

UNIVERZA V LJUBLJANI  
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

David Možina

**STOLPČNO USMERJENI SISTEMI ZA UPRAVLJANJE  
PODATKOVNIH BAZ**

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE  
STOPNJE  
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Damjan Vavpotič

Ljubljana, 2013



Rezultati diplomskega dela so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljane ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.





Št. naloge: 00422/2013

Datum: 05.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DAVID MOŽINA**

Naslov: **STOLPČNO USMERJENI SISTEMI ZA UPRAVLJANJE PODATKOVNIH BAZ**

**COLUMN-ORIENTED DATABASE MANAGEMENT SYSTEMS**

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V okviru diplomske naloge predstavite delovanje stolpčno usmerjenih sistemov za upravljanje podatkovnih baz (SUPB). Primerjajte jih z običajnimi relacijskimi SUPB ter predstavite načine za shranjevanje, ažuriranje in brisanje podatkov v obeh tipih SUPB. Posebej se posvetite poizvedovanju ter primerjajte delovanje obeh tipov SUPB za primere različnih poizvedb. Preučite v kakšnih okoliščinah stolpčni SUPB prinašajo prednosti. V drugem delu naloge izdelajte primeren podatkovni model ter ga uporabite za izdelavo sheme podatkovne baze v običajnem relacijskem in stolpčno usmerjenem SUPB. Tako pripravljeni podatkovni bazi napolnite s testnimi podatki, nato pa primerjajte hitrost izvajanja poizvedb v obeh primerih. Rezultate kritično ovrednotite.

Mentor:

doc. dr. Damjan Vavpotič

Dekan:

prof. dr. Nikolaj Zimic





## IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani **David Možina**, z vpisno številko **63090320**, sem avtor diplomskega dela z naslovom:

**Stolpčno usmerjeni sistemi za upravljanje podatkovnih baz**

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom **doc. dr. Damjana Vavpotiča**,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne \_\_\_\_\_ Podpis avtorja: \_\_\_\_\_



## **Zahvala**

Iskreno se zahvaljujem mentorju doc. dr. Damjanu Vavpotiču za spodbujanje in pomoč pri izdelavi diplomskega dela.

Rad bi se zahvalil tudi staršem, ki so me podpirali in verjeli vame v času študija.

Posebna zahvala gre tudi podjetju SRC d.o.o., ki mi je s štipendiranjem omogočilo študij v Ljubljani.



# Kazalo

## SLOVAR POJMOV

## POVZETEK

## ABSTRACT

|          |                                                                             |           |
|----------|-----------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Uvod.....</b>                                                            | <b>1</b>  |
| 1.1      | Podatkovna baza .....                                                       | 2         |
| 1.2      | Relacijski podatkovni model .....                                           | 3         |
| 1.2.1    | SQL .....                                                                   | 5         |
| <b>2</b> | <b>Potreba po stolpčno usmerjeni podatkovni bazi.....</b>                   | <b>7</b>  |
| 2.1      | Zakaj stolpčno usmerjena podatkovna baza .....                              | 7         |
| 2.2      | Zakaj stolpčno usmerjena podatkovna baza še ni v množični uporabi .....     | 8         |
| 2.3      | Lastnosti analitičnih poizvedb .....                                        | 8         |
| <b>3</b> | <b>Stolpčno usmerjena podatkovna baza .....</b>                             | <b>11</b> |
| 3.1      | Različni tehniki implementacije stolpčno usmerjene PB v vrstičnem SUPB..... | 11        |
| 3.1.1    | Vertikalno particioniranje (Vertical partitioning) .....                    | 12        |
| 3.1.2    | Indeksirano organizirana tabela (IOT) .....                                 | 13        |
| 3.2      | Gradnja n-teric (Tuple materialization) .....                               | 13        |
| 3.2.1    | Postopek gradnje n-teric .....                                              | 14        |
| 3.2.2    | Kdaj se postopek gradnje n-teric prične.....                                | 14        |
| 3.2.3    | Zgodnja gradnja n-teric (Early materialization) .....                       | 15        |
| 3.2.4    | Pozna gradnja n-teric (Late materialization).....                           | 16        |
| 3.2.5    | Invisible join .....                                                        | 18        |
| 3.3      | Kompresija.....                                                             | 21        |
| 3.3.1    | Slovar.....                                                                 | 22        |
| 3.3.2    | RLE kompresija (Run length encoding) .....                                  | 22        |
| 3.3.3    | Krajšanje nizov (Trim Compression) .....                                    | 23        |
| 3.4      | Souporaba (Coexistence) .....                                               | 23        |
| 3.5      | Pregled različnih stolpčno usmerjenih SUPB .....                            | 24        |

|          |                                                             |           |
|----------|-------------------------------------------------------------|-----------|
| <b>4</b> | <b>Primerjava stolpčne z vrstično podatkovno bazo .....</b> | <b>27</b> |
| 4.1      | V/I zmogljivosti.....                                       | 27        |
| 4.1.1    | <i>Primer prikaza V/I operacij .....</i>                    | <i>28</i> |
| 4.2      | Primerjava sistemov za upravljanje podatkovnih baz .....    | 30        |
| 4.2.1    | <i>Vrstični relacijski SUPB (R-RDBMS).....</i>              | <i>30</i> |
| 4.2.2    | <i>Stolpčni relacijski SUPB (C-RDBMS) .....</i>             | <i>33</i> |
| 4.3      | Shranjevanje v podatkovno bazo.....                         | 34        |
| 4.4      | Brisanje zapisa.....                                        | 36        |
| 4.5      | Dodajanje novega atributa.....                              | 36        |
| 4.6      | Ažuriranje podatkov .....                                   | 37        |
| 4.7      | Prednosti/slabosti.....                                     | 37        |
| <b>5</b> | <b>Praktični primer.....</b>                                | <b>41</b> |
| 5.1      | Podatkovni model.....                                       | 41        |
| 5.2      | Testiranje in analiza performanc .....                      | 43        |
| 5.2.1    | <i>Pregled podatkov .....</i>                               | <i>43</i> |
| 5.2.2    | <i>Ažuriranje podatkov.....</i>                             | <i>47</i> |
| 5.2.3    | <i>Vstavljanje novih podatkov .....</i>                     | <i>49</i> |
| 5.2.4    | <i>Brisanje podatkov .....</i>                              | <i>50</i> |
|          | <b>Zaključek .....</b>                                      | <b>53</b> |
|          | <b>Literatura.....</b>                                      | <b>55</b> |

## Kazalo slik

|                                                                     |    |
|---------------------------------------------------------------------|----|
| Slika 1: Primer zgodnje gradnje n-teric .....                       | 15 |
| Slika 2: Primer pozne gradnje n-teric .....                         | 17 |
| Slika 3: Postopek Invisible join .....                              | 20 |
| Slika 4: Nadaljevanje postopka Invisible join .....                 | 21 |
| Slika 5: Optimizator poizvedbe izbira med podatkovnima bazama ..... | 24 |
| Slika 6: Blok v podatkovni bazi Oracle .....                        | 27 |
| Slika 7: Podatkovni model.....                                      | 42 |
| Slika 8: Čas izvajanja poizvedbe 1 .....                            | 44 |
| Slika 9: Čas izvajanja poizvedbe 2 .....                            | 45 |
| Slika 10: Čas izvajanja poizvedbe 3 .....                           | 46 |
| Slika 11: Čas izvajanja poizvedbe 4 .....                           | 47 |
| Slika 12: Čas izvajanja poizvedbe 5 .....                           | 48 |
| Slika 13: Čas izvajanja poizvedbe 6 .....                           | 49 |
| Slika 14: Čas izvajanja poizvedbe 7 .....                           | 50 |
| Slika 15: Čas izvajanja poizvedbe 8 .....                           | 51 |

## Kazalo tabel

|                                                                         |    |
|-------------------------------------------------------------------------|----|
| Tabela 1: Primer relacije .....                                         | 4  |
| Tabela 2: Podatki v vrstični podatkovni bazi.....                       | 12 |
| Tabela 3: Podatki v tabelah s polnim vertikalnim particioniranjem ..... | 12 |
| Tabela 4: Primer podatkov v tabeli .....                                | 19 |
| Tabela 5: Prikaz RLE kompresije .....                                   | 23 |
| Tabela 6: Primer podatkov .....                                         | 28 |
| Tabela 7: Zgrajene n-terice iz zgornjih podatkov .....                  | 29 |
| Tabela 8: Analitična zahteva v vrstični podatkovni bazi .....           | 32 |
| Tabela 9: Analitična zahteva v stolpčni podatkovni bazi .....           | 34 |
| Tabela 10: Dodajanje novega zapisa v vrstično podatkovno bazo .....     | 35 |
| Tabela 11: Dodajanje zapisa v stolpčno podatkovno bazo .....            | 35 |
| Tabela 12: Brisanje zapisa v vrstični podatkovni bazi.....              | 36 |
| Tabela 13: Brisanje zapisa v stolpčni podatkovni bazi .....             | 36 |
| Tabela 14: Dodajanje zapisa v vrstično podatkovno bazo.....             | 36 |
| Tabela 15: Dodajanje zapisa v stolpčni podatkovno bazo .....            | 37 |
| Tabela 16: Ažuriranje zapisa v vrstični podatkovni bazi.....            | 37 |
| Tabela 17: Ažuriranje zapisa v stolpčni podatkovni bazi .....           | 37 |

## Slovar pojmov

**Query optimizer:** Optimizator poizvedbe. Je funkcija večine relacijskih sistemov za upravljanje s podatkovnimi bazami. Optimizator poizvedbe poizkuša najti najboljši način za izvedbo dane poizvedbe.

**Query Execution Plan:** Potek izvedbe poizvedbe. Je rezultat optimizatorja poizvedbe.

**Data page:** Stran na disku. Je najmanjša enota, ki jo prebere podatkovna baza.

**CRM (Customer Relationship Management):** Sistem za upravljanje odnosov s strankami.

**Overhead:** Prazen prostor na disku, ki nastane pri brisanju podatkov iz podatkovne baze.

**BI (Business Intelligence):** Poslovna inteligenca omogoča analizo podatkov in sprejemanje ključnih odločitev. Namen poslovne inteligenice je razumevanje delovanja organizacije.

**DBMS (Database management system):** Sistem za upravljanje podatkovnih baz (SUPB).

**Invisible join:** Hitro povezovanje tabel v poizvedbi. Način, ki optimizira pozno gradnjo n-teric.

**I/O:** Vhodno/izhodne operacije. Branje in pisanje iz oz. v pomnilnik.

**DSM (Decomposed storage model):** Predhodnik stolpčno usmerjenih podatkovnih baz.



## Povzetek

V diplomskem delu bom predstavil stolpčno usmerjene podatkovne baze. Med drugim bom odgovoril na vprašanje, zakaj se je pojavila potreba po stolpčno usmerjeni podatkovni bazi in zakaj je šele v zadnjem času več govora o njej, čeprav njen obstoj sega v 70 leta prejšnjega stoletja. Primerjal bom oba sistema za upravljanje podatkovnih baz, vrstičnega in stolpčnega, ter podal ključne lastnosti, ki stolpčni podatkovni bazi omogočajo hitrejšo izvedbo poizvedb. V vrstičnem sistemu za upravljanje podatkovnih baz lahko zagotovimo, da se ta obnaša kot stolpčni sistem. Na praktičnem primeru bom primerjal dve implementaciji stolpčne podatkovne baze v vrstičnem sistemu za upravljanje podatkovnih baz – vertikalno particioniranje in indeksirano organizirano tabelo. V nadaljevanju bom predstavil in na praktičnem primeru potrdil, zakaj je stolpčna podatkovna baza pri analitičnih poizvedbah boljša od vrstične oz. vseh implementacij stolpčne podatkovne baze v vrstičnem sistemu za upravljanje podatkovnih baz.

**Ključne besede:** Stolpčno usmerjena podatkovna baza, podatkovna baza, primerjava, vrstična podatkovna baza, analitična poizvedba, podatkovno skladišče, SUPB.



## Abstract

In the following thesis I will present column-oriented database. Among other things, I will answer on a question why there is a need for a column-oriented database. In recent years there have been a lot of attention regarding a column-oriented database, even if the existence of a columnar database management systems dates back in the early seventies of the last century. I will compare both systems for a database management – a column-oriented database system and a row-oriented database system and describe key features of the column-oriented database that allow faster query execution. In the practical part I will introduce two different techniques that can be used to implement a columnar-oriented database design in row-oriented database management system. These two techniques are Vertical partitioning and Index organized table (IOT). I will compare the performance of previous two techniques, row-store and a column-store and show that the row-store performance is significantly slower on analytical queries than a column-store.

**Keywords:** Column-oriented database, comparison, database, row-oriented database, analytical query, data warehouse, DBMS.



# Poglavje 1

## 1 Uvod

Danes se različne organizacije in podjetja srečujejo z eksplozijo podatkov. Te zbirajo, analizirajo in hranijo velike količine strukturiranih in nestrukturiranih podatkov, ki prihajajo iz različnih virov, kot so npr. vsakodnevne transakcije, različni podatki iz interneta in mobilne telefonije. Vedno več je tudi strojno generiranih podatkov, med katere sodijo razni podatki iz računalniških iger, družabnih omrežij, senzorjev, samodejno generiranih dnevniških datotek, satelitov... Poleg podatkov iz dneva v dan narašča tudi število uporabnikov, ki potrebujejo dostop do podatkov, nad katerimi izvajajo različne analize [1].

Glavni izzivi, s katerimi se srečujemo oz. se bomo srečevali v prihodnosti, so zagotovo velika količina podatkov, kako te podatke najbolje shraniti, predstaviti in z vidika sistema zagotoviti najbolj optimalno delovanje. Tu je pomembna zmogljivost in dostopnost sistema, hitrost pridobitve podatkov in njihova kakovost. Za doseganje teh ciljev so se pojavili BI sistemi in rešitve.

Ljudje, ki se ukvarjajo s temi orodji, potrebujejo hiter in zanesljiv dostop do informacij, saj le tako lahko zagotovijo pravilne in hitre odločitve v stalno spreminjajočem se okolju. Včasih so bila poročila posebej pripravljena za vodilne kadre podjetja, danes pa orodja za poročanje uporablja večina zaposlenih. To se odraža na količini podatkov in informacij, ki jih uporabniki potrebujejo za pregled tekočega poslovanja podjetja. Poleg tega ne smemo zanemariti zgodovine podatkov, ki mora biti dostopna za potrebe različnih analiz. Z večanjem količine podatkov in številom poizvedb se veča tudi zahteva po bolj zmogljivi strojni opremi. Količina poizvedb vpliva na število procesorskih ciklov, količina podatkov pa na velikost podatkovnega skladišča, ki se vsakih nekaj let podvoji. Podatkovna skladišča so največkrat implementirana na relacijskih podatkovnih modelih, ki podatke shranjujejo v vrstičnih podatkovni bazah.

Ker tradicionalne – vrstične podatkovne baze niso bile narejene z namenom prikazovanja velike količine podatkov, nad katerimi bi izvajali različne analize, in ker je njihova največja ovira hitrost prikaza velike količine podatkov, so prišli do novega pristopa shranjevanja podatkov – shranjevanja podatkov v stolpce. Tako se je pojavila potreba po stolpčno usmerjeni podatkovni bazi. Stolpčno usmerjena podatkovna baza ima bistveno boljšo

## Stolpčno usmerjen SUPB

zmogljivost in hitrejšo izvedbo analitičnih poizvedb, zato jih tudi mnogi vidijo kot novo generacijo sistemov za upravljanje s podatki.

### 1.1 Podatkovna baza

Podatkovna baza ali podatkovna zbirka je organizirana zbirka podatkov, ki je shranjena v računalniškem sistemu [2].

Podatkovne baze so bile narejene z namenom vnosa, shranjevanja, pridobivanja in upravljanja s podatki. Vsaka informacija se shrani v vrstico, ki ustvari zapis. Zapisi so urejeni in upravljeni na več različnih načinov, odvisno od programske opreme, ki se uporablja. Podatkovne baze bi lahko primerjali z razpredelnicami. Bistvena prednost podatkovnih baz je zmogljivost in možnost urejanja shranjenih podatkov.

Podatkovno bazo sestavljajo:

- Podatki,
- uporabniki in uporabniški program,
- skrbnik podatkovne baze in
- sistem za upravljanje podatkovnih baz.

Kadar govorimo o podatkovnih bazah, moramo omeniti tudi sistem za upravljanje podatkovnih baz – SUPB. To je zapleten sistem, ki predstavlja vmesnik med uporabnikom in podatkovno bazo. Narejen je bil z namenom kreiranja, vzdrževanja in nadzora nad dostopom do podatkov v podatkovni bazi [3]. Današnji SUPB predstavlja desetletja raziskav in razvoja na tem področju. Sistemi za upravljanje podatkovnih baz so bili eni izmed prvih sistemov, ki so bili množično nameščeni na strežnikih.

Interakcije s sistemom upravljanja podatkovnih baz lahko razdelimo v štiri večje skupine:

- Kreiranje podatkovnih struktur ali DDL (angl. Data Definition Language). Ta omogoča spreminjanje same strukture podatkovne baze (Alter, Create, Drop, Grant, Truncate).
- Vzdrževanje podatkovne baze ali DML (angl. Data Manipulation language), ki omogoča spreminjanje podatkov (Update, Insert, Delete).

