3. Upravljanje podacima

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

3.1 Informacije-Podaci-Znanje

U računalnim informacijskim sustavima prikaz informacija varira ovisno o njihovoj namjeni i upotrebi. Može biti alfanumerički ili binarni s obzirom na to predstavlja li tekst, sliku, zvuk ili izvršni program. Da bi mogao pohraniti, dohvatiti, obraditi i prenijeti podatke različitih vrsta (npr. brojeva, znakova, datuma, valuta itd.).) moraju biti ispravno kodirani. Alfanumerički formati predstavljanja podataka potječu 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 formati. Stoga omogućuju zajedničko tumačenje podataka od strane poslovnih partnera koji pripadaju različitim etničkim skupinama i geografskim sredinama. Dok alfanumerički nizovi ovise o njihovom kodiranju, numerički se podaci uglavnom razlikuju po veličini i/ili preciznosti.

Proces pretvaranja podataka iz izvornog analognog u digitalni oblik popularno se naziva digitalizacija. Odgovarajuće dizajnirani pomoćni i aplikacijski programi koji prihvaćaju podatke iz različitih izvora (npr. optički skeneri, električni senzori, EDI, itd.) omogućuju organizacijama da automatiziraju svoje prikupljanje podataka, pohranjivanje, obradu i prijenos unutar i između svojih računalnih informacijskih sustava.

Kada se prikupe, podaci različitih vrsta mogu se spojiti i organizirati u tablice podataka, baze podataka, skladišta podataka (kronološki redoslijed) ili baze znanja (konceptualni redoslijed). Više razine organizacije podataka omogućuju automatizirano razvrstavanje, zaključivanje i predstavljanje time akumuliranog znanja u analitičke svrhe.

3.2 Logistički podaci

U logistici se EDI koristi za prijenos transakcijskih 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 informacijski sustavi partnera (W3schools, 2023). Za brzu identifikaciju i manipulaciju osmišljeni su bar kodovi i RFID oznake.

Kako bi se omogućila međunarodna suradnja, trebalo je definirati 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 standardizirani i kodirani kao optički čitljivi bar-kodovi ili elektromagnetski čitljivi radiofrekvencijski identifikacijski (RFID) kodovi.

Bar-kod (EAN/UCC) je višesektorski i međunarodni oblik numeriranja stavki (POS EAN-8 i EAN-13, promjenjivi EAN-128, podatkovna traka, pakiranje ITF-14, QR, podatkovna matrica 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 primjene.

RFID kodovi se prvenstveno koriste na isti način kao bar kodovi. Oni jedinstveno identificiraju artikle ili usluge. Obično, osim izbornog crtičnog koda, RFID naljepnice nose još više informacija na čipu veličine glave pribadače. Osim identifikacije, RFID oznake omogućuju bilježenje podataka o praćenju, što je često potrebno u logističkim aplikacijama. Za razliku od crtičnih kodova, RFID omogućuje njihovo skeniranje bez izravnog vidnog polja, kao i skeniranje više naljepnica 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 izravnu, automatiziranu identifikaciju i praćenje logističkih jedinica unutar opskrbnog lanca.

EPCglobal standardi također predstavljaju osnovu GDSN-a (engl. *Global Data Synchronization Network*). Omogućuje automatizirano prikupljanje i razmjenu specifikacijskih podataka o

proizvodima i njihovoj ambalaži, čime se poduzećima omogućuje centralizirano upravljanje tim podacima kako bi ih oni i njihovi partneri naizmjenično koristili.

Tablica 3.1 sažima različite identifikacijske tehnologije s njihovim primjenama. Otkriva niz jednodimenzionalnih i dvodimenzionalnih bar kodova, kao i različite klase RFID kodova s njihovim mogućnostima.

Tablica 3.1Tehnologije označavanja.

