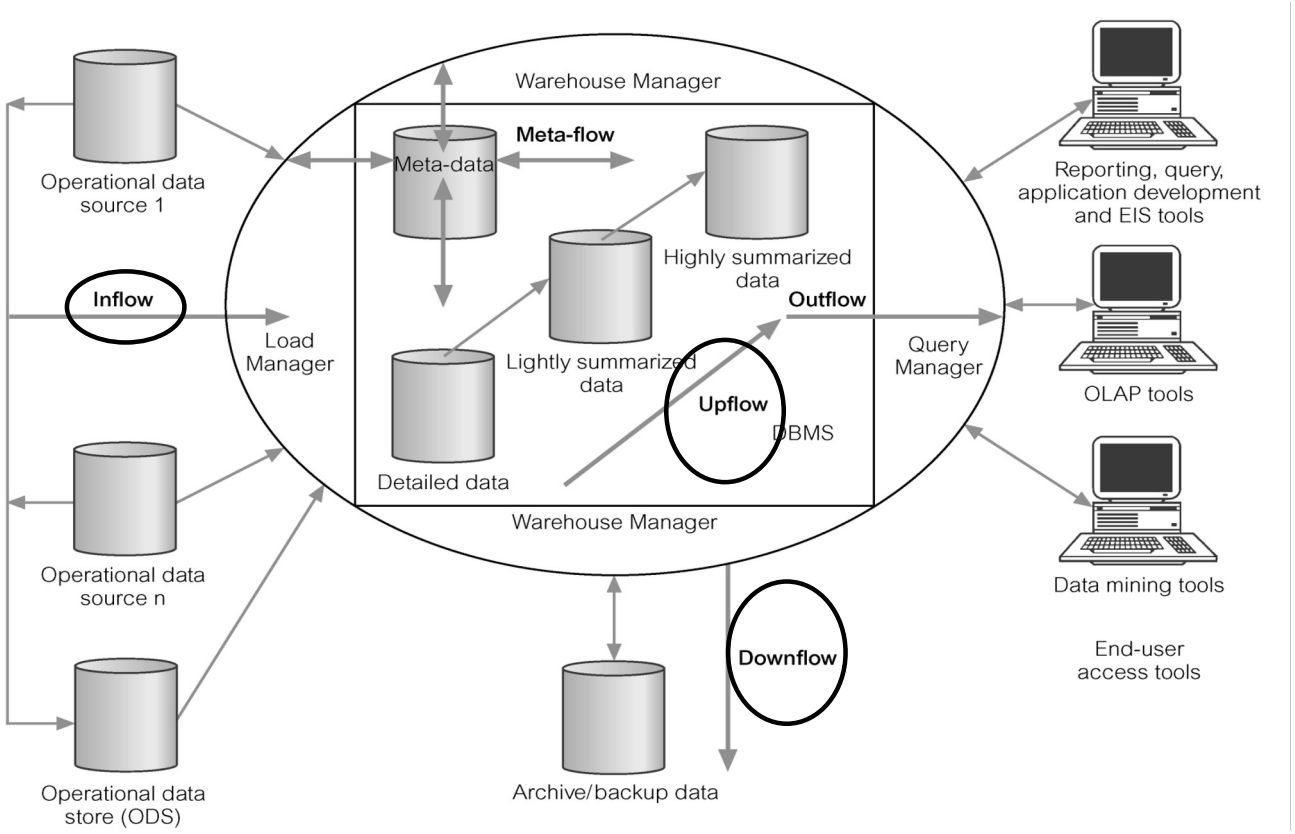
Arhitektura DW (meta-podatki, podatki o podatkih)



- Definicije podatkov v DW
- Podatki o agregacijskih tabelah
- Podatki o najprimernejših podatkovnih virih (usmerjanje poizvedb)

- Uporaba:
  - Pri ekstrakciji in polnjenju DW (polnjenje DW)
  - Pri izdelavi agregiranih tabel (upravljanje DW)
  - Pri poizvedbah, na podlagi meta podatkov se določijo najprimernejši podatkovni viri in indeksi (delovanje DW)

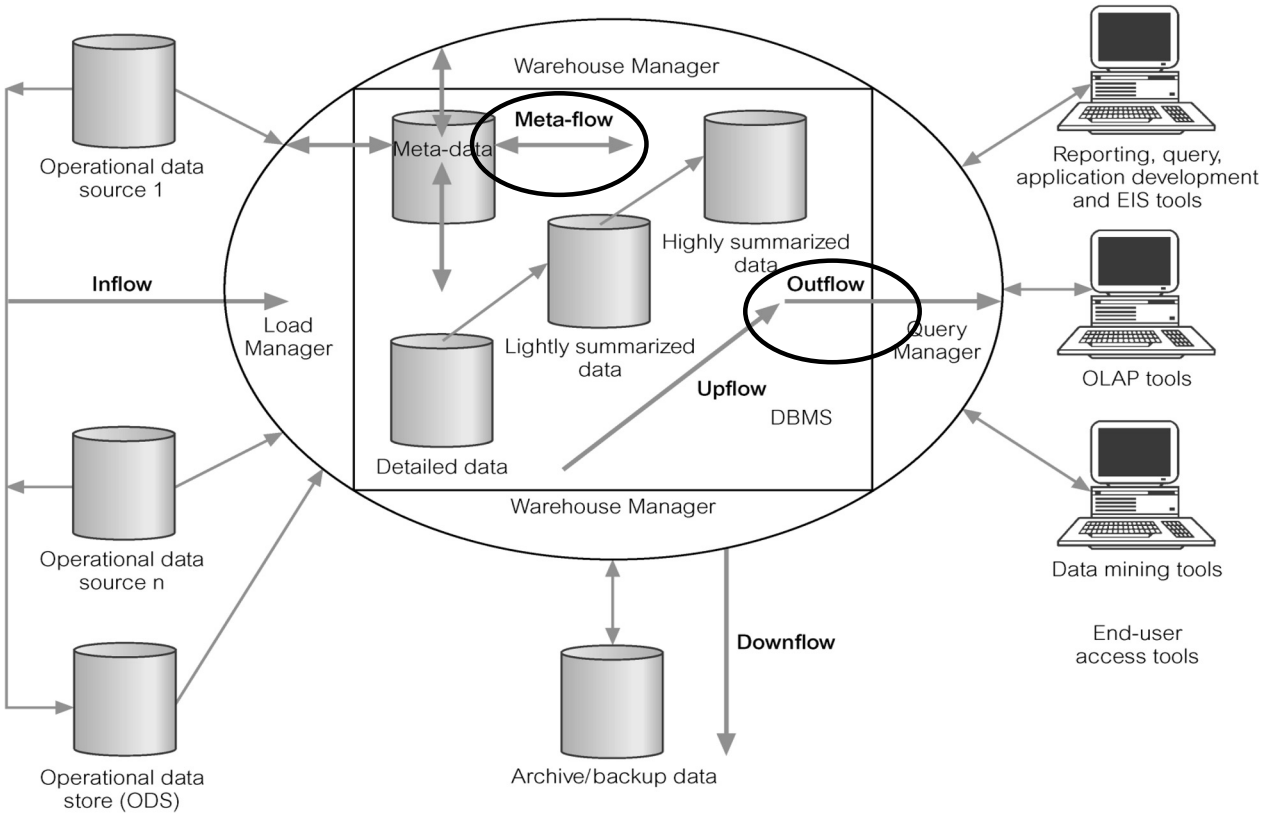
Podatkovni tokovi pri podatkovnih skladiščih: polnjenje, agregacija, arhiviranje



