Izrada baze za prikupljanje podataka o kvaliteti proizvodnog procesa

Bašić, Tea

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike

Tea Bašić

Izrada baze za prikupljanje podataka o kvaliteti proizvodnog procesa

Završni rad

Osijek, 2021.

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike

Tea Bašić

Izrada baze za prikupljanje podataka o kvaliteti proizvodnog procesa

Završni rad

Mentor: doc. dr. sc. Slobodan Jelić
Komentori: dr. sc. Mateja Đumić i Matej Đuroković

Osijek, 2021.

Sažetak

U ovom završnom radu će biti prikazane osnovne informacije o SQL programskom jeziku te kako pomoću njega izraditi bazu za prikupljanje podataka o kvaliteti proizvodnog procesa. Prilikom izrade baze koristile su se zakonitosti izrade relacijskih baza podataka i sustav za upravljanje bazom podataka MySQL. Kako bi se pobliže razradila tema ovoga rada izrađen je i praktični dio rada koji je rađen u suradnji s tvrtkom Orqa d.o.o i mentorom M. Đurokovićem. Radi tajnosti i zaštite podataka tvrtke Orqa d.o.o. u radu neće biti prikazani stvarni podaci korišteni prilikom izrade praktičnog dijela.

Ključne riječi

Baza podataka, SQL, MySQL, RDBMS

Development of a database for the collection of data on production process quality

Summary

This final paper will present basic information on the programming language SQL, show the development of a database for the collection of data on production process quality. Regularities of development of relational databases and database management system MySQL were used to create the database. In order to elaborate on the topic of this paper in more detail, there is also a practical part of the paper, made in collaboration with the company Orqa d.o.o and mentor M. Đuroković. For the purpose of confidentiality and data protection of Orqa d.o.o, the paper will not present real data used in the making of the practical part of this paper.

Key words

Database, SQL, MySQL, RDBMS

Sadržaj

Uvod

"Proizvodni proces je osnova svake industrijske proizvodnje, a podrazumijeva sve aktivnosti i djelovanja koja rezultiraju pretvaranjem ulaznih materijala u gotov proizvod."
(2019, Anonymous, [7])

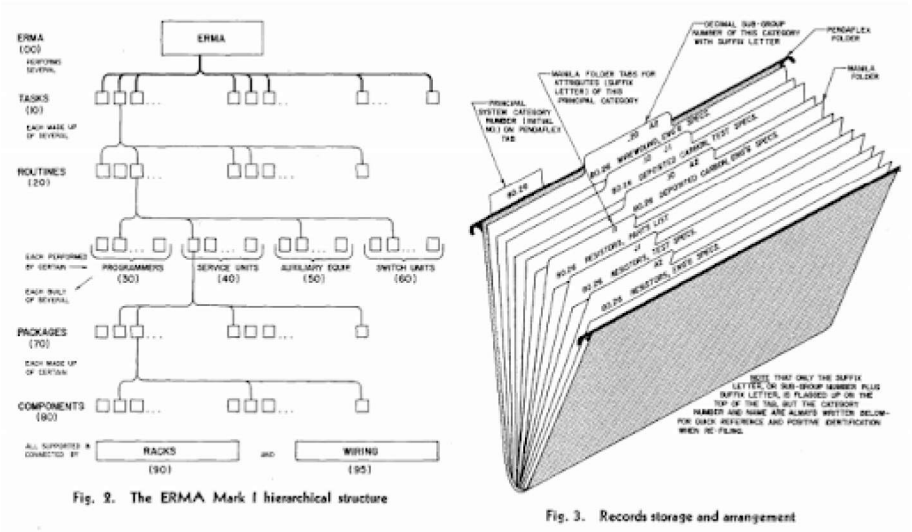
Današnje tržište je dinamično i svakodnevno se mijenja. Postoje određeni zahtjevi kojima se potrebno prilagođavati u cilju postizanja efikasnosti i efektivnosti proizvodnje. Kako bismo dobili kvalitetne informacije o svim segmentima proizvodnog procesa, potrebno je na određen način organizirati sve podatke. Upravo iz tih razloga baze podataka su postale esencijalni dio gotovo svake tvrtke koja se bavi proizvodnjom.

Jedan od najraširenijih sustava za upravljanje relacijskim bazama podataka koji se danas koristi je MySQL. MySQL je sustav otvorenog koda koji se pokreće na poslužitelju, te pruža pristup višestrukom broju korisnika i pohranu višestrukog broja baza podataka. Neke od najvećih svjetskih kompanija poput YouTube-a, Netflix-a, Facebook-a, Tesle, te mnogih drugih koriste MySQL baze podataka. Upravo je Tesla primjer kompanije koja MySQL bazu podataka koristi za praćenje proizvodnog procesa. [1]

1 Definicija i kratka povijest baza podataka

"Baze podataka definiramo kao organizirane skupove podataka koji su međusobno povezani. To su organizirane zbirke podataka koje podatke drže na jednom mjestu. Pristup podacima je kontroliran i podaci su spremljeni bez redundancije" (Carić i Buntić, 2015, [2]).

Ne tako davno u prošlosti, dok nisu postojale baze podataka, sve se moralo bilježiti pomoću papira i olovke. Postojali su popisi, dnevnici, glavne knjige te arhive koje su sadržavale i milijune zapisa u kartotekama, a samim tim su bile vrlo nepregledne. Pronalaženje i fizičko dobivanje zapisa bilo je sporo i problematično. Nerijetko se događalo da se zapis stavi na krivo mijesto ili zagubi, a događali su se i požari koji su znali uništiti cijele arhive, a ponekad i cijelu povijest nekog društava, organizacije ili vlade. Bilo je tu i sigurnosnih problema jer je fizički pristup zapisima bilo vrlo jednostavno dobiti.



Slika 1.1: Organizacija povijesnih baza podataka
Izvor: http://avant.org/project/history-of-databases/

Baza podataka stvorena je kako bi pokušala riješiti ta ograničenja tradicionalnog pohranjivanja podataka na papiru. Charles Bachman dizajnirao je prvu računalnu bazu podataka početkom 1960-ih. Ova prva baza podataka bila je poznata pod nazivom "Integrated Data Store" (IDS). Nedugo nakon toga IBM je odgovorio i objavio svoju bazu pod nazivom "Information Management System". Obje baze podataka bile su preteča navigacijske baze podataka i od korisnika su zahtijevale da se kreću kroz cijelu bazu podataka kako bi došli do željene informacije. Postojala su dva glavna modela navigacijskih baza podataka: hijerarhijski model i mrežni model.

Edgar F. Codd, matematičar školovan u Oxfordu, 1970.-ih godina objavljuje rad u kojemu objašnjava kako se može pristupiti informacijama pohranjenim u velikim bazama podataka, a da se ne zna kako su informacije strukturirane ili gdje se nalaze u bazi podataka. Njegov rad, "A Relational Model of Data for Large Shared Data Banks", informatičari su prozvali revolucionarnom idejom.

Narednih deset godina obilježio je rast i popularizacija baza podataka. Prevladao je relacijski model baza podataka, dok su se navigacijski modeli gotovo u potpunosti prestali koristiti. Također osamdesetih godina je SQL, o kojemu će biti rečeno nešto više u nastavku rada, postao standardni jezik koji se koristi prilikom izrade baze podataka, a u upotrebi se zadržao sve do danas.

Sredinom 80-ih se događa još jedan vrlo važan događaj u povijesti baza podataka, predstavljen je objektno orijentiran sustav upravljanja bazama podataka (OODBMS). U objektnim bazama podataka, podaci su predstavljeni objektima te se koriste programski jezici koji podržavaju objektno orijentirani pristup.

Pojava World Wide Weba 90-ih godina bio je jedan od ključnih događaja u povijesti baza podataka. Velika ulaganja u internetska poduzeća potaknula su potražnju za sustavima baza podataka klijent-poslužitelj. Kao takav, internet je pomogao snažnom eksponencijalnom rastu industrije baza podataka u 1990-ima. Značajan ishod toga bilo je stvaranje MySQL-a 1995. godine.