Tehnologija	**Primjena**
Bar kod 1D	Maloprodajni artikli i komponente proizvoda
Bar kod 2D	Usluge (npr. UPS, zrakoplovne karte), veleprodajni artikli koji zahtijevaju praćenje
RFID klasa 1 (pasivno, R-oznake)	Predmeti koji zahtijevaju masovnu identifikaciju, kontrolu pristupa
RFID klasa 2 (pasivno, RW-tagovi)	Stavke koje zahtijevaju praćenje
RFID klasa 3 (poluaktivan, RW-tagovi)	Kontrola pristupa s dodanim 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 temeljene na lokaciji

Budući trendovi u označavanju, praćenju i sljedivosti slijede dva glavna smjera: minijaturizacija i raznolikost. Bar kodovi (1D) također moraju omogućiti označavanje minijaturnih predmeta (npr. medicinskih kapsula). Novi kodovi podatkovne matrice (2D) ne samo da će omogućiti ispravljanje pogrešaka tijekom skeniranja, već i šifriranje podataka.

RFID se nastavlja širiti na druga područja upotrebe kao što je identifikacija od strane pružatelja usluga (npr. željeznička kartica, prijava na posao, itd.), beskontaktna plaćanja (npr. bežični

prijenos novca, plaćanja na automatima) i pametna rješenja (npr. upravljanje pametnim domom, daljinski upravljani pametni uređaji itd.) kao i e-valute.

3.3 Organizacija podataka

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

Proračunske tablice

Prvi oblik organizacije podataka su dvodimenzionalni nizovi polja, koji se također nazivaju tablice ili proračunske tablice. Obično prvi redak proračunske tablice označava značenja vrijednosti pohranjenih u temeljnim stupcima, nakon čega slijede redovi podataka.

Polje ili ćelija tablice je najmanja jedinica podataka. Ima određeni tip (broj, datum, valuta itd.). Njegov sadržaj se može adresirati oznakama retka i stupca (npr. A1, koji predstavlja prvi redak stupca A).

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

Baze podataka

Datoteka ili tablica baze podataka zbirka je zapisa iste vrste. Baza podataka (engl. *Data Base* - DB) sastoji se od više međusobno povezanih tablica. 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 pohranjena u računalu

Iz gornje definicije mogu se izvući neki zaključci o klijentsko-poslužiteljskoj arhitekturi gdje poslužitelj drži DB, kojem pristupaju njegovi klijenti. Naravno, za pristup DB-u mora biti uspostavljena komunikacijska mreža između poslužitelja i njegovih klijenata. DB poslužitelj se obično naziva njegov "*back-end*", dok klijenti predstavljaju njegov "*front-end*". Sustav za upravljanje bazom podataka (engl. *Data Base Management System* - DBMS) na poslužitelju omogućuje svojim klijentima pristup podacima pohranjenim u DB-u putem svog aplikacijskog programskog sučelja (API) i DBM funkcija. DBM funkcije su mehanizmi koji omogućuju unos, dohvaćanje, obradu i prezentaciju podataka u DB-u. Za pozivanje ovih funkcija definirani su standardni jezici upita (engl. *Standard Query Languages* - SQL).

Model relacijske baze podataka

Postoje različiti oblici organizacije DB-a, a najčešći je relacijski model (RDB). Osnovna ideja iza ovog modela je činjenica da korisnik ne može unaprijed znati sve moguće upotrebe podataka pohranjenih u DB-u. Budući da obično ne postoje fiksni putevi pretraživanja kroz datoteke baze podataka, osmišljeni su različiti upitni jezici za dohvaćanje i manipulaciju podacima. RDB model temelji se na konceptu entiteta i odnosa:

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

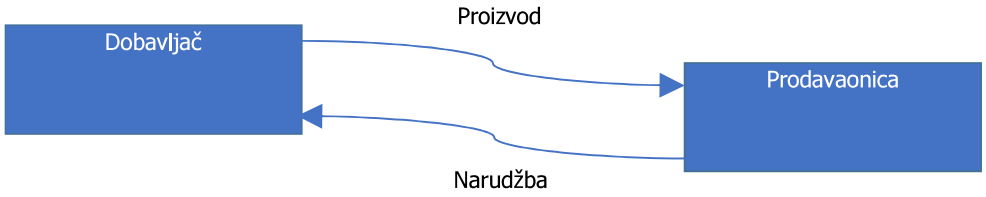
RDB tablice, koje predstavljaju entitete ili relacije, međusobno su povezane pomoću ključeva. Skup atributa koji jedinstveno identificiraju entitet naziva se njegov primarni ključ. Kada se primarni ključ pojavljuje kao polje u drugoj tablici s ciljem ispunjavanja relacije s izvornom tablicom, naziva se sekundarnim ili stranim ključem. Dok tablica može sadržavati samo jedan primarni ključ za jedinstvenu identifikaciju svojih zapisa, može sadržavati više sekundarnih ključeva.

