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

6-1 Cross Joins and Natural Joins

- Dosadašnji rad je bio limitiran time što su svi podaci bili smešteni u jednu tabelu
- Od modelovanja podataka si upoznat da odvajanje podataka po individualnim tabelama je moguće pošto se izvodi udruživanje tabela
- SQL ima join uslove koji pružaju informacije za upite iz odvojenih tabela i udružuje ih u jedan izveštaj

Join Commands

- Postoje dva seta komandi ili sintaksi koje se mogu koristiti za pravljenje veza između tabela u db: Oracle proprietary joins i ANSI/ISO SQL 99 compliant standard joins
- ANSI (American National Standards Institute) i promoviše standarde na osnovu dobrovoljnog koncezusa i čuva njihiv integritet

SQL

- Structured Query Language je procesno-informatički standardni jezik menadžment sistema relacionih baza podataka (RDBMS)
- Jezk je dizajniran 1970-tih u IBM i 1986 postao industrijski standard od ANSI
- Do sada su postojale tri ANSI standardizacije za SQL: ANSI-86, ANSI-92, ANSI-99

NATURAL JOIN

- SQL može da udruži polja od dve (ili više) tabela u rdb
- Natural join je na osnovama sve kolone u dve tabele koje imaju isto ime i bira vrste iz dve tabele koje imaju jednake vrednosti u svim jednakim kolonama

employees

EMPLOYEE_ID	LAST_NAME	JOB_ID
100	King	AD_PRES
101	Kochhar	AD_VP
...		
202	Fay	MK_REP
205	Higgins	AC_MGR
206	Gietz	AC_ACCOUNT

jobs

JOB_ID	JOB_TITLE
AD_PRES	President
AD_VP	Administration Vice President
AD_ASST	Administration Assistant
AC_MGR	Accounting Manager
AC_ACCOUNT	Public Accountant
SA_MAN	Sales Manager

- Tabela employees ima job_id kolonu; nacrtana je referenca na kolonu istog imena u tabeli jobs
- Kada se koristi NATURAL JOIN moguće je udružiti tabele bez eksplicitnog specifikovanja kolona u odgovarajućoj tabeli; ipak, imena i tip podataka obe kolone moraju biti isti

```
SELECT first_name, last_name, job_id, job_title
FROM employees NATURAL JOIN jobs
WHERE department_id > 80;
```

- Ovo udruživanje će vratiti kolone iz employees tabele i sa njima u relaciji job_title iz jobs tabele bazirano na zajedničkoj koloni job_id
- Izraz WHERE je dodat da bi se primenila dodatna restrikcija na jednu od tabela, limit redova na izlazu

FIRST_NAME	LAST_NAME	JOB_ID	JOB_TITLE
Steven	King	AD_PRES	President
Neena	Kochhar	AD_VP	Administration Vice President
Lex	De Haan	AD_VP	Administration Vice President
Shelley	Higgins	AC_MGR	Accounting Manager
William	Gietz	AC_ACCOUNT	Public Accountant

- Sledeći primer: i sektori (departments) tabela i locations tabela imaju kolonu location_id koja se korsiti za udruživanje dve tabele
- Primetiti da natural join kolona ne mora da se pojavi u SELECT izrazu

```
SELECT department_name, city
FROM departments NATURAL JOIN locations;
```

DEPARTMENT_NAME	CITY
Marketing	Toronto
Sales	Oxford
IT	Southlake
Shipping	South San Francisco
Administration	Seattle
Executive	Seattle
Accounting	Seattle
Contracting	Seattle

CROSS JOIN

- ANSI/ISO SQL: 1999 SQL CROSS JOIN udružuje svaki red u jednoj tabeli sa svakim redom u drugoj tabeli
- Dobijeni set rezultata predstavlja sve moguće kombinacije redova iz dvetabele i potencijalno može se dobiti ogromna tabela (20 redova x 100 redova = 2000 redova)
- Primer: tabela employees sadrži 20 redova a tabela departments ima 8 redova
- Upotrebom CROSS JOIN vraća se 160 redova

```
SELECT last_name, department_name
FROM employees CROSS JOIN departments;
```