Podatkovni tokovi pri podatkovnih skladiščih..

- Polnjenje (Inflow): procesi, ki so povezani z ekstrakcijo, čiščenjem in prestrukturiranjem podatkov, preverjanjem njihove konsistentnosti in s samim polnjenjem DW
- Agregacija (Upflow): procesi, povezani z agregacijo podatkov in s tem dodajanem vrednosti podatkom v DW:
  - povzemanje, preoblikovanje povzetih podatkov,
  - dostava podatkov skupinam uporabnikov.
  - problem: performanse
- Arhiviranje in izdelovanje varnostnih kopij (Downflow): particioniranje tabel po času

Podatkovni tokovi pri podatkovnih skladiščih: uporaba podatkov, meta podatki



Podatkovni tokovi pri podatkovnih skladiščih

- Uporaba (Outflow): procesi, ki so povezani s pripravo in dostavo podatkov v obliko, ki je primerna za uporabnike in orodja, ki jih uporabljajo: omogočanje učinkovitega dostopa in dostava novih poslovnih objektov do končnih uporabnikov

- Upravljanje meta podatkov (Metaflow): procesi, ki so povezani z upravljanjem in uporabo meta podatkov

Orodja in tipi aplikacij, ki uporabljajo DW

- Generatorji poročil, orodja za kreiranje poizvedb (QBE – Query by Example)
- Razvojna orodja
- Transakcijski sistemi
- Direktorski aplikativni sistemi (EIS – Executive (Everybody) Information Systems)
- OLAP orodja in OLAP aplikacije
- Orodja za odkrivanje zakonitosti v podatkih (Data Mining)

Orodja in tehnologije področja DW

- Pri izgradnji in vpeljavi DW se večkrat uporabi orodja različnih proizvajalcev, kar predstavlja dodaten element v kompleksnosti procesa razvoja in vpeljave DW
  - Strogo namenski sistemi (npr. Teradata) so zelo dragi
  - Uporaba obstoječe/znane tehnologije predstavlja prednost
- Doseči "sodelovanje" orodij iz različnih virov je običajno velik izziv

DBMS za DW..

- Standardni DBMS imajo posebne dodatne funkcionalnosti oz. karakteristike, ki omogočajo učinkovite instance PB, v katerih lahko vzpostavimo DW
- Instanca PB: kompletno (navidezno) izvajalno okolje za vse operacije PB.
  - Vsaka instanca ima svoj nabor nastavitev, prilagojen njeni uporabi.
  - Isti SUPB lahko nadzoruje vec instanc.
  - V nadaljevanju so predstavljena področja, ki so še posebej pomembna za DBMS za DW oz. predstavljajo zahteve, ki jih mora izpolnjevati instanca DBMS, da je primerna za DW

DBMS za DW..

- Karakteristike polnjenja
  - Visoke zahteve za polnjenje, zahteva po sposobnosti polnjenja več sto milijonov vrstic na uro
  - Brez omejitev navzgor glede velikosti tabel
- Procesiranje pri polnjenju
  - Konverzije tipov, transformiranje, filtriranje, preverjanje konsistentnosti (lokalne in globalne)
  - Indeksiranje
  - Ažuriranje meta podatkov
  - Procesiranje mora potekati kot celota

DBMS za DW..

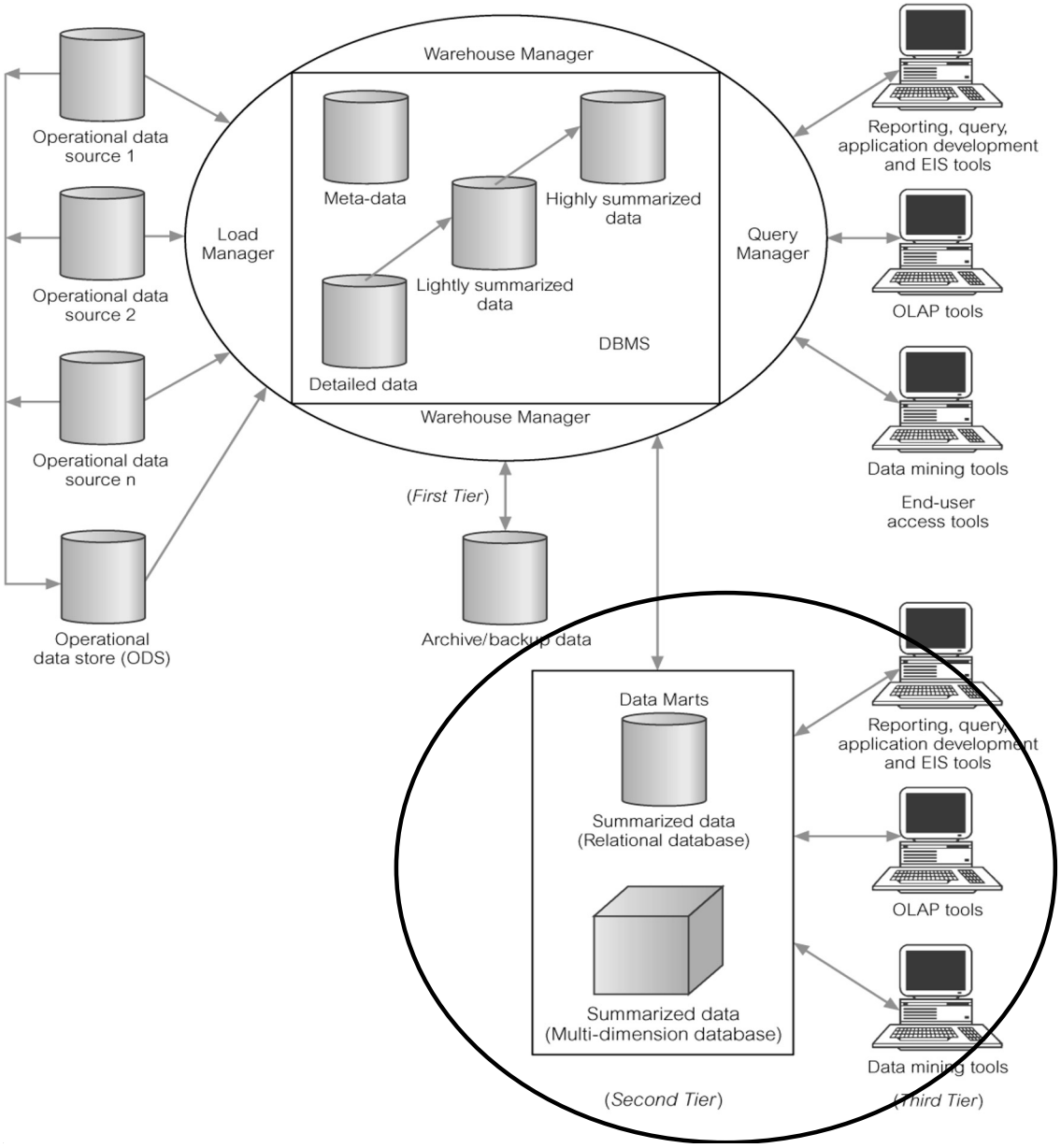
- Zagotavljanje kvalitete podatkov
  - lokalna in globalna konsistentnost
  - upoštevanje referenčnih in integritetnih omejitev ne glede na kvaliteto virov
- Odzivnost poizvedb:
  - rezultati v primernem času
  - neodvisno od velikosti baze
- Skalabilnost:
  - za upravljanje z enormnimi količinami podatkov (tera- in petabytne PB z vsemi funkcionalnimi mehanizmi)
  - možnost sočasne uporabe s strani veliko (??) uporabnikov
- DW kot distribuirana PB (ni težav z ažuriranji)