- Iskanje po podatkih. Pridobivanje informacij v sklopu poizvedb za končne uporabnike in poročila ter izvajanje različnih transakcij (Select).
- Upravljanje s sistemom – registracija in kontrola nad uporabniki, varnost nad podatki, nadzor nad delovanjem, relevantnost podatkov in obnovitev podatkov v primeru izgube (angl. Data Control Language – DCL).

Sistem za upravljanje podatkovnih baz je odgovoren za ohranjanje integritete in varnosti shranjenih podatkov ter zagotavljanje razpoložljivosti in celovitosti podatkov. Razpoložljivost zagotavlja tako, da omogoča večim uporabnikom sočasni dostop do vseh podatkov, celovitost pa tako, da podatkovna baza v primeru sistemskih težav poskrbi za obnovitev podatkov. Omogoča tudi nadzor nad sočasnim dostopom do podatkov in preverjanje vhodnih podatkov.

Podatkovni del podatkovne baze sestavlja fizična podatkovna baza, ki shranjuje podatke, in metapodatkovna baza, ki shranjuje različne informacije o podatkih. Podatkovni del je shranjen v zunanjem pomnilniku – največkrat je to trdi disk. Od njega pričakujemo, da ima sposobnost hranitve velike količine podatkov, omogoča hiter dostop do podatkov, njihovo ažuriranje in navsezadnje tudi nizko ceno.

Trinivojska arhitektura podatkovne baze je sestavljena iz zunanje sheme, ki opredeljuje različne poglede uporabnikov na podatke, konceptualne ali logične sheme, ki predstavlja strukturo podatkovne baze in fizične sheme, s katero upravlja operacijski sistem. Glavni razlog za trinivojsko arhitekturo tiči v zagotavljanju podatkovne neodvisnosti. To pomeni, da spremembe na določenem nivoju ne vplivajo na delovanje ostalih nivojev [3,4].

## 1.2 Relacijski podatkovni model

Sistemi za shranjevanje podatkov lahko podatke shranjujejo v različne podatkovne modele. Podatkovni model je zbirka konceptov in pravil, ki opisuje strukturo podatkovne baze. V modelu se nahajajo smiselno poimenovane in med seboj povezane tabele. Relacijski podatkovni model je le ena, vendar daleč največkrat uporabljena implementacija podatkovnega modela. Model se je prvič pojavil leta 1970. V relacijskem podatkovnem modelu morajo biti vsi podatki shranjeni v relacijah ali tabelah.

Vsaka relacija je dvodimenzionalna tabela, ki vsebuje vrstice in stolpce, vsebovati pa mora tudi glavo in telo tabele, kar prikazuje tabela 1. Glava predstavlja seznam stolpcev ali

## Stolpčno usmerjen SUPB

atributov v relaciji, telo pa vsebuje podatke, ki se nahajajo v vrsticah. Vrstice imenujemo tudi n-terice (angl. tuple).

Entiteta je v podatkovni bazi predstavljena z množico atributov. To je objekt opazovanja iz realnega sveta, ki ga lahko ločimo od ostalih objektov.

stolpci ali atributi

| ID | stolpec1 | stolpec2       |
|----|----------|----------------|
| 1  | podatek1 | drugi podatek1 |
| 2  | podatek2 | drugi podatek2 |
| 3  | podatek3 | drugi podatek3 |
| 4  | podatek4 | drugi podatek4 |

Tabela 1: Primer relacije

Stopnja relacije je v zgornjem primeru 3. To je število atributov, ki se nahajajo v tabeli. Števnost relacije (kardinalnost) predstavlja število vrstic v tabeli (n-teric). V zgornjem primeru je števnost relacije 4 [5].

Razmerja med atributi relacije opisuje funkcionalna odvisnost. To je razmerje med dvema atributoma ali skupini atributov. V relaciji so vse vrstice med seboj različne. Tako se je potrebno sklicevati na posamezno vrstico oz. jo identificirati, kar pa zahteva poznavanje vrednosti atributov. V primeru, da v relaciji nastopajo funkcionalne odvisnosti, nam tako ni potrebno poznati vseh vrednosti atributov, ampak le tiste, ki zadoščajo razlikovanju vrstic. Tej množici atributov pravimo ključi relacije. Ključi v relaciji so posebni stolpci, ki se uporabljajo za povezovanje relacij oz. tabel. Relaciji lahko pripada več ključev, samo eden od njih pa je lahko primarni ključ, ki je najbolj pomemben in mora biti enolično določen. Drugi ključ je tuj ključ in se uporablja za povezovanje dveh relacij. Tuj ključ neke relacije je povezan s primarnim ključem druge relacije. Unikatni ključ je sestavljen iz večih atributov in je v relaciji enolično določen iz vseh atributov, ki ga sestavljajo [4,5].

### 1.2.1 SQL

Jezik SQL (angl. Structured Query Language) se je razvil z namenom upravljanja podatkovnih baz. Razvit je bil v laboratorijih IBM v začetku 70 let prejšnjega stoletja. Na začetku je bil sicer poimenovan SEQUEL (angl. Structured English Query Language), z uradno izdajo pa se je ime spremenilo v SQL. Jezik SQL je leta 1986 Ameriški državni inštitut za standarde (ANSI) priznal kot standard. Temu je leta 1987 sledila tudi Mednarodna organizacija za standarde (ISO). Od takrat so standard nekajkrat spremenili in dodali nove funkcije:

- Omejitve integritete,
- rekurzivne poizvedbe,
- podporo za procedure,
- sprožilce,
- sekvence za avtomatsko generiranje vrednosti,
- XML novosti,
- sortiranje poizvedb (ORDER BY),
- itd.

Jezik SQL je danes še vedno vodilni pri upravljanju s podatki. Ker je ponudnikov podatkovnih baz več, se je razvilo več verzij jezika SQL, kot so npr. Oracle, MS strežnik SQL, MySQL, PostgreSQL [6].



## **Poglavje 2**

### **2 Potreba po stolpčno usmerjeni podatkovni bazi**

Podatkovne baze, če gledamo zgodovinsko, so se prvič pojavile v 70 letih prejšnjega stoletja. Takrat so se razvijalci orientirali po implementacijah vrstičnih podatkovnih baz, saj so te dosegale najboljše rezultate v transakcijskih sistemih. Glavna podjetja, ki so takrat ponujala vrstične podatkovne baze, obstajajo še danes. Med njimi so Oracle, IBM in Microsoft. Okoli leta 1990 so se začele pojavljati aplikacije, nad katerimi so vrstične podatkovne baze dosegale slabše zmogljivosti. Te aplikacije so bolj analitične narave, njihov cilj pa je prebrati velike količine podatkov in pridobiti pomembne informacije za podporo odločitvam in planiranju, kar je privedlo do dveh težav. Analitične poizvedbe so po naravi (lahko) bolj kompleksne, zagotovo pa vsebujejo veliko količino podatkov, zato potrebujejo dalj časa za izvedbo poizvedbe. Nastal je problem, da je morala aplikacija pred zapisom v bazo počakati na izvedbo analitične poizvedbe, saj je le to preprečilo, da ne bi prišlo do nekonsistentnosti podatkov. Drugi problem pa je količina podatkov. Analitične poizvedbe potrebujejo večjo količino podatkov za analize trenutnega poslovanja in poslovanja podjetja v preteklosti. Nekatera podjetja so problem rešila z dvema podatkovnima bazama. Vrstična podatkovna baza se tako uporablja za transakcijske sisteme, stolpčno usmerjena podatkovna baza pa za analitične namene [7].

#### **2.1 Zakaj stolpčno usmerjena podatkovna baza**

Vrstična relacijska podatkovna baza je sicer fleksibilna in uporabna za podporo transakcijskim in analitičnim potrebam, vendar z večanjem podatkovnega skladišča, naraščanjem kompleksnosti podatkov in s stalnim večanjem časa izvedbe določene poizvedbe je bil predlagan nov pristop shranjevanja podatkov, ki podatke v relacijskem podatkovnem modelu ne bo shranjeval v vrsticah, ampak v stolpcih. Tako se je začela omenjati stolpčno usmerjena podatkovna baza, ki je namenjena hitremu prikazu podatkov in njihovi analizi, hkrati pa zagotavlja razširljivost in visoko razpoložljivost. Hitrost prikaza analitičnih podatkov v stolpčno usmerjeni podatkovni bazi je nekajkrat hitrejši od navadne vrstične podatkovne baze, kar bom v nadaljevanju diplomske naloge tudi prikazal na praktičnem primeru. V veliki meri razlog za to tiči pri branju podatkov, kjer stolpčno usmerjena podatkovna baza prebira le podatke, ki so aktualni za določene poizvedbe in tako zmanjša število bralnih ciklov.

## **Stolpčno usmerjen SUPB**

### **2.2 Zakaj stolpčno usmerjena podatkovna baza še ni v množični uporabi**

Ideja o stolpčno usmerjeni podatkovni bazi se je prvič pojavila okoli leta 1970. Takrat se je prvič omenilo tudi transponiranje datotek in vertikalno particioniranje. V letih med 1980 in 1990 so nastali prvi dokumenti o primerjavi med takratnim predhodnikom stolpčno usmerjenih podatkovnih baz (DSM) in vrstično usmerjenih podatkovnih baz. V istem obdobju so nastale tudi prve raziskave na področju povezovanja tabel in projekcij, kar je še bolj okrepilo prednosti stolpčno usmerjenih v primerjavi z vrstično usmerjenimi podatkovnimi bazami. Kljub primernosti stolpčno usmerjenih podatkovnih baz za analitične poizvedbe je na trgu še vedno vladala vrstično usmerjena podatkovna baza [8]. Ker je princip stolpčno usmerjenih podatkovnih baz še dokaj svež in se je ideja začela pojavljati v zadnjih letih (prvič resneje leta 2006), teh še ni v množični uporabi. Večjo vlogo bodo začele imeti v prihajajočih letih, ko bo množično naraslo število podatkov in se bodo povečali časi izvajanja poizvedb v vrstičnih podatkovnih bazah.

### **2.3 Lastnosti analitičnih poizvedb**

Narava poizvedb v podatkovnih skladiščih se razlikuje od tistih v transakcijskih sistemih. Te poizvedbe se v grobem razdelijo v štiri skupine:

- so manj predvidljive

V transakcijskih sistemih so ponavadi poizvedbe znane vnaprej oz. so kodirane v samem procesu. Procesu tako dodamo le določene parametre, ki se napolnijo ob izvedbi določene poizvedbe. V analitičnem svetu so poizvedbe bolj raziskovalne narave oz. odvisne od vsakega posameznika (kaj si uporabnik ob določenem času želi).

- čas izvajanja poizvedbe je daljši

Pri transakcijskih sistemih je bistvo, da je izvajanje določene poizvedbe čim krajše (npr. dodaj stranko, najdi račun, prenesi določeno vsoto denarja iz računa ena na račun dva). Drugače je to pri poizvedbah v podatkovnih skladiščih, ki preberejo več podatkov in tako pridobijo več pomembnih informacij, npr.: za določenega komitenta bi radi izvedeli, ali je primeren za odobritev posojila. Preiskati je potrebno vse zapise in zgodovino posojil izbranega komitenta ter jih primerjati s trenutnim stanjem komitenta.

- več bralnih kot pisalnih ciklov

Analitiki podatkovna skladišča potrebujejo večinoma za branje in pregled podatkov. V praksi se podatki v podatkovna skladišča polnijo v okviru paketov. Ti paketi se izvajajo ponoči in vpisujejo podatke, ki so bili zbrani tekom dneva.

- večji poudarek je na atributih in ne entitetah (ena stranka, en klic)

Večina poizvedb v podatkovnih skladiščih ne poizveduje o točno določeni entiteti. Te poizvedbe večinoma poizvedujejo o večjih entitetah in nekaj atributih, ki se potem agregirajo. Torej, poizvedbe na način "pregled povprečnega stanja na računu" so bolj pogoste od "katero številko računa ima določen komitent" [9].



## **Poglavje 3**

### **3 Stolpčno usmerjena podatkovna baza**

Stolpčno usmerjen sistem za upravljanje podatkovnih baz je sistem, ki podatke shranjuje v stolpce in ne vrstice kot večina relacijskih podatkovnih baz. Vrstično shranjevanje podatkov je primerno za transakcijske sisteme, kot so npr. bančni sistemi, kjer so vsi atributi komitenta shranjeni v eni sami vrstici, ni pa primerno za analitične sisteme, kjer je veliko bralnih operacij z namenom dostopa le do nekaterih atributov. V večini primerov analitične poizvedbe pregledajo vse zapise, procesirajo pa le nekatere. Iz tega stališča ima prednost stolpčno usmerjena podatkovna baza, ki ima v istem stolpcu shranjene enake tipe podatkov. Tako ji ni potrebno prebrati vseh atributov, kar tudi privede do boljše učinkoviti in hitrejše izvedbe poizvedbe.

Posebna značilnost BI sistemov je možnost izdelave različnih poslovnih analiz in rudarjenje po podatkih. Ker to zahteva hitro odzivnost, hiter prikaz podatkov oz. hitre izvedbe določenih poizvedb, je lahko jasno, da za to niso več primerne vrstične relacijske podatkovne baze in so jih oz. jih bodo nadomestile stolpčne. V stolpčno usmerjenih podatkovnih bazah je vsak atribut shranjen v svoji tabeli. Zaporedne vrednosti istega atributa so zapisane zaporedno v isto tabelo.

To je velika prednost za podatkovna skladišča, kjer so velike potrebe po hitrem izračunu različnih agregacij (sum, count, avg, min ...) na veliki količini podatkov. Stolpčno usmerjene podatkovne baze so prav tako primerne za CRM sisteme, knjižnice in sprotne (ad-hoc) poizvedovanja [10].

#### **3.1 Različni tehniki implementacije stolpčno usmerjene PB v vrstičnem SUPB**

V vrstično organiziranem sistemu za upravljanje podatkovnih baz obstaja nekaj možnosti oz. tehnik implementacije stolpčno usmerjene podatkovne baze. Med njih spadata vertikalno particioniranje (angl. vertical partitioning) in indeksirano organizirana tabela (angl. index organized table).

## Stolpčno usmerjen SUPB

### 3.1.1 Vertikalno particioniranje (Vertical partitioning)

Vertikalno particioniranje se naredi izključno v primeru, ko bi radi dosegli boljše performance. Pri vertikalnem particioniraju se tabele loči po njihovih atributih. Najbolj optimalen način je, da se iz določene tabele odstrani odvečne attribute in tako izboljša performance samih poizvedb. Na ta način lahko že normalizirano tabelo, prikazano v tabeli 2, razdelimo na več manjših tabel. Stolpčno usmerjeni podatkovni bazi se najbolje približamo s polnim vertikalnim particioniranjem, kot je prikazano v tabeli 3, kjer se za vsak atribut v tabeli naredi nova tabela z dvema stolpcema (ključ, atribut). Nova tabela poleg atributa vsebuje polje (ključ), ki služi kot mehanizem za združevanje atributov, ki pripadajo isti vrstici oz. zapisu v tabeli. To se najbolj preprosto naredi z dodatnim številskim atributom v tabeli. Za ta atribut ni priporočena uporaba primarnega ključa, saj je ta lahko velik ali pa sestavljen iz večjega števila atributov. Ker pri tem postopku iz ene tabele nastane več tabel, je potreben popravek sheme in sprememba obstoječih poizvedb [11].

Podatki v vrstični podatkovni bazi:

| ID | A1 | A2 | A3 |
|----|----|----|----|
| 1  | A  | B  | C  |
| 2  | D  | E  |    |
| 3  |    | F  |    |
| 4  | G  |    |    |

Tabela 2: Podatki v vrstični podatkovni bazi

Podatki v tabelah s polnim vertikalnim particioniranjem:

| ID | ID | A1 | ID | A2 | ID | A3 |
|----|----|----|----|----|----|----|
| 1  | 1  | A  | 1  | B  | 1  | C  |
| 2  | 2  | D  | 2  | E  |    |    |
| 3  | 4  | G  | 3  | F  |    |    |
| 4  |    |    |    |    |    |    |

Tabela 3: Podatki v tabelah s polnim vertikalnim particioniranjem

Vertikalno particioniranje ima nekaj pomanjkljivosti. Ena od njih je potreba po shranjevanju dodatnega polja, ki združuje podatke o istem zapisu, drugi problem pa je dolžina glave zapisa, ki je relativno dolga. Vsak zapis v vrstični podatkovni bazi vsebuje eno glavo, ki služi upravljanju delov zapisa shranjenega v bloku (tega problema pri stolpčno usmerjeni podatkovni bazi ni, saj se glave zapisa shranijo ločeno od podatkov).

Ta pristop je mogoče implementirati na trenutnih sistemih za upravljanje podatkovnih baz, kar bom v nadaljevanju diplomskega dela tudi prikazal in predstavil. Glavno vprašanje, ki se nanaša na to implementacijo, je: »Zakaj potem, če so performance res toliko boljše, ne implementiramo vertikalnega particioniranja v vseh podatkovnih skladiščih, ki uporabljajo vrstično podatkovno bazo?«

Odgovor na to vprašanje bom podal v praktičnem delu diplomskega dela, kjer bom predstavil, primerjal in ocenil različne pristope oz. implementacije stolpčno usmerjene podatkovne baze v primerjavi s pravo stolpčno podatkovno bazo.