NoSQL (engl. *Not only SQL*) baze podataka nastaju 1998. godine. Odnose se na baze podataka koje za pohranu i dohvaćanje podataka ne koriste nužno SQL za jezik upita. NoSQL baze podataka korisne su za nestrukturirane podatke, a postale su vrlo popularne u 2000-ima. Ovo je značajan razvoj u povijesti baza podataka jer je NoSQL omogućio bržu obradu većih i raznovrsnijih skupova podataka. NoSQL baze podataka fleksibilnije su od tradicionalnih relacijskih baza podataka koje su nastale desetljeće prije.

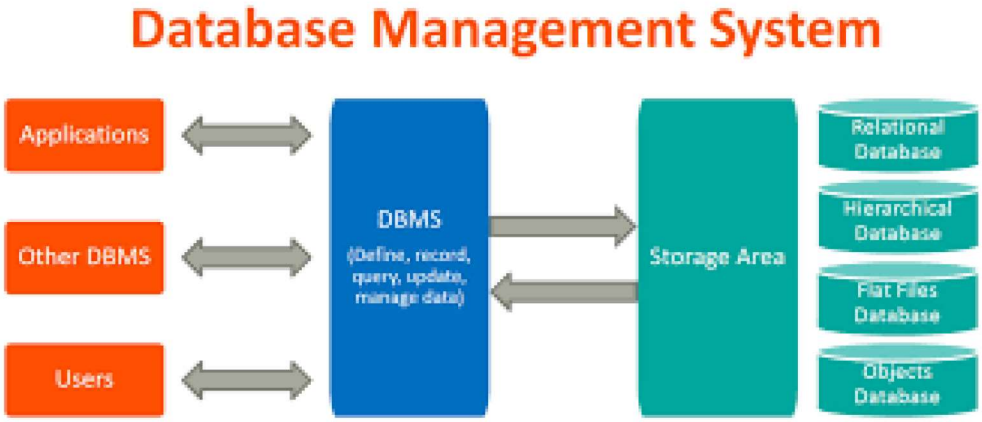
Od 2000. godine do danas SQL i sustavi za upravljanje podacima su se popularizirali i standardizirali. Baze podataka su danas svugdje i koriste se za poboljšanje svakodnevnog života. Mnoge usluge koje danas koristimo moguće su zahvaljujući bazama podataka. I dalje se najviše koriste relacijske baze podataka, ali popularnost NoSQL baza podataka je sve veća. [6]

2 Sustav za upravljanje bazom podataka

Sustav za upravljanje bazom podataka (DBMS) je programski sustav koji omogućava upravljanje podacima u bazi podataka. DBMS se temelji na odabranom modelu podataka. Osnovne zadaće koje se mora obavljati svaki DBMS su:

- zaštita objekata baza podataka od neovlaštenog korištenja,
- očuvanje integriteta podataka u bazi podataka,
- omogućavanje obnove podataka različitim načinima u slučaju gubitka podataka,
- omogućavanje konkurentnosti, tj. omogućavanje višekorisničkog pristupa istim podacima u bazi podataka istovremeno,
- omogućavanje opisa podataka metapodacima,
- identificiranje optimalne strukture za najprikladnije upravljanje podacima,
- opis i rukovanje podacima. [2]

Danas postoji nekoliko važnih i široko zastupljenih sustava za upravljanje bazom podataka, a najpoznatiji su: Microsoft SQL Server, MySQL, PostgreSQL, Oracle, Sybase i IBM DB2.



Slika 2.1: Sustav za upravljanje bazom podataka
Izvor: https://www.finahost.co.ke/2019/02/27/database-management-systems/

2.1 Modeli baze podataka

U bazi podataka podaci su logički organizirani prema nekom modelu. Model čini osnovu za osmišljavanje, definiranje i implementiranje baze podataka. Današnji DBMS-ovi podržavaju četiri osnovna modela:

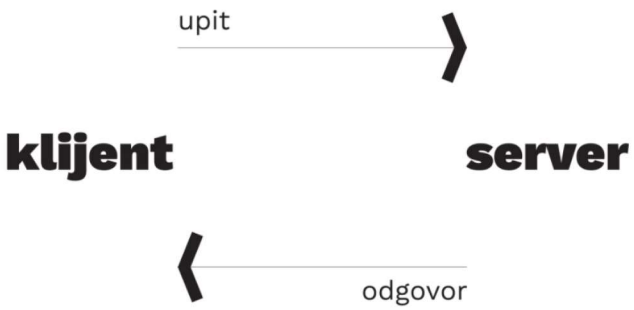
- **Relacijski model** - Ovaj je model zasnovan na matematičkom pojmu relacije. Podaci i veze među podacima prikazuju se tablicama koje imaju retke i stupce. U današnje vrijeme ovo je najzastupljeniji model, a upravo je ovaj model korišten prilikom izrade projektnog dijela ovog rada.
- **Mrežni model** - U ovom modelu baza je predočena mrežom koja se sastoji od čvorova i usmjerenih lukova. Čvorovi predstavljaju tipove zapisa (slogova podataka), a lukovi definiraju veze među tipovima zapisa.
- **Hijerarhijski model** - Model u kojemu je baza predočena jednim stablom (hijerarhijom) ili skupom stabala. Svako stablo sastoji se od čvorova i veza „nadređeni-podređeni“ između čvorova. Čvorovi su tipovi zapisa, a odnos „nadređeni-podređeni" izražava hijerarhijske veze među tipovima zapisa.
- **Objektni model** - Ovaj je model inspiriran objektno-orijentiranim programskim jezicima. Baza je predočena kao skup trajno pohranjenih objekata koji se sastoje od svojih internih „atributa“ (podataka) i „metoda“ (operacija) za rukovanje tim podacima. Svaki objekt pripada nekoj klasi. Između klasa se uspostavljaju veze nasljeđivanja, agregacije i druge vrste veza. [3]

2.2 MySQL

MySQL je jedan od najraširenijih, relacijskih sustava otvorenog koda za upravljanje bazom podataka. Pokreće se na poslužitelju, te podržava višekorisnički pristup bazama podataka. Izradila ga je švedska tvrtka MySQL AB 1995. godine. Osnivači tvrtke su Michael Widenius, David Axmark i Allan Larsson. MySQL je ime dobio po kćeri jednog od suosnivača Montya Wideniusa, koja se zove My. [8]

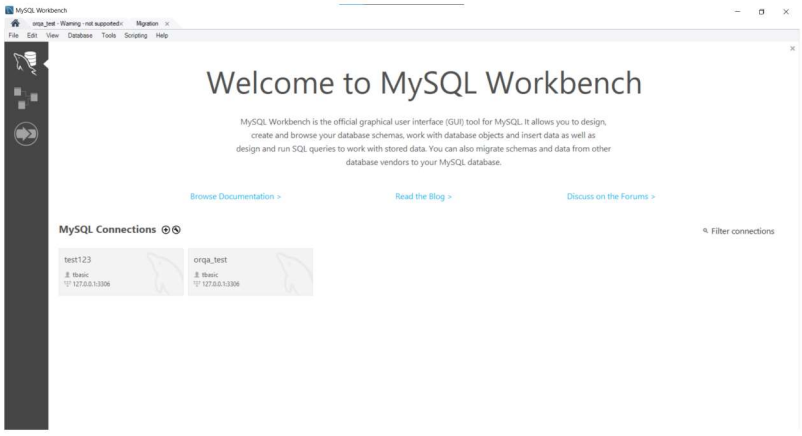
Slika 2.2: MySQL logo
Izvor: https://itsilesia.com/6-tips-that-every-mysql-user-should-know/

MySQL funkcionira po modelu klijent-server. Računala koja imaju instaliran neki RDBMS softver (kao što je MySQL) se zovu klijenti. RDBMS serveru pristupaju kada im treba pristup podacima. To je princip rada klijent-server modela. Nekoliko je procesa kojima se definira način rada unutar MySQL okruženja. Prvi je kreiranje baze podataka. Zatim slijedi unos podataka u bazu, a zadnji proces je vezan za odnos klijenta i baze podataka. Klijent zadaje upite serveru pomoću SQL jezika, a server na kraju procesa vraća željene podatke koji se onda prikazuju klijentu. [10]