- Vidi se da dobijeni redovi nisu logički u relaciji
- CROSS JOIN se koristi prilikom testiranja performansi velikih db

LAST_NAME	DEPARTMENT_NAME
Abel	Administration
Davies	Administration
De Haan	Administration
Ernst	Administration
Fay	Administration
Gietz	Administration
Grant	Administration
Hartstein	Administration
Higgins	Administration
Hunold	Administration

6-2 Join Clauses

- Smisao udruživanja je spajanje podataka, preko tabela, bez ponavljanja svih podataka iz svake od tabela

USING Clause

- U natural join, ako tabele imaju kolone sa istim imenima ali drugačijim tipovima podataka, korišćenje join izaziva grešku
- Da bi se ovakva greška sprečila, koristi se USING izraz; ovaj izraz specificira kolone koje bi se trebale koristiti za udruživanje
- Izraz USING se češće koristi umesto natural join čak iako kolone imaju iste tipove podataka kao i isto ime, pošto ona jasno ukazuje koja join kolona se koristi
- U primeru kolone referencirane u USING izrazu ne bi trebale imati qualifier (ime tabele ili alijas) bilo gde u SQL iskazu

```
SELECT first_name, last_name, department_id, department_name
FROM employees JOIN departments USING (department_id);
```

FIRST_NAME	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Jennifer	Whalen	10	Administration
Michael	Hartstein	20	Marketing
Pat	Fay	20	Marketing
...	...	...	...

- Ako kolona u USING izrazu ima qualifier, vraća se sledeća greška: ORA-25154: column part of USING clause cannot heve qualifier
- Izraz USING omogućava da se koristi WHERE za restrikciju redova iz jedne ili obe tabele

```
SELECT first_name, last_name, department_id, department_name
FROM employees JOIN departments USING (department_id)
WHERE last_name = 'Higgins';
```

FIRST_NAME	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Shelley	Higgins	110	Accounting

ON Clause

- Šta ako kolone koje treba da se udružuju imaju različita imena ili ako join koristi operatore upoređivanja (<, >, BETWEEN) ? Tada se ne može koristiti USING, pa umesto toga se koristi ON izraz
- To doprinosi da raznolikost join uslova bude specificirana
- ON izraz takođe omogućava korišćenje WHERE čime se vrši restrikcija redova iz jedne ili obe tabele
- Primer: ON izraz se koristi za udruživanje tabele employees sa tabelom jobs
- Udruživanje sa ON izrazom je neophodno kada zajedničke kolone imaju različita imena u dve tabele

```
SELECT last_name, job_title
FROM employees e JOIN jobs j
  ON (e.job_id = j.job_id);
```

LAST_NAME	JOB_TITLE
King	President
Kochhar	Administration Vice President
De Haan	Administration Vice President
Whalen	Administration Assistant
Higgins	Accounting Manager
Gietz	Public Accountant
Zlotkey	Sales Manager
Abel	Sales Representative
Taylor	Sales Representative
...	

- Pri korišćenju ON izraza na kolone sa istim imenom u obe tabele, treba dodati qualifier (ili ime tabele ili alijas) inače će se pojaviti greška: ORA-00918: column ambiguosly defined
- Prikazani primer koristi alijase tabele kao qualifier e.job_id = j.job_id, ali je takođe mogao biti napisan korišćenjem imena tabela (employees.job_id = jobs.job_id)

ON Clouse with WHERE Clause

- Ovo je isti upit sa WHERE izrazom koji ograničava izabrane redove

```
SELECT last_name, job_title
FROM employees e JOIN jobs j
  ON (e.job_id = j.job_id)
WHERE last_name LIKE 'H%';
```

LAST_NAME	JOB_TITLE
Higgins	Accounting Manager
Hunold	Programmer
Hartstein	Marketing Manager

