Poizvedovanje z jezikom SQL

Osnove podatkovnih baz

2. letnik univerzitetnega študija na FRI

2019/20

Poizvedovalni jezik SQL (MySQL 5.7)

Data Manipulation Language (DML):

CALL, **DELETE**, DO, HANDLER, **INSERT**, LOAD DATA INFILE, LOAD XML, REPLACE, **SELECT**, **UPDATE**

Data Definition Language (DDL):

ALTER DATABASE, ALTER EVENT, ALTER FUNCTION, ALTER PROCEDURE, ALTER SERVER, **ALTER TABLE**, ALTER VIEW, **CREATE DATABASE**, CREATE EVENT, **CREATE INDEX**, CREATE PROCEDURE, CREATE FUNCTION, CREATE SERVER, **CREATE TABLE**, CREATE TABLESPACE, CREATE TRIGGER, **CREATE VIEW**, DROP DATABASE, DROP EVENT, DROP FUNCTION, DROP INDEX, DROP PROCEDURE, DROP FUNCTION, DROP SERVER, **DROP TABLE**, DROP TABLESPACE, DROP TRIGGER, DROP VIEW, RENAME TABLE, TRUNCATE TABLE

DESCRIBE, EXPLAIN, HELP, USE

Poizvedovalni jezik SQL (MySQL 5.7)

Data Control Language (DCL):

SET, SHOW, **ALTER USER**, **CREATE USER**, **DROP USER**, **GRANT**, RENAME USER, **REVOKE**, SET PASSWORD, ANALYZE TABLE, CHECK TABLE, CHECKSUM TABLE, OPTIMIZE TABLE, REPAIR TABLE, CREATE FUNCTION, DROP FUNCTION, INSTALL PLUGIN, UNINSTALL PLUGIN, BINLOG, CACHE INDEX, FLUSH, KILL, LOAD INDEX INTO CACHE, RESET

Transaction Processing Option (TPO):

START TRANSACTION, **COMMIT**, **ROLLBACK**, SAVEPOINT, ROLLBACK TO SAVEPOINT, RELEASE SAVEPOINT, LOCK TABLES, UNLOCK TABLES, SET TRANSACTION

PURGE BINARY LOGS, RESET MASTER, SET sql_log_bin, CHANGE MASTER TO, CHANGE REPLICATION FILTER, MASTER_POS_WAIT, RESET SLAVE, SET GLOBAL sql_slave_skip_counter, START SLAVE, STOP SLAVE, START GROUP_REPLICATION, STOP GROUP_REPLICATION

Oblika stavka SELECT (MySQL 5.7)

```
SELECT [ALL | DISTINCT]
    izraz [, izraz ...]
    [FROM stiki_tabel [, stiki_tabel ...]
    [WHERE pogoji_nad_stolpci]
    [GROUP BY {stolpci | izrazi}]
    [HAVING pogoji_nad_skupinami]
    [ORDER BY {stolpci | izrazi} [ASC | DESC]]
    [LIMIT [preskoci_vrstic,] stevilo_vrstic]]
```

Oblike stavka SELECT (MySQL 5.7)

Projekcija $\pi_{A_1,\ldots,A_n}(T)$: **SELECT** $A_1, \ldots, A_n$ **FROM** T

Selekcija $\pi_{A_1,\ldots,A_n}(\sigma_P(T))$: **SELECT** $A_1, \ldots, A_n$ **FROM** T **WHERE** P

Kartezični produkt $\pi_{A_1,\ldots,A_n,B_1,\ldots,B_m}(T_1 \times T_2)$:

SELECT $A_1, \ldots, A_n, B_1, \ldots, B_m$ **FROM** T_1, T_2

Pogojni stik $\pi_{A_1,\ldots,A_n,B_1,\ldots,B_m}(T_1 \bowtie_{T_1.A_i=T_2.B_j} T_2)$:

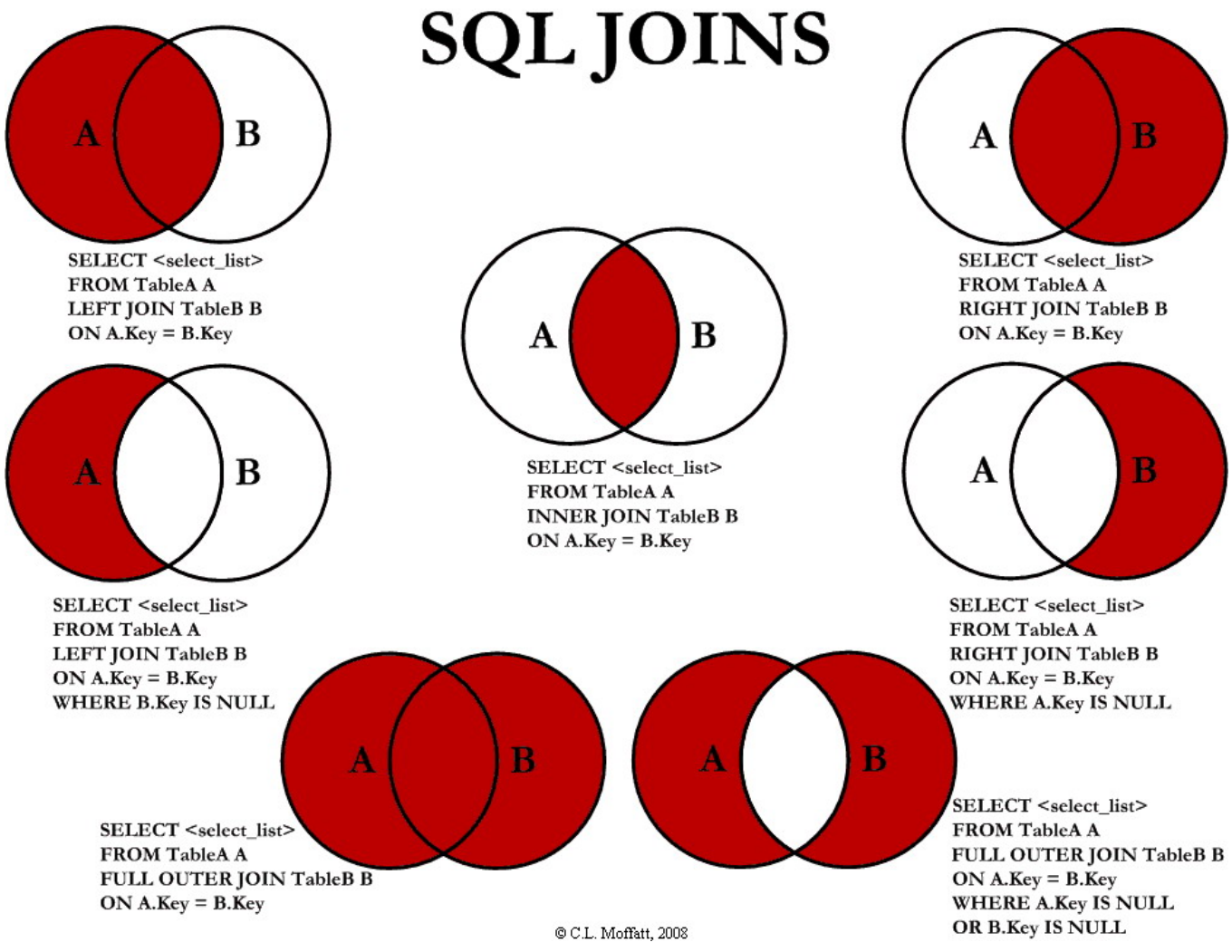
SELECT $A_1, \ldots, A_n, B_1, \ldots, B_m$ **FROM** T_1 **JOIN** T_2 **ON** $T_1.A_i = T_2.B_j$

