3. Upravljanje podacima

U B2B elektronskoj razmeni podataka (engl. *Electronic Data Interchange* - EDI), poruke koje sadrže kodove proizvoda ili usluga, identifikaciju transportnih jedinica, kao i pravne dokumente, razmenjuju se među partnerima u lancu snabdevanja. Obično imaju oblik alfanumeričkih nizova.

3.1 Informacije-Podaci-Znanje

U računarskim informacionim sistemima prikaz informacija varira zavisno od njihove namene i upotrebe. Prikaz može biti alfanumerički ili binarni s obzirom na to da li predstavlja tekst, sliku, zvuk ili izvršni program. Da bi se podaci različitih vrsta (npr. brojeva, znakova, datuma, valuta itd.). mogli sačuvati, preuzeti, obraditi i preneti) moraju biti ispravno kodirani. Alfanumerički formati predstavljanja podataka potiču iz ASCII (ask-key) abecede, nakon što su se razvili u smislu nacionalnih skupova znakova (npr. standardi 8859-1, Latin 1 i 8859-2, Latin 2) i konačno napredovali u međunarodne UTF-8 i UTF-16 formate. Stoga omogućavaju zajedničko tumačenje podataka od strane poslovnih partnera koji pripadaju različitim etničkim gripama i geografskim sredinama. Dok alfanumerički nizovi zavise od njihovog kodiranja, numerički podaci se uglavnom razlikuju po veličini i/ili preciznosti.

Proces transformisanja podataka iz izvornog analognog u digitalni oblik popularno se naziva digitalizacija. Odgovarajuće dizajnirani pomoćni i aplikacioni programi koji prihvataju podatke iz različitih izvora (npr. optički skeneri, elektronski senzori, EDI, itd.) omogućavaju organizacijama da automatizuju svoje prikupljanje podataka, čuvanje, obradu i prenos unutar i između svojih računarskih informacionih sistema.

Kada se prikupe, podaci različitih vrsta mogu se spojiti i organizovati u tabele podataka, baze podataka, skladišta podataka (hronološki redosled) ili baze znanja (konceptualni redosled). Viši nivoi organizacije podataka omogućavaju automatizovano razvrstavanje, zaključivanje i predstavljanje tako akumuliranog znanja u analitičke svrhe.

3.2 Logistički podaci

U logistici se EDI koristi za prenos transakcionih podataka između poslovnih partnera. Budući da mogu koristiti različite jezike i aplikacije, potrebna je njihova konverzija u zajednički format (npr. XML, JSON) kako bi ih mogli interpretirati različiti informacioni sistemi partnera (W3schools, 2023). Za brzu identifikaciju i manipulaciju osmišljeni su bar kodovi i RFID oznake.

Kako bi se omogućila međunarodna saradnja, trebalo je definisati globalno prihvaćene formate podataka za potrebe logistike. Formati logističkih podataka odgovaraju oznakama usluga i proizvoda, identifikaciji transportnih jedinica i kodovima transakcija, obično u obliku alfanumeričkih nizova s vremenskim žigom. Radi jednostavnosti rukovanja i brzine obrade, ovi su kodovi standardizovani i kodirani kao optički čitljivi bar-kodovi ili elektromagnetski čitljivi radiofrekventni identifikacioni (RFID) kodovi.

Bar-kod (EAN/UCC) je višesektorski i međunarodni oblik numerisanja stavki (POS EAN-8 i EAN-13, promenjivi EAN-128, traka podataka, ITF-14, QR, matrica podataka itd.). Koriste se za identifikaciju proizvoda, serija proizvoda ili pošiljaka (1D kodovi) kao i usluga (2D kodovi). Među 2D bar kodovima QR kod je najprisutniji, čitljiv i pametnim telefonima, što povećava njihovu upotrebljivost u različitim područjima primene.

RFID kodovi se prvenstveno koriste na isti način kao bar kodovi. Oni jedinstveno identifikuju artikle ili usluge. Obično, RFID nalepnice nose još više informacija na čipu veličine glave čiode. Osim identifikacije, RFID oznake omogućavaju beleženje podataka o praćenju, što je često potrebno u logističkim aplikacijama. Za razliku od linijskih kodova, RFID omogućava njihovo skeniranje bez direktnog vidnog polja, kao i skeniranje više nalepnica odjednom.

GS1 standardom EPC Gen2 (ISO/IEC 18000-6:2013) uspostavljen je tehnološki standard koji određuje komunikaciju između RFID tagova i čitača. Slično bar kodovima, EPCglobal standardi povezuju RFID tehnologiju s EPC (engl. *Electronic Product Code*) označavanjem proizvoda, logističkih transportnih jedinica, lokacija, zaliha, povratnih artikala, dokumenata itd. za direktnu, automatiziranu identifikaciju i praćenje logističkih jedinica unutar lanca snabdevanja.

