2. Podatkovne baze

- Uvod v podatkovne baze
- Predstavitev podatkov
- Relacijski podatkovni model
- Povpraševalni jezik SQL
- Modeliranje podatkov

2.1 Uvod v podatkovne baze

- Podatkovna osnova informacijskih sistemov v podjetjih
  - uspešna integracija mora reševati tri ključne naloge:
    - zagotavljanje pravih informacij, ob pravem času, na pravem mestu
  - nosilci informacij so podatki
    - brez zmožnosti za upravljanje z velikimi količinami podatkov in zmožnosti za hitro iskanje ustreznih podatkov je velika količina podatkov bolj breme kot korist
    - zaradi preveč informacij potrebujemo še več informacij
  - potrebujemo ustrezne mehanizme za upravljanje s podatki in učinkovito iskanje po njih → podatkovne baze

Kaj je podatkovna baza?

- angl. *data base* oz. *database*
- Podatkovna osnova (baza)
  - omogoča izvajanje različnih uporabniških programov, ki morajo od nekod črpati podatke za svoje izvajanje
  - opis okolja s pomočjo podatkov
  - temelj, ki omogoča komunikacijo/interakcijo z okoljem
- Podatki so organizirani na način, ki je
  - čim bolj neodvisen od programov, ki jih uporabljajo
  - čim bliže človeškemu razumevanju

Podatek in informacija

- Podatek je na zgoščen način zapisano dejstvo
- Različne definicije
  - poljubna predstavitev s pomočjo simbolov ali analognih veličin, ki ji je pripisan, ali se ji lahko pripiše nek **pomen**
  - predstavitev dejstva, koncepta ali instrukcije na **formaliziran** način, ki je primeren za komunikacijo, interpretacijo ali obdelavo s strani človeka ali stroja
  - dejstva, predstavljena z vrednostmi (številke, znaki, simboli), ki imajo pomen v določenem **kontekstu**

Podatek in informacija /2

LMSV LAMS

- Informacija
  - pomen, ki ga človek pripiše podatkom s pomočjo znanih konvencij, uporabljenih pri njihovi predstavitvi
  - ovrednoteni podatki v specifični situaciji
  - novo spoznanje, ki ga človek doda svojemu poznavanju sveta
- Zveza med podatki in informacijo
  - podatki niso informacija in ne vsebujejo informacije
  - podatki lahko posredujejo informacijo
  - če je podatkov preveč, da bi jih v ustreznem času interpretirali, ne posredujejo nobene informacije

Sistem za obdelavo podatkov

LMSV LAMS

- Štiri osnovne komponente
  - človek
  - podatki
  - program
  - računalnik
- Pri prvih tovrstnih sistemih v ospredju računalnik, kasneje program, danes podatki
  - podatkovna revolucija
  - strukturiranje, različne predstavitve za različne uporabe
  - shranjevanje podatkov o podatkih - bliže človeku
  - neodvisnost podatkov od strojne in programske opreme
  - središče obdelave je podatkovna baza

Zgodovina shranjevanja podatkov (vir: M. Bajec, FRI)

- Zgodnja 60' leta 20. stol: Charles Bachman iz podjetja General Electric razvije prvi splošno-namenski SUPB* (Integrated Data Store – IDS)
  - predstavlja osnovo za mrežni podatkovni model, ki je predlagan za standard v okviru konzorcija za jezike podatkovnih sistemov (CODASYL)
  - za mrežni podatkovni model Bachman prejme Turingovo nagrado (ACM Turing Award – na področju računalniških znanosti ekvivalent Nobelove nagrade)
  - ima velik vpliv na razvoj SUPB v 60' letih.

*SUPB – Sistem za upravljanje s podatkovno bazo

Zgodovina shranjevanja podatkov

- Pozna 60' leta 20. stol: IBM razvije Information Management System (IMS), ki se ponekod uporablja še danes
  - predstavlja osnovo za hierarhični podatkovni model.
  - American Airlines in IBM razvijeta sistem SABRE za rezervacije letalskih kart – sistem omogoča več uporabnikom dostop do skupnih podatkov preko mreže
  - zanimivost: isti SABRE še danes uporablja spletna potovalna agencija Travelocity.

Zgodovina shranjevanja podatkov

- 70’ leta 20. stol: Edgar Codd (IBM) predlaga relacijski podatkovni model
  - razvije se mnogo relacijskih SUPB
  - podatkovne baze postanejo akademsko področje – razvije se izjemno močna teoretična podlaga
  - Codd dobi Turingovo nagrado za svoje delo
  - relacijske podatkovne baze postanejo standard za upravljanje s podatki v organizacijskih sistemih

Zgodovina shranjevanja podatkov

- 80’ leta 20. stol: Relacijski model si še utrdi položaj kot SUPB
  - razvije se poizvedovalni jezik SQL (projekt IBM System R)
  - SQL postane standardni jezik za izvajanje poizvedb v relacijski podatkovni bazi
  - SQL je bil standardiziran v poznih 80’ letih – SQL-92.
  - standard prevzamejo American National Standard Institute (ANSI) in International Standards Organization (ISO)
  - skrb za sočasen dostop do podatkov prevzame SUPB; programerji programirajo, kot bi do podatkov dostopali samo oni; Jim Grey dobi za dosežke na tem področju Turingovo nagrado.

Zgodovina shranjevanja podatkov

- Pozna 80' in 90' leta 20. stol: veliko napredka na področju podatkovnih baz
  - veliko raziskav se opravi na področju poizvedovalnih jezikov in bogatejših (razširjenih) podatkovnih modelov
  - velik poudarek na kompleksnih analizah podatkov iz vseh področij organizacijskih sistemov
  - veliko proizvajalcev SUPB-jev (IBM – DB2, Oracle 8, Informix UDS) razširi svoje sisteme s podporo novim podatkovnim tipom: slike, tekst in s podporo kompleksnejšim poizvedbam

Zgodovina shranjevanja podatkov

- Pozna 80' in 90' leta 20. stol: nadaljevanje
  - Pojavijo se podatkovna skladišča, ki združujejo podatke iz več podatkovnih baz in omogočajo izvajanje specializiranih analiz (iskanje zakonitosti v podatkih)
  - pojavijo se orodja ERP
    - podpirajo skupne funkcije v poslovnih sistemih (npr. skladiščno poslovanje, planiranje človeških virov, finančne analize,...)
    - predstavljajo obsežno aplikacijsko plast nad skupno podatkovno bazo
    - primeri: Oracle, SAP, Baan, PeopleSoft in Siebel

Zgodovina shranjevanja podatkov

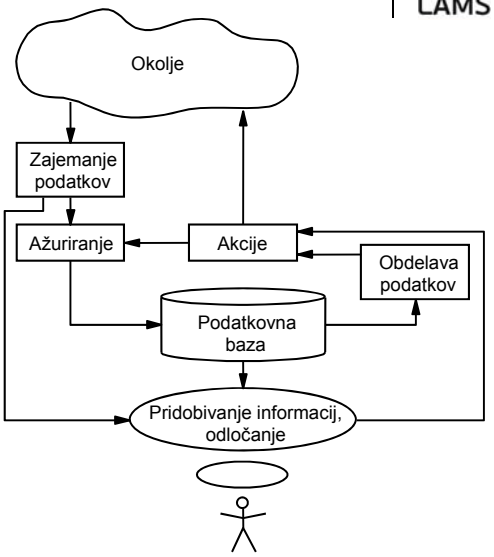
- Naslednja stopnja: vstop SUPB v svet Interneta
  - prva generacija spletnih mest shranjuje podatke v datotekah operacijskega sistema; uporaba podatkovnih baz za shranjevanje podatkov, ki so dostopni preko Interneta, postaja vsakdanja
    - poizvedbe se generira preko spletnih form, odgovore pa se nazaj posreduje v obliki jezika HTML za lažji prikaz v spletnem brskalniku
  - vsi proizvajalci dodajajo svojim SUPB možnosti za čim lažjo uporabo v spletu

Zgodovina shranjevanja podatkov

- Najnovejša področja:
  - multimedijske podatkovne baze
  - podatkovne baze za interaktivni video
  - digitalne knjižnice
  - odločitveni sistemi, vrtanje po podatkih (rudarjenje podatkov), odkrivanje zakonitosti
  - NoSQL

Podatkovna baza

- Model (slika) okolja, ki služi kot osnova za sprejemanje odločitev in izvajanje akcij
- Podpira interakcije med človekom (podjetjem) in okoljem



Uporaba in upravljanje podatkovne baze

- Uporabniki podatkovne baze so lahko:
  - posredni,
  - neposredni (komunicirajo s podatkovno bazo neposredno preko terminalov)
- Obstaja več vrst neposrednih uporabnikov:
  - upravitelj podatkovne baze,
  - sistemski programerji,
  - aplikacijski programerji,
  - uporabniki povpraševalnega jezika,
  - menijsko vodeni uporabniki,
  - parametrični uporabniki (dostopajo do baze s pomočjo programov, ki so napisani v višjih programskih jezikih).

Upravljanje podatkovne baze

- S tehničnega vidika
  - podatkovna baza je baza med seboj pomensko povezanih podatkov, ki so shranjeni v računalniškem sistemu, dostop do njih je centraliziran in omogočen s sistemom za upravljanje podatkovne baze
- Upravljanje obsega
  - zagotavljanje razpoložljivosti podatkov,
  - nadzor uporabe podatkov, to se pravi skrb za
    - celovitost podatkov – obnavljanje baze, nadzor nad sočasnim dostopom, preverjanje vhodnih podatkov
    - uporabo podatkov v skladu z njihovim namenom – up. pravice
    - uporabnost podatkov v prihodnosti – večnivojska zasnova

Upravljanje podatkovne baze

- Sistem za upravljanje s podatkovno bazo – SUPB
  - programska oprema, ki opravlja nalogo posrednika med uporabniki in podatkovno bazo
  - je programska oprema za obvladovanje velikih količin podatkov
- Obstaja veliko vrst SUPB
  - relacijski, objektni, XML,…
  - primeri SUPB: Oracle, Sybase, DB2, MS SQL Server, Ingres, Postgres, MySQL, ObjectStore, Jasmine, Objectivity/DB, Versant Object Database,...
- Alternativa – shranjevanje v aplikaciji lastni obliki; problemi: neprenosljivost idr.

Upravljanje podatkovne baze

- Funkcije SUPB:
  - nadzorne: zaščita celovitosti in performančni nadzor
  - dostopne, ki jih lahko naprej delimo na:
    - gradnjo, to je kreiranje shem, kreiranje fizične podatkovne baze in reorganizacijo podatkovne baze,
    - uporabo, to je zajemanje podatkov iz podatkovne baze, ažuriranje podatkovne baze in podporo pri razvoju podatkovne baze
- Prednosti uporabe SUPB:
  - podatkovna neodvisnost: programi so neodvisni od predstavitve podatkov in načina shranjevanja podatkov; SUPB zagotavlja abstrakcijo podatkov in ločuje programe od podrobnosti predstavitve podatkov

Upravljanje podatkovne baze

- Prednosti uporabe SUPB (nadaljevanje):
  - učinkovit dostop do podatkov: SUPB zagotavlja tehnike za učinkovito hranjenje in dostop do podatkov
  - varnost in integriteta podatkov: če do podatkov dostopamo preko SUPB, je možno definirati omejitve, ki zagotavljajo skladnost podatkov
  - administracija podatkov: lažje upravljanje s centralno shranjenimi podatki
  - sočasen dostop do podatkov in obnavljanje podatkovne baze: SUPB razporeja sočasne dostope tako, da je videti, kot da do podatkov dostopa en uporabnik
  - skrajšan čas razvoja programov: SUPB podpira številne mehanizme za dostop do podatkov

Moderni koncepti relacijskih podatkovnih baz

- Podatkovni strežnik
  - računalniški sistem, ki je namenjen zgolj upravljanju podatkovne baze
- Predvsem se srečujemo z dvema konceptoma
  - koncept odjemalec/strežnik
    - funkcije ločimo na dve skupini: strežne funkcije in funkcije odjemalca
  - porazdeljena podatkovna baza
    - porazdelitev baze na več podatkovnih strežnikov

Koncept odjemalec/strežnik

- Vključuje dva procesa, ki tečeta vsak na svojem ali pa na istem računalniku
  - oba procesa sta podprta s sistemskimi servisi operacijskega sistema in komunikacijske programske opreme
  - odjemalčev proces pošilja strežnikovemu procesu zahteve po določenih storitvah
  - strežnik se odziva s poročili o uspehu
  - tipično nudi svoje usluge večjemu številu odjemalcev.
  - strežnik je lahko en sam, ali pa en primarni (master) strežnik, ki deli naloge sekundarnim (slave) strežnikom.

Koncept porazdeljene podatkovne baze

- baza je logično povezana, a fizično porazdeljena po različnih podatkovnih strežnikih.
- Prednosti porazdeljene podatkovne baze so:
  - aplikacije so po naravi porazdeljene in je taka arhitektura samoumevna,
  - večja zanesljivost in dostopnost podatkov ob vsakem trenutku,
  - paralelno delovanje procesov, kar povečuje hitrost.
- Glavna slabost porazdeljenih podatkovnih baz je njihova kompleksnost.

