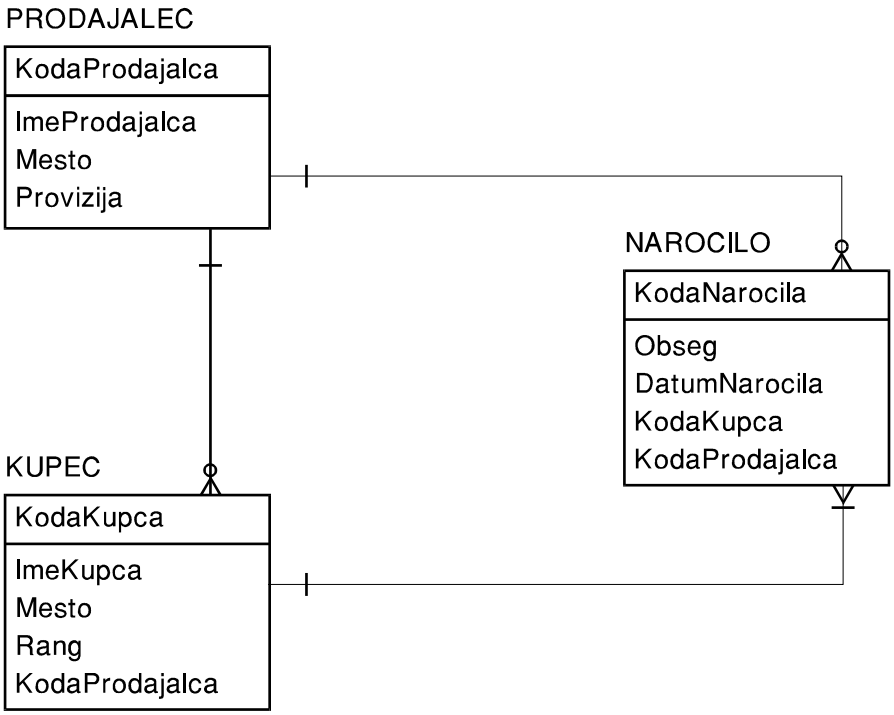
SQL - Primeri in povzetki

ER diagram



Slika 1: ER diagram

Vsebina treh tabel

PRODAJALEC			
KodaProdajalca	ImeProdajalca	Mesto	Provizija
1001	Sisek	Ljubljana	0.12
1002	Kres	Novo Mesto	0.13
1004	Palek	Ljubljana	0.11
1007	Rakus	Maribor	0.15
1006	Dekar	Celje	0.21
1003	Aron	Portoroz	0.10

KUPEC				
KodaKupca	ImeKupca	Mesto	Rang	KodaProdajalca
2001	Hord	Ljubljana	100	1001
2002	Gregr	Rogla	200	1003
2003	Lukic	Novo Mesto	200	1002
2004	Porkaro	Koper	300	1002
2006	Cikl	Ljubljana	100	1001
2008	Curcio	Novo Mesto	300	1007
2007	Pristl	Rogla	100	1004

NAROCILO				
KodaNarocila	Obseg	DatumNarocila	KodaKupca	KodaProdajalca
3001	18.69	02.10.1990	2008	1007
3003	767.19	03.10.1990	2001	1001
3002	1900.1	09.10.1990	2007	1004
3005	5160.45	10.10.1990	2003	1002
3006	1098.16	11.10.1990	2008	1007
3009	1713.23	12.10.1990	2002	1003
3007	75.75	12.10.1990	2004	1002
3008	4723	14.10.1990	2006	1001
3010	1309.95	22.10.1990	2004	1002
3011	9891.88	23.10.1990	2006	1001

Zajemanje informacij iz tabel s pomočjo jezika SQL

Primeri

Vprašanje izpiše kodo, obseg in datum vseh naročil iz tabele Narocilo (vertikalna izbira - projekcija):

```
SELECT      KodaNarocila, Obseg, DatumNarocila
FROM        Narocilo;
```

