UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Peter Pavkovič

Primerjava časovne zahtevnosti zahtev SQL pri podatkovni bazi IBM DB2 na operacijskem sistemu Z/OS

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: prof. dr. Saša Divjak

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00094/2011
Datum: 04.04.2011

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

Kandidat: **PETER PAVKOVIČ**

Naslov: **PRIMERJAVA ČASOVNE ZAHTEVNOSTI ZAHTEV SQL PRI PODATKOVNI BAZI IBM DB2 NA OPERACIJSKEM SISTEMU Z/OS**

COMPARISON OF TIME COMPLEXITY OF SQL QUERRIES BY IBM DB2 DATABASE ON Z/OS OPERATING SYSTEM

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Naloga naj obravnava problematiko, ki nastopi pri obdelavi velikih količin podatkov v okviru sistemov za upravljanje s podatkovnimi bazami na velikih računalnikih. Kot poligon uporabite operacijski sistem IBM z/OS in podatkovno bazo DB2. Za izvedbo ustreznih meritev razvijte namizno aplikacijo v programskih jezikih Java oziroma C ter SQL. Hitrost izvajanja povpraševanj poizkusite pospešiti s primernimi procedurami. Rezultate meritev primerjajte glede na porabljen računalniški čas, zasedenost pomnilnika in glede na čas, potreben za izvedbo zahtev SQL na danem operacijskem sistemu ob izbranem programskem jeziku in dani tehnologiji.

Mentor:

prof. dr. Saša Divjak

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Peter Pavkvič,

z vpisno številko 63050335,

sem avtor/-ica diplomskega dela z naslovom:

Primerjava časovne zahtevnosti zahtev SQL pri podatkovni bazi IBM DB2 na operacijskem sistemu z/OS

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

prof. Dr. Saša Divjak,

in somentorstvom (naziv, ime in priimek)

- 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 10.05.2011 Podpis avtorja/-ice:

Zahvala

Zahvaljujem se mentorju na Fakulteti za računalništvo in informatiko v Ljubljani prof. Dr. Saši Divjaku za pomoč in podporo pri izdelavi diplomske naloge.

Posebno bi se rad zahvalil mentorju v podjetju IBM Slovenija gospodu Sašu Preku (zSeries Software Sales Specialist & Member of zChampion Team), za ves čas, trud, pomoč in podporo pri izdelavi diplomske naloge ter odlično mentorstvo v podjetju.

Zahvaljujem se staršem, bratu Urošu, punci Kaji ter ostalim, ki so me ves čas študija podpirali, ter skupaj z mano prehodili pot mojega študija.

Kazalo vsebine

Kazalo slik

Kazalo tabel

Kazalo grafov

Terminologija

- Auxillary table – vsaka kolona tipa LOB potrebuje sebi pripadajočo auxillary tabelo
- BLOB – podatkovni tip (angl. Bineary large object)
- Boolean – podatkovni tip, ki shranjuje logično vrednost
- Bufferpool – pogosto iskani podatki se shranijo v bufferpool in so tako še hitreje dosegljivi
- Char – podatkovni tip, ki shranjuje znake
- CPU – centralna procesna enota (angl. Central process unit)
- DB2 – IBM podatkovna baza
- DDL – podatkovna struktura, iz katere DB2 zgradi podatkovno bazo
- Float – podatkovni tip, ki shranjuje decimalna števila
- IBM – international business machine
- Index – z določanjem indexov je dostop do podatkov hitrejši
- Int – podatkovni tip, ki shranjuje cela števila
- JDBC – knjižnica v Javi za dostop do podatkovne zbirke
- Mainframe- centralni računalnik
- MIPS – miljon zahtev na sekundo (angl. Million instruction per second)
- SQL – programski jezik za delo z podatkovnimi bazami (angl. Structured query language)
- Stored procedure – podatkovna procedura, ki se izvaja na strežniku
- Table space – določa fizično lokacijo shranjenih podatkov na trdem disku
- Varchar – podatkovni tip, ki shranjuje besedilo
- XML – razširljiv označevalni jezik
- zAAP – aplikacijski procesor v mainframe sistemu (angl. Application assist processor)
- ZIIP – informacijski procesor v mainframe sistemu(angl. Integrated information processor)
- Z-series – moderno ime za IBM mainframe sisteme

Terminology

- Auxillary table – every LOB table need own auxillary table,
- BLOB – data type (Bineary large object),
- Boolean – data type which store logical value,
- Bufferpool – frequently sought data stored in bufferpool,
- Char – data type which store character,
- CPU – Central process unit,
- DB2 – IBM database,
- DDL – data structure, from which DB2 build database,
- Float – data type which store decimal number,
- IBM – international business machine,
- Index – index make faster access to data,
- Int – data type which store number,
- JDBC – Java library for database access,
- Mainframe- central computer,
- MIPS – Million instruction per second,
- SQL – Structured query language,
- Stored procedure – data procedures which running on server,
- Table space – physical location where we sotre data ,
- Varchar – data type which store string,
- XML – Extensible Markup Language,
- zAAP – Application assist processor,
- ZIIP – Integrated information processor,
- Z-series – mainframe system.

Povzetek

V okviru diplomske naloge sem s pomočjo namenskega informacijskega sistema pokazal, kako se v informacijski dobi spopadamo z veliko količino podatkov, pomen dostopnega časa do želenega podatka ter pomen razvoja strojne in programske opreme glede na rast shranjenih podatkov.

Informacijski sistem je v današnjem svetu srce vsakega podjetja. Za izvedbo diplomske naloge sem uporabil informacijski sistem, ki temelji na IBM strojni ter programski opremi. Zavedati se moramo, da je zanesljiv ter hiter operacijski sistem še kako pomemben za uspešno delovanje podjetja.

V diplomski nalogi sem predstavil zgradbo operacijskega sistema, pomen uporabe namenskih produktov glede na namen ter količino podatkov. Seznanil sem se z orodji in tehnologijami, ki sem jih uporabil. Za izvedbo meritev sem uporabil profesionalno strojno ter programsko opremo podjetja IBM ter programski jezik C in Java s katerima sem se spoznal že v času študija. Uporabljen informacijski sistem temelji na IBM Mainframe (Z-serija) sistemu ter podatkovni bazi DB2. Za razvoj namizne aplikacije sem uporabil programska jezika Java in SQL.

Glede na to, da gre za tehnologije, ki so v veliki meri nepoznane, sem na uvodnih straneh opisal pomen ter razvoj posamezne tehnologije. Za izvajanje meritev ter primerjavo sem uporabil namenski informacijski sistem z zelo dobrimi zmogljivostmi. Na hitrost izvajanja aplikacije v veliki meri vpliva programski jezik v katerem je razvita, zato sem v ta namen uporabil programski jezik Java ter programski jezik C. Hitrost izvajanja zahtev SQL (angl. SQL Query) v programskem jeziku Java sem optimiziral s shranjenimi procedurami (angl. Stored Procedure) ter uporabo SQLJ tehnologije.

V podatkovni bazi za moj primer hranim naslednje tipe podatkov:

- celo število (int),
- besedilo (Varchar),
- BLOB (slike in video datoteke),
- Znak (char),
- XML,
- Decimalno število (float),
- boolean

Za konec sledi primerjava podatkov, grafični prikaz porabljenega CPU časa, zasedenosti pomnilnika, časa izvajanja določenih zahtev SQL glede na uporabljen informacijski sistem, programski jezik in tehnologijo.

Abstract

For my thesis I used special information system to show means of elapse time to selected data and the importance of hardware and software on data growth.

Today information system is the core of every company. In my work I used information system that is based on IBM hardware and software. It is important to know that a reliable and quick operation system is most important for achieving successful results of the company

In my thesis I have presented the architecture of the operation system, the use of products by the means and by the quantity of used data. I have learned about tools and technologies used in my work. I have used professional hardware and software equipment from IBM for the measurements made in my thesis as well as programming language C and Java I have know during the time of my study. Information system I have used is based on IBM Mainframe Z-series system and on database DB2.

Considering these are technologies that are now very well known I have described the use and development of each specific technology. In my measurements and comparisons I have used information system with great performances. The type of programming language used for developing applications has a great influence on the speed of implementation of the application. For the development of the desktop application I have used programming language Java and SQL. I have optimized the speed of implementing SQL query in programming language Java with stored procedures and by using SQLJ technology.

In the database of my work I have stored following information:

- integer
- varchar
- BLOB (binary large object)
- character
- XML
- decimal number
- boolean

In the end I have showed comparison of the data, graph of the used CPU time, memory usage, time used for executing SQL query by the information system used, programming language and technology.

1 UVOD

V poplavi podatkov se dnevno srečujemo s težavo, kako shranjevati podatke, da bodo ob morebitni uporabi le ti kar se da hitro na voljo. Včasih imam občutek, da podatke shranjujemo ne glede na to ali jih bomo še kdaj rabili ali ne. Vendar podatek tudi če se nikoli ne uporabi obstaja in zaseda prostor na disku ter predstavlja dodaten korak pri iskanju želenega podatka. Da pa naše iskanje podatka v podatkovni bazi ne preseže razumnega časa iskanja potrebujemo namensko računalniško strojno in programsko opremo, ki skupaj tvorijo informacijski sistem. Eden izmed takih informacijskih sistemov je IBM System Z z operacijskim sistemom z/OS ter podatkovno bazo IBM DB2. System Z je bil v začetku razvit kot sistem za upravljanje z veliki količinami podatkov ter izvajanje transakcij in je v tem segmentu ohranil svoj primat vse do danes.

1.1 Namen diplomske naloge

- predstaviti pomen informacijskega sistema
- spoznati ter preizkusiti sodobne tehnologije in programske rešitve za upravljanje s podatki
- izdelati javansko namizno aplikacijo, ki bo za svoje delovanje uporabljala zgolj podatke iz podatkovne baze
- izdelati primerjavo opazovanih parametrov na dveh zmogljivostno različnih informacijskih sistemih z izvajanjem enakih testov in uporabo enake količine podatkov.

1.2 Cilji diplomske naloge

- spoznati strojno opremo IBM Z-series
- spoznati način upravljanja z operacijskim sistem Z/os in podatkovno bazo DB2
- s pomočjo namenskih orodij samostojno izdelati podatkovni model ter ga uporabiti
- ugotoviti katere tehnologije so hitrejše pri upravljanju s podatki
- izdelati namizno aplikacijo s katero sem prikazal dejansko funkcionalnost zanesljivega informacijskega sistema.

2 Tehnologije in orodja

V naslednjem poglavju bomo na kratko predstavili tehnologije in orodja, ki smo jih uporabili pri pripravi testnega okolja in izvajanju testov.

2.1 Strežnik – Mainframe

so veliki računalniški sistemi. Prvi stroj je bil izdelan leta 1952, nosil je oznako IBM 700/7000 series. Največjo slavo je doživljal v letih med 1960 in 1970, ko je bil sinonim podjetja IBM. V začetku so se mainframe sistemi prodajali brez operacijskega sistema, tako so si morali uporabniki razviti lastne programske rešitve. Mainframe je v osnovi omogočal zgolj sprotno vnašanje ukazov. Kasneje je IBM v sistem vključil prevajalnike za programski jezik COBOL in Fortran, kar je pomenilo začetek razvoja operacijskih sistemov ter drugih programskih rešitev. Prvi operacijski sistem imenovan GM-NAA I/O za IBM 704 sta leta 1965 razvila Robert L. Patric iz podjetja General Motors in Owen Mock iz North American Aviation. Prvi operacijski sistem je deloval na podlagi obdelave vrste opravil. Mainframe je bil osnova za kasnejši razvoj IBM S/360 sistem, ki je že imel lasten operacijski sistem ter podporo različnim aplikacijam.

2.1.1 Tehnične karakteristike uporabljenih strežnikov:

- število procesorjev: 5
- hitrost procesorja: 2,4GHz
- število zahtev/sek: 160 MIPS
- ram: 16 GB

Slika 1: IBM System Z strežnik.

2.2 Odjemalec – PC

2.2.1 Vrste odjemalcev:

- **tanki odjemalci:** popolnoma odvisni od delovanja strežnika, saj za celotno delovanje poslovne aplikacije skrbi strežnik. Odjemalec služi zgolj kot prikazovalnik podatkov, ki nam jih posreduje strežnik. V tem primeru so sistemske zahteve odjemalca minimalne.

- **debeli odjemalci:** pri tej arhitekturi se večina poslovne aplikacije izvaja na odjemalcu. Strežnik služi le za posredovanje informacij na zahtevo, obdelavo podatkov, shranjevanje v bazo, izvajanje shranjenih procedur itd. Sistemske zahteve odjemalca so pogojene z aplikacijami, ki se izvajajo.