EPCglobal standardi takođe predstavljaju osnovu GDSN-a (engl. *Global Data Synchronization Network*). Omogućavaju automatizivano prikupljanje i razmenu specifikacijskih podataka o proizvodima i njihovoj ambalaži, čime se preduzećima omogućava centralizovano upravljanje tim podacima kako bi ih oni i njihovi partneri naizmenično koristili.

Tabela 3.1 sažima različite identifikacione tehnologije s njihovim primenama. Otkriva niz jednodimenzionalnih i dvodimenzionalnih bar kodova, kao i različite klase RFID kodova s njihovim mogućnostima.

Tabela 3.1Tehnologije označavanja.

Tehnologija	**Primena**
Bar kod 1D	Maloprodajni artikli i komponente proizvoda
Bar kod 2D	Usluge (npr. UPS, avionske karte), veleprodajni artikli koji zahtevaju praćenje
RFID klasa 1 (pasivno, R-oznake)	Predmeti koji zahtevaju masovnu identifikaciju, kontrolu pristupa
RFID klasa 2 (pasivno, RW-tagovi)	Stavke koje zahtevaju praćenje
RFID klasa 3 (poluaktivan, RW-tagovi)	Kontrola pristupa s dodatnim informacijama za praćenje
RFID klasa 4 (aktivan, RW-tagovi)	Trasiranje i praćenje zatvorenog prostora
RFID klasa 5 (aktivne oznake/ispitivači)	Praćenje otvorenog prostora, blizina usluge s omogućenim uređajima, usluge zasnovane na lokaciji

Budući trendovi u označavanju, praćenju i sledivosti slede dva glavna smera: minijaturizacija i raznolikost. Bar kodovi (1D) takođe moraju omogućiti označavanje minijaturnih predmeta (npr. medicinskih kapsula). Novi kodovi matrice podataka (2D) ne samo da će omogućiti ispravljanje grešaka tokom skeniranja, već i šifrovanje podataka.

RFID se nastavlja širiti na druga područja upotrebe kao što je identifikacija od strane davaoca usluga (npr. železnička kartica, prijava na posao, itd.), beskontaktna plaćanja (npr. bežični prenos novca, plaćanja na automatima) i pametna rešenja (npr. upravljanje pametnom kućom, daljinski upravljani pametni uređaji itd.) kao i e-valute.

3.3 Organizacija podataka

Osim što imaju određeni format, podaci se mogu organizovati na različite načine kako bi se olakšalo njihovo upravljanje, obrada i prezentacija. Iako su podaci na svom ulazu uglavnom nestrukturirani, njihovim čuvanjem, prenosom i obradom povećava se njihova organizovanost. U nastavku su prikazani uobičajeni oblici organizacije podataka po rastućoj složenosti od polustrukturiranih (npr. CSV) do strukturiranih (npr. proračunske tabele, baze podataka, itd.) formata.

Proračunske tabele

Prvi oblik organizacije podataka su dvodimenzionalni nizovi polja, koji se takođe nazivaju tabele ili proračunske tabele. Obično prvi red proračunske tabele označava značenja vrednosti unetih u osnovne kolone, nakon čega slede redovi podataka.

Polje ili ćelija tabele je najmanja jedinica podataka. Ima određeni tip (broj, datum, valuta itd.). Njegov sadržaj se može adresirati oznakama reda i kolone (npr. A1, koji predstavlja prvi red kolone A).

Svaki red tabele je grupa povezanih polja, koja predstavljaju zapis (npr.: transakcija, studentski zapis, podaci o proizvodu itd.). Budući da svi redovi tabele imaju istu strukturu, tip zapisa možemo definisati kao popis atributa (npr. podaci o studentu (ime, prezime, datum rođenja, mesto rođenja, ID...)) odgovarajućih tipova podataka.

Baze podataka

Datoteka ili tabela baze podataka je zbirka zapisa iste vrste. Baza podataka (engl. *Data Base* - DB) sastoji se od više međusobno povezanih tabela. Dakle, ANSI definicija baze podataka:

- Podaci baze podataka su međusobno povezani i sortirani;
- Bazu podataka može istovremeno koristiti više korisnika;

- Podaci u bazi podataka se ne ponavljaju;
- Baza podataka je sačuvana u računaru.

Iz gornje definicije mogu se izvući neki zaključci o klijent-server arhitekturi gde server drži DB, kojoj pristupaju njegovi klijenti. Naravno, za pristup DB-u mora biti uspostavljena komunikaciona mreža između servera i njegovih klijenata. DB server se obično naziva njen "*back-end*", dok klijenti predstavljaju njen "*front-end*". Sistem za upravljanje bazom podataka (engl. *Data Base Management System* - DBMS) na serveru omogućava svojim klijentima pristup podacima skladištenim u DB-u putem svojih aplikacija programskog interfejsa (API) i DBM funkcija. DBM funkcije su mehanizmi koji omogućavaju unos, preuzimanje, obradu i prezentaciju podataka u DB-u. Za pozivanje ovih funkcija definisani su standardni jezici upita (engl. *Standard Query Languages* - SQL).

