Database Programming with SQL
kurs 2017 – database design and programming with sql
students slajdovi

10-1 Fundamentals of Subqueries

- U SQL podupiti (subqueries) omogućavaju nalazak informacije koja nam treba da bi našli neku drugu informaciju

Subquery Overview

- Šta ako treba napisati upit, ali programer nema sve informacije za njeno konstruisanje?
- Problem se rešava nestovanjem upita – smeštanjem jednog upita unutar drugog upita
- Unutrašnji upit se naziva podupit i on služi da pronađe informaciju koja je nepoznata
- Spoljni upit koristi tu informaciju za otkrivanje šta je potrebno znati
- Ovo se koristi kada je potrebno izabrati redove iz tabele sa uslovom koji zavisi od podataka u samoj tabeli
- Podupit je SELECT iskaz koji je umotan u izraz drugog SELECT iskaza
- Podupit se izvršava jednom pre glavnog upita
- Rezultat podupita se koristi od strane main ili spoljnog upita
- Podupiti se mogu smestiti u više SQL izraza, uključujući WHERE, HAVING, FROM izraze

- The subquery syntax is:

```
SELECT select_list
FROM table
WHERE expression operator
  (SELECT select_list
  FROM table);
```

The SELECT statement in parentheses is the inner query or 'subquery'. It executes first, before the outer query.

Guidelines for Using Aubqueries

- Podupit je u zagradi i uvek na desnoj strani uslova upoređivanja
- Spoljni i unutrašnji upiti mogu uzeti podatke iz različitih tabela
- Samo jedan ORDER BY izraz se može koristiti za SELECT iskaz; ako se koristi mora biti poslednji izraz u spoljnjem upitu
- Podupit ne može imati sopstveni ORDER BY izraz
- Jedini limit u broju podupita je veličina bafera koji koristi upit

Two Types of Subqueries

- Podupiti sa jednim redom; oni koriste single-row operatore (<, >, =, >=, <>, <=) i vraćaju samo jedan red iz unutrašnjeg upita
- Podupiti višestrukih redova; koriste multiple-row operatore (IN, ANY, ALL) i vraćaju više od jednog reda iz unutrašnjeg upita

Subquery Example

- Želimo naći imena zaposlenih koji su zaposleni posle Peter Vargas; prvo treba naći kada je Peter Vargas zaposlen a zatim selektovati one zaposlene čiji datum zaposlenja je posle njegovog

```
SELECT first_name, last_name,
  hire_date
FROM employees
WHERE hire_date >
 (SELECT hire_date
  FROM employees
  WHERE last_name = 'Vargas');
```

FIRST_NAME	LAST_NAME	HIRE_DATE
Eleni	Zlotkey	29-Jan-2000
Kimberely	Grant	24-May-1999
Kevin	Mourgos	16-Nov-1999
Diana	Lorentz	07-Feb-1999

Subquery and Null

- Ako podupit vraća null vrednost ili nijedan red, spoljni upit uzima rezultat podupita (null) i koristi ovaj rezultat u njegovom WHERE izrazu
- Spoljni upit neće onda vratiti nijedan red, pošto poređenje bilo koje vrednosti sa null uvek daje null

```
SELECT last_name
FROM employees
WHERE department_id =
  (SELECT department_id
   FROM employees
   WHERE last_name = 'Grant');
```

no data found

- Ko radi u istom sektoru kao Grant ? Grantov sektor je null, tako da podupit vraća null
- Spoljni nupit onda zamenjuje ovu vrednost u WHERE izrazu (WHERE department_id = NULL)
- Spoljni upit nevraća nikakav red, pošto poređenje bilo čega sa null vraća null

10-2 Single-Row Subqueries

Facts About Single-Row Subqueries

- Oni vraćaju jedan red i koriste single-row operatore upoređivanja (<, >, =, >=, <>, <=)
- Uvek staviti podupit u zagradu i smestiti podupit na desnu stranu uslova upoređivanja

Subqueries from Differetn Tables

- Ko radi u Marketing sektoru ?

```
SELECT last_name, job_id, department_id
FROM employees
WHERE department_id =
    (SELECT department_id
     FROM departments
     WHERE department_name = 'Marketing')
ORDER BY job_id;
```

