3. Upravljanje podatkov

Pri elektronski izmenjavi podatkov B2B (EDI) se med partnerji v oskrbovalni verigi izmenjujejo sporočila, ki vsebujejo oznake izdelkov ali storitev, identifikacije prevoznih enot in pravne dokumente. Običajno so v obliki alfanumeričnih nizov.

3.1 Informacije-podatki-znanje

V računalniško podprtih informacijskih sistemih se predstavitev informacij razlikuje glede na njihov namen in uporabo. Lahko je alfanumerična ali binarna, glede na to, ali predstavlja besedilo, sliko, zvok ali izvedljiv program ... Da bi lahko shranjevali, priklicali, obdelovali in prenašali podatke različnih vrst (npr. številke, znake, datume, valute itd.), jih je treba ustrezno kodirati. Alfanumerični formati za predstavitev podatkov izvirajo iz abecede ASCII (ask-key), ki se je razvila v smislu nacionalnih naborov znakov (npr. standardov 8859-1, Latin 1 in 8859-2, Latin 2) in nazadnje napredovala v mednarodne formate UTF-8 in UTF-16. Tako omogočajo skupno razlago podatkov s strani poslovnih partnerjev, ki pripadajo različnim etničnim skupinam in geografskim okoljem. Medtem ko so alfanumerični nizi odvisni od njihovega kodiranja, se numerični podatki razlikujejo predvsem po svoji velikosti in/ali natančnosti.

Postopek pretvorbe podatkov iz prvotne analogne v digitalno obliko se popularno imenuje digitalizacija. Ustrezno zasnovani uporabni in aplikativni programi, ki sprejemajo podatke iz različnih virov (npr. optični čitalniki, električni senzorji, EDI itd.), omogočajo organizacijam, da avtomatizirajo pridobivanje, shranjevanje, obdelavo in prenos podatkov znotraj svojih računalniško podprtih informacijskih sistemov in med njimi.

Ko so zbrani, se lahko podatki različnih vrst združijo in organizirajo v podatkovne tabele, podatkovne baze, podatkovna skladišča (kronološki vrstni red) ali baze znanja (konceptualni vrstni red). Višje ravni organizacije podatkov omogočajo avtomatizirano klasifikacijo, sklepanje in predstavitev tako zbranega znanja za analitične namene.

3.2 Logistični podatki

V logistiki se EDI uporablja za prenos podatkov o transakcijah med poslovnimi partnerji. Ker ti lahko uporabljajo različne jezike in aplikacije, je treba podatke pretvoriti v skupno obliko (npr. XML, JSON), da jih lahko interpretirajo informacijski sistemi različnih partnerjev (W3schools, 2023). Za hitro identifikacijo in manipulacijo so bile razvite črtne kode in oznake RFID.

Da bi omogočili mednarodno sodelovanje, je bilo treba za logistične namene opredeliti svetovno sprejete podatkovne formate. Logistični podatkovni formati ustrezajo oznakam storitev in izdelkov, identifikacijam prevoznih enot in kodam transakcij, ki so običajno v obliki alfanumeričnih nizov s časovnim žigom. Zaradi enostavnosti manipulacije in hitrosti obdelave so bile te kode standardizirane in kodirane kot optično berljive črtne kode ali elektromagnetno berljive kode za radiofrekvenčno identifikacijo (RFID).

Črtna koda (EAN/UCC) je več-sektorska in mednarodna oblika številčenja predmetov (POS EAN-8 in EAN-13, spremenljiva EAN-128, podatkovna vrstica, embalaža ITF-14, QR, podatkovna matrika itd.). Uporabljajo se za identifikacijo izdelkov, serij izdelkov ali pošiljk (kode 1D) in storitev (kode 2D). Med črtnimi kodami 2D je najbolj uveljavljena koda QR, ki jo je mogoče prebrati tudi s pametnimi telefoni, kar povečuje njihovo uporabnost na različnih področjih uporabe.

Kode RFID se uporabljajo predvsem na enak način kot črtne kode. Z njimi je mogoče enolično identificirati predmete ali storitve. Običajno je na nalepkah RFID poleg izbirne črtne kode še več informacij na čipu, velikem kot glavica pincete. Poleg identifikacije oznake RFID omogočajo beleženje informacij o sledenju, kar se pogosto zahteva v logističnih aplikacijah. V nasprotju s črtnimi kodami omogoča RFID njihovo skeniranje brez neposredne vidljivosti in tudi več etiket hkrati.