2.2.2 Tehnične karakteristike odjemalca:

- **tip odjemalca:** debeli odjemalec
- **procesor:** Intel core2 Duo CPU T7300 @2.0 GHz
- **velikost rama:** 2GB

Slika 2: Osebni računalnik – odjemalec.

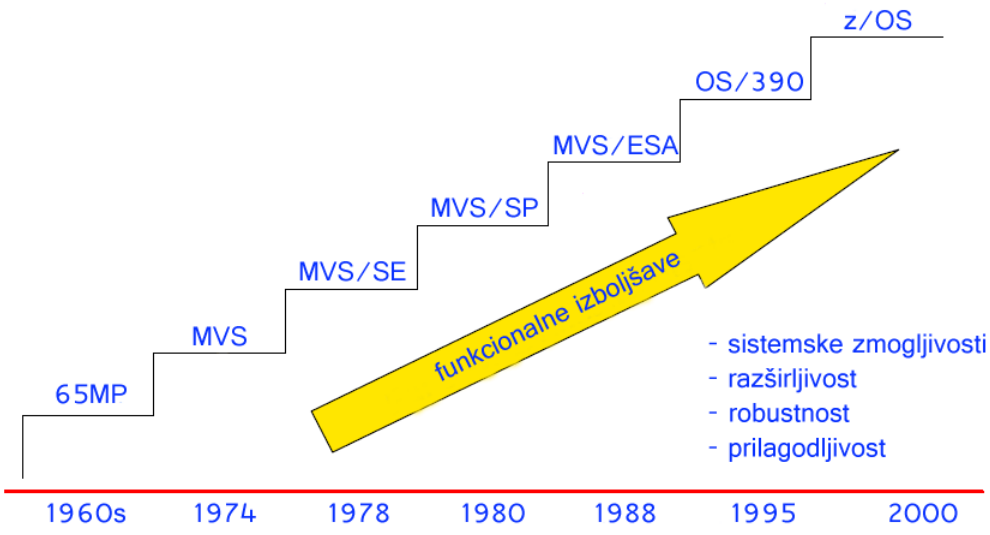
2.3 Operacijski sistem

2.3.1 Strežnik (z/OS)

V šestdesetih letih je IBM predstavil družino računalnikov System/360. Gre za prvo družino računalnikov, kjer je bila postavljena jasna meja med arhitekturo in izdelkom, kar pomeni, da so vsi računalniki, ki so temeljili na arhitekturi 360, imeli enak nabor strojnih ukazov. OS/360 je bil operacijski sistem, ki se ga je dalo namestiti na vse mainframe sisteme. Zaradi uporabe enakega operacijskega sistema so bili programi med posameznimi računalniki povsem prenosljivi. Pri razvoju OS/360 je imel IBM nemalo težav, saj so bili posamezni modeli računalnikov iz družine 360 med sabo zelo različni. System/360 je uvedel kopico standardov, ki se v računalništvu uporabljajo še danes: osem bitov dolg bajt, 32-bitne besede, naslavljanje pomnilnika po bajtih (namesto po besedah) in disketno enoto. V operacijskem sistemu z/OS, ki teče na IBMovih strežnikih zSeries, lahko izvajamo vse programe, pisane za System/360. Z/OS je 64 bitni operacijski sistem. Z/OS podpira izvajanje vseh aplikacij, ki so bile napisane za uporabo na predhodnih operacijksih sistemih.

2.3.1.1 Uporabljeni operacijski sistemi:

- **ime:** z/OS,
- **izdaja:** 1.7.



Slika 3: Razvoj operacijskega sistema z/OS.

2.3.2 Odjemalec (Microsoft Windows XP)

Windows XP je serija operacijskih sistemov Microsoft Windows iz družine Windows NT, namenjenih uporabi na osebnih računalnikih v domačih in poslovnih okoljih. Izšel je 24. avgusta 2001 kot naslednik Microsoftovih operacijskih sistemov Windows 2000 in Windows Me. »XP« v imenu je okrajšava za eXPerience (slovensko izkušnja, doživetje), razvijali pa so jih pod kodnim imenom Whistler.

2.3.2.1 Uporabljen operacijski sistem:

- **ime:** Windows XP Professional
- **izdaja:** Version 2002
- **varnostni paket:** SP3

2.4 Podatkovna baza (DB2):

DB2 ima dolgo zgodovino in je po mnenju mnogih prva SQL podatkovna zbirka. Uradno je prva verzija DB2 izšla leta 1983, ko so preimenovali produkt Database Management System (DBMS). V začetku je bil produkt namenjen uporabi na mainframe MVS platformi. Predhodnik prvega DB2 produkta je bil SQL/DS, ki ga je bilo moč izvajati zgolj na mainframe VM platformi.

Prvi produkt, ki pa ni imel kaj veliko skupnega z DB2, se je imenoval System relation ali System R. Ta je bil raziskovalni prototip leta 1970, za katerega je E.F.Codd izdelal teorijo relacijskih baz. Junija 1970 je bil objavljen Coddsov model upravljanja s podatki. Za upravljanje Coddsovega podatkovnega modela se uporablja jezik Aplha. IBM takrat ni zaupal v ta model zato je razvoj produkta zaupal skupini programerjev, ki ni bila pod Coddsovim vplivom. Nastal je „Structured English Query Language“ - SEQUEL. Ker pa je IBM pred prvo uradno izdajo želel poleg tehničnih kvalitet doseči tudi dobre komercialne lastnosti so SEQUEL prenovili in preimenovali, tako je nastal povsem nov jezik imenovan System Querry Language – SQL.

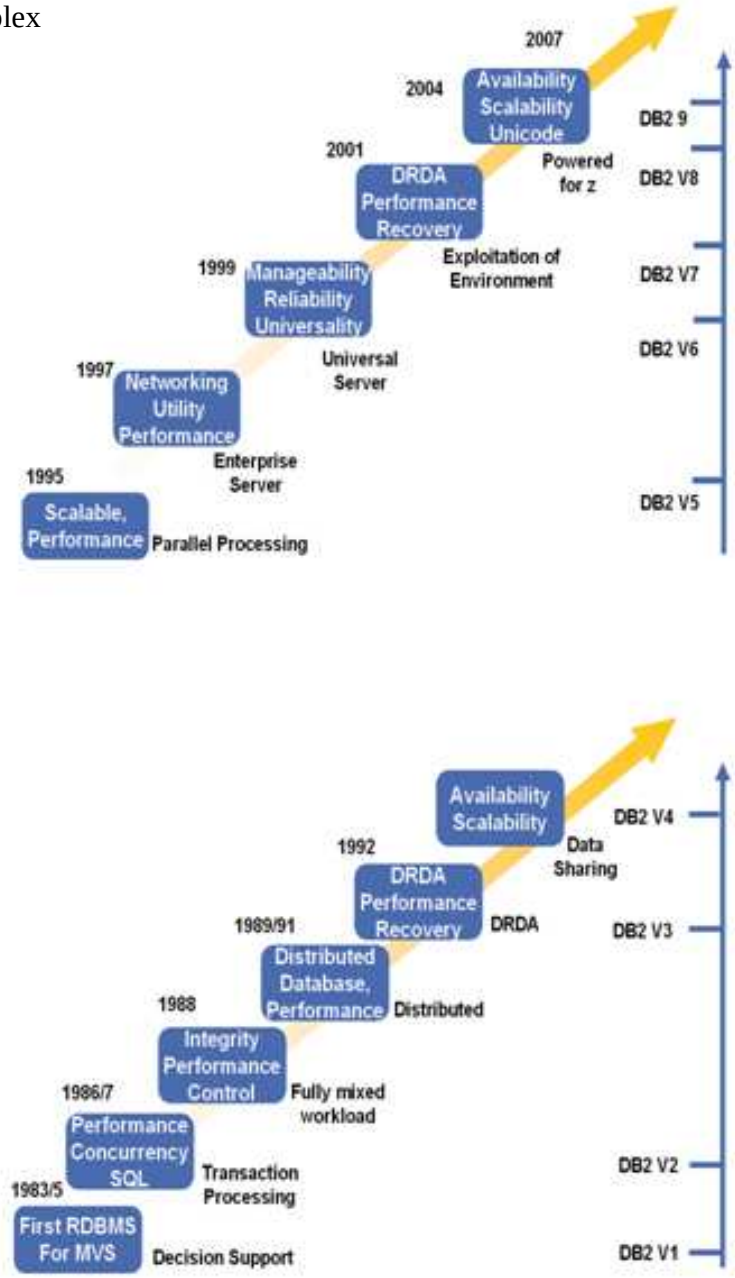
Leta 1990 je IBM sprejel odločitev, da DB2 ponudi na distribuiranem okolju OS/2, UNIX in Windows. Koda iz katere je nastala distribuirana verzija DB2 je bila del „Data Management“ produkta. IBM je Data Management izboljšal in mu dodal nekja zelo uporabnih distribuiranih lastnosti, med katerimi je v tistem času najbolj izstopal oddaljen dostop do podatkovne baze znotraj omrežja. Zaradi nejasnosti in kompliciranosti izvorne kode je IBM celoten produkt preprogramiral. Nova verzija Database Managerja je dobila ime DB2. Zaradi identičnega poimenovanja z ostalimi verzijami je prišlo do težav z razpoznavnostjo posameznih distribucij. Rešitev problema je bilo sprememba poimenovanja in sicer so poleg oznake DB2 dodali še oznako sistema npr. DB2/2 za OD/2. Kasneje je IBM sprejel nov standard poimenovanja, ki velja še danes. Poleg imena produkta je potrebno navesti tudi ime platforme kateri je posamezna distribucija namenjena npr. DB2 for z/OS. Ne glede na enak namen uporabe sta bili distribuciji zelo različne, saj je bila distribucija za uporabo na mainframe sistemu razvita v programskem jeziku PL/S, distribucija za distribuirano okolje pa v programskem jeziku C++. Obe pa sta imeli zelo podobno arhitekturo SQL optimizacije imenovano „The Starburst Optimizer“. V letih vzpona so kvalitete DB2 dokazovali s pomočjo zmogljive strojne opreme, ki je omogočala Parallel Sysplex data Sharing.

Z DB2 8.0 for z/OS je prišla zahteva po uporabi 64-bitnega naslavljanja zato so stranke, ki želijo migrirati na novejše verzije ter uporabljati nove funkcionalnosti primorane zamenjati strojno opremo. V letu 2006 je IBM najavil „VIPER“ kodno ime za DB2 v.9 za vse platforme vključno z z/OS. To je bila prva podatkovna baza, ki je omogočala shranjevanja XML datotek.

Najboljše zmogljivostne in varnostne rezultate dosežemo na sistemu System Z z operacijskim sistemom z/OS.

2.4.1 Prednosti DB2 na System Z in z/OS pred ostalimi sistemi:

- integriran namenski informacijski procesor ZIIP
- integriran namenski javanski procesor ZAAP
- mednarodno potrjen najvarnejši sistem
- možnost uporabe do 54 procesorjev
- 64 bitni navidezni pomnilnik
- parallel sysplex
- IP v6
- SSL



Slika 4: Razvoj podatkovne baze IBM DB2.

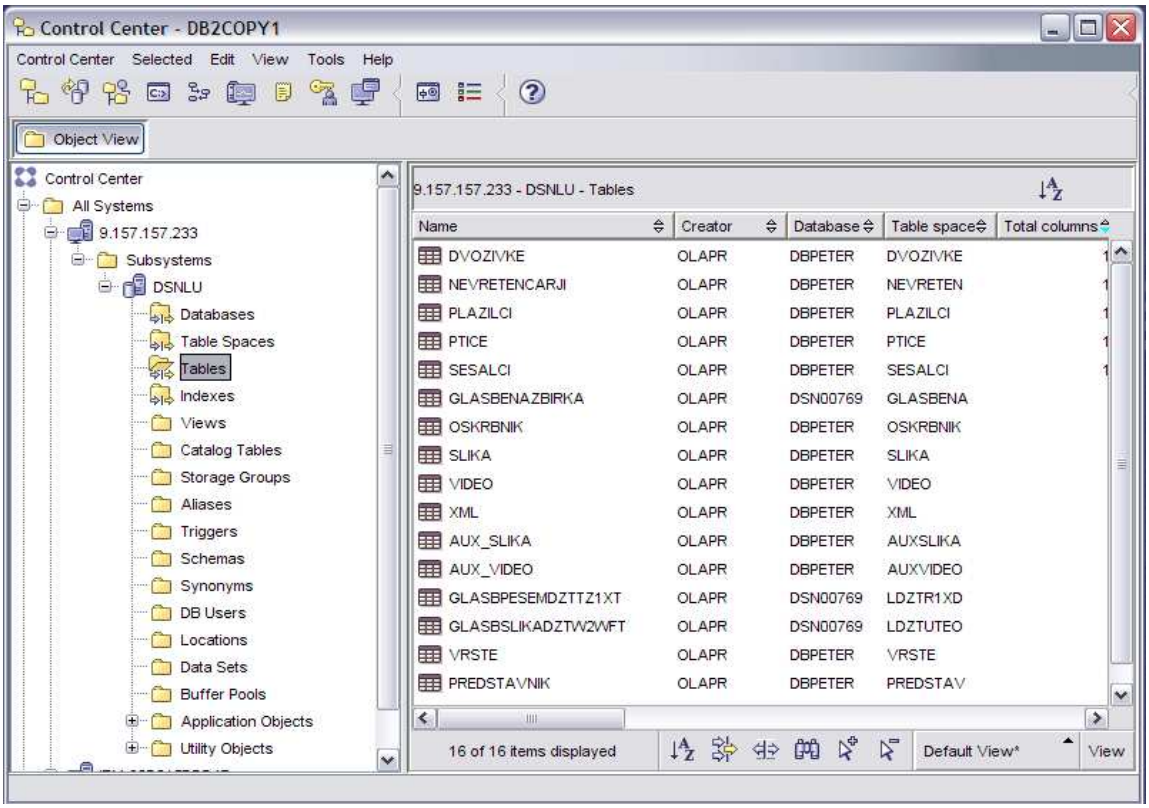
2.5 Programska orodja

2.5.1 IBM DB2 Control Center

IBM DB2 Control Center je orodje, ki je napisano v programskem jeziku Java in je na voljo kot samostojna aplikacija. Control center je aplikacija kamenjana upravljanju podatkov podatkovne baze DB2 na operacijskem sistemu OS/390 in Z/OS. Enostavnejše upravljanje s podatki nam omogoča uporabniku prijazen grafični vmesnik.