Pol-odprti stik $\pi_{A_1,\ldots,A_n,B_1,\ldots,B_m}(T_1 \rtimes_{T_1.A_i=T_2.B_j} T_2)$:

SELECT $A_1, \ldots, A_n, B_1, \ldots, B_m$ **FROM** T_1 **LEFT JOIN** T_2 **ON** $T_1.A_i = T_2.B_j$

Naravni stik $T_1 \bowtie T_2$: **SELECT * FROM** T_1 **NATURAL JOIN** T_2

Stične operacije SELECT (MySQL 5.7)

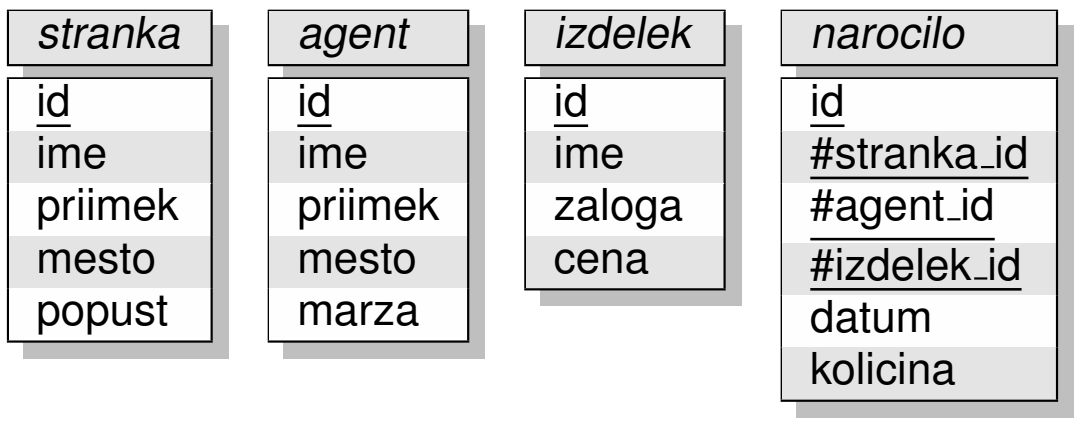


Primerjalni operatorji SELECT (MySQL 5.7)

$=$	je enako
$!=$ ali <>	je različno
$>$ in $>=$	je večje in je večje ali enako
$<$ in $<=$	je manjše in je manjše ali enako
IN (...)	ustreza poljubnemu elementu množice
BETWEEN ... AND ...	je med podanima vrednostima
LIKE '..._%...'	ustreza vzorcu niza znakov
IS NULL	je neznana vrednost
NOT ...	negacija operatorja

Domena Trgovina

V podatkovni bazi *trgovina.sql* vodimo podatke manjše potujoče trgovine, ki deluje v slovenskih ruralnih področjih. Stranke, ki trgovino osebno obiščejo, lahko naročijo več enakih ali različnih izdelkov. Zaradi osebnega pristopa strankam o njih hranimo tudi nekaj osebnih podatkov, vključno s pridobljenim popustom zvestobe. Lastnik trgovine želi zagotoviti rigorozno sledljivost naročil, zato pri vsakem naročilu zabeležimo, kateri agent je določeni stranki prodal izdelke.