S standardom GS1 EPC Gen2 (ISO/IEC 18000-6:2013) je bil vzpostavljen tehnološki standard, ki določa komunikacijo med oznakami RFID in bralniki. Podobno kot pri črtnih kodah standardi EPCglobal povezujejo tehnologijo RFID z označevanjem izdelkov, logističnih transportnih enot, lokacij, zalog, vračljivih predmetov, dokumentov itd. z oznako EPC za neposredno, avtomatizirano identifikacijo in sledenje logističnih enot v oskrbovalnih verigah.

Standardi EPCglobal so tudi osnova za GDSN (Global Data Synchronization Network). Omogoča avtomatizirano pridobivanje in izmenjavo podrobnih podatkov o izdelkih in njihovi embalaži, s čimer podjetjem omogoča centralno upravljanje teh podatkov, ki jih lahko izmenično uporabljajo sama in njihovi partnerji.

V preglednici 3.1 so povzete različne tehnologije identifikacije in njihove uporabe. Razkriva različne eno- in dvodimenzionalne črtne kode ter različne razrede kod RFID z njihovimi zmogljivostmi.

Preglednica 3.1 Tehnologije označevanja.

Tehnologija	Aplikacija
Bar 1D	Maloprodajni izdelki in sestavni deli izdelkov
Bar 2D	storitve (npr. UPS, letalske vozovnice), blago na debelo, ki zahteva sledenje
RFID razreda 1 (pasivni, R-oznake)	Predmeti, ki zahtevajo množično identifikacijo, nadzor dostopa
RFID razreda 2 (pasivne, RW-oznake)	Predmeti, ki jih je treba spremljati
RFID razreda 3 (pol-aktivne, RW-oznake)	Nadzor dostopa z dodanimi informacijami o sledenju
RFID razred 4 (aktivni, RW-oznake)	Sledenje in sledenje v zaprtem prostoru
RFID razred 5 (aktivne oznake/bralci)	Sledenje in sledenje prostemu prostoru, storitve bližine z omogočenimi napravami, storitve, ki temeljijo na lokaciji.

Prihodnji trendi na področju označevanja, sledenja in izsleditve sledijo dvema glavnima smerema: miniaturizaciji in raznolikosti. Podatkovne črtne (1D) kode omogočajo tudi označevanje miniaturnih predmetov (npr. medicinskih kapsul). Nove matrične kode (2D) ne bodo omogočale le odpravljanja napak med skeniranjem, temveč tudi šifriranje podatkov.

RFID se še naprej širi na druga področja uporabe, kot so identifikacija s strani ponudnikov storitev (npr. železniška kartica, registracija na delovnem mestu itd.), brezstična plačila (npr. brezžični prenosi denarja, plačila na prodajnih avtomatih) in pametne rešitve (npr. upravljanje pametnega doma, daljinsko upravljanje pametnih naprav itd.) ter e-valute.

3.3 Organizacija podatkov

Poleg določene oblike so lahko podatki organizirani na različne načine, da se olajša njihovo upravljanje, obdelava in predstavitev. Čeprav so podatki na vhodu večinoma nestrukturirani, se z njihovim shranjevanjem, prenosom in obdelavo povečuje njihova organiziranost. V nadaljevanju so predstavljene običajne oblike organizacije podatkov z naraščajočo kompleksnostjo od pol-strukturiranih (npr. CSV) do strukturiranih (npr. preglednice, podatkovne zbirke itd.) oblik.

Preglednice

Prva oblika organizacije podatkov je z dvodimenzionalnimi polji, imenovanimi tudi tabele ali preglednice. Običajno prva vrstica preglednice označuje pomen vrednosti, shranjenih v osnovnih stolpcih, ki jim sledijo vrstice s podatki.

Polje ali celica tabele je najmanjša enota podatkov. Ima določeno vrsto (niz, številka, datum, valuta itd.). Njegovo vsebino je mogoče nasloviti z oznakami vrstic in stolpcev (npr. A1, ki predstavlja prvo vrstico stolpca A).

Vsaka vrstica tabele je skupina povezanih polj, ki predstavljajo zapis (npr.: transakcija, zapis študenta, podatki o izdelku itd.). Ker imajo vse vrstice tabele enako strukturo, lahko tip zapisa opredelimo kot seznam atributov (npr. podatki o študentu (ime, priimek, datum rojstva, kraj rojstva, ID ...)) ustreznih podatkovnih tipov.

Podatkovne zbirke

Datoteka ali tabela podatkovne zbirke je zbirka zapisov istega tipa. Podatkovna baza (DB) je sestavljena iz več medsebojno povezanih tabel. Tako je zbirka podatkov opredeljena v standardu ANSI:

- Podatki v DB so medsebojno povezani in razvrščeni.
- Podatke iz DB lahko hkrati uporablja več uporabnikov.
- Podatki v DB se ne ponavljajo
- DB je shranjena v računalniku.