Control center nam omogoča:

- dodajanje UDB primerkov (angl. Instance) in podatkovnih baz ter njihovih objektov.
- Upravljanje s podatkovnimi bazami in njihovimi objekti: ustvarjamo, odstranjujemo ter urejamo podatkovne baze, tabele, poglede, indekse, prožilce in sheme.
- Upravljanje z uporabniki DB2 podatkovnega strežnika in njihovimi pravicami.
- Upravljanje s podatki: nalagamo, izvažamo, uvažamo, reorganiziramo ter na podlagi obstoječih podatkov izdelamo statistiko.
- Izvajanje preventivnih vzdrževalnih del za obnovitev podatkov v tabelah.
- Upravljanje s povezavami na podatkovno bazo.
- Izvajanje shranjevalne procedure (angl. Stored procedure)



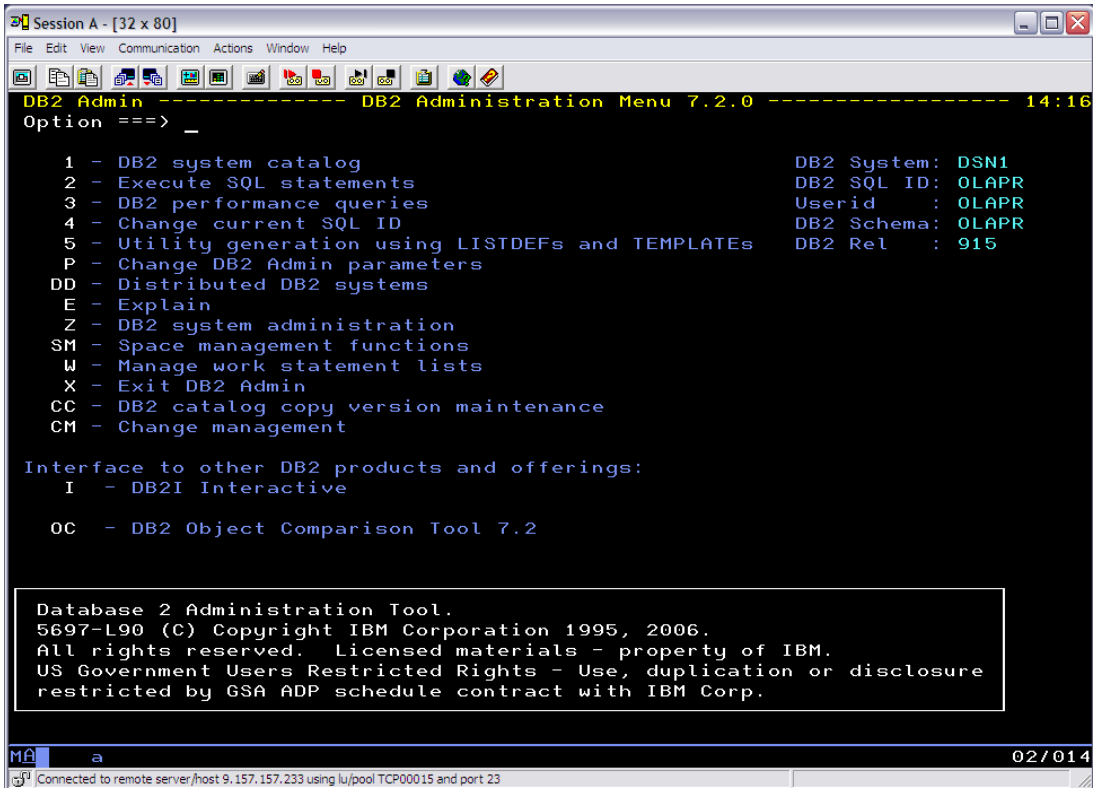
Slika 5: IBM Control Center.

2.5.2 IBM DB2 Administration tool 7.2 for Z/OS

Aplikacija je namenjena uporabi na operacijskem sistemu Z/OS in je vez med podatkovno bazo DB2 in uporabnikom.

Administration tool nam omogoča:

- hitro in enostavno upravljanje s podatki
- pisanje in izvajanje dinamičnih SQL stavkov
- pomoč pri ustvarjanju novih podatkovnih baz in tabel
- neposredno izvajanje ukazov ustvari, migriraj, izbriši, revezno inženirstvo nad objekti podatkovne baze
- ustvarjanje in izvajanje novih nalog (angl. JOB)



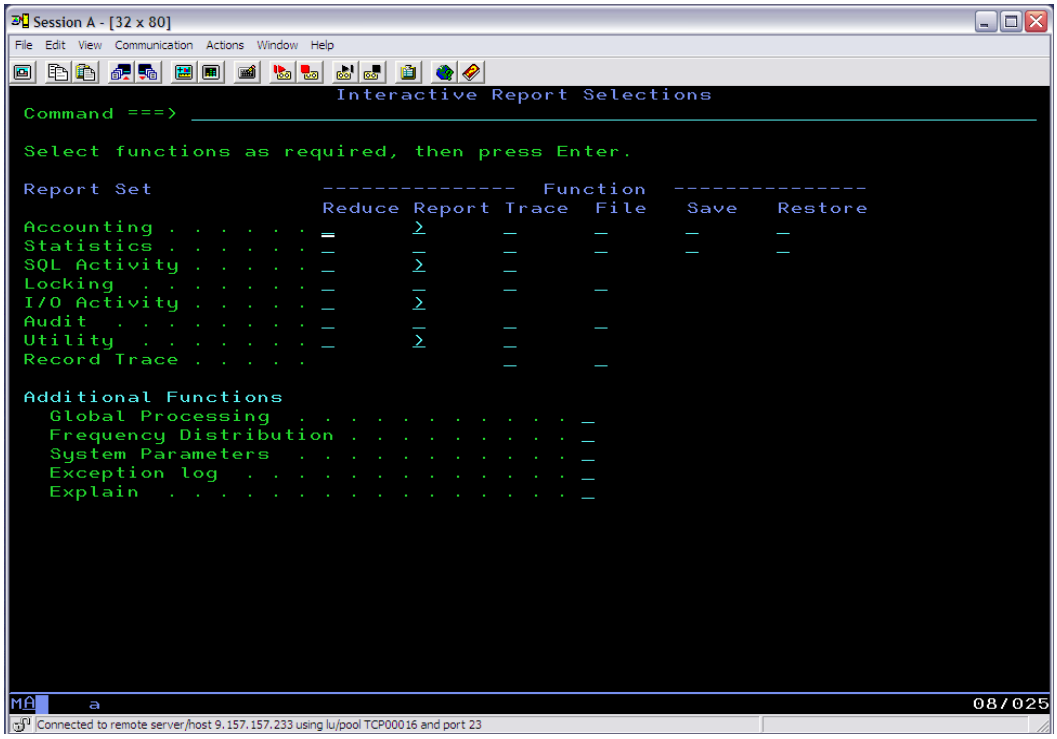
Slika 6: IBM DB2 Administration tool.

2.5.3 IBM Omegamon

Uporablja se za optimizacijo, nadzor ter oceno učinkovitosti delovanja podatkovne baze na Z/OS operacijskem sistemu.

IBM Omegamon nam omogoča:

- pregled obsežnih podatkov, kot so dostopni časi, časi izvajanja zahtev
- nadziranje, analiziranje in optimiziranje zmogljivosti DB2 -ja na Z/OS
- odkrivanje ozkih grl s pomočjo vnaprej definiranih pravil
- nastavljanje urnika nadzora



Slika 7: IBM Omegamon.

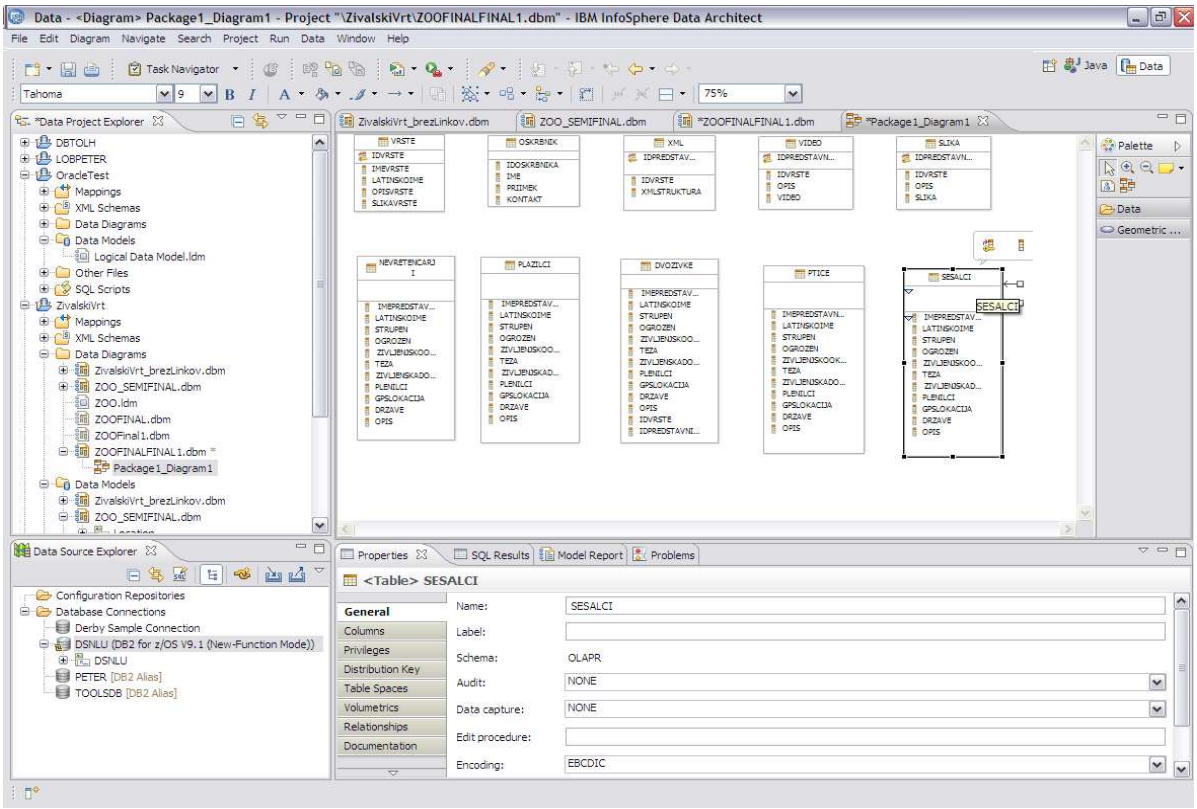
2.6 Razvojna orodja

2.6.1 IBM Infosphere Data Architect

Temelji na eclipse platformi. Je popolnoma samostojen produkt, ki nam olajša snovanje naše podatkovne strukture. V osnovi je namenjen izdelovanju podatkovnih modelov za DB2 podatkovno bazo. Z namestitvijo dodatnih vtičnikov lahko razvijamo podatkovne modele za poljubno podatkovno bazo.