2.2 Organizacija podatkovne baze

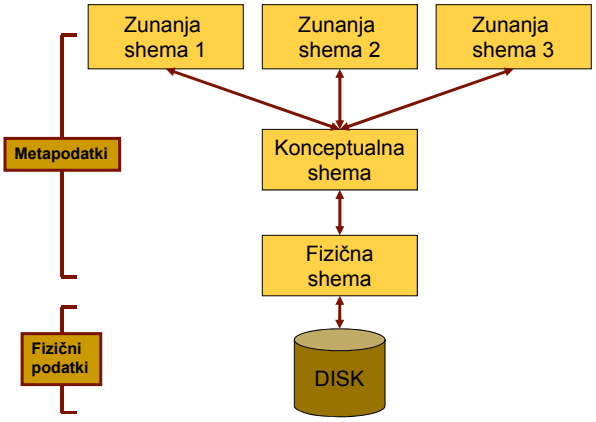
- Podatkovna baza v ožjem smislu:
  - podatki, ki jih le-ta vsebuje.
- Podatkovna baza v širšem smislu je sistem, ki ga sestavljajo:
  - podatki,
  - uporabniki in uporabniški programi,
  - upravitelj podatkovne baze,
  - sistem za upravljanje podatkovne baze.
- Podatkovni del baze
  - fizična podatkovna baza - vrednosti elementov
  - metapodatkovna baza - opisi fizičnih podatkov

Organizacija podatkovne baze

- Obe bazi sta shranjeni v zunanjem pomnilniku v obliki fizičnih datotek v datotečnem sistemu, ki ga podpira operacijski sistem.
- Metapodatkovna baza ima tri vrste opisov fizičnih podatkov (glede na stopnjo abstrakcije):
  - notranja shema – pove, **kako so podatki shranjeni**
  - konceptualna shema - podaja način modeliranja okolja (tipe podatkov, medsebojne relacije, omejitve), pove, **kateri podatki so v bazi**
  - zunanja shema - podaja način uporabe podatkov in predstavlja **uporabniku lasten pogled**

Metapodatkovna baza

- Tri-nivojski opis podatkov



Fizična podatkovna baza

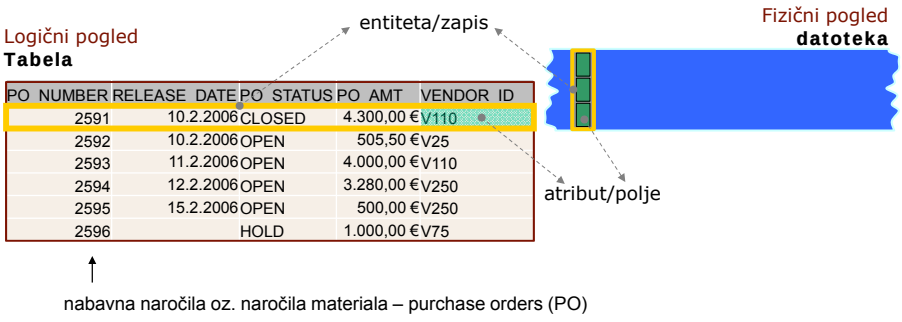
LMSV LAMS

- Fizična podatkovna baza ima štiri nivoje:
  - nivo operacijskega sistema: baza **fizičnih datotek**,
  - notranji nivo: baza fizičnih datotek se prikaže kot baza **logičnih zapisov** različnih tipov in njihovih medsebojnih povezav,
  - konceptualni nivo: baze logičnih zapisov (in njihovih povezav) se s pomočjo opisov v konceptualni shemi kažejo kot **imena, lastnosti in povezave entitet**,
  - zunanji nivo: konceptualna baza se s pomočjo opisov v zunanji shemi prikaže kot **uporabnikov model okolja**.

Logični in fizični pogled

LMSV LAMS

- Uporabnik zahteva od SUPB zapis naročila “2591”:
  - SUPB naredi preslikavo logičnega zapisa v fizični;
  - poišče fizični zapis in ga prepiše v primarni pomnilnik oziroma v medpomnilnik;



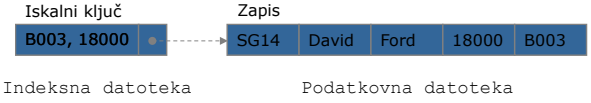
Indeksiranje

- Indeks je podatkovna struktura, ki SUPB omogoča hitrejše lociranje zapisov v datoteki

- Analogija z indeksom knjige
  - indeks knjige vsebuje ključne besede (urejene po abecedi) ter za vsako pove, na kateri strani ali straneh se ključna beseda pojavlja.
  - indeks omogoča, da nam ni potrebno prelistati cele knjige, ko želimo najti določeno vsebino

Indeksiranje

- Terminologija:
  - Podatkovna datoteka: datoteka s podatki (imenujemo tudi osnovna datoteka)
  - Indeksna datoteka: datoteka z indeksom. Indeks sestavlja iskalni ključ ter kazalec na zapis v podatkovni datoteki.
  - Iskalni ključ: sestavljen iz vrednosti polj, po katerih je datoteka indeksirana. Imenujemo tudi indeksno polje.



Podatkovni modeli

- S pomočjo podatkovnega modela opišemo posamezne vrste shem
  - strukturni in opisni mehanizem
  - jezik za opis podatkov
  - operatorji, ki tvorijo jezik za manipulacijo s podatki (povpraševalni jezik)
- Podatkovni model sestavljajo tri komponente:
  - podatkovna struktura, operacije nad podatkovno strukturo, integritetne omejitve, ki so lastne podatkovni strukturi
- Podatkovne modele delimo v:
  - logične, ki opisujejo zunanjo in konceptualno shemo,
  - fizične, ki opisujejo notranjo shemo.

Podatkovni modeli /2

- Logični podatkovni modeli:
  - površinski – temelje na datotečnih zapisih
  - globinski – modelirajo podobo sveta s stališča podatkov, shranjenih v fizični podatkovni bazi
- Površinski podatkovni model je lahko:
  - relacijski - temelji na relacijah
  - mrežni - temelji na grafih
  - hierarhični - temelji na drevesih
  - objektno orientirani model (po strukturi podoben mrežnemu modelu)

Poizvedovanje

- Relacijske podatkovne baze omogočajo uporabnikom postavljati enostavna vprašanja (poizvedbe), s katerimi pridobivajo podatke/informacije
  - ključna prednost za uporabnike
- Poizvedbe vpisujemo v jeziku, ki je prirejen za opisovanje poizvedb – poizvedovalni jezik.
- Relacijski model podpira zelo močne poizvedovalne jezike
- Ena pomembnih nalog SUPB je optimizacija poizvedb, tako da se te čim hitreje izvedejo

Primer poizvedbe v jeziku SQL

- Zanima nas, katera naročila imajo znesek višji od 3.000 EUR.

```
USE PURCHASE_ORDER
GO TO 1
DO WHILE NOT EOF
   IF PO_AMT > 3000
   THEN Print(PO_NUMBER, VENDOR_ID, PO_AMT)
END WHILE
```

Proceduralno

PURCHASE_ORDER
PO_NUMBER
RELEASE_DATE
PO_STATUS
PO_AMT
VENDOR_ID
…

```
SELECT PO_NUMBER, VENDOR_ID, PO_AMT
FROM PURCHASE_ORDER
WHERE PO_AMT >3000
```

Deklarativno

2.3 Relacijski podatkovni model

- Leta 1970 Edgar F. Codd objavi članek o relacijskem modelu podatkov za večuporabniške banke podatkov
  - rojstvo novega podatkovnega modela, ki nadomesti starejše modele
- Podatkovna baza, ki temelji na relacijskem modelu, je predstavljena z množico relacij, kjer je vsaka relacija tabela z vrsticami in stolpci.
- Je zelo enostaven za razumevanje:
  - tudi neizkušeni lahko razumejo vsebino podatkovne baze;
  - na voljo so enostavni vendar močni jeziki za poizvedovanje po vsebini podatkovne baze

Prednosti relacijskega podatkovnega modela

- Osnovne prednosti:
  - je definiran formalno in osnovan na matematičnih strukturah ali relacijah
  - ne vsebuje elementov fizičnega shranjevanja podatkov, s čimer je zagotovljena podatkovna neodvisnost
  - relacije so predstavljive s tabelami, ki so človeku dobro razumljive
- Najbolj razširjen v dostopni programski opremi za operiranje s podatkovnimi bazami

Terminologija

- Pri relacijskem modelu uporabljamo določeno terminologijo:
  - relacija
  - atribut
  - domena
  - n-terica
  - stopnja relacija
  - številnost relacije
  - relacijska podatkovna baza

Terminologija

- Relacijo si lahko predstavljamo kot dvodimenzionalno tabelo s stolpci in vrsticami
  - velja za logično strukturo podatkovne baze in ne za fizično

PO_NUMBER	RELEASE_DATE	PO_STATUS	PO_AMT	VENDOR_ID
2591	10.2.2006	CLOSED	4.300,00 €	V110
2592	10.2.2006	OPEN	505,50 €	V25
2593	11.2.2006	OPEN	4.000,00 €	V110
2594	12.2.2006	OPEN	3.280,00 €	V250
2595	15.2.2006	OPEN	500,00 €	V250
2596		HOLD	1.000,00 €	V75

Relacija

Terminologija

LMSV LAMS

- Atribut je poimenovani stolpec relacije

PO_NUMBER	RELEASE_DATE	PO_STATUS	PO_AMT	VENDOR_ID
2591	10.2.2006	CLOSED	4.300,00 €	V110
2592	10.2.2006	OPEN	505,50 €	V25
2593	11.2.2006	OPEN	4.000,00 €	V110
2594	12.2.2006	OPEN	3.280,00 €	V250
2595	15.2.2006	OPEN	500,00 €	V250
2596		HOLD	1.000,00 €	V75

Atribut relacije

Terminologija

LMSV LAMS

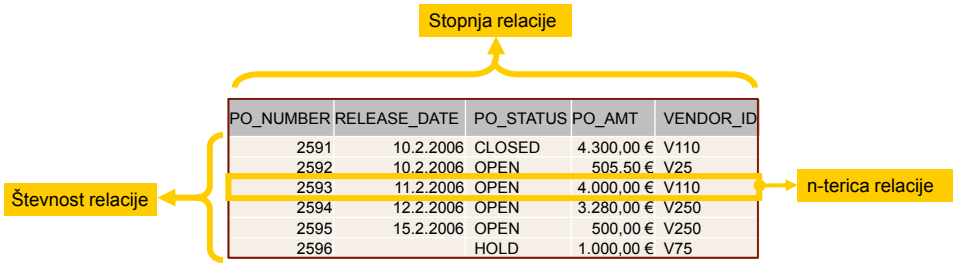
- Domena je množica dovoljenih vrednosti enega ali več atributov, ki so vključeni v to domeno
- Primeri domen:

Attribute	Domain Name	Meaning	Domain Definition
branchNo	BranchNumbers	The set of all possible branch numbers	character: size 4, range B001–B999
street	StreetNames	The set of all street names in Britain	character: size 25
city	CityNames	The set of all city names in Britain	character: size 15
postcode	Postcodes	The set of all postcodes in Britain	character: size 8
sex	Sex	The sex of a person	character: size 1, value M or F
DOB	DatesOfBirth	Possible values of staff birth dates	date, range from 1-Jan-20, format dd-mmm-yy
salary	Salaries	Possible values of staff salaries	monetary: 7 digits, range 6000.00–40000.00

Terminologija

LMSV LAMS

- N-terica je ena vrstica v relaciji.
- Številnost relacije je število n-teric relacije.
- Stopnja relacije je število atributov v relaciji.



PO_NUMBER	RELEASE_DATE	PO_STATUS	PO_AMT	VENDOR_ID
2591	10.2.2006	CLOSED	4.300,00 €	V110
2592	10.2.2006	OPEN	505,50 €	V25
2593	11.2.2006	OPEN	4.000,00 €	V110
2594	12.2.2006	OPEN	3.280,00 €	V250
2595	15.2.2006	OPEN	500,00 €	V250
2596		HOLD	1.000,00 €	V75

Terminologija pri relacijskem modelu

LMSV LAMS

- Relacijska podatkovna baza je množica normaliziranih relacij z enoličnimi imeni
- Normalizirane relacije dobimo po normalizaciji
  - normalizacija preoblikuje relacijsko shemo tako, da zniža stopnjo redundančnosti podatkov
  - **vsebinsko** gledano – postopek, ki pomaga pri doseganju boljše kakovosti načrta podatkovne baze
  - **tehnično** gledano – proces, ki vodi relacijsko shemo podatkovne baze skozi zaporedje testov
    - z vsakim testom preverimo, ali so izpolnjeni določeni pogoji
    - sheme relacij, ki ne zadoščajo pogojem, razdelimo na več manjših relacij tako, da dobljene relacije ustrezajo predpisanim zahtevam

Matematična definicija relacije

- Relacija, ki si jo predstavljamo kot tabelo, je matematična relacija stopnje n nad domenami atributov oziroma podmnožica kartezičnega produkta domen atributov.

$$r \subseteq (dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$$

Relacijska shema

- Vsaki relaciji pripada relacijska shema.
- Relacijsko shemo sestavlja oznaka sheme R ter lista oznak atributov A_i s pripadajočimi oznakami domen D_i:

$$R(A_1 : D_1, A_2 : D_2, \ldots, A_n : D_n)$$

- Definiramo še funkcijo

$$Sh : Relacija \rightarrow Shema \quad \text{oz.} \quad R = Sh(r)$$

- Relacijska shema predstavlja semantiko ali pomen relacije.