1. naloga Trgovina

stranka (id, ime, priimek, mesto, popust)

agent (id, ime, priimek, mesto, marza)

izdelek (id, ime, zaloga, cena)

narocilo (id, #stranka_id, #agent_id, #izdelek_id, datum, kolicina)

Z uporabo jezika SQL poiščite:

(a) vse podatke o vseh strankah

(b) identifikatorje in imena kranjskih agentov

(c) identifikatorje vsaj enkrat naročenih izdelkov

(d) pare imen strank in agentov, ki so sklenili naročilo

Poizvedovanje z jezikom SQL

1. naloga Trgovina

1. naloga Trgovina

stranka (id, ime, priimek, mesto, popust)
agent (id, ime, priimek, mesto, marza)
izdelek (id, ime, zaloga, cena)
narocilo (id, #stranka_id, #agent_id, #izdelek_id, datum, kolicina)

Z uporabo jezika SQL poiščite:

(a) vse podatke o vseh strankah

(b) identifikatorje in imena kranjskih agentov

(c) identifikatorje vsaj enkrat naročenih izdelkov

(d) pare imen strank in agentov, ki so sklenili naročilo

Rešitve nalog:

(a) SELECT * FROM stranka

(b) SELECT id, ime FROM agent WHERE mesto = 'Kranj'

(c) SELECT DISTINCT izdelek_id FROM narocilo

(d) SELECT DISTINCT s.ime, a.ime FROM stranka s, narocilo n, agent a WHERE s.id = n.stranka_id AND a.id = n.agent_id

2. naloga Trgovina

stranka (id, ime, priimek, mesto, popust)

agent (id, ime, priimek, mesto, marza)

izdelek (id, ime, zaloga, cena)

narocilo (id, #stranka_id, #agent_id, #izdelek_id, datum, kolicina)

Z uporabo jezika SQL poiščite:

(a) dobiček vsakega naročila po spodnji formuli

$$0,4 \times \text{količina} \times \text{cena kosa} - \frac{\text{popust} + \text{marža}}{100} \times \text{količina} \times \text{cena kosa}$$

(b) identifikatorje parov strank s sedežem v istem mestu

(c) identifikatorje izdelkov, ki sta jih kupili vsaj dve stranki

---

(d) stranke, ki so kupile izdelek, ki ga je prodal tudi agent 6

Poizvedovanje z jezikom SQL

2. naloga Trgovina

2. naloga Trgovina

stranka (id, ime, priimek, mesto, popust)
agent (id, ime, priimek, mesto, marza)
izdelek (id, ime, zaloga, cena)
narocilo (id, #stranka_id, #agent_id, #izdelek_id, datum, kolicina)

Z uporabo jezika SQL poiščite:

(a) dobiček vsakega naročila po spodnji formuli
$0,4 \times \text{količina} \times \text{cena kosa} - \frac{\text{popust} + \text{marža}}{100} \times \text{količina} \times \text{cena kosa}$

(b) identifikatorje parov strank s sedežem v istem mestu

(c) identifikatorje izdelkov, ki sta jih kupili vsaj dve stranki

(d) stranke, ki so kupile izdelek, ki ga je prodal tudi agent 6

Rešitve nalog:

(a) SELECT n.id, 0.4*n.kolicina*i.cena-(s.popust+a.marza)*0.01*(n.kolicina*i.cena) AS dobicek FROM narocilo n, stranka s, agent a, izdelek i WHERE s.id = n.stranka_id AND a.id = n.agent_id AND i.id = n.izdelek_id

(b) SELECT s1.id, s2.id FROM stranka s1, stranka s2 WHERE s1.mesto = s2.mesto AND s1.id < s2.id

(c) SELECT DISTINCT n1.izdelek_id FROM narocilo n1, narocilo n2 WHERE n1.izdelek_id = n2.izdelek_id AND n1.stranka_id <> n2.stranka_id

(d) SELECT DISTINCT n1.stranka_id FROM narocilo n1, narocilo n2 WHERE n1.izdelek_id = n2.izdelek_id AND n2.agent_id = 6

1. domača naloga GSM

prodaja (operater, telefon)

kupuje (stranka, operater)

najraje (stranka, telefon)

Z uporabo jezika SQL poiščite:

(a) stranke, ki kupujejo pri Mobitelu

(b) operaterje pri katerih kupuje Petra

(c) telefone, ki jih lahko kupi Petra z upoštevanjem (b)

(d) operaterje, ki prodajajo Janezov najljubši telefon

(e) telefone, ki jih prodaja posamezni operater

Operater prodaja telefon
Mobitel prodaja telefone znamke *Apple*
Simobil prodaja telefone znamke *HTC*
...

Poizvedovanje z jezikom SQL

1. domača naloga GSM

1. domača naloga GSM

prodaja (operater, telefon)
kupuje (stranka, operater)
najraje (stranka, telefon)

Z uporabo jezika SQL poiščite:
(a) stranke, ki kupujejo pri Mobitelu
(b) operaterje pri katerih kupuje Petra
(c) telefone, ki jih lahko kupi Petra z upoštevanjem (b)
(d) operaterje, ki prodajajo Janezov najljubši telefon
(e) telefone, ki jih prodaja posamezni operater

Operater prodaja telefon
Mobitel prodaja telefone znamke *Apple*
Simobil prodaja telefone znamke *HTC*
...

Rešitve nalog:

(a) SELECT stranka FROM kupuje WHERE operater = ’Mobitel’

(b) SELECT operater FROM kupuje WHERE stranka = ’Petra’

(c) SELECT telefon FROM kupuje, prodaja WHERE stranka = ’Petra’ AND kupuje.operater = prodaja.operater

(d) SELECT operater FROM prodaja, najraje WHERE stranka = ’Janez’ AND prodaja.telefon = najraje.telefon

(e) SELECT CONCAT(operater, ’ prodaja telefone znamke ’, telefon) AS ’Operater prodaja telefon’ FROM prodaja

Operacije množic SELECT (MySQL 5.7)

UNION	vse vrstice dveh tabel (brez duplikatov)
UNION ALL	vse vrstice dveh tabel (z duplikati)
INTERSECT	le vrstice v obeh tabelah (ni podprta)
MINUS	vrstice le v prvi tabeli (ni podprta)

EXISTS (...)	neprazna množica ali tabela
... IN (...)	vsebovanost v množici ali tabeli
... $>$ **ALL (...)**	ujemanje z vsemi v množici ali tabeli
... $<$ **ANY (...)**	obstoj ujemanja v množici ali tabeli