Iz zgornje opredelitve lahko potegnemo nekaj zaključkov o arhitekturi odjemalec-strežnik, kjer ima strežnik DB, do katerega dostopajo odjemalci. Seveda je treba za dostop do DB vzpostaviti komunikacijsko omrežje med strežnikom in njegovimi odjemalci. Strežnik DB se običajno imenuje "zaledni del", medtem ko odjemalci predstavljajo njegov "sprednji del". Sistem za upravljanje

podatkovnih zbirk (DBMS) na strežniku omogoča svojim odjemalcem dostop do podatkov, shranjenih v DB, prek vmesnika za aplikacijske programe (API) in funkcij DBM. Funkcije DBM so mehanizmi, ki omogočajo vnos, pridobivanje, obdelavo in predstavitev podatkov v DB. Za priklic teh funkcij so bili opredeljeni standardni poizvedovalni jeziki (SQL).

Relativni model podatkovne baze

Obstajajo različne oblike organizacije DB, med katerimi je najpogostejši relacijski model (RDB). Osnovna ideja tega modela je dejstvo, da uporabnik ne more vnaprej poznati vseh možnosti uporabe podatkov, shranjenih v DB. Ker običajno ni fiksnih poti iskanja po datotekah zbirke podatkov, so bili za iskanje in manipulacijo s podatki razviti različni poizvedovalni jeziki. Model RDB temelji na konceptu entitet in relacij:

- Entiteta je oseba, stvar ali koncept, ki ga je mogoče enolično identificirati in ima atribute.
- Relacija predstavlja način povezovanja dveh ali več entitet.

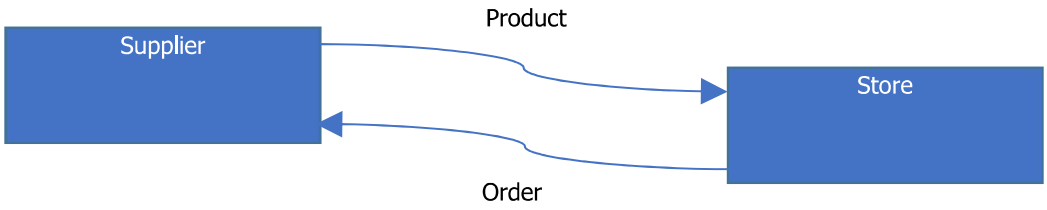
Tabele RDB, ki predstavljajo entitete ali relacije, so med seboj povezane s ključi. Nabor atributov, ki enoznačno identificirajo entiteto, se imenuje njen primarni ključ. Kadar se primarni ključ pojavi kot polje v drugi preglednici, da bi izpolnil razmerje s svojo prvotno preglednico, se imenuje sekundarni ali tuji ključ. Medtem ko lahko tabela vsebuje samo en primarni ključ za enolično identifikacijo svojih zapisov, lahko vsebuje več sekundarnih ključev.

Na splošno obstajata dva pristopa k izgradnji RDB: analitični in sintetični.

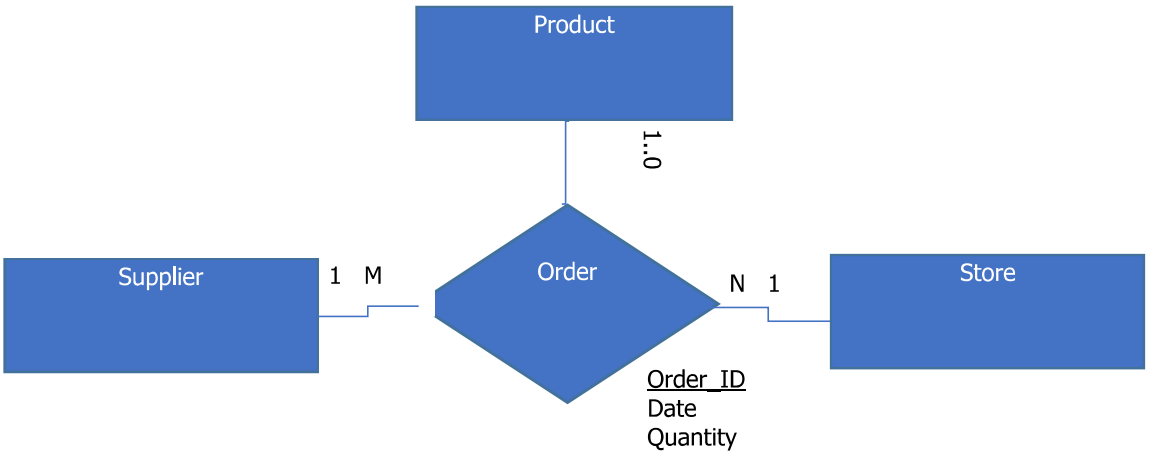
Analitični pristop k izdelavi RDB obsega naslednje štiri korake:

1. Analiza realnega sveta - globalni model
2. Določite entitete in odnose - konceptualni model (npr. E-R diagram)
3. Določitev logičnega modela - relacijska shema
4. Izdelava podatkovne zbirke (DBMS) - fizični model

Za ponazoritev pristopa si oglejmo primer verige trgovin s hitro prehrano in njihovih dobaviteljev (slika 3.1). Vsaka trgovina ima več dobaviteljev. Vsak dobavitelj lahko oskrbuje različne trgovine. Cikel dopolnjevanja zalog se začne z naročilom iz trgovine. V zameno dobavitelj dostavi blago v trgovino. Naročila so transakcije, v katerih so združeni podatki o trgovini, dobavitelju in dobavljenem blagu (slika 3.2).



Slika 3.1 Globalni model.



Slika 3.2 Konceptualni model.

SUPPLIER (Supplier_ID#, Supplier_name, Supplier_contact)

STORE (Store_ID#, Store_name, Store_address)

ORDER (Supplier_ID#, Store_ID#, Order_ID#, Date, Quantity, EPC)

PRODUCT (EPC#, Product_name, Product_price)

Slika 3.3 Logični model.

Logični model predstavlja tabele RDB z vrstami njihovih zapisov. V logičnem modelu (slika 3.3) nekateri atributi vsebujejo simbol hashtag (#). To pomeni, da je polje, ki ga predstavljajo, primarni (sestavljeni) ključ ali mu pripada. Nekatera polja so podčrtana. Imenujemo jih sekundarni ali tuji ključi, saj se sklicujejo na primarne ključe povezanih tabel.

Sintetični pristop k RDB obsega naslednje tri korake:

1. Analiza podatkov - seznam vseh ustreznih atributov
2. Določitev logičnega modela z normalizacijo - relacijska shema

3. Fizični model (DBMS)

Normalizacija ali kanonična sinteza (Kent, 1983) zagotavlja, da se s povratnim inženiringom iz ustreznih atributov oblikuje DB, ki izpolnjuje pogoje RDB (prim. sliko 4). Medtem ko začetna normalna oblika (0NF) atributov predstavlja tabelo neurejenih atributov, poznejše normalne oblike, kot jih opredeljuje (Codd, 1970), predstavljajo višje ravni organizacije podatkov. Lahko trdimo, da je z dosegom tretje normalne oblike dosežena shema, ki izpolnjuje zahteve za logični model RDB.

Tabela je v 1NF, če predstavlja relacijo. S tem je zagotovljeno, da se vse ponavljajoče se skupine podatkov hranijo ločeno in se torej ne ponavljajo.

Tabela v 2NF je v 1NF. Poleg tega noben atribut ključa ne sme biti delno funkcionalno odvisen od primarnega ključa. Pri tem so vsi ključi, ki enolično identificirajo določene atribute, ločeni. S tem se predvsem zagotovi, da so v eni tabeli samo atributi, ki so odvisni od vseh (delov) primarnega ključa.

Tabela v 3NF je v 2NF. Poleg tega noben atribut, ki ni ključ, ni prehodno odvisen od primarnega ključa. To pomeni, da so vsi neključni atributi, ki lahko predstavljajo ključ za nekatere druge neključne atribute, shranjeni v ločeni tabeli, pri čemer se v prvotni tabeli kot tuji ključ ohranja le njihov ključ.

Z normalizacijo od 1NF do 3NF dobimo logični model, ki ustreza predpisom relacijske podatkovne baze (RDB). Višje oblike normalizacije so namenjene predvsem optimizaciji modela RDB.

Tabela v Boyce-Codd NF (BCNF) je v 3NF; poleg tega je vsaka determinanta ključ. S tem se iz prvotne tabele odstranijo vse relacije, ki niso zajete z obstoječim vrstnim redom ključev, s čimer se za vsak kandidatni ključ oblikujejo dodatne tabele. Tabela v 4NF je v 3NF in BCNF. Poleg tega je vsak večvrednostni atribut, ki je delno odvisen od ključa, v svoji tabeli. Namen 4NF je odstraniti vse možne preostale večvrednostne atribute iz prvotne tabele. Tabela 5NF je v 4NF; poleg tega je vsaka operacija JOIN predvidena s ključi. Tabela v 6NF je v 5NF; poleg tega so upoštevane vse netrivialne odvisnosti JOIN.