Relacijska shema

- Primeri relacijske sheme in relacije

  Študent(VpŠt, Ime, Priimek, Pošta, Kraj, Spol);

  Študent(VpŠt: number(8), Ime: char(20), Priimek: char(20), Pošta: number(4), Kraj: char(30), Spol: char(1));

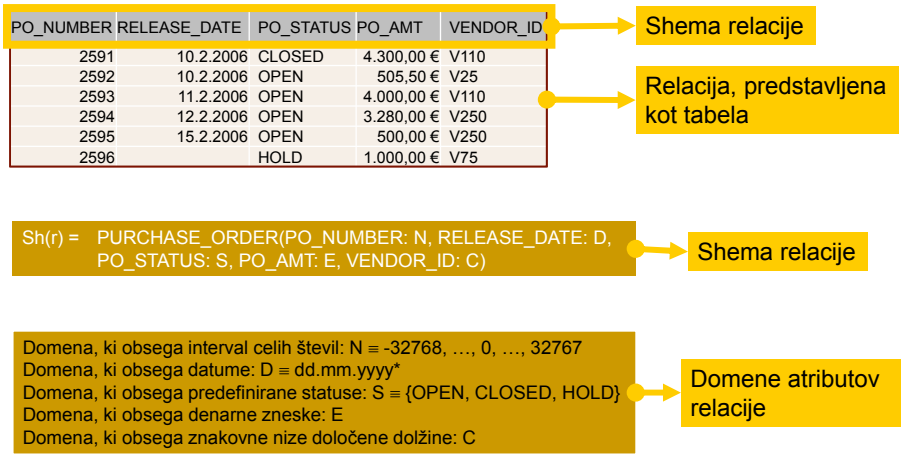
VpŠt	Ime	Priimek	Pošta	Kraj	Spol
24010632	Marko	Bric	5270	Ajdovščina	M
25089888	Iztok	Jerin	2000	Maribor	M
24135344	Maja	Klepec	1000	Ljubljana	Ž
24090909	Anita	Terčelj	4000	Kranj	Ž

Relacijska shema

- Relacijske sheme so del konceptualnih oziroma zunanjih shem. Razlagajo pomen relacij.

- Glede na skromno izrazno možnost nudijo informacijo le poznavalcem podatkovne baze, ki znajo relacijske sheme pravilno interpretirati.

Relacijska shema



Sh(r) = PURCHASE_ORDER(PO_NUMBER: N, RELEASE_DATE: D, PO_STATUS: S, PO_AMT: E, VENDOR_ID: C)

Shema relacije

Domena, ki obsega interval celih števil: N ≡ -32768, …, 0, …, 32767
Domena, ki obsega datume: D ≡ dd.mm.yyyy*
Domena, ki obsega predefinirane statuse: S ≡ {OPEN, CLOSED, HOLD}
Domena, ki obsega denarne zneske: E
Domena, ki obsega znakovne nize določene dolžine: C

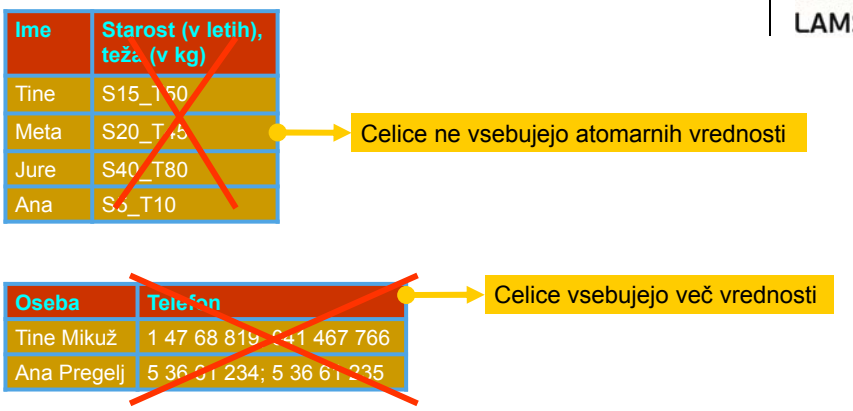
Domene atributov relacije

*dejanski format in podprt razpon vrednosti je odvisen od SUPB

Lastnosti relacij

- Ime relacije je enolično
  - v logični enoti (odvisno od SUPB, praviloma gre za shemo) podatkovne baze ni dveh relacij z enakim imenom
- Vsaka celica tabele (polje), ki predstavlja relacijo, vsebuje natančno eno atomarno vrednost
- Vsak atribut relacije ima enolično ime
  - v isti relaciji ni dveh atributov, ki bi imela isto ime
- Vrednosti nekega atributa so vse iz iste domene
- Vsaka n-terica relacije je enolična
  - v relaciji ni dveh enakih n-teric
- Vrstni red atributov v relaciji je nepomemben
- Vrstni red n-teric v relaciji je nepomemben

Primeri



Funkcionalne odvisnosti

- Relacija je model nekega stanja v svetu → njena vsebina ne more biti poljubna.
- Realne omejitve ne omogočajo, da bi bili odnosi v svetu kakršnikoli; možna so le določena stanja.

- Poznamo več vrst odvisnosti:
  - Funkcionalne odvisnosti (functional dependency)
  - Večvrednostne odvisnosti (multivalued dependency)
  - Stične odvisnosti (join dependency)

Funkcionalne odvisnosti

- Obravnavali bomo funkcionalne odvisnosti, ostale so pomembne za razumevanje višjih ravni normalizacije.

- **Funkcionalna odvisnost** je sredstvo, s katerim pri relacijskem podatkovnem modelu lahko povemo, katere vrednosti relacij so veljavne ali bi lahko bile veljavne, ter katere ne morejo obstajati

Funkcionalne odvisnosti

- Predpostavimo, da obstaja relacijska shema R z množico atributov, katere podmnožici sta X in Y.

- Pravimo, da v relacijski shemi R velja X → Y, če
  - v nobeni relaciji, ki pripada shemi R, ne moreta obstajati dve n-terici, ki bi se ujemali v vrednostih atributov X in se razlikovali v vrednostih atributov Y
  - pomeni, da X funkcionalno določa Y, oziroma Y je funkcionalno odvisen od X

Funkcionalne odvisnosti

- Množico funkcionalnih odvisnosti, ki veljajo med atributi relacijske sheme R in v vseh njenih relacijah, označimo z F

$$X \to Y \in F$$
$$\Leftrightarrow \forall r\big(Sh(r) = R$$
$$\Rightarrow \forall t, \forall u(t \in r \ \wedge \ u \in r \ \wedge \ t.X = u.X \ \Rightarrow \ t.Y = u.Y)\big)$$

kjer

t.X, u.X, t.Y in u.Y označujejo vrednosti atributov X oziroma Y v n-tericah t oziroma u

Primeri funkcionalnih odvisnosti

- Imamo relacijo s shemo
  Izpit(VpŠt, Priimek, Ime, ŠifraPredmeta, DatumIzpita, OcenaPisno, OcenaUstno)
- z naslednjim pomenom:
  Študent z vpisno številko VpŠt ter priimkom Priimek in imenom Ime je na DatumIzpita opravljal izpit iz predmeta s šifro ŠifraPredmeta. Dobil je oceno OcenaPisno in OcenaUstno.
- Funkcionalne odvisnosti relacijske sheme Izpit so:
  F ≡ { VpŠt → (Priimek, Ime), (VpŠt, ŠifraPredmeta, DatumIzpita) → (OcenaPisno, OcenaUstno) }

Ključi relacije

- Relacija je množica n-teric
  - v njej so si vse n-terice med seboj različne
- Enolična identifikacija posamezne n-terice
  - navedemo vrednosti atributov, ki jo sestavljajo
- Če v relaciji nastopajo funkcionalne odvisnosti
  - ni potrebno poznati vseh atributov, pač pa le tiste, ki zadoščajo za razlikovanje n-teric med seboj
- Množici atributov, ki n-terico enolično določajo, pravimo ključ relacije oziroma ključ relacijske sheme
- Če med atributi relacijske sheme ni funkcionalnih odvisnosti, potem tvorijo ključ vsi atributi skupaj

Ključi relacije

- Predpostavimo, da obstaja relacijska shema z atributi A_1 A_2 ... A_n, katere podmnožica je množica atributov X
- Atributi X so ključ relacijske sheme oziroma pripadajočih relacij, če sta izpolnjena naslednja dva pogoja:

  (1) $X \rightarrow A_1 A_2 \ldots A_n$

  (2) ne obstaja X', ki bi bila prava podmnožica od X, in ki bi tudi funkcionalno določala A_1 A_2 ... A_n

  lahko obstaja več različnih množic X, ki izpolnjujejo pogoje za ključ relacijske sheme

Ključi relacije

- Poznamo več konceptov, ki jih imenujemo ključ:
  - Kandidat za ključ (a key candidate)
    - množica atributov, ki izpolnjuje pogoje za ključ relacijske sheme
  - Primarni ključ (primary key)
    - iz praktičnih razlogov izmed kandidatov izberemo enega
  - Superključ ali nadključ (superkey)
    - vsaka množica atributov, v kateri je vsebovan ključ
    - v resnici nima lastnosti ključa, saj ne zadošča pogoju (2) – pogoju minimalnosti
  - Tuji ključ (foreign key)
    - atribut relacije, ki povezuje to relacijo z vrednostmi primarnega ključa druge relacije

Primeri ključev

VENDOR

VENDOR_ID	V_NAME	V_STREET	V_CITY	V_STATE	V_ZIP
V110	Jersey Vegetable	2 Main St.	Patterson	NJ	07055
V25	General Provisions	125 Common St.	Boise	ID	44830
V250	Spices Unlimited	25 Salty Lane	East Hampton	NY	10027
V75	Pasta Supply, Inc.	34 Henry St.	Philadelphia	PA	09098

Primarni ključ v tabeli VENDOR

PURCHASE_ORDER

PO_NUMBER	RELEASE_DATE	PO_STATUS	PO_AMT	VENDOR_ID
2591	10.2.2006	CLOSED	4.300,00 €	V110
2592	10.2.2006	OPEN	505,50 €	V25
2593	11.2.2006	OPEN	4.000,00 €	V110
2594	12.2.2006	OPEN	3.280,00 €	V250
2595	15.2.2006	OPEN	500,00 €	V250
2596		HOLD	1.000,00 €	V75

Primarni ključ v tabeli PURCHASE_ORDER

Tuji ključ v tabeli PURCHASE_ORDER → kaže na primarni ključ v tabeli VENDOR

Omejitve nad podatki

- Za celovitost ter skladnost podatkov v podatkovni bazi skrbimo s pomočjo omejitev
- Poznamo več vrst omejitev:
  - Omejitve domene (Domain constraints)
  - Pravila za celovitost podatkov (Integrity constraints)
    - Celovitost entitet (Entity Integrity)
    - Celovitost povezav (Referential Integrity)
  - Številnost (Multiplicity)
  - Splošne omejitve (General constraints)

Posebnost - oznaka “Null”

- Oznaka “Null”:
  - predstavlja vrednost atributa, ki je trenutno neznana ali irelevantna za n-terico.
  - gre za nepopolne podatke ali podatke pri izjemnih primerih.
  - predstavlja odsotnost podatka; to ni enako kot 0 ali prazen znak, kar je veljavna vrednost podatka
- Oznaka “Null” je problematična pri implementaciji
  - relacijski je model osnovan na predikatnem računu prvega reda (Boolean logic), kjer sta edini možni vrednosti true in false

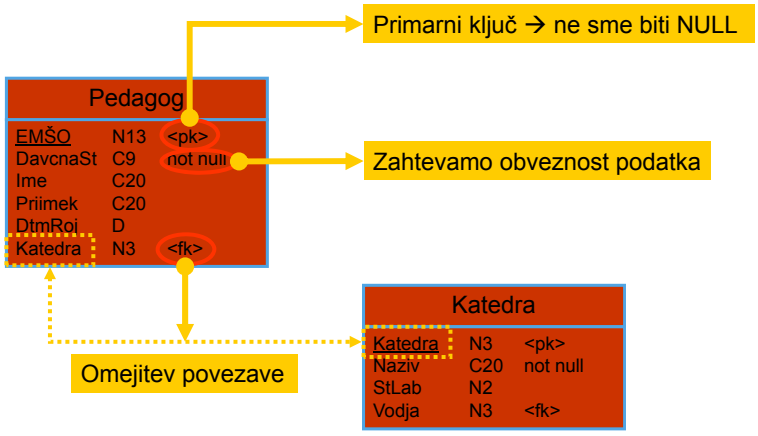
Omejitve entitet in povezav

LMSV LAMS

- Omejitev entitete
  - v osnovni relaciji noben atribut, ki je del ključa, ne sme biti enak Null
- Omejitve povezav
  - če v relaciji obstajajo tuji ključi, potem morajo:
    - njihove vrednosti ustrezati tistim, ki so v obliki ključa zapisane v eni izmed n-teric neke druge ali iste relacije
    - ali pa mora biti tuji ključ v celoti enak Null.
- Splošne omejitve
  - dodatna pravila, ki jih določi uporabnik ali skrbnik podatkovne baze, ki definirajo ali omejujejo nek vidik področja, za katerega je narejena podatkovna baza