### 3.1.2 Indeksirano organizirana tabela (IOT)

Drugi pristop implementacije stolpčno usmerjene podatkovne baze je indeksirano organizirana tabela. Ta v indeksu hrani vse stolpce, ki se nahajajo v tabeli. Indeksno organizirana tabela je tip tabele, ki podatke shranjuje v strukturi indeks B drevesa. Razlika med indeksirano organizirano tabelo in normalno relacijsko tabelo (kopico) je, da ima indeksirano organizirana tabela podatke logično urejene po primarnem ključu.

Indeksirano organizirane tabele same po sebi niso tabele in nimajo točno določenega fizičnega naslova lokacije (imajo logični naslov – ID vrstice), zato se v pomnilniku lahko premikajo (npr. pri vnosu novega podatka v tabelo se zapis lahko shrani v drugi blok) in ohranjajo sortiran vrstni red. Sprememba podatkov v tabeli – dodajanje nove vrstice, brisanje ali ažuriranje določenega zapisa povzroči spremembo indeksa [12].

Prednosti indeksirano organizirane tabele:

- Hiter naključen dostop do podatkov – iskanje po indeksu je hitro in učinkovito,
- za vsak atribut zgradi list (ključ, vrednosti) in tako ne podvaja enakih podatkov, kar zmanjša porabo prostora,
- hiter dostop do večjega razpona podatkov po primarnem ključu (podatki s podobnimi primarnimi ključi so shranjeni skupaj).

## 3.2 Gradnja n-teric (Tuple materialization)

V stolpčno usmerjeni podatkovni bazi se podatki o entiteti nahajajo na različnih lokacijah v pomnilniku (npr. podatki o komitentu, kot so ime, priimek, starost, telefonska številka, ipd. so

## Stolpčno usmerjen SUPB

shranjeni v različnih stolpcih). Večina poizvedb je takšnih, da dostopa do večjega števila atributov določene entitete, zato mora podatkovna baza vedeti, kako združiti in prikazati vse podatke, ki pripadajo isti entiteti. Gradnja n-teric mora delovati dobro, saj je najbolj pogosta operacija stolpčno usmerjene podatkovne baze. Ker gre za kompleksno operacijo, sta bistvena dva izziva, s katerima se sooča [13]:

- Kako se pravilna gradnja n-teric naredi in
- kdaj se ta postopek začne izvajati.

### 3.2.1 Postopek gradnje n-teric

Za vsako entiteto  $e$ , ki se mora zgraditi, je potrebno najti  $e$ -to pozicijo v prvem stolpcu in prebrati ustrezno vrednost. Ko se vrednost prebere, je potrebno najti drugi stolpec in spet prebrati vrednost, ki se nahaja na  $e$ -ti poziciji. To je potrebno storiti za vse attribute, ki se nahajajo v poizvedbi. Če bi stolpčno usmerjena podatkovna baza delovala na tak način, bi bilo to zelo zamudno in posledično bi prišlo do daljšega izvajanja poizvedb. Stolpčno usmerjena podatkovna baza se tem zamudam izogne na način, da podatke prvega stolpca shrani v pomnilnik in jih zadrži za čas branja ostalih podatkov, kar ji omogoča branje podatkov vzporedno. Ko so vsi atributi prebrani, se v pomnilniku zgradijo n-terice [13].

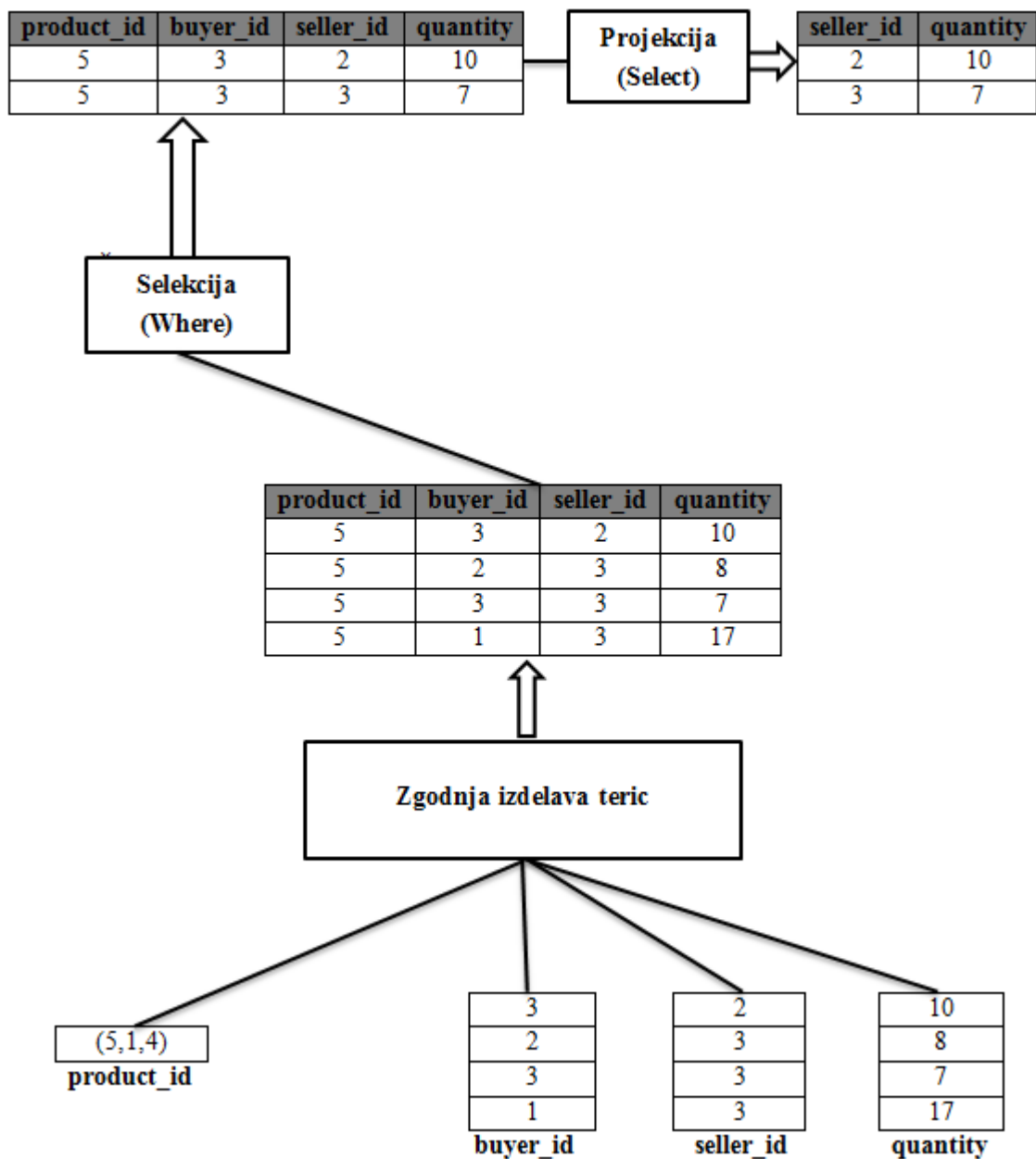
### 3.2.2 Kdaj se postopek gradnje n-teric prične

Obstajata dve možnosti gradnje n-teric. Ena od možnosti je na začetku izvajanja poizvedbe takoj, ko se določijo atributi, ki so aktualni v dani poizvedbi. Druga možnost pa je, da se gradnja n-teric naredi proti koncu izvajanja poizvedbe. Na ta način operatorji v poizvedbi delujejo na posameznih kompresiranih stolpcih toliko časa, kot je le mogoče. Izkazalo se je, da je za poizvedbe na eni sami tabeli najboljši način izdelave n-teric na koncu poizvedbe. Povezovanje stolpcev (angl. join) bistveno zaplete način izdelave n-teric, zato je v primeru večih povezanih tabel najboljša, da se n-terice izdelajo pred združevanjem tabel. Ker je način pozne izdelave n-teric s stališča zmogljivosti stolpčno usmerjene podatkovne baze bolj primeren (deluje na kompresiranih podatkih), so ga izboljšali s tehniko, imenovano Invisible join [13].

## 3.2.3 Zgodnja gradnja n-teric (Early materialization)

Pri zgodnji gradnji n-teric se te zgradijo in povežejo pred procesiranjem in s tem dekompresirajo vse kompresirane podatke v stolpcih, kar pomeni večje število podatkov.

```
SELECT seller_id, quantity
FROM orders
WHERE product_id = 5 AND buyer_id = 3
```



Slika 1: Primer zgodnje gradnje n-teric

## Stolpčno usmerjen SUPB

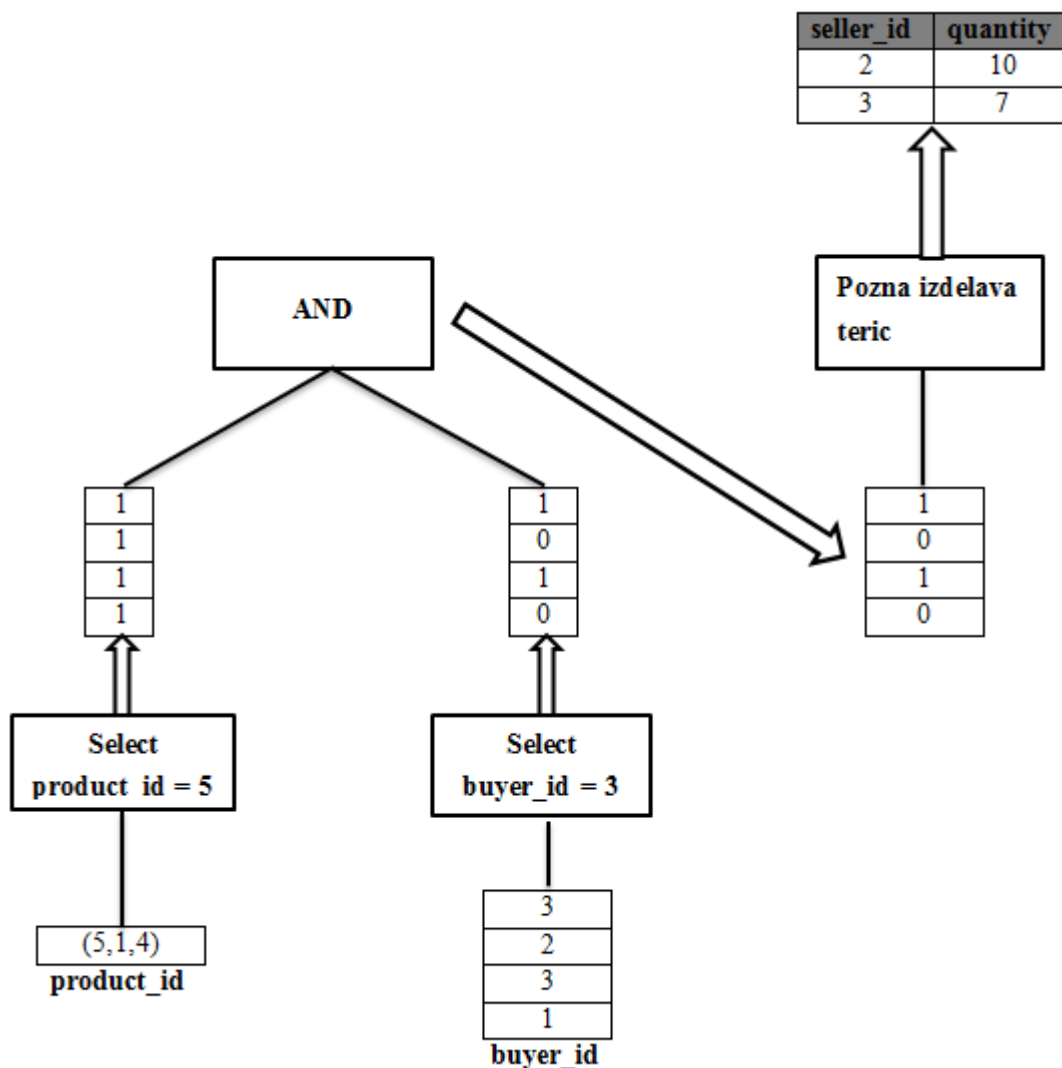
Slika 1 prikazuje primer gradnje n-teric. Z RLE kompresijo je stisnjen stolpec product\_id (vrednost, začetna pozicija, končna pozicija). Ker gre za zgodnje povezovanje n-teric, optimizator poizvedbe najprej izbere pogoj, za katerega misli, da bo odstranil največ odvečnih podatkov. Tabele oz. podatke združi skupaj in filtrira polja, ki ne zadovoljujejo pogoja. Enako stori za ostale poizvedbe. Tako se iz aktualnih podatkov zgradi dvodimenzionalna tabela. Selekcija iz tabele izbere vse podatke, ki zadoščajo pogoju (where) v poizvedbi – v zgornjem primeru sta to vrstica ena in tri, ostali podatki pa se odstranijo. Projekcija (select) prikaže izbrane attribute [13].

### 3.2.4 Pozna gradnja n-teric (Late materialization)

Pozna izdelava n-teric počaka, da se n-terice povežejo na koncu poizvedbe. V tem postopku se operacije izvajajo na kompresiranih podatkih.

Spodnji primer prikazuje pozno gradnjo n-teric:

```
SELECT seller_id, quantity  
  FROM orders  
WHERE product_id = 5 AND buyer_id = 3
```



Slika 2: Primer pozne gradnje n-teric

Slika 2 prikazuje primer pozne gradnje n-teric. Pri pozni gradnji n-teric se zgradijo biti, ki imajo vrednosti 0 in 1, odvisno od tega, ali podatek ustreza pogoju v poizvedbi. V zgornjem primeru vsi zapisi v product\_id ustrezajo pogoju v poizvedbi, pri buyer\_id pa samo prvi in tretji zapis. V primeru, da je povezava na več tabel, se tabeli povežeta skupaj. Načeloma je to hitra operacija, saj so stolpci v tabelah sortirani po poziciji. Vrednosti, ki jih dobimo, se združijo z logičnim operatorjem IN (angl. AND), tako da ostanejo le vrednosti z bitno vrednostjo 1. Za te vrednosti se pogleda, kateri zapis se nahaja v tabeli dejstev in ostalih dimenzijah. Zapisi se preberejo, dekompresirajo, združijo v n-terico in prikažejo uporabniku [13].

## Stolpčno usmerjen SUPB

### Prednost pozne gradnje n-teric:

Prednost pozne gradnje n-teric je možnost uporabe hitrih logičnih operacij nad kompresiranimi podatki. N-terice se zgradijo proti koncu izvajanja poizvedbe. Ker so takrat zapisi, ki ne zadovoljujejo pogojev v poizvedbi, že odstranjeni, se zgradi manjše število n-teric.

### Slabost pozne gradnje n-teric:

Slabost pozne gradnje n-teric je možnost večkratnega dostopanja do istih stolpcev. Npr. prvič se do stolpca dostopa za pridobitev pozicij vseh zapisov, ki ustrezajo pogoju, in drugič (v primeru, da se ujemanje pozicije ne da določiti iz indeksa) kasneje v načrtu poizvedbe, ko je potrebno prebrati vrednosti, ki se nahajajo na tej poziciji.

Na drugi omenjen način izdelave n-teric (pozna izdelava) deluje večina stolpčno usmerjenih podatkovnih baz.

### 3.2.5 Invisible join

Invisible join je tehnika povezovanja tabel, ki izboljša performance pozne gradnje n-teric v stolpčno usmerjeni podatkovni bazi. Tehnika združevanja se uporablja v stolpčno usmerjeni podatkovni bazi nad povezavami primarnih in tujih ključev v zvezdni shemi. Postopek je podoben poznemu povezovanju n-teric z razliko, da zmanjša število dimenzijskih vrednosti. Deluje tako, da ponovno izdela povezave glede na tuje ključe v tabeli dejstev in s tem zmanjša število zapisov v dimenzijah. Odvečnih zapisov v dimenzijski tabeli tako ni potrebno brati. To naredi tako, da iz podatkov, ki zadoščajo pogoju v poizvedbi, zgradi več razpršenih tabel, ki vsebujejo primarne ključe dimenzijskih tabel.

S tem vzporedno izvede več selekcij na stolpcih tabel dejstev, ki jih potem združi z logičnimi operatorji (bitmap operacijo) [11].

```
SELECT SELL.name as name, ORD.quantity  
FROM orders o, seller s,  
      buyer b, product p  
WHERE o.seller_id = s.id  
      AND o.buyer_id = b.id  
      AND o.product_id = p.id
```

AND p.name in ('Jeans', 'Boots')

AND b.name in ('Carlo', 'Bud')

Za povezovanje tabele dejstev z dimenzijsko tabelo, ki je prikazana v tabeli 4, se naredi razpršena tabela, ki vsebuje primarne ključne zapise, kjer je pogoj v poizvedbi zadoščen.

| id | name  | Surname  | email                |
|----|-------|----------|----------------------|
| 1  | Jame  | Churchil | me@gmail.com         |
| 2  | Carlo | Thomas   | him@gmail.com        |
| 3  | Bud   | Kruger   | specialone@gmail.com |

Tabela 4: Primer podatkov v tabeli

Za zgornji primer in pogoj:

WHERE b.name in ('Carlo', 'Bud')

se zgradi razpršena tabela, ki vsebuje vrednosti 2 in 3. Podobno se zgradi tudi razpršena tabela za produkte z vrednostima 3 in 12, kjer je številka 3 id in ime produkta 'Jeans' ter zapis z id 12 in imenom produkta 'Boots'. Izdelava razpršenih tabel je prikazana na sliki 3.