Result of subquery

DEPARTMENT_ID
20

LAST_NAME	JOB_ID	DEPARTMENT_ID
Hartstein	MK_MAN	20
Fay	MK_REP	20

- Više od jednog podupita može vratiti informaciju spoljašnjem upitu

```
SELECT last_name, job_id, salary, department_id
FROM employees
WHERE job_id =
   (SELECT job_id
    FROM employees
    WHERE employee_id = 141)
AND department_id =
   (SELECT department_id
    FROM departments
    WHERE location_id = 1500);
```

Result of 1st subquery

JOB_ID
ST_CLERK

Result of 2nd subquery

DEPARTMENT_ID
50

LAST_NAME	JOB_ID	SALARY	DEPARTMENT_ID
Rajs	ST_CLERK	3500	50
Davies	ST_CLERK	3100	50
Matos	ST_CLERK	2600	50
Vargas	ST_CLERK	2500	50

- Prvi podupit vraća job_id od employee 141 (ST_CLERK). Drugi koristi tabelu sektora za pronalaženje department_id na location_id 1500 (50)
- Zatim spoljni upit vraća redove iz employees tabele koji odgovaraju za obe vrednosti

- U ovj instanci, pošto svaki sektor je smešten samo na jednoj lokaciji, bezbedno je koristiti sungle-row podupit. Ako sektor može biti na više lokacija mora se koristiti multiple-row podupit

Group Functions in Subqueries

- Funkcije grupe se mogu koristiti u podupitima
- Funkcija grupe bez GROUP BY izraza u podupitu vraća jedan red
- Pitanje je koji zaposleni zarađuje manje od srednje vrednosti plate ?
- Podupit prvo nalazi srednju platu za sve zaposlene, spoljnji upit vraća zaposlene sa platom manjom od srednje

```
SELECT last_name, salary
FROM employees
WHERE salary <
   (SELECT AVG(salary)
    FROM employees);
```

Result of subquery

AVG(SALARY)
8775

LAST_NAME	SALARY
Whalen	4400
Gietz	8300
Taylor	8600
Grant	7000
Mourgos	5800
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500
Ernst	6000
Lorentz	4200
Fay	6000

Subqueries in the HAVING Clause

- Podupiti mogu takođe biti smešteni u HAVING izraz
- HAVING izraz je sličan WHERE izrazu, osim što HAVING se koristi za ograničenje grupa i uvek uključuje funkciju grupe poput MIN, MAX ili AVG
- Iz tog razloga podupit skoro uvek takođe uključuje funkciju grupe
- Primer: Koji sektor ima najmanju platu koja je veća ode najmanje plate u sektoru 50 ?
- U ovom primeru, podupit bira i vraća najmanju platu u sektoru 50

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary) >
     (SELECT MIN(salary)
     FROM employees
     WHERE department_id = 50);
```

Result of subquery

MIN(SALARY)
2500

DEPARTMENT_ID	MIN(SALARY)
-	7000
90	17000
20	6000
110	8300
80	8600
10	4400
60	4200

- Vidi se da spoljni upit koristi ovu vrednost za biranje department ID i najnižu platu od svih sektora čija najmanja plata je veća od tog broja
- HAVING izraz je eliminisala one sektore čiji MIN plata je manja od sektora 50 MIN plate

10-3 Multiple-Row Subqueries

- Vraćanje nekoliko redova se izvodi korišćenjem multiple-row podupita i tri operatora upoređivanja IN, ANY, ALL

Query Comparison

- Čija plata je jednaka sa platom zaposlenog u sektoru 20 ?
- Ovaj primer vraća grešku pošto više od jednog zaposlenog postoji u sektoru 20 pa podupit vraća više redova
- Ovo se naziva multiple-row podupit:

LAST_NAME	DEPT_ID	SALARY
Hartstein	20	13000
Fay	20	6000