SELECT * FROM T_1 **UNION SELECT * FROM** T_2

SELECT * FROM T_1 **WHERE** A **IN** (**SELECT** A **FROM** T_2)

SELECT * FROM T **WHERE** $A >=$ **ALL** (**SELECT** A **FROM** T)

SELECT * FROM T **WHERE** $A <$ **ANY** (**SELECT** A **FROM** T)

3. naloga Trgovina

stranka (id, ime, priimek, mesto, popust)

agent (id, ime, priimek, mesto, marza)

izdelek (id, ime, zaloga, cena)

narocilo (id, #stranka_id, #agent_id, #izdelek_id, datum, kolicina)

Z uporabo jezika SQL poiščite:

(a) imena strank, ki so naročile izdelek 2 (s stikom)

(b) imena strank, ki so naročile izdelek 2 (brez stikov)

(c) stranke, ki so že naročile pri agentu iz Kranja ali Kopra (b)

Poizvedovanje z jezikom SQL

3. naloga Trgovina

3. naloga Trgovina

stranka (id, ime, priimek, mesto, popust)
agent (id, ime, priimek, mesto, marza)
izdelek (id, ime, zaloga, cena)
narocilo (id, #stranka_id, #agent_id, #izdelek_id, datum, kolicina)

Z uporabo jezika SQL poiščite:

(a) imena strank, ki so naročile izdelek 2 (s stikom)

(b) imena strank, ki so naročile izdelek 2 (brez stikov)

(c) stranke, ki so že naročile pri agentu iz Kranja ali Kopra (b)

Rešitve nalog:

(a) SELECT DISTINCT s.ime FROM stranka s, narocilo n WHERE s.id = n.stranka_id AND n.izdelek_id = 2

(b) SELECT s.ime FROM stranka s WHERE 2 IN (SELECT n.izdelek_id FROM narocilo n WHERE n.stranka_id = s.id)

(c) SELECT * FROM stranka s WHERE s.id IN (SELECT n.stranka_id FROM narocilo n WHERE n.agent_id IN (SELECT id FROM agent WHERE mesto IN ('Kranj','Koper')))

4. naloga Trgovina

stranka (id, ime, priimek, mesto, popust)

agent (id, ime, priimek, mesto, marza)

izdelek (id, ime, zaloga, cena)

narocilo (id, #stranka_id, #agent_id, #izdelek_id, datum, kolicina)

Z uporabo jezika SQL poiščite:

(a) identifikatorje agentov z najnižjim odstotkom marže

(b) identifikatorje agentov, ki nimajo najvišjega odstotka marže

---

(c) stranke, ki so kupovale le izdelke dražje od 50 EUR

Poizvedovanje z jezikom SQL

4. naloga Trgovina

4. naloga Trgovina

stranka (id, ime, priimek, mesto, popust)
agent (id, ime, priimek, mesto, marza)
izdelek (id, ime, zaloga, cena)
narocilo (id, #stranka_id, #agent_id, #izdelek_id, datum, kolicina)

Z uporabo jezika SQL poiščite:

(a) identifikatorje agentov z najnižjim odstotkom marže

(b) identifikatorje agentov, ki nimajo najvišjega odstotka marže

(c) stranke, ki so kupovale le izdelke dražje od 50 EUR

Rešitve nalog:

(a) SELECT id FROM agent WHERE marza <= ALL (SELECT marza FROM agent)

(b) SELECT id FROM agent WHERE marza < ANY (SELECT marza FROM agent)

(c) SELECT s.id FROM stranka s WHERE 50 <= ALL (SELECT i.cena FROM narocilo n, izdelek i WHERE i.id = n.izdelek_id AND s.id = n.stranka_id) AND EXISTS (SELECT * FROM narocilo n WHERE s.id = n.stranka_id)

2. domača naloga GSM

prodaja (operater, telefon)

kupuje (stranka, operater)

najraje (stranka, telefon)

Z uporabo jezika SQL poiščite:

(a) stranke, ki kupujejo pri vseh operaterjih
(implementirajte deljenje z dvema NOT EXISTS)

Poizvedovanje z jezikom SQL

2. domača naloga GSM

2. domača naloga GSM

prodaja (operater, telefon)
kupuje (stranka, operater)
najraje (stranka, telefon)

Z uporabo jezika SQL poiščite:

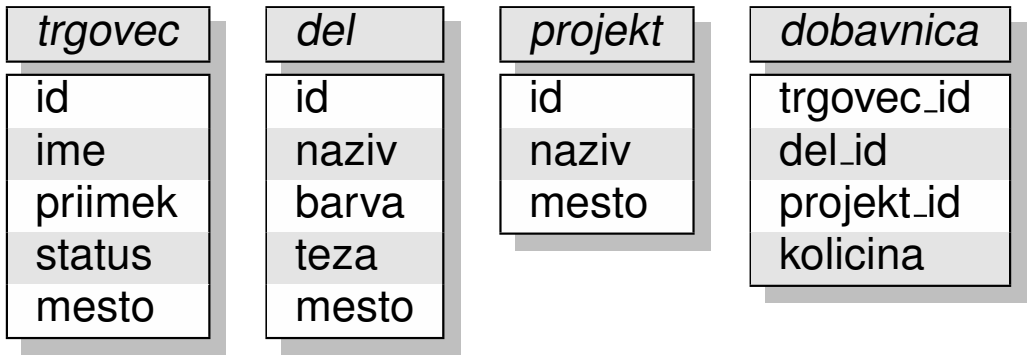
(a) stranke, ki kupujejo pri vseh operaterjih
(implementirajte deljenje z dvema NOT EXISTS)

Rešitve nalog:

(a) SELECT DISTINCT stranka FROM kupuje k1 WHERE NOT EXISTS (SELECT prodaja.operater FROM prodaja WHERE NOT EXISTS (SELECT k2.stranka FROM kupuje k2 WHERE k1.stranka = k2.stranka AND k2.operater = prodaja.operater))

Domena Dobavitelj

V podatkovni bazi *dobavitelj.sql* vodimo podatke o dobavljanju delov za namene izvajanja projektov. Dele zagotavljajo različni trgovci na podlagi izpolnjene dobavnice. Dobavnice pripadajo izbranim projektom, ki določene dele potrebujejo.