DBMS za DW

- Podpora za administracijo DW
  - beleženje storitev
  - določanje prioritete poizvedb
  - Uglaševanje/optimizacija
- Podpora več-dimenzionalnim podatkovnim strukturam (za potrebe OLAP orodij)
  - vnaprejšnja sumarizacija in agregacija podatkov
  - napredni analitični operatorji
- Razširjen nabor funkcionalnosti pri poizvedbah
- Pogosta implementacija v porazdeljenih sistemih

Področno podatkovno skladišče - Data Mart

- Del (podmnožica) DW, ki pokriva zahteve določenega oddelka ali funkcionalnega področja
- Značilnosti področnih podatkovnih skladišč:
  - Fokusiran je na zahteve določenega oddelka ali funkcionalnega področja (pogosto en OLTP sistem – en data mart)
  - Enostavnejše polnjenje
  - Praviloma ne vsebuje podrobnih transakcijskih podatkov, temveč le sumirane oz. agregirane podatke (tiste ki zadevajo ustrezno področje)

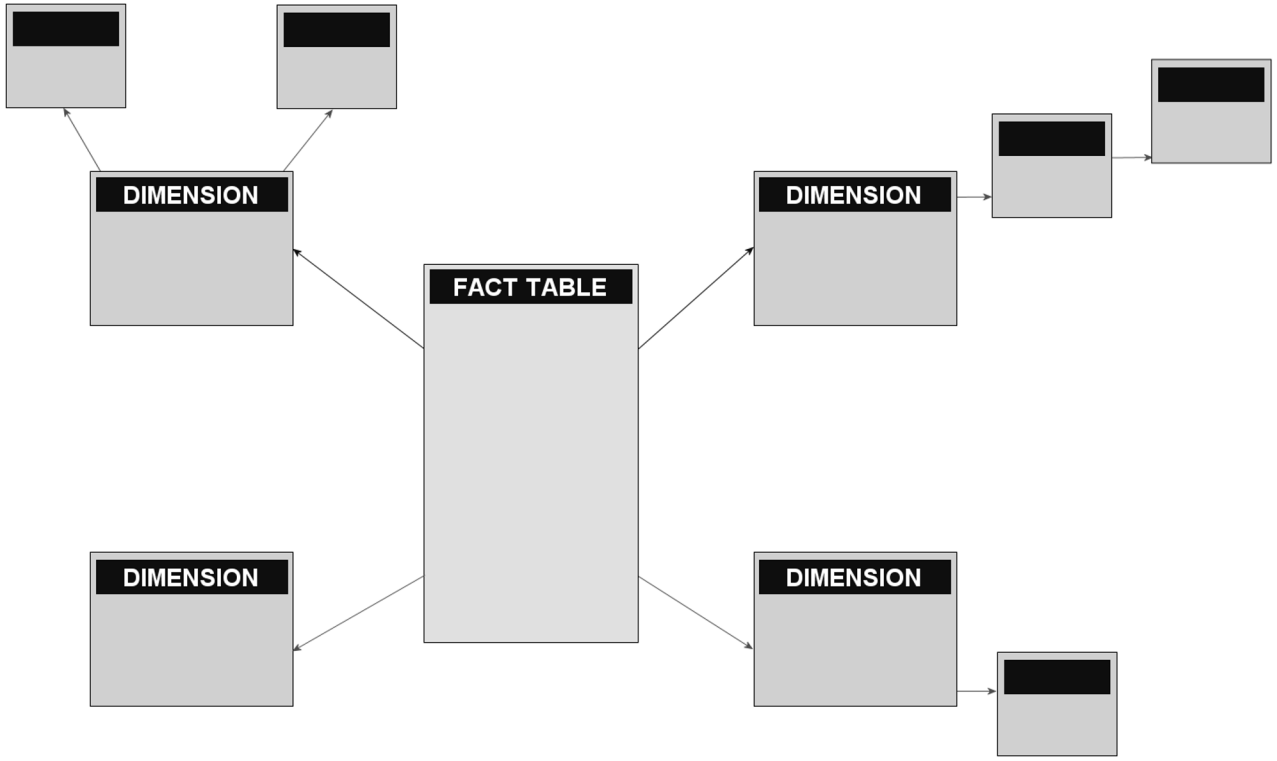


Data Mart

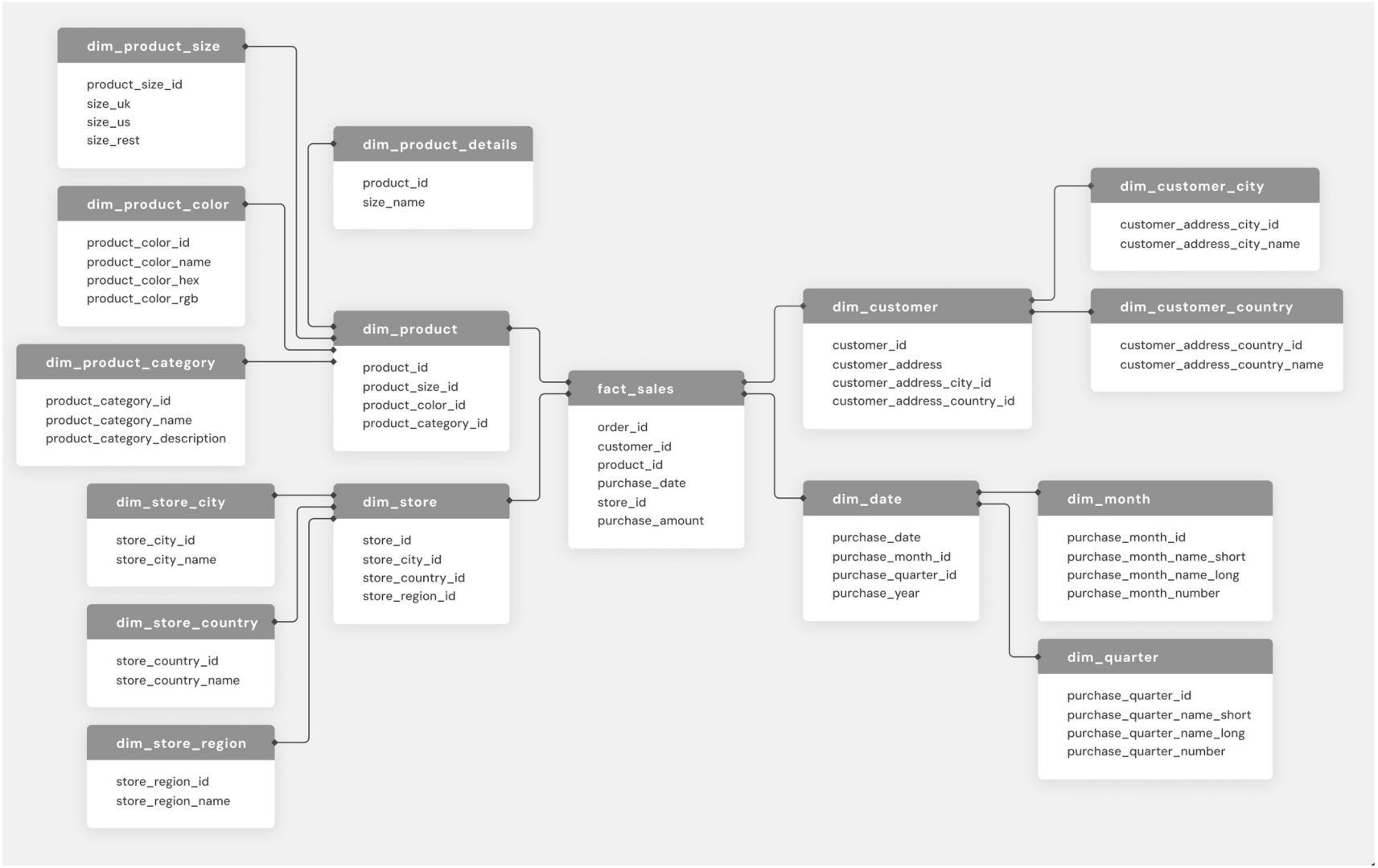
- Razlogi za kreiranje Data mart:
  - Dati uporabnikom na voljo le podatke, ki jih najpogosteje analizirajo
  - Omogočiti posameznikom ali skupinam, da opazujejo podatke (oddelka ali funkcionalnega področja) formatirane na način, na katerega so navajeni
  - Izboljšati odzivne čase uporabnikov zaradi manjše količine podatkov
- Analiza podatkov v Data martu:
  - OLAP
  - data mining

Načrtovanje podatkovnega skladišča

- Shema snežinke:
  - tabele dejstev in dimenzij
  - Ni problemov z ažurirnimi anomalijami (denormalizacija)
- Tabela dejstev:
  - Zapisi o dogodkih v času (npr. nakup)
  - Identifikator, čas in podatki
  - Povezuje dimenzije
- Dimenzija (dimension):
  - Opisujejo atribute dejstev
  - Razgradnja dimenzij za lažje analitično poizvedovanje