Infosphere data Architect nam omogoča:

- vizualno izdelavo logičnega podatkovnega modela
- pretvorbo logičnega podatkovnega modela v fizični podatkovni model
- določanje tablespace-ov, podatkovnih tipov, podatkovnih baz, bufferpoolov, indexov.
- generiranje DDL datoteke iz fizičnega modela
- izvedbo DDL datoteke neposredno na podatkovno bazo
- ustvarjanje relacij med različnimi podatkovnimi bazami
- generiranje XML datoteke iz fizičnega modela
- objavo podatkovnega modela neposredno na spletu
- izdelavo poročila glede pravilnosti podatkovnega modela



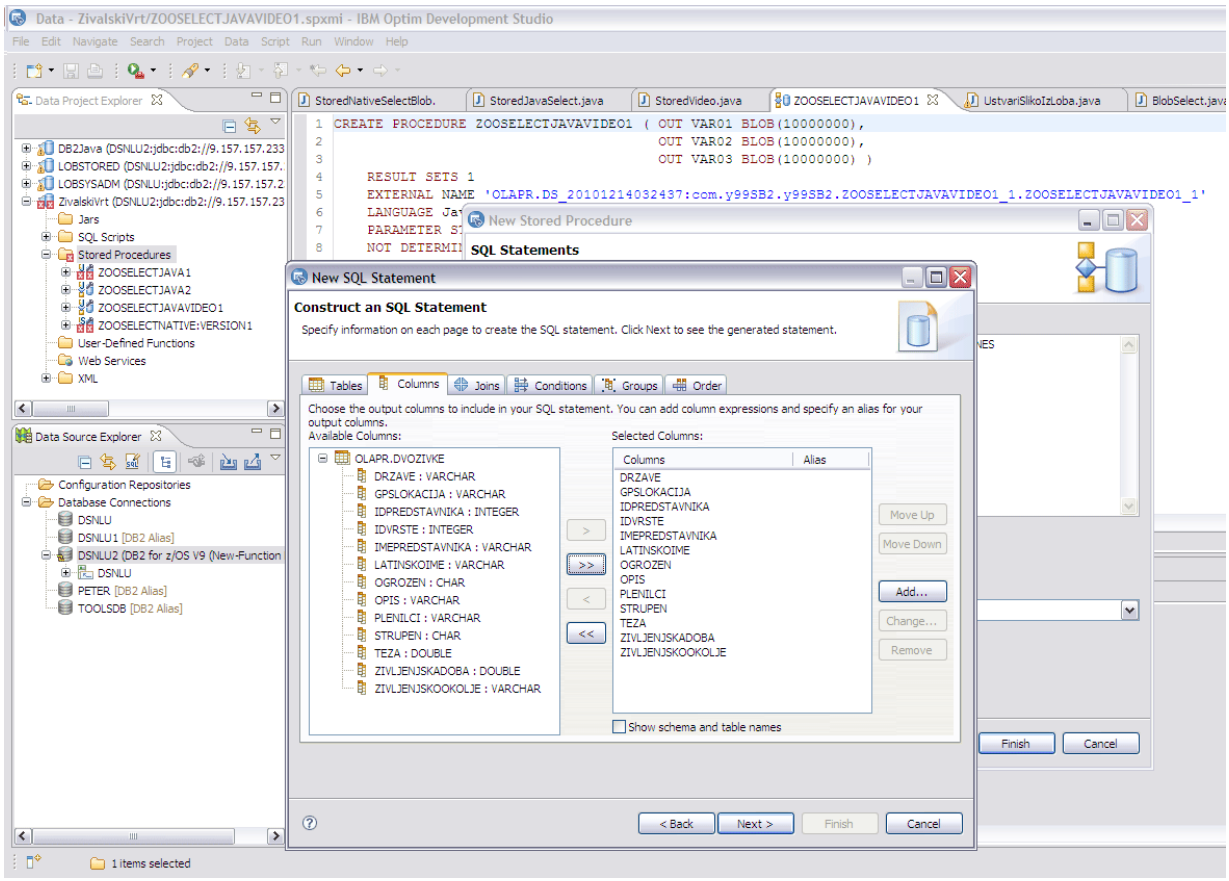
Slika 8: IBM Infosphere data architect.

2.6.2 IBM Optim development studio

Zagotavlja integrirano okolje za razvoj podatkovnih struktur za DB2, Oracle in Informix podatkovno bazo. Temelji na Eclipse platformi, ki je standard za večino IBM produktov.

Optim development studio nam omogoča:

- ustvarjanje povezav na podatkovno bazo ter testiranje aplikacij, ki uporabljajo javanske objekte, JSON ali XML.
- Enostavnejšo uporabo JDBC vtičnika, zato je dostop do podatkov hitrejši in enostavnejši.
- Enostavno analiziranje podatkov, spreminjanje izvorne kode ter spreminjanje podatkovnih shem.
- Izdelavo ter izvajanje shranjenih procedur.



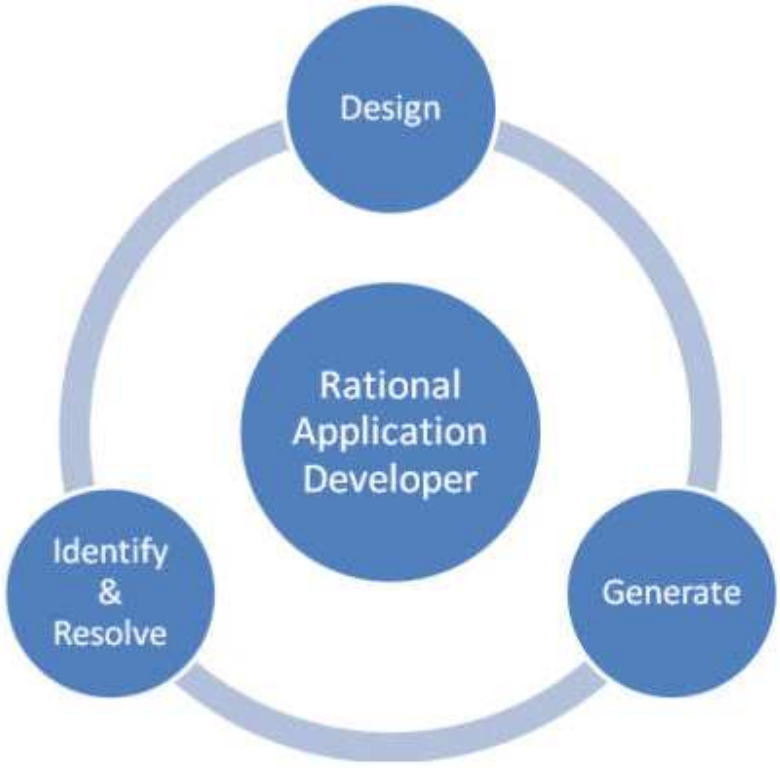
Slika 9: IBM Optim development studio.

2.6.3 IBM Rational application development

Je integrirano razvojno orodje, namenjeno razvijalcem Javanskih aplikacij. Vključuje specializirane čarovnike, urejevalnike in testerje, ki nam omogočajo izkoristiti moč javanske arhitekture.

Rational application development nam omogoča:

- podporo JAVA EE 6 z JPA 2.0 in EJB 3.1 za enostavnejši razvoj poslovnih aplikacij
- izdelati dinamične, preprosto vodene ter modularne aplikacije s podporo za OSG
- izdelavo web 2.0 aplikacij s podporo sodobnih tehnologij kot so ajax, dojo ter rest services
- uporabo vizualnih komponent za lažjo gradnjo aplikacije.

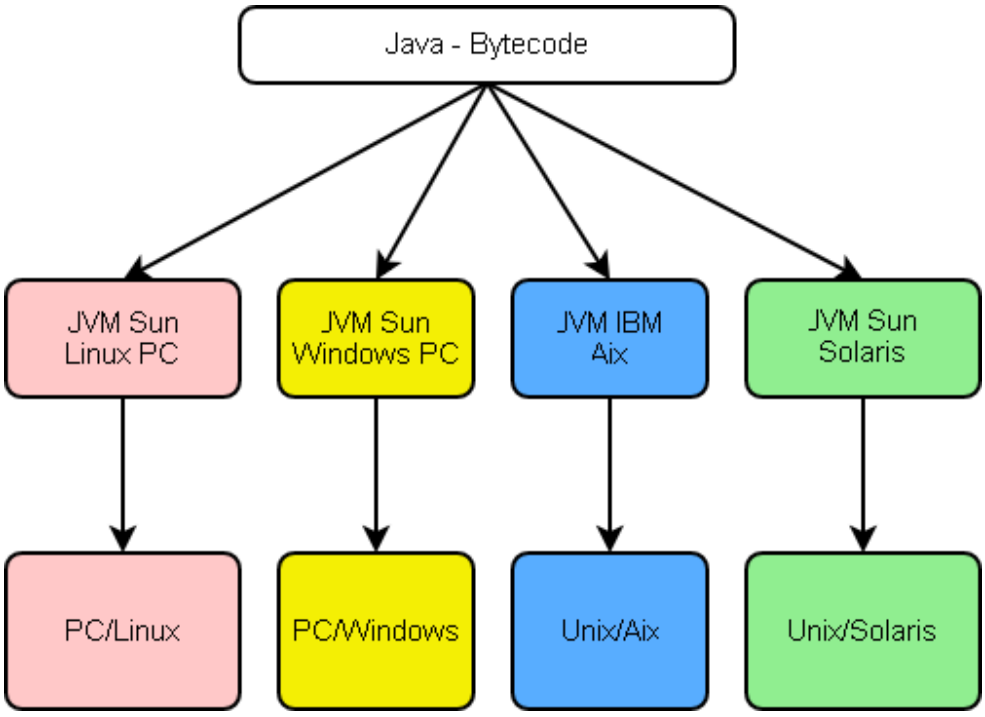


Slika 10: IBM Rational Application Developer.

2.7 Programski jeziki

2.7.1 Java

Programski jezik Java so razvili pri Sun Microsystems. Java je bila prvič v javnosti predstavljena 23. maja 1995. V tem času je bila Java zgolj programski jezik za popestritev spletnih strani, saj je bila Java počasna. Marca 1997 je Sun predstavil novo različico javanskega razvojnega kompleta, ki je nosil oznako 1.1. Java je do tega trenutka doživela kar precejšnje spremembe, podvojilo se je število razredov, dodali so standardno knjižnico za delo z zbirkami podatkov, prišlo pa je tudi do sprememb v jeziku samem (nov model obravnave dogodkov, nove rezervirane besede). Sedaj je Java moderen programski jezik. Java je opremljena z bogato standardno knjižnico programskih struktur in funkcij (za delo z datotekami, za dostop do podatkovnih baz, za mrežne povezave, hkratno izvajanje več programov, grafične aplikacije, aplikacije vgrajene v brskalnike, itn.). Standardna knjižnica je dobro dokumentirana.



Slika 11: Vmesna koda se s pomočjo JVM lahko uporabi na vseh platformah.

2.7.2 PL/I

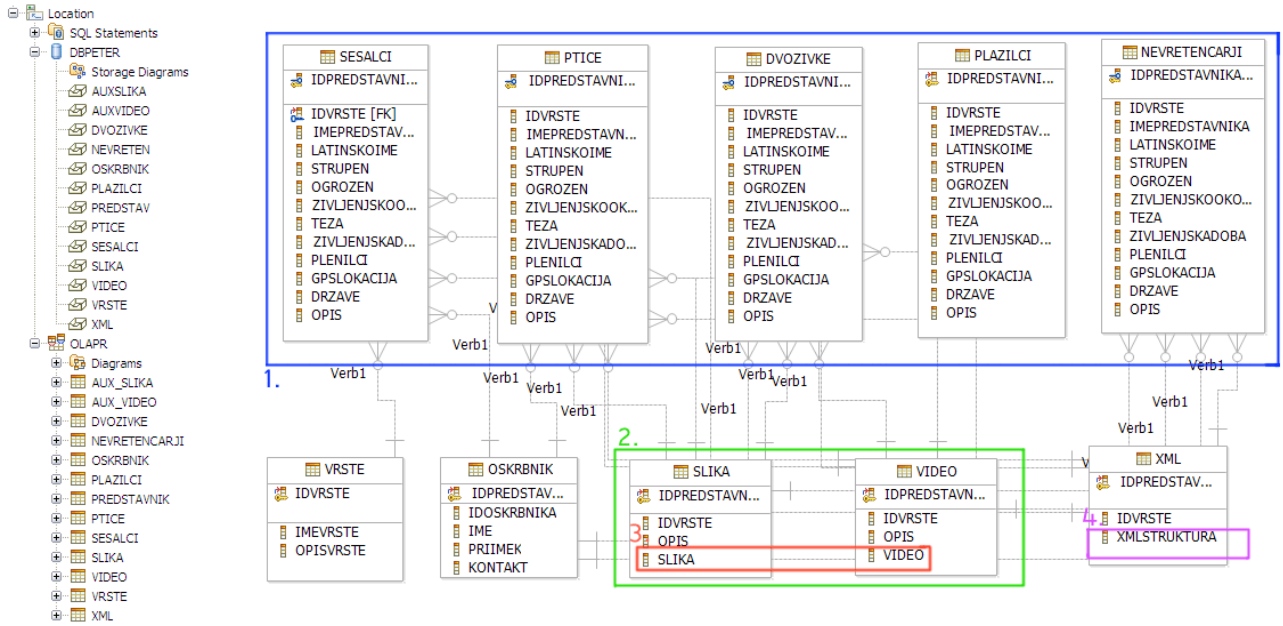
PL/I je programski jezik, ki se izvaja na strežniku. Omogoča nam polno funkcionalnost pri upravljanju s podatkovno bazo DB2 na z/OS sistemu. Začetki programskega jezika segajo v leto 1960, bil je prvi blokovno usmerjen programski jezik, ki se na IBM mainframe sistemih uporablja še danes. Velja za zelo zanesljiv in hiter programski jezik.