5. naloga Dobavitelj

trgovec	(id, ime, priimek, status, mesto)
del	(id, naziv, barva, teza, mesto)
projekt	(id, naziv, mesto)
dobavnica	(trgovec_id, del_id, projekt_id, kolicina)

Z uporabo jezika SQL poiščite:

(a) imena in priimke kranjskih trgovcev s statusom večjim od 20

(b) nazive rdečih delov, ki so težki vsaj 17kg

(c) imena in priimke trgovcev, ki so v okviru ene dobave prodali največ kosov poljubnega dela

---

Poizvedovanje z jezikom SQL

5. naloga Dobavitelj

5. naloga Dobavitelj

trgovec (id, ime, priimek, status, mesto)
del (id, naziv, barva, teza, mesto)
projekt (id, naziv, mesto)
dobavnica (trgovec_id, del_id, projekt_id, kolicina)

Z uporabo jezika SQL poiščite:

(a) imena in priimke kranjskih trgovcev s statusom večjim od 20

(b) nazive rdečih delov, ki so težki vsaj 17kg

(c) imena in priimke trgovcev, ki so v okviru ene dobave prodali največ kosov poljubnega dela

Rešitve nalog:

(a) SELECT ime, priimek FROM trgovec WHERE mesto = 'Kranj' AND status $>$ 20

(b) SELECT naziv FROM del WHERE teza $>=$ 17 AND barva = 'Rdeča'

(c) SELECT ime, priimek FROM trgovec t, dobavnica d WHERE t.id = d.trgovec_id AND d.kolicina $>=$ ALL (SELECT kolicina FROM dobavnica)

6. naloga Dobavitelj

trgovec	(id, ime, priimek, status, mesto)
del	(id, naziv, barva, teza, mesto)
projekt	(id, naziv, mesto)
dobavnica	(trgovec_id, del_id, projekt_id, kolicina)

Z uporabo jezika SQL poiščite:

(a) trgovce, ki so prodajali novomeščanskim projektom

(b) nazive še nikoli prodanih delov

(c) identifikatorje delov, ki sta jih prodala vsaj dva trgovca

---

(d) nazive delov, ki so jih kupovali kranjski in mariborski projekti

---

Poizvedovanje z jezikom SQL

6. naloga Dobavitelj

6. naloga Dobavitelj

trgovec (id, ime, priimek, status, mesto)
del (id, naziv, barva, teza, mesto)
projekt (id, naziv, mesto)
dobavnica (trgovec_id, del_id, projekt_id, kolicina)

Z uporabo jezika SQL poiščite:

(a) trgovce, ki so prodajali novomeščanskim projektom

(b) nazive še nikoli prodanih delov

(c) identifikatorje delov, ki sta jih prodala vsaj dva trgovca

(d) nazive delov, ki so jih kupovali kranjski in mariborski projekti

Rešitve nalog:

(a) SELECT DISTINCT t.ime, t.priimek FROM trgovec t, dobavnica d, projekt p WHERE t.id = d.trgovec_id AND d.projekt_id = p.id AND p.mesto = 'Novo mesto'

(b) SELECT naziv FROM del WHERE id NOT IN (SELECT del_id FROM dobavnica)

(c) SELECT DISTINCT d1.del_id FROM dobavnica d1, dobavnica d2 WHERE d1.del_id = d2.del_id AND d1.trgovec_id < d2.trgovec_id

(d) SELECT DISTINCT del.naziv FROM del, dobavnica d, projekt p WHERE del.id = d.del_id AND d.projekt_id = p.id AND p.mesto = 'Kranj' AND d.del_id IN (SELECT d2.del_id FROM dobavnica d2, projekt p2 WHERE d2.projekt_id = p2.id AND p2.mesto = 'Maribor')

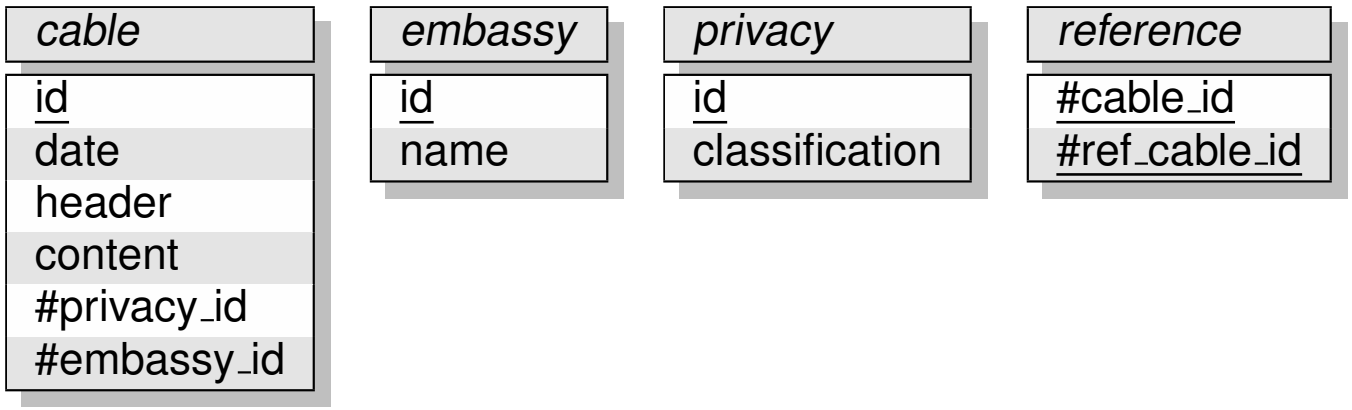
Agregacija z grupiranjem SELECT (MySQL 5.7)