- Pošto job_id kolona u employees i jobs tabelama imaju ista imena i tipove podataka, prethodni upit bi mogao da s enapiše sa NATURAL JOIN ili JOIN USING

ON Clause with non-equality operator

- Ponekad treba povratiti podatke iz tabele koji nemaju odgovarajuću kolonu u drugoj tabeli
- Npr, treba znati grade_level za platu svakog zaposlenog
- Tabela job_grades nema zajedničku kolonu sa tabelom employees
- Korišćenje ON izraza omogućava udruživanje dve tabele

job_grades table

GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

```
SELECT last_name, salary, grade_level, lowest_sal, highest_sal
FROM employees JOIN job_grades
  ON(salary BETWEEN lowest_sal AND highest_sal);
```

LAST_NAME	SALARY	GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
Vargas	2500	A	1000	2999
Matos	2600	A	1000	2999
Davies	3100	B	3000	5999
Rajs	3500	B	3000	5999
Lorentz	4200	B	3000	5999
Whalen	4400	B	3000	5999
Mourgos	5800	B	3000	5999
Fay	6000	C	6000	9999
...				

- Pri korišćenju JOIN ON izraza, mogu se koristiti operatori drugačiji od =

Joining Three Tables

- Za udruživanje tri ili više tabela koriste se i USING i ON
- Ako je potrebno napraviti izveštaje o radnicima, sektorima i gradovima gde su sektori smešteni treba udružiti tri tabele: employees, departments, locations

LAST_NAME	Department	CITY
Hartstein	Marketing	Toronto
Fay	Marketing	Toronto
Zlotkey	Sales	Oxford
Abel	Sales	Oxford
Taylor	Sales	Oxford
Hunold	IT	Southlake
Ernst	IT	Southlake
Lorentz	IT	Southlake
Mourgos	Shipping	South San Francisco
...		

```
SELECT last_name, department_name AS "Department", city
FROM employees JOIN departments USING (department_id)
  JOIN locations USING (location_id);
```

6-3 Inner versus Outer Joins

- Do sada, sva udruživanja se vratila podatke koji su odgovarali uslov udruživanja
- Ponekad je potrebno vratiti podatak koji odgovara join uslovu a ponekad podatak koji ne odgovara join uslovu
- Spoljna udruživanja (outer joins) u ANSI-99 SQL ovo omogućavaju

INNER And OUTER Joins

- U ANSI-99 SQL, udruživanje dve ili više tabele koje vraća samo uparene (matched) redove se naziva inner join
- Kada udruživanje vrati neuparene (unmatched) redove kao i uparene redove, to se naziva outer join
- Sintaksa za outer join koristi termine left, full, right; ovi nazivi su u korelaciji sa redosledom imena tabela u FROM izrazu kao dela SELECT iskaza
- NATURAL JOIN, JOIN ON, JOIN USING su tipovi inner joins

LEFT and RIGHT OUTER Joins

- Primetiti da u primeru prikazano ime tabele levo od reči "left outer join" je referisano kao "left table"

```
SELECT e.last_name, d.department_id, d.department_name
FROM employees e
LEFT OUTER JOIN departments d ON (e.department_id = d.department_id);
```

LAST_NAME	DEPT_ID	DEPT_NAME
Whalen	10	Administration
Fay	20	Marketing
…		
Zlotkey	80	Sales
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant	-	-

- Ovaj upit vraća prezimena svih zaposlenih, i onih koji su dodeljeni nekom od sektora i one koji nisu
- RIGHT OUTER JOIN će vratiti sve department IDs i imena sektora, koji imaju zaposlene dodeljene njima i one koji im nisu dodeljeni

```
SELECT e.last_name, d.department_id, d.department_name
FROM employees e
RIGHT OUTER JOIN departments d ON (e.department_id = d.department_id);
```