Općenito, postoje dva pristupa konstruiranju RDB-a: analitički i sintetički.

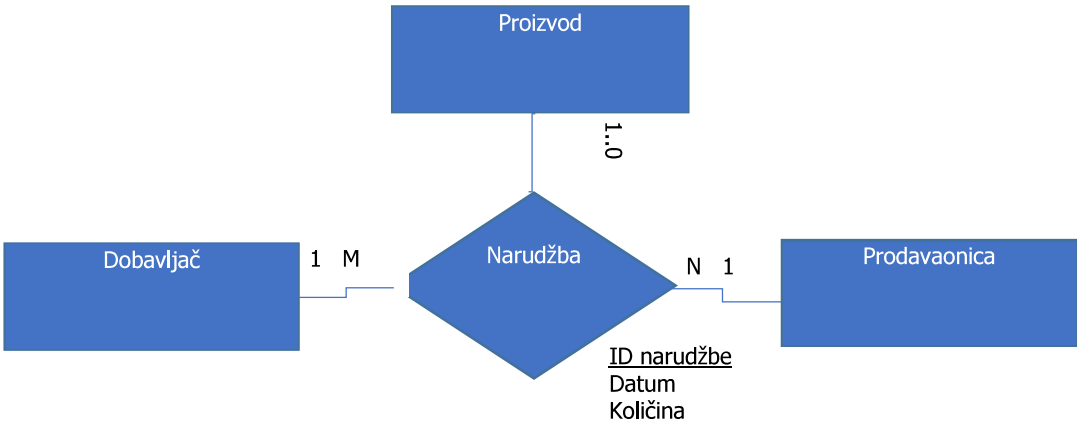
Analitički pristup izradi RDB-a sastoji se od sljedeća četiri koraka:

1. Analiza stvarnog svijeta - globalni model
2. Određivanje entiteta i odnosa – konceptualni model (npr. E-R dijagram)
3. Određivanje logičkog modela – relacijska shema
4. Izgradnja baze podataka (DBMS) – fizički model

Kako bismo ilustrirali pristup, razmotrimo primjer trgovačkog lanca i njegovih dobavljača (slika 3.1). Svaka prodavaonica ima više dobavljača. Svaki dobavljač može opskrbljivati različite prodavaonice. Ciklus nadopune započinje narudžbom iz prodavaonice. Zauzvrat dobavljač isporučuje robu u prodavaonicu. Narudžbe su transakcije u kojima se spajaju podaci o prodavaonici, dobavljaču i isporučenoj robi (slika 3.2).



Slika 3.1 Globalni model.



Slika 3.13 Konceptualni 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.

Logički model predstavlja RDB tablice prema vrstama njihovih zapisa. U logičkom modelu (slika 3.3) određeni atributi sadrže simbol ljestve (engl. *hashtag*) (#). To znači da polje koje oni predstavljaju jest ili pripada (kompozitnom) primarnom ključu. Neka su polja podcrtana.

Nazivaju se sekundarnim ili stranim ključevima, budući da referenciraju primarne ključeve povezanih tablica.

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

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

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

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

Tablica u 2NF je u 1NF. Dodatno, nijedan atribut ključa ne smije djelomično funkcionalno ovisiti o primarnom ključu. Ovime se svi ključevi koji jedinstveno identificiraju određene atribute čuvaju odvojeno. Ovo je uglavnom kako bi se osiguralo da se samo atributi koji ovise o svim (dijelovima) primarnog ključa čuvaju u jednoj tablici.

Tablica u 3NF je u 2NF. Osim toga, nijedan atribut koji nije ključ nije tranzitivno ovisan o primarnom ključu. To znači da se svi ne-ključni atributi koji mogu predstavljati ključ za određene druge ne-ključne atribute čuvaju u zasebnoj tablici, pri čemu se samo njihov ključ održava u izvornoj tablici kao strani ključ.