```
SELECT first_name, last_name
FROM employees
WHERE salary =
   (SELECT salary
    FROM employees
    WHERE department_id = 20);
```

ORA-01427: single-row subquery returns more than one row

- Problem je znak = u WHERE izrazu spoljnjeg upita
- Kako može jedna vrednost biti ista (ili različita) sa više od jedne vrednosti istovremeno?
- Iz istog razloga se ne može koristiti <, > ili <> u WHERE izrazu
- Ne može se porediti jedna vrednost sa više: Da li je 100 manje od (1300, 600) ?

IN, ANY, ALL

- Podupiti koji vraćaju više od jende vrednosti su multiple-row podupiti
- Pošto se ne mogu koristiti single-row operatori, treba koristiti multiple-row operatore IN, ANY, ALL, dok NOT operator se može koristiti sa bilo kojim od ovih operatora

IN

- Operator IN se koristi unutar spoljnjeg upita WHERE izraza za selektovanje samo onih redova koji su IN liste vrednosti vraćenih iz unutrašnjeg upita
- Npr, traže se svi zaposleni koji su zaposleni iste godine kao i zaposleni u sektoru 90

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) IN
    (SELECT EXTRACT(YEAR FROM hire_date)
     FROM employees
     WHERE department_id=90);
```

LAST_NAME	HIRE_DATE
King	17-Jun-1987
Kochhar	21-Sep-1989
De Haan	13-Jan-1993
Whalen	17-Sep-1987

- THE EXTRACT funkcija se može koristiti za ekstrakciju Year, MONTH ili DAY polja iz datum tipa podataka
- Unutrašnji upit će vratiti listu godina u kojima su zapošljavani zaposleni iz sektora 90
- Spoljnji upit će onda vratiti svakog zaposlenog koji je zaposlen iste godine sa godinama u listi unutrašnjeg upita

ANY

- Operator ANY se koristi kada želimo da WHERE izaraz spoljnog upita izabere redove koji odgovaraju kriterijumu (<, >, =...) od najmanje jedne vrednosti u podupitu seta rezultata
- Prikazani primer će vratiti svakog zaposlenog čija godina zaposlenja je manja od najmanje jedne godine zaposlenja zaposlenoh iz sektora 90
- The example shown will return any employee whose year hired is less than at least one year hired of employees in department 90.

Year Hired
1987
1989
1993

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) < ANY
  (SELECT EXTRACT(YEAR FROM hire_date)
   FROM employees
   WHERE department_id=90);
```

LAST_NAME	HIRE_DATE
King	17-Jun-1987
Kochhar	21-Sep-1989
Whalen	17-Sep-1987
Hunold	03-Jan-1990
Ernst	21-May-1991

ALL

- ALL operator se koristi kada želimo da WHERE izraz spoljnog upita bira redove koji odgovaraju kriterijumu (<, >, =...) od svih vrednosti u podupitu seta rezultata
- ALL operator upoređuje vrednosti sa svakom vrednosti vraćenom od strane unutrašnjeg upita
- Pošto nijedan zaposleni nije zaposlen pre 1987, neće se vratiti nijedan red

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) < ALL
  (SELECT EXTRACT(YEAR FROM hire_date)
   FROM employees
   WHERE department_id=90);
```

Year Hired
1987
1989
1993

no data found

- =ALL, da li može jedna vrednost da bude identična sa svakom vrednosti u setu vrednosti ? Zato se =ALL retko koristi

NULL vrednosti

- Pretpostavimo da jedna od vrednosti vraćena sa multiple-row podupitom je null, ali ostale nisu
- Ako IN ili ANY se koriste, spoljni upit će vratiti redove koji odgovaraju non-null vrednostima

the values
ple-row
other

```
SELECT last_name, employee_id
FROM employees
WHERE employee_id IN
    (SELECT manager_id
     FROM employees);
```

MANAGER_ID
-
100
100
101
101
205
100
...

Result of subquery

LAST_NAME	EMPLOYEE_ID
King	100
Kochhar	101
De Haan	102
Higgins	205
...	