Vprašanje izpiše vse kupce, ki kupujejo pri prodajalcu z kodo 1001 (horizontalna izbira - selekcija):

```
SELECT      *
FROM        Kupec
WHERE       KodaProdajalca = 1001;
```

Vprašanje izpiše vsa polja tabele Prodajalec (projekcija) v naslednjem vrstnem redu: Mesto, ImeProdajalca, KodaProdajalca, Provizija:

```
SELECT      Mesto, ImeProdajalca, KodaProdajalca, Provizija
FROM        Prodajalec;
```

Vprašanje izpiše rang in ime vseh kupcev (projekcija) iz mesta Ljubljana (selekcija):

```
SELECT      Rang, ImeKupca
FROM        Kupec
WHERE       Mesto = 'Ljubljana';
```

Vprašanje izpiše kode vseh kupcev, ki so naročili izdelek ali storitev (so zapisani v tabel Narocilo), brez ponavljanja vrednosti (uporaba argumenta `DISTINCT`):

```
SELECT      DISTINCT KodaKupca
FROM        Narocilo;
```

Povzetek

Člena `WHERE` in `HAVING` oblikujeta horizontalno podmnožico podatkov oziroma izbiro (*SELECTion*). V členu `SELECT` oblikujemo vertikalno podmnožico podatkov oziroma projekcijo (*projection*).

Prva dva člena ukaza `SELECT` (`SELECT` in `FROM`) sta obvezna, vsi ostali so opcijski.

Vprašanje in vse ostale interaktivne SQL ukaze zaključimo s podpičjem.

V vprašanju lahko uporabimo argument `DISTINCT`, ki nam omogoči odpravo dvojnih vrednosti pri izpisu. `DISTINCT` uporabljamo samo z polji, ki niso primarni ključi.

Parameter `DISTINCT` uredi izpis v naraščajočem vrstnem redu (*ascending*).

Uporabljene predikate v členu `WHERE` nam ni potrebno navajati v členu `SELECT`.

Znakovni časovni predikat moramo navesti med enojnimi narekovaji ('), (npr. '05.06.1991').

Uporaba primerjalnih in logičnih (*boolean*) operatorjev pri oblikovanju predikatov

Primeri

Vprašanje izpiše vsa naročila večja od 1000 enot:

```
SELECT      *
FROM        Narocilo
WHERE       Obseg > 1000.00;
```

Vprašanje izpiše vse kupce, ki imajo rang večji od 100, in vse kupce iz Celja ne glede na njihov rang:

```
SELECT      *
FROM        Kupec
WHERE       Rang > 100
OR          Mesto = 'Celje';
```

Vprašanje izpiše vsa naročila, katerih obseg je večji od 1000 enot ali naročila, ki jih ni naročil kupec z kodo 2003 in niso izvršena dne 3.10.1990:

```
SELECT      *
FROM        Narocilo
WHERE       (Obseg > 1000)
OR          NOT (DatumNarocila = '03.10.1990' AND KodaKupca > 2003);
```

Povzetek
Pri primerjavi znakovnih vrednosti je uporaba primerjalnih operatorjev odvisna od formata, ki ga uporabljamo (ASCII ali EBCDIC).
Vrednost, ki jo v členu `WHERE` primerjamo z poljem iz tabele, mora biti enakega tipa kot podatkovni tip polja.
Ne moremo primerjati alfanumeričnih vrednosti z numeričnimi vrednostmi.
Logični izrazi so tisti izrazi, ki zavzamejo vrednost resnično (*true*) ali neresnično (*false*).
Logični operator `NOT` lahko stoji v členu `WHERE` pred predikatom.