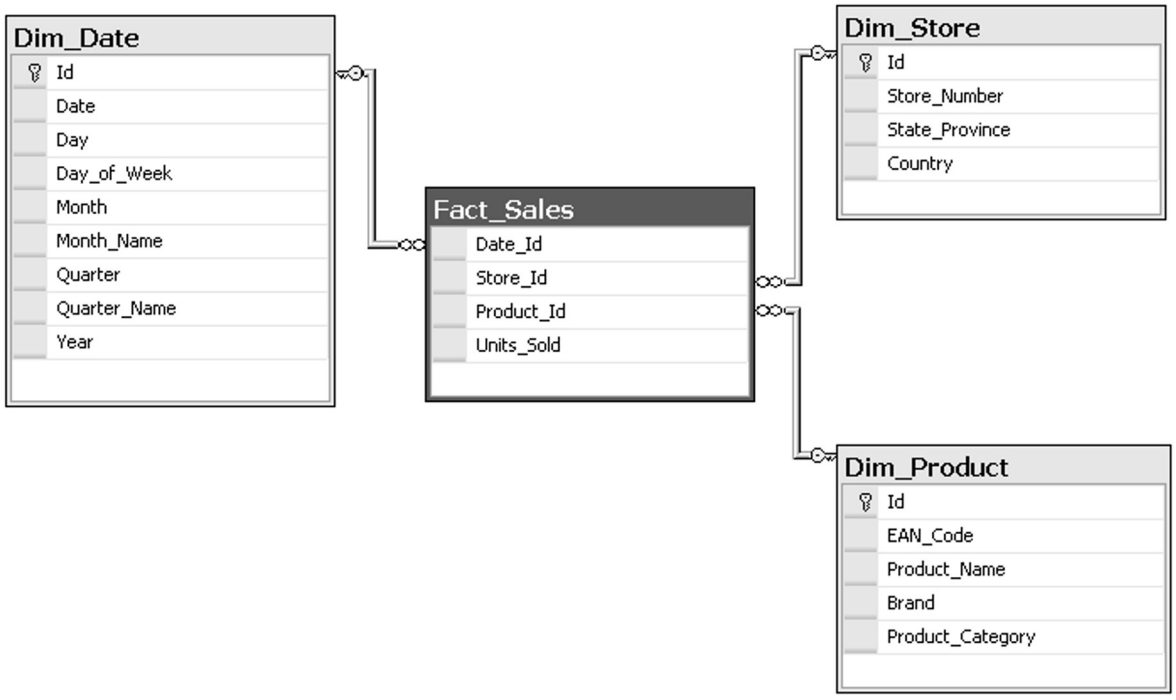


Načrtovanje podatkovnega skladišča – shema snežinke



Načrtovanje podatkovnega skladišča

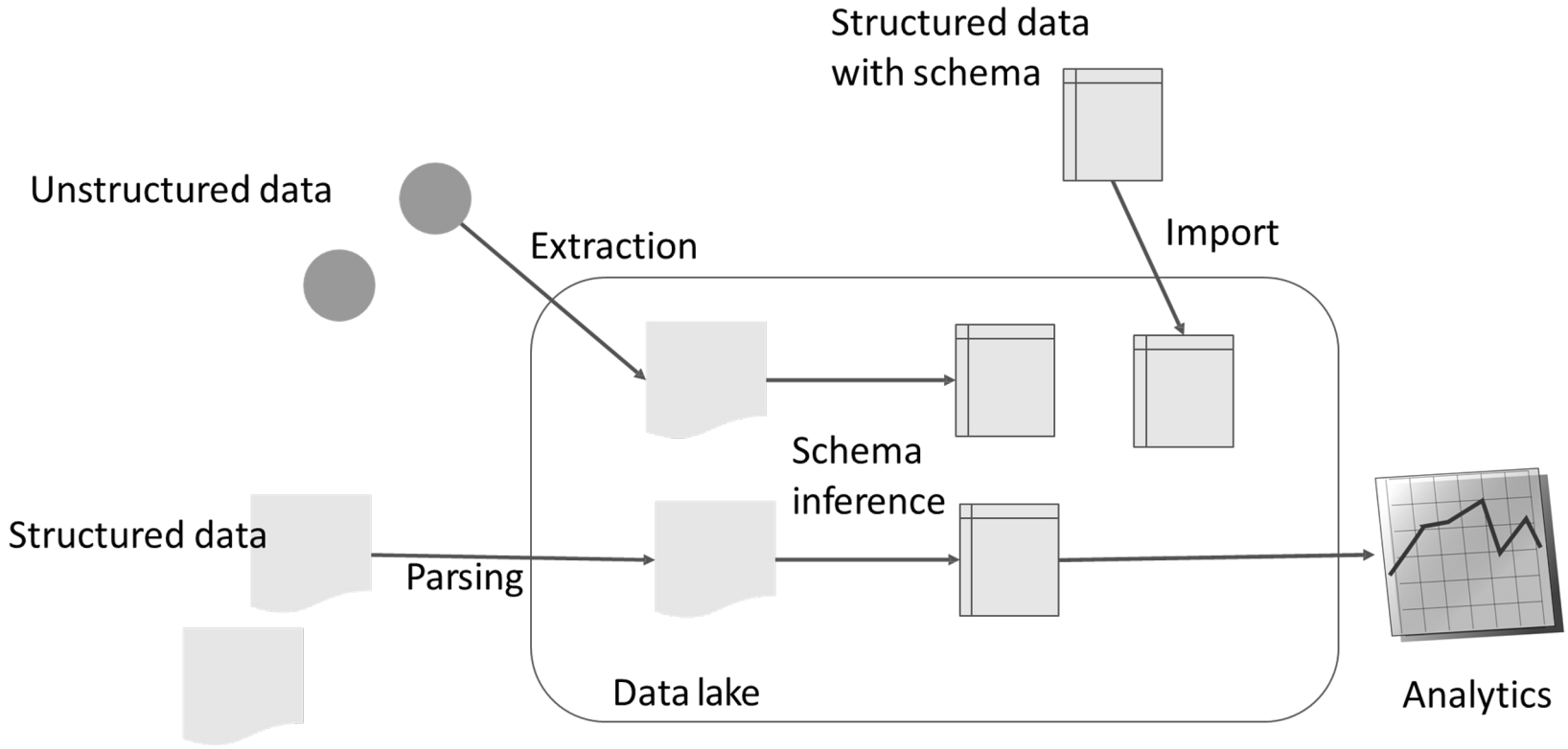
- Delček sheme snežinke: zvezdna shema
- Ali je normalizirana? V kateri normalni obliki je?



Podatkovna jezera

- Podatkovno skladišče: ETL, relacijski SUPB
  - Extraction, Transformation, Loading
  - Schema-on-write
  - Tehnični problemi: skalabilnost, togost, omejena uporabnost porazdeljene analize
- Moderne podatkovne potrebe
  - Integracija raznolikih podatkov
  - Odkrivanje relevantnih podatkov
  - Naravni jezik
  - Pričakovani tudi napačni podatki
  - Verzije podatkov skozi čas
- Podatkovno jezero: ELT, nerelacijski SUPB (pogosto objektni)
  - Extraction, Loading, Transformation
  - Schema-on-read
  - Semantični problemi: podatkovno močvirje
- Primeri orodij
  - Azure Data Lake Storage
  - AWS Lake Formation
  - Databricks
  - **MinIO** (tudi za osebno rabo)

Heterogeni podatki v modernem svetu



Analitično poizvedovanje in optimizacija

- V podatkovnih skladiščih je časovna zahtevnost poizvedovanja lahko zelo velika, zato je optimizacija poizvedb nepogrešljiva.
- Poizvedbe se optimizirajo tudi v transakcijskih relacijskih bazah.
- Algoritmi za izvajanje učinkovitih kompleksnih poizvedb.
  - Poizvedbo je moč izvesti na številne semantično ekvivalentne načine.
  - Eden od ciljev optimizacije poizvedb je najti časovno najučinkovitejši način.
- Relacijski proti nerelacijskim sistemom
  - V nerelacijskih sistemih za optimalnost poizvedovanja večinoma skrbi programer (proceduralno poizvedovanje kot del programa)
  - V relacijskih sistemih programer pove samo KAJ in ne KAKO!

Tehnike optimizacije

- Dve osnovni tehniki optimizacije (v praksi gre navadno za kombinacijo):
  - Na osnovi hevrističnih pravil, ki optimalno razvrstijo operacije poizvedbe;
  - Na osnovi primerjave relativnih stroškov različnih strategij in izbiro tiste, ki predstavlja minimalno porabo sredstev.