COUNT	število vrednosti brez NULL
MIN	najmanjša vrednost
MAX	največja vrednost
SUM	vsota vseh vrednosti
STDDEV	deviacija vrednosti
VARIANCE	varianca vrednosti

SELECT $A_1, \ldots, A_k$, **SUM(**A_n**)** / **COUNT(**A_n**)**
FROM T_1 **NATURAL JOIN** T_2
WHERE $A_i < \ldots$
GROUP BY $A_1, \ldots, A_k$
HAVING COUNT(A_n**)** $> \ldots$
$\ldots$

Normalizirana domena WikiLeaks

V podatkovni bazi *wikileaks.sql* hranimo podatke o depešah, ki so si jih med seboj pošiljala veleposlaništva.



Zaradi omejitev velikosti datoteka ne vsebuje tekstovnih polj. Polno različico primerno za uvoz v podatkovno bazo PostgreSQL si lahko prenesete iz:

`http://file.wikileaks.org/torrent/cable_db_full.7z.torrent`

7. naloga WikiLeaks

cable (id, date, header, content, #privacy_id, #embassy_id)

embassy (id, name)

privacy (id, classification)

reference (#cable_id, #ref_cable_id)

Z uporabo jezika SQL poiščite:

(a) število zaupnih depeš ambasade v Ljubljani po letu 2000

(b) število depeš za vsak nivo zaupanja z imeni atributov *Nivo* in *Stevilo*

(c) dneve, ko je bilo odposlanih več kot 30 tajnih depeš

(d) dan, ko je bilo odposlanih največ tajnih depeš z ORDER BY in LIMIT

(e) dneve, ko je bilo odposlanih največ tajnih depeš *

Poizvedovanje z jezikom SQL

7. naloga WikiLeaks

7. naloga WikiLeaks

cable (id, date, header, content, #privacy_id, #embassy_id)
embassy (id, name)
privacy (id, classification)
reference (#cable_id, #ref_cable_id)

Z uporabo jezika SQL poiščite:

(a) število zaupnih depeš ambasade v Ljubljani po letu 2000

(b) število depeš za vsak nivo zaupanja z imeni atributov *Nivo* in *Stevilo*

(c) dneve, ko je bilo odposlanih več kot 30 tajnih depeš

(d) dan, ko je bilo odposlanih največ tajnih depeš z ORDER BY in LIMIT

(e) dneve, ko je bilo odposlanih največ tajnih depeš *

Rešitve nalog:

(a) SELECT COUNT(*) FROM cable c, privacy p, embassy e WHERE c.privacy_id = p.id AND c.embassy_id = e.id AND p.classification LIKE 'confidential%' AND e.name LIKE '%ljubljana%' AND c.date > '2000-01-01'

(b) SELECT p.classification AS Nivo, COUNT(c.id) AS Stevilo FROM cable c, privacy p WHERE c.privacy_id = p.id GROUP BY p.id, p.classification

(c) SELECT c.date, COUNT(c.id) FROM cable c, privacy p WHERE c.privacy_id = p.id AND p.classification LIKE 'secret%' GROUP BY c.date HAVING COUNT(c.id) > 30

(d) SELECT c.date FROM cable c, privacy p WHERE c.privacy_id = p.id AND p.classification LIKE 'secret%' GROUP BY c.date ORDER BY COUNT(c.id) DESC LIMIT 1

(e) SELECT c.date, COUNT(c.id) FROM cable c, privacy p WHERE c.privacy_id = p.id AND p.classification LIKE 'secret%' GROUP BY c.date HAVING COUNT(c.id) = (SELECT MAX(stevilo) FROM (SELECT COUNT(c.id) AS stevilo FROM cable c, privacy p WHERE c.privacy_id = p.id AND p.classification LIKE 'secret%' GROUP BY c.date) t)

8. naloga WikiLeaks

cable (id, date, header, content, #privacy_id, #embassy_id)

embassy (id, name)

privacy (id, classification)

reference (#cable_id, #ref_cable_id)

Z uporabo jezika SQL poiščite:

(a) dneve z največjim številom poslanih depeš, urejene padajoče, od 11. do 20. zadetka, in seznam veleposlaništev, ki so te depeše pošiljala

---

(b) depeše, ki se sklicujejo le na depeše ljubljanske ambasade

---

(c) veleposlaništvo, na čigar depeše se sklicuje največ depeš

(d) veleposlaništvo, na čigar depeše se sklicujejo depeše največ različnih veleposlaništev

---

Poizvedovanje z jezikom SQL

8. naloga WikiLeaks

8. naloga WikiLeaks

cable (id, date, header, content, #privacy_id, #embassy_id)
embassy (id, name)
privacy (id, classification)
reference (#cable_id, #ref_cable_id)

Z uporabo jezika SQL poiščite:

(a) dneve z največjim številom poslanih depeš, urejene padajoče, od 11. do 20. zadetka, in seznam veleposlaništev, ki so te depeše pošiljala

(b) depeše, ki se sklicujejo le na depeše ljubljanske ambasade

(c) veleposlaništvo, na čigar depeše se sklicuje največ depeš

(d) veleposlaništvo, na čigar depeše se sklicujejo depeše največ različnih veleposlaništev

Rešitve nalog:

(a) SELECT c.date, COUNT(c.id), GROUP_CONCAT(DISTINCT e.name SEPARATOR ',') FROM cable c, embassy e WHERE c.embassy_id = e.id GROUP BY c.date ORDER BY COUNT(c.id) DESC LIMIT 10 OFFSET 10