- Ako se koristi ALL, spoljni upit ne vraća redove pošto ALL upoređuje red spoljnog upita sa svakom vrednosti vraćenom od podupita, uključujući null; a poređenje ničega sa null vraća null

```
SELECT last_name, employee_id
FROM employees
WHERE employee_id <= ALL
    (SELECT manager_id
     FROM employees);
```

no data found

GROUP BY and HAVING

- GROUP BY i HAVING izrazi se mogu koristiti u multiple-row podupitima
- Šta ako želimo naći sektore čija minimum plata je manja od plate bilo kojeg zaposlenog koji radi u sektoru 10 ili 20 ?
- Treba nam multiple-row podupit koji vraća plate zaposlenih u sektorima 10 i 20

- Spoljni upit će koristiti funkciju grupe (MIN) tako da treba grupisati spoljni upit po department_id

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary) < ANY
  (SELECT salary
   FROM employees
   WHERE department_id IN (10,20))
ORDER BY department_id;
```

DEPARTMENT_ID	MIN(SALARY)
10	4400
20	6000
50	2500
60	4200
80	8600
110	8300
-	7000

LAST_NAME	DEPT_ID	SALARY
Whalen	10	4400
Hartstein	20	13000
Fay	20	6000

Result of subquery

Multiple-Column Subqueries

- Podupiti mogu koristiti jednu ili više kolona
- Ako koriste više od jedne kolone, zovu se multiple-column podupiti
- Multiple-column podupiti mogu biti ili pair-wise upoređivanja ili non-pair-wise upoređivanja
- Sledeći primer pokazuje multiple-column pair-wise podupit sa podupitom označenim u crveno i rezultat u tabeli
- Upit lista zaposlene čiji menadžer i sektor su isti kao i menadžer i sektor zaposlenih 149 ili 174

```
SELECT employee_id, manager_id, department_id
FROM employees
WHERE(manager_id,department_id) IN
      (SELECT manager_id,department_id
       FROM   employees
       WHERE  employee_id IN (149,174))
AND employee_id NOT IN (149,174)
```

EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
176	149	80

- Prvo podupit vraća MANAGER_ID i DEPARTMENT_ID vrednosti za zaposlene sa EMPLOYEE_ID 149 ili 174
- Ove vrdnosti se porede sa MANAGER_ID kolonom i DEPARTMENT_ID kolonom svakog reda u EMPLOYEES tabeli. Ako se vrednosti podudare, red se prikaže
- Na ulazu, zapisi o zaposlenima sa EMPLOYEE_ID 149 ili 174 neće biti prikazani
- Non-pair-wise multiple-column podupit takođe koristi više od jedne kolone u podupitu, ali poredi ih jedan po jedan, tako da upoređivanja se izvode u različitim podupitima
- Treba pisati jedan podupit po koloni koje treba uporediti pri izvođenju non-pair-wise multiple-column podupita
- Primer pokazuje multiple-column non-pair-wise podupit sa podupitima označenim u crveno
- Ovaj upit lista zaposlene koji imaju istog manager_id i department_id kao i zaposleni 149 i 174

```
SELECT  employee_id,
        manager_id,
        department_id
FROM    employees
WHERE  manager_id IN
       (SELECT  manager_id
        FROM    employees
        WHERE   employee_id IN
                  (149,174))
AND    department_id IN
        (SELECT  department_id
         FROM    employees
         WHERE   employee_id IN
                  (149,174))
AND  employee_id NOT IN(149,174);
```

Result of 1st subquery

MANAGER_ID
100
149

Result of 2nd subquery

DEPARTMENT_ID
80
80

EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
176	149	80

- Prvo, podupit vraća MANAGER_ID vrednosti za zaposlene sa EMPLOYEE_ID 149 ili 174. Slično tome, drugi podupit vraća DEPARTMENT_ID vrednosti za zaposlene sa EMPLOYEE_ID 149 ili 174
- Vraćene vrednosti se porede sa MANAGER_ID i DEPARTMENT_ID kolonom za svaki red u EMPLOYEES tabeli

EXISTS & NOT EXISTS in Subqueries

- EXISTS i njegova suprotnost NOT EXISTS su dva izraza koja mogu da se koriste pri testiranju za jednakost u podupitima
- EXISTS testira za TRUE ili identičan rezultat u podupitu
- Da bi se odgovorilo na pitanje “Koji zaposleni nisu menadžeri ?“ prvo treba pitati Ko jesu menadžeri ? a onda i Ko NOT EXISTS na listi menadžera ?
- U ovom primeru, podupit selektuje zaposlene koji su menadžeri
- Spoljni upit onda vraća redove iz tabele zaposlenih koji NOT EXISTS u podupitu

table that do NOT EXIST in the subquery.