3 Predstavitev podatkovne sheme ter opis testnega okolja

Najprej sem na podlagi ideje s pomočjo Infosphere Data Architect orodja zgradil logični podatkovni model, ki je ustrezal vsem zahtevam. Nato sem logični model pretvoril v fizičnni model iz katerega generiramo DDL datoteke. DDL datoteke vsebujejo sql stavke, ki na podatkovni bazi definirajo naš podatkovni model. Po uspešnem deployu podatkovega modela na podatkovno bazo sem se lotil razvoja aplikacije s pomočjo katerih sem podatke vpisoval v tabele. Uporabil sem tri vrste tehnologij programskega jezika Java (dinamične zahteve SQL, SQLJ ter shranjene procedure (angl. Stored Procedure) ter terogramski jezik PL/I. Pri izvajanju testov sem se omejil na poizvedovanje LOB datotek. Pri vseh testih sem uporabil »SELECT SLIKA FROM OLAPR.SLIKA FETCH FIRST "+stevilo+" ROWS ONLY« zahtevo SQL, kjer »stevilo« predstavlja število zahtevanih objektov.

3.1 Podatkovna shema

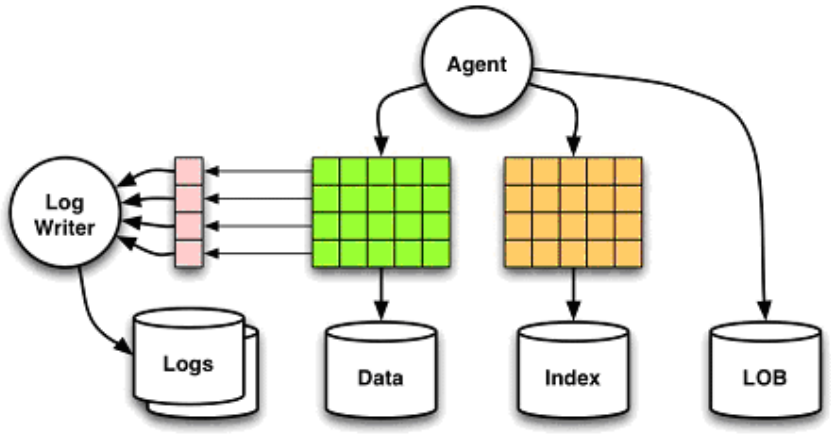
Naredil sem testno bazo, v kateri sem podatkovne sheme izdelal hierarhično. Pri vnosu podatkov moramo biti pozorni na vrstni red vnosa. V podatkovnih tabelah sem poleg osnovnih podatkovnih tipkov definiral še XML, BLOB ter CLOB podatkovne tipe, ki zahtevajo drugačen pristop za delo s podatki.



Slika 12: Podatkovna shema.

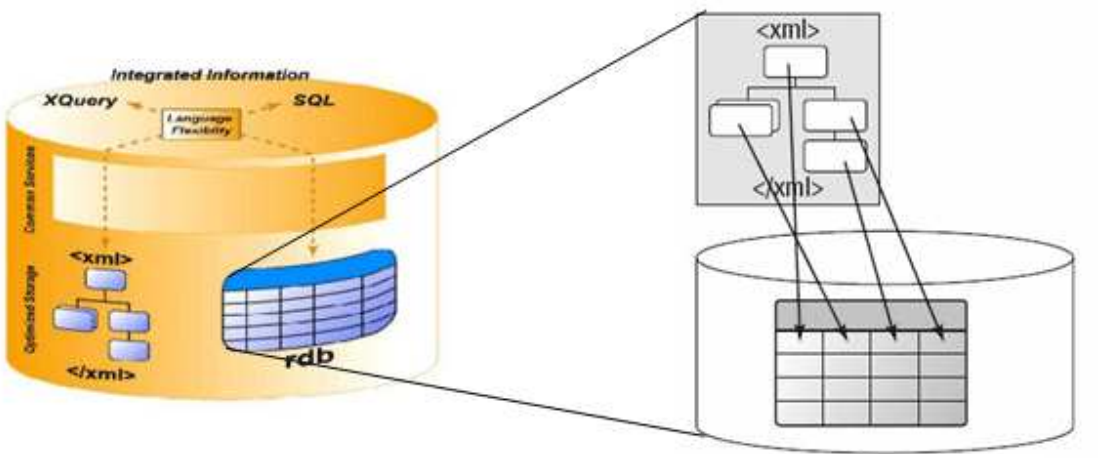
1. Podatkovna shema prikazuje logično razdelitev podatkov glede na skupino v katero žival spada. Povezava med posameznimi tabelami sta »IDVRSTE« in »IDPREDSTAVNIKA«. Vsaka tabela ima svoj »prostor za tabelo« (angl. Tablespace), ki predstavlja fizično lokacijo na disku.

2.,3. Poleg osnovnih podatkovnih tipov nam DB2 omogoča shranjevanje LOB (angl. large object) objektov. Za shranjevanje teh objektov je potrebno poleg standardne tabele definirati še »AUXILLARY TABLE« ter pripadajoči tablespace. V tabeli kamor preko vprašanja SQL vpisujemo LOB podatek se vpiše le referenca na auxillary table kjer se dejansko nahajajo vpisani podatki.



Slika 13: Zapisovanje LOB datotek v podatkovno bazo.

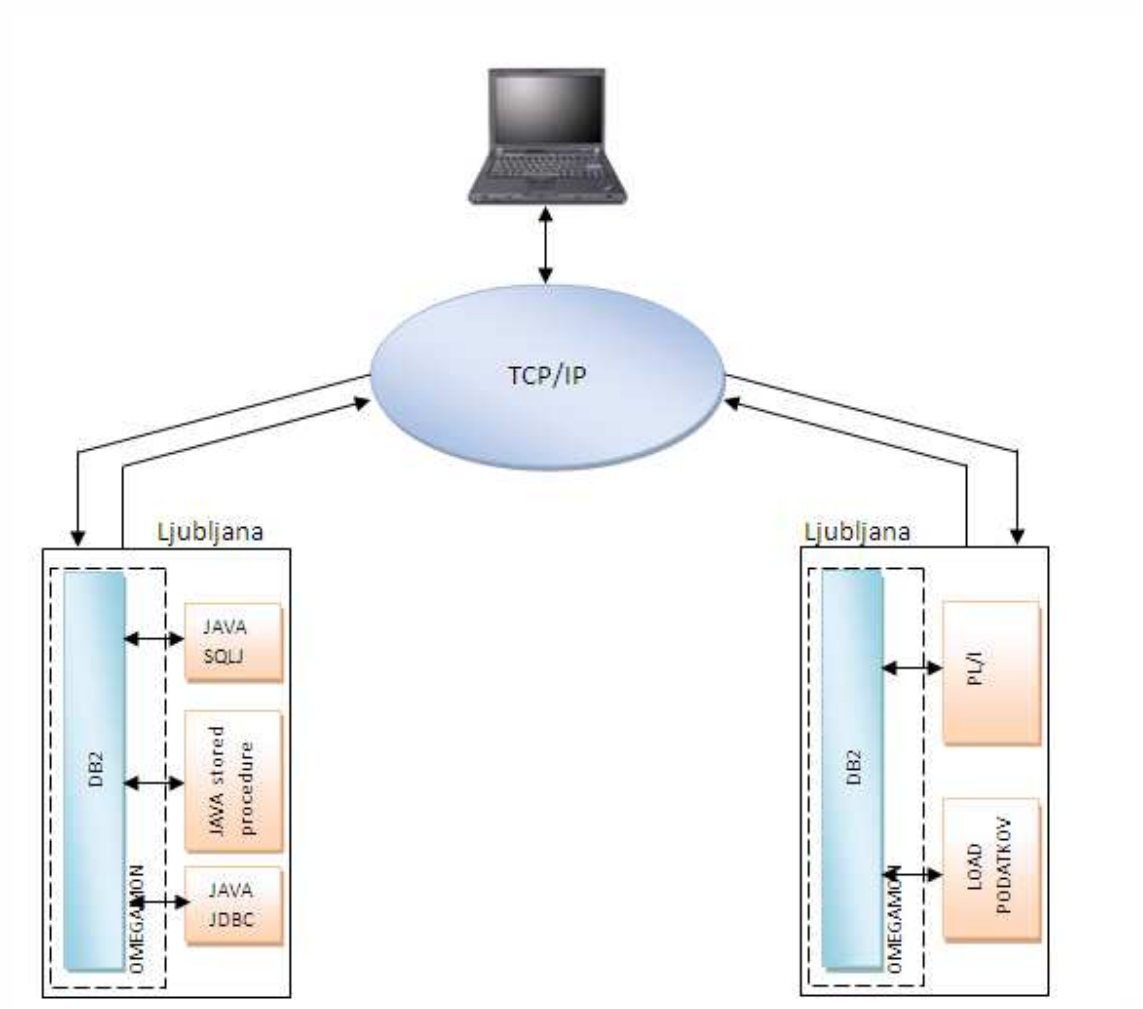
4. DB2 je prva podatkovna baza, ki omogoča polno funkcionalnost pri delu z XML datotekami. Princip shranjevanja xml datotek je podoben kot pri LOB datotekah le, da tukaj podatkovna baza sama določi auxillary table zato nimamo neposrednega vpliva nanjo. V bazo se shranjuje celotna xml struktura v enaki obliki kot jo imam zapisano v datoteki.



Slika 14: Zapisovanje XML datotek v bazo.

3.2 Testno okolje

Testno okolje temelji na IBM System Z strežniku, operacijskem sistemu z/OS ter podatkovni bazi DB2. Za primerjavo testov sem uporabil dva strežnika različnih sistemskih zmogljivosti. Podatkovna struktura, tipi in količina podatkov so na obeh strežnikih enaki. Pred vsako ponovitvijo testa sem počistil bufferpoll. Aplikacije s katerimi sem izvajal testiranje sem razvil s programskim jezikom Java in PL/I. Za nalaganje podatkov v podatkovno strukturo sem izdelal JOB. Meritve in poročila sem naredil z orodjem Omegamon.



Slika 15: Struktura testnega okolja.

3.3 Načina izvajanja zahtev SQL in razike med njima:

3.3.1 Statične zahteve SQL

Pri statičnih zahtevah SQL se preko funkcije deploy stavek SQL skupaj z programsko kodo, ki ga izvaja shrani na strežniku. Zahteva SQL je vseskozi enaka. Izvajajo se kot provedure na strežniku katere klient kliče znotraj aplikacije, kot rezultat klica dobimo podatke glede na SQL zahtevo.

- Avtentikacija na strežniku se zgodi samo v času shranjevanja procedure na strežnik
- Zahtevek za izvedbo procedure je klic funkcije
- Večja varnost
- Daljši postopek za spreminjanje zahtev SQL

3.3.2 Dinamične zahteve SQL

Zahtevo SQL strežniku posreduje aplikacija, ki jo izvaja klient. Kot rezultat na poslano SQL zahtevo nam strežnik vrne zahtevane podatke.

- Avtentikacija se zgodi ob vsaki zahtevi SQL
- Ob zahtevi se pošilja celotna zahteva SQL
- Večja nevarnost vrivanja kode SQL
- Lažje spreminjanje zatev SQL

3.4 Programska koda za delo s podatki v programskem jeziku Java

DB2 nudi popolno podporo programskemu jeziku Java. Java za povezovanje uporablja JDBC knjižnico. JDBC je neodvisen od platfomre in izvedbe podatkovne zbirke ter omogoča, da s podatkovnimi viri delamo enotno, ne glede na izbran podatkovni strežnik.