Uporaba specialnih operatorjev v pogojih

Primeri

Vprašanji izpišeta vsa naročila izvedena 3. in 4. oktobra 1990:

```
SELECT      *
FROM        Narocilo
WHERE       DatumNarocila IN ('03.10.1990', '04.10.1990');
```

ali

```
SELECT      *
FROM        Narocilo
WHERE       DatumNarocila BETWEEN '03.10.1990' AND '04.10.1990';
```

Vprašanje izpiše vse kupce dveh prodajalcev Sisek in Rakus (za prodajalca moramo vedeti njegovo kodo):

```
SELECT      *
FROM        Kupec
WHERE       KodaProdajalca IN (1001, 1007);
```

Vprašanje izpiše imena vseh kupcev od črke A do vključno črke G:

```
SELECT      *
FROM        Kupec
WHERE       ImeKupca BETWEEN 'A' AND 'H';
```

Vprašanje izpiše vse kupce katerih ime se začne z črko P:

```
SELECT      *
FROM        Kupec
WHERE       ImeKupca LIKE 'P%';
```

Vprašanje izpiše vse kupce za katere nismo vnesli mesta:

```
SELECT      *
FROM        Kupec
WHERE       Mesto IS NULL;
```

Povzetek

Operator IN eksplicitno definira množico vseh vrednosti, v kateri se mora vrednost polja nahajati, da lahko postane predikat člena `WHERE` pravilen in zapis vključen v izpisu.

Operator `BETWEEN` definira interval vrednosti v kateri mora biti vrednost polja, da lahko postane predikat člena `WHERE` pravilen in zapis vključen v izpisu.

Operator `BETWEEN` je občutljiv na vrstni red, tako da mora veljati *vrednost1* <= *vrednost2*.

Operator `BETWEEN` je inkluziven. Vključuje tudi robne vrednosti.

Operator `LIKE` lahko uporabljamo le z polji tipa CHAR ali VARCHAR.

Podčrtaj (_) predstavlja zamenjavo enega znaka v podnizu.

Procentni znak (%) predstavlja zamenjavo poljubnega niza.

Operator `IS NULL` uporabljamo v primeru, ko želimo ločiti neznano vrednost od vrednosti neresnično (*false*).

Za vsemi specialnimi operatorji lahko uporabljamo Boolean `NOT`.

Uporaba skupinskih funkcij v vprašanjih

Primeri

Vprašanje prešteje vsa naročila izvršena 3. oktobra, 1990:

```
SELECT      COUNT(*)
FROM        Narocilo
WHERE       DatumNarocila = '03.10.1990';
```

Vprašanje prešteje vse različne ne NULL vrednosti mest v tabeli Kupec:

```
SELECT      COUNT(DISTINCT Mesto)
FROM        Kupec;
```

Vprašanje izpiše najmanjše naročilo vsakega kupca:

```
SELECT      KodaKupca, MIN(Obseg)
FROM        Narocilo
GROUP BY    KodaKupca;
```

Vprašanje izpiše ime prvega kupca, katerega ime se začne z črko P:

```
SELECT      MIN(ImeKupca)
FROM        Kupec
WHERE       ImeKupca LIKE 'P%';
```

Vprašanje izpiše najvišji rang kupca za vsako mesto:

```
SELECT      Mesto, MAX(Rang)
FROM        Kupec
GROUP BY    Mesto;
```

Vprašanje prešteje vso prodajo vsakega prodajalca za vsak dan posebej. V primeru, da je opravil prodajalec na dan več prodaj, štejemo le eno prodajo:

```
SELECT      DatumNarocila, COUNT(DISTINCT KodaProdajalca)
FROM        Narocilo
GROUP BY    DatumNarocila;
```

Povzetek

Skupinske funkcije vrnejo eno vrednost za določeno skupino zapisov v tabeli.

Argument skupinske funkcije je polje. V primeru `SUM` in `AVG` lahko uporabljamo le numerična polja, pri vseh ostalih funkcijah pa tudi znakovne nize.

Vprašanje, ki vsebuje skupinsko funkcijo, vrne le eno vrednost, ne glede na število zapisov v tabeli. Prav zaradi tega ne moremo v takšnem vprašanju izpisovati ostalih polj.

Funkcijo `COUNT` lahko uporabljamo tudi pri štetju zapisov tabele in ne le pri štetju vrednosti izbranega polja.

`GROUP BY` priredi vrednost skupinske funkcije vsaki skupini posebej.