Model relacione baze podataka

Postoje različiti oblici organizacije DB-a, a najčešći je relacioni model (RDB). Osnovna ideja ovog modela je činjenica da korisnik ne može unapred znati sve moguće upotrebe podataka koji se skladište u DB-u. Budući da obično ne postoje fiksni putevi pretraživanja kroz datoteke baze podataka, osmišljeni su različiti jezici upit za preuzimanje i manipulaciju podacima. RDB model se bazira na konceptu entiteta i odnosa:

- Entitet je osoba/stvar/koncept koji se može jedinstveno identifikovati i ima atribute.
- Relacija predstavlja način povezivanja dva ili više entiteta.

RDB tabele, koje predstavljaju entitete ili relacije, međusobno su povezane pomoću ključeva. Skup atributa koji jedinstveno identifikuje entitet naziva se njegov primarni ključ. Kada se primarni ključ pojavljuje kao polje u drugoj tabeli s ciljem ispunjavanja relacije s izvornom tabelom, naziva se sekundarnim ili stranim ključem. Tabela može sadržati samo jedan primarni ključ za jedinstvenu identifikaciju svojih zapisa, dok može sadržati više sekundarnih ključeva.

Generalno, postoje dva pristupa konstruisanju RDB-a: analitički i sintetički.

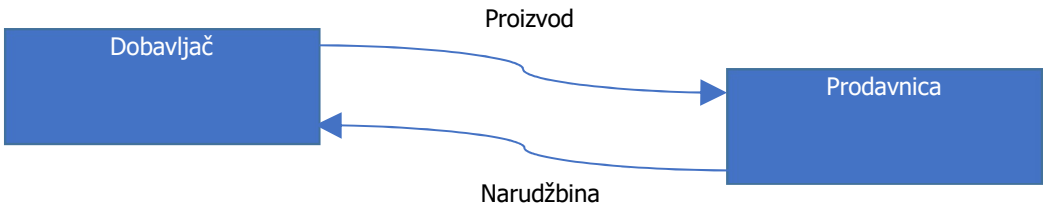
Analitički pristup izradi RDB-a sastoji se od sledeća četiri koraka:

1. Analiza stvarnog sveta - globalni model;
2. Određivanje entiteta i odnosa – konceptualni model (npr. E-R dijagram);

3. Određivanje logičkog modela – relaciona shema;

4. Izgradnja baze podataka (DBMS) – fizički model.

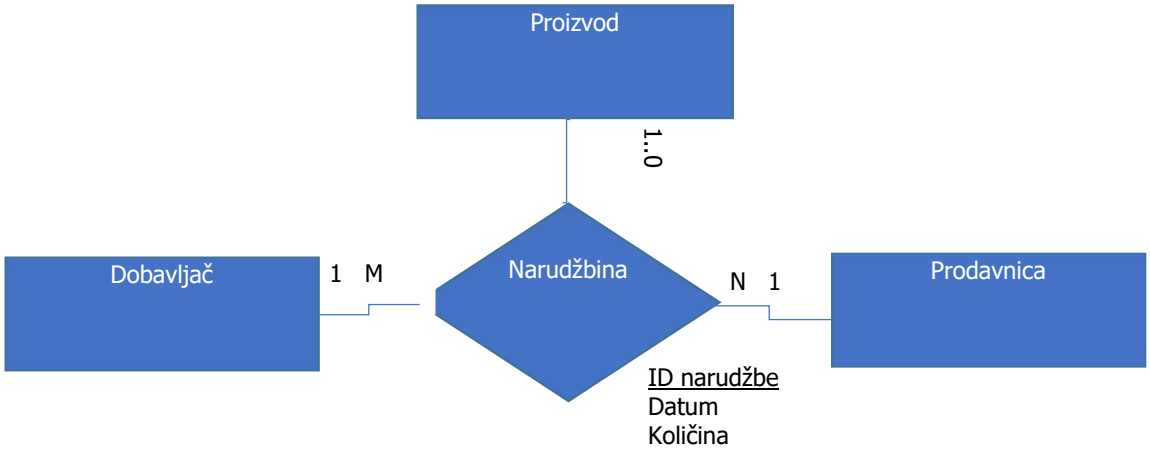
Kako bismo ilustrovali pristup, razmotrimo primer trgovinskog lanca i njegovih dobavljača (slika 3.1). Svaka prodavnica ima više dobavljača. Svaki dobavljač može snabdevati različite prodavnice. Ciklus popunjavanja započinje narudžbinom iz prodavnice. Zauzvrat dobavljač isporučuje robu u prodavnicu. Narudžbine su transakcije u kojima se spajaju podaci o prodavnici, dobavljaču i isporučenoj robi (slika 3.2).