```
SELECT last_name AS "Not a Manager"
FROM   employees emp
WHERE NOT EXISTS
 (SELECT *
  FROM employees mgr
  WHERE  mgr.manager_id = emp.employee_id);
```

Not a Manager
Whalen
Gietz
Abel
Taylor
Grant
Rajs
Davies
Matos
Vargas
Ernst
...

- Ako se isti upit izvršava sa NOT IN umesto sa NOT EXISTS, rezultat će biti drugačiji
- Rezultat ovoga upita pokazuje da nema zaposlenih koji nisu takođe i menadžeri, tako da svi zaposleni su menadžeri, što nije tačno

```
SELECT last_name AS "Not a Manager"
FROM   employees emp
WHERE emp.employee_id NOT IN
 (SELECT mgr.manager_id
  FROM employees mgr);
```

no data found

- Razlog za ovako čudan rezultat je zbog NULL vrednosti koja je vraćena od strane podupita
- Jedan od redova u tabeli employees nema menadžera i to čini da ceo rezultat bude netačan

- Podupiti mogu vratiti tri vrednosti: TRUE, FLASE, UNKNOWN
- NULL u podupit setu rezultata će vratiti UNKNOWN koje Oracle ne može da evaluira
- Voditi računa o NULL u podupitima pri korišćenju IN ili NOT IN
- Ako niste sigurni da li podupit će uključiti null vrednost eliminišite null korišćenjem IS NOT NULL u WHERE izrazu
- Npr, WHERE emp.manager_id IS NOT NULL ili koristiti NOT EXISTS

One Last Point About Subqueries

- Neki podupiti mogu vratiti jedan red ili više redova, u zavisnosti od vrednosti podataka u redovima
- Ako čak najmanja mogućnost postoji u vraćanju višestrukih redova, treba da se napiše multiple-row podupit
- Npr, ko ima isti job_id kao Ernest ?
- Ovaj single-row upit radi dobro pošto postoji samo jedan Ernest u tabeli
- Ali šta ako kasnije, biznis zaposli novog zaposlenog koj se zove Ernest Susan

```
SELECT first_name, last_name, job_id
FROM employees
WHERE job_id =
   (SELECT job_id
    FROM employees
    WHERE last_name = 'Ernst');
```

FIRST_NAME	LAST_NAME	JOB_ID
Alexander	Hunold	IT_PROG
Bruce	Ernst	IT_PROG
Diana	Lorentz	IT_PROG

FIRST_NAME	LAST_NAME	JOB_ID
Bruce	Ernst	IT_PROG

Result of subquery

- Bolje bi bilo koristiti multiple-row podupit; takva sintaksa bi radila čak i ako podupit vrati jedan red; ako sumnjate, koristite multiple-row podupit

```
SELECT first_name, last_name, job_id
FROM employees
WHERE job_id IN
   (SELECT job_id
    FROM employees
    WHERE last_name = 'Ernst');
```

FIRST_NAME	LAST_NAME	JOB_ID
Alexander	Hunold	IT_PROG
Bruce	Ernst	IT_PROG
Diana	Lorentz	IT_PROG

FIRST_NAME	LAST_NAME	JOB_ID
Bruce	Ernst	IT_PROG

Result of subquery
There are 2 people with last name 'Ernst'

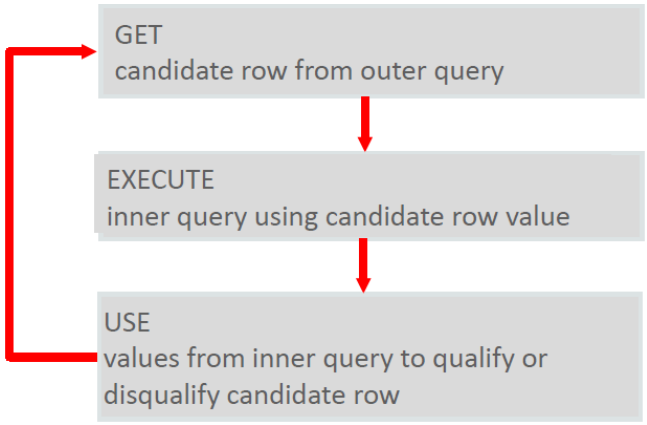
- Kada koristimo IN, ANY, ALL pri upoređivanju sa listom vrednosti, oni će raditi čak i ako postoji samo jedna vrednost na listi

10-4 Correlated Subqueries

- Ponekad je potrebno odgovoriti na više od jednog pitanja u jednoj rečenici
- U biznisu bi mogli biti pitani za produkciju izveštaja svih zaposlenih koji zarađuju više od srednje vrednosti u njihovim sektorima
- Pa prvo treba izračunati srednju vrednost po sektoru a zatim uporediti platu za svakog zaposlenog sa srednjom platom iz sektora tog zaposlenog

Correlated Subqueries

- Oracle server izvodi povezane podupite kada podupit referencira kolonu iz tabele referisane u roditeljskom iskazu



- Povezan podupit se evaluira jednom za svaki red procesiran od strane roditeljskog iskaza
- Roditeljski iskaz može biti SELECT, UPDATE ili DELETE iskaz
- Primer: Čija plata je veća od srednje plate njihovih sektora ?
- Povezani podupiti se koriste za row-by-row procesiranje