## Stolpčno usmerjen SUPB

Tabela dejstev: Orders

| id | store_name  | Seller_id | Product_id | Buyer_id | Date_sold_id | quantity |
|----|-------------|-----------|------------|----------|--------------|----------|
| 1  | Classic     | 15        | 12         | 2        | 20130716     | 15       |
| 2  | All In One  | 4         | 3          | 15       | 20130916     | 16       |
| 3  | Stream      | 7         | 1          | 6        | 20130803     | 7        |
| 4  | Good d.o.o. | 8         | 5          | 7        | 20130716     | 1        |
| 5  | Classic     | 19        | 3          | 3        | 20131013     | 6        |
| 6  | Stream      | 15        | 7          | 3        | 20130825     | 7        |
| 7  | Blue Moon   | 4         | 12         | 10       | 20130928     | 5        |

Razpršena tabela za  
product\_id z  
vrednostima 3 in  
12.

=

|   |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 1 |
| 0 |
| 1 |

Razpršena tabela  
za buyer\_id z  
vrednostima 2 in 3

=

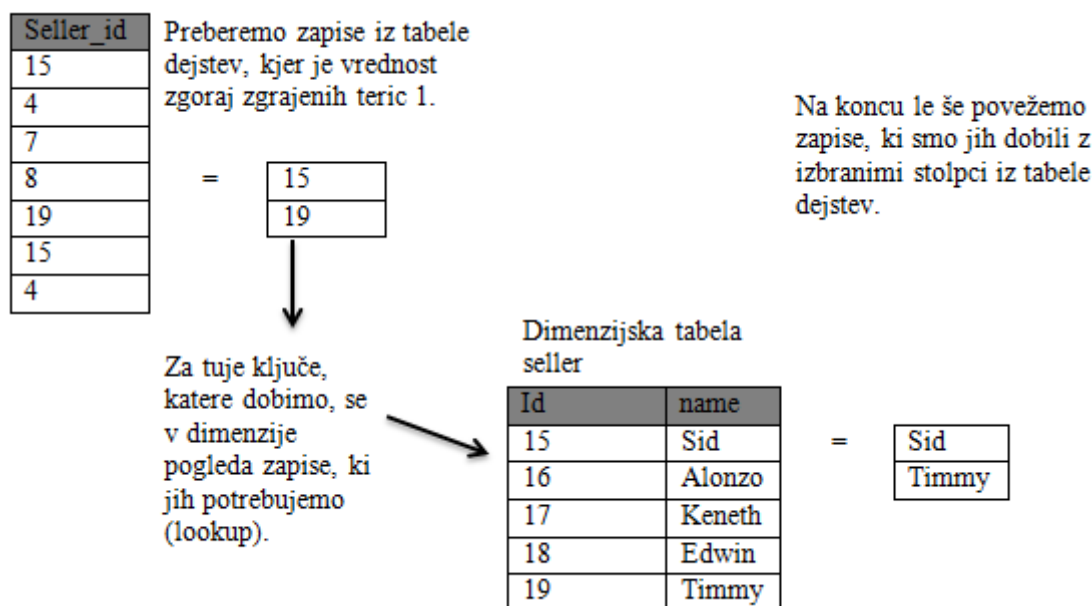
|   |
|---|
| 1 |
| 0 |
| 0 |
| 0 |
| 1 |
| 1 |
| 0 |

AND  
(IN)

|   |
|---|
| 1 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |
| 0 |

Slika 3: Postopek Invisible join

Iz razpršenih tabel se zgradijo n-terice, ki vsebujejo vrednost 1, kjer je pogoj zadoščen, in 0, kjer ni. Za vse stolpce v tabeli dejstev, kjer je vrednost 1, se preberejo zapisi. Za te zapise se v dimenzijske tabele pogleda, katera vrednost se nahaja v določeni vrstici, kar prikazuje slika 4.



Slika 4: Nadaljevanje postopka Invisible join

### 3.3 Kompresija

Kompresija je tehnika krčenja podatkov, ki je uporabljena v večini sistemov za upravljanje podatkovnih baz. Njen namen je izboljšanje performanc podatkovne baze in zmanjšanje zasedenosti prostora na disku. Grobo krčenje podatkov je izrazita lastnost stolpčno usmerjene podatkovne baze in se razlikuje pri ponudnikih podatkovnih baz. Kompresijsko razmerje je v stolpčno usmerjeni podatkovni bazi veliko boljše kot v vrstični podatkovni bazi, saj so si zaporedni zapisi v stolpcu zelo podobni. Razmerje se še vidno izboljša v primeru, da so podatki v stolpcu urejeni. Kompresija v stolpčno usmerjenih podatkovnih bazah izboljša performance procesorja, saj ta izvaja logične operacije nad kompresiranimi podatki [11].

Stolpčno usmerjena podatkovna baza MonetDB samodejno kompresira podatke na različnih nivojih. Podatki v tej podatkovni bazi so shranjeni v seznamu. Velika prednost seznama je, da ne vsebuje praznih podatkov in tako ne hrani odvečnega prostora. Taka predstavitev podatkov omogoča neposredno mapiranje datotek v pomnilnik. Vsi podatkovni nizi so shranjeni s pomočjo slovarjev, kar občutno zmanjša porabo prostora na disku. Navsezadnje je uporaba slovarja lahko tudi dvorezen meč, saj za velike slovarje porabi več stroškov za njihovo vzdrževanje. Tega problema se zaveda tudi podatkovna baza MonetDB in za večje tabele ne uporablja kompresije [14].

## Stolpčno usmerjen SUPB

### 3.3.1 Slovar

Dobra uporaba kompresije za dolge vrednosti posameznih nizov je uporaba slovarja. Vsakemu unikatnemu zapisu se doda številna vrednost. Slovar tako vsebuje vrednosti števil in nizov, ki so shranjeni kot metapodatki. Tako so podatki le številskih vrst, kar omogoča polja fiksne dolžine. Večje, kot je število enakih pojavitev določenega niza, manj prostora bo porabljenega za shranjevanje [15].

Če vzamemo primer nekaj dolgih nizov:

Slovar:

|   |                                                              |
|---|--------------------------------------------------------------|
| 1 | Shop Bare Boot Hiking Long Sleeve T-Shirt.                   |
| 2 | A T-shirt is a style of shirt.                               |
| 3 | Football kit has evolved significantly since the early days. |

Izgled strani s podatki:

|   |   |   |   |   |   |   |   |     |
|---|---|---|---|---|---|---|---|-----|
| 1 | 2 | 2 | 1 | 3 | 2 | 3 | 3 | ... |
|---|---|---|---|---|---|---|---|-----|

### 3.3.2 RLE kompresija (Run length encoding)

Vsak stolpec ima potencialno lahko več enakih zaporednih vrednosti. Npr. če imamo datumska polja in več izdelkov prodanih na isti dan, podatkovna baza kompresira te podatke na način, da shrani vrednost, začetno pozicijo vrednosti in število ponovitev. S tem se na stran dodajo metapodatki, ki dodatno povečajo velikost stolpca na strani[8].

V tabeli 5 je prikazan primer shranjenih podatkov, urejenih po kvartalih v naraščajočem vrstnem redu. Pri produktu in prodajalcu vidimo, da RLE kompresija ni uporabna, saj ni velikega števila ponovitev. RLE kompresija se tako razlikuje glede na stolpec v tabeli. Dobre stolpčno usmerjene podatkovne baze imajo to kompresijo realizirano tako, da pregledajo vse kolone in se potem odločijo, katera kompresija je najboljša za dane podatke [11].

| kvart | l | product_id | seller_id | quantity |
|-------|---|------------|-----------|----------|
| Q1    |   | 4          | 3         | 1        |
| Q1    |   | 3          | 8         | 5        |
| Q1    |   | 1          | 9         | 67       |
| Q1    |   | 15         | 4         | 6        |
| Q1    |   | 17         | 3         | 9        |
| Q1    |   | 1          | 2         | 6        |
| Q2    |   | 2          | 6         | 4        |
| Q2    |   | 5          | 15        | 5        |
| Q2    |   | 7          | 7         | 5        |
| Q2    |   | 8          | 17        | 5        |
| Q2    |   | 17         | 14        | 10       |
| Q2    |   | 15         | 16        | 8        |

|            | vrednost | začetna pozicija | število |
|------------|----------|------------------|---------|
| Kvartal    | Q1       | 1                | 500     |
|            | Q2       | 501              | 555     |
|            | Q3       | 1057             | 331     |
|            | Q4       | 1389             | 245     |
| Product_ID | 4        | 1                | 1       |
|            | 3        | 2                | 1       |
|            | 1        | 3                | 1       |
|            | 15       | 4                | 1       |
|            | 17       | 5                | 1       |
| Seller_ID  | 3        | 1                | 1       |
|            | 8        | 2                | 1       |
|            | 9        | 3                | 1       |

Tabela 5: Prikaz RLE kompresije

### 3.3.3 Krajšanje nizov (Trim Compression)

Krajšanje nizov odstrani odvečne bajte, ki nastanejo zaradi praznega prostora. S tem poskrbijo, da je velikost polja enaka velikosti dejanskega podatka, ki se nahaja v polju [11].

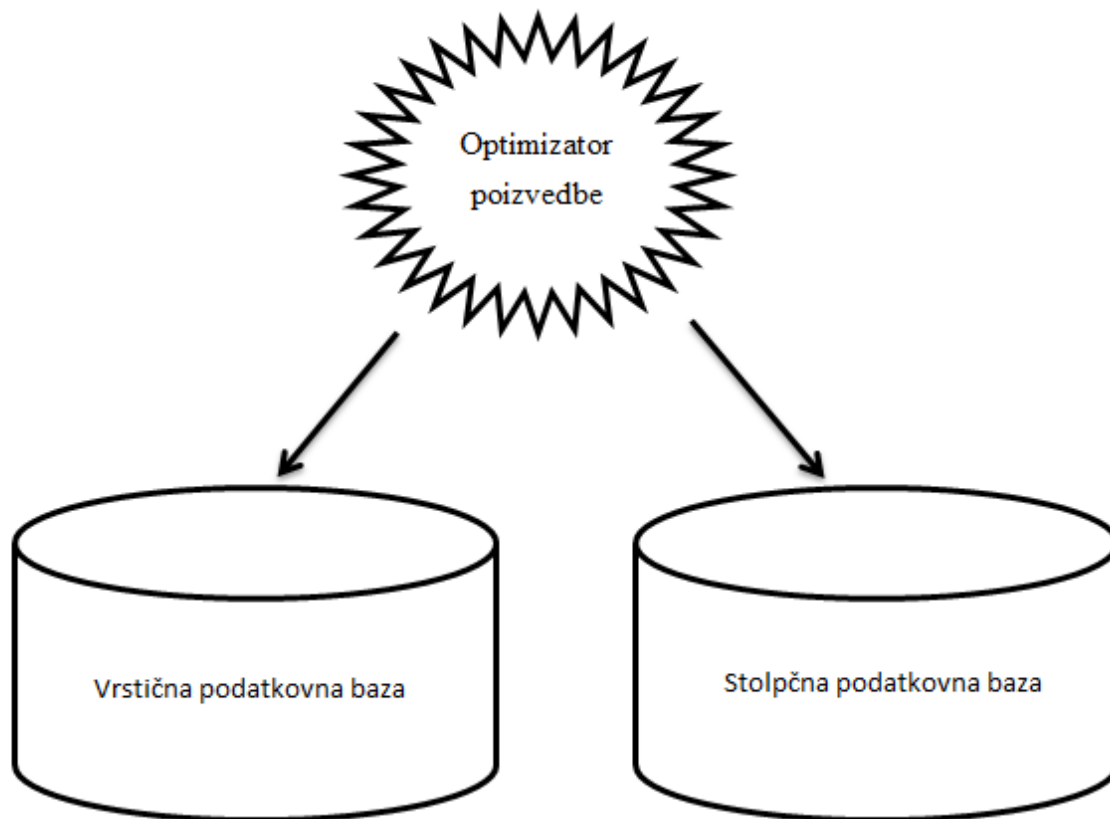
## 3.4 Souporaba (Coexistence)

V realnem svetu skorajda ne obstaja rešitev, ki bi ustrezala vsem zahtevam. Vedno je prisotnih več rešitev ali celo podjetij, ki ponujajo svoje rešitve. Nič drugače ni pri podatkovnih bazah, kjer določena podjetja v svojem podatkovnem skladišču uporabljajo obe podatkovni bazi – stolpčno in vrstično usmerjeno. Ta odločitev je prepuščena sistemskemu administratorju oz. vzdrževalcu podatkovne baze, ki se odloči, kako bo podatkovna baza organizirana – vrstično ali stolpčno.

Uporaba obeh podatkovnih baz je smiselna, kadar bi radi imeli dobre rezultate in manjše obremenitve sistema. Nekatera podjetja se tako odločajo za vzporedno uporabo obeh podatkovnih baz v podatkovnem skladišču. V vrstično podatkovno bazo se shranijo vsi podatki. Podatki, ki so primerni za različne analize, pa se, poleg v vrstično, shranijo še v stolpčno podatkovno bazo. Na tak način deluje SQL Server 2012, ki zahteva, da se vsi podatki shranijo v vrstično podatkovno bazo. Ko so podatki shranjeni, se izbere stolpce, ki se jih želi imeti zapisane tudi v stolpčni podatkovni bazi. S tem pride do podvajanja podatkov, saj so enaki podatki zapisani v obeh podatkovnih bazah. V nekaterih sistemih izbiro podatkovne baze določa optimizator poizvedbe (angl. Query optimizer), prikazan na sliki 5, ki izbere

## Stolpčno usmerjen SUPB

bazo, nad katero bo izvedena poizvedba. V primeru, da ta funkcionalnost ni podprta, se mora optimizator, ki bo pravilno izbiral podatkovno bazo glede na poizvedbo, narediti naknadno [11,16].



Slika 5: Optimizator poizvedbe izbira med podatkovnima bazama

### 3.5 Pregled različnih stolpčno usmerjenih SUPB

#### MonetDB:

MonetDB je odprtokoden stolpčno usmerjen sistem za upravljanje podatkovnih baz. Podatkovna baza je bila narejena z namenom izboljšanja performanc velikih tabel, ki vsebujejo veliko stolpcev in vrstic. MonetDB je bil eden izmed prvih podatkovnih sistemov, ki se je posvetil optimizaciji začasnega shranjevanja v procesorju (angl. CPU Cache). Ostale funkcije, ki jih vsebuje, so še samodejno indeksiranje, optimizacija poizvedb v realnem času in modularna programska arhitektura [17].

MonetDB sem uporabil v praktičnem delu naloge za primerjanje performanc med omenjeno podatkovno bazo, vrstično podatkovno bazo in različnimi implementacijami stolpčno usmerjene podatkovne baze v vrstičnem sistemu za upravljanje podatkovnih baz.

### **SAP Sybase IQ:**

Sybase IQ je bila nekaj let edina komercialna stolpčno usmerjena podatkovna baza. Družbo je dobila v zadnjih letih, ko so na trg stopili še ostali ponudniki stolpčno usmerjenih podatkovnih baz. Podjetje Sybase je od leta 2010 naprej last podjetja SAP, ki se med drugim ukvarja z BI orodji, zato je glavna lastnost te baze dober sistem za upravljanje podatkovnih baz, ki omogoča izdelavo analitičnih poročil, napredno analizo in skladiščenje velikih količin podatkov. Po zadnjih podatkih naj bi bila ta podatkovna baza nameščena že na okoli 2000 strežnikih [18,19].

### **Infobright:**

Infobright ponuja komercialno (IEE) in brezplačno (ICE) različico. To je kombinacija stolpčne podatkovne baze z njihovo »Knowledge Grid« arhitekturo. Njihov večji dosežek je zagotovo velika kompresija podatkov, ki je v razmerju od 10:1 do 40:1 boljše od ostalih podatkovnih baz, kar močno zmanjša potrebo po velikih diskovnih prostorih in dragi strojni opremi. Podatki v tej podatkovni bazi so shranjeni v podatkovnih paketih. Vsako vozlišče za svoje podatke že vsebuje vnaprej izračunane statistike. Mreža znanja (angl. Knowledge Grid) poskrbi za metapodatke podatkov, ki se nahajajo v podatkovni bazi. Potrebe po indeksih, particioniranju ali agregiranih tabelah ni, saj se različne statistike izračunavajo in upravljajo samodejno. Mreža znanja je narejena celo do take mere, da nekaterim poizvedbam ni potrebno dostopati do podatkov v bazi in odgovor najdejo že v sami mreži [19].

### **Vertica (HP):**

Vertica Systems je podjetje, ki se ukvarja z analitičnimi podatkovnimi sistemi. Ustanovili so ga leta 2005, od leta 2011 pa je podjetje v lasti HP. Kot produkt ponujajo tudi stolpčno podatkovno bazo, ki je bila narejena z namenom upravljanja velike količine hitro rastočih podatkov. Baza zagotavlja dobre performance in hitre izvedbe poizvedb v realnem času [19].