Slika 2.3: Prikaz principa rada modela klijent-server
Izvor: https://lcetinic1611.wordpress.com/2019/12/09/mysql-seminarski-rad/

Jedan od alata koji dolazi uz MySQL, a korišten je za izredu ER modela u projektnom dijelu rada, je MySQL Workbench. MySQL Workbench je grafički alat za dizajniranje baza podataka koji integrira SQL razvoj, administraciju, dizajn i održavanje u jedno zajedničko sučelje za MySQL baze podataka. Prva ogledna inačica je razvijena 2005. godine i nasljednik je DBDesigner4 alata. MySQL Workbench postao je dio MySQL paketa 2007. godine. Inačica 5.0, koja je ujedno i prva inačica ovog sustava, je bila namijenjena za upotrebu na MS Windows operacijskom sustavu, a od inačice 5.1 dostupna je i na drugim platformama. Alat postoji u besplatnom i komercijalnom izdanju. MySQL Workbench je drugi alat po preuzimanjima, sa web stranice MySQL, s preko 250.000 mjesečnih preuzimanja. [9]



Slika 2.4: Početna forma MySQL Workbench-a

3 Razvojni ciklus baze podataka

Ključni dio razvoja baze podataka smatra se projektiranje baze podataka. Izradu baze podataka možemo podijeliti u nekoliko faza, a to su: utvrđivanje i analiza zahtjeva, projektiranje, implementacija, testiranje i na kraju održavanje.

3.1 Utvrđivanje i analiza zahtjeva

Kako bi se utvrdili zahtjevi potrebno je proučiti tok informacija i proizvodni proces nekog poduzeća. Također, potrebno je proučiti dokumentaciju sustava koji su trenutno u opticaju, radne procese, potrebno je razgovarati s korisnicima i proučiti postojeći softver. U velikim poduzećima, gdje postoje razne skupine korisnika odnosno razni timovi, pojavit će se različita tumačenja i svrha neki podataka i različiti načini uporabe tih istih podataka. Analiza zahtjeva služi nam da te razlike dovedemo "na istu valnu duljinu", te da se izbjegnu redundancije i nezakonitosti. Rezultat ove faze projektiranja baze podataka je dokument koji nazivamo specifikacija i on je obično pisan neformalno u prirodnom jeziku. [3]

3.2 Projektiranje

Cilj projektiranja je da se obzirom na specifikacije, izrađene u prethodnoj fazi, oblikuje građa baze podataka. Analiza je dala uvid o vrsti podataka koje baza treba sadržavati i što će se s njima moći raditi. Projektiranje to dovodi na potpuno novu dimenziju predlaganjem pogodnih načina za grupiranje, strukturiranje i međusobno povezivanje podataka. Glavni rezultat projektiranja je shema cijele baze, oblikovana u skladu s pravilima korištenog modela podataka i zapisana tako da ju korišteni DBMS može razumjeti i realizirati. Projektiranje baze podataka složena je aktivnost i zbog toga se dijeli u tri faze:

- **Projektiranje na konceptualnoj razini** - Rezultat prve faze projektiranja je model entiteta i veza, odnosno konceptualna shema cijele baze sastavljena od entiteta, atributa i veza. Model entiteta i veza opisuje sadržaj baze i načine kojima su podaci povezani u njoj. Prikaz ovog modela je neformalan i lako je razumljiv ljudima, ali je i dalje nedovoljno razrađen da bi omogućio izravnu implementaciju.
- **Projektiranje na logičkoj razini** - Kao rezultat druge faze projektiranja nastaje logička shema, koja je u slučaju relacijskog modela sastavljena od relacija (tablica).

- **Projektiranje na fizičkoj razini** - Rezultat treće faze projektiranja je fizička shema cijele baze koja je zapravo opis njezine fizičke građe. U slučaju korištenja DBMS-a zasnovanog na jeziku SQL, fizička shema je zapravo niz SQL naredbi kojima se relacije iz logičke sheme realiziraju kao tablice u sustavu. Prilikom kreiranja dodaju se pomoćne strukture i mehanizmi kojima se postižu tražene preformanse i čuva se integritet baze i sigurnost podataka. [3]

3.3 Implementacija

Implementacija predstavlja fizičku realizaciju projektirane baze. U slučaju RDBMS-a zasnovanog na SQL jeziku, implementacija se odnosi na pokretanje skripte koja sadrži SQL naredbe te se na disku stvaraju tablice sa svim ograničenjima, strukturama i mehanizmima. Idući korak je popuniti te tablice početnim podacima. Nakon što su početni podaci uneseni u bazu, razvijaju se aplikacije koje obavljaju najvažnije transakcije s podacima. Nakon toga je sve spremno za iduću fazu projektiranja-testiranje. [3]

3.4 Testiranje

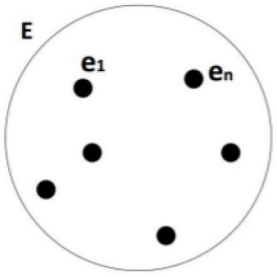
Testiranje se provodi tako da korisnici rade s bazom i provjeravaju udovoljava li ona svim zadanim zahtjevima. Prati se učinak baze te se mjere njene preformanse. Također od izrazite je važnosti provjeriti stabilnost i pouzdanost rada baze pod velikim opterećenjem. Prilikom testiranja potrebno je pronači sve greške koje su se potkrale u nekoj od prethodnih faza. Ako se pogreške iz ranijih faza ne otkriju prilikom testiranja nastaju teže posljedice. [3]

3.5 Održavanje

Održavanje je kontinuirani proces u kojemu se baza i dokumentacija neprestano mijenjaju. Ova faza se odvija u vrijeme kada se baza već redovito koristi. Održavanje baze će biti lagano i bezbolno ako je baza od početka projektiranja imala zdravu strukturu i ako su se problemi otkrili u fazi testiranja. [3]

4 Modeliranje baze podataka

"Entitet (engl. *Entity*) je skup objekata iz stvarnog svijeta koji imaju naglašena zajednička svojstva. Svaki entitet ima svojstva koja ga opisuju i nazivaju se atributima. Entitet je definiran kao skup $E = \{e_1, e_2, ..., e_n\}$, gdje su $e_1, ..., e_n$ elementi entiteta" (Carić i Buntić, 2015, [2]).



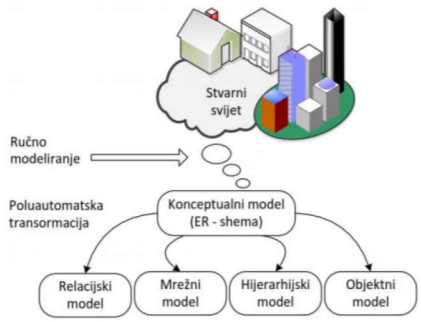
Slika 4.1: Prikaz entiteta kao skupa
(Carić i Buntić, 2015, [2])

Kao primjer imamo entitet ZAPOSLENIK, a njegovi atributi su: ime, prezime, email adresa, broj mobitela, broj RFID kartice i dr.

Veza je nešto što povezuje dva entiteta.

Kako bi modelirali stvarni svijet i napravili odgovarajuću bazu podataka potrebno je napraviti model entiteta i veza. Razvijeni konceptualni model dalje se pretvara u relacijski ili neki drugi model. Modeliranje entiteta i veza zahtijeva da svijet promatramo preko tri kategorije:

- entiteti: stvari, bića, pojave ili događaji koji su nam od interesa,
- veze: odnosi među entitetima koji su nam od interesa,
- atributi: svojstva entiteta ili veza koja su nam od interesa.[2]

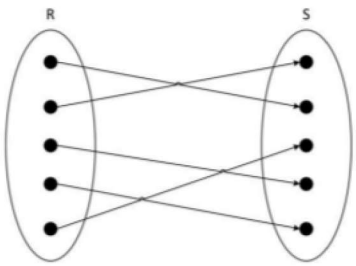


Slika 4.2: Modeliranje podataka
(Carić i Buntić, 2015, [2])

U projektnom dijelu ovog rada su korištene tzv. binarne veze. Binarne veze su veze između dva entiteta, a postoje tri vrste binarnih veza. To su: 1:1, 1:N i M:N.