```
SELECT o.first_name,
       o.last_name,
       o.salary
FROM employees o
WHERE o.salary >
   (SELECT AVG(i.salary)
    FROM employees i
    WHERE i.department_id =
          o.department_id);
```

FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Shelley	Higgins	12000
Eleni	Zlotkey	10500
Ellen	Abel	11000
Kevin	Mourgos	5800
Alexander	Hunold	9000
Michael	Hartstein	13000

- Svaki podupit je izvršen jednom za svaki red spoljnog upita
- Sa normalnim podupitom, inner SELECT upit startuje prvo i izvršava se jednom, vraćajući vrednosti koje koristi spoljni upit
- Red dohvaćen iz tabele employees
- Srednja vrednost iz department_id od tog zaposlenog je izračunata sa podupitom
- Srednja vrednost plate u department_id tog zaposlenog je prikazana sa podupitom
- Spoljni upit poredi platu zaposlenog sa srednjom vrednosti za njihov sektor, i ako njihova plata je veća od srednje vrednosti za njihov sektor, red se vraća
- Sledeći red je onda dohvaćen i prosec je ponovljen
- Povezani podupit se izvodi samo jednom za svaki red ispitan od strane spoljnog upita
- Drugim rečima, unutrašnji upit je vođen spoljnim upitom
- Povezani podupit u primeru je ispisan crvenim
- Povezani podupiti se izvršavaju potpuno drugačije sa non-correlated podupitima, kao što su vođeni spoljnim upitima. Tako da je spoljni upit izvršen, prvi red vraćen i za taj red unutrašnji upit je izvršen

WITH Clause

- Ako se mora pisati veoma kompleksan upit sa udruživanjem i agregacijom korišćenim više puta, može se pisati drugačiji dao iskaza kao blok upita a onda koristiti te iste blok upite u SELECT iskazu

- Oracle dopušta pisanje imenovanih podupita u jednom iskazu, sve dok se startuje iskaz sa službenom reči WITH
- WITH izraz vraća rezultat jednog ili više blokova upita i smešta ove rezultate za korisnika upita
- WITH izraz poboljšava performanse i čini upit lakšim za čitanje
- Sintaksa:

```
WITH subquery-name AS (subquery),
     subquery-name AS (subquery)
     SELECT  column-list
     FROM     {table | subquery-name | view}
     WHERE  condition is true;
```

- Napisati upit za sledeći zahtev: pokazati listu prezimena zaposlenih koji nisu menadžeri
- Za izradu ovakvog upita, prvo nabaviti listu manager_id iz employee tabele, a zatim vratiti imena zaposlenih čiji id nije na listi menadžera
- Možemo napraviti imenovani upit korišćenjem WITH izraza za vraćanje manager_id iz employees tabele, a zatim spoljni upit će vratiti zaposlene koji se ne pojavljuju na listi

```
WITH managers AS
   (SELECT DISTINCT manager_id
    FROM employees
    WHERE manager_id IS NOT NULL)

SELECT last_name AS "Not a manager"
FROM employees
WHERE employee_id NOT IN
   (SELECT *
    FROM managers);
```

Not a manager
Whalen
Gietz
Abel
Taylor
Grant
Rajs
Davies
Vargas
Ernst
...