Ko uporabimo za argument člena `GROUP BY` primarni ključ, nam mora vprašanje vrniti prav toliko vrednosti kot je zapisov v tabeli.

Skupinskih funkcij ne moremo uporabljati v pogoju oziroma členu `WHERE`, razen pri podvprašanjih.

Argumenti člena `HAVING` morajo imeti le eno vrednost za eno skupino zapisov.

Zajemanje podatkov iz več tabel naenkrat

Primeri

Vprašanje izpiše kodo naročila in ime kupca, ki je naročilo izvršil:

```
SELECT      KodaNarocila, ImeKupca
FROM        Narocilo, Kupec
WHERE       Kupec.KodaKupca = Narocilo.KodaKupca;
```

Vprašanje izpiše imena kupcev in prodajalcev za vsako naročilo, urejeno po kodi naročila:

```
SELECT      KodaNarocila, ImeKupca, ImeProdajalca
FROM        Narocilo, Kupec, Prodajalec
WHERE       Kupec.KodaKupca = Narocilo.KodaKupca
AND         Prodajalec.KodaProdajalca = Narocilo.KodaProdajalca
ORDER BY    KodaNarocila;
```

Vprašanje izpiše polja ImeKupca, ImeProdajalca in Provizija za vse kupce servisirane iz strani prodajalcev, ki imajo provizija nad 12%:

```
SELECT      ImeKupca, ImeProdajalca, Provizija
FROM        Prodajalec, Kupec
WHERE       Prodajalec.KodaProdajalca = Kupec.KodaProdajalca
AND         Provizija > 0.12;
```

Vprašanje izračuna obseg prodajalčeve provizije za vsako izvršeno prodajo kupcev z rangom večjim od 100:

```
SELECT      KodaNarocila, Provizija*Obseg
FROM        Prodajalec, Narocilo, Kupec
WHERE       Rang > 100
AND         Narocilo.KodaKupca = Kupec.KodaKupca
AND         Narocilo.KodaProdajalca = Prodajalec.KodaProdajalca;
```

Povzetek

Tabele, ki jih združujemo, navedemo v členu `FROM`.

Ko imamo v obeh tabelah polje z enakim imenom, moramo uporabiti prefiksno obliko v členu `SELECT`.

Osnovna oblika združitve je kartezični produkt vseh zapisov tabel, ki jih združujemo.

Vprašanje imenujemo naravna združitev (*natural join*), ko v izpisu projeciramo le eno polje, ki je v obeh tabelah enako.

Združitveno vprašanje, ki uporablja predikate, kateri slonijo na matematičnih izrazih enakosti, imenujemo *equijoin*.

Združitev, v katerem uporabimo relacijski operator, ki je različen od enakosti, se imenuje *thetajoin*.

Ko združujemo *n* tabel, potrebujemo najmanj *n-1* pogojev v členu `WHERE`.

Združevanje tabele same s seboj

Primeri

Vprašanje izpiše vse pare prodajalcev, ki živijo v istem mestu (npr. Sisek = Palek). Izpis izloči kombinacije prodajalcev samih s seboj (npr. Kres = Kres) in obratne izpise prodajalcev (npr. Palek = Sisek):

```
SELECT    A.ImeProdajalca, B.ImeProdajalca
FROM      Prodajalec A, Prodajalec B
WHERE     A.Mesto = B.Mesto
AND       A.ImeProdajalca < B.ImeProdajalca;
```

Vprašanje izpiše vse pare naročil za vsakega kupca. Izpis izloči kombinacije naročil samih s seboj in obratne izpise:

```
SELECT    ImeKupca, A.KodaNarocila, B.KodaNarocila
FROM      Narocilo A, Narocilo B, Kupec
WHERE     A.KodaKupca = B.KodaKupca
AND       A.KodaKupca = Kupec.KodaKupca
AND       A.KodaNarocila < B.KodaNarocila;
```

Vprašanje izpiše vsa imena in mesta kupcev z enakim rangom kot kupec Hord. Vprašanje uporablja Hordovo kodo in ne vrednosti ranga. Tako oblikovano vprašanje je uporabno tudi po spremembi Hordovega ranga:

```
SELECT    A.ImeKupca, A.Mesto
FROM      Kupec A, Kupec B
WHERE     A.Rang = B.Rang
AND       B.KodaKupca = 2001;
```

Povzetek

Združitev tabele seme s seboj si predstavljamo kot združevanje dveh kopij ene tabele.