Veza 1:1

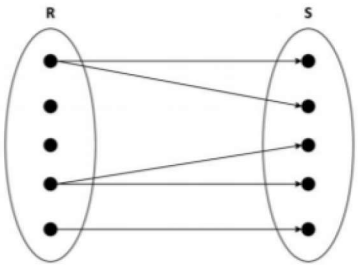
Svaki element skupa R može biti povezan sa samo jednim ili nijednim elementom skupa S. Isto vrijedi za elemente skupa S. [2]



Slika 4.3: Prikaz veze jedan naprema jedan (1:1)
(Carić i Buntić, 2015, [2])

Veza 1:N

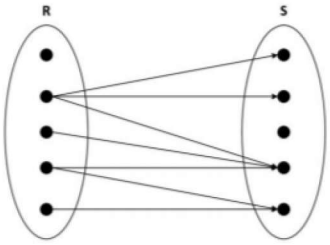
Svaki element skupa R može biti povezan nijednim, jednim ili više elemenata skupa S, dok svaki element skupa S može biti povezan sa samo jednim elementom skupa R. [2]



Slika 4.4: Prikaz veze jedan naprema više (1:N)
(Carić i Buntić, 2015, [2])

Veza M:N

Svaki element skupa R može biti povezan jednim ili više elemenata skupa S. Isto vrijedi za elemente skupa S". [2]



Slika 4.5: Prikaz veze više naprema više (M:N)
(Carić i Buntić, 2015, [2])

Proces oblikovanja baze podataka iz korisničkih zahtjeva nije formalno definiran te zahtjeva kreativnost i individualno se razlikuje od osobe do osobe.

4.1 Relacijski model i osnove relacijske algebre

Kod opisivanja relacijskog modela mogu se koristiti različiti termini. U tablici 1 prikazani su matematički nazivi elemenata relacijskog modela te odgovarajući tradicionalni nazivi relacijskog modela koji se nadalje koriste u tekstu.

Matematički naziv	**Tradicionalni naziv**
Relacija	Tablica
N-torka	Redak ili red
Atribut	Stupac ili kolona
Vrijednost atributa	Pojedinačni podatak
Veza	Veza ili relacija

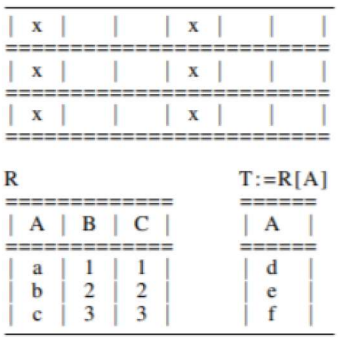
Tablica 4.1: Terminologija relacijskog modela baza podataka, (Carić i Buntić, 2015, [2])

E. F. Codd je uveo pojam relacije i osnove relacijske algebre sredinom 70-ih godina. Prema Coddu relacijska algebra podrazumijeva definirane operacije nad entitetima (tablicama) i podacima koji im pripadaju.

4.1.1 Prirodne relacijske operacije

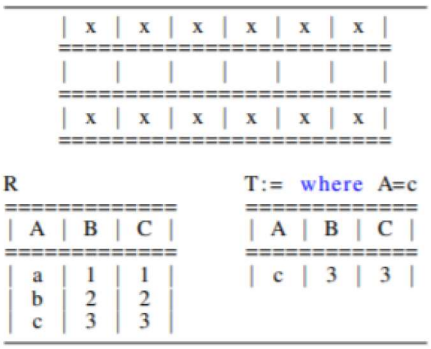
Prirodnim relacijskim operacijama pripadaju: projekcija ($T := R[a]$), selekcija ($T := R$ gdje je $a = 12$), spajanja (inner join, left/right outer join,...) te dijeljenje ($T := R \div S$).[2]

Projekcija ($T := R[a]$)
Operacijom projekcije tablice nad atributima izdvajaju se atributi tablice na kojima se vrši projekcija. Projekcija tablice R nad atributima A, B, C jest tablica T sa zaglavljem(T) = A, B, C koja sadrži sve redove koji su sadržani u tablici R. Rezultat projekcije je nova tablica koja predstavlja vertikalni podskup zadane tablice.[2]



Slika 4.6: Primjer $T := R[a]$
(Carić i Buntić, 2015, [2])

Selekcija ($T := R$ **gdje je** $a = 12$)
Operacijom selekcije nad zadanom tablicom R izdvaja se skup redova koji zadovoljavaju uvjet po kojem se selekcija vrši. Rezultat operacije selekcije je tablica koja sadrži sve atribute izvorne tablice, ali samo one redove koji zadovoljavaju traženi uvjet. Dobivena tablica predstavlja horizontalni podskup izvorne tablice.[2]



Slika 4.7: Primjer $T := R$ gdje je $a = 12$
(Carić i Buntić, 2015, [2])

Spajanja
Operacija spajanja ima nekoliko podvrsta. Te podvrste su navedene i ukratko objašnjene u nastavku.

Inner join ($T := R \bowtie S$)
Inner join operacijom povezuju se tablice na način da se spajaju redovi tablica po istim vrijednostima zajedničkog atributa, tj. spajaju se redovi koji u stupcima istog naziva u obje tablice imaju istu vrijednost.[2]

R

A	B	D
a	1	1
b	2	2
c	3	3

S

D	E	F
2	2	2
3	3	3
4	4	4

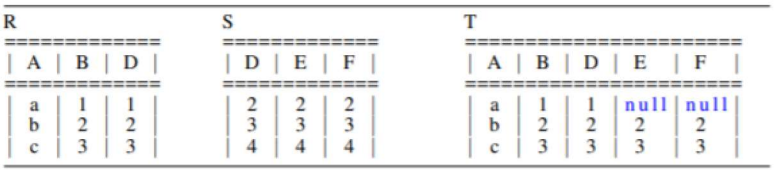
T

A	B	D	E	F
b	2	2	2	2
c	3	3	3	3

Slika 4.8: Primjer $T := R \bowtie S$
(Carić i Buntić, 2015, [2])

U navedenom primjeru tablice R i S imaju zajednički atribut D, te se spajanje odvija po tom atributu. Prvi red tablice R ima vrijednost atributa D = 1, te ne sudjeluje u vezi, jer u tablici S ne postoji red sa D = 1. Drugi i treći red tablice R imaju vrijednost D = 2 i D = 3, te se spajaju sa prvim i drugim redom u tablici S.[2]

Left outer join ($T := R \bowtie_{LO} S$)
Lijeva vanjska veza je proširenje unutrašnje veze. Osnova za realizaciju lijeve vanjske veze je unutrašnja veza, kojoj se dodaju elementi tablice koji ne sudjeluju u vezi. Tablica čiji se elementi dodaju je ona koja je u relacijskom izrazu sa lijeve strane.[2]



R

A	B	D
a	1	1
b	2	2
c	3	3

S

D	E	F
2	2	2
3	3	3
4	4	4

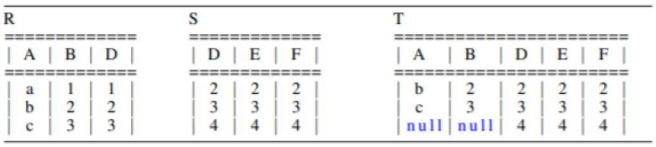
T

A	B	D	E	F
a	1	1	null	null
b	2	2	2	2
c	3	3	3	3

Slika 4.9: Primjer $T := R \bowtie_{LO} S$
(Carić i Buntić, 2015, [2])

U prethodnom primjeru objašnjeno je nastajanje tablice $R \bowtie S$. U stvaranju $R \bowtie_{LO} S$ najprije se realizira $R \bowtie S$ (time se dobiju drugi i treći redak). Potom se dodaju svi redci tablice R, koji ne sudjeluju u unutrašnjoj vezi. U ovom slučaju jedini redak u tablici R koji nije obuhvaćen unutrašnjom vezom je prvi redak. Budući da on ne sudjeluje u unutrašnjoj vezi atributi E i F su null vrijednosti.[2]