### **Microsoft SQL Server 2012:**

Microsoft je šele z izidom strežnika SQL 2012 izdal prvo idejo o stolpčni podatkovni bazi. Trenutna izdaja strežnika vsebuje funkcijo imenovano xVelocity, ki podatke shranjuje podobno kot stolpčno usmerjen sistem za upravljanje podatkovnih baz. Ta tehnika omogoča izdelavo pomnilniško optimiziranih indeksov, ki združijo in shranijo podatke vsakega stolpca.

## **Stolpčno usmerjen SUPB**

Stolpce med seboj povežejo in s tem dobijo popolni indeks. Indeksi so uporabni pri velikih tabelah dejstev v zvezdni shemi, kjer izboljšujejo zmogljivosti [16,19].

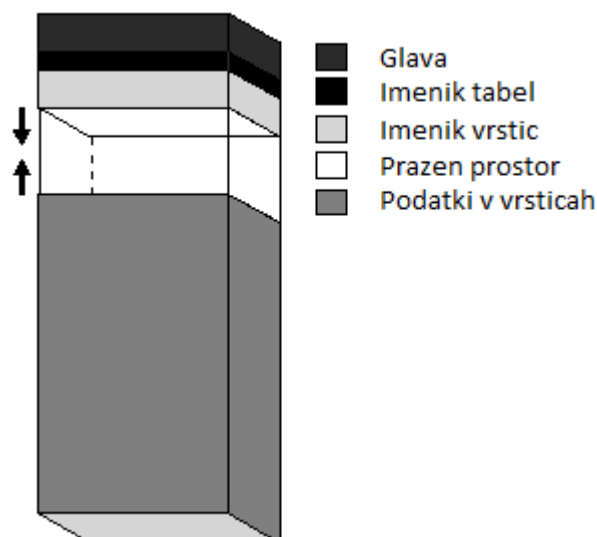
## Poglavje 4

### 4 Primerjava stolpčne z vrstično podatkovno bazo

#### 4.1 V/I zmogljivosti

Največja razlika med stolpčno in vrstično podatkovno bazo je število bralnih in pisalnih ciklov.

Podatki so na trdi disk shranjeni v straneh oz. blokih [20]. Disk je naprava, ki je sestavljena iz večih okroglih kovinskih plošč. Plošče so prevlečene z magnetno plastjo in razdeljene v trakove, trakovi pa so razdeljeni na sektorje ali bloke. Blok je najmanjša enota, ki se jo lahko bere ali zapisuje na disk in je prikazan na sliki 6. Ponavadi so bloki dolgi 512 bajtov. Za podatkovno bazo to pomeni, da tudi v primeru, kadar bi prebrala samo 1 bajt podatkov, prebere celotno dolžino bloka. Kadar govorimo o zmogljivosti, je potrebno dodati še iskalni čas za pridobitev pravilnega naslova na disku in čas dejanskega branja podatka iz diska. Za vrstično podatkovno bazo lahko predvidevamo, da se v enem sektorju nahaja več zapisov. Ker se posamezni atributi ne morejo prebrati iz bloka, mora podatkovna baza prebrati celoten blok podatkov. Tako je vrstična podatkovna baza neučinkovita v primeru majhnega števila atributov v poizvedbi.



Slika 6: Blok v podatkovni bazi Oracle

## Stolpčno usmerjen SUPB

### Glava (skupna in posameznih vrednosti):

Glava vsebuje splošne informacije. Te so npr. naslov bloka in tip vrednosti (ali gre za podatke ali indekse).

### Imenik tabel (angl. Table directory):

Ta del bloka vsebuje informacije o tabeli, katere podatki oz. zapisi se nahajajo v tem delu bloka.

### Imenik vrstic (angl. Row directory):

Imenik vrstic vsebuje informacije o dejanskih zapisih v bloku (med drugim tudi naslov za vsak posamezen zapis). Za zapise v bazi se v imenik vrstic dodeli določeno količino prostora. V primeru, da pride do izbrisa določenih zapisov, se ta prostor ne pobriše in ostane prazen. To je tako imenovana prekomerna količina podatkov (overhead). Prekomerna količina podatkov se pobriše, kadar se v podatkovno bazo dodajo novi zapisi.

### Podatki v vrsticah (angl. Row data):

Tu se nahajajo dejanski podatki oz. zapisi v tabeli ali indeksih. V primeru, da so tabele večje, lahko segajo čez večje število blokov.

#### 4.1.1 Primer prikaza V/I operacij

Za primer prikaza vhodno/izhodnih operacij vzemimo tabelo (tabela 6), ki vsebuje podatke o naročilih (ime kupca, kraj dostave in regija dostave). Tabela je velika 10 milijonov vrstic in ima povprečno dolžino vrstice 91 bajtov.

| Ime kupca | Kraj dostave | Regija dostave |
|-----------|--------------|----------------|
| John      | Liverpool    | UK             |
| Daniel    | Liverpool    | Merseyside     |
| George    | Los Angeles  | East LA        |
| Daniel    | Liverpool    | Merseyside     |
| Daniel    | Tokio        | East Kanto     |

Tabela 6: Primer podatkov

Vprašajmo se:

Koliko izdelkov je potrebno dostaviti v mesto Liverpool, ki se nahaja v regiji Merseyside, in jih je kupil kupec z imenom Daniel?

Za tako poizvedbo v vrstični podatkovni bazi se izračuna število vhodno/izhodnih operacij s spodnjo formulo:

$$\frac{\text{dolžina vrstice} \times \text{število zapisov v tabeli}}{\text{velikost strani}}$$

Ker za poizvedbo preberemo vse vrstice, je število zapisov v tabeli 10 milijonov. Povprečna dolžina vrstice, ki se nahaja v tabeli Orders (iz praktičnega primera), je 91 bajtov. Za velikost strani pa vzemimo 16K.

$$\frac{91 \text{ B} \times 10 \text{ M}}{16 \text{ K}} = 56875 \text{ I/O operacij}$$

Enak izračun naredimo še za stolpčno podatkovno bazo. Ker gre za princip pozne izdelave n-teric, se podatki nahajajo v bitih. Tam, kjer je pogoj v poizvedbi zadoščen, se bit nastavi na 1, drugače 0 (prikazano v tabeli 7).

|   |   |   |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 1 | 0 | 0 |

Tabela 7: Zgrajene n-terice iz zgornjih podatkov

Za tako poizvedbo v stolpčni podatkovni bazi izračunamo število vhodno/izhodnih operacij:

$$\frac{\text{število zapisov v stolpcu} \times \text{število stolpcev v poizvedbi}}{\text{velikost strani}} \div 8$$

Število zapisov v stolpcu je enako kot v vrstični podatkovni bazi (10 milijonov). Število stolpcev, ki se nahajajo v poizvedbi, je 3. S številko 8 je potrebno deliti, ker potrebujemo podatek v bajtih. Velikost strani je 16 K.

## Stolpčno usmerjen SUPB

$$\frac{10\text{ M} \times 3 \div 8}{16\text{ K}} = 234\text{ I/O operacij}$$

Kot vidimo iz zgornjega primera, je razlika očitna. Vrstična podatkovna baza poleg zapisov, ki jih potrebuje, prebere še velike količine ostalih, neuporabljenih zapisov in podatkov. Zgornji primer je prikazan na vrstični podatkovni bazi, ki ne uporablja indeksa, kar je tudi razlog za tako veliko razliko.

## 4.2 Primerjava sistemov za upravljanje podatkovnih baz

V vrstični podatkovni bazi branje celotnih datotek s podatki pomeni veliko količino odvečnih vhodno/izhodnih ciklov za branje podatkov, ki so nekoristni v dani poizvedbi. Nekoristni podatki lahko predstavljajo velik procent vhodno/izhodnih ciklov. V stolpčno usmerjeni podatkovni bazi se ta procent precej zmanjša, saj je velika večina prebranih podatkov zahtevana v poizvedbi.

### 4.2.1 Vrstični relacijski SUPB (R-RDBMS)

V vrstičnem sistemu za upravljanje podatkovnih baz so podatki predstavljeni v n-tericah (vrsticah). Te so na podatkovni disk shranjene ena za drugo. SUPB uporablja strani za zapisovanje in branje podatkov iz oz. na podatkovni disk. Če je velikost zapisa v vrstici manjša od polovične velikosti strani, se na strani zapiše več vrstic. Kadar pride zahteva za branje določenega zapisa, se v pomnilnik prebere celotna stran z vsemi zapisi.

Relacijski SUPB, poleg podatkov na stran, shranjuje tudi dodatne informacije o strani in zapisih, ki se nahajajo na njej. Za izboljšanje performanc sistema ta lahko uporablja indekse. Indeks shrani vrednost zapisa, stran, kjer se ta zapis nahaja, in odmik, ki je potreben za dostop do željenega podatka. Če poizvedba vsebuje stolpec, ki je indeksiran, lahko SUPB uporabi indeks za identifikacijo zapisov in s tem prebere le strani, ki so aktualne in zadoščajo pogoju v poizvedbi. V primeru, da indeks ni prisoten, mora SUPB za željen zapis prebrati vse strani, ki so povezane s tabelo, ki se nahaja v poizvedbi [21].

## Stolpčno usmerjen SUPB

V vrstični podatkovni bazi si podatki sledijo eden za drugim:

|                |                      |                      |                      |                      |
|----------------|----------------------|----------------------|----------------------|----------------------|
| glava vrstica0 | stolpec0<br>vrednost | stolpec1<br>vrednost | stolpec2<br>vrednost | stolpec3<br>vrednost |
| glava vrstica1 | stolpec0<br>vrednost | stolpec1<br>vrednost | stolpec2<br>vrednost | stolpec3<br>vrednost |
| glava vrstica2 | stolpec0<br>vrednost | stolpec1<br>vrednost | stolpec2<br>vrednost | stolpec3<br>vrednost |

Glava vrstice vsebuje dolžino zapisa, vrednosti bita null ...

Če bi za zgornje podatke odprli datoteko, v kateri se nahajajo, bi izgledala takole:

|       |          |       |          |       |          |
|-------|----------|-------|----------|-------|----------|
| glava | vrednost | glava | vrednost | glava | vrednost |
|-------|----------|-------|----------|-------|----------|

Več, kot je zapisov, večja je velikost datoteke. Podatki v vrstici so shranjeni po določenem odmiku. Sam odmik je odvisen od tipa stolpca (4 biti za številski tip int32, 8 bitov za številski tip int64).

| glava | vrednost |    |    | glava | vrednost |    |    | glava | Vrednost |    |    |
|-------|----------|----|----|-------|----------|----|----|-------|----------|----|----|
|       | c0       | c1 | c2 |       | c0       | c1 | c2 |       | c1       | c2 | c3 |

Za prebiranje posamezne vrstice iz vrstične podatkovne baze se mora najprej izračunati, kje se ta vrstica nahaja [21]. To je povsem preprosto, če so dolžine vrstic končnih dolžin:

Položaj = številka vrstice \* dolžina vrstica + dolžina glave

Dolžina vrstice = glava vrstice + dolžina vrednosti

Dolžina vrednosti = dolžina c0 + dolžina c1 + dolžina c2

## Stolpčno usmerjen SUPB

Primer branja iz vrstične podatkovne baze prikazuje tabela 8:

|     |      |         |    |
|-----|------|---------|----|
| 1   | Mike | Jones   | 26 |
| 2   | Jane | Rodgers | 37 |
| 3   | Eva  | Mendes  | 33 |
| ... |      |         |    |
| N   |      |         |    |

Tabela 8: Analitična zahteva v vrstični podatkovni bazi

V realnem svetu seveda to ni tako preprosto. Obstajajo še npr. podatkovna polja, ki imajo polja različnih dolžin. Tako se ne da preprosto prebrati samo določenega dela željenega zapisa, saj podatkovna baza nikakor ne more vedeti, kje se ta zapis nahaja. Za izračun odmika posamezne vrstice potrebujemo vse dolžine zapisov, ki so shranjene v glavah, kar povzroči odvečno porabo diskovnega prostora (angl. overhead). Vsi podatki se nahajajo v straneh (angl. data page-ih). Če bi radi prebrali le določeno vrstico ali stolpec, je potrebno prebrati celotno stran, kar povzroči prebiranje večih zapisov kot jih potrebujemo. Krajše, kot so vrstice, več podatkov se bo lahko shranilo na stran. V realnem svetu strani niso popolnoma zasedene, saj morajo imeti prostor v primeru sprememb. Stran je enota vhodno/izhodnih operacij, ki so ozko grlo vrstičnih podatkovnih baz. Vrstice morajo ustrezati velikosti strani, s tem pa lahko pride do neuporabljenih delov strani.

| stran |           |    |    |                         | Stran |           |    |    |       |           |    |    |
|-------|-----------|----|----|-------------------------|-------|-----------|----|----|-------|-----------|----|----|
| glava | vrednos i |    |    | neuporabljen<br>prostor | glava | vrednosti |    |    | glava | vrednosti |    |    |
|       | c0        | c1 | c2 |                         |       | c0        | c1 | c2 |       | c0        | c1 | c2 |

### Učinkovitost vrstične podatkovne baze:

Vrstične podatkovne baze so dobre, če potrebujemo podatke ene vrstice. Ta zahteva ponavadi potrebuje branje oz. zapisovanje le ene strani. Koristne so tudi, kadar bi radi prebrali/zapisali vse ali večino stolpcev.

### Neučinkovitost vrstične podatkovne baze:

Neučinkovitost vrstične podatkovne baze se pokaže, če ne potrebujemo vseh stolpcev, potrebujemo pa veliko vrstic. V tem primeru podatkovna baza prebere vse vrstice, vključno s stolpci, ki jih poizvedba ne vsebuje. Ker strani ponavadi niso popolnoma zasedene, pride do branja neuporabljenega prostora. Poizvedba skupaj s podatki prebere tudi vse glave vrstic (tudi glave strani), ki ne vsebujejo dejanskih podatkov.

### 4.2.2 Stolpčni relacijski SUPB (C-RDBMS)

Stolpčno usmerjena podatkovna baza poizveduje po stolpcih, kjer je vsak stolpec shranjen v svoji datoteki in obravnavan samostojno. Podatki se v stolpcih nahajajo zaporedoma, kar omogoča hitro procesiranje podatkov. To vodi v izboljšanje zmogljivosti, ker poizvedbe preberejo le tiste stolpce, ki jih vsebujejo.

Primer: Tabela vsebuje 9GB podatkov:

stolpec1: 2GB

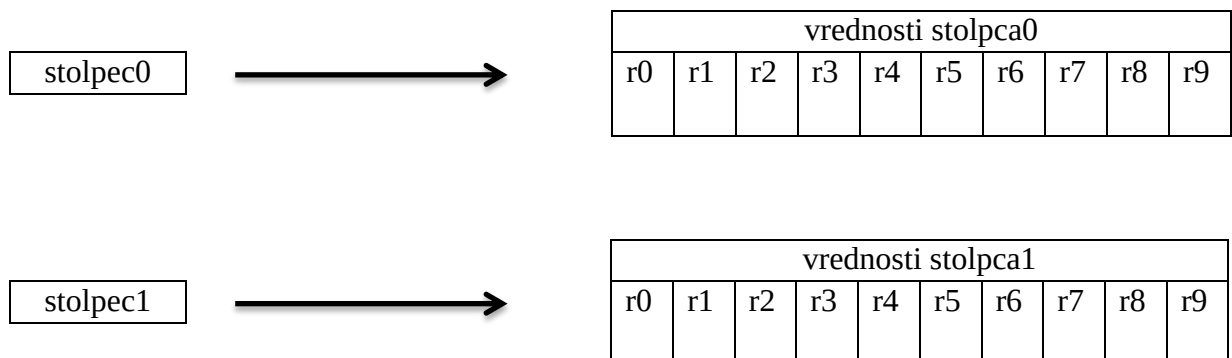
stolpec2: 3GB

stolpec3: 1GB

stolpec4: 3GB

V poizvedbi se nahaja stolpec3. Stolpčna podatkovna baza bi tako prebrala največ 1GB podatkov, medtem ko bi vrstična podatkovna baza prebrala vseh 9GB podatkov.

Spodnja slika prikazuje vrednosti stolpcev, ki so v stolpčni podatkovni bazi shranjeni eden za drugim:



## Stolpčno usmerjen SUPB

Z vsakim branjem se prebere več vrednosti kot v vrstični podatkovni bazi. Tako je potrebnih manj vhodno/izhodnih ciklov za prebiranje večje količine podatkov. V primeru, da so vrednosti vrstic končnih dolžin, lahko do njih dostopamo naključno. Metapodatki se v stolpčni podatkovni bazi shranijo ločeno od vrednosti (npr. pozicije, kjer so null vrednosti, se shranijo kot vektor za vsak stolpec). Ker so podatki v vsakem stolpcu enakih podatkovnih tipov, je to idealno za kompresijo podatkov. Kompresija podatkov zmanjša število diskovnih vhodno/izhodnih ciklov. Za večje zbirke podatkov je to ključno, saj je dekompresija podatkov v CPU v veliki meri hitrejša od hitrosti branja iz podatkovnega diska [21].