Slika 3.1 Globalni model.

DOBAVLJAČ (Dobavljač_ID#, Dobavljač_naziv, Dobavljač_kontakt)
PRODAVAONICA (Prodavaonica_ID#, Prodavaonica_naziv, Prodavaonica_adresa)
NARUDŽBA (Dobavljač_ID#, Prodavaonica_ID#, Narudžba_ID#, Datum, Količina, EPC)
PROIZVOD (EPC#, Proizvod_naziv, Proizvod_cijena)

Slika 3.14Logički model.



Slika 3.13 Konceptualni model.

Logički model predstavlja RDB tabele prema vrstama njihovih zapisa. U logičkom modelu (slika 3.3) određeni atributi sadrže simbol tarabe (engl. *hashtag*) (#). To znači da polje koje oni predstavljaju jeste ili pripada (kompozitnom) primarnom ključu. Neka polja su podvučena. Nazivaju se sekundarnim ili stranim ključevima, budući da referenciraju primarne ključeve povezanih tabela.

Sintetički pristup RDB-u sastoji se od sledeća tri koraka:

1. Analiza podataka – popis svih relevantnih atributa;
2. Određivanje logičkog modela normalizacijom – relaciona shema;
3. Fizički model (DBMS).

Normalizacija ili kanonička sinteza (Kent, 1983) osigurava da se obrnutim inženjeringom iz relevantnih atributa formira DB koja ispunjava uslove RDB-a. Dok početni normalni oblik (0NF) atributa predstavlja tabelu neuređenih atributa, naknadni normalni oblici, kako ih definiše (Codd, 1970), predstavljaju više nivoe organizacije podataka. Može se tvrditi da je dostizanjem 3. normalne forme postignuta shema koja ispunjava zahteve za RDB logički model.

Tabela je u 1NF, ako predstavlja relaciju. Time je osigurano da se sve grupe podataka koje se ponavljaju čuvaju odvojeno i stoga se ne ponavljaju.

Tabela u 2NF je u 1NF. Dodatno, nijedan atribut ključa ne sme delomično funkcionalno zavisiti od primarnog ključa. Tako se svi ključevi koji jedinstveno identifikuju određene atribute čuvaju odvojeno. Ovo je uglavnom da bi se osiguralo da se samo atributi koji zavise od svih (delova) primarnog ključa čuvaju u jednoj tabeli.

Tabela u 3NF je u 2NF. Osim toga, nijedan atribut koji nije ključ nije tranzitivno zavisan od primarnog ključa. To znači da se svi ne-ključni atributi koji mogu predstavljati ključ za određene druge ne-ključne atribute čuvaju u zasebnoj tabeli, pri čemu se samo njihov ključ održava u izvornoj tabeli kao strani ključ.

Prateći korake normalizacije od 1NF do 3NF, završava se s logičkim modelom koji odgovara propisima relacione baze podataka (RDB). Viši oblici normalizacije uglavnom su za optimizaciju RDB modela.

Tabela u Boyce-Codd NF (BCNF) je u 3NF; dodatno, svaka odrednica je ključ. Ovo uklanja sve odnose koji nisu obuhvaćeni postojećim redosledom ključeva iz izvorne tabele, čime se formiraju dodatne tabele za svaki ključ kandidata. Tabela u 4NF je u 3NF i BCNF. Osim toga, svaki atribut s više vrednosti koji delimično zavisi od ključa nalazi se u sopstvenoj tabeli. 4NF je namenjen uklanjanju svih mogućih preostalih atributa s više vrednosti iz originalne tabele. Tabela u 5NF je u 4NF; dodatno, svaka JOIN operacija je predviđena ključevima. Tabela u 6NF je u 5NF; dodatno, uzimaju se u obzir sve netrivijalne JOIN-zavisnosti.

Može se uočiti da kod viših oblika organizacije broj tabela raste sa svakim korakom. Stoga je razumno posmatrati fragmentaciju podataka kako bi se sprečilo stvaranje nepotrebnih tabela kojima se retko pristupa.

Tabela 3.2Primer normalizacije RBD-a.