3.4.1 Branje podatkov iz podatkovne baze preko JDBC vtičnika:

```
public class test {
        Statement s = con.createStatement();
ResultSet rs = s.executeQuery("SELECT STRUPEN FROM OLAPR.DVOZIVKE WHERE
        IDPREDSTAVNIKA = "+id+" UNION "+"SELECT STRUPEN FROM
OLAPR.NEVRETENCARJI WHERE IDPREDSTAVNIKA = "+id+" UNION "+" SELECT
STRUPEN FROM OLAPR.PLAZILCI WHERE IDPREDSTAVNIKA = "+id+" UNION
"+"SELECT STRUPEN FROM OLAPR.PTICE WHERE IDPREDSTAVNIKA = "+id+"
UNION "+"SELECT STRUPEN FROM OLAPR.SESALCI WHERE IDPREDSTAVNIKA =
"+id);
         if (rs.next()) {
                strupen = rs.getString(1); } }
```

3.4.2 Izvajanje shranjenih procedur (angl. Stored Procedure):

Shranjena procedura se vedno izvaja na strežniku. Odjemalec poskrbi le za klic procedure z ustreznimi argumenti. Procedura ima lahko vhodne ali izhodne argumente s pomočjo katerih vpisujemo oz. izbiramo podatke v bazi.

```
CREATE PROCEDURE ZOOSELECTJAVA1 ( OUT VAR01 BLOB(500000),
                        OUT VAR02 BLOB(500000),
                        OUT VAR03 BLOB(500000),
                        OUT VAR04 BLOB(500000),
                        OUT VAR05 BLOB(500000) )
        RESULT SETS 1
        EXTERNAL NAME
'OLAPR.DS_20101118102849:com.y99SB2.y99SB2.ZOOSELECTJAVA1_1.ZOOSELECTJA
VA1_1'
        LANGUAGE Java
        PARAMETER STYLE JAVA
        NOT DETERMINISTIC
        COLLID DSNJDBC
        WLM ENVIRONMENT WLMENVJ
ALLOW DEBUG MODE
```

3.4.3 Branje podatkov iz podatkovne z uporabo SQLJ tehnologije:

SQLJ tehnologija je ena izmed statičnih metod branja podatkov iz baze. Zahteva SQL je v naprej definirana in je ni moč spreminjati. SQLJ struktura zahtev SQL je shranjena na strežniku, in je dosegljiva preko klicev posamezne funkcije znotraj SQLJ objekta. V main metodi najprej kreiramo nov objekt nato pa za izvedbo zahteve SQL uporabimo v naprej definirane funkcije.

SQLJ funkcija za izvedbo zahteve SQL:

```
public void execute(String username, String password) throws Exception {
                this.username = username;
                this.password = password;
                establishConnection();
                System.out.println("Retrieve some data from the database.");
                #sql [ctx] cursor1 = {SELECT SLIKA FROM OLAPR.SLIKA FETCH FIRST
                                1000 ROWS   ONLY};
                ArrayList cache = new ArrayList();
                int noOfCols = cursor1.getResultSet().getMetaData().getColumnCount();
                ArrayList row;
```

```
            while (cursor1.next()) {
                row = new ArrayList(noOfCols);
                for (int i = 1; i <= noOfCols; i++) {
                    row.add(cursor1.getResultSet().getObject(i));
                }
                cache.add(row);
            }
        iter = cache.iterator();
}
```

Testni razred za izvedbo SQLJ programa:

```
public static void main(String[] args)throws Exception {
        Select10000 slika = new Select10000();
        slika.execute("test","test");
        while (slika.next()) {
            System.out.println(slika.getSLIKA_SLIKA());
       } slika.close(); }
```

3.4.4 Branje LOB podatkov iz podatkovne baze:

LOB v bazo zapisujemo binarno kar pomeni, da jih tako tudi beremo. Če želimo datoteko prikazati kot sliko, pesem ali video moramo podatke pravilno zapisati v datoteko in jo shraniti kot ustrezni tip podatka.

```
        Statement s = con.createStatement();
        ResultSet rs = s.executeQuery ("SELECT SLIKA FROM OLAPR.SLIKA WHERE
                                IDPREDSTAVNIKA ="+id);
        if (rs.next()) {
                Blob photo = rs.getBlob(1);   }
        try {
            File blobFile   = new File(filename);
            FileOutputStream  outStream  = new FileOutputStream(blobFile);
            InputStream       inStream   = blob.getBinaryStream();
                int length  = -1;
                byte[]  buffer  = new byte[1024];
              while ((length = inStream.read(buffer)) != -1) {
                outStream.write(buffer, 0, length);
                outStream.flush(); } }
```

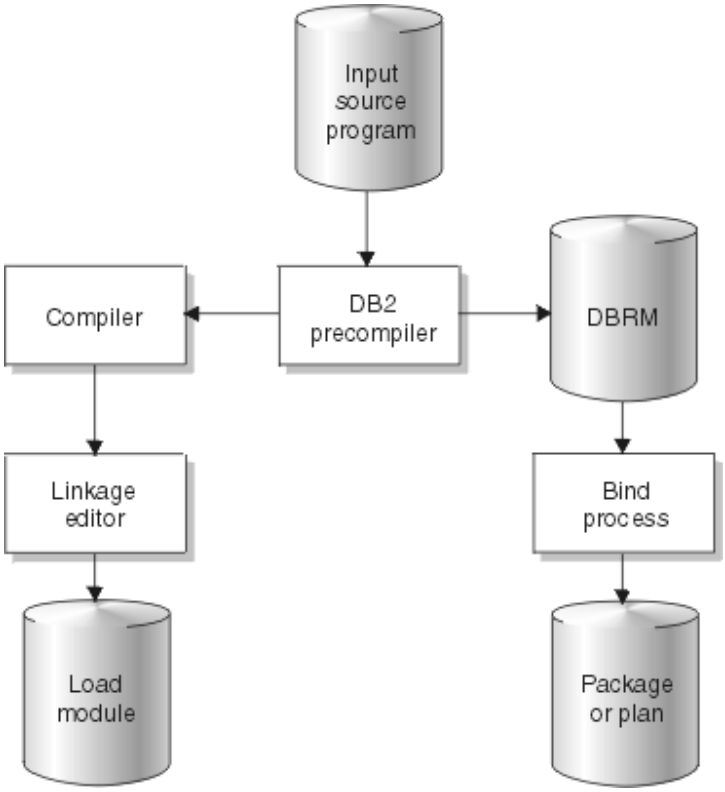
3.5 Programska koda za delo s podatki v programskem jeziku PL/I

PL/I je programski jezik, ki se izvaja na strežniku. Omogoča nam polno funkcionalnost pri upravljanju s podatkovno bazo DB2 na z/OS sistemu. Je eden prvih programskih jezikov, ki je omogočal manipulacijo s podatki v podatkovni bazi DB2.

PL/I program za izvedbo zahteve SQL:

```
EXEC SQL DECLARE OLAPR.SLIKA TABLE
        ( IDVRSTE                   INTEGER,
          IDPREDSTAVNIKA              INTEGER NOT NULL,
          OPIS                    VARCHAR(50),
          SLIKA                    BLOB(5000000)
        ) ;
/******************************************************************/
/* PLI DECLARATION FOR TABLE OLAPR.SLIKA                          */
/******************************************************************/
DCL 1 DCLSLIKA,
    5 IDVRSTE  BIN FIXED(31),
    5 IDPREDSTAVNIKA  BIN FIXED(31),
    5 OPIS     CHAR(50) VAR,
    5 SLIKA    SQL TYPE IS BLOB_LOCATOR;
I=J=K=0;

PUT SKIP LIST ('DECLARING C1 CURSOR ');

EXEC SQL DECLARE C1 CURSOR
  FOR SELECT IDVRSTE FROM OLAPR.SLIKA
  FETCH FIRST 50000 ROWS ONLY ;

PUT SKIP EDIT ('DECLARE CURSOR SQLCODE = ', SQLCODE)(A);
PUT SKIP(1);

PUT SKIP LIST ('OPENING CURSOR C1 ');
EXEC SQL OPEN C1;
PUT SKIP EDIT ('OPEN SQLCODE = ', SQLCA.SQLCODE)(A);
PUT SKIP(1);
PUT SKIP LIST ('FETCHING CURSOR C1 ');
EXEC SQL FETCH C1
   INTO :DCLSLIKA.IDVRSTE;
IDVR = IDVRSTE;
```

```
IF SQLCODE=0 THEN DO;
I=I+1;
J=J+1;
              END;
ELSE DO;
  PUT SKIP EDIT ('FETCH SQLCODE =',SQLCODE)(A);
  GOTO KONEC;
END;

DO WHILE (SQLCODE=0);
EXEC SQL FETCH C1
    INTO :DCLSLIKA.IDVRSTE;
      IF SQLCODE=0 THEN DO;
                I=I+1;
                J=J+1;
                    END;
                ELSE DO;
            PUT SKIP EDIT ('FETCH SQLCODE =',SQLCODE)(A);
                GOTO KONEC;
                    END;
IF J > 1000 THEN  DO;
 K=K+1;
    PUT SKIP EDIT ('KJ=',K!!' '!!J)(A);
   IF K>=50  THEN GOTO KONEC;
   J =0;
 END;
 END; /* END DO WHILE */

 KONEC:
    EXEC SQL CLOSE C1;
    PUT SKIP EDIT ('KONEC ')(A);
 END;
```



Slika 16: Delovanje PL/I prevajalnika v operacijskem sistemu z/OS.

.

3.6 Programska koda za vpisovanje podatkov v programskem jeziku JCL (job control language)

JCL je skripti programski jezik, ki je se uporablja na IBM maninframe sistemih. Uporablja se za pisanje ter izvajanje posameznih zahtev (angl. JOBs). Omogoča nam pisanje tako enostavnih kot kompleksnih sistemskih skript, ki nam olajšajo nadzor nad sistemom.

```
//SLIKA     DD  DSN=DSN910.PETER.LOAD.SLIKA,DISP=SHR
LOAD DATA INDDN(SLIKA) LOG NO RESUME YES
    INTO TABLE OLAPR.SLIKA
        (IDVRSTE           POSITION(01: 5) INTEGER EXTERNAL ,
         OPIS              POSITION(11:60) CHAR(50),
         SLIKA             POSITION(61)  CHAR(25) BLOBF)
```

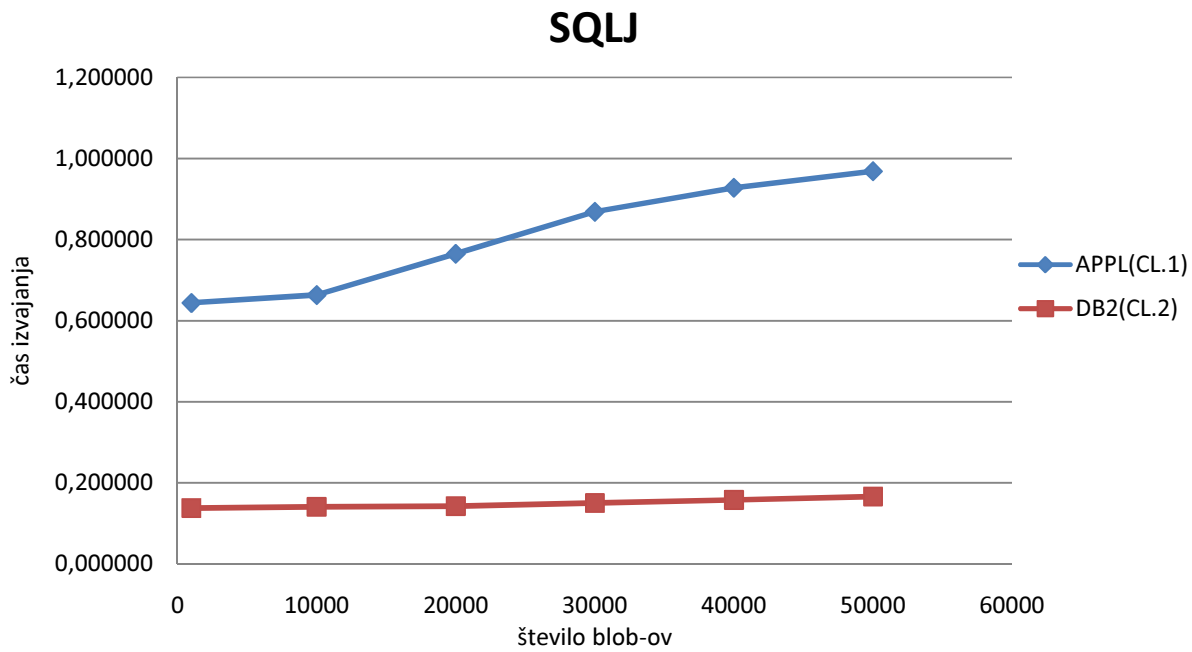
4 Predstavitev rezultatov meritev:

4.1 Meritev z uporabo SQLJ tehnologije:

Table 1: Podatki SQLJ meritev.