Primer branja iz stolpčne podatkovne baze prikazuje tabela 9:

|   |      |         |    |
|---|------|---------|----|
| 1 | Mike | Jones   | 26 |
| 2 | Jane | Rodgers | 37 |
| 3 | Eva  | Mendes  | 33 |
| . |      |         |    |
| . |      |         |    |
| . |      |         |    |
| n |      |         |    |

Tabela 9: Analitična zahteva v stolpčni podatkovni bazi

Podatki v stolpčni podatkovni bazi so po navadi grupirani v pakete določenih velikosti. Za vsak paket se izračunajo podatki, kot so npr:

- Najmanjša vrednost v paketu,
- največja vrednost v paketu,
- število ne ničtih (not null) vrednosti v paketu,
- histogrami in
- kompresija metapodatkov.

Dobra lastnost teh metapodatkov je, da vsebujejo informacije o podatkih. V primeru, da neka poizvedba ne potrebuje podatka iz določenega paketa, se vrednosti paketa ne preberejo, kar zmanjša število vrednosti, ki jih mora poizvedba procesirati.

### 4.3 Shranjevanje v podatkovno bazo

Tabela vsebuje podatke ID, podatkovna tipa (ime in priimek) ter starost. Dvodimenzionalni model obstaja samo na konceptualnem nivoju. V fizičnem nivoju se dvodimenzionalna tabela

## Stolpčno usmerjen SUPB

premapira v enodimenzijsko strukturo, ta pa se shrani v magnetni disk ali začasni pomnilnik (RAM), iz katerega beremo oz. zapisujemo vrstico po vrstico (s pomočjo linearnega odmika). Poznamo dva očitna načina mapiranja podatkov v enodimenzijsko strukturo. Eden od načinov je zapisovanje podatkov vrstico za vrstico, drugi pa stolpec za stolpcem. Način, ki zapisuje podatke vrstice za vrstico, hrani skupaj vse podatke, ki pripadajo isti entiteti (primer komitenta – skupaj so shranjeni vsi njegovi podatki – lastno ime, priimek, starost ...). V vrstični podatkovni bazi si podatki sledijo v tabularičnem vrstnem redu po vrsticah eden za drugim. Ko se določena vrstica konča z zadnjim podatkom, mu sledi podatek, ki se nahaja na začetku novega zapisa v naslednji vrstici.

Nov zapis se v vrstično podatkovno bazo doda na konec tabele. Primer dodajanja novega zapisa je prikazan v tabeli 10.

| ID | Name   | Surname | Age |
|----|--------|---------|-----|
| 1  | Mike   | Jones   | 26  |
| 2  | Jane   | Rodgers | 37  |
| 3  | Eva    | Mendes  | 33  |
| 4  | Daniel | Owen    | 25  |

Tabela 10: Dodajanje novega zapisa v vrstično podatkovno bazo

Drugi način – zapisovanje podatkov stolpec za stolpcem hrani skupaj vse informacije o določenem atributu (pri primeru komitenta banke – ta način skupaj hrani vsa lastna imena oz. druge lastnosti komitentov). Shranjevanje podatkov v stolpčno podatkovno bazo je dokaj podobno shranjevanju podatkov v vrstično podatkovno bazo. Stolpčna podatkovna baza shrani skupaj vse podatke enega stolpca, ki mu sledijo ostali stolpci. Tako zadnjemu podatku v določenem stolpcu sledi prvi podatek, ki se nahaja v naslednjem stolpcu, kot je prikazano v tabeli 11.

Ker je v stolpčni podatkovni bazi vsak atribut predstavljen s svojo datoteko, se zapis doda na konec vsake datoteke.

| ID | Name   | Surname | Age |
|----|--------|---------|-----|
| 1  | Mike   | Jones   | 26  |
| 2  | Jane   | Rodgers | 37  |
| 3  | Eva    | Mendes  | 33  |
| 4  | Daniel | Owen    | 25  |

Tabela 11: Dodajanje zapisa v stolpčno podatkovno bazo

## Stolpčno usmerjen SUPB

V stolpčni podatkovni bazi je vstavljanje novih podatkov zamudna operacija. Iz tega vidika stolpčne podatkovne baze tudi niso primerne za transakcijske sisteme. Vstavljanje podatkov v podatkovno bazo je priporočljivo v seriji paketnih uvozov [22].

### 4.4 Brisanje zapisa

Brisanje zapisa je podobno kot dodajanje novega zapisa. V vrstični podatkovni bazi se pobriše celotna vrstica s podatki, kar ponazarja tabela 12.

| ID | Name | Surname | Age |
|----|------|---------|-----|
| 1  | Mike | Jones   | 26  |
| 2  | Jane | Rodgers | 37  |
| 3  | Eva  | Mendes  | 33  |

Tabela 12: Brisanje zapisa v vrstični podatkovni bazi

V stolpčni podatkovni bazi se pobriše zapis iz vsake datoteke posebej. Brisanje zapisa v stolpčni podatkovni bazi je prikazan v tabeli 13.

| ID | Name | Surname | Age |
|----|------|---------|-----|
| 1  | Mike | Jones   | 26  |
| 2  | Jane | Rodgers | 37  |
| 3  | Eva  | Mendes  | 33  |

Tabela 13: Brisanje zapisa v stolpčni podatkovni bazi

### 4.5 Dodajanje novega atributa

Atribut, ki se doda v tabelo, se mora dodati za vsak zapis, ki se nahaja v tabeli, kot je prikazano v tabeli 14.

| ID | Name | Surname | Age | Email        |
|----|------|---------|-----|--------------|
| 1  | Mike | Jones   | 26  | mike@it.com  |
| 2  | Jane | Rodgers | 37  | jane@me.at   |
| 3  | Eva  | Mendes  | 33  | eva_m@gm.com |

Tabela 14: Dodajanje zapisa v vrstično podatkovno bazo

V stolpčni podatkovni bazi se doda le nova datoteka, uredijo pa se tudi metapodatki. Tabela 15 prikazuje dodajanje novega atributa v stolpčni podatkovni bazi.

| ID | Name | Surname | Age | Email        |
|----|------|---------|-----|--------------|
| 1  | Mike | Jones   | 26  | mike@it.com  |
| 2  | Jane | Rodgers | 37  | jane@me.at   |
| 3  | Eva  | Mendes  | 33  | eva_m@gm.com |

Tabela 15: Dodajanje zapisa v stolpčni podatkovno bazo

### 4.6 Ažuriranje podatkov

V primeru ažuriranja določenega zapisa sta podatkovni bazi primerljivi. V obeh primerih gre za spremembo določene vrednosti. Za vrstično podatkovno bazo je primer spremembe zapisa prikazan v tabeli 16.

| ID | Name | Surname                 | Age |
|----|------|-------------------------|-----|
| 1  | Mike | <del>Jones</del> Miller | 26  |
| 2  | Jane | Rodgers                 | 37  |
| 3  | Eva  | Mendes                  | 33  |

Tabela 16: Ažuriranje zapisa v vrstični podatkovni bazi

Stolpčna podatkovna baza se zaveda, katere vrednosti pripadajo določenemu zapisu. Iz tega vidika je urejanje atributov enako kot v vrstični podatkovni bazi. Tabela 17 prikazuje spremembo zapisa v stolpčni podatkovni bazi [22].

| ID | Name | Surname                 | Age |
|----|------|-------------------------|-----|
| 1  | Mike | <del>Jones</del> Miller | 26  |
| 2  | Jane | Rodgers                 | 37  |
| 3  | Eva  | Mendes                  | 33  |

Tabela 17: Ažuriranje zapisa v stolpčni podatkovni bazi

### 4.7 Prednosti/slabosti

Stolpčna podatkovna baza je veliko bolj primerna za analitične poizvedbe kot vrstična. V vrstični podatkovni bazi se zmogljivosti v primeru večjih sočasnih poizvedb precej zmanjšajo. Izdelava dodatnih indeksov na atributih v tabeli za hitrejšo poizvedovanje porabi več prostora in procesorskega časa za vzdrževanje. Z vidika zmogljivosti ta način postane neučinkovit pri veliki količini raznolikih poizvedb. V stolpčni podatkovni bazi so indeksi zgrajeni za shranjevanje podatkov, medtem ko v vrstični podatkovni bazi predstavljajo kazalec na vrstico v disku, kjer se zapis nahaja. Prednost stolpčne podatkovne baze je tudi, da prebere le vrednosti oz. attribute, ki se nahajajo v poizvedbi, kar omogoča hitrejšo izvedbo poizvedb. S tem zmanjša število bralnih ciklov, saj iz blokov, ki se nahajajo na disku, prebere samo

## Stolpčno usmerjen SUPB

vrednosti, ki so v dani poizvedbi aktualne. Stolpčno usmerjen sistem je bolj učinkovit pri veliki količini vrstic s podatki in malim številom stolpcev (atributov). Če je primer obraten in je potrebnih več stolpcev (z manjšim številom podatkov), je bolj uporabna vrstična podatkovna baza. Kljub obstoju nekaj možnosti implementacij stolpčne podatkovne baze v vrstičnem sistemu, npr. vertikalno particioniranje tabel ali izdelava indeksirano organizirane tabele, so performance še vedno precej slabše od stolpčne podatkovne baze. Velik del zaslug za slabše performance vrstičnih podatkovnih baz imajo vhodno/izhodne operacije, ki so ozko grlo podatkovnih baz. V stolpčni podatkovni bazi so tabele že sortirane in jih ni potrebno urejati pred povezovanjem drugih tabel. Druga prednost, ki jo stolpčne baze ponujajo, je vzporeden dostop do različnih atributov, kar občutno izboljša zmogljivosti sistema. Stolpčne podatkovne baze v primerjavi z vrstičnimi potrebujejo manj diskovnega prostora, ker ne potrebujejo dodatnih indeksov, dimenzij z agregacijmi ali materializiranih pogledov. Za to poskrbijo indeksi, ki so uporabljeni za optimizacijo shranjevanja in pridobivanja podatkov. Stolpčne podatkovne baze imajo v primerjavi z vrstičnimi tudi boljšo kompresijo podatkov. Ker stolpci vsebujejo ponavljajoče se vrednosti ali pa so si vrednosti med seboj zelo podobne, jim to omogoča boljšo kompresijo podatkov. Pri vrstičnih podatkovnih bazah lahko vrstica s podatki vsebuje različne tipe atributov in posledično tudi slabšo kompresijo podatkov. Poizvedbe SQL se med vrstično in stolpčno podatkovno bazo ne razlikujejo [7].

Prednosti stolpčne podatkovne baze:

- Stolpčna podatkovna baza bere podatke blok za blokom (vrstična podatkovna baza bere podatke vrstico za vrstico),
- operacije nad blokom omogočajo različne optimizacije (paralelno izvajanje, izboljšanje ad-hoc procesiranja (angl. Pipelining – izboljšuje hitrost izvajanja poizvedb),
- preberejo se le atributi, ki so zahtevani v poizvedbi,
- dobra kompresija podatkov (skupaj so shranjeni podobni podatki),
- hitrost agregacij podatkov.

Slabosti stolpčno podatkovnih baz (proti vrstični):

- V stolpčni podatkovni bazi je prebiranje vseh atributov ene vrstice zelo draga operacija, zato so bolj primerne vrstične,
- niso primerne za transakcijske sisteme,
- čas polnjenja podatkov je daljši,
- poizvedbe z velikim številom povezav na druge tabele lahko poslabšajo hitrost izvajanja poizvedb,

## **Stolpčno usmerjen SUPB**

- tabele z manjšim številom zapisov se izvedejo počasneje,
- v primeru, da stolpec vsebuje veliko enoličnih zapisov, kompresija ni učinkovita.



## Poglavje 5

### 5 Praktični primer

Za potrebe praktičnega primera sem izdelal podatkovni model, ki bo uporabljen v vrstični in stolpčni podatkovni bazi. Glavni poudarek praktičnega dela bo performančna analiza med podatkovnima bazama. Za doseg tega cilja sem uporabil dva načina, ki vrstično podatkovno bazo približata stolpčni. To sta vertikalno particioniranje in indeksirano organizirana tabela.

V praktičnem primeru bom prikazal performančno analizo vertikalnega particioniranja tabel, indeksirano organizirane tabele, vrstične podatkovne baze (Oracle) in stolpčne podatkovne baze (MonetDB). Analiza bo temeljila na hitrosi izvedbe raznih poizvedovalnih stavkov (SELECT, DELETE, UPDATE, INSERT).

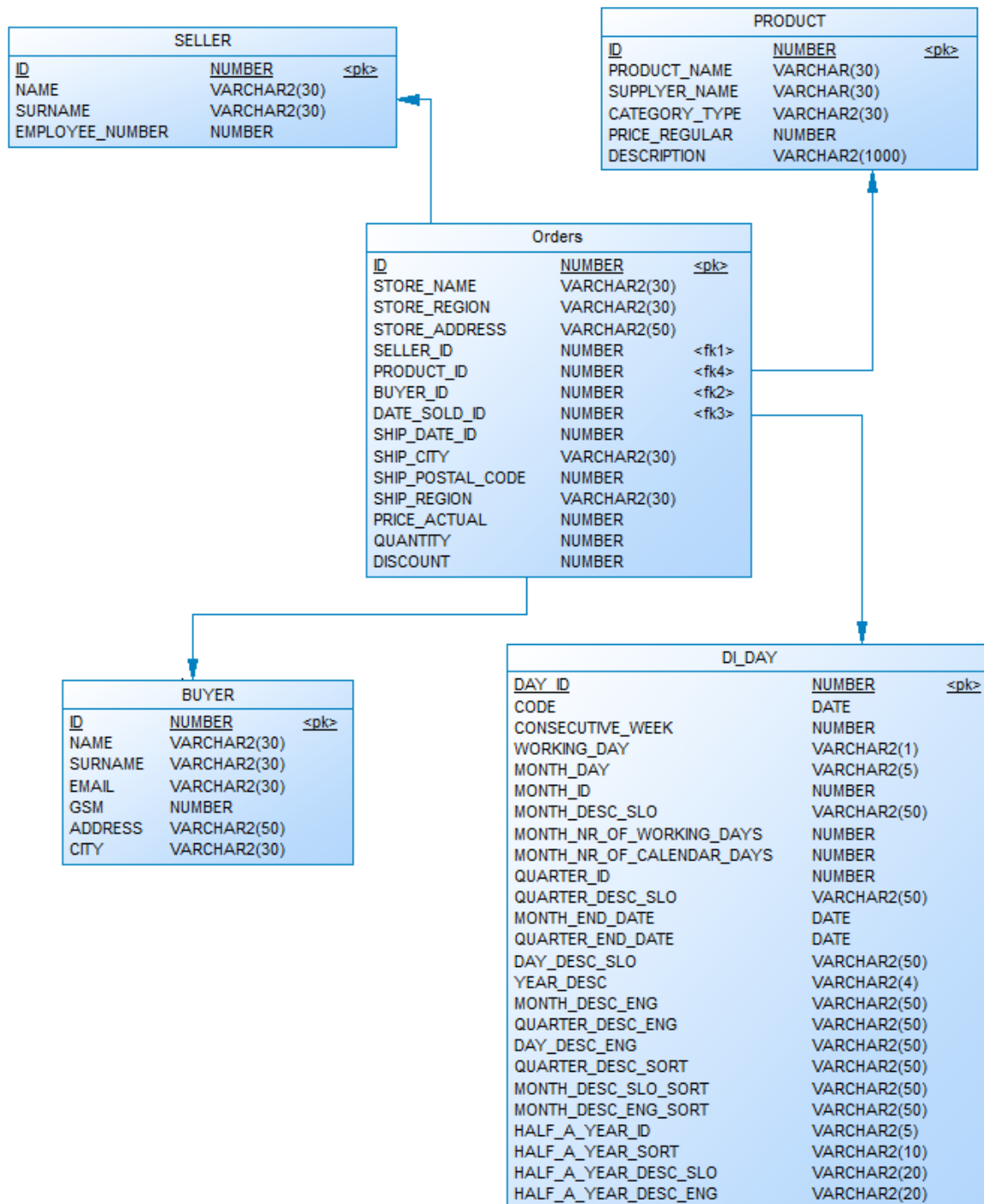
Podatkovni bazi se nahajata na enakem strežniku. Za strežnik sem uporabil računalnik, ki se nahaja v lokalnem omrežju in ima naslednje karakteristike:

- 3.0Ghz dvojedrni procesor (CPU),
- 2GB začasnega pomnilnika (RAM),
- 200GB trdega diska s 7200 obrati na minuto.

#### 5.1 Podatkovni model

Podatkovni model, prikazan na sliki 7, sem izdelal za potrebe diplomskega dela in je namenjen prikazu ter performančni analizi med vrstično podatkovno bazo, različnimi načini implementacije stolpčne podatkovne baze v vrstičnem sistemu za upravljanje podatkovnih baz in stolpčne podatkovne baze. Model je sestavljen iz petih tabel, ki so med seboj povezane s primarnimi in tujimi ključi. Podatkovni model ima tabele razporejene v obliki zvezde, zato ga imenujemo zvezdna shema. Osrednjo tabelo imenujemo tabela dejstev, ostale tabele pa so dimenzije. V splošnem se tabele dejstev večajo in polnijo z novimi podatki hitreje od dimenzij, ki se večajo počasi. Testne podatke sem si izmislil. Kot primer sem vzel trgovino, kjer nakupuješ izdelke. Za vsak prodan izdelek je mogoče preveriti ceno prodanega izdelka, kdo je izdelek prodal oz. kupil.