- V centraliziranih SUPB predstavlja dostop do sekundarnega pomnilnika največji "strošek".
- V porazdeljenih SUPB pa predstavlja dostop preko omrežja največji "strošek".

Procesiranje in optimizacija poizvedb

- Procesiranje poizvedbe zajema aktivnosti v zvezi z razčlenjevanjem, preverjanjem, optimizacijo in izvedbo poizvedbe.
  - Cilj: poizvedbo, zapisano v visoko-nivojskem jeziku, tipično SQL, pretvoriti v pravilno in učinkovito strategijo izvedbe, zapisano v nizko-nivojskem jeziku (implementacija relacijske algebre) in izvedba te strategije.

- Optimizacija poizvedbe: izbiranje najučinkovitejše izvedbene strategije za procesiranje poizvedbe.

Dva pogleda na optimizacijo

- Z optimizacijo poizvedbe skušamo izbrati tisto transformacijo poizvedbe, ki optimizira porabo časa in/ali sredstev.
- Možnosti:
  - Minimiziramo čas izvajanja poizvedbe ≈ celotni čas poizvedbe je seštevek časov, potrebnih za izvedbo posamezne operacije poizvedbe.
  - Maksimiziramo uporabo razpoložljivih sredstev, npr. z vzporednim izvajanjem operacij.

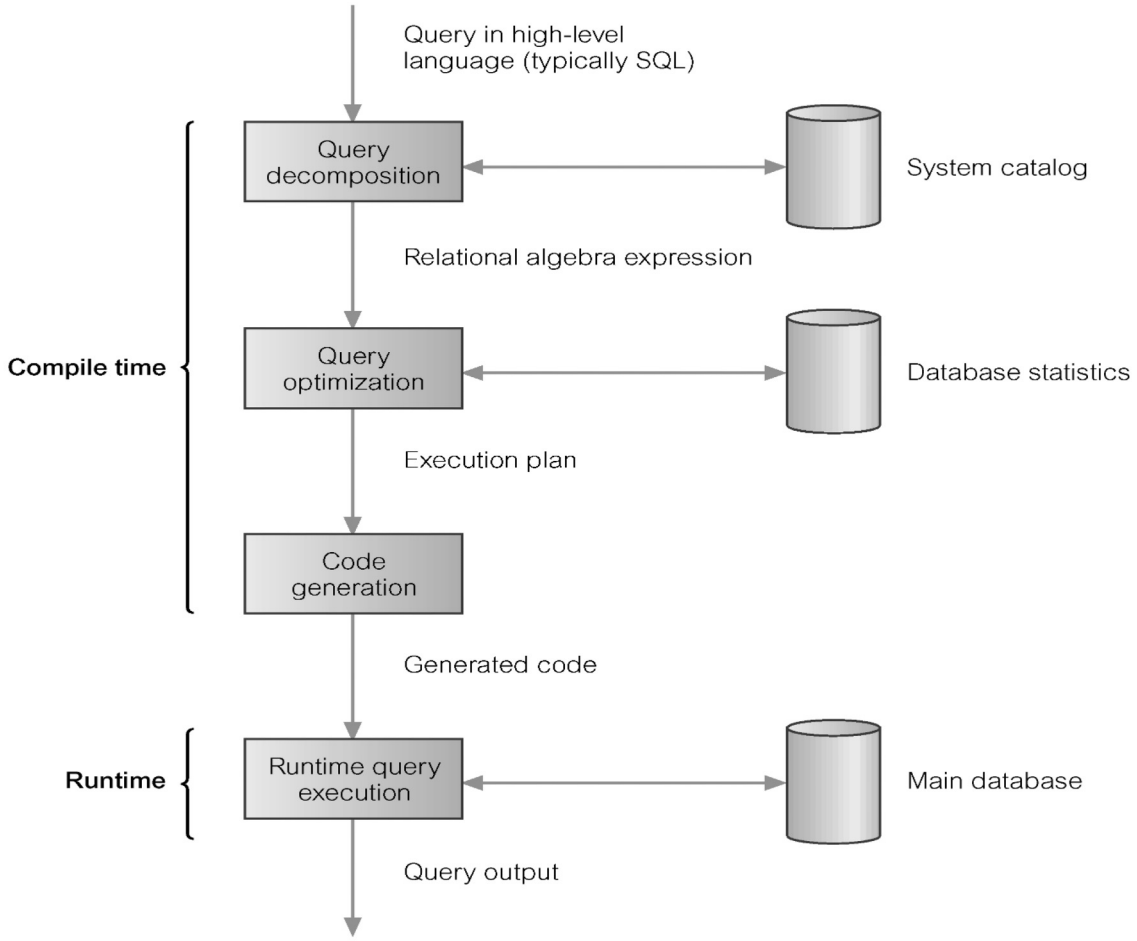
- Potrebujemo dodatne podatke, ki nam pomagajo pri optimizaciji (vključno s statistikami uporabe in značilnostmi podatkov).

Beleženje statistike

- Za izvajanje optimizacije potrebno poznati dodatne podatke – statistike
  - Statistika se nanaša na podatke o relacijah (npr. kardinalnost), atributih (npr. število različnih vrednosti) in indeksih (npr. višina indeksnega drevesa).
  - Vzdrževanje točne in "sveže" statistike je pomembno a zelo potratno… tipičen pristop – osveževanje statistike v nočnih urah.

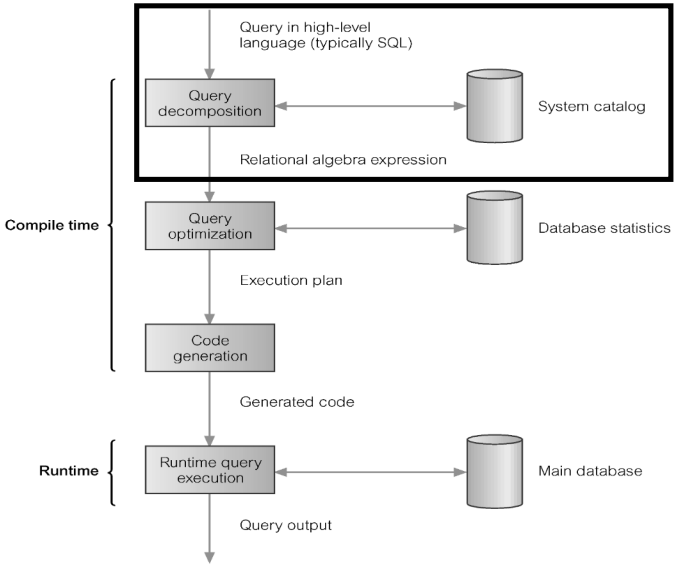
Faze procesiranja poizvedb

- Procesiranje poizvedb sestoji iz štirih faz:
  - Dekompozicija (razčlenjevanje in preverjanje);
  - Optimizacija;
  - Generiranje kode;
  - Izvedba.