LAST_NAME	DEPT_ID	DEPT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
…		
King	90	Executive
Kochhar	90	Executive
De Haan	90	Executive
Higgins	110	Accounting
Gietz	110	Accounting
-	190	Contracting

FULL OUTER Join

- Moguće je napraviti join uslov za vraćanje svih uparenih redova i svih neuparenih redova iz obe tabele
- Korišćenje full outer join rešava ovaj problem
- Rezultat dobijen sa full outer join uključuje sve redove iz left outer join i sve redove iz right outer join kombinovano bez dupliciranja

```
SELECT e.last_name, d.department_id, d.department_name
FROM employees e
FULL OUTER JOIN departments d ON (e.department_id = d.department_id);
```

LAST_NAME	DEPT_ID	DEPT_NAME
King	90	Executive
Kochhar	90	Executive
...		
Taylor	80	Sales
Grant	-	-
Mourgos	50	Shipping
...		
Fay	20	Marketing
-	190	Contracting

Join Scenario

- Konstruisati join za prikaz liste zaposlenih, njihovih trenutnih job_id i bilo kojih prethodnih poslova koji su mogli imati
- Tabela job_history sadrži detalje od prethodnim poslovima zaposlenih

```
SELECT last_name, e.job_id AS "Job", jh.job_id AS "Old job", end_date
FROM employees e LEFT OUTER JOIN job_history jh
  ON(e.employee_id = jh.employee_id);
```

LAST_NAME	Job	Old job	END_DATE
King	AD_PRES	-	-
Kochhar	AD_VP	AC_MGR	15-Mar-1997
Kochhar	AD_VP	AC_ACCOUNT	27-Oct-1993
De Haan	AD_VP	IT_PROG	24-Jul-1998
Whalen	AD_ASST	AD_ASST	17-Jun-1993
Whalen	AD_ASST	AC_ACCOUNT	31-Dec-1998
Higgins	AC_MGR	-	-

6-4 Self-Joins and Hierarchical Queries

- U modelovanju podataka, ponekad je neophodno pokazati entitet sa relacijom na samog sebe
- Npr, zaposleni može biti takođe i menadžer; to je prikazano korišćenjem rekurzije ili “pigs ear“ relacije
- Posebna vrsta join koja se naziva self-join je neophodna za pristup ovim podacima
- Self-join se koristi za udruživanje tabela na samu sebe kao da su to dve tabele


```
SELECT worker.last_name || ' works for ' || manager.last_name
 AS "Works for"
FROM employees worker JOIN employees manager
ON (worker.manager_id = manager.employee_id);
```

SELF-JOIN

- Za udruživanje tabele sa samom sobom, tabeli se daju dva imena ili alijasi. To će učiniti da db “misli“ da postoje dve tabele

EMPLOYEES (worker)

employee_id	last_name	manager_id
100	King	
101	Kochar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

EMPLOYEES (manager)

employee_id	last_name
100	King
101	Kochar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

- Manager_id u worker tabeli je identičan sa employee_id u manager tabeli
- Izabrati alijase imena koji su u relaciji sa pridruženim podacima te tabele
- Primer:

```
SELECT worker.last_name, worker.manager_id, manager.last_name
  AS "Manager name"
FROM employees worker JOIN employees manager
  ON (worker.manager_id = manager.employee_id);
```

LAST_NAME	MANAGER_ID	Manager name
Kochhar	100	King
De Haan	100	King
Zlotkey	100	King
Mourgos	100	King
Hartstein	100	King
Whalen	101	Kochhar
Higgins	101	Kochhar
Hunold	102	De Haan
...	...	...