Število blob-ov	APPL(CL.1)	DB2(CL.2)
1000	0,643882	0,137710
10000	0,663314	0,140700
20000	0,764667	0,142030
30000	0,868307	0,149780
40000	0,927806	0,157720
50000	0,968278	0,166318

APPL(CL.1) – čas, ki ga je potrebovala aplikacija za izvedbo zahteve SQL
DB2(CL.2) – čas, ki ga potrebovala podatkovna baza za izvedbo zahteve SQL (čas je pomnožen s faktorejm 10 zaradi lažje predstavitve na grafu)



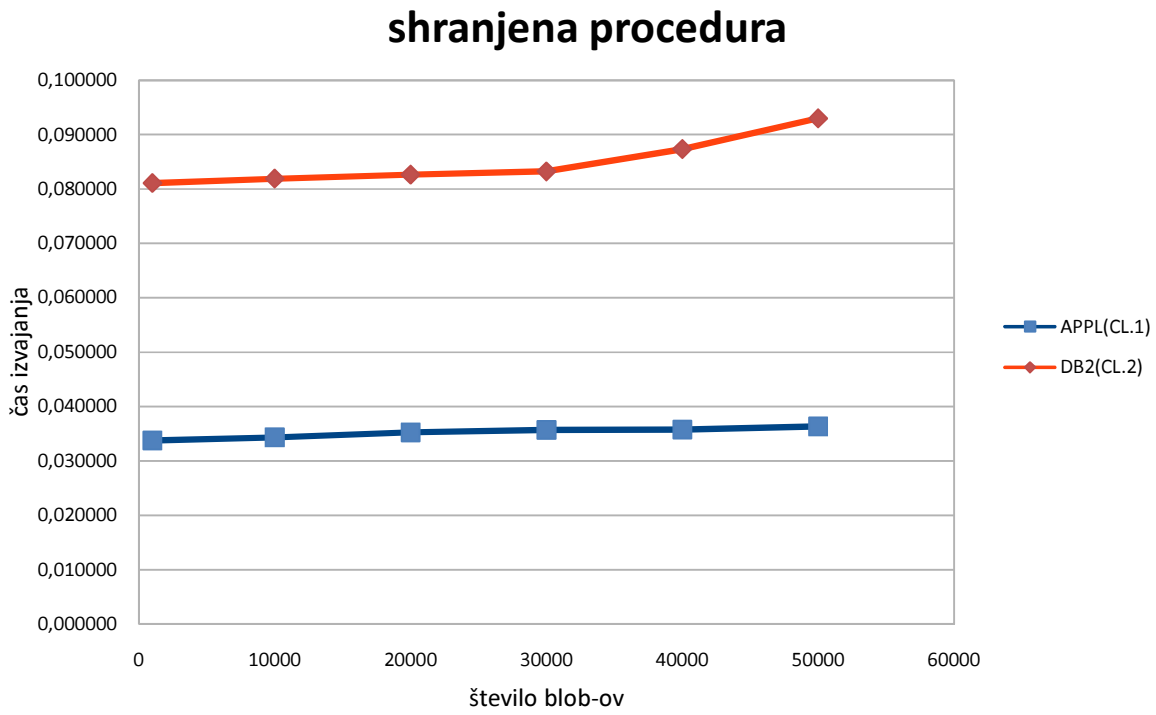
Graf 1: Graf SQLJ meritve.

4.2 Meritev z uporabo shranjenih procedur:

Table 2: Podatki shranjenih procedur meritev.

Število blob-ov	APPL(CL.1)	DB2(CL.2)
1000	0,033729	0,081090
10000	0,034299	0,081910
20000	0,035244	0,082640
30000	0,035666	0,083250
40000	0,035759	0,087350
50000	0,036322	0,092990

APPL(CL.1) – čas, ki ga je potrebovala aplikacija za izvedbo zahteve SQL
DB2(CL.2) – čas, ki ga je potrebovala podatkovna baza za izvedbo zahteve SQL (čas je pomnožen s faktorejm 10 zaradi lažje predstavitve na grafu)



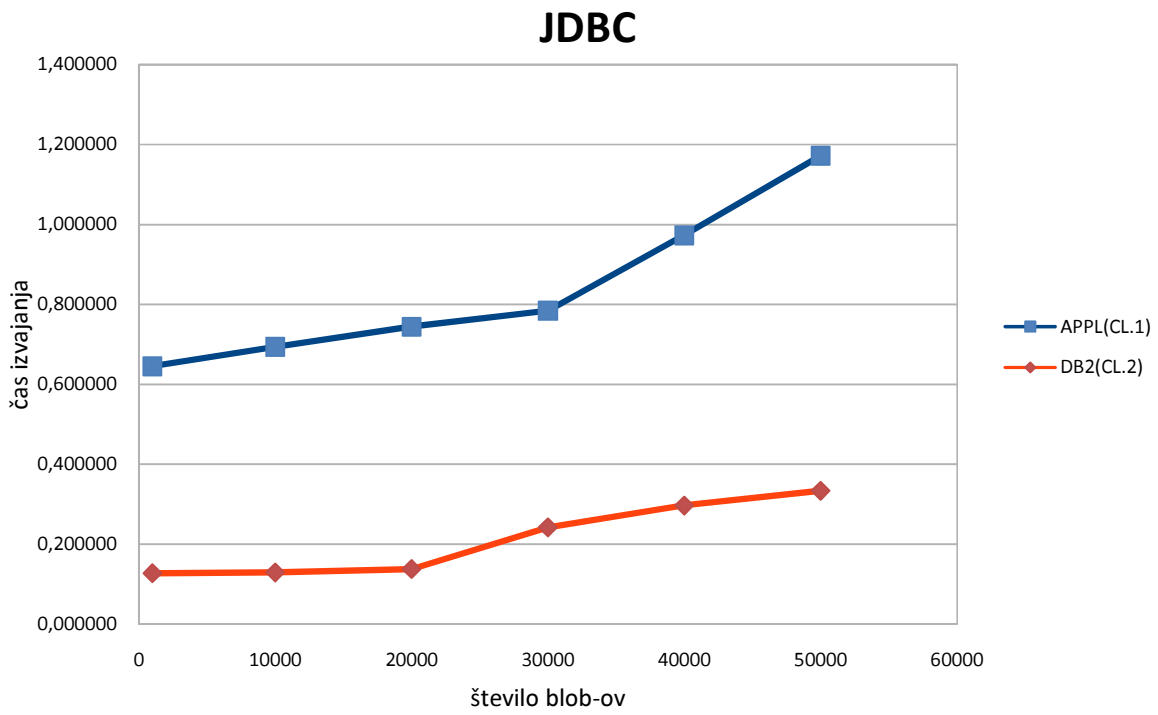
Graf 2: Graf meritve shranjenih procedur.

4.3 Meritev z uporabo dinamične zahteve SQL:

Table 3: Podatki meritev dinamične zahteve SQL.

Število blob-ov	APPL(CL.1)	DB2(CL.2)
1000	0,645434	0,127410
10000	0,693829	0,129420
20000	0,744103	0,137700
30000	0,784325	0,242130
40000	0,972786	0,296820
50000	1,171599	0,333720

APPL(CL.1) – čas, ki ga je potrebovala aplikacija za izvedbo zahteve SQL
DB2(CL.2) – čas, ki ga potrebovala podatkovna baza za izvedbo zahteve SQL (čas je pomnožen s faktorejm 10 zaradi lažje predstavitve na grafu)



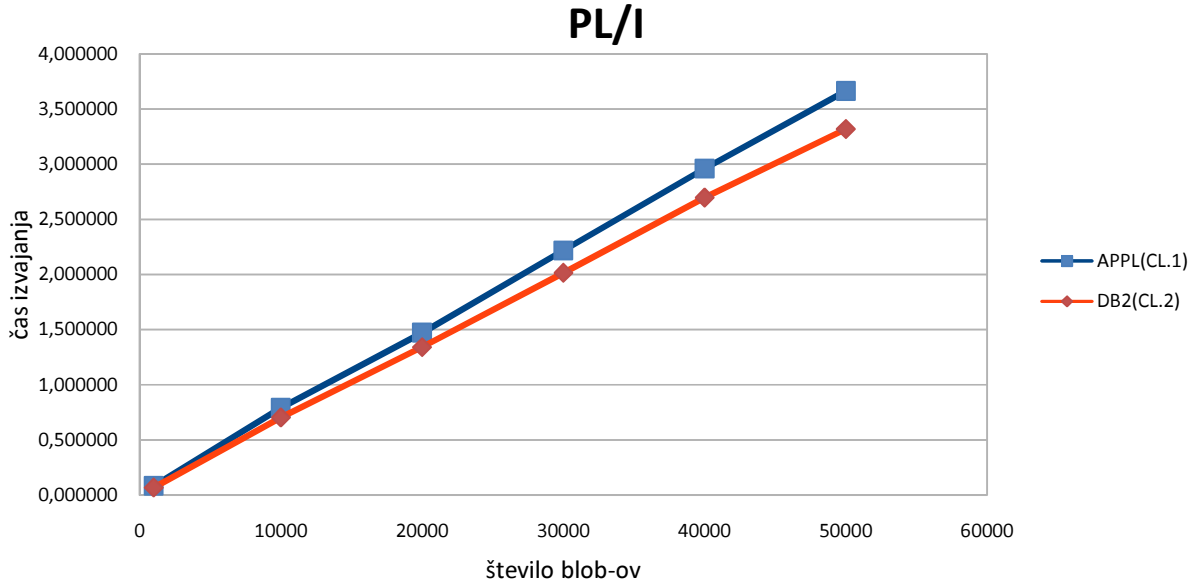
Graf 3: Graf meritve pri uporabi dinamične zahteve SQL.

4.4 Meritev z uporabo programskega jezika PL/I:

Table 4: Podatki meritev PL/I.

Število blob-ov	APPL(CL.1)	DB2(CL.2)
1000	0,086634	0,067869
10000	0,792064	0,702399
20000	1,473467	1,340410
30000	2,218311	2,014029
40000	2,959484	2,695362
50000	3,662885	3,317371

APPL(CL.1) – čas, ki ga je potrebovala aplikacija za izvedbo zahteve SQL
DB2(CL.2) – čas, ki ga potrebovala podatkovna baza za izvedbo zahteve SQL (čas je pomnožen s faktorejm 10 zaradi lažje predstavitve na grafu)



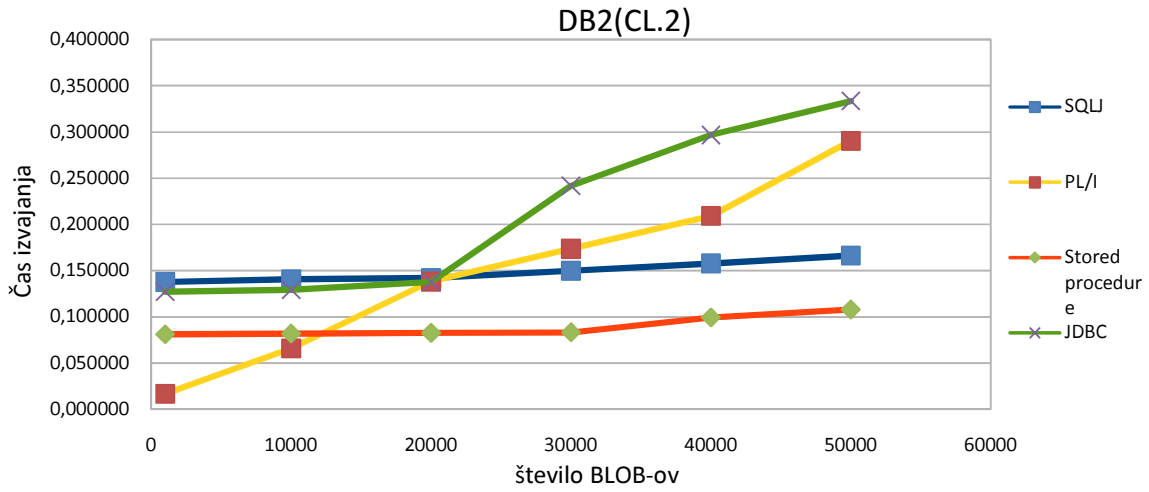
Graf 4: Graf meritve z uporabo programskega jezika PL/I.