Opazimo lahko, da se pri višjih oblikah organizacije število tabel povečuje z vsakim korakom. Zato je smiselno opazovati razdrobljenost podatkov, da se prepreči ustvarjanje nepotrebnih tabel, do katerih se dostopa le redko.

Preglednica 3.2 Primer normalizacije RBD.

<table>
<tr><td>0NF

NAROČILO

• Supplier_ID*
• Ime dobavitelja
• Dobavitelj_kontakt
• Store_ID*
• Ime_trgovine
• Store_address
• Order_ID*
• Datum
• EPC
• Količina
• Ime_izdelka
• Cena_izdelka</td><td>1NF

NAROČILO

• Supplier_ID#
• Ime dobavitelja
• Dobavitelj_kontakt
• Store_ID#
• Ime_trgovine
• Store_address
• Order_ID#
• Datum
• EPC
• Količina
• Ime_izdelka
• Cena_izdelka</td></tr>
<tr><td colspan="2">* Kandidatski ključi ponavljajoče se skupine atributov, ki enolično identificirajo naročilo</td></tr>
<tr><td>2NF

DOBAVITELJ

• Supplier_ID#
• Ime dobavitelja
• Dobavitelj_kontakt

TRGOVINA

• Store_ID#
• Ime_trgovine
• Store_address</td><td>

NAROČILO

• Order_ID#
• Supplier_ID#
• Store_ID#
• EPC
• Ime_izdelka
• Cena_izdelka
• Datum
• Količina</td></tr>
<tr><td>3NF

NAROČILO

• Supplier_ID#
• Store_ID#
• Order_ID#
• Datum
• Količina
• ID_izdelka</td><td>

IZDELEK

• EPC#
• Ime_izdelka
• Cena_izdelka</td></tr>
</table>

Jeziki poizvedb

Večinoma so dveh vrst: Strukturirani poizvedovalni jezik (SQL) in poizvedovanje po zgledu (QBE). Medtem ko SQL velja za programski jezik in API DBMS, se QBE večinoma uporablja neposredno z DBMS za upravljanje DB in skladiščenje podatkov.

Standard SQL (ISO/IEC 9075, 1986-2016) je programski jezik četrte generacije za manipulacijo s podatkovnimi bazami. Omogoča iskanje, dodajanje, spreminjanje in brisanje podatkovnih zapisov. Kljub njegovi standardizaciji obstajajo manjše razlike v njegovi implementaciji pri različnih sistemih za upravljanje podatkovnih zbirk (DBMS).

V nadaljevanju je jezik na kratko predstavljen z najpogostejšimi možnostmi. Po dogovoru so ključne besede SQL zapisane z velikimi črkami, vsak stavek pa se konča s podpičjem. Stavki so predstavljeni s povezavami do povezanih referenčnih gradiv, ki ponujajo dodatne informacije.

Vsaka manipulacija s podatkovno zbirko se začne z njenim ustvarjanjem. Stavek

CREATE DATABASE *Ime_podatkovne_baze*;

ustvari novo prazno zbirko podatkov z navedenim imenom.

Kot je opisano zgoraj, so podatki v zbirkah podatkov organizirani v tabelah podatkovnih zapisov določenega tipa, kjer imajo vse vrstice skupno strukturo. Za ustvarjanje tabele se uporablja naslednji stavek:

CREATE TABLE *ime_tabele* (
column1 type1
, column2 type2,
column3 type3,
....);

Vsak poimenovani stolpec predstavlja atribut z določenega podatkovnega tipa. Na primer, v:

CREATE TABLE *TRGOVINA* (*Store_ID* int NOT NULL PRIMARNI KLJUČ,...);

CREATE TABLE *NAROČILO* (*Order_ID* int NOT NULL PRIMARNI KLJUČ, ..., *Product_Id* int FOREIGN KEY REFERENCES *Product* (*EPC*));

ustvarjeni sta dve tabeli. Prva vsebuje podatke o strankah, druga pa podatke o njihovih naročilih in se sklicuje na prvo tabelo prek številke stranke kot tujega ključa.

Medtem ko so podatki v tabeli že razvrščeni po primarnem ključu, jih je mogoče dodatno razvrstiti po drugih atributih, če so indeksirani. Indeksiramo ga lahko tako, da ustvarimo indeks na danem atributu (atributih) z naslednjim stavkom:

CREATE INDEX *index_name* ON *table_name* (*column_name*);

Vsaka manipulacija s podatki v indeksirani tabeli traja nekoliko dlje, saj je treba za doslednost preveriti ne le podatkov iz ključev in jih ustrezno urediti, temveč tudi druge atribute iz določenega indeksa.