Primeri omejitev

LMSV LAMS



Pogledi

- Osnovna relacija (base relation)
  - poimenovana relacija, ki ustreza nekemu entitetnemu tipu v konceptualnem modelu, katere n-terice so fizično shranjene v podatkovni bazi.
- Pogled (view)
  - rezultat ene ali več operacij nad osnovnimi relacijami z namenom pridobitve nove relacije
  - s pogledom zajemamo določeno informacijo iz ene ali več relacij
  - uporabniku se pogledi kažejo kot običajne tabele
  - v resnici obstaja le preslikava med osnovnimi relacijami in pogledom

Značilnosti pogledov

- Navidezna relacija
  - ne obstaja v relacijski bazi, temveč se dinamično kreira takrat, ko nekdo po njej povprašuje
- Vsebina pogleda
  - definirana kot poizvedba nad eno ali več osnovnimi relacijami
- Pogledi so dinamični
  - spremembe nad osnovnimi relacijami, katerih atributi so zajeti tudi v pogledu, so v pogledu takoj vidne

Namen uporabe pogledov

- Zagotavljanje varnosti
  - skrivajo posamezne dele podatkovne baze pred določenimi uporabniki
- Uporabnikom prilagojen način dostopa do podatkov
  - različni uporabniki vidijo iste podatke v istem času na različne načine
- Poenostavljajo kompleksne operacije nad osnovnimi relacijami

Primer pogleda

VENDOR

VENDOR_ID	V_NAME	V_STREET	V_CITY	V_STATE	V_ZIP
V110	Jersey Vegetable	2 Main St.	Patterson	NJ	07055
V25	General Provisions	125 Common St.	Boise	ID	44830
V250	Spices Unlimited	25 Salty Lane	East Hampton	NY	10027
V75	Pasta Supply, Inc.	34 Henry St.	Philadelphia	PA	09098

PURCHASE_ORDER

PO_NUMBER	RELEASE_DATE	PO_STATUS	PO_AMT	VENDOR_ID
2591	10.2.2006	CLOSED	4.300,00 €	V110
2592	10.2.2006	OPEN	505,50 €	V25
2593	11.2.2006	OPEN	4.000,00 €	V110
2594	12.2.2006	OPEN	3.280,00 €	V250
2595	15.2.2006	OPEN	500,00 €	V250
2596		HOLD	1.000,00 €	V75

```
SELECT V.VENDOR_ID, V.V_NAME, sum(P.PO_AMT) AS TOTAL_AMT
FROM VENDOR AS V, PURCHASE_ORDER AS P
WHERE V.VENDOR_ID = P.VENDOR_ID
GROUP BY V.VENDOR_ID, V.V_NAME
```

VENDOR_ID	V_NAME	TOTAL_AMT
V110	Jersey Vegetable	8300,000 €
V25	General Provisions	505,5000 €
V250	Spices Unlimited	3780,000 €
V75	Pasta Supply, Inc.	1000,000 €

2.4 Povpraševalni jezik SQL

- Dostop do vsebine relacij - s pomočjo povpraševalnih jezikov
  - lahko temeljijo na relacijski algebri (postopkovni jeziki)
  - ali na relacijskem računu (nepostopkovni jeziki)
- Slednji so bolj razširjeni predvsem po zaslugi SQL
  - angl. *Structured Query Language* - strukturirani povpraševalni jezik
  - standardiziran s strani ANSI in ISO
- SQL je več kot le povpraševalni jezik
  - tudi ukazi za deklariranje, kreiranje in zaščito

Uvod v SQL

- SQL sestavljata dve skupini ukazov
  - skupina ukazov DDL (Data Definition Language)
    - opredelitev strukture podatkovne baze
  - skupina ukazov DML (Data Manipulation Language)
    - poizvedovanje in ažuriranje podatkov
  - skupina ukazov DCL (Data Control Language)
    - odobritev dostopa uporabnikom
- SQL do izdaje SQL:1999 ne vključuje ukazov kontrolnega toka
  - kontrolni tok je bilo potrebno določiti s programskim jezikom ali interaktivno s posegi uporabnika

Uvod v SQL

- Osnovna podatkovna struktura – tabela
  - sorodna relaciji
  - razlika – v tabeli se lahko nahaja več vrstic z enako vrednostjo atributov
- Deklaracija tabele
  - enakovredna relacijski shemi
    - ime tabele,
    - seznam imen podatkovnih elementov skupaj z njihovimi osnovnimi tipi (domenami)
    - integritetne omejitve
- Pogledi v jeziku SQL
  - vsebina je določena z izborom (angl. query)

Uvod v SQL

- Lastnosti SQL
  - enostaven
  - nepostopkoven jezik (kaj in ne kako)
  - široka uporabnost
    - skrbniki PB, vodstvo, razvijalci informacijskih rešitev, končni uporabniki
  - obstaja ISO standard za SQL
  - SQL je de-facto in tudi uradno standardni jezik za delo z relacijskimi podatkovnimi bazami

Zgodovina SQL

- 70’ leta 20. stol.:
  - IBM na osnovi novega relacijskega podatkovnega modela E. Codda razvija projekt System R
  - 1974 – D.D. Chamberlin in R.F. Boyce (IBM San Jose Laboratory) definirata jezik ‘Structured English Query Language’ (SEQUEL)
  - SEQUEL se kasneje preimenuje v SQL
- Pozna 70’ leta 20. stol.:
  - Relational Software (danes Oracle) prepozna potencial idej Codda, Chamberlina in Boyce-a in razvije svoj SUPB na osnovi relacijskega modela
  - Poleti 1979 izda Oracle prvo komercialno različico SQL (Oracle V2 za računalnike VAX) – nekaj tednov pred IBM-ovo implementacijo System/38

Standardizacija SQL

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First formalized by ANSI.
1989	SQL-89	FIPS 127-1	Minor revision, in which the major addition were integrity constraints. Adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries (e.g. transitive closure), triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features (e.g. structured types). Support for embedding SQL in Java (SQL/OLB) and vice-versa (SQL/JRT).
2003	SQL:2003	SQL 2003	Introduced XML-related features (SQL/XML), *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006	SQL 2006	ISO/IEC 9075-14:2006 defines ways in which SQL can be used in conjunction with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form. In addition, it enables applications to integrate into their SQL code the use of XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.[37]
2008	SQL:2008	SQL 2008	Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers. Adds the TRUNCATE statement.[38]
2011	SQL:2011		

Vir: Wikipedia (http://en.wikipedia.org/wiki/SQL)

Pomembnost jezika SQL

- SQL je do sedaj edini široko sprejet standardni podatkovni poizvedovalni jezik
- SQL je del aplikacijskih arhitektur
  - npr. v okviru IBM-ove arhitekture - Systems Application Architecture (SAA)
- Strateška podpora
  - konzorcij X/OPEN za UNIX
  - Federal Information Processing Standard (FIPS)
    - standard, kateremu morajo ustrezati vsi SUPB prodani državnim organom v ZDA

Pomembnost jezika SQL

- SQL je uporabljen tudi v drugih standardih
  - ISO Information Resource Dictionary System (IRDS)
  - Remote Data Access (RDA),...
- Interes v akademskih krogih daje jeziku teoretično osnovo in tehnike za implementacijo
  - optimizacija poizvedb
  - distribucija podatkov
  - varnost podatkov
- Pojavljajo se specializirane implementacije SQL, npr. za OLAP (online analytical processing)

Implementacija SQL

- Najpogostejše ključne besede:
  - ALIAS, ALL, AS, ASC, BETWEEN, BY, DESC, DISTINCT, FROM, GROUP, IN, INNER, INTO, IS, JOIN, LIKE, ON, ORDER, SELECT, WHERE
  - funkcije AVG, MAX, MIN, SUM
  - in logični operatorji AND, OR, NOT
- Razlike standardi SQL-92, SQL:1999; SQL:2008,…
- Proizvajalci programske opreme SUPB so dodali v svoje produkte nove besede
  - narečja (dialekti) jezika SQL

Stavki skupine SQL DML

- Sklop jezika DML zajema stavke SQL za manipulacijo s podatki
  - SELECT → Izbira
  - INSERT → Dodajanje
  - DELETE → Brisanje
  - UPDATE → Spreminjanje
- Najbolj kompleksna je sintaksa stavka SELECT
  - tudi najbolj uporabljan stavek
  - z njim sestavljamo poizvedbe

Stavek SELECT

- Osnovna sintaksa

```
SELECT [ALL|DISTINCT]
        {* | [stolpniIzraz [AS novoIme]] [,...] }
FROM        imeTabele [alias] [, ...]
[WHERE      pogoj]
[GROUP BY   listaStolpcev]  [HAVING pogoj]
[ORDER BY   listaStolpcev [ASC|DESC]];
```

- Struktura stavka je zelo podobna naravni poizvedbi v angleškem jeziku

Pomen sklopov stavka SELECT

- SELECT
  - izberemo podatke, ki jih definirajo stolpni izrazi, ti podatki bodo stolpci v ciljni tabeli – pogledu
  - ALL – v ciljni tabeli so vsebovane vse vrstice, ki izpolnjujejo pogoj; tudi, če je več ciljnih vrstic enakih
  - DISTINCT – podvojene vrstice se izločijo
- FROM
  - izbiramo iz navedenih tabel
- WHERE
  - preko pogoja filtriramo vrstice, ki jih hočemo imeti v ciljni tabeli (npr. stolpec1>20); pri sestavljanju pogojev lahko uporabljamo operatorje AND, OR, NOT, BETWEEN, …

Pomen sklopov stavka SELECT

- GROUP BY
  - vrstice združujemo po vrednostih izbranih stolpcev
- HAVING
  - skupine (združene vrstice) filtriramo glede na določene pogoje (v zvezi z agregacijo podatkov)
- ORDER BY
  - povemo, kateri stolpec se izbere kot osnova za razvrščanje vrstic v ciljni tabeli
  - elemente v stolpcu lahko razvrščamo od najmanjšega proti največjemu (ASC) ali obratno (DESC)

Vrstnega reda sklopov ni možno spreminjati!
Obvezna sta samo sklopa SELECT in FROM!

Primeri

- Za primere bomo uporabljali shemo PB o naročilih

  Vendor(vendor_id, v_name, v_street, v_city, v_state, v_zip)

  Purchase_order(po_number, release_date, po_status, po_amt, vendor_id)

  Po_detail(po_number, po_line_item, material_id, units, quantity, balance, promised_del_date, unit_cost, status)

- Izpis vseh podatkov o naročilih

  SELECT po_number, release_date, po_status, po_amt, vendor_id
  FROM Purchase_order;

  ali krajše

  SELECT * FROM Purchase_order;

Uporaba DISTINCT

- Izpis oznak dobaviteljev, od katerih smo naročali

```
SELECT DISTINCT vendor_id
FROM Purchase_order
```

Ukaz DISTINCT izloči ponavljajoče se zapise

Izračunana polja

- Izpis seznama naročil z zneskom v SIT

```
SELECT po_number, po_amt*239.64 AS po_amt_sit
FROM Purchase_order
```

Lahko uporabljamo matematične izraze, ki vključujejo imena stolpcev

Izračunanemu stolpcu dodelimo naziv

Iskalni kriteriji

- Izpis odprtih naročil v znesku večjem od 1000 EUR

```
SELECT *
FROM Purchase_order
WHERE po_status = "OPEN" AND po_amt > 1000
```

Pogoj 1

Pogoj 2

Združitev pogojev z logičnim operatorjem

Iskanje z uporabo BETWEEN

- Izpis naročil v znesku med 1000 in 4000 EUR

```
SELECT po_number, po_amt
FROM Purchase_order
WHERE po_amt BETWEEN 1000 AND 4000
```

BETWEEN vključuje spodnjo in zgornjo mejo!
Uporabimo lahko tudi negacijo NOT BETWEEN
BETWEEN je vedno možno izraziti tudi posredno

Iskanje po članstvu množice

- Izpis naročil in pripadajočih zneskov za skupino dobaviteljev "V25", "V75" in "V110"

```
SELECT po_number, po_amt
FROM Purchase_order
WHERE vendor_id IN ("V25", "V75", "V110")
```

Članstvo množice

Uporabimo lahko tudi negacijo NOT IN

IN je uporaben pri večjih množicah, sicer lahko ga izrazimo tudi posredno

Iskanje z vzorcem

- Izpis naročil materialov, ki v oznaki vključujejo niz „31“, in pripadajočih količin

```
SELECT po_number, material_id, quantity
FROM Po_detail
WHERE material_id LIKE '%31%'
```

Iskanje z vzorcem

SQL ima dva posebna znaka za iskanje z vzorcem:

Znak % nadomešča katerikoli niz znakov (v MS Accessu *)

Znak _ nadomešča katerikoli znak

Iskanje z vrednostjo NULL v pogoju

- Izpis vseh naročil, ki še niso bila izdana (ni podatka o datumu izdaje)

```
SELECT *
FROM Purchase_order
WHERE release_date IS NULL
```

Iskanje z vrednostjo NULL

Uporabljamo lahko tudi negacijo **IS NOT NULL**

Sortiranje vrstic v izhodni relaciji

- Izpis podatkov o naročilih, pri čemer naj bo izpis urejen po oznaki dobavitelja od najmanjše do največje in pri naročilih od istega dobavitelja po zneskih od največjega do najmanjšega

```
SELECT vendor_id, po_amt, po_number, po_status
FROM Purchase_order
ORDER BY vendor_id ASC, po_amt DESC
```