4.5 Primerjava vseh opravljenih meritev:

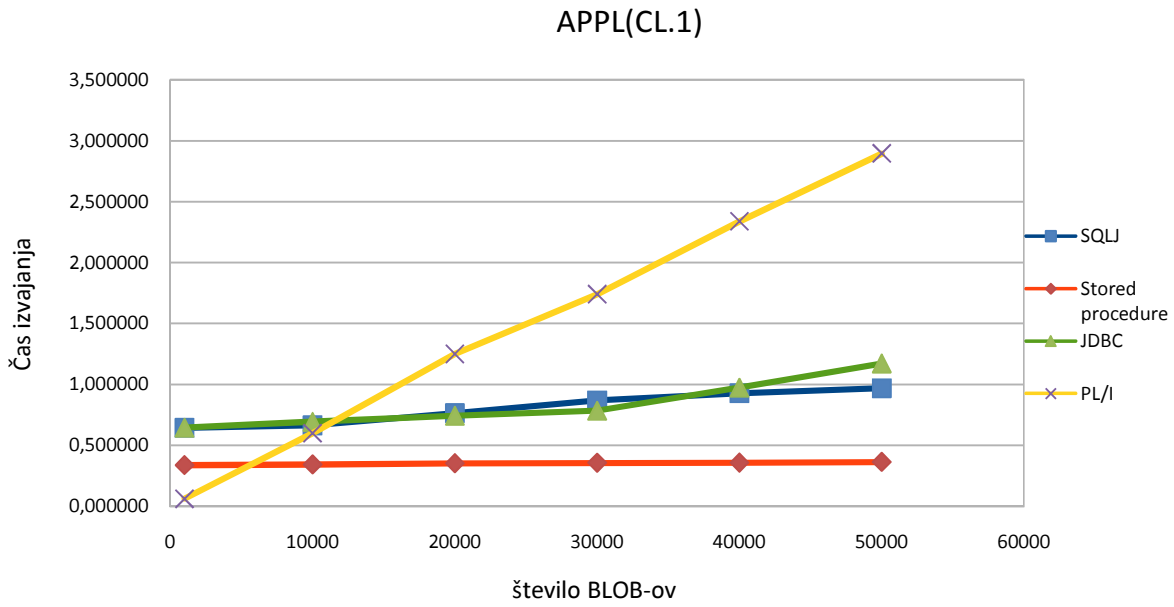
Table 5: Tabela podatkov vseh meritev.

	SQLJ		*Stored Procedure*		*PL/I*		*JDBC*	
Število blob-ov	**APPL (CL.1)**	**DB2 (CL.2)**	**APP L(CL.1)**	**DB2 (CL.2)**	**APPL (CL.1)**	**DB2 (CL.2)**	**APPL (CL.1)**	**DB2 (CL.2)**
1000	0,64388	0,13771	0,33729	0,08109	0,06197	0,01671	0,64543	0,12741
10000	0,66331	0,14070	0,34299	0,08191	0,60065	0,06600	0,69382	0,12942
20000	0,76466	0,14203	0,35244	0,08264	1,25095	0,13826	0,74410	0,13770
30000	0,86830	0,14978	0,35666	0,08325	1,74082	0,17370	0,78432	0,24213
40000	0,92780	0,15772	0,35759	0,09935	2,33853	0,20931	0,97278	0,29682
50000	0,96827	0,16631	0,36322	0,10799	2,89605	0,29047	1,17159	0,33372

APPL(CL.1) – čas, ki ga je potrebovala aplikacija za izvedbo zahteve SQL
DB2(CL.2) – čas, ki ga potrebovala podatkovna baza za izvedbo zateve SQL (čas je pomnožen s faktorejm 10 zaradi lažje predstavitve na grafu)



Graf 5: Graf primerjave vseh opravljenih meritev DB2.



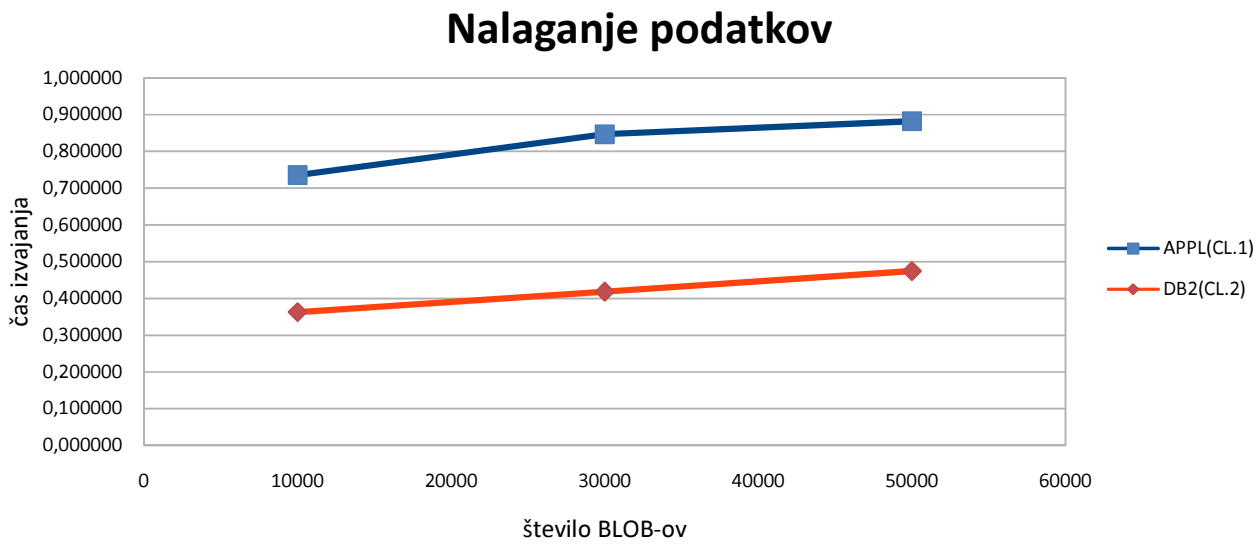
Graf 6: Graf primerjave vseh opravljenih meritev APPL.

4.6 Meritev nalaganja BLOB podatkov v podatkovno bazo s programskim jezikom JCL:

Table 6: Podatki o nalaganju podatkov v podatkovno bazo.

Število blob-ov	APPL(CL.1)	DB2(CL.2)
10000	0,735543	0,362861
30000	**0,846890**	**0,418291**
50000	0,882751	0,474457

APPL(CL.1) – čas, ki ga je potrebovala aplikacija za izvedbo zahteve SQL
DB2(CL.2) – čas, ki ga potrebovala podatkovna baza za izvedbo zateve SQL (čas je pomnožen s faktorejm 10 zaradi lažje predstavitve na grafu)



Graf 7: Graf nalaganja podatkov.

5 Povzetek

Namen mojega testiranja je bilo ugotoviti, katere tehnologije se trenutno performančno najbolje obnesejo pri delu s podatkovno bazo DB2 na operacijskem sistemu z/OS. Podatki s katerimi sem izvajal meritve so bili tipa BLOB, CLOB in XML. Kot primerjavo sem uporabil vse bolj razširjeni programski jezik Java ter v mainframe svetu poznan jezik PL/I.

Podatki APPL(CL.1)
Predstavljajo čas, ki ga je za izvedbo posamezne zahteve SQL potrebovala aplikacija. Kot je razvidno iz grafa na sliki 17 so razlike pri majhni količini do 10000 podatkov minimalne, nato pa se začnejo vrednosti spreminjati. Tukaj je lepo vidna razlika pri uporabi dinamične (JDBC) in statične (SQLJ, shranjene procedure, PL/I) zahteve SQL. V splošnem velja, da je uporaba statičnih zahtev SQL hitrejša ter procesorsko manj zahtevna. Izjema na grafu je premica, ki prikazuje časovno zahtevnost pri programskem jeziku PL/I. V našem primeru se je PL/I izkazal kot časovno najzahtevnejši programski jezik glede aplikacijske zahtevnosti. Razlog za to je v tem, da smo samo pri tem testu podatke iz baze zapisali na disk. V skupni čas izvajanja aplikacije je tako poleg izvajanja zahteve SQL vključen še zapis podatkov iz baze na disk.

Podatki DB2(CL.2)
Predstavljajo čas, ki ga je za izvedbo posamezne zahteve SQL potrebovala podatkovna baza DB2. Iz grafa na sliki 16 je razvidno kako časovno zahteven je določen programski jezik oz. uporabljena tehnologija glede na količino podatkov. V našem primeru je bilo časovno najučinkovitejše poizvedovanje z uporabo shranjenih procedur (statični način zahtev SQL). Čas se je s količino podatkov spreminjal minimalno, tako smo dokazali, da so pri velikih količinah časovno najugodnejše SQL zahteve pri katerih uporabljamo statične metode zahtev.

Med samo pripravo okolja, sem naletel na kar nekaj težav, katere sem s pomočjo mentorja Saša Preku (zSeries Software Sales Specialist & Member of zChampion Team) uspešno obvladal, ter se pri tem naučil veliko novega in zanimivega. Na projektu sem uspešno uporabil ter nadgradil praktično in teoretično znanje pridobljeno v času študija. Glede na količino podatkov, ki dnevno narašča, je upravljanje ter hiter dostop do le teh bistvenega pomena.

6.0 Literatura

[1] M. S. Almeida, K. Condon, M. Fischer, J. Stuhler, *DB2 Java Stored Procedures Learning by Example*, North Castle, NY: IBM,2004,str: 780.

[2] P. Bruni, P. Becker, T. Bohlsen, B. Diekmann, D. Etkin, D. Goethals, *International Technical Support Organization LOBs with DB2 for z/OS: Stronger and Faster,* North Castle, NY: IBM, 2006, str:453.

[3] P. Bruni, P. Becker, M. Dewert, B. Riehle *International Technical Support Organization Large Objects with DB2 for z/OS and OS/390*, North Castle, NY: IBM, 2003, str: 430.

[4] P. Bruni, S. Kaschta, M. Kutsch, G. McGeoch, M. Scanlon, J. Vandensande, *DB2 9 for z/OS Stored Procedures: Through the CALL and Beyond*, North Castle, NY: IBM, 2007, str: 615.

[5] M. Ebbers, J. Kettner, W. O'Brien, B. Ogden, *Introduction to the New Mainframe: z/OS Basics*, North Castle, NY: IBM, 2000, str: 915.

[6] V. Gopal, S. Bhagavan, *Data modeling with InfoSphere Data Architect and Informix Dynamic Server*, North Castle, NY: IBM, 2003, str. 380.

[7] IBM, *DB2 10 for z/OS Installation and Migration Guide*, North Castle, NY: IBM, 2002, str:763.

[8] (2010) A history of Windows. Dostopno na: http://www.microsoft.com/windows/winhistoryprographic.mspx.

[9] (2011) CallableStatement. Dostopno na: http://download.oracle.com/javase/1.4.2/docs/guide/jdbc/getstart/callablestatement.html.

[10] (2010) Client (computing). Dostopno na: http://en.wikipedia.org/wiki/Client_%28computing%29.

[11] (2010) Control Center overview. Dostopno na: http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.cc.doc/db2_udb/intro.htm.

[12] (2010) Create Stored Procedure wizard. Dostopno na: http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.dc.doc/dc/g_spcreate.htm.

[13] (2010) DB2 Administration Tool for z/OS. Dostopno na:: http://www-01.ibm.com/software/data/db2imstools/db2tools/db2admin/.

[14] (2011) IBM DB2. Dostopno na:
http://en.wikipedia.org/wiki/IBM_DB2.

[15] (2010) DB2 development. Dostopno na:
http://forums.devshed.com/db2-development-114/select-from-db2-with-php-invalid-value-specified-for-keyword-concurrency-467662.html.

[16] (2011) IBM Optim Development Studio product overview. Dostopno na:
http://publib.boulder.ibm.com/infocenter/idm/v2r2/index.jsp?topic=/com.ibm.datatools.dwb.nav.doc/topics/cprodover_dwb.html.

[17] (2010) IBM Optim Development Studio: Routine development simplified. Dostopno na:
http://www.ibm.com/developerworks/data/library/techarticle/dm-1010devstudioroutines/index.html.

[18] (2010) IBM Rational Application Developer. Dostopno na:
http://en.wikipedia.org/wiki/IBM_Rational_Application_Developer.

[19] (2011) Installing the IBM DB2 Driver for JDBC and SQLJ. Dostopno na:
http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.apdv.java.doc/doc/t0010264.htm.

[20] (2010) Java insert an image. Dostopno na:
http://www.java2s.com/Code/Java/Database-SQL-JDBC/InsertanImage.htm.

[21] (2010) Java (programming language). Dostopno na:
http://en.wikipedia.org/wiki/Java_%28programming_language%29.

[22] (2010) Windows XP. Dostopno na:
http://sl.wikipedia.org/wiki/Windows_XP.

[23] (2011) Rational Data Architect guide. Dostopno na:
http://chinajava.net/resource/Rational%20Data%20architect%20guide.pdf.

[24] (2011) z/OS. Dostopno na:
http://en.wikipedia.org/wiki/Z/OS.

[25] (2010) z/OS. Dostopno na:

http://groups.engin.umd.umich.edu/CIS/course.des/cis400/pl1/pl1.html.

Izjava o samostojnosti dela:

Izjavaljam, da sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Saše Divjaka. Izkazano pomoč sodelovcev sem v celoti navedel v zahvali.