Dekompozicija poizvedbe (DP)…

- Dekompozicija – prva faza procesiranja poizvedbe.
  - Cilj dekompozicije:
    - pretvoriti poizvedbo iz visoko-nivojskega jezika v relacijsko algebro.
    - Preveriti sintaktično in semantično pravilnost poizvedbe.
  - Tipični koraki dekompozicije:
    - Analiza
    - Normalizacija
    - Semantična analiza
    - Poenostavitev
    - Reorganizacija poizvedbe



DP – analiza…

- Zajema leksikalno in sintaktično analizo – podobno kot prevajalniki programskih jezikov.
- Dodatno preveri:
  - Če so relacije in atributi poizvedbe zapisani v sistemskem katalogu;
  - Če so operacije poizvedbe primerne glede na tip objekta, nad katerim se izvajajo.
- Primer

SELECT staffNumber ——————→ Ne obstaja v katalogu (staffNo)
FROM Staff
WHERE position > 10; ——————→ Nekompatibilnost operacije in podatkovnega tipa → position je tipa character

DP – normalizacija...

- Zajema pretvorbo poizvedbe v normalizirano obliko, ki jo je lažje obdelovati.
  - Gre za pretvorbo predikatnega dela (WHERE, JOIN, ...).

- S pomočjo transformacijskih pravil se poizvedba pretvori v eno izmed dveh možnih oblik:
  - Konjunktivna normalna oblika: zaporedje konjunkcij, povezanih z operatorjem AND ($\wedge$).
  - Disjunktivna normalna oblika: zaporedje disjunkcij, povezanih z operatorjem OR ($\vee$).

DP – semantična analiza...

- Namen semantične analize je preprečiti normalizirane poizvedbe, ki so napačno formulirane ali kontradiktorne.
  - Napačna formulacija: posamezne komponente ne prispevajo k ciljnemu rezultatu.
  - Kontradiktornost: predikatu ne zadošča nobena n-terica. Npr. (position = 'Manager' **AND** position = 'Assistant').

- Kontradiktornost sistemi SUPB obravnavajo in rešujejo na različne načine.

DP – poenostavitev…

- Zajema
  - detekcijo odvečnih kvalifikatorjev,
  - odpravo ponavljajočih sklopov in
  - transformacijo poizvedbe v semantično ekvivalentno, vendar enostavnejšo in za računanje učinkovitejšo obliko!

- V tej fazi se upoštevajo omejitve dostopa, definicije pogledov ter omejitve skladnosti, kar včasih botruje pojavi redundance.

Dinamična in statična optimizacija…

- Dve možnosti, kdaj se izvedejo prve tri faze procesiranja poizvedbe:
  - Dinamično procesiranje: dekompozicija in optimizacija vsakokrat, ko se požene poizvedba.
    - Prednost: statistika je "sveža"
    - Slabost: časovna potratnost zaradi vsakokratnega razčlenjevanja, preverjanja in optimizacije pred izvedbo. Zaradi prevelike časovne zahtevnosti se včasih preveri samo nekaj strategij, kar lahko privede do neoptimalne izbire…

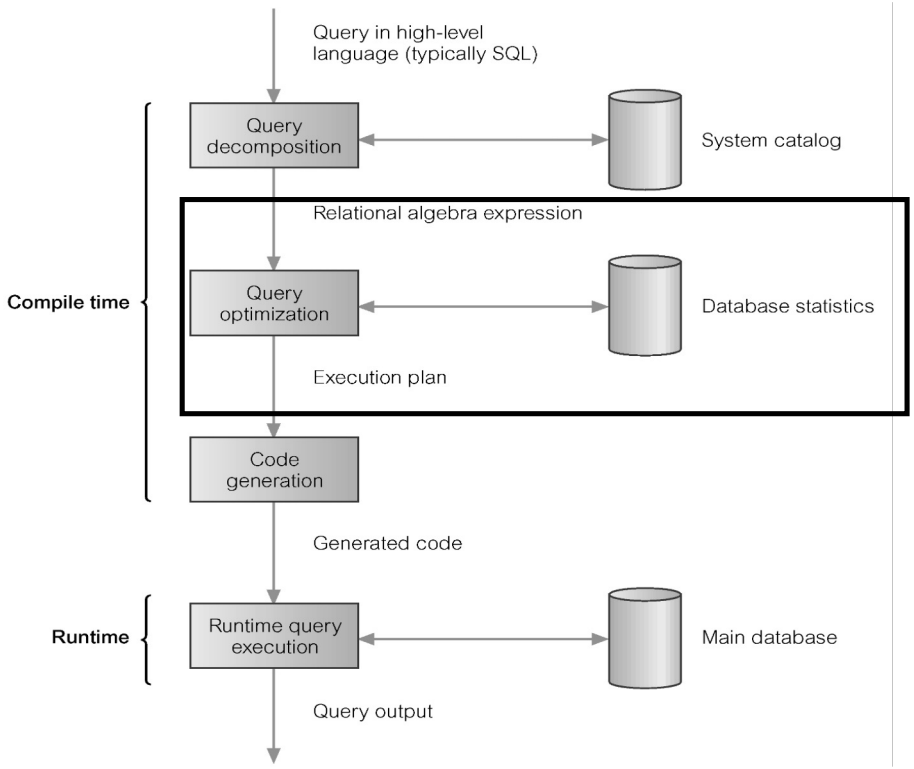
Dinamična in statična optimizacija

- Dve možnosti, kdaj se izvedejo prve tri faze procesiranja poizvedbe (nadaljevanje):
  - Statično procesiranje: dekompozicija in optimizacija se za določeno poizvedbo izvede samo enkrat.
    - Prednost: ker se dekompozicija in optimizacija ne izvedejo, več časa za preverjanje različnih strategij.
    - Slabost: stara statistika. Reševanje – (I) reoptimizacija, ko sistem zazna večje spremembe v statistiki; (II) optimizacija enkrat za vsako sejo.

Hevristična optimizacija poizvedb

- Eden od pristopov optimizacije poizvedb je uporaba hevrističnih pravil.
- S pomočjo hevrističnih pravil lahko nek izraz v relacijski algebri r_1 transformiramo v ekvivalenten izraz r_2, pri čemer je r_2 učinkovitejši.