Right outer join ($T := R \bowtie_{RO} S$)
Desna vanjska veza je proširenje unutrašnje veze. Osnova za realizaciju desne vanjske veze je unutrašnja veza, kojoj se dodaju oni elementi tablice koji ne sudjeluju u unutrašnjoj vezi. Tablica čiji se elementi dodaju je ona koja je u relacijskom izrazu sa desne strane.[2]



R		
A	B	D
a	1	1
b	2	2
c	3	3

S		
D	E	F
2	2	2
3	3	3
4	4	4

T				
A	B	D	E	F
b	2	2	2	2
c	3	3	3	3
null	null	4	4	4

Slika 4.10: Primjer $T := R \bowtie_{RO} S$
(Carić i Buntić, 2015, [2])

Outer join ($T := R \bowtie_O S$)
Vanjska veza je proširenje unutrašnje veze. Osnova za realizaciju lijeve vanjske veze je unutrašnja veza, kojoj se dodaju elementi obje tablice koji ne sudjeluju u unutrašnjoj vezi. Na ovaj način vrijedi da je $T := R \bowtie_{RO} S = R \bowtie_{LO} S \cup R \bowtie_{RO} S$.[2]

R		
A	B	D
a	1	1
b	2	2
c	3	3

S		
D	E	F
2	2	2
3	3	3
4	4	4

T				
A	B	D	E	F
a	1	1	null	null
b	2	2	2	2
c	3	3	3	3
null	null	4	4	4

Slika 4.11: Primjer $T := R \bowtie_O S$
(Carić i Buntić, 2015, [2])

Dijeljenje $T := R \div S$
Tablica T koja se dobije dijeljenjem R i S je najveća tablica za koju vrijedi da se svi redovi produkta $T \times S$ nalaze u tablici R.[2]

R		
A	B	D
a	1	1
b	2	2
c	3	3

S		
D	E	F
2	2	2
3	3	3
4	4	4

T				
A	B	D	E	F
a	1	1	null	null
b	2	2	2	2
c	3	3	3	3
null	null	4	4	4

Slika 4.12: Primjer $T := R \div S$
(Carić i Buntić, 2015, [2])

4.1.2 Logičke operacije

U relacijskoj algebri osim prirodnih relacijskih operacija koriste se često i logičke operacije i to posebno kod primjene operacije selekcije, odnosno kod složenih upita nad bazom. Logički operatori su AND, OR i NOT. Operatori AND i OR su funkcije koje se primijenjuju nad dva argumenta, a funkcija NOT nad jednim argumentom.[2]

AND			OR			NOT	
T	T	T	T	T	T	F	T
F	T	F	T	T	F	T	F
F	F	T	T	F	T		
F	F	T	F	F	F		

Slika 4.13: Logičke operacije
(Carić i Buntić, 2015, [2])

5 Strukturni jezik upita (SQL)

SQL je standardizirani i najrašireniji jezik za rad s bazama podataka koji omogućuje kreiranje baza podataka, manipuliranje podacima te dohvaćanje podataka iz baze. Sama skraćenica SQL dolazi od eng. Structured Query Language, što prevodimo kao strukturirani jezik upita.

SQL se sastoji od određenog broja naredbi, a svaka naredba se sastoji od određenih dijelova (klauzula). SQL naredbe se dijele u slijedeće grupe:

- DDL (engl. Data Definition Language) naredbe
- DML (engl. Data Manipulation Language) naredbe
- DQL (engl. Data Query Language) naredbe
- DCL (engl. Data Control Language) naredbe

DDL naredbe su naredbe za kreiranje objekata. U DDL pripadaju naredbe CREATE, ALTER i DROP. Spomenute naredbe omogućuju kreiranje, mijenjanje ili brisanje postojećih objekata.

DML naredbe su naredbe za manipuliranje podacima. U DML pripadaju naredbe INSERT, UPDATE i DELETE, ali u novije vrijeme i naredbe MERGE.

DQL naredbe odnosno naredbe za postavljanje upita. Osnovna nareba za zadavanje upita je naredba SELECT.

DCL grupi naredba pripadaju naredbe GRANT i REVOKE koje se uglavnom bave pravima, dozvolama i drugim kontrolama sustava baze podataka.[4]

5.1 Tipovi podataka

Prilikom kreiranja tablice i dodavanja stupca u tablicu, za svaki stupac potrebno je definirati naziv stupca i odrediti mu tip podatka. Ispravan odabir tipa podataka izravno utječe na potrebnu količinu mjesta za pohranu. U SQL 2008 standardu spominju se slijedeći tipovi:

CHARACTER	BIGINT
CHARACTER VARYING	FLOAT
CHARACTER LARGE OBJECT	REAL
BINARY	DOUBLE PRECISION
BINARY VARYING	BOOLEAN
BINARY LARGE OBJECT	DATE
NUMERIC	TIME
DECIMAL	TIME STAMP
SMALLINT	INTERVAL
INTEGER	

Tablica 5.1: Tipovi podataka u SQL standardu, (Rabuzin, 2011, [4])

Svaki tip podatka podržava i tzv. NULL znak. NULL unosimo kada vrijednost stupca nije poznata ili definirana.

Sustav za upravljanje bazom podataka MySQL podržava sljedeće numeričke tipove:

Naziv	**Vrijednost**
BIT	-128 do 127 normal 0 do 255 UNSIGNED
TINYINT()	Ekvivalentan tipu podataka BIT.
SMALLINT()	-32768 do 32767 normal 0 do 65535 UNSIGNED
MEDIUMINT()	8388608 do 8388607 normal 0 do 16777215 UNSIGNED
INT() / INTEGER()	2147483648 do 2147483647 normal 0 do 4294967295 UNSIGNED
BIGINT()	-9223372036854775808 do 9223372036854775807 normal 0 do 18446744073709551615 UNSIGNED
DECIMAL(,) / DEC (,)	DOUBLE spremljen kao string i to točno određene duljine
FLOAT	Manji broj s pomičnim zarezom
DOUBLE(,)	Veći broj s pomičnim zarezom

Tablica 5.2: Numerički tipovi podržani u sustavu MySQL, (Mićić, 2009, [11])

Kod cijelih brojeva nakon definiranja tipa podatka, može se napisati UNSIGNED. Na taj se način definirani numerički tip umjesto u raspon od negativnog do pozitivnog prebacuje u pozitivni raspon koji započinje nulom.

Naziv	Opis
CHAR	String fiksne dužine od 0 do 255 znakova.
VARCHAR	String promjenjive dužine od 0 do 255 znakova.
TINYTEXT	String maksimalne dužine 255 znakova.
BLOB	String maksimalne dužine 65535 znakova.
MEDIUMTEXT	String maksimalne dužine 16777215 znakova.
MEDIUMBLOB	String maksimalne dužine 16777215 znakova.
LONGTEXT	String maksimalne dužine 4294967295 znakova.
LONGBLOB	String maksimalne dužine 4294967295 znakova.

Tablica 5.3: Tekstualni tipovi podržani u sustavu MySQL, (Mićić, 2009, [11])

Kada za neki atribut definiramo tip podatka kao CHAR ili VARCHAR potrebno je u zagradi dodati broj koji označava maksimalan broj znakova koji se mogu upisati na mjesto tog atributa. Tako atribut JMBG tipa VARCHAR(13) znači da se može upisati 13 znakova u svaku ćeliju u stupcu JMBG. Važno je definirati dovoljan broj znakova, ali treba paziti da se ne definira ni previše ni premalo. Ako je definiran broj znakova prevelik, zauzima se prostor u memoriji bez razloga i na taj način se usporava rad baze podataka. S druge strane ako je definiran nedovoljan broj znakova događaju se situacije da netko ima predugačku adresu i sl., a nema dovoljno mjesta za unos.