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

Tablica u Boyce-Codd NF (BCNF) je u 3NF; dodatno, svaka odrednica je ključ. Ovo uklanja sve odnose koji nisu obuhvaćeni postojećim redoslijedom ključeva iz izvorne tablice, čime se formiraju dodatne tablice za svaki ključ kandidata. Tablica u 4NF je u 3NF i BCNF. Osim toga, svaki atribut s više vrijednosti koji djelomično ovisi o ključu nalazi se u vlastitoj tablici. 4NF je namijenjen uklanjanju svih mogućih preostalih atributa s više vrijednosti iz originalne tablice.

Tablica je 5NF je u 4NF; dodatno, svaka JOIN operacija je predviđena ključevima. Tablica u 6NF je u 5NF; dodatno, uzimaju se u obzir sve netrivijalne JOIN-ovisnosti.

Može se uočiti da kod viših oblika organizacije broj tablica raste sa svakim korakom. Stoga je razumno promatrati fragmentaciju podataka kako bi se spriječilo stvaranje nepotrebnih tablica kojima se rijetko pristupa.

Tablica 3.2Primjer normalizacije RBD-a.

<table>
<tr><td>ONF

NARUDŽBA

• Dobavljač_ID *
• Ime_dobavljača
• Dobavljač_kontakt
• Prodavaonica_ID *
• Naziv_prodavaonice
• Adresa_prodavaonice
• ID_narudžbe*
• Datum
• EPC
• Količina
• Naziv_proizvoda
• Cijena_proizvoda</td><td>1NF

NARUDŽBA

• Dobavljač_ID *
• Ime_dobavljača
• Dobavljač_kontakt
• Prodavaonica_ID *
• Naziv_prodavaonice
• Adresa_prodavaonice
• ID_narudžbe*
• Datum
• EPC
• Količina
• Naziv_proizvoda
• Cijena_proizvoda</td></tr>
<tr><td colspan="2">* Ključevi kandidata za ponavljajuću grupu atributa, koji jedinstveno identificiraju narudžbu</td></tr>
<tr><td>2NF

DOBAVLJAČ

• Dobavljač _ID #
• Ime_dobavljača
• Dobavljač_kontakt

PRODAVAONICA

• Prodavaonica _ID #
• Naziv_prodavaonice
• Adresa_prodavaonice</td><td>

NARUDŽBA

• Narudžba_ID#
• Dobavljač_ID#
• Prodavaonica _ID #
• EPC
• Naziv_proizvoda
• Cijena_proizvoda
• Datum
• Količina</td></tr>
<tr><td>3NF

NARUDŽBA

• Dobavljač _ID #
• Prodavaonica _ID #
• Narudžba_ID#</td><td>

PROIZVOD

• EPC #
• Naziv_proizvoda</td></tr>
</table>

• Datum • Količina • ID_proizvoda	• Cijena_proizvoda

Upitni jezici

Uglavnom su dvije vrste: 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 izravno za upravljanje bazom podataka i skladištenje podataka.

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

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

Svaka manipulacija bazom podataka počinje njezinim stvaranjem. Rečenica

CREATE DATABASE *baza podataka_naziv* ;

stvara novu praznu bazu podataka s navedenim imenom.

Kao što je gore objašnjeno, podaci unutar baza podataka organizirani su u tablice zapisa podataka određenog tipa gdje svi redovi dijele zajedničku strukturu. Za izradu tablice koristi se sljedeća rečenica:

CREATE TABLE *tablica_naziv* (
stupac1 tip1,
stupac2 tip2, stupac3 tip3,
....);

Svaki imenovani stupac predstavlja atribut s određenim tipom podataka. Na primjer, u:

CREATE TABLE *Prodavaonica* (Prodavaonica _ *ID* int NOT NULL PRIMARY KEY,...);

CREATE TABLE *Narudžba* (Narudžba_ *ID int* NOT NULL PRIMARY KEY, ..., Product_Id int FOREIGN KEY REFERENCES *Proizvod* (*EPC*));

kreiraju se dvije tablice. Prva sadrži podatke o kupcima, dok druga sadrži podatke o njihovim narudžbama, pozivajući se na prvu tablicu preko broja kupca kao stranog ključa.

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

CREATE INDEX *indeksa_naziv* ON *tablica_naziv* (*naziv_stupca*);

Svaka manipulacija podacima na indeksiranoj tablici traje malo dulje, budući da za njezinu konzistentnost ne samo da treba provjeriti podatke koje daju ključevi i pravilno poredati podatke, već i druge atribute iz navedenog indeksa.