(b) SELECT r.cable_id FROM cable c, embassy e, reference r WHERE c.embassy_id = e.id AND e.name LIKE '%ljubljana%' AND r.ref_cable_id = c.id AND r.cable_id NOT IN (SELECT r.cable_id FROM cable c, embassy e, reference r WHERE c.embassy_id = e.id AND e.name NOT LIKE '%ljubljana%' AND r.ref_cable_id = c.id)

(c) SELECT c.embassy_id, e.name FROM cable c, reference r, embassy e WHERE c.id = r.ref_cable_id AND e.id = c.embassy_id GROUP BY c.embassy_id, e.name HAVING COUNT(r.cable_id) = (SELECT MAX(stevilo) FROM (SELECT COUNT(r.cable_id) AS stevilo FROM cable c, reference r WHERE c.id = r.ref_cable_id GROUP BY c.embassy_id) t)

(d) SELECT e.id, e.name FROM cable c1, cable c2, reference r, embassy e WHERE c2.id = r.ref_cable_id AND e.id = c2.embassy_id AND c1.id = r.cable_id GROUP BY e.id, e.name HAVING COUNT(DISTINCT c1.embassy_id) = (SELECT MAX(stevilo) FROM (SELECT COUNT(DISTINCT c1.embassy_id) AS stevilo FROM cable c1, cable c2, reference r WHERE c2.id = r.ref_cable_id AND c1.id = r.cable_id GROUP BY c2.embassy_id) t)

3. domača naloga WikiLeaks

cable	(id, date, header, content, #privacy_id, #embassy_id)
embassy	(id, name)
privacy	(id, classification)
reference	(#cable_id, #ref_cable_id)

Z uporabo jezika SQL poiščite:

(a) veleposlaništvo, ki se sklicuje na depeše največ različnih veleposlaništev z ORDER BY in LIMIT

(b) veleposlaništvo, ki se je že sklicevalo na depeše vseh veleposlaništev

---

(c) poiščite število veleposlaništev, ki so se sklicevala na svoje depeše, in število depeš, ki so se sklicevala same nase

---

Poizvedovanje z jezikom SQL

3. domača naloga WikiLeaks

3. domača naloga WikiLeaks

cable (id, date, header, content, #privacy_id, #embassy_id)
embassy (id, name)
privacy (id, classification)
reference (#cable_id, #ref_cable_id)

Z uporabo jezika SQL poiščite:

(a) veleposlaništvo, ki se sklicuje na depeše največ različnih veleposlaništev z ORDER BY in LIMIT

(b) veleposlaništvo, ki se je že sklicevalo na depeše vseh veleposlaništev

(c) poiščite število veleposlaništev, ki so se sklicevala na svoje depeše, in število depeš, ki so se sklicevala same nase

Rešitve nalog:

(a) SELECT e.id, e.name FROM cable c1, cable c2, embassy e, reference r WHERE c1.embassy_id = e.id AND r.cable_id = c1.id AND c2.id = r.ref_cable_id GROUP BY e.id, e.name ORDER BY COUNT(DISTINCT c2.embassy_id) DESC LIMIT 1

(b) SELECT e.id, e.name FROM cable c1, cable c2, embassy e, reference r WHERE c1.embassy_id = e.id AND r.cable_id = c1.id AND c2.id = r.ref_cable_id GROUP BY e.id, e.name HAVING COUNT(DISTINCT c2.embassy_id) = (SELECT COUNT(*) FROM embassy)

(c) SELECT (SELECT COUNT(DISTINCT c1.embassy_id) FROM cable c1, cable c2, reference r WHERE r.cable_id = c1.id AND c2.id = r.ref_cable_id AND c1.embassy_id = c2.embassy_id) AS Veleposlanistva, (SELECT COUNT(*) FROM reference r WHERE r.cable_id = r.ref_cable_id) AS Depese

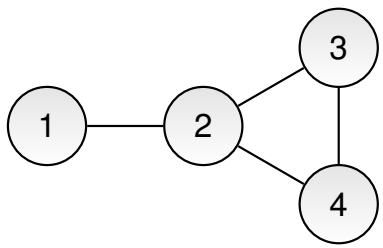
Normalizirana domena Facebook & Twitter

oseba(id, ime, rojstni_dan)

id	*ime*	*rojstni_dan*
1	Jill	9.3.1990
2	Jack	2.6.1950
3	Joe	1.8.1989
4	Jenn	7.1.2001

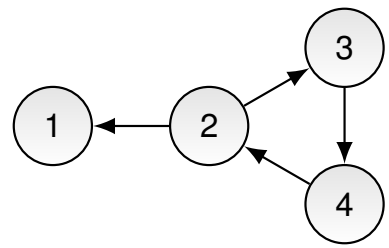
facebook(#oseba_id, #prijatelj_id)

#oseba_id	*#prijatelj_id*
1	2
2	1
2	3
3	2
3	4
...	...



twitter(#oseba_id, #sledilec_id)

#oseba_id	*#sledilec_id*
1	2
3	2
4	3
2	4



9. naloga Facebook & Twitter

oseba (id, ime, rojstni_dan)

facebook (#oseba_id, #prijatelj_id)

twitter (#oseba_id, #sledilec_id)

Z uporabo jezika SQL:

(a) ustvarite podatkovno bazo *social*

(b) ustvarite vse tabele s primarnimi in tujimi ključi

(c) napolnite tabelo *oseba* z ustreznimi vrednostmi

(d) napolnite tabelo *facebook* s prošnjami za prijateljstvo

Poizvedovanje z jezikom SQL

9. naloga Facebook & Twitter

9. naloga Facebook & Twitter

oseba (id, ime, rojstni_dan)
facebook (#oseba_id, #prijatelj_id)
twitter (#oseba_id, #sledilec_id)