Naziv	Opis
DATE	YYYY-MM-DD
DATETIME	YYYY-MM-DD HH:MM:SS
TIMESTAMP	YYYY-MM-DD HH:MM:SS
TIME	HH:MM:SS

Tablica 5.4: Tipovi za datum i vrijeme podržani u sustavu MySQL, (Mićić, 2009, [11])

Razlika između TIMESTAMP I DATETIME je što DATETIME poprima vrijednosti od '1000-01-01 00:00:00' do '9999-12-31 23:59:59' dok TIMESTAMP poprima vrijednosti od '1970-01-01 00:00:01' UTC do '2038-01-19 03:14:07' UTC.

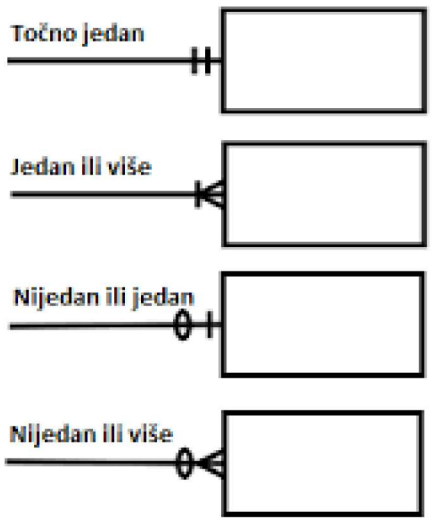
6 Projektiranje baze podataka za kvalitetu proizvodnog procesa "Orqa"

Projektni dio ovog rada obuhvaćao je izradu baze podataka za prikupljanje podataka o kvaliteti proizvodnog procesa. Prilikom izrade baze koristile su se zakonitosti izrade relacijskih baza podataka, sustav za upravljanje bazom podataka MySQL i XAMPP - open source sustav za jednostavnu instalaciju Apache servera na računala.

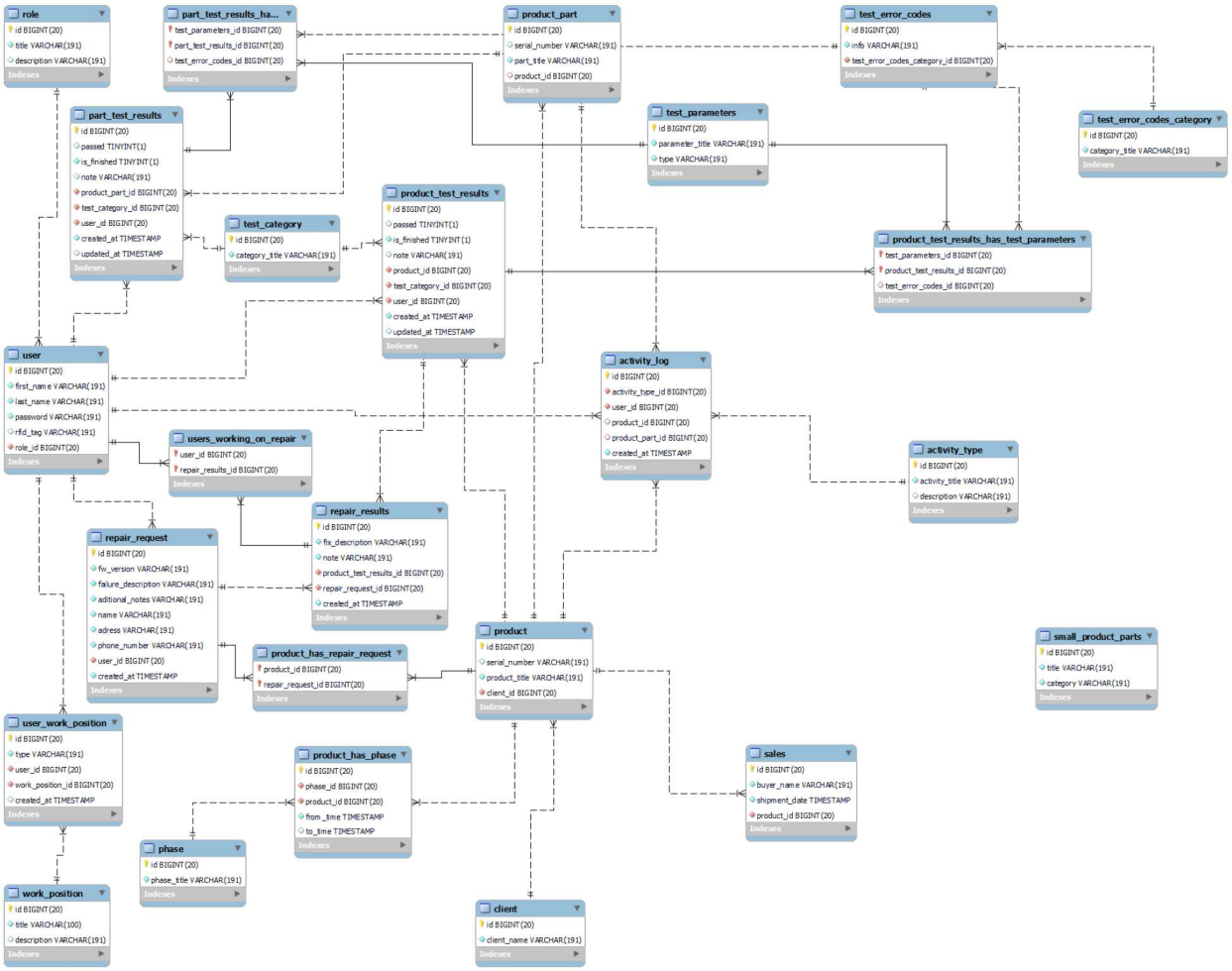
6.1 ER model

Prije same izrade ER modela (engl. *Entity-relationship model*) bilo je potrebno proučiti proizvodni proces tvrtke Orqa d.o.o. i dobro analizirati korisničke zahtjeve. Na osnovu tih zahtjeva bilo je potrebno kreirati ER model.

Model sa slike 6.2. izrađen je u grafičkom alatu MySQL Workbench. Sastoji se od 25 entiteta, svaki entitet se sastoji od atributa, a entiteti su međusobno povezani vezama. Veze se mogu prikazati različitim notacijama. Notacija koja je korištena u ER modelu sa slike 6.2 naziva se "Crow's foot" notacija. Ovu notaciju je osmislio C. Finkelstein, a naziv "Crow's foot" je dobila zbog izgleda kardinalnosti "više" koja podsjeća na vranino stopalo.



Slika 6.1: Veze u notaciji "Crow's foot"
(Kemić, 2013, [12])



Slika 6.2: ER model baze podataka za "Orqa"

6.2 Kreiranje strukture baze podataka

Kako bi se podaci mogli pohraniti u bazu podataka, najprije ju je potrebno kreirati. U ovom poglavlju obrađene su teme kreiranja baze podataka, pripadajućih tablica te postavljanje ograničenja na definirane stupce unutar tablica. Sve je popraćeno primjerima iz projekta.

6.2.1 Kreiranje baze podataka

Kreiranje baze podataka u sustavu za upravljanje bazom podataka MySQL izvršava se naredbom CREATE DATABASE.[2] Na primjeru 1 može se vidjeti primjer kreiranja baze orqa_db koja nema neke dodatne postavke. Izvršavanjem naredbe kreirana je baza podataka naziva orqa_db. Osim navedenog načina, baza podataka može se kreirati i preko grafičkog sučelja MySQL Workbencha.

```
01 | CREATE DATABASE orqa_db;
```

Primjer 1: Kreiranje baze podataka

Brisanje baze podataka izvršava se naredbom DROP DATABASE. DROP naredba se također koristi za brisanje bilo kojeg strukturnog objekta u bazi, dok naredba DELETE briše podatke.[2] Na primjeru 2 može se vidjeti brisanje baze podataka. Treba znati da DBMS automatski nepovratno briše bazu podataka sa svim podacima i shemom nakon izvršavanja naredbe za brisanje baze podataka.

```
01 | DROP DATABASE orqa_db;
```

Primjer 2: Brisanje baze podataka

6.2.2 Integritet baze podataka, kreiranje i brisanje tablica

Prilikom izrade baze podataka važno je da su podaci u bazi konzistentni i korektni. Nekorektni podaci mogu izazvati niz problema. Očuvanje integriteta baze podataka postižemo postavljanjem određenih ograničenja. Ograničenja definiraju pravila koja dopuštaju unos samo valjanih vrijednosti u stupce. Osim za očuvanje integriteta baze podataka ona imaju važnu ulogu u kreiranju planova izvršavanja upita, što pridonosi boljim performansama baze podataka. Neka od najčešće korištenih ograničenja su:

NULL ili NOT NULL ograničenje sprječava unos NULL vrijednosti u stupac koji je definiran kao NOT NULL.