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

SELECT *stupac1, stupac2* , …
FROM *tablica_ naziv*;

Ovaj podatkovni upit vraća podatke u stupcu1, stupcu2 itd. iz tablice. Rečenice upita obično se formiraju pružanjem dodatnih opcija, filtriranjem podataka, ispunjavanjem navedenih uvjeta:

WHERE navodi uvjet koji određuje kriterije odabira zapisa.

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

HAVING specificira agregatne funkcije na grupama definiranim naredbama GROUP BY.

ORDER BY specificira atribute prema kojima su povratni zapisi poredani.

Na primjer:

```
SELECT "Prodavaonica" . " Prodavaonica_naziv " , "Proizvod" . " Proizvod_naziv " , "
Narudžba" . "Količina " FROM "Narudžba" , "Proizvod" , "Dobavljač" , " Prodavaonica "
WHERE "Narudžba" . " ID_proizvoda " = " Proizvod" . "EPC " AND "Narudžba" . "
Dobavljač_ID " = "Dobavljač" . "Dobavljač_ID" AND "Narudžba" . " Prodavaonica _ID" =
"Prodavaonica"."Store_ID" ORDER BY "Prodavaonica"."Prodavaonica_naziv" ASC
```

vraća popis prodavaonica s njihovim naručenim proizvodima i količinama, poredanih prema nazivu prodavaonice.

Najvažnija operacija u procesu selekcije je operacija JOIN. Često zamjenjuje uvjet WHERE kao JOIN ON, nakon čega slijedi uvjet. Uspoređuje vrijednosti stupaca i na temelju usporedbe određuje treba li ih uključiti u rezultat ili ne. U LEFT JOIN zapis se vraća ako su kriteriji ispunjeni u lijevoj tablici i obrnuto u operaciji RIGHT JOIN. Kao što je gore navedeno, uvjet mora biti ispunjen u obje tablice kako bi bio u skladu s operacijom INNER JOIN ili FULL JOIN. Budući da se potonji najčešće koristi , može se koristiti JOIN kao sinonim. Pozivajući se na uvjete 5 i 6 normalne forme, ovo je ista JOIN operacija, koju je potrebno ispuniti da bi se ispunili uvjeti odgovarajućeg NF-a.

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

INSERT INTO *tablica_naziv* (stupac*1* , [stupac*2* , ...])
VALUES (*vrijednost1* , [*vrijednost2* , ...]);

Da bi bile uspješne, vrijednosti u operaciji trebaju ispuniti sve uvjete atributa označenih nazivima stupaca. Ne treba navesti nazive stupaca u slučaju da su sve vrijednosti navedene. U slučaju da su u tablici predviđene neke DEFAULT vrijednosti, ne treba ih navoditi, osim ako se razlikuju.

Kada se podaci unesu, mogu se modificirati naredbom UPDATE :

UPDATE *tablica_naziv*
SET *stupac1=vrijednost1, stupac2=vrijednost 2* ,...
WHERE *neki_stupac = neka_vrijednost*;

U izjavi su dane nove vrijednosti za polja u navedenim stupcima. Kriteriji odabira retka označeni su specifikatorom WHERE, koji određuje sve vrijednosti stupca na koje se odnosi naredba UPDATE. Kako bi se spriječile neželjene promjene, potreban je dodatni oprez pri formuliranju kriterija odabira.

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

DELETE FROM *tablica_naziv*
WHERE *neki_stupac = neka_vrijednost*;

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

Naravno, upravljanje bazom podataka ne završava ovdje. Svaki element baze podataka također se može ukloniti, izmijeniti i/ili zamijeniti novim. U slučaju da se indeks, tablica ili baza podataka trebaju ukloniti, mogu se primijeniti sljedeće izjave:

DROP INDEX *indeks_ime* ON *iablic_ime*;

DROP TABLE *tablica_naziv*;

DROP DATABASE *baza podataka_naziv*;

Ako netko samo želi ukloniti podatke iz tablice, može se koristiti naredba TRUNCATE:

TRUNCATE TABLE *tablica_naziv* ;