ASC – ascending → naraščujoče
DESC – descending → padajoče

Agregiranje podatkov

- Standard ISO definira pet operacij agregiranja
  - COUNT vrne število vrednosti v določenem stolpcu
  - SUM vrne seštevek vrednosti v določenem stolpcu
  - AVG vrne povprečje vrednosti v določenem stolpcu
  - MIN vrne najmanjšo vrednost v določenem stolpcu
  - MAX vrne največjo vrednost v določenem stolpcu
- Vse operacije delujejo na enem stolpcu in vračajo eno samo vrednost
- Operacije agregiranja lahko uporabimo le v sklopu SELECT ali HAVING

Agregiranje podatkov

- Tipi podatkov
  - COUNT, MIN in MAX se uporabljajo za numerične in ne-numerične vrednosti
  - SUM in AVG zahtevata numerične vrednosti
- Obravnava manjkajočih podatkov
  - vse operacije razen COUNT(*) najprej odstranijo vrstice z NULL vrednostjo v stolpcu, po katerem agregiramo.
  - COUNT(*) prešteje vse vrstice, ne glede na vrednosti NULL ali duplikate
- Duplikati
  - DISTINCT pred imenom stolpca odstrani duplikate
  - DISTINCT nima učinka na MIN/MAX, lahko pa vpliva na SUM/AVG

Agregiranje podatkov

- Če vsebuje sklop SELECT poleg operacije agregiranja tudi neagregiran stolpec tabele, mora obstajati tudi sklop GROUP BY, sicer ni možno povezati agregirane vrednosti in vrednosti stolpca

~~SELECT po_number, AVG(quantity)~~
~~FROM Po_detail~~

Napačna raba agregacije

Uporaba COUNT in DISTINCT

- Koliko različnih naročil ima planiran rok dobave v prvi polovici marca 2006

```
SELECT COUNT (DISTINCT po_number) AS num_po
FROM Po_detail
WHERE promised_del_date
        BETWEEN '1.3.2006' AND '15.3.2006'
```

MS Access:

DISTINCT tu ne dela, obvoz je funkcija DCOUNT;
BETWEEN #3/1/06# AND #3/15/06#

Uporaba več operacij agregiranja istočasno

- Izpis povprečnega, minimalnega in maksimalnega zneska odprtih naročil

```
SELECT AVG(po_amt), MIN(po_amt), MAX(po_amt)
FROM Purchase_order
WHERE po_status = "OPEN"
```

Združevanje podatkov

- Za združevanje podatkov v skupine uporabimo sklop GROUP BY
- Operacija agregiranja nato deluje na vsaki skupini posebej
- Sklopa SELECT in GROUP BY sta povezana
  - vsi stolpci, ki so navedeni v sklopu SELECT zunaj operacij agregiranja, se morajo nahajati tudi v sklopu GROUP BY
  - če uporabljamo sklop WHERE v kombinaciji z GROUP BY, se najprej upošteva WHERE, združevanje se nato izvede na preostalih vrsticah
  - standard ISO jemlje vrednosti NULL kot enake, ko gre za združevanje

Primer združevanja

- Izpis števila materialov z enakimi cenami na kos po posameznih naročilih

```
SELECT po_number, unit_cost, COUNT(material_id)
FROM Po_detail
GROUP BY po_number, unit_cost
```

Za vsako skupino naročilo, cena kosa vrne število pripadajočih materialov

Kriteriji za prikaz skupin

- Za določanje/izločanje skupin, ki se pojavijo v rezultatu, uporabimo sklop HAVING
- Deluje podobno kot WHERE
  - WHERE filtrira posamezne vrstice
  - HAVING filtrira skupine
- Stolpci, ki so navedeni v sklopu HAVING, morajo biti tudi v sklopu SELECT
  - lahko so tudi rezultat operacije agregiranja

Uporaba sklopa HAVING

- Izpis minimalnih in maksimalnih količin tistih naročil, pri katerih je maksimalna količina večja od 1000

```
SELECT po_number, MIN(quantity) , MAX(quantity)
FROM Po_detail
GROUP BY po_number
HAVING MAX(quantity) > 1000
```

Uporaba sklopa HAVING

- Izpis števila materialov z enakimi cenami na kos po posameznih naročilih in njihovo povprečno količino. Upoštevamo samo tiste primere, ko je število materialov z enakimi cenami na kos večje od 1.

```
SELECT po_number, unit_cost, COUNT(material_id),
AVG(quantity)
FROM Po_detail
GROUP BY po_number, unit_cost
HAVING COUNT(material_id) > 1
```

Gnezdenje poizvedb

- Določeni stavki SQL imajo lahko vgnezdene stavke SELECT
  - rezultat vgnezdene poizvedbe se uporabi kot argument
- Vgnezdeni stavki SELECT se lahko uporabijo v sklopih WHERE ali HAVING drugega stavka SELECT (podpoizvedba oz. subselect)
- Vgnezdeni stavki SELECT se lahko pojavijo tudi v drugih stavkih skupine SQL DML – INSERT, UPDATE in DELETE

Primer vgnezdenega stavka SELECT

- Izpis naročil in pripadajočih zneskov, pri čemer upoštevamo le naročila z zneskom, ki je večji od zneska povprečnega naročila

```
SELECT po_number, po_amt
FROM Purchase_order
WHERE po_amt >
    (SELECT AVG(po_amt)
    FROM Purchase_order)
```

Primer vgnezdenega stavka SELECT, ki vključuje dve tabeli

- Izpis podrobnosti naročil pri dobavitelju "V250"

```
SELECT *
FROM Po_detail
WHERE po_number IN
    ( SELECT po_number
    FROM Purchase_order
    WHERE vendor_id = "V250" )
```

Primer vgnezdenega stavka SELECT, ki vključuje dve tabeli

- Izpis naročil in pripadajočih zneskov, pri čemer upoštevamo le naročila z vsaj dvema različnima postavkama v količini vsaj 1000 enot na postavko

```
SELECT po_number, po_amt
FROM Purchase_order
WHERE po_number IN
    (SELECT po_number
    FROM Po_detail
    GROUP BY po_number
    HAVING COUNT(po_number) >= 2)
```

Ali je stavek pravilen?

Manjka WHERE pogoj v vgnezdenem SELECT stavku:

WHERE quantity >= 1000

Pravila gnezdenja stavkov SELECT

- Vgnezdeni stavki SELECT ne smejo uporabljati sklopa ORDER BY
- Sklop SELECT vgnezdenega SELECT stavka lahko vključuje samo en stolpec, razen v primeru uporabe ukaza EXISTS
- Imena stolpcev v vgnezdenem SELECT stavku se privzeto nanašajo na tabele iz vgnezdenega ali zunanjega SELECT stavka (uporaba alias-ov)
- Ko je vgnezden SELECT stavek operand v primerjavi, se mora nahajati na desni strani le-te
- Vgnezdeni SELECT stavek ne more biti operand v izrazu

Uporaba operatorjev ANY in ALL

- V vgnezedenih SELECT stavkih, ki vračajo en sam stolpec, lahko uporabljamo operatorja ANY in ALL.
- Z uporabo ALL bo pogoj izpolnjen samo, če bo veljal za vse vrednosti, ki ji vrača poizvedba.
- Z uporabo ANY, bo pogoj izpolnjen, če bo veljal za vsaj eno od vrednosti, ki ji poizvedba vrača.
- Če je rezultat poizvedbe prazen, bo ALL vrnil true, ANY pa false.
- Namesto ANY lahko uporabljamo tudi SOME.

Primer uporabe operatorja ANY

- Izpis številk naročil ter pripadajočih materialov, katerih cena na enoto je večja ali enaka od vsaj enega materiala v naročilu št. 2591

```
SELECT po_number, material_id
FROM Po_detail
WHERE po_number <> 2591 AND
        unit_cost >= ANY ( SELECT unit_cost
                           FROM Po_detail
                           WHERE po_number = 2591 );
```

Primer uporabe operatorja ALL

- Izpis številk naročil ter pripadajočih materialov, katerih cena na enoto je večja ali enaka od vseh materialov v naročilu št. 2591

```
SELECT po_number, material_id
FROM Po_detail
WHERE po_number <> 2591 AND
        unit_cost >= ALL ( SELECT unit_cost
                           FROM Po_detail
                           WHERE po_number = 2591 );
```

Poizvedbe, ki zajemajo podatke iz več tabel

- Vgnezdeni stavki SELECT so en način poizvedovanja, ki vključuje več tabel
  - omejitev: v rezultatu so lahko le stolpci iz ene tabele (na katero se nanaša glavna poizvedba)
- Če želimo v rezultatu kombinirati stolpce iz različnih tabel, moramo uporabljati t.i. stik tabel (table join)
- Izvedba stika
  - v sklopu FROM navedemo več kot eno tabelo
  - v sklopu WHERE tabele povežemo preko stolpcev

Poizvedbe, ki zajemajo podatke iz več tabel

- Pri sklicevanju na stolpce moramo definirati, iz katere tabele je stolpec
  - potrebno za ločevanje med istoimenskimi stolpci različnih tabel
- Za tabele v razdelku FROM lahko uvedemo sinonime (alias) – bolj kompakten zapis
- Sintaksa:

```
SELECT O.po_number, D.material_id
FROM Purchase_order O, Po_detail D
...
```

Primer poizvedbe po dveh tabelah

- Izpis datuma, št. naročila, materiala, količine in cene na enoto za naročila od dobavitelja z oznako "V250"

```
SELECT O.release_date, O.po_number, D.material_id,
    D.quantity, D.unit_cost
    FROM Purchase_order O, Po_detail D
    WHERE O.po_number = D.po_number
    AND O.vendor_id = "V250"
```

  - za povezavo smo uporabili tuji ključ po_number v tabeli Po_detail, ki je primarni ključ v tabeli Purchase_order

Primer poizvedbe po več tabelah

- Izpis imena dobavitelja, datuma, št. naročila, materiala, količine in cene na enoto za naročila od dobavitelja z oznako "V250"

```
SELECT V.v_name, O.release_date, O.po_number,
    D.material_id, D.quantity, D.unit_cost
    FROM Vendor V, Purchase_order O, Po_detail D
    WHERE V.vendor_id = O.vendor_id
    AND O.po_number = D.po_number
    AND O.vendor_id = "V250"
```

Alternativni načini stika več tabel

- SQL omogoča alternativne načine stika med več tabelami
  - primer stika dveh tabel:

    FROM Purchase_order O JOIN Po_detail D
    ON O.po_number = D.po_number

    FROM Purchase_order JOIN Po_detail USING po_number

    FROM Purchase_order NATURAL JOIN Po_detail
- Zgornji zapisi nadomestijo sklopa FROM in WHERE
- V prvem primeru rezultat vsebuje dva identična stolpca po_number

V MS Accessu so na voljo ukazi INNER JOIN, LEFT JOIN, RIGHT JOIN

Zunanji stik

- Za zapis SELECT stavka, ki vsebuje zunanji stik med dvema tabelama, uporabimo naslednjo sintakso:

```
SELECT DISTINCT V.v_name, O.po_number
FROM Vendor V LEFT JOIN Purchase_order O
        ON V.vendor_id = O.vendor_id;
```

Kaj izpiše zgornja poizvedba?

V izpisu se pojavijo vsi dobavitelji na seznamu, tudi če pri njih nismo naročali,

Polni zunanji stik

- SQL omogoča tudi izvedbo polnega zunanjega stika (Full Outer Join)
- Polni zunanji stik med tabelama A in B kot rezultat vrne tudi tiste vrstice, ki v tabeli A ali B nimajo stičnega para.

- Sintaksa:

```
SELECT DISTINCT V.v_name, O.po_number
FROM Vendor V FULL OUTER JOIN  Purchase_order O
            ON V.vendor_id = O.vendor_id;
```

V MS Accessu ukaz FULL OUTER JOIN ni na voljo.

Uporaba EXISTS in NOT EXISTS

- EXISTS in NOT EXISTS lahko uporabljamo le v vgnezdenih poizvedbah.
- Vračajo logičen rezultat true/false.
  - True dobimo, če obstaja vsaj ena vrstica v tabeli, ki je rezultat vgnezdene poizvedbe.
  - False dobimo, če vgnezdena poizvedba vrača prazno množico.

- NOT EXISTS je negacija EXISTS.

Uporaba EXISTS in NOT EXISTS

- (NOT) EXISTS preveri samo, če v rezultatu vgnezdene poizvedbe (ne) obstajajo vrstice

- Število stolpcev v SELECT sklopu vgnezdene poizvedbe je zato irelevantno
- Navadno uporabimo sintakso:

  (SELECT * ...)

Primer uporabe EXISTS

- Izpis vseh materialov, ki smo jih naročali od izbranega dobavitelja (General Provisions)

```
SELECT material_id
FROM Po_detail D
WHERE EXISTS
        ( SELECT *
          FROM Purchase_order O, Vendor V
          WHERE O.po_number = D.po_number AND
          O.vendor_id = V.vendor_id AND
          V.v_name = "General Provisions")
```

Kaj dobimo, če ta pogoj izpustimo?