## Stolpčno usmerjen SUPB



Slika 7: Podatkovni model

### 5.2 Testiranje in analiza performanc

Testiranje performanc sem razdelil na 4 večje sklope:

- Pregled podatkov (SELECT),
- ažuriranje podatkov (UPDATE),
- vstavljanje novih podatkov (INSERT),
- brisanje podatkov (DELETE).

Najbolj podrobno bo razdelan sklop pregled podatkov. Za vsako testiranje bo v tabeli prikazan rezultat izvajanja določene poizvedbe. Podatkovna baza je enaka za indeksirano organizirano tabelo, vertikalno particioniranje tabel in vrstično podatkovno bazo. V vseh treh primerih je uporabljena podatkovna baza Oracle 11g. Za stolpčno podatkovno bazo je uporabljena odprtokodna podatkovna baza MonetDB. Upoštevati je potrebno, da v nobeni tabeli ni narejenih dodatnih indeksov (tabele vsebujejo primarne ključe).

Število zapisov v podatkovni bazi je nekaj več kot 9 milijonov. Osrednja tabela dejstev – Orders vsebuje 9.291.249 zapisov. Na tej tabeli bo tudi največji poudarek v praktičnem delu. Tabela Seller vsebuje 32 zapisov, tabela Buyer prav tako 32, Product 15.513 in tabela Di\_day 18.631.

Za izračun časa izvajanja posamezne poizvedbe bo uporabljena že vgrajena funkcija *dbms\_utility.get\_time()*.

#### 5.2.1 Pregled podatkov

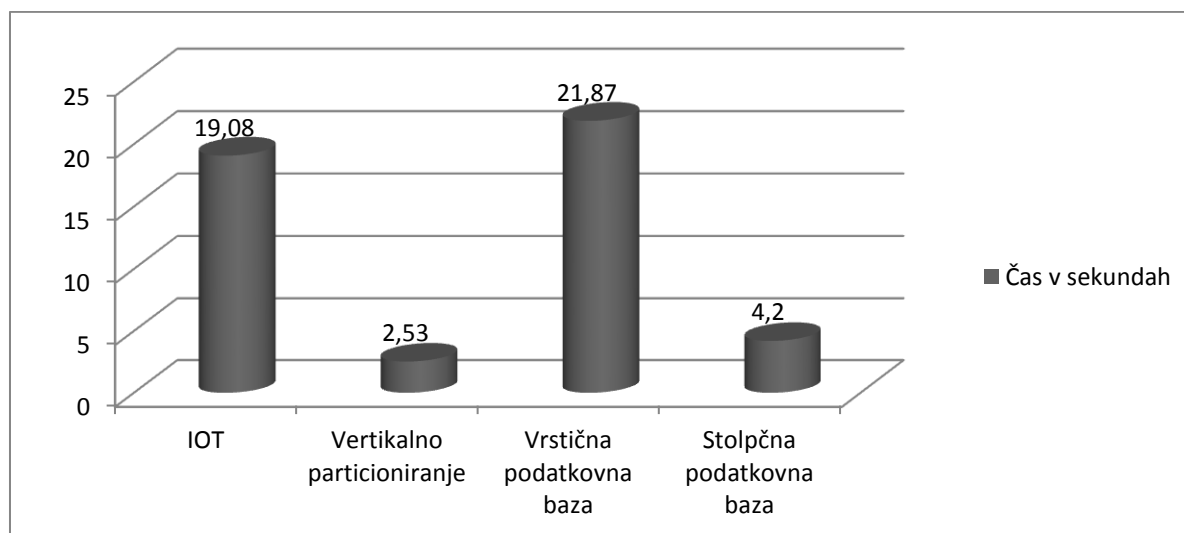
Za prvo testiranje performanc bo prikazana povprečna cena prodanega izdelka. Za ta namen bo polno vertikalno particionirana tabela Orders, tako da bo vsebovala samo *product\_id* in *price\_actual*.

Poizvedba 1:

```
SELECT AVG(price_actual) FROM orders;
```

Rezultati za poizvedbo 1 so vidni na sliki 8.

## Stolpčno usmerjen SUPB



Slika 8: Čas izvajanja poizvedbe 1

Rezultati so dokaj presenetljivi. Pričakoval sem, da bo stolpčna podatkovna baza pometla s konkurenco in zasedla prvo mesto. Kljub temu da jo je prehitel postopek vertikalnega particioniranja, ima še vedno 5x boljši rezultat od vrstične podatkovne baze. Omeniti je potrebno, da je v tem primeru uporabljen postopek polnega vertikalnega particioniranja in vsebuje samo dva atributa (product\_id in price\_actual). V realnem svetu bi tabela vsebovala vsaj še nekaj atributov, kar bi koristilo boljši analizi podatkov, s tem pa bi se povečal tudi čas izvajanja poizvedbe.

Za drugi performančni test se bomo vprašali:

Koliko izdelkov je prodal določen prodajalec v točno določenem dnevu?

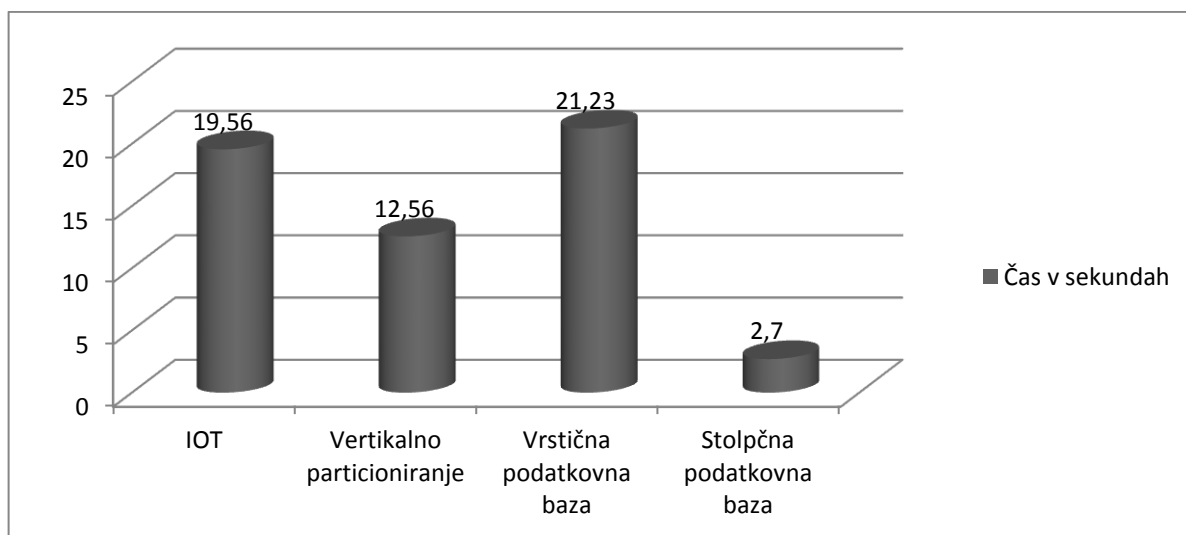
Poizvedba 2:

```
SELECT s.name, s.surname, COUNT(*) FROM orders o, seller s
WHERE o.seller_id = s.id
AND s.name = 'Carlo'
AND o.date_sold_id = 20130718
GROUP BY s.name, s.surname;
```

V tem primeru bo postopek vertikalnega particioniranja vseboval več atributov. Vertikalno bo particionirana samo tabela Orders. Vsebovala bo naslednje attribute: id, store\_name, store\_region, store\_address, seller\_id, product\_id, buyer\_id, date\_sold\_id, price\_actual, quantity, discount. Izdelano tabelo sem poimenoval Orders\_vert. Tabela bo uporabljena v

nadaljevanju praktičnega dela. Z izdelavo te tabele sem se najbolj približal izzivu iz realnega sveta, saj sem odstranil nekaj atributov, ki me v danem trenutku niso zanimali oz. so bili nepomembni.

Rezultati izvajanja poizvedbe 2 so vidni na sliki 9:



Slika 9: Čas izvajanja poizvedbe 2

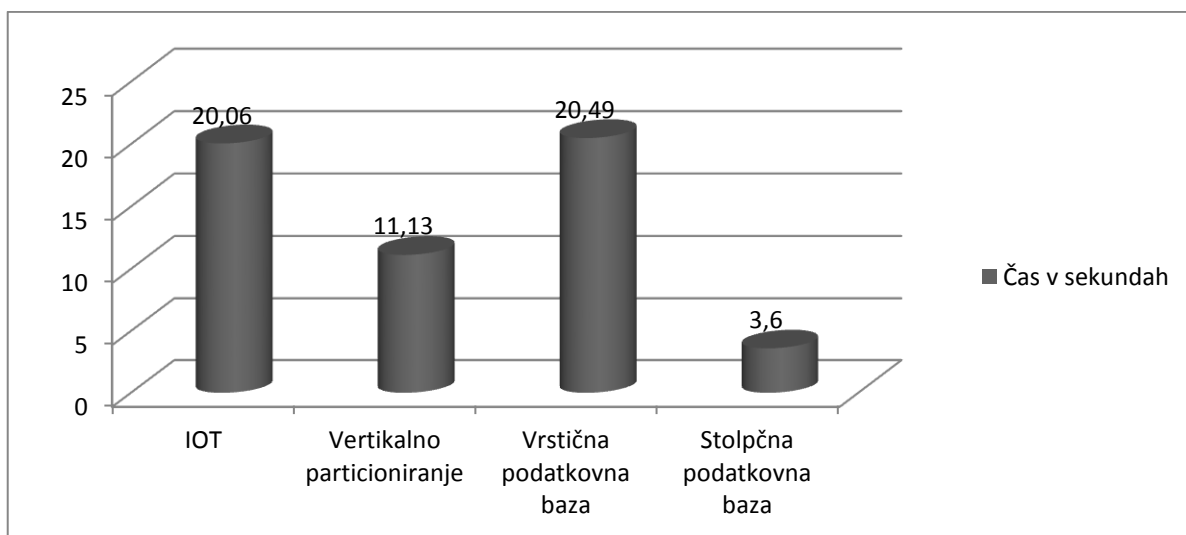
V zgornjem grafu se lepo vidi prednost stolpčne podatkovne baze. Izvajanje poizvedbe je v tem primeru celo 10x boljše od vrstične podatkovne baze. Način vertikalnega particioniranja vsebuje več atributov kot v prejšnjem testu, kar se pozna tudi na času izvajanja poizvedbe, še vedno pa so rezultati boljši kot v pri sami vrstični podatkovni bazi.

Za tretji performančni test bo prikazana povprečna prodaja izdelkov, ki so jih prodajalci prodali v mesecu juliju leta 2013.

Poizvedba 3:

```
SELECT p.name, s.name, AVG(o.quantity) FROM orders o, product p, seller s
  WHERE o.product_id = p.id
  AND s.id = o.seller_id
  AND o.date_sold_id between 20130701 and 20130731
GROUP BY p.name, s.name;
```

## Stolpčno usmerjen SUPB



Slika 10: Čas izvajanja poizvedbe 3

Rezultati poizvedbe 3 se nahajajo na sliki 10. Razlika je zopet očitna. Stolpčna podatkovna baza za izvedbo enake poizvedbe potrebuje tudi 10x manj časa od vrstične podatkovne baze. Pri večji količini podatkov bi to razmerje še naraslo.

Naslednji test bo narejen na manjši tabeli Buyer. Izbrani bodo vsi atributi, ki se nahajajo v tabeli. Ker tabela Buyer vsebuje le 32 zapisov, sem pričakoval, da se bo v tem primeru boljše obnesla vrstična podatkovna baza. Za potrebe testa sem naredil polno vertikalno particionirano tabelo Buyer. Vsak atribut bo skupaj s primarnim ključem tabele (primarni ključ tabele je celega številskega tipa, zato ga lahko uporabim kot ključ za povezovanje tabel) nastopal v svoji tabeli. Pričakoval sem slabše performance vertikalno particionirane tabele, saj ta vsebuje manjše število podatkov. Izdelava polno vertikalno particioniranih tabel vzame kar nekaj časa, zato je potrebno dobro premisliti, če izdelava teh tabel izboljša performance, hkrati pa je potrebno upoštevati potrebne spremembe poizvedb ter način vnosa in ažuriranja podatkov v tabelah.

Izdelava polno vertikalno particioniranih tabel:

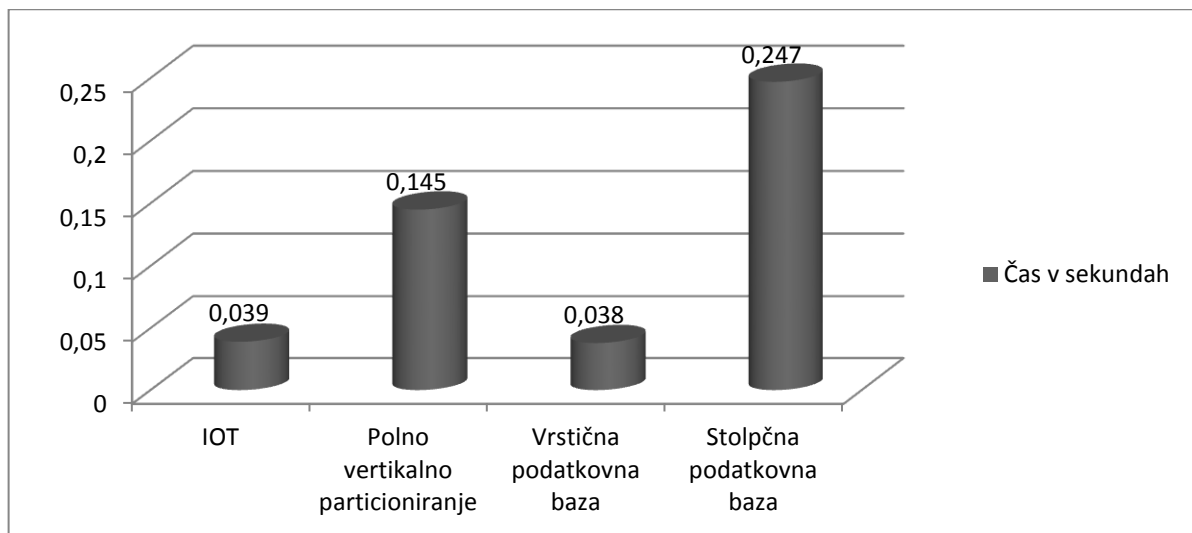
```
create table buyer_name as (select id, name from buyer);
create table buyer_surname as (select id, surname from buyer);
create table buyer_email as (select id, email from buyer);
create table buyer_gsm as (select id, gsm from buyer);
create table buyer_address as (select id, address from buyer);
create table buyer_city as (select id, city from buyer);
```

```
alter table buyer_name add primary key (id);
alter table buyer_surname add primary key (id);
```

```
alter table buyer_email add primary key (id);  
alter table buyer_gsm add primary key (id);  
alter table buyer_address add primary key (id);  
alter table buyer_city add primary key (id);
```

Poizvedba 4:

```
SELECT name, surname, email, gsm, address, city FROM Buyer;
```



Slika 11: Čas izvajanje poizvedbe 4

Slika 11 prikazuje rezultate poizvedbe 4. Rezultati so pričakovani. Najbolje se je obnesla vrstična podatkovna baza, ki je za pridobitev in prikaz podatkov potrebovala le 38ms. Način vertikalnega particioniranja tabele z majhnim številom zapisov se ne izplača. Ker so časi zelo majhni (ms), razlike skoraj niso opazne. Pri tem testiranju je potrebno upoštevati tudi prikaz podatkov v grafičnem vmesniku.

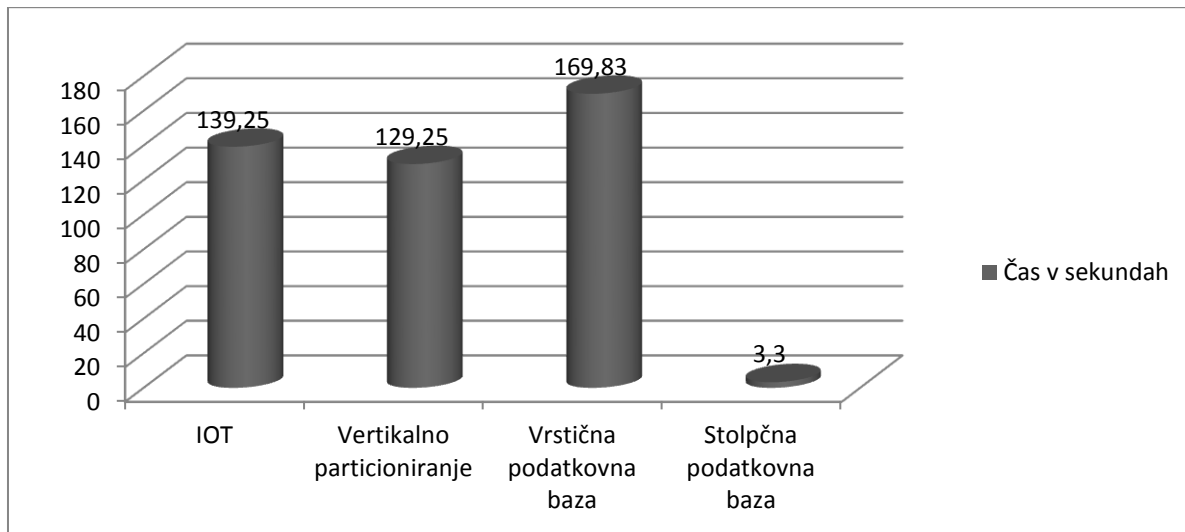
### 5.2.2 Ažuriranje podatkov

Poizvedba 5:

```
UPDATE orders  
SET store_region = 'North'  
WHERE store_address = 'Golden Coast';
```

## Stolpčno usmerjen SUPB

Rezultati izvajanja poizvedbe 5 se nahajajo na sliki 12.