- Optimizacija je faza, ki sledi fazi dekompozicije.



Semantična optimizacija…

- Gre za alternativen pristop k optimizaciji
  - na osnovi omejitev, ki so določene v shemi podatkovne baze, zmanjša iskalni prostor.
  - Lahko se uporablja v kombinaciji z drugimi pristopi optimizacije.

Uporaba pomožnih orodij

- Orodja za profiliranje (odkrivanje ozkih grl)
- MySQL slow query log
- Ukaz v SQL (MySQL):
  - EXPLAIN, EXPLAIN EXTENDED
  - SHOW STATUS
  - SHOW PROCESSLIST
- Sistemsko (ne)odvisni namigi za optimizacijo
- MySQL:
  - SELECT/INSERT/UPDATE … NAMIG ….
  - SELECT/INSERT/UPDATE /*! NAMIG */ …
  - podobno tudi Oracle (znotraj komentarjev)

MySQL: namigi za optimizacijo

- SQL_NO_CACHE, SQL_CACHE
  - eksplicitno povemo, katere poizvedbe se predpomnijo in katere ne
  - MySQL Query Caching mora biti v eksplicitnem načinu (set query_cache_type = 2)
- HIGH_PRIORITY
  - poizvedba z visoko prioriteto (SELECT/INSERT)
- LOW_PRIORITY
  - poizvedba z nizko prioriteto (UPDATE/INSERT),
  - počaka dokler nihče več ne bere iz tabele, ki se spreminja
  - vrne rezultat (status) šele ob dejanski izvedbi operacije

MySQL: namigi za optimizacijo

- INSERT DELAYED
  - kot INSERT LOW_PRIORITY, vendar program nadaljuje takoj (čeprav dejanska sprememba še vedno čaka)
  - deluje samo na MyISAM tabelah
- STRAIGHT_JOIN
  - stakne tabele v vrstnem redu iz FROM stavka (brez optimizacije vrstnega reda)
  - koristna souporaba z ukazom EXPLAIN
  - s slabim vrstnim redom lahko povzočimo ogromno porabo časa
- SQL_BUFFER_RESULT
  - rezultat poizvedbe se shrani v začasno tabelo
  - tabele se odklenejo že med pošiljanjem rezultatov povpraševalcu

MySQL: namigi za optimizacijo

- SQL_BIG_RESULT
  - uporaba pri DISTINCT in GROUP BY ukazih
  - MySQL bo uporabil disk za začasno shranjevanje in sortiranje
- SQL_SMALL_RESULT
  - obratno kot SQL_BIG_RESULT
  - MySQL bo uporabil hitre začasne pomnilniške tabele za sortiranje
  - običajno optimizator sam izbere ta namig
- USE INDEX, IGNORE INDEX, FORCE INDEX
  - eksplicitno povemo, katere indekse naj uporabi oziroma izpusti

MySQL: EXPLAIN

Sintaksa: **`EXPLAIN poizvedba ...`**

Rezultat stavka EXPLAIN bo prikazal:

- Vrstni red skeniranja tabel
- Vrsto uporabljenega združevanja (npr. zanke z vgnezdenimi zankami).
- Uporabo indeksov
- Ocenjeno število vrstic, ki jih je treba pregledati za vsako tabelo.

Na podlagi rezultatov stavka EXPLAIN lahko:

- Dodatmo ali spremenimo indekse za povečanje hitrosti iskanja.
- Prepišemo poizvedbo, da uporablja učinkovitejše stike ali filtriranje.
- Prilagodimo konfiguracijo MySQL, da bo bolje izkoristila razpoložljive vire.
- Končni cilj je zmanjšati število pregledanih vrstic in se izogniti dragim operacijam, kot je npr. preiskovanje ali sortiranje celotnih tabel.

MySQL: EXPLAIN

- Pomembni stolpci v izpisu
  - **type**: Označuje, kako je tabela dostopna (npr. ALL pomeni celotno skeniranje tabele, range pomeni skeniranje po obsegu indeksa).
  - **key**: Označuje, kateri indeks se uporablja, če sploh.
  - **rows**: Ocenjeno število vrstic, ki jih je treba pregledati. Visoka vrednost lahko kaže na ozko grlo.
  - **extra**: Dodatne informacije, kot so "Using filesort" ali "Using temporary", ki lahko kažejo na težave z zmogljivostjo.

MySQL: EXPLAIN

Stolpec type – možne vrednosti:

- **System, const**: Tabela ima največ eno ujemajočo se vrstico, ki jo preberemo na začetku poizvedbe in se obravnava kot konstanta.
- **eq_ref**: Za vsako kombinacijo vrstic iz prejšnjih tabel natančno ena vrstica iz te tabele z uporabo UNIQUE ali PRIMARY KEY indeksa.
- **ref**: Za vsako kombinacijo vrstic iz prejšnjih tabel vse ujemajoče se vrstice iz te tabele na podlagi iskanja po ne-UNIQUE indeksu.
- **fulltext**: Združitev se izvede z uporabo indeksa FULLTEXT.
- **ref_or_null**: Podobno kot ref, vendar MySQL dodatno išče vrstice z NULL.

MySQL: EXPLAIN

- **index_merge**: Združitev uporablja kombinacijo več indeksov, kjer se rezultati združijo.
- **unique_subquery**: Podobno kot eq_ref, vendar se uporablja za podpoizvedbe z IN, ki vračajo enolične vrednosti iz primarnega ključa.
- **index_subquery**: Podobno kot unique_subquery, vendar deluje na ne-enoličnih indeksih v podpoizvedbah z IN.
- **range**: Pridobijo se samo vrstice, ki so znotraj določenega obsega, z uporabo indeksa
- **index**: Pregleda se celotno drevo indeksa, vendar se ne pregleda dejanska tabela (podatki se pridobijo neposredno iz indeksa).
- **all**: Izvede se celoten pregled tabele (brez uporabe indeksa) za vsako kombinacijo vrstic iz prejšnjih tabel, kar je običajno najslabše za zmogljivost.

MySQL: EXPLAIN

```
EXPLAIN SELECT count(*)
FROM player p, alliance a, tribe t, village v
WHERE p.aid=a.aid and p.tid=t.tid and p.pid=v.pid;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	a	index	PRIMARY	PRIMARY	4		1321	Using index
1	SIMPLE	p	ref	PRIMARY,tid,aid	aid	4	travian.a.aid	4	
1	SIMPLE	t	eq_ref	PRIMARY	PRIMARY	4	travian.p.tid	1	Using index
1	SIMPLE	v	ref	pid	pid	4	travian.p.pid	1	Using index