Namesto EXISTS lahko uporabimo stik


```
SELECT DISTINCT D.material_id
FROM Po_detail D, Purchase_order O, Vendor V
WHERE O.po_number = D.po_number AND
        O.vendor_id = V.vendor_id AND
        V.v_name = "General Provisions"
```

Uporaba operacij nad množicami

- Rezultate dveh ali več poizvedb lahko združujemo z ukazi:
  - Union (unija),
  - Intersection (Presek)
  - Difference (EXCEPT) (Razlika)

- Da lahko izvajamo naštete operacije, morata biti tabeli A in B skladni (domene atributov morajo biti enake)

Primer unije

- Predpostavimo dodatno tabelo s podatki o kupcih
  Buyer(buyer_id, b_name, b_street, b_city, b_state, b_zip)

- Izpis seznama poslovnih partnerjev (dobaviteljev in kupcev)

```
SELECT v_name, v_street, v_city FROM Vendor
UNION
SELECT b_name, b_street, b_city FROM Buyer;
```

Stavek INSERT

- Osnovna sintaksa

```
INSERT INTO imeTabele [( imeStolpca1 [, imeStolpca2] [,…])]
VALUES (vrednostPodatka1 [, vrednostPodatka2] [,...]);
```

  - seznam stolpcev ni obvezen
    - če seznama ne podamo, se privzamejo vsi stolpci
  - pri vnosu moramo vpisati najmanj vse stolpce z obvezno vrednostjo (not null), izjema so stolpci s privzeto vrednostjo (DEFAULT).
  - seznam podatkov mora ustrezati seznamu stolpcev

Primeri stavkov INSERT

- Vnos nove vrstice v tabelo Purchase_order

```
INSERT INTO Purchase_order
VALUES (2597, '17.2.2006', 'OPEN', 1500.00, 'V25');
```

- Vnos brez podatka o datumu

```
INSERT INTO Purchase_order
VALUES (2598, null, 'HOLD', 500.00, 'V75')
```

ali

```
INSERT INTO Purchase_order (po_number, po_status,
po_amt, vendor_id)
VALUES (2598, 'HOLD', 500.00, 'V75')
```

Primeri stavkov INSERT

- Dodajanje vrstic iz ene ali več drugih tabel

INSERT INTO imeTabele [(listaStolpcev)]
SELECT ...

- Primer: nova tabela z imeni in naslovi

ContactList(contact_id, c_name, c_street, c_city)

```
INSERT INTO ContactList (contact_id, c_name, c_street, c_city)
SELECT vendor_id, v_name, v_street, v_city FROM Vendor;
```

Stavek UPDATE

- Osnovna sintaksa

  UPDATE imeTabele
  SET imeStolpca1 = vrednostPodatka1/izraz1
  [, imeStolpca2 = vrednostPodatka2/izraz2]
  [, ...]
  [**WHERE** pogoj];

  - imeTabele se lahko nanaša na ime osnovne tabele ali ime pogleda
  - sklop SET določa nazive enega ali več stolpcev ter nove vrednosti teh stolpcev (morajo biti ustreznega podatkovnega tipa)
  - sklop WHERE ni obvezen

Primera stavka UPDATE

- Zaradi podražitev materiala se je znesek še neizdanih naročil povišal za 5 %

  ```
  UPDATE Purchase_order
  SET po_amt = po_amt * 1.05
  WHERE po_status = 'HOLD’
  ```

- Dobavitelj V250 je obvestil kupca, da naročilo 2594 ne bo dobavljeno pred 22.3.2006

  ```
  UPDATE Po_detail
  SET promised_del_date = '22.3.2006'
  WHERE po_number = 2594;
  ```

Stavek DELETE

- Osnovna sintaksa

  DELETE FROM imeTabele
  [WHERE pogoj]

  - imeTabele se lahko nanaša na ime osnovne tabele ali ime pogleda
  - sklop WHERE ni obvezen; ukaz DELETE brez WHERE zbriše vse vrstice v tabeli; tabela pri tem ostane v bazi

Primera stavkov DELETE

- Odločeno je bilo, da se naročilo 2596 ne izvede, zato ga je potrebno izbrisati iz baze

  ```
  DELETE FROM Po_detail
  WHERE Po_number = 2596;
  DELETE FROM Purchase_order
  WHERE Po_number = 2596;
  ```

  - ukaz DELETE lahko naenkrat izbriše le vrstice ene tabele
  - vrstni red brisanja je pomemben
    - da ohranimo integriteto relacij, moramo nazadnje brisati tabelo, v kateri je brisani atribut primarni ključ
    - npr. po_number je tuji ključ v tabeli Po_detail in primarni ključ v tabeli Purchase_order, zato ju brišemo v tem vrstnem redu

Stavki skupine SQL DDL

- Skupina SQL DDL zajema stavke za manipulacijo s strukturo podatkovne baze
  - CREATE – ustvarjanje novega objekta znotraj SUPB (npr. podatkovne baze, tabele, pogleda, domene ipd.)
  - DROP – brisanje baze, tabele, pogleda
  - ALTER – spreminjanje obstoječega objekta v SUPB
- Dodatno so na voljo še stavki za definicijo integritetnih omejitev, npr. določanje primarnega in tujega ključa
  - te stavke uporabljamo znotraj stavkov CREATE in ALTER
- Najbolj razvejena je sintaksa stavka CREATE

Ustvarjanje podatkovne baze

- Osnovna sintaksa

  CREATE DATABASE imePodatkovneBaze;

  - ukaz vnesemo neposredno v ukazni vrstici SUPB ali posredno, preko menujskega sistema SUPB
    - npr. v MS Accessu: File/New …
  - ustvari se podatkovna baza z danim imenom
  - bazo je potrebno napolniti s strukturo tabel

Ustvarjanje tabele

- Osnovna sintaksa

CREATE TABLE imeTabele
(imeStolpca1 tipPodatka1,
imeStolpca2 tipPodatka2,
... ...
imeStolpcaN tipPodatkaN);

  - v ustvarjeni tabeli so stolpci razvrščeni v vrstnem redu, ki ga določa stavek CREATE

Ustvarjanje tabele

- Sintaksa z vključenimi omejitvami

CREATE TABLE imeTabele
({imeStolpca tipPodatka omejitev}

  - Omejitve (natančna sintaksa je odvisna od izbranega SUPB)
    - **NOT NULL** – v stolpcu ne more biti vrednosti NULL
    - **UNIQUE** – vsaka vrstica stolpca mora imeti unikatno vrednost
    - **PRIMARY KEY** – definira stolpec (ali več stolpcev) kot primarni ključ; vključuje NOT NULL in UNIQUE
    - **FOREIGN KEY** – zagotavlja referenčno integriteto podatkov v eni tabeli, ki se navezujejo na podatke v drugi tabeli
    - **CHECK** – zagotavlja, da vrednosti podatkov v stolpcu izpoljujejo določen pogoj
    - **DEFAULT** – določa privzeto vrednost v stolpcu

Primer ustvarjanja tabele

- Primer ustvarjanja tabele Purchase_order v MS Accessu

```
CREATE TABLE Purchase_order
(po_number      SMALLINT,
release_date    DATE,
po_status       CHAR(6),
po_amt          CURRENCY,
vendor_id       CHAR(5),
CONSTRAINT index1 PRIMARY KEY (po_number));
```

Primer ustvarjanja tabele /2

- Opombe k primeru
  - pri znakovnih nizih smo omejili dolžino, s čimer zmanjšamo rabo pomnilnika
  - z ukazom CONSTRAINT smo definirali po_number kot primarni ključ in posredno s tem določili omejitve, da mora biti po_number **Not Null** in **Unique**
    - vsaka vrstica tabele mora vsebovali vrednost po_number, ta pa se v tabeli ne sme ponoviti
  - definirali smo tudi indeks (index1), s katerim pohitrimo iskanje po tabeli

Primer ustvarjanja tabele

- Najprej ustvarimo domene

```
CREATE DOMAIN hotelNumber AS NUMERIC(3)
CHECK (VALUE IN (SELECT hotelNo FROM Hotel));

CREATE DOMAIN guestNumber AS NUMERIC(3)
CHECK (VALUE IN (SELECT guestNo FROM Guest));

CREATE DOMAIN rezervDate AS DATE;
CHECK(VALUE BETWEEN '1.1.1995' AND '1.1.2200');

CREATE DOMAIN roomNumber AS INTEGER;
CHECK(VALUE BETWEEN 100 AND 545);

CREATE DOMAIN comments AS VARCHAR(100);
```

Primer ustvarjanja tabele

- Nato ustvarimo tabelo

```
CREATE TABLE Booking (
    hotelNo      hotelNumber  NOT NULL,
                 CONSTRAINT PrevecRezervacij…
    guestNo      guestNumber NOT NULL,
    dateFrom     rezervDate    NOT NULL    DEFAULT date(),
    dateTo       rezervDate    NOT NULL,
    roomNo       roomNumber  NOT NULL,
    comments     comments,
    PRIMARY KEY (hotelNo),
    FOREIGN KEY (guestNo) REFERENCES Guest
    ON DELETE SET NULL ON UPDATE CASCADE …);
```

Ustvarjanje sheme

- Relacije in drugi podatkovni objekti obstajajo v nekem okolju
- Okolje vsebuje enega ali več katalogov, vsak katalog pa množico shem
- **Shema** je poimenovana kolekcija povezanih podatkovnih objektov
- Objekti v shemi so lahko tabele, pogledi, domene, trditve, dodelitve, pretvorbe in znakovni nizi; vsi objekti imajo istega lastnika

Ustvarjanje in brisanje sheme

CREATE SCHEMA [Name |
AUTHORIZATION CreatorId]

DROP SCHEMA Name [RESTRICT | CASCADE]

- RESTRICT (privzeto): shema mora biti prazna, sicer brisanje ni možno.
- CASCADE: kaskadno se brišejo vsi objekti, povezani s shemo. Če katerokoli brisanje ne uspe, se zavrne celotna operacija.

Brisanje objektov

- Stavek DROP
  - s pomočjo stavka DROP TABLE ukinemo tabelo. Obenem se zbrišejo vsi zapisi tabele.

  DROP TABLE TableName [RESTRICT | CASCADE]

  - Restrict: Ukaz se ne izvede, če obstajajo objekti, ki so vezani na tabelo, ki jo brišemo.
  - Cascade: kaskadno se brišejo vsi vezani objekti.
- Primer:

  DROP TABLE Purchase_order RESTRICT;
- Podobno lahko brišemo tudi druge objekte

Spreminjanje objektov

- Stavek ALTER
  - na različne načine spremeni strukturo obstoječega objekta v bazi
- S stavkom ALTER TABLE lahko:
  - dodajamo ali ukinjamo stolpce v tabeli
  - dodajamo ali ukinemo omejitve tabele
  - za stolpce v tabeli določamo ali ukinjamo privzete vrednosti
  - spreminjamo podatkovne tipe stolpcev v tabeli

Primeri spreminjanja objektov


```
ALTER TABLE Booking
        ALTER fromDate DROP DEFAULT;

ALTER TABLE Booking
        DROP CONSTRAINT prevecRezervacij;

ALTER TABLE Gost
        ADD Spol NOT NULL DEFAULT = 'M';
```

Transakcije

- SQL definira transakcijski model z ukazoma COMMIT in ROLLBACK
  - COMMIT – spremembe so trajne
  - ROLLBACK – s transakcijo narejene spremembe se razveljavijo
- Transakcija je logična enota dela z enim ali več ukazi SQL; s stališča zagotavljanja skladnega stanja je atomarna
- Spremembe, ki so narejene znotraj poteka transakcije, praviloma niso vidne navzven drugim transakcijam, dokler transakcija ni končana

Transakcije

- Osnovna sintaksa (MS SQL Server)

```
BEGIN TRAN T1;

UPDATE table1…;
UPDATE table1…;
INSERT INTO table3 …;

COMMIT TRAN T1;
```

Proceduralni elementi SQL

- V prvotni različici je bil SQL zgolj deklarativni jezik in ni vseboval proceduralnih elementov in elementov kontrole toka
- Razširitev Persistent Stored Modules
  - prvič objavljena 1996, poznana tudi kot PSM-96
  - postane del standarda 1999: ISO/IEC 9075-4:1999 Information technology – Database languages – SQL – Part 4: Persistent Stored Modules (SQL/PSM)
- Druge razširitve ponudnikov SUPB
  - Interbase / Firebird (PSQL) – Procedural SQL
  - IBM DB2 (SQL PL) – SQL Procedural Language (implementira SQL/PSM)

Proceduralni elementi SQL

- Druge razširitve ponudnikov SUPB
  - IBM Informix (SPL) – Stored Procedural Language
  - Microsoft / Sybase (T-SQL) – Transact-SQL
  - Mimer SQL (SQL/PSM) – SQL/Persistent Stored Module (implementira SQL/PSM)
  - MySQL (SQL/PSM) – SQL/Persistent Stored Module (implementira SQL/PSM)
  - Oracle (PL/SQL) – Procedural Language/SQL (na osnovi jezika Ada)
  - PostgreSQL (PL/pgSQL) – Procedural Language/PostgreSQL (na osnovi Oracle PL/SQL)
  - …

Microsoft Transact-SQL (T-SQL)