Slika 12: Čas izvajanja poizvedbe 5

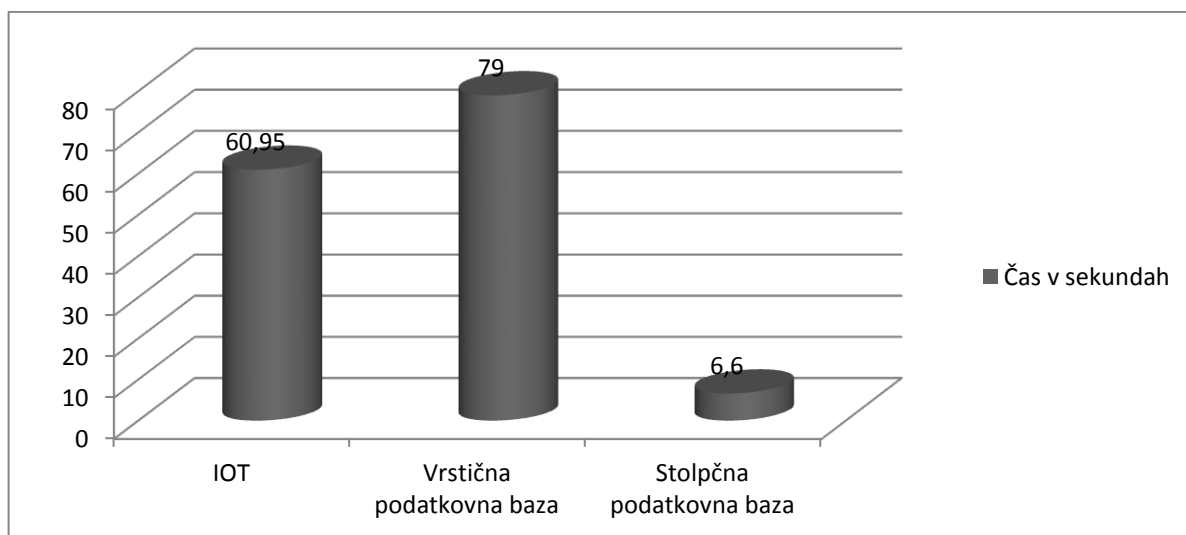
Gre za spremembo vrednosti za 566.312 zapisov. Ker gre za ažuriranje le enega stolpca, je ta operacija v stolpčni podatkovni bazi izjemno hitra.

Spodaj je prikazan graf ažuriranja večjega števila atributov.

Poizvedba 6:

```
UPDATE orders
SET store_region = 'North',
    date_sold_id = 20130101,
    ship_postal_code = 999,
    ship_city = 'Ljubljana',
    ship_region = 'Central',
    ship_date_id = '20130101',
    store_name = 'Sports 2000'
WHERE store_address = 'Golden Coast'
AND seller_id = 24
AND product_id = 24;
```

Ker vertikalno particionirana tabela, ki sem jo izdelal, ne vsebuje atributov ship\_postal\_code, ship\_city, ship\_region in ship\_date\_id, bo v tem testu izpuščena. Rezultati poizvedbe 6 se nahajajo na sliki 6.



Slika 13: Čas izvajanja poizvedbe 6

Z ažuriranjem podatkov se je spremenilo 231 zapisov. Vrstična podatkovna baza je pridobila nekaj sekund na času izvajanja zaradi količine spremenjenih zapisov, ki je precej manjši kot v prejšnjem primeru. V stolpčni podatkovni bazi je čas ažuriranja zapisov narasel. Razlog za to tiči v številu atributov. Ker se vsak atribut nahaja v svoji datoteki, je tako popravila več datotek.

### 5.2.3 Vstavljanje novih podatkov

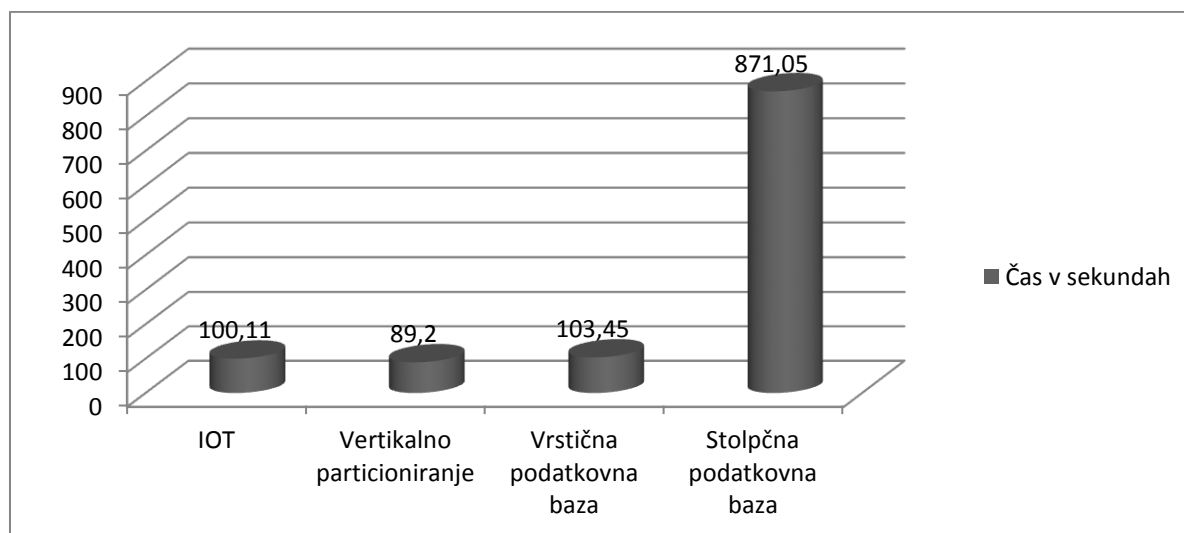
V vsako od tabel Orders sem vstavil 1.000 naključno generiranih podatkov. To sem naredil s pomočjo skripte, ki bo prebrala 1.000 zapisov iz datoteke (Insert stavkov).

Poizvedba 7. Primer enega izmed insert stavkov:

```
INSERT INTO orders VALUES (9297614, 'Well Done', 'North', 'Golden Coast', 15,  
29, 20, 20130927, 'Kamnik', 1001, 'Europe', 76.04, 6, 7, 20130927);
```

Slika 14 prikazuje čas izvajanja poizvedbe 7:

## Stolpčno usmerjen SUPB



Slika 14: Čas izvajanja poizvedbe 7

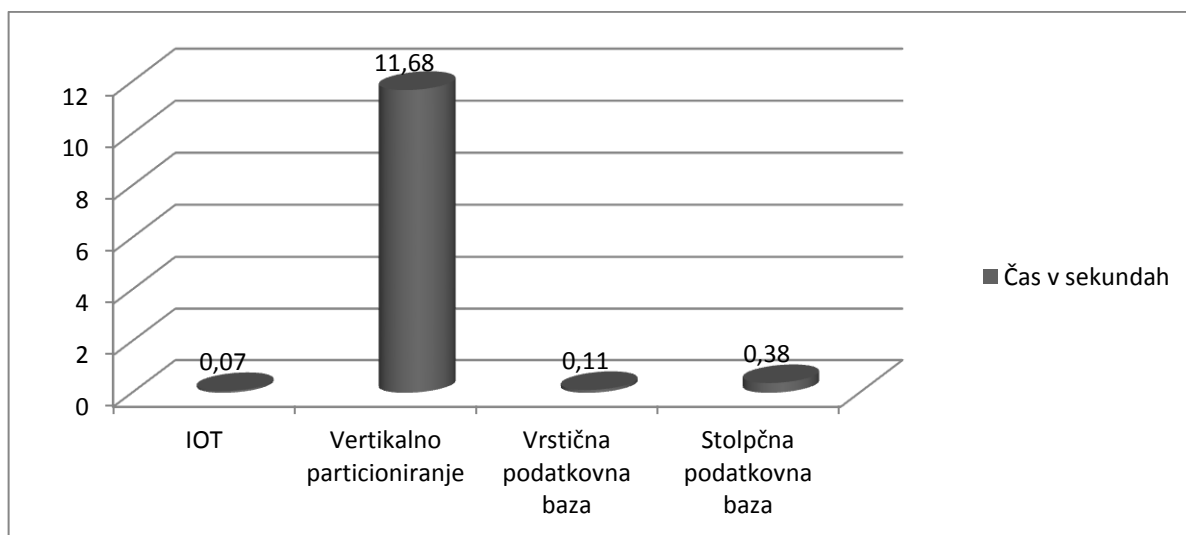
V stolpčni podatkovni bazi čas za izvajanje skripte krepko naraste proti ostalim. Razlog za to tiči v posodabljanju podatkov. MonetDB ima privzeto nastavljen "Auto Commit". To pomeni, da se za vsak na novo vstavljen zapis zgodi sprememba tudi na podatkovnem disku. Tipični podatkovni disk s 7.200 obrati na minuto lahko procesira med 75 in 100 bralno/pisalnih operacij na sekundo. Vstavljanje novega zapisa je ena bralno/pisalna operacija, kar povzroči ozko grlo. MonetDB se omenjenemu ozkemu grlu izogne na način, da vse spremembe naredi v enem dostopu do podatkovnega diska oz. eni transakciji. To se naredi na način, da se stavke obda z ukazom "START TRANSACTION".

### 5.2.4 Brisanje podatkov

Naključno bo izbranih 1.000 podatkov.

Poizvedba 8:

```
DELETE FROM orders WHERE id BETWEEN 80500 AND 81500;
```



Slika 15: Čas izvajanja poizvedbe 8

Na sliki 15 je prikazan čas izvajanja poizvedbe 8. V zgornjem primeru je na atributu `id` narejen indeks, saj je to tudi primarni ključ tabele. Ker ga vertikalno particionirana tabela ne vsebuje, je čas izvajanja poizvedbe daljši. V primeru, da se v vertikalno particionirano tabelo doda indeks:

```
CREATE INDEX orders_vert_index ON orders_vert (id),
```

je čas izvajanja poizvedbe podoben ostalim – 0,42ms.

Razlike so minimalne. Seveda je potrebno upoštevati, da je bilo izbrisanih zgolj 1.000 zapisov.



### Zaključek

V diplomskem delu sem predstavil stolpčno podatkovno bazo in jo primerjal z vrstično podatkovno bazo. Na praktičnem primeru sem analiziral primerjavo performanc različnih implementacij stolpčne podatkovne baze v vrstičnem sistemu za upravljanje podatkovnih baz, vrstično podatkovno bazo in stolpčno podatkovno bazo. Kljub temu da obstaja več načinov implementacije stolpčne podatkovne baze v vrstičnem sistemu za upravljanje podatkovnih baz, so performance še vedno slabše kot v stolpčni podatkovni bazi. Izdelava posameznih vertikalno particioniranih tabel sicer pohitri izvajanje določenih poizvedb, s stališča vzdrževalcev podatkovne baze pa je zamudna, saj morajo ti spremeniti model in poizvedovalne stavke za pravilen vnos podatkov. Stolpčna podatkovna baza je primerna za analitične poizvedbe v realnem času. Razlike med performancami se še izdatno povečajo z večjo količino podatkov. Slabost, ki sem jo opazil v stolpčni podatkovni bazi, je polnjenje. Pri polnjenju se morajo napolniti vsi stolpci, kar je pri veliki količini podatkov dolgotrajen postopek. Temu problemu se enostavno izognemo s paketnim polnjenjem podatkovnega skladišča, ki se izvede ponoči.

Dejstvo je, da so stolpčne podatkovne baze prisotne že od samega začetka. Razlog, da se danes govori več o njih, je potreba po hitri analizi podatkov. Z večanjem količine shranjenih podatkov se večajo tudi časi izvajanja določenih poizvedb. Ker se število podatkov vsako leto izdatno poveča, lahko v prihodnosti pričakujemo porast uporabe stolpčnih podatkovnih baz v podatkovnih skladiščih.



## Literatura

- [1] Infobright. (2011) User's Guide to the Emerging Database Landscape: Row vs. Columnar vs NoSQL. Dostopno na:  
<http://clu.timheckman.net/~theckman/EmergingDatabaseLandscapeJuly2011.pdf>
- [2] (2013) Opredelitev termina podatkovna baza. Dostopno na: [http://colos1.fri.uni-lj.si/ERI/RACUNALNISTVO/PODATKOVNE\\_BAZE/opredelitev\\_termina\\_podakovna\\_baza.html](http://colos1.fri.uni-lj.si/ERI/RACUNALNISTVO/PODATKOVNE_BAZE/opredelitev_termina_podakovna_baza.html)
- [3] prof. Dr. Marko Bajec. (2013) Uvod v podatkovne baze. Dostopno na:  
[http://bajecm.fri.uni-lj.si/downloads/Uvod%20v%20PB\\_v04.pptx](http://bajecm.fri.uni-lj.si/downloads/Uvod%20v%20PB_v04.pptx)
- [4] (2013) Arhitektura PB. Dostopno na: [http://colos.fri.uni-lj.si/eri/RACUNALNISTVO/PODATKOVNE\\_BAZE/arhitektura\\_pb.html](http://colos.fri.uni-lj.si/eri/RACUNALNISTVO/PODATKOVNE_BAZE/arhitektura_pb.html)
- [5] prof. Dr. Marko Bajec. (2013) Relacijski podatkovni model. Dostopno na:  
[http://bajecm.fri.uni-lj.si/downloads/Relacijski%20podatkovni%20model\\_v03.pptx](http://bajecm.fri.uni-lj.si/downloads/Relacijski%20podatkovni%20model_v03.pptx)
- [6] (2013) Jezik SQL. Dostopno na: <http://en.wikipedia.org/wiki/SQL>
- [7] Gheorghe Matei, "Column-Oriented Databases, an Alternative for Analytical Environment," Database Systems Journal, vol. I, Feb. 2010.
- [8] Peter A. Boncz, Stavros Harizopoulos Daniel J. Abadi. Column-Oriented Database Systems. Dostopno na: <http://www.vldb.org/pvldb/2/vldb09-tutorial6.pdf>
- [9] Simranjit Kaur Dhindsa Punam Bajaj, "A Vision towards Column-Oriented Databases," vol. I, no. 4, June 2012.
- [10] (2011) Column-Oriented DBMS. Dostopno na: [http://en.wikipedia.org/wiki/Column-oriented\\_DBMS](http://en.wikipedia.org/wiki/Column-oriented_DBMS)
- [11] Samuel R. Madden, Nabil Hachem Daniel J. Abadi. (2013) Column-Stores vs. Row Stores: How Different Are They Really? Dostopno na:  
<http://db.csail.mit.edu/projects/cstore/abadi-sigmod08.pdf>
- [12] (2013) Managing Index-Organized Tables. Dostopno na:  
[http://docs.oracle.com/cd/B28359\\_01/server.111/b28310/tables012.htm#i1007016](http://docs.oracle.com/cd/B28359_01/server.111/b28310/tables012.htm#i1007016)
- [13] William McKnight. (2011) Columnar Databases. NoSQL konferenca. Dostopno na:  
<http://www.slideshare.net/Dataversity/columnar-databases-william-mcknight-mcknight-consulting-group>
- [14] (2013) Data compression. Dostopno na:  
<http://www.monetdb.org/Documentation/Guide/Compression>
- [15] (2013) Columnar compression vs. column storage. Dostopno na:

<http://www.dbms2.com/2011/02/06/columnar-compression-database-storage/>

- [16] SQL Server Architects. (2013) Columnar Database and Row oriented Database -SQL Server 2012. Dostopno na: <http://www.sqlserverspecialists.com/2013/04/columnar-database-and-row-oriented.html>
- [17] (2013) Column store features. Dostopno na: <http://www.monetdb.org/Home/Features>
- [18] (2013) Sybase IQ. Dostopno na: [http://en.wikipedia.org/wiki/Sybase\\_IQ](http://en.wikipedia.org/wiki/Sybase_IQ)
- [19] Dale Andrerson. (2013) Column Oriented Database Technologies. Dostopno na: <http://www.dbbest.com/blog/column-oriented-database-technologies/>
- [20] (2013) Data Blocks, Extents and Segments. Dostopno na: [http://docs.oracle.com/cd/B10500\\_01/server.920/a96524/c03block.htm](http://docs.oracle.com/cd/B10500_01/server.920/a96524/c03block.htm)
- [21] Jan Steemann. (2012) Column-oriented databases. Dostopno na: <http://www.slideshare.net/arangodb/introduction-to-column-oriented-databases-contiguously>
- [22] June Tong. (2012) Demystifying Columnar Databases. Dostopno na: <http://www.slideshare.net/arangodb/introduction-to-column-oriented-databases-contiguously>