<table>
<tr>
<td>ONF

NARUDŽBINA

• Dobavljač_ID *
• Ime_dobavljača
• Dobavljač_kontakt
• Prodavnica_ID *
• Naziv_prodavnice
• Adresa_prodavnice
• ID_narudžbine*
• Datum
• EPC
• Količina
• Naziv_proizvoda
• Cena_proizvoda</td>
<td>1NF

NARUDŽBINA

• Dobavljač_ID *
• Ime_dobavljača
• Dobavljač_kontakt
• Prodavnica_ID *
• Naziv_prodavnice
• Adresa_prodavnice
• ID_narudžbine*
• Datum
• EPC
• Količina
• Naziv_proizvoda
• Cena_proizvoda</td>
</tr>
<tr>
<td colspan="2">* Ključevi kandidata za ponavljajuću grupu atributa, koji jedinstveno identifikuju narudžbinu</td>
</tr>
<tr>
<td>2NF

DOBAVLJAČ

• Dobavljač _ID #
• Ime_dobavljača
• Dobavljač_kontakt

PRODAVNICA

• Prodavnica _ID #
• Naziv_prodavnice
• Adresa_prodavnice</td>
<td>

NARUDŽBINA

• Narudžbina_ID#
• Dobavljač_ID#
• Prodavnica _ID #
• EPC
• Naziv_proizvoda
• Cena_proizvoda
• Datum
• Količina</td>
</tr>
</table>

3NF NARUDŽBINA • Dobavljač _ID # • Prodavnica _ID # • Narudžbina_ID# • Datum • Količina • ID_proizvoda	 PROIZVOD • EPC # • Naziv_proizvoda • Cena_proizvoda

Programski jezici za rad sa DB

Za upravljanje podacima uglavnom se koriste dva programska jezika: Structured Query Language (SQL) i Query by Example (QBE). Dok se SQL smatra programskim jezikom i API-jem DBMS-a, QBE se uglavnom koristi s DBMS-om direktno za upravljanje bazom podataka i skladištenje podataka.

Standardni SQL (ISO/IEC 9075, 1986-2016) je programski jezik četvrte generacije za manipulaciju bazama podataka. Omogućava pretraživanje, dodavanje, menjanje kao i brisanje zapisa podataka. Uprkos njegovoj standardizaciji, postoje male razlike u njegovoj implementaciji s različitim sistemima za upravljanje bazama podataka (DBMS).

U nastavku je jezik ukratko predstavljen s najčešćim opcijama. Prema konvenciji, SQL ključne reči pišu se velikim slovima i svaka rečenica završava tačkom sa zarezom. Rečenice su predstavljene s poveznicama na povezane referentne materijale koji nude dodatne informacije.

Svaka manipulacija bazom podataka počinje njenim stvaranjem. Rečenica

CREATE DATABASE *baza podataka_naziv*;

stvara novu praznu bazu podataka s navedenim imenom.

Kao što je gore objašnjeno, podaci unutar baza podataka organizovani su u tabele zapisa podataka određenog tipa gde svi redovi dele zajedničku strukturu. Za izradu tabele koristi se sledeća rečenica:

CREATE TABLE *tabela_naziv* (
kolona1 tip1,
kolona2 tip2, kolona3 tip3,
….);

Svaka imenovana kolona predstavlja atribut s određenim tipom podataka. Na primer, u:

CREATE TABLE *Prodavnica* (Prodavnica _ *ID* int NOT NULL PRIMARY KEY,…);

CREATE TABLE *Narudžbina* (Narudžbina_ *ID int* NOT NULL PRIMARY KEY, …, Product_Id int FOREIGN KEY REFERENCES *Proizvod* (*EPC*));

kreiraju se dve tabele. Prva sadrži podatke o kupcima, dok druga sadrži podatke o njihovim narudžbinma, pozivajući se na prvu tabelu preko broja kupca kao stranog ključa.

Dok su podaci u tabeli već sortirani po primarnom ključu, mogu se dodatno sortirati po drugim atributima, pod uslovom da su indeksirani. Možemo ga indeksirati stvaranjem indeksa na datom atributu(ima) sedećom rečenicom:

CREATE INDEX *indeksa_naziv* ON *tabela_naziv* (*naziv_kolone*);

Svaka manipulacija podacima na indeksiranoj tabeli traje malo duže, budući da za njenu konzistentnost treba ne samo proveriti podatke koji daju ključeve i pravilno poređati podatke, već i druge atribute iz navedenog indeksa.

Najčešća operacija u bazi podataka je upit podataka omogućen naredbom SELECT :

SELECT *kolona1, kolona2*, …
FROM *tabela_ naziv*;

Ovaj upit vraća podatke u koloni1, koloni2 itd. iz tabele. Rečenice upita obično se formiraju pružanjem dodatnih opcija, filtriranjem podataka, ispunjavanjem navedenih uslova:

WHERE navodi uslov koji određuje kriterijume izbora zapisa.

GROUP BY spaja zapise, imajući zajedničko svojstvo za omogućavanje zajedničkih funkcija.

HAVING specificira agregatne funkcije na grupama definiranim naredbama GROUP BY.