- T-SQL je razširitev jezika SQL s strani podjetij Microsoft in Sybase
- T-SQL vključuje
  - proceduralno programiranje
  - lokalne spremenljivke
  - podporne funkcije za obdelavo znakovnih nizov, datumov
  - matematične funkcije
  - spremembe stavkov DELETE in UPDATE
- T-SQL je jedro strežnika Microsoft SQL Server
  - ni popolnoma skladen s standardom SQL, vključuje znaten del standardnega SQL

Funkcije

- Funkcije (row functions) sprejemajo vhodne parametre in vračajo vrednost
- Skupine funkcij
  - Math Functions
  - String Functions
  - Date and Time Functions
  - Data Type Conversions
  - CASE Statements
  - NULL Function

Primeri funkcij

- Matematične funkcije
  - ABS, DEGREES, RAND, ACOS
  - EXP, ROUND, ASIN, LOG, SIN
  - ATN2, LOG10, SQRT, CEILING
  - FLOOR, SIGN, ATAN, PI
  - SQUARE, COS, POWER
  - TAN, COT, RADIANS
- Primer uporabe:

  SELECT ROUND (123.9994, 3)
  Rezultat: 123.9990

Primeri funkcij

- Funkcije za obdelavo znakovnih nizov
  - ASCII, NCHAR, SOUNDEX, CHAR, PATINDEX
  - SPACE, CHARINDEX, DIFFERENCE, REPLACE
  - STUFF, LEFT, REPLICATE, SUBSTRING
  - QUOTENAME, STR, LEN, REVERSE
  - UNICODE, LOWER, RIGHT
  - UPPER, LTRIM, RTRIM
- Primer uporabe:

  SELECT LEFT('abcdefg', 2)
  Rezultat: ab

Primeri funkcij

- Funkcije za delo z datumi in časi
  - DATEADD, DATEDIFF
  - DATENAME, DATEPART
  - DAY, MONTH, YEAR
  - GETDATE,GETUTCDATE
- Primer uporabe:

  SELECT GETDATE()
  Rezultat: November 4 2013 8:00AM

Primeri funkcij

- Izraz CASE omogoča vpeljavo pogojnega izvajanja v stavke SQL
- Primer:

```
SELECT title, price,
Budget = CASE price
WHEN price > 20.00 THEN 'Expensive'
WHEN price BETWEEN 10.00 AND 19.99 THEN 'Moderate'
WHEN price < 10.00 THEN 'Inexpensive'
ELSE 'Unknown'
END
FROM titles
```

Lokalne spremenljivke

- Lokalno spremenljivko označuje znak @
- Spremenljivke je potrebno pred uporabo deklarirati z DECLARE
  - npr:

    DECLARE @i INT

- Za prirejanje vrednosti uporabimo SET
  - npr:

    SET @i = 0

Stavki kontrole toka

- Stavki kontrole toka vključujejo BEGIN in END, BREAK, CONTINUE, GOTO, IF in ELSE, RETURN, WAITFOR ter WHILE

- Primer:

```
DECLARE @i INT
SET @i = 0
WHILE @i < 5
BEGIN
      PRINT 'Hello world.'
      SET @i = @i + 1
END
```

Shranjene procedure in uporabniško definirane funkcije

- Prednosti
  - možnost izvajanja vnaprej prevedene kode
  - zmanjšan promet na relaciji odjemalec-strežnik
  - ponovna uporaba kode, abstrakcija
  - centralizirano vzdrževanje
  - dodaten nadzor nad dostopom
    - pravice uporabnikov za poganjanje shranjenih procedur so neodvisne od pravice uporabnikov za dostop do tabel

Primer shranjene procedure

- CREATE PROCEDURE sp_GetInventory
  @location varchar(10)
  AS
  SELECT Product, Quantity
  FROM Inventory
  WHERE Warehouse = @location

- EXECUTE sp_GetInventory 'FL'

Primer uporabniško definirane funkcije

- CREATE FUNCTION whichContinent
  (@Country nvarchar(15))
  RETURNS varchar(30)
  AS BEGIN
  declare @Return varchar(30)
  select @return = case @Country
  when 'Argentina' then 'South America'
  when 'Belgium' then 'Europe'
  when 'Brazil' then 'South America'
  else 'Unknown'
  end
  return @return
  end

Razlika med procedurami in funkcijami

- Procedure kličemo samostojno preko stavka EXEC, funkcije kličemo iz drugega SQL stavka
- Procedure omogočajo izboljšano varnost aplikacije zaradi neodvisnega podeljevanja pravic uporabnikom, ki smejo procedure zaganjati
- Funkcija mora vselej vrniti vrednost (skalar ali tabelo)
- Procedura lahko vrača skalar, tabelo, ali pa ne vrača vrednosti

Prožilne procedure

- Prožilne procedure (Triggers) so posebne procedure, ki se sprožijo samodejno ob določenih stavkih za spremembo določene tabele
- Primer:

```
CREATE TRIGGER trigAddStudents
ON Students
FOR INSERT
AS
DECLARE' @Newname VARCHAR(100)
SELECT @Newname =(SELECT Name FROM
INSERTED)
PRINT 'THE STUDENT' + @Newname + ' IS ADDED.';
```

2.5 Modeliranje podatkov

- Podatkovni model
  - informacija o tem, kako so podatkovni elementi med seboj povezani
- Od izvedbe baze neodvisna predstavitev podatkov
- Trinivojska arhitektura
  - zunanji nivo
  - konceptualni nivo
  - notranji nivo
- Pripadajoči modeli
  - logični – opisujejo zunanjo in konceptualno shemo
  - fizični – opisujejo notranjo shemo

Tabela - ponovitev

- Predstavlja relacijo, ki je osnovni element relacijskega podatkovnega modela
- Definirana z relacijsko shemo
- Vrstice (n-terice) in stolpci (atributi)
- Vsaka vrstica je samostojna, brez odvisnosti od drugih vrstic v tabeli
- Vrstica mora predstavljati posamezno kompletno instanco množice entitet, za predstavitev katere je bila tabela ustvarjena
- Stolpci vrstice vsebujejo specifične atribute, ki definirajo posamezno entiteto

Razmerja med tabelami

- Razmerje je logična povezava med dvema tabelama
- Vrste razmerij
  - ena-proti-ena (angl. *one-to-one relationship*)
  - ena-proti-mnogo (angl. *one-to-many relationship*)
  - mnogo-proti-ena (*many-to-one relationship*)
  - mnogo-proti-mnogo (*many-to-many relationship*)
- Razmerja lahko predstavimo s pripadajočimi omejitvami
- Razmerje mnogo-proti-mnogo potrebuje za predstavitev pripadajoče omejitve tri tabele

Razmerja med tabelami /2

- Vrste razmerij
  - Razmerje ena-proti-ena (angl. one-to-one relationship)
    - najbolj enostavna povezava med tabelami
    - vsaki relaciji v eni tabeli pripada natanko ena vrstica v drugi tabeli
    - enostavno »zlepimo« stolpce prve in druge tabele glede na vrednost enega izmed stolpcev
    - isto informacijo bi lahko predstavili z eno večjo tabelo
    - smisel takšne povezave je v pohitritvi dela, če je baza manjša, in različnih pravicah posameznih uporabnikov, ki do baze dostopajo

Razmerja med tabelami /3

- Razmerje ena-proti-mnogo (angl. one-to-many relationship)
  - najpogostejša povezava med tabelami
  - ena od vrstic v prvi tabeli je povezana z več vrsticami v drugi tabeli, medtem ko ima vsaka vrstica v drugi tabeli le eno pripadajočo vrstico v prvi tabeli
  - vrstice so običajno povezane preko primarnega ključa v prvi tabeli in enega izmed stolpcev – tuji ključ – v drugi tabeli (ta seveda ni primarni ključ, saj lahko obstaja več zapisov, ki imajo enako vrednost v tem stolpcu, in se na ta način sklicujejo na en zapis v prvi tabeli)
  - s formalnega stališča to pomeni, da so relacije v prvi tabeli funkcionalno odvisne od relacij v drugi tabeli

Razmerja med tabelami /4

- Razmerje mnogo-proti-ena (many-to-one relationship)
  - je nasprotje razmerja ena-proti-mnogo
  - če na povezavo gledamo z druge strani in vidimo razmerje ena-proti-mnogo, pravimo, da je povezava refleksivna; v večini SUPB so podprte le refleksivne preslikave, zato lahko vse povezave opišemo kot razmerja ena-proti-mnogo
- Razmerje mnogo-proti-mnogo (many-to-many relationship)
  - ni ga mogoče predstaviti kot enostavne povezave med dvema stolpcema v dveh tabelah
  - vsaki relaciji iz prve tabele pripada več relacij v drugi tabeli in obratno; problem rešimo z dodatkom nove vmesne tabele, ki je z obema osnovnima tabelama povezana s razmerjema mnogo-proti-ena

Načrtovanje relacijske podatkovne baze

- Bistveno vprašanje: Kako grupirati atribute v ustrezne relacijske sheme?
  - da ne bo težav pri ažuriranju relacij
  - da pri dopolnjevanju podatkovne baze z novimi atributi ne bo treba bistveno spreminjati obstoječih relacijskih shem
    - s tem ogrozili obstoječo rabo podatkovne baze
    - potrebovali bi nove uporabniške programe (programi so prirejeni obstoječi strukturi baze)
- Fokus na konceptualni nivo
  - uporabimo logični podatkovni model
  - podatkovni model predstavlja tabele, njihove atribute in razmerja med tabelami

Diagram entiteta-razmerje

- Diagram entiteta-razmerje (Entity-Relationship Diagram – ERD)
  - pomaga načrtovalcu pri načrtovanju relacijske podatkovne baze
  - terminologija: množica entitet (tabela) -> entiteta; entiteta (n-terica) -> entitetna instanca
- Pomen
  - primarnega ključa
    - za identifikacijo posameznih vrstic v tabeli
    - poenostavlja združevanje tabel
  - tujega ključa (ki se referencira na primarni ključ druge tabele)
    - omogoča integriteto razmerij (referential integrity)

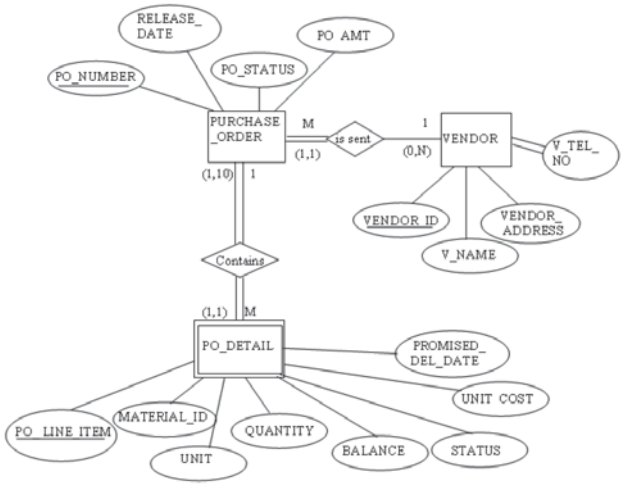
Osnovni elementi diagrama

- Entitete – pravokotniki
  - šibka entiteta – pravokotnik z dvojnim robom
    - njen obstoj je odvisen od druge entitete
    - otrok v razmerju starši-otroci
    - primarni ključ šibke entitete je vsaj deloma izpeljan iz primarnega ključa starševske entitete
    - primer: Purchase_order in Po_detail
  - močna entiteta – pravokotnik z enojnim robom
    - njen obstoj ni odvisen od druge entitete
- Atributi – ovali
  - s črto povezani z entitetami, ki jim pripadajo
  - dvojna črta označuje večvrednostne atribute
  - primarni ključ označimo s podčrtanjem atributov

Osnovni elementi diagrama

- Razmerja
  - vključujejo tri komponente
    - povezave – črte med entitetami
    - oznako razmerja – romb z tekstovno oznako
    - številnost (kardinalnost) – številke ali simboli na koncih povezave
      - ena-proti-mnogo (1:M) – najpogostejše razmerje: na strani M je lahko 0, 1, ali več instanc entitete
      - ena-proti-ena (1:1)
      - mnogo-proti-mnogo (M:N)
    - omejitve številnosti lahko ponazorimo z dodatnimi oznakami v oklepajih – odraz pravil, ki veljajo v modeliranem procesu
      - npr. (1,10) pomeni od 1 do 10 pripadajočih instanc
      - omejitev se nanaša na število instanc entitete na drugi strani