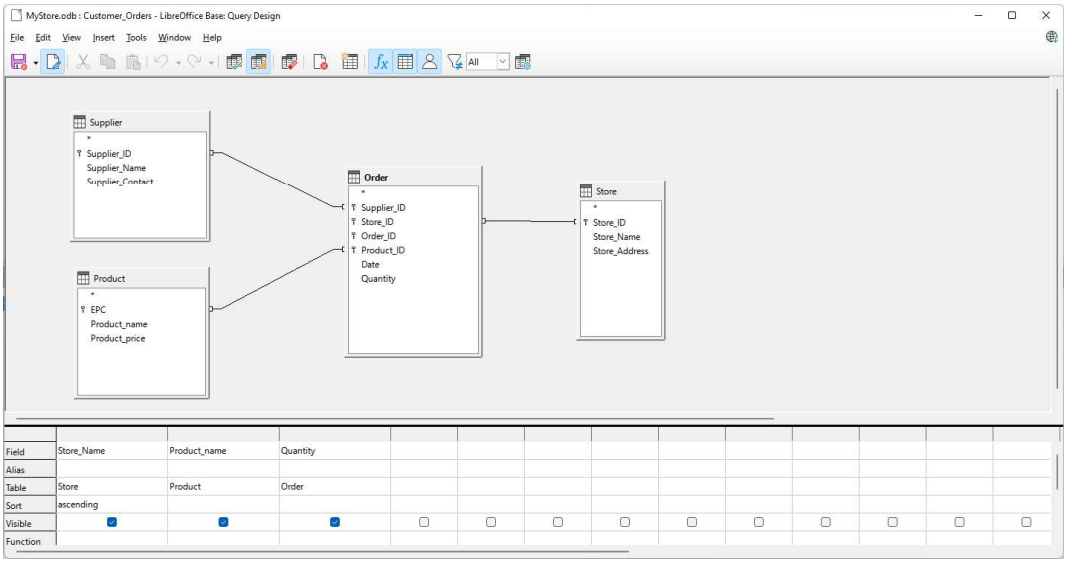
U slučaju da želite dodati ili ukloniti atribut (stupac) u/iz tablice, to možete učiniti naredbom ALTER:

ALTER TABLE *tablica_naziv* ADD stupac_*naziv vrsta podataka*;

ALTER TABLE *tablica_naziv* DROP COLUMN stupac_*naziv;*

Ovo zaključuje ovaj kratki pregled SQL jezika i njegovih najčešćih scenarija upotrebe. SQL se obično koristi u klijent-poslužitelj arhitekturama s DBMS-om koji je na glavnom računalu poslužitelja. Da bi mu se pristupilo, izdaju se SQL naredbe, bilo od strane klijentskog aplikacijskog programa ili web-sučelja DBMS poslužitelja.

S druge strane, QBE se također često koristi s relacijskim DBMS-om s grafičkim korisničkim sučeljem (engl. *Graphical User Interface* - GUI), kao što su MS Access ili LibreOffice Base. S QBE-om se baza podataka i njezine tablice stvaraju puno interaktivnije, a njihovu strukturu lakše je održavati. Kao što mu ime govori, nudi i jednostavniji oblik unosa i otkrivanja podataka. Za izvođenje pretraživanja potrebno je sastaviti sve tablice koje se koriste u pretraživanju i zatim uspostaviti uvjete kao uzorke u poljima stupaca kako bi se filtrirali relevantni podaci (Slika 3.4). Formulacija upita dopunjena je postojećim relacijama između tablica. Kao i obično, rezultat takvog upita je još jedna tablica s rezultirajućim podacima, koji se kasnije mogu dalje obraditi. Na ovaj način također se mogu formirati kaskadni ili višefazni upiti.



Slika 3.15QBE upit ekvivalentan gornjem SQL upitu.