Hierarchical Queries

- U blisko povezani sa self-join su hierarchical queries
- Vezano sa prethodnim primerom: sa hijerarhijskim upitom može se videti i za koga menadžer radi itd
- Sa ovim tipom upita, može se napraviti Organization Chart koji pokazuje strukturu kompanije ili sektora
- Korišćenjem hijerarhijskih upita mogu se vratiti podaci bazirani na prirodnim hijerarhijskim relacijama između redova u tabeli
- Relaciona db ne smešta zapise na hijerarhijski način; ipak, gde postoji hijerarhijska relacija između redova jedne tabele procesi koji se nazivaju tree walking omogućavaju da se formira hijerarhija
- Hijerarhijski upit je metoda izveštavanja o granama drveta u posebnom redosledu
- Hijerarhijsko drvo se koristi u: ljudskoj genealogiji (drvo familije), livestock (razlozi odgajanja), upravljanje korporacijom (hijerarhija menadžmenta), proizvodnja (sastavljenje proizvodnje)

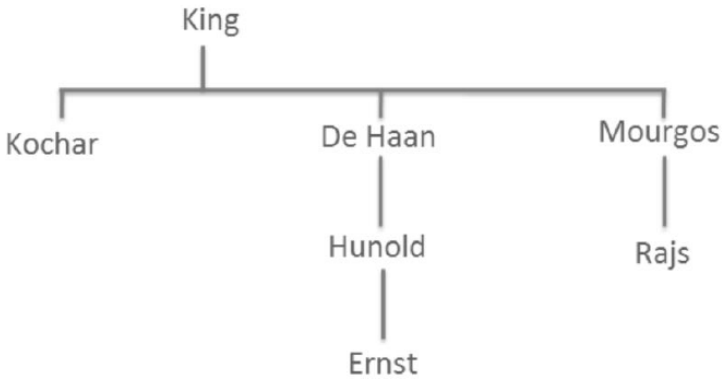
Hierarchical Queries Data

- Pogledati primer podataka ispod, iz tabele EMPLOYEES, i videti kako se može manuelno napraviti veza da bi se videlo ko radi za koga počevši od Steven King i prolazeći kroz drvo odatle

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMM_PCT	MGR_ID	DEPT_ID
100	Steven	King	SKING	515.123.4567	17-Jun-1987	AD_PRES	24000	(null)	(null)	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-Sep-1989	AD_VP	17000	(null)	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-Jan-1993	AD_VP	17000	(null)	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-Jan-1990	IT_PROG	9000	(null)	102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-May-1991	IT_PROG	6000	(null)	103	60
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-Nov-1999	ST_MAN	5800	(null)	100	50
141	Trenna	Rajs	TRAJS	650.121.8009	17-Oct-1995	ST_CLERK	3500	(null)	124	50

Hierarchical Queries Illustrated

- Organizaciona mapa iz koje možemo crtati podatke u tabeli EMPLOYEES će izgledati ovako:



Hierarchical Queries Keywords

- Hijerarhijski upiti imaju svoje nove službene reči: START WITH (identifikuje koji red da se koristi kao root za drvo), CONNECT BY PRIOR (objašnjava kako da se uradi unter-row joins), LEVEL (specificira koliko grana duboko će drvo da se razgrana; to je pseudo kolona koja vraća 1 za root drveta, 2 za sledeći niži nivo, 3 za sledeći itd)
- Primer:

```
SELECT employee_id, last_name, job_id, manager_id
FROM employees
START  WITH  employee_id = 100
CONNECT BY PRIOR employee_id = manager_id
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	MANAGER_ID
100	King	AD_PRES	-
101	Kochhar	AD_VP	100
200	Whalen	AD_ASST	101
205	Higgins	AC_MGR	101
206	Gietz	AC_ACCOUNT	205
102	De Haan	AD_VP	100
103	Hunold	IT_PROG	102
104	Ernst	IT_PROG	103

- Primer:

```
SELECT last_name ||' reports to ' || PRIOR last_name AS "Walk Top Down"
FROM employees
START WITH last_name = 'King'
CONNECT BY PRIOR employee_id = manager_id;
```

Walk Top Down
King reports to
Kochhar reports to King
Whalen reports to Kochhar
Higgins reports to Kochhar
Gietz reports to Higgins
De Haan reports to King
Hunold reports to De Haan
Ernst reports to Hunold

Hierarchical Queries Level Example

- LEVEL je pseudo kolona koja se koristi sa hijerahijskim upitima i broji koliko koraka je učinjeno od korena (root) drveta