ORDER BY specificira atribute prema kojima su povratni zapisi poređani.

Na primer:

```
SELECT "Prodavnica" . " Prodavnica_naziv " , "Proizvod" . " Proizvod_naziv " , " Narudžbina" .
"Količina " FROM "Narudžbina" , "Proizvod" , "Dobavljač" , " Prodavnica " WHERE
"Narudžbina" . " ID_proizvoda " = " Proizvod" . "EPC " AND "Narudžbina" . " Dobavljač_ID "
= "Dobavljač" . "Dobavljač_ID" AND "Narudžbina" . " Prodavnica _ID" =
"Prodavnica"."Store_ID" ORDER BY "Prodavnica"."Prodavnica_naziv" ASC
```

vraća popis prodavnica s njihovim naručenim proizvodima i količinama, poređanih prema nazivu prodavnice.

Najvažnija operacija u procesu selekcije je operacija JOIN. Često zamenjuje uslov WHERE kao JOIN ON, nakon čega sledi uslov. Upoređuje vrednosti kolona i na osnovu poređenja određuje da li ih treba uključiti u rezultat ili ne. U LEFT JOIN zapis se vraća ako su kriterijumi ispunjeni u levoj tabeli i obrnuto u operaciji RIGHT JOIN. Kao što je gore navedeno, uslov mora biti ispunjen u obe tabele kako bi bio u skladu s operacijom INNER JOIN ili FULL JOIN. Budući da se ovo drugo najčešće koristi, kao sinonim može se koristiti JOIN. Pozivajući se na uslove 5 i 6 normalne forme, ovo je ista JOIN operacija, koju je potrebno ispuniti da bi se ispunili uslovi odgovarajućeg NF-a.

Za unos novih podataka u tabelu koristi se operacija INSERT INTO:

```
INSERT INTO tabela_naziv (kolona1 , [kolona2 , ... ] )
VALUES ( vrednost1 , [vrednost2 , ...]);
```

Da bi bile uspešne, vrednosti u operaciji trebaju ispuniti sve uslove atributa označenih nazivima kolona. Nazive kolona ne treba navesti u slučaju da su sve vrednosti navedene. U slučaju da su u tabeli predviđene neke DEFAULT vrednosti, ne treba ih navoditi, osim ako se razlikuju.

Kada se podaci unesu, mogu se modifikovati naredbom UPDATE :

```
UPDATE tabela_naziv
SET kolona=vrednost1, kolona2=vrednost 2 ,...
WHERE neka_kolona = neka_vrednost;
```

U izjavi su date nove vrednosti za polja u navedenim kolonama. Kriterijumi izbora reda označeni su specifikatorom WHERE, koji određuje sve vrednosti kolona na koje se odnosi naredba UPDATE. Kako bi se sprečile neželjene promene, potreban je dodatni oprez pri formulisanju kriterijuma izbora.

Zapis ili više zapisa može se izbrisati iz tabele operacijom DELETE :

DELETE FROM *tabela_naziv*
WHERE *neka_kolona = neka_vrednost*;

Kao i s naredbom UPDATE, specifikator WHERE koristi se za određivanje svih redova koje treba izbrisati.

Naravno, upravljanje bazom podataka se ne završava ovde. Svaki element baze podataka takođe se može ukloniti, izmeniti i/ili zameniti novim. U slučaju da se indeks, tabela ili baza podataka trebaju ukloniti, mogu se primeniti sledeće izjave:

DROP INDEX *indeks_ime* ON *tablic_ime*;

DROP TABLE *tabela_naziv*;

DROP DATABASE *baza podataka_naziv*;

Ako neko samo želi ukloniti podatke iz tabele, može se koristiti naredba TRUNCATE:

TRUNCATE TABLE *tabela_naziv* ;

U slučaju da želite dodati ili ukloniti atribut (kolonu u/iz tabele, to možete učiniti naredbom ALTER:

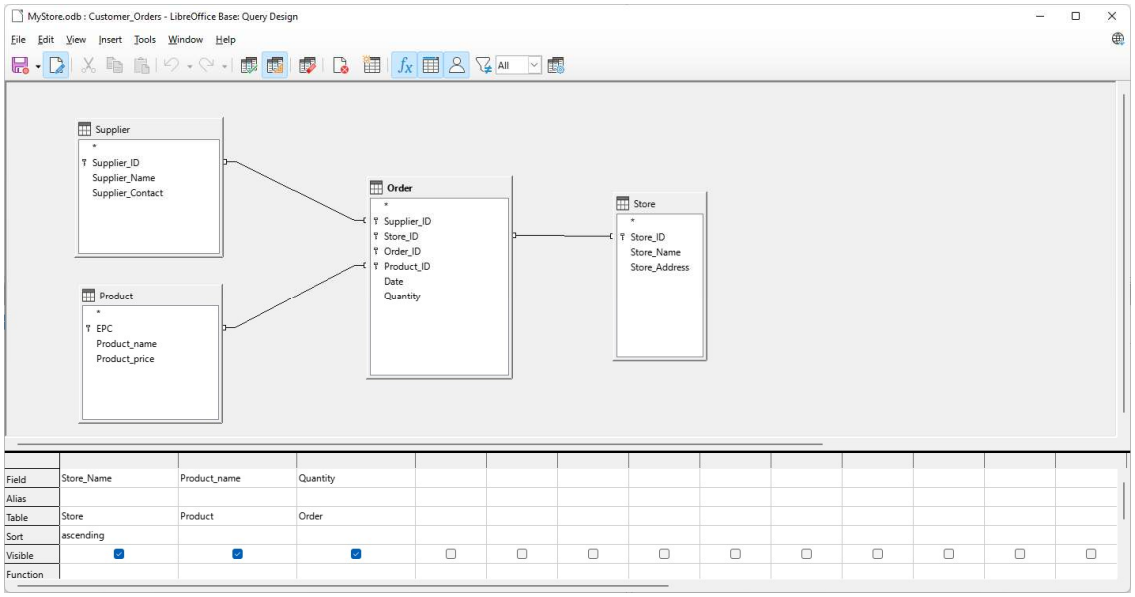
ALTER TABLE *tabela_naziv* ADD kolona_*naziv vrsta podataka*;

ALTER TABLE *tabela_naziv* DROP COLUMN kolona_*naziv;*

Ovim objašnjenjima se zaključuje kratki pregled SQL jezika i njegovih najčešćih scenarija upotrebe. SQL se obično koristi u klijent-server arhitekturama s DBMS-om koji je na glavnom računaru servera. Da bi mu se pristupilo, izdaju se SQL naredbe, bilo od strane aplikativnog programa klijenta ili web-interfejsa DBMS servera.

S druge strane, QBE se takođe često koristi s relacionim DBMS-om s grafičkim korisničkim interfejsom (engl. *Graphical User Interface* - GUI), kao što su MS Access ili LibreOffice Base. S QBE-om se baza podataka i njene tabele stvaraju mnogo interaktivnije, a njihovu strukturu je lakše održavati. Kao što mu ime govori, nudi i jednostavniji oblik unosa i otkrivanja podataka. Za izvođenje pretraživanja potrebno je sastaviti sve tabele koje se koriste u pretraživanju i zatim uspostaviti uslove kao uzorke u poljima kolona kako bi se filtrirali relevantni podaci (Slika 3.4). Formulacija upita dopunjena je postojećim relacijama između

tabela. Kao i obično, rezultat takvog upita je još jedna tabela s rezultirajućim podacima, koji se kasnije mogu dalje obraditi. Na ovaj način takođe se mogu formirati kaskadni ili višefazni upiti.



Slika 3.15QBE upit ekvivalentan gornjem SQL upitu.

Filteri i maske podataka

Kako bi se sprečili pogrešni ili nepotpuni podaci, koji bi mogli ometati njihovu obradu i tumačenje, potrebno je primeniti dodatne mere opreza:

1. Filtriranje praznih redova i kolona;
2. Primena stroge tipizacije podataka za sprečavanje računarskih grešaka;
3. Definisanje maski za unos kako bi se sprečio unos pogrešnih podataka;
4. Pristranost podataka kako bi se sprečili pogrešni rezultati.

Prazni redovi i kolone čest su izvor grešaka koje uglavnom potiču od lošeg interfejsa u aplikacijama za prikupljanje podataka. Otkazane ili nedovršene transakcije obično rezultiraju praznim redovima ili nedostajućim podacima u zapisima transakcija. Oni se samo delimično mogu rešiti aplikacijama proračunskih tablica gde se prazni redovi i podaci koji nedostaju mogu otkriti proverom ukupnog broja u odnosu na broj podataka koji nisu nula u redovima/kolonama. Mogu se filtrirati uklanjanjem praznih redova/kolona, ali možda to nije uvek željena aktivnost jer bismo mogli izgubiti i neke vredne podatke. Najbolji način da se to

spreči je korišćenjem DBMS-a koji bi s jedne strane omogućio unos samo kompletnih transakcionih podataka, dok bi s druge strane takođe sprečio prazne redove/kolone, budući da ih u bazama podataka nema.

Slabo upisivanje podataka je još jedan uobičajeni izvor grešaka. Ako se umesto numeričkih podataka unesu alfanumerički podaci, poput datuma ili iznosa valute, to bi rezultiralo greškama prilikom obrade tih podataka. U proračunskim tabelama, kao i u bazama podataka, pojedinačnim ćelijama podataka, koje predstavljaju vrednosti atributa u slogovima podataka, mogu se dodeliti tipovi podataka, što bi nas alarmiralo pri unosu podataka u pogrešnom formatu. Stoga se ovakvom merom može sprečiti da pogrešni podaci ometaju njihovu obradu.