Podatkovni filtri i maske

Kako bi se spriječili pogrešni ili nepotpuni podaci, koji bi mogli ometati njihovu obradu i tumačenje, potrebno je primijeniti dodatne mjere opreza:

1. Filtriranje praznih redaka i stupaca
2. Primjena snažnog tipiziranja podataka za sprječavanje računalnih pogrešaka
3. Definiranje maski za unos kako bi se spriječio unos pogrešnih podataka
4. Pristranost podataka kako bi se spriječili pogrešni rezultati

Prazni redovi i stupci čest su izvor pogrešaka koje uglavnom potječu od loših sučelja u aplikacijama za prikupljanje podataka. Otkazane ili nedovršene transakcije obično rezultiraju praznim redovima ili nedostajućim podacima u zapisnicima transakcija. Oni se samo djelomično mogu riješiti aplikacijama proračunskih tablica gdje se prazni reci i podaci koji nedostaju mogu otkriti provjerom ukupnog broja u odnosu na broj podataka koji nisu nula u recima/stupcima. Mogu se filtrirati uklanjanjem praznih redaka/stupaca, no to možda nije uvijek željena radnja jer bismo mogli izgubiti i neke vrijedne podatke. Najbolji način da se to spriječi je korištenjem DBMS-a koji bi s jedne strane omogućio unos samo kompletnih transakcijskih podataka, dok bi s druge strane također spriječio prazne redove/stupce, budući da ih u bazama podataka nema.

Slabo upisivanje podataka još je jedan uobičajeni izvor pogrešaka. Ako se umjesto numeričkih podataka unesu alfanumerički podaci, poput datuma ili iznosa valute, to bi rezultiralo greškama prilikom obrade tih podataka. U proračunskim tablicama, kao i u bazama podataka, pojedinačnim podatkovnim ćelijama, koje predstavljaju vrijednosti atributa u slogovima podataka, mogu se dodijeliti tipovi podataka, što bi nas alarmiralo pri unosu podataka u krivom formatu. Stoga se ovom mjerom može spriječiti da pogrešni podaci ometaju njihovu obradu.

Prilikom konstruiranja baza podataka, ograničenja se mogu primijeniti na podatkovna polja koja predstavljaju atribute entiteta ili relacije. Osim što im se može dodijeliti odgovarajuća vrsta, maske za unos mogu se definirati čime se omogućuje unos podataka, poput datuma, valuta, EAN kodova itd., samo u određenom formatu. To obično rješava mnoge pogrešne predodžbe koje bi inače mogle nastati tijekom obrade podataka.

Još jedan čest izvor pogrešaka su nepristrani podaci, koji predstavljaju podatke koji su reda veličine veći ili manji od očekivanog. Opet, mogli bi ometati našu obradu, dajući pogrešne rezultate. Teže ih je otkriti i mogu se filtrirati samo gledanjem podataka. U aplikacijama za proračunske tablice dobra uobičajena praksa bila bi određivanje minimalnih, maksimalnih i srednjih vrijednosti podataka u odgovarajućim stupcima kako bi se otkrila moguća odstupanja. Ako se otkriju, mogu se istaknuti i ručno obraditi, ako ih je malo, ili filtrirati i modificirati upitom u tablici baze podataka ako ih je puno. U svakom slučaju treba ih pažljivo procijeniti, 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 katalogiziraju kronološkim redom. Obično se na podacima provode neke poslovne analize, a analitički odjel također pohranjuje rezultate za kasnije potrebe. Nakon što se pohrane u skladište, ti se podaci obično ne mijenjaju kako bi se očuvala njihova dosljednost.

Osim kronološkog reda, prilikom izgradnje baza znanja uzimaju se u obzir i kontekstualni redovi. Ovdje su subjekti predstavljeni kao derivati entiteta najviše razine ili njegovih podružnica. Njihovi se odnosi uspostavljaju slobodnije jer su namijenjeni ažuriranju i nadogradnji kako se koriste. Uspostavljeni su u obliku pravila, temeljenih na svojstvima entiteta. Stoga je oblik u kojem su pohranjeni nešto drugačiji. Često se pohranjuju u obliku ontologija koje sadrže dublje znanje o prikupljenim podacima. Slično pohranjivanju rezultata

upita u skladištima podataka, upiti u bazama znanja također se pohranjuju za kasniju upotrebu za prikaz trenutnih rezultata, kako se mijenjaju entiteti, relacije i instance podataka.

Za razliku od baza podataka i skladišta, koji su specifični za aplikaciju, baze znanja mogu biti neovisne o aplikaciji i često ih različite aplikacije koriste među domenama. Primjer 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 pohranu podataka, prikazana je organizacija podataka i mehanizmi pronalaženja. Konačno, neke uobičajene pogreške u automatiziranoj obradi podataka su riješene kako bi svjesni čitatelj ostao na oprezu. Osim navedenih primjera, 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]