```
SELECT LEVEL, last_name ||
    ' reports to ' ||
    PRIOR last_name
  AS "Walk Top Down"
FROM employees
START WITH last_name = 'King'
CONNECT BY PRIOR
  employee_id = manager_id;
```

LEVEL	Walk Top Down
1	King reports to
2	Kochhar reports to King
3	Whalen reports to Kochhar
3	Higgins reports to Kochhar
4	Gietz reports to Higgins
2	De Haan reports to King
3	Hunold reports to De Haan
4	Ernst reports to Hunold

Hierarchical Query Report

- Ako treba napraviti izveštaj koji prikazuje nivoe menadžmenta kompanije, počevši od najvišeg nivoa, uvlačenjem svakog od sledećih nivoa, onda će ovo biti lako uraditi korišćenjem LEVEL pseudo kolone i LPAD funkcije uvlačenjem zaposlenih bazirano na njihovom nivou

```
SELECT LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-2,'_')
  AS "Org Chart"
FROM employees
START WITH last_name = 'King'
CONNECT BY PRIOR employee_id = manager_id;
```

Hierarchical Query Output Levels

- Po rezultatu se vidi da je svaki red uvučen sa dve undeskor linije po nivou

```
SELECT LPAD(last_name, LENGTH(last_name)+
 (LEVEL*2)-2,'_') AS "Org_Chart"
FROM employees
START WITH last_name = 'King'
CONNECT BY PRIOR employee_id = manager_id;
```

Org_Chart
King
__Kochhar
____Whalen
____Higgins
______Gietz
__De Haan
____Hunold
______Ernst
______Lorentz
____Rajs
____Davies
____Matos
____Vargas
__Zlotkey
____Abel
____Taylor
____Grant
__Hartstein
____Fay

Bottom Up Hierarchical Query

- Sledeći primer pokazuje kako se kreira Bottom Up hijerarhijski upit korišćenjem službene reči PRIOR posle znaka jednakosti i korišćenjem 'Grant' u START WITH izrazu

```
SELECT LPAD(last_name, LENGTH(last_name) + (LEVEL*2)-2, '_') AS
    ORG_CHART
FROM employees
START WITH last_name = 'Grant'
CONNECT BY employee_id = PRIOR manager_id
```

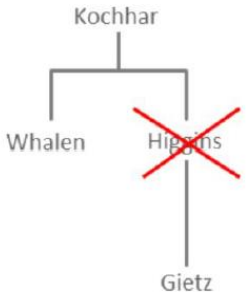
ORG_CHART
Grant
__Zlotkey
____King

Hirarchical Queries Pruning

- Pruning grana od drveta se može uraditi korišćenjem ili WHERE izraza ili CONNECT BY PRIOR izraza
- Ako WHERE izraz se koristi, isključuju se samo red imenovan u iskazu; ako CONNECT BY PRIOR izraz se koristi, cela grana se isključuje

- Npr, ako se želi isključiti jedan red iz rezultata, koristiće se WHERE izraz za izbacivanje tog reda; ipak, u rezultatu će izgledati kao da je Gietz radio direktno sa Kochhar iako nije

```
SELECT last_name
FROM employees
WHERE last_name != 'Higgins'
START WITH last_name = 'Kochhar'
CONNECT BY PRIOR  employee_id = manager_id;
```



- Ako pak se želi izbaciti jedan red i svi redovi ispod njega, treba učiniti izbacivanje kao0 deo iskaza CONNECT BY
- U sledećem primeru se isključuje Higgins, ali takođe i Gietz

```
SELECT last_name
FROM employees
START  WITH last_name = 'Kochhar'
CONNECT BY PRIOR employee_id = manager_id
AND last_name != 'Higgins';
```