Najpogostejša operacija v zbirki podatkov je poizvedba podatkov, ki jo omogoča stavek SELECT:

SELECT *stolpec1, stolpec2*, ...
FROM *ime tabele*;

Ta podatkovna poizvedba vrne podatke iz tabele v stolpcu1, stolpcu2 itd. Stavki poizvedbe so običajno oblikovani z zagotavljanjem dodatnih možnosti, filtriranjem podatkov, izpolnjevanjem določenih pogojev:

WHERE določa pogoj, ki določa merila za izbor zapisov.

GROUP BY združi zapise, ki imajo skupno lastnost, ki omogoča združevalne funkcije.

Funkcija HAVING določa zbirne funkcije za skupine, opredeljene z izjavami GROUP BY.

ORDER BY določa atribute, na podlagi katerih so urejeni zapisi za vračilo.

Na primer:

```
SELECT "Store". "Ime_trgovine", "Izdelek". "Ime_izdelka", "Naročilo". "Količina" FROM "Naročilo",
"Izdelek", "Dobavitelj", "Trgovina" WHERE "Naročilo". "ID_izdelka" = "Izdelek". "EPC" IN
"Naročilo". "ID_dobavitelja" = "Dobavitelj". "ID_dobavitelja" IN "Naročilo". "ID_trgovine" =
"Trgovina". "ID_trgovine" ORDER BY "Store". "Ime_trgovine" ASC
```

vrne seznam trgovin z naročenimi izdelki in količinami, urejen po imenu trgovine.

Najpomembnejša operacija v postopku izbire je operacija JOIN. Pogosto se namesto pogoja WHERE uporablja operacija JOIN ON, ki ji sledi pogoj. Primerja vrednosti stolpcev in na podlagi primerjave določi, ali jih je treba vključiti v rezultat ali ne. Pri operaciji LEFT JOIN se vrne zapis, če so merila izpolnjena v levi tabeli, in obratno pri operaciji RIGHT JOIN. Kot je opisano zgoraj, mora biti pogoj izpolnjen v obeh tabelah, da je skladen z operacijo INNER JOIN ali FULL JOIN. Ker se slednja najpogosteje uporablja, lahko kot sopomenko uporabimo JOIN. Glede na pogoje normalnih oblik 5th in 6th gre za isto operacijo JOIN, ki mora biti izpolnjena, da je skladna s pogoji ustrezne NF.

Za vnos novih podatkov v tabelo se uporablja operacija INSERT INTO:

INSERT INTO *ime tabele (stolpec1*, [*stolpec2*, ...])
VALUES (*value1*, [*value2*, ...]);

Da bi bila operacija uspešna, morajo vrednosti v operaciji izpolnjevati vse pogoje atributov, ki jih označujejo imena stolpcev. Če so navedene vse vrednosti, imen stolpcev ni treba navesti. Če so v tabeli predvidene nekatere vrednosti DEFAULT, jih ni treba navesti, razen če so drugačne.

Ko so podatki vneseni, jih lahko spremenite z ukazom UPDATE:

UPDATE ime_tabele
SET *column1=vrednost1, column2=vrednost2*,...
WHERE *stolpec=neka_vrednost*;

V izjavi so podane nove vrednosti za polja v navedenih stolpcih. Kriteriji za izbiro vrstic so označeni s specifikatorjem WHERE, ki določa vse vrednosti stolpcev, na katere se nanaša izjava UPDATE. Da bi preprečili neželene spremembe, je treba biti pri oblikovanju izbirnih meril še posebej previden.

Z operacijo DELETE lahko iz tabele izbrišete zapis ali več zapisov:

DELETE FROM *ime* tabele
WHERE *some_column=some_value*;

Tako kot pri stavku UPDATE se za določitev vseh vrstic, ki jih je treba izbrisati, uporabi določilo WHERE.

Upravljanje podatkovne zbirke se tu seveda ne konča. Vsak element zbirke podatkov lahko tudi odstranite, spremenite in/ali nadomestite z novim. Če je treba odstraniti indeks, tabelo ali zbirko podatkov, lahko uporabite naslednje stavke:

DROP INDEX *index_name* ON *table_name*;

DROP TABLE *ime_tabele*;

DROP DATABASE *podatkovne baze ime_podatkovne baze*;

Če želite iz tabele odstraniti le podatke, lahko uporabite stavek TRUNCATE:

TRUNCATE TABELE *ime_tabele*;