Privzeta imena (*alias*) definiramo v členu `FROM` takoj za imenom tabele.

Redundanci v izpisu se izognemo tako, da uvedemo vrstni red v izpisu oziroma naredimo predikat nesimetričen.

Vgnezdena vprašanja

Primeri

Vprašanje izpiše vsa naročila kupca Porkaro. Poznamo le ime kupca in ne njegove kodne številke:

```
SELECT      *
FROM        Narocilo
WHERE       KodaKupca =
            (SELECT     KodaKupca
            FROM        Kupec
            WHERE       ImeKupca = 'Porkaro');
```

ali

```
SELECT      *
FROM        Narocilo
WHERE       KodaKupca IN
            (SELECT     KodaKupca
            FROM        Kupec
            WHERE       ImeKupca = 'Porkaro');
```

Vprašanje izpiše vsa imena in range vseh kupcev, ki so izvršili naročila nad povprečjem vseh naročil:

```
SELECT      DISTINCT ImeKupca, Rang
FROM        Kupec, Narocilo
WHERE       Obseg >
            (SELECT     AVG(Obseg)
            FROM        Narocilo)
AND         Narocilo.KodaKupca = Kupec.KodaKupca;
```

Vprašanje izpiše vsoto vseh prodaj tistih prodajalcev, ki so prodali več, kot je obseg največjega naročila v tabeli Narocilo:

```
SELECT      KodaProdajalca, SUM(Obseg)
FROM        Narocilo
GROUP BY    KodaProdajalca
HAVING      SUM(Obseg) >
            (SELECT     MAX(Obseg)
            FROM        Narocilo);
```

Povzetek

Podvprašanje mora izbrati v členu `SELECT` le eno polje, katerga vrednosti nato vrne glavnemu vprašanju.

V podvprašanju ne moremo navesti projekcije na vseh poljih tabele (`SELECT *`), razen pri podvprašanjih uporabljenih z operatorjem `EXISTS`.

Podvprašanje mora vrniti eno in le eno vrednost za izbrano polje, razen v primeru uporabe operatorja `IN`.

`DISTINCT` omogoči, da notranje vprašanje vrne le eno vrednost.

Skupinska funkcija vrne za poljubno število zapisov natanko eno vrednost.

Uporaba člena `GROUP BY` v podvprašanjih ni dovoljena.

Operator IN uporabljamo v primeru podvprašanja, ki vrne več vrednosti.

V podvprašanju, za katerega vemo, da mora vrniti le eno vrednost, uporabljamo z relacijskim operatorjem enakosti (=) in ne z operatorjem `IN`.

Pogledi

Primeri

Izpišite tiste kupce, ki imajo vsoto naročil večje, kot je povprečna vsota naročil:

```
SELECT      kodakupca, SUM(obseg)
FROM        Narocilo
GROUP BY    kodakupca;
```

Izpiše le vsote po posameznih kupcih.

Želeli pa bi imeti izpisano le tiste vsote, ki so večje od povprečja. Zato potrebujemo »začasno« tabelo – pogled, ki ga izdelamo kot (skupinske funkcije je potrebno poimenovai z aliasom – v našem premeru jo bomo poimenovali kot »VSOTA):

```
CREATE VIEW  vsoteprodaje AS
SELECT       kodakupca, SUM(obseg) VSOTA
FROM         Narocilo
GROUP BY     kodakupca;
```

Sedaj lahko iz te navidezne tabele izberemo tiste zapise, ki ustrezajo pogoju (vsota večja od povprečja vsot):

```
SELECT       kodakupca, vsota
FROM         vsoteprodaje
WHERE        vsota >
(SELECT      AVG(vsota)
FROM         vsoteprodaje);
```