Primer diagrama



Primer diagrama

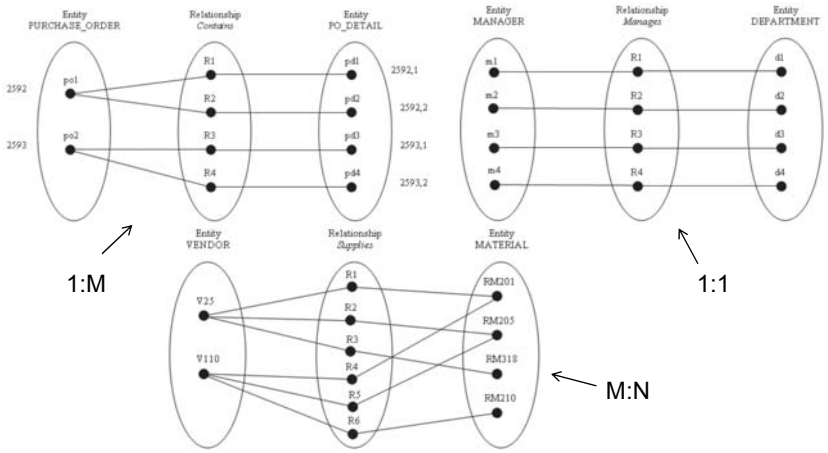
- Opombe
  - diagram ne prikazuje tujih ključev
    - predpostavljamo njihovo vpeljavo, kjer so entitete v medsebojnem razmerju (npr. vendor_id v Purchase_order)
    - pri šibkih entitetah se ustrezno dopolni primarni ključ (npr. v Po_detail bo primarni ključ vseboval tudi po_number)
  - na tej stopnji se ne ukvarjamo z dejanskim zapisom atributov
    - npr. naslov bomo pri izvedbi razgradili v ulico, kraj itd., a v diagramu E-R naslov obravnavamo kot en atribut
    - dopuščamo tudi večvrednostne atribute, npr. telefonskih številk je lahko več; takšne atribute v diagramu povežemo z dvojno črto

Sodelovanje entitet v razmerjih

- Odvisno od pravil, ki veljajo v modeliranem procesu
  - opcijsko ali delno
    - pojav ene entitete ne zahteva pojava druge entitete, ki je s prvo v razmerju
    - nekatere instance prve entitete so povezane, druge ne
    - v diagramu to označimo z enojno črto
  - obvezno
    - pojav ene entitete zahteva tudi pojav druge entitete, ki je s prvo v razmerju
    - npr. pojav instance v Po_detail je vselej vezan na instanco Purchase_order in obratno
    - takšno razmerje v diagramu označimo z dvojno črto

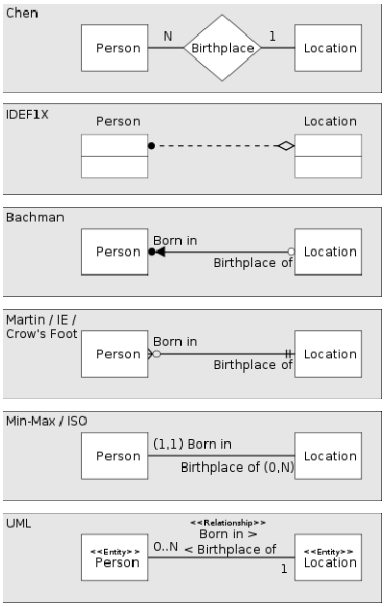
Ilustracija številnosti

- Semantična mreža (semantic net diagram)



Različne predstavitve diagrama E-R

- Diagram je predlagal Peter Chen l. 1976
- V uporabi so tudi druge notacije
- Številna programska orodja



LMSV LAMS

Normalizacija

LMSV LAMS

- Proces učinkovitega organiziranja podatkov v bazi
- Cilj: vsaka tabela v bazi naj opisuje eno samo stvar
  - eliminacija redundantnih podatkov
    - npr. shranjevanje istega podatka v več vrsticah tabele
    - npr. shranjevanje istega podatka v več kot eni tabeli (izjema so tuji ključi)
  - zagotavljanje smiselnih podatkovnih odvisnosti
    - v tabeli naj bi bili shranjeni le podatki, ki so med seboj logično povezani
- Smisel
  - zmanjšanje obsega baze
  - zagotavljanje logičnega shranjevanja podatkov

Težave pri ažuriranju

- Pokažejo se pri spreminjanju vsebine relacij (tabel), če te niso pravilno zasnovane
- Dodajanje:
  - če se pri dodajanju enega podatka v takšno tabelo zmotimo, lahko postane tabela nekonsistentna - to lahko »pokvari« tudi že obstoječe podatke
- Spreminjanje:
  - pri slabo zasnovani bazi je isti podatek treba spremeniti večkrat
- Brisanje:
  - z brisanjem ene vstice zbrišemo več podatkov

Prva normalna oblika – 1NO

- First Normal Form – 1NF
- Eliminacija ponavljajočih se atributov
  - zagotovimo, da ne obstajajo atributi ali skupine atributov, ki bi imele več vrednosti pri isti vrednosti ostalih atributov (na presečišču ene vrstice in enega stolpca je več vrednosti)
- Identifikacija vsake vrstice v tabeli s primarnim ključem
  - določimo funkcionalne odvisnosti
  - določimo primarni ključ

Druga normalna oblika – 2NO

- Second Normal Form – 2NF
- Odstranitev podmnožic podatkov, ki se ponavljajo v več vrsticah tabele in zapis le-teh v ločene tabele
  - odstranimo parcialne odvisnosti, kjer bi bil posamezen atribut, ki ni del ključa, funkcionalno odvisen le od dela primarnega ključa
- Ustvarjanje razmerij med novimi in prvotnimi tabelami z uporabo tujih ključev
- Z opisanim postopkom zmanjšamo količino redundantnih podatkov

Tretja normalna oblika – 3NO

- Third Normal Form – 3NF
- Zagotovitev, da je unikatnost vrstice pogojena s ključem, celotnim ključem in samo ključem
  - relacija ne vsebuje tranzitivnih funkcionalnih odvisnosti – odvisnosti med atributi, ki niso del primarnega ključa
- Vključuje zahteve 1NO in 2NO
- Odstranitev stolpcev, ki niso v celoti odvisni od primarnega ključa
- Obstajata se četrta in peta normalna oblika

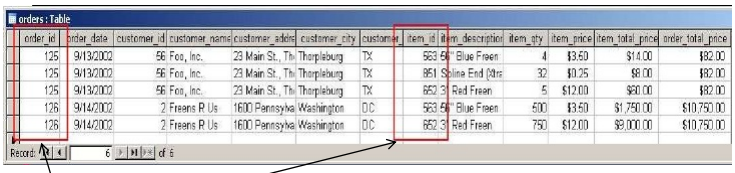
Primer normalizacije

- Primer tabele z naročili – želimo jo normalizirati v 3NO
- Nenormalizirana tabela:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Invoice No.	Date	Cust. No.	Cust. Name	Cust. Address	Cust. City	Cust. State	Item ID	Item Descri	Item Qty.	Item Price	Item Total	Order Total Price
2	125	9/13/2002	56	Foo, Inc.	23 Main St., Thorpleburg	Thorpleburg	TX	563	96" Blue Fre	4	$ 3.50	$ 14.00	$ 82.00
3								851	Spline End I	32	$ 0.25	$ 8.00	$ 82.00
4								852	3" Red Free	5	$ 12.00	$ 60.00	$ 82.00
5	126	9/14/2002	2	Freens R Us	1600 Pennsylvania Avenue	Washington	DC	563	96" Blue Fre	500	$ 3.50	$1,750.00	$ 10,750.00
6								852	3" Red Free	750	$ 12.00	$9,000.00	$ 10,750.00

- Med postopkom normalizacije bomo ustvarili tudi E-R diagram
- Ponovimo
  - 1NO: brez ponavljajočih se elementov ali skupin
  - 2NO: brez delnih odvisnosti pri sestavljenem ključu
  - 3NO: brez odvisnosti od atributov izven ključa

Primer: 1NO



order_id	order_date	customer_id	customer_name	customer_addre	customer_city	customer	item_id	item_description	item_qty	item_price	item_total_price	order_total_price
125	9/13/2002	56	Foo, Inc.	23 Main St., Th	Thorpleburg	TX	563	96" Blue Freen	4	$3.50	$14.00	$82.00
125	9/13/2002	56	Foo, Inc.	23 Main St., Th	Thorpleburg	TX	851	Spline End (Qtr	32	$0.25	$8.00	$82.00
125	9/13/2002	56	Foo, Inc.	23 Main St., Th	Thorpleburg	TX	852	3" Red Freen	5	$12.00	$60.00	$82.00
126	9/14/2002	2	Freens R Us	1600 Pennsylva	Washington	DC	563	96" Blue Freen	500	$3.50	$1,750.00	$10,750.00
126	9/14/2002	2	Freens R Us	1600 Pennsylva	Washington	DC	852	3" Red Freen	750	$12.00	$9,000.00	$10,750.00

primarni ključ

Funkcionalne odvisnosti:
{ order_id → (order_date, customer_id, customer_name, customer_address, customer_city, customer_state, order_total_price),
item_id → (item_description, item_price),
(order_id, item_id) → (item_qty, item_total_price) }

Primer: 1NO

- S pretvorbo v 1NO smo izpolnili dve zahtevi
  - vrstica tabele ne vsebuje ponavljajočih se skupin podobnih podatkov (atomarnost)
  - vsaka vrstica je identificirana s primarnim ključem (order_id, item_id)

orders
order_id (PK)
order_date
customer_id
customer_name
customer_address
customer_city
customer_state
item_id (PK)
item_description
item_qty
item_price
item_total_price
order_total_price

Primer: 2NO

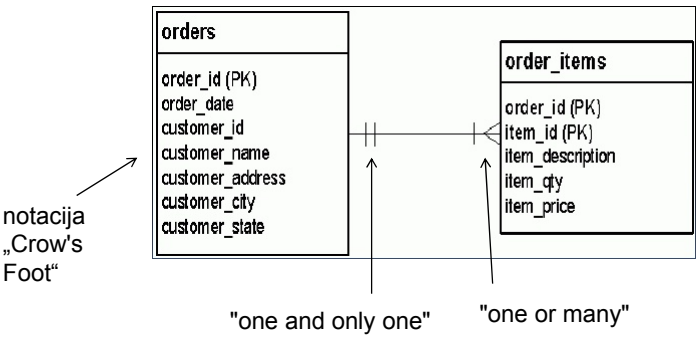
- Analiziramo tabelo naročil
  - iščemo odvisnosti od dela ključa (samo od order_id ali samo od item_id)

orders
order_id (PK)
order_date ✗
customer_id ?
customer_name ?
customer_address ?
customer_city ?
customer_state ?
item_id (PK)
item_description ✗
item_qty ✓
item_price ✗
~~item_total_price~~
~~order_total_price~~

Primer: 2NO

LMSV LAMS

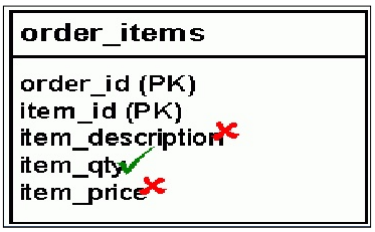
- Ugotovitve
  - vsako naročilo je lahko povezano z več naročenimi artikli, a vedno vsaj z enim
  - vsak naročen artikel je povezan z natanko enim naročilom



Primer: 2NO

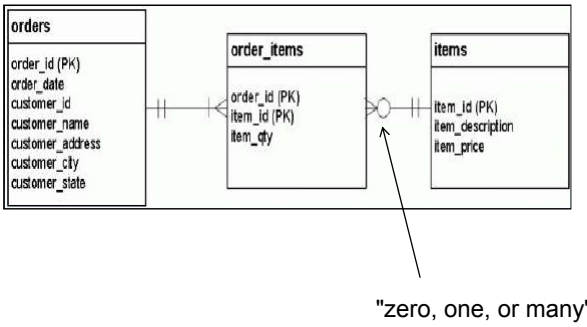
LMSV LAMS

- Testiranje 2NO je potrebno le za tabele s sestavljenim ključem
  - tabela orders nima sestavljenega ključa, je že v 2NO
  - tabela order_items ima sestavljen ključ, potrebno je ponovno preveriti skladnost z 2NO



Primer: 2NO

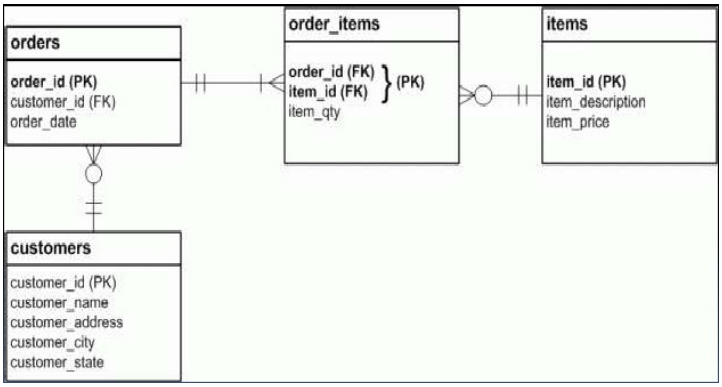
- Vsako naročilo lahko vsebuje več artiklov, vsak artikel lahko pripada več naročilom



Primer: 3NO

- Iščemo odvisnosti od atributov, ki niso del ključa
  - informacija o strankah se ponavlja
  - če ista stranka izda več naročil, bomo morali vse podatke o stranki vpisati večkrat
  - vsako naročilo pride s strani ene in samo ene stranke
  - vsaka stranka lahko izda poljubno število naročil, vključno z nič

Primer: 3NO



Potrebnost ali nepotrebnost normalizacije

- Odvisno od namena baze
  - vnos podatkov ali poročanje
- Normalizirane baze zahtevajo kompleksne poizvedbe za potrebe poročanja ali analize podatkov
- Pri snovanju novih baz za podporo procesom bazo običajno normaliziramo
- Pri bazah namenjenih analizi podatkov dopustimo posamezne nenormalizirane tabele
  - nikoli ne kršimo prve in druge normalne oblike
  - višjim normalnim oblikam se včasih odrečemo na račun doseganja boljše učinkovitosti