Prilikom konstruisanja baza podataka, ograničenja se mogu primeniti na polja podataka koja predstavljaju atribute entiteta ili relacije. Osim što im se može dodeliti odgovarajuća vrsta, maske za unos, mogu se definisati čime se omogućava unos podataka, poput datuma, valuta, EAN kodova itd., samo u određenom formatu. To obično rešava mnoge pogrešne percepcije koje bi inače mogle nastati tokom obrade podataka.

Još jedan čest izvor grešaka su netačni podaci, koji predstavljaju podatke koji su s aspekta veličine veći ili manji od očekivanog. Ovakvi podaci bi mogli ometati našu obradu, dajući pogrešne rezultate. Teže ih je otkriti i mogu se filtrirati samo gledanjem podataka. U aplikacijama za proračunske tabele dobra uobičajena praksa bila bi određivanje minimalnih, maksimalnih i srednjih vrednosti podataka u odgovarajućim kolonama kako bi se otkrila moguća odstupanja. Ako se otkriju, mogu se naglasiti i ručno obraditi, ako ih je malo, ili filtrirati i modifikovati upitom u tabeli baze podataka ako ih je puno. U svakom slučaju treba ih pažljivo proceniti, kako se situacija ne bi još više pogoršala, a u tom slučaju bi bilo bolje ukloniti te podatke.

Skladišta podataka i baze znanja

Podaci u skladištima podataka prikupljaju se iz RDB-a i katalogizuju hronološkim redom. Obično se na podacima sprovode neke poslovne analize, a odeljenje analitike takođe čuva rezultate za kasnije potrebe. Nakon što se memorišu, ti se podaci obično ne menjaju kako bi se očuvala njihova doslednost.

Osim hronološkog reda, prilikom izgradnje baza znanja uzimaju se u obzir i kontekstualni redovi. Ovde su subjekti predstavljeni kao derivati entiteta najvišeg nivoa ili njegovih jedinica. Onosi između njih se uspostavljaju slobodnije jer su namenjeni ažuriranju i nadogradnji kako se koriste. Odnosi se uspostavljaju u obliku pravila i zasnovani su na svojstvima entiteta. Stoga je oblik u kojem se skladište nešto drugačiji. Često se skladište u obliku ontologija koje sadrže dublje znanje o prikupljenim podacima. Slično čuvanju rezultata upita u skladištima podataka, upiti u bazama znanja takođe se čuva za kasniju upotrebu za prikaz trenutnih rezultata, kako se menjaju entiteti, relacije i instance podataka.

Za razliku od baza podataka i skladišta podataka, koji su specifični za aplikaciju, baze znanja mogu biti nezavisne od aplikacije i često ih različite aplikacije koriste među domenama. Primer je prikazan u (Gumzej i dr., 2023).

3.4 Zaključak

U ovom poglavlju obrađeni su različiti aspekti upravljanja podacima u logistici. Osim prikaza podataka i standarda za skladištenje podataka, prikazana je organizacija podataka i mehanizmi pronalaženja. Konačno, neke uobičajene greške u automatizovanoj obradi podataka su rešene kako bi čitalas ostao na oprezu. Osim navedenih primera, više se može otkriti u pridruženim materijalima za učenje.

Literatura 3. poglavlja

- Codd E.F. (1970). A relational model of data for large shared data banks. Communications. ACM 13, 6, pp. 377–387.
- Gumzej, R., Kramberger, T., Dujak, D. (2023). A knowledge base for strategic logistics planning, Proceedings of the 23rd International Scientific Conference Business Logistics in Modern Management: October 5-6, 2023, Osijek, Croatia, Dujak, Davor (ed.) Osijek: Josip Juraj Strossmayer University of Osijek, Faculty of Economics and Business, pp. 317-330. [available at: https://blmm-conference.com/past-issues/, access November 3rd, 2023]
- GS1 (2023). GS1 Standards. [available at: https://www.gs1.org/standards, access October 27th, 2023]

- Kent, W. (1983). A Simple Guide to Five Normal Forms in Relational Database Theory, Communications of the ACM, vol. 26, pp. 120-125.
- W3schools (2023). XML Tutorial [available at: https://www.w3schools.com/xml/, access November 3rd, 2023]
- W3schools (2023). JSON - Introduction [available at: https://www.w3schools.com/js/js_json_intro.asp, access November 3rd, 2023]
- W3schools (2023). Database Normalization [available at: https://www.w3schools.in/DBMS/database-normalization/, access December 7th, 2023]