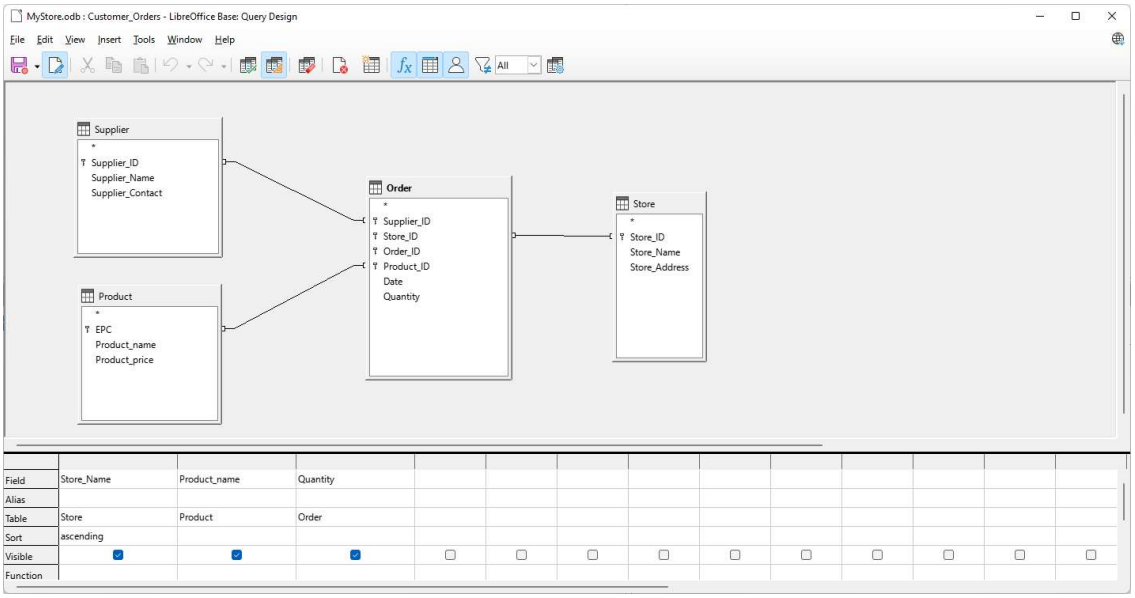
Če želimo dodati ali odstraniti atribut (stolpec) v/iz tabele, lahko to storimo z ukazom ALTER:

ALTER TABLE *table_name* ADD *column_name datatype*;

ALTER TABLE *table_name* DROP COLUMN *column_name*;

S tem zaključujemo kratek pregled jezika SQL in najpogostejših scenarijev njegove uporabe. Jezik SQL se pogosto uporablja v arhitekturah odjemalec-strežnik, kjer je DBMS nameščen v strežniku. Za dostop do nje se izdajajo stavki SQL, bodisi s programom odjemalske aplikacije bodisi s spletnim vmesnikom strežnika DBMS.

Po drugi strani pa se QBE pogosto uporablja tudi z relacijskimi DBMS z grafičnim uporabniškim vmesnikom (GUI), kot sta MS Access ali LibreOffice Base. S QBE se zbirka podatkov in njene tabele ustvarjajo veliko bolj interaktivno, njihova struktura pa se lažje vzdržuje. Kot pove že njeno ime, ponuja tudi preprostejšo obliko vnosa in odkrivanja podatkov. Za izvedbo iskanja je treba sestaviti vse tabele, ki se uporabljajo pri iskanju, nato pa določiti pogoje kot vzorce v poljih stolpcev za filtriranje ustreznih podatkov (slika 3.4). Formulacija poizvedbe je dopolnjena z obstoječimi razmerji med tabelami. Kot običajno je rezultat take poizvedbe druga tabela z dobljenimi podatki, ki jih je mogoče pozneje dodatno obdelati. Na ta način se lahko oblikujejo tudi kaskadne ali večfazne poizvedbe.



Slika 3.4 Poizvedba z zgledom, enakovredna zgornji poizvedbi SQL.

Filtri in maske podatkov

Da bi preprečili napačne ali nepopolne podatke, ki bi lahko vplivali na njihovo obdelavo in razlago, je treba uporabiti dodatne previdnostne ukrepe:

1. Filtriranje praznih vrstic in stolpcev
2. Uporaba strogega tipiziranja podatkov za preprečevanje računskih napak.
3. Opredelitev vhodnih mask za preprečevanje vnosa napačnih podatkov

4. Prilagajanje podatkov za preprečevanje napačnih rezultatov

Prazne vrstice in stolpci so pogost vir napak, ki je predvsem posledica slabih vmesnikov v aplikacijah za zbiranje podatkov. Preklicane ali nepopolne transakcije običajno povzročijo prazne vrstice ali manjkajoče podatke v dnevnikih transakcij. Le delno jih je mogoče obravnavati z aplikacijami preglednic, kjer je mogoče prazne vrstice in manjkajoče podatke odkriti s preverjanjem skupnega števila v primerjavi s številom neničelnih podatkov v vrsticah/stolpcih. Lahko jih filtriramo tako, da odstranimo prazne vrstice/stolpce, vendar to ni vedno želeno dejanje, saj lahko izgubimo tudi nekaj dragocenih podatkov. To lahko najbolje preprečimo tako, da uporabimo sistem DBMS, ki bi na eni strani dovoljeval vnos le popolnih podatkov o transakcijah, na drugi strani pa bi preprečeval tudi vnos praznih vrstic/stolpcev, saj jih v podatkovnih zbirkah ni.