Z uporabo jezika SQL:

(a) ustvarite podatkovno bazo *social*

(b) ustvarite vse tabele s primarnimi in tujimi ključi

(c) napolnite tabelo *oseba* z ustreznimi vrednostmi

(d) napolnite tabelo *facebook* s prošnjami za prijateljstvo

Rešitve nalog:

(a) CREATE DATABASE social DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
USE social;

(b) CREATE TABLE oseba (id INT NOT NULL AUTO_INCREMENT, ime VARCHAR(45) NOT NULL, rojstni_dan DATE NULL, PRIMARY KEY (id));
CREATE TABLE facebook(oseba_id INT NOT NULL, prijatelj_id INT NOT NULL, PRIMARY KEY (oseba_id, prijatelj_id), CONSTRAINT fb_fk1 FOREIGN KEY (oseba_id) REFERENCES oseba(id) ON DELETE RESTRICT ON UPDATE CASCADE, CONSTRAINT fb_fk2 FOREIGN KEY (prijatelj_id) REFERENCES oseba(id) ON DELETE RESTRICT ON UPDATE CASCADE);
CREATE TABLE twitter(oseba_id INT NOT NULL, sledilec_id INT NOT NULL, PRIMARY KEY (oseba_id, sledilec_id), CONSTRAINT tw_fk1 FOREIGN KEY (oseba_id) REFERENCES oseba(id) ON DELETE RESTRICT ON UPDATE CASCADE, CONSTRAINT tw_fk2 FOREIGN KEY (sledilec_id) REFERENCES oseba(id) ON DELETE RESTRICT ON UPDATE CASCADE);

(c) INSERT INTO oseba(ime,rojstni_dan) VALUES ('Jill', '1990-03-09'), ('Jack', '1950-06-02'), ('Joe', '1989-08-01'), ('Jenn', '2001-01-07');

(d) INSERT INTO facebook(oseba_id, prijatelj_id) VALUES (1,2), (2,3), (3,4), (2,4);

10. naloga Facebook & Twitter

oseba (id, ime, rojstni_dan)

facebook (#oseba_id, #prijatelj_id)

twitter (#oseba_id, #sledilec_id)

Z uporabo jezika SQL:

(a) izbrišite Facebook prošnjo za prijateljstvo med osebama 3 in 4

(b) dodajte uporabnika *misko*/*Kranjec.01* s polnim dostopom do baze *social*

(c) dodajte uporabnika *pavle*/*Sedmak.01* z bralnim dostopom do baze *social* in mu omogočite dostop le iz izbranega IP naslova

---

(d) preskusite delovanje ustvarjenih uporabniških računov

Poizvedovanje z jezikom SQL

10. naloga Facebook & Twitter

10. naloga Facebook & Twitter

oseba (id, ime, rojstni_dan)
facebook (#oseba_id, #prijatelj_id)
twitter (#oseba_id, #sledilec_id)

Z uporabo jezika SQL:

(a) izbrišite Facebook prošnjo za prijateljstvo med osebama 3 in 4

(b) dodajte uporabnika *misko*/*Kranjec.01* s polnim dostopom do baze *social*

(c) dodajte uporabnika *pavle*/*Sedmak.01* z bralnim dostopom do baze *social* in mu omogočite dostop le iz izbranega IP naslova

(d) preskusite delovanje ustvarjenih uporabniških računov

Rešitve nalog:

(a)
```
DELETE FROM facebook WHERE (oseba_id = 3 AND prijatelj_id = 4) OR (oseba_id = 4 AND prijatelj_id = 3);
```

(b)
```
CREATE USER 'misko'@'%' IDENTIFIED BY 'Kranjec.01';
GRANT ALL PRIVILEGES ON social.* TO 'misko'@'%';
FLUSH PRIVILEGES;
```

(c)
```
CREATE USER 'pavle'@'193.77.50.139' IDENTIFIED BY 'Sedmak.01';
GRANT SELECT ON social.* TO 'pavle'@'193.77.50.139';
FLUSH PRIVILEGES;
```

4. domača naloga Facebook & Twitter

oseba (id, ime, rojstni_dan)

facebook (#oseba_id, #prijatelj_id)

twitter (#oseba_id, #sledilec_id)

Z uporabo jezika SQL:

(a) napolnite tabelo *twitter* z ustreznimi vrednostmi

(b) izdelajte proceduro, ki poskrbi za obojestranska prijateljstva na Facebook

---

Poizvedovanje z jezikom SQL

4. domača naloga Facebook & Twitter

4. domača naloga Facebook & Twitter

oseba (id, ime, rojstni_dan)
facebook (#oseba_id, #prijatelj_id)
twitter (#oseba_id, #sledilec_id)

Z uporabo jezika SQL:
(a) napolnite tabelo *twitter* z ustreznimi vrednostmi

(b) izdelajte proceduro, ki poskrbi za obojestranska prijateljstva na Facebook

Rešitve nalog:

(a) INSERT INTO twitter(oseba_id, sledilec_id) VALUES (1,2), (3,2), (4,3), (2,4);

(b)

```
DELIMITER //
CREATE PROCEDURE fb_friends_repair()
BEGIN
  DECLARE done INT;
  DECLARE id1 INT;
  DECLARE id2 INT;

  DECLARE curs CURSOR FOR  SELECT oseba_id, prijatelj_id FROM facebook;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  OPEN curs;

  SET done = 0;
  REPEAT
    FETCH curs INTO id1, id2;
        INSERT IGNORE INTO facebook(oseba_id, prijatelj_id) VALUES (id2, id1);
  UNTIL done > 0 END REPEAT;

  CLOSE curs;
END;
//
DELIMITER ;

CALL fb_friends_repair;
```