CHECK omogućuje definiranje uvjeta kojeg mora zadovoljavati vrijednost stupca.

UNIQUE ograničenjem sprječava se unos duplih vrijednosti u jednom stupcu, tj. ograničenje se koristi kada je potrebno da svi redci jednog stupca imaju različitu vrijednost.

DEFAULT omogućava da stupac sa ovim ograničenjem tijekom unosa vrijednosti poprimi definiranu početnu vrijednost ako se ne unese neka druga vrijednost.

PRIMARY KEY je jedan atribut ili kombinacija atributa koji jedinstveno identificira svaki redak u tablici.

FOREIGN KEY je jedan atribut ili kombinacija atributa koji se koriste za uspostavljanje veza izmedu tablica.

Postavljanje većeg broja ograničenja tijekom kreiranja strukture baze podataka omogućava kasnije jednostavnije održavanje baze podataka.[2]

Ova ograničenja postavljamo prilikom kreiranja tablice. Tablice u sustavu za upravljanje bazom podataka MySQL kreiramo naredbom CREATE TABLE.

```
CREATE TABLE `users`(
    `id` BIGINT(20) unsigned NOT NULL AUTO_INCREMENT,
    `first_name` VARCHAR(191) NOT NULL,
    `last_name` VARCHAR(191) NOT NULL,
    `password` VARCHAR(191) NOT NULL,
    `rfid_tag` VARCHAR(191) NULL DEFAULT NULL,
    PRIMARY KEY (`id`)
);
```

Primjer 3: Kreiranje tablice ‘users‘

Na primjeru 3 prikazana je naredba kojom je kreirana tablica ‘users‘. Tablica ima 5 atributa. Atribut ‘id‘ je tipa BIGINT(20), on je jedinstveni identifikator ove tablice što je omogućeno postavljanjem ograničenja PRIMARY KEY, također on ne može poprimiti NULL vrijednosti. AUTO_INCREMENT naredba koristi se za automatsko generiranje inkrementalnih brojeva za svaki redak - prvi će redak biti numeriran 1, drugi redak 2 itd. Atributi ‘first_name‘, ´last_name´ i ‘password‘ su atributi koji kao ograničenje imaju NOT NULL što znači da vrijednosti moraju biti upisane i ne mogu imati NULL vrijednost. Posljednji atribut ‘rfid_tag‘ može biti NULL vrijednost i definirana mu je početna vrijednost kao NULL.

Brisanje tablice kao i svakog objekta u bazi podataka radi se naredbom DROP. U primjeru 4 briše se tablica ‘users‘. Brisanje je moguće ako se ne narušava referencijalni integritet u protivnom tablica se neće obrisati i DBMS će javiti pogrešku.

```
DROP TABLE users;
```

Primjer 4: Brisanje tablice users

Za mijenjanje strukture tablice odnosno operacije dodavanje, brisanja ili modifikacije stupca na postojećoj tablici koristi se naredba ALTER TABLE.

Na primjeru 5 prikazano je dodavanje stupca ‘phone_number‘ koji je tipa VARCHAR(10) i koji je obavezan za unos.

```
ALTER TABLE users
ADD phone_number VARCHAR(10) NOT NULL;
```

Primjer 5: Dodavanje novog stupca u postojeću tablicu

Također ako je potrebno može se obrisati određeni stupac iz postojeće tablice, a to je prikazano na primjeru 6.

```
ALTER TABLE users
DROP COLUMN phone_number;
```

Primjer 6: Brisanje stupca iz postojeće tablice

6.3 Modifikacija podataka

Jednom kada su tablice kreirane, one su prazne. Da bi u tablicu upisali podatke ili promijenili vrijednosti nekih stupaca koristimo naredbe INSERT, UPDATE i DELETE. INSERT i DELETE rade na razini retka dok UPDATE mijenja vrijednosti na razini stupca.

Za dodavanje podataka u već postojeću tablicu koristi se naredba INSERT.

```
01 | INSERT INTO table_name (column1, column2, column3, ...)
02 | VALUES (value1, value2, value3, ...);
```

Primjer 7: Osnovna sintaksa INSERT naredbe za dodavanje više zapisa

Prilikom dodavanja podataka u tablicu treba obratiti pozornost na ograničenja zadana prilikom kreiranja tablice. Također treba provjeriti narušava li se referencijalni integritet stranog ključa.

```
01 | INSERT INTO users
02 | VALUES(1, "Tea", "Basic", "12345678", NULL);
```

Primjer 8: Dodavanje podataka u tablicu ‘users‘

Na primjeru 8 je prikazano dodavanje podataka u tablicu ‘users‘ koja je opisana u primjeru 3. Možemo vidjeti kako su sva zadana ograničenja poštivana.

Za brisanje zapisa iz tablice koristi se DELETE naredba. Bitno je ne izostaviti WHERE dio jer će se u protivnom svi podaci iz tablice izbrisati. Najčešće se briše redak po redak, a podaci se filtriraju po primarnom ključu tablice.

```
01 | DELETE users
02 | WHERE users.id = 1;
```

Primjer 9: Primjer brisanja podatka iz tablice ‘users‘

Na primjeru 9 možemo vidjeti sintaksu za brisanje podatka iz tablice users gdje je id jednak 1.

Za mijenjanje podataka u tablici koristi se naredba UPDATE. Bitno je ne izostaviti WHERE dio jer će se u protivnom na sve podatke u tablici postaviti nove vrijednosti. UPDATE se odnosi samo na one retke koji zadovoljavaju kriterije u WHERE dijelu.

Primjer mijenjanja jednog zapisa iz tablice ‘users‘ prikazana je idućim primjerom:

```
01 | UPDATE users SET
02 |     password = "100001"
03 |     rfid_tag = "896358746"
04 | WHERE users.id = 1;
```

Primjer 10: Primjer brisanja podatka iz tablice ‘users‘

6.4 Dohvaćanje podataka iz tablica

Sljedeći korak, nakon kreiranja tablica i punjenja istih podacima, je testiranje baze podataka. Bazu testiramo pisanjem upita nad bazom kako bi smo se uvjerili možemo li doći do svih potrebnih podataka. Dohvaćanje podataka iz tablica izvršava se naredbom SELECT. Upiti mogu biti rađeni na jednoj ili više tablica. Upite dijelimo na jednostavne i složene. Jednostavni upiti su oni koji u FROM klauzuli imaju samo jednu tablici, dok složeni upiti u FROM klauzuli imaju više tablica. U nastavku će biti prikazani neki primjeri upita korištenih za testiranje baze iz projektnog dijela ovog rada.

```
SELECT * FROM users;
+-----+------------+-----------+----------+----------+
| id  | first_name | last_name | password | rfid_tag |
+-----+------------+-----------+----------+----------+
| 100 | Marko      | Marulic   | 1234     | AS0293C4 |
| 200 | Ivana      | Mazuranic | 1234     | AS0293C4 |
| 300 | Ivan       | Tisov     | 1234     | AS0293C4 |
+-----+------------+-----------+----------+----------+
3 rows in set (0.001 sec)
```