Slabo tipiziranje podatkov je še en pogost vir napak. Če namesto številčnih podatkov vnesete alfanumerične podatke, na primer datum ali znesek v valuti, lahko pride do napak pri obdelavi teh podatkov. Tako v preglednicah kot v podatkovnih zbirkah lahko posameznim podatkovnim celicam, ki predstavljajo vrednosti atributov v podatkovnih zapisih, pripišemo podatkovne tipe, kar bi nas ob vnosu podatkov v napačni obliki alarmiralo. Tako lahko s tem ukrepom preprečimo, da bi napačni podatki ovirali njihovo obdelavo.

Pri oblikovanju podatkovnih zbirk lahko za podatkovna polja, ki predstavljajo atribute entitete ali relacije, uporabimo omejitve. Poleg tega, da se jim dodeli ustrezen tip, se lahko določijo maske za vnos, ki omogočajo vnos podatkov, kot so datumi, valute, kode EAN itd., samo v določeni obliki. S tem se običajno odpravijo številne napake, do katerih bi sicer lahko prišlo pri obdelavi podatkov.

Drug pogost vir napak so nepristranski podatki, ki predstavljajo podatke, ki so za red velikosti večji ali manjši od pričakovanih. Tudi ti lahko motijo našo obdelavo in povzročijo napačne rezultate. Težje jih je odkriti in jih je mogoče izločiti le s pregledom podatkov. Pri uporabi preglednic je dobra običajna praksa, da določimo najmanjše, največje in srednje vrednosti podatkov v ustreznih stolpcih, da bi odkrili morebitna odstopanja. Če odkrijemo le malo takih primerov, jih lahko poudarimo in obravnavamo ročno. Če jih je veliko, pa jih lahko filtriramo in spremenimo s poizvedbo v tabeli podatkovne zbirke. V vsakem primeru jih je treba skrbno oceniti, da ne bi še poslabšali stanja, saj bi bilo v tem primeru boljše te podatke odstraniti.

Podatkovna skladišča in baze znanja

Podatki v podatkovnih skladiščih so zbrani iz RDB in katalogizirani v kronološkem vrstnem redu. Običajno se na podatkih opravijo nekatere poslovne analize, rezultate pa analitični oddelek shrani

tudi za poznejše sklicevanje. Ko so ti podatki shranjeni v skladišču, se običajno ne spreminjajo, da se ohrani njihova doslednost.

Poleg kronološkega vrstnega reda se pri gradnji baz znanja upoštevajo tudi kontekstualni redi. Pri tem so entitete predstavljene kot izpeljanke entitete najvišje ravni ali njene podrejene entitete. Njihova razmerja se vzpostavljajo bolj svobodno, saj so namenjena posodabljanju in nadgrajevanju ob njihovi uporabi. Vzpostavljeni so v obliki pravil, ki temeljijo na lastnostih entitet. Zato je oblika, v kateri so shranjeni, nekoliko drugačna. Pogosto so shranjeni v obliki ontologij, ki vsebujejo poglobljeno znanje o zbranih podatkih. Podobno kot pri shranjevanju rezultatov poizvedb v podatkovnih skladiščih so tudi poizvedbe v bazah znanja shranjene za poznejšo uporabo za prikaz trenutnih rezultatov, saj se entitete, razmerja in primerki podatkov spreminjajo.

V nasprotju s podatkovnimi bazami in skladišči, ki so vezani na posamezno aplikacijo, so lahko baze znanja neodvisne od aplikacije in se pogosto uporabljajo v različnih aplikacijah. Primer je predstavljen v (Gumzej et. al., 2023).

3.4 Zaključek

V tem poglavju so bili obravnavani različni vidiki upravljanja podatkov v logistiki. Poleg predstavitev podatkov in standardov za shranjevanje podatkov so bili predstavljeni tudi mehanizmi za organizacijo in pridobivanje podatkov. Na koncu so bile obravnavane nekatere pogoste napake pri avtomatizirani obdelavi podatkov, da bi bil zaveden bralec pozoren. Poleg naštetih primerov lahko v pripadajočem učnem gradivu odkrijete še več.

LITERATURA POGLAVJE 3

- 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]