Primjer 11: Upit nad jednom tablicom

Na primjeru 11 možemo vidjeti upit nad jednom tablicom, odnosno jedan jednostavan upit. Ovaj upit vraća sve podatke koji se nalaze u tablici ‘users‘. Upit pripada jednostavnim upitima. SELECT * označava da želimo dohvatiti sve stupce iz tablice, a FROM dio upita nam omogućava da naznačimo iz koje tablice želimo dohvatiti stupce.

```
SELECT
    CONCAT(users.first_name, " ", users.last_name) AS "Worker",
    roles.title AS "Role"
FROM user_has_roles
INNER JOIN users ON users.id = user_has_roles.user_id
INNER JOIN roles ON roles.id = user_has_roles.role_id
ORDER BY users.last_name;
+---------------------+---------------+
| Worker              | Role          |
+---------------------+---------------+
| Marko Marulic       | User          |
| Ivan Mazuranic      | User          |
| Ivan Tisov          | Administrator |
+---------------------+---------------+
3 rows in set (0.001 sec)
```

Primjer 12: Upit nad više tablica

Primjer 12 prikazuje dohvaćanje podataka iz tri tablice, odnosno složeni upit. Iz tablice ‘users‘ dohvaća se ime i prezime radnika. Iako su u tablici to dva zasebna atributa, pomoću funkcije CONCAT() AS možemo podake iz više stupaca spojiti u jedan stupac. Iz tablice ‘roles‘ dohvaćamo atribut title koji označava naziv uloge koja je dodjeljena korisniku. Tablica ‘user_has_roles‘ naziva se pivot tablica, a ona služi kako bi rastavila vezu "više na više". U ovom podaci su dohvaćeni korištenjem inner join spajanja. Također rezultati su

sortirani pomoću naredbe ORDER BY po prezimenu. Također sortirati možemo uzlazno, odnosno silazno ako nakon vrijednosti prema kojoj sortiramo dodamo ASC odnosno DESC.

```
SELECT
    CONCAT(u.first_name, " ", u.last_name) AS "Worker",
    wp.title AS 'Work position'
FROM users u, user_work_positions uwp, work_positions wp
WHERE u.id = uwp.user_id
AND wp.id = uwp.work_position_id
AND uwp.type = 'Work position created'
AND uwp.created_at BETWEEN "2021-06-28 00:00:00" AND "2021-06-28 23:59:59"
   ;
+---------------------+---------------+
| Worker              | Work position |
+---------------------+---------------+
| Marko Marulic       | Pozicija 1    |
| Ivan Mazuranic      | Pozicija 2    |
| Ivan Tisov          | Pozicija 3    |
+---------------------+---------------+
3 rows in set (0.001 sec)
```

Primjer 13: Filtriranje korištenjem operatora BETWEEN

Primjer 13 je također jedan složeni upit. Tablice ‘users‘, ‘user_work_positions‘ i ‘work_positions‘ su u ovom primjeru spojene korištenjem klauzula FROM i WHERE. Ovo je najstariji način spajanja tablica. U FROM klauzuli navode se tablice iz kojih se dohvaćaju podaci, a u WHERE klauzuli se spajaju navedene tablice. Tablice je potrebni spojiti na pravilan način, odnosno potrebno je izjednačiti primarne i vanjske ključeve. Također u ovom primjeru podatke filtriramo korištenjem operatora BETWEEN. Ovaj operator omogućuje definiranje raspona kojega vrijednost mora zadovoljavati. U ovom slučaju upit će nam vratiti ime i prezime radnika te poziciju na kojoj je radio dana 28.06.2021. godine.

```
SELECT
    COUNT(*) AS "Broj sastavljenih proizvoda 2021-07-01"
FROM activity_logs al
WHERE al.created_at LIKE '%2021-07-01%'
AND al.activity_type_id = 28
AND al.user_id = 6;
+----------------------------------------+
| Broj sastavljenih proizvoda 2021-07-01 |
+----------------------------------------+
| 6                                      |
+----------------------------------------+
1 row in set (0.001 sec)
```

Primjer 14: Korištenje operatora LIKE

Operator LIKE služi za uspoređivanje niza znakova. Primjer 14 prikazuje upit koji koristi operator LIKE da bi pretražio sve proizvode koji su sastavljeni 01.07.2021. godine, odnosno atribut ‘created_at‘ u sebi sadrži 2021-07-01.

Upit koji je sadržan unutar drugog upita nazivamo podupitom. Podupiti se mogu nalaziti u različitim klauzulama naredbe. Klauzule u kojima se mogu nalaziti podupiti su sljedeće: SELECT, FROM, WHERE, ORDER BY.

```
01 | SELECT DISTINCT
02 | u.id AS "Id",
03 | CONCAT(u.last_name, " ", u.first_name) AS "Worker",
04 | uwp.work_position_id AS "Work position id",
05 | wp.title AS "Work position",
06 | uwp.created_at AS "Created at"
07 | FROM users u
08 | INNER JOIN user_work_positions uwp ON uwp.user_id = u.id
09 | AND uwp.created_at = (SELECT MAX(created_at)
10 |                       FROM user_work_positions
11 |                       WHERE user_work_positions.user_id = uwp.user_id
12 |                       AND user_work_positions.type = 'Work position
13 |                       created')
14 | INNER JOIN work_positions wp ON uwp.work_position_id = wp.id;
```

Primjer 15: Upit s podupitom

Primjer 15 je primjer jednog složenog upita koji koristi podupit. Ovaj upit vraća id zaposlenika, ime i prezime zaposlenika, posljednju kreiranu radnu poziciju i kada je ona kreirana. Kako bi bilo moguće odrediti koja mu je posljednja kreirana pozicija korišten je podupit. U podupitu također je vidljivo korištenje funkcije MAX() koja služi za pronalaženje najveće vrijednosti u stupcu. Funkcija MAX() jedna je od agregirajućih funkcija. Agregirajuće funkcije su one funkcije koje na temelju skupa vrijednosti vraćaju jednu. Osim funkcije MAX() agregirajućim funkcijama pripadaju i AVG() koja vraća prosječnu vrijednost u stupcu, COUNT() koja broji redove, MIN() koja vraća minimalnu vrijednost stupca i funkcija SUM() koja sumira vrijednosti u stupcu. Također u ovom primjeru prikazana je i uporaba opcije DISTINCT. Ova opcija omogućava izbjegavanje ispisivanje iste vrijednosti više puta.

Popis slika

Popis tablica

Popis primjera

Literatura

[1] MySQL, *Customers*, dostupno na `https://www.mysql.com/customers/`, [08.06.2021.]

[2] T. Carić, M. Buntić, *Uvod u relacijske baze podataka*, dostupno na `http://files.fpz.hr/Djelatnici/tcaric/Tonci-Caric-Baze-podataka.pdf`, [08.06.2021.]

[3] R. Manger, *Osnove projektiranja baza podataka*, Sveučilište u Zagrebu, Zagreb, dostupno na `https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/d310_polaznik.pdf`, [08.06.2021.]

[4] K. Rabuzin, *Uvod u SQL*, Fakultet organizacije i informatike Sveučilište u Zagrebu, Varaždin, 2011.

[5] K. Rabuzin, *SQL - napredne teme*, Fakultet organizacije i informatike Sveučilište u Zagrebu, Varaždin, 2011.

[6] D. Buytaert, *The history of MySQL AB*, dostupno na `https://dri.es/the-history-of-mysql-ab`, [16.06.2021.]

[7] Wikipedia, *Proizvodni proces*, dostupno na `https://hr.wikipedia.org/wiki/Proizvodni_proces`, [13.07.2021.]

[8] Wikipedia, *MySQL*, dostupno na `https://en.wikipedia.org/wiki/MySQL`, [13.07.2021.]

[9] Wikipedia, *MySQL Workbench*, dostupno na `https://en.wikipedia.org/wiki/MySQL_Workbench`, [13.07.2021.]

[10] L. Cetinić, *MySQL – seminarski rad*, dostupno na `https://lcetinic1611.wordpress.com/2019/12/09/mysql-seminarski-rad/`, [13.07.2021.]

[11] I. Mišić, *MySQL tipovi podataka*, dostupno na `https://www.hdonweb.com/programiranje/mysql-tipovi-podataka-data-types` [14.07.2021.]

[12] Ž. Kemić, *Primje baze podataka u sustavu MySQL*, Fakultet organizacije i informatike Sveučilište u Zagrebu, Varaždin, 2013.