Tutoriat 10

FORMA NORMALA 3; EXPRESIA ALGEBRICA A UNEI CERERI;

FORMA NORMALA 3

Pentru ca o relatia sa fie in forma normala 3 (FN3) aceasta trebuie:

- **Sa fie in FN2.**
- **Fiecare atribut care nu este cheie primara depinde de cheia primara, NUMAI de cheia primara si de nici-un alt atribut.**

In alte cuvinte, ca un tabel sa fie in FN3 toate coloanele care nu sunt chei primare trebuie sa depinda de acestea. Daca 2 coloane depind una de alta si niciuna nu este cheia primara atunci tabelul nu este in FN3.

Sa exemplificam acest lucru pe tabelul DOCTOR ce are atributele: id(PK), nume, specializare, salariu si numar_pacienti.

DOCTOR

id(PK)	nume	specializare	salariu	numar_pacienti
1	Andrei	Endocrinologie	8000	25
2	Ana	Ortopedie	7500	40
3	Ion	Endocrinologie	8000	32
4	Alex	Neurologie	10000	15
5	Maria	Ortopedie	7500	22
6	Dr. Phil	Psihologie	11000	~2191

In acest tabel putem observa ca coloanele „specializare” si „salariu” sunt dependete una de cealalta (fiecare specializare are un singur salariu) dar niciuna nu este cheia primara. Acest lucru incalca regula a doua pentru ca tabelul sa fie in FN3. Putem remedia aceasta problema intr-un mod simplu, spargand tabelul astfel:

DOCTOR

id(PK)	nume	specializare(FK)	numar_pacienti
1	Andrei	Endocrinologie	25
2	Ana	Ortopedie	40
3	Ion	Endocrinologie	32
4	Alex	Neurologie	15
5	Maria	Ortopedie	22
6	Dr. Phil	Psihologie	~2191

SALARIU

specializare(PK)	salariu
Endocrinologie	8000
Neurologie	7500
Ortopedie	10000
Psihologie	11000

Deoarece cele 2 coloane erau dependete una de cealalta si nici una nu era cheia primara am scos coloana „salariu” in afara tabelului DOCTOR, am facut un tabel special SALARIU pentru ea si am unit noul tabel cu DOCTOR prin coloana „specializare” ce a devenit cheie externa (aceasta unire este posibila tocmai pentru ca cele 2 coloane sunt dependente). In cazul de fata nu conteaza ce coloana scot din tabel („specializare” sau „salariu”) deoarece ambele aveau aceasi relatie cu cheia primara dar, daca s-ar fi observat ca o coloana este dependenta de cheia primara si una nu (lucru ce are loc in cazuri mai complexe) atunci coloana scoasa din tabel ar fi aceea care nu este dependenta de cheia primara.

EXPRESIA ALGEBRICA A UNEI CERERI

Expresia algebrica este o modalitate matematica de a scrie o cerere. Orice cerere se poate exprima sub forma uneia sau mai multor realatii si o relatie poate fi construita folosind relatiile anterioare si diversi operatori. O expresie algebrica se scrie in urmatorul mod:

Nume_relatie = Operator(relatie, parametru1, parametru2, …, paramentruN);

Unde „nume_relatie” este relatia rezultata de operator si de parametrii sai**. Printr-o relatie ne putem gandi si la un tabel ce are mai multe coloane(atribute).**

Operatorii pot fii:

Operatorul PROJECT

Primeste o relatie si anumite atribute din aceea relatie si extrage acele atribute (asemanator unui SELECT simplu, fara nici-o conditie)

Sintaxa:

Nume_relatie = PROJECT(relatie, atribut1, atribut2, …, atributN)

In continuare sa vedem un exemplu prin care vrem sa obtinem first_name, last_name si salary din tabelul employees.

SQL:

```
1. SELECT first_name, last_name, salary
2. FROM employees;
```

Expresia algebrica:

Rez = PROJECT(employees, first_name, last_name, salary)

Operatorul PROJECT este de obicei folosit cand dorim sa afisam sau sa utilizam doar anumite atribute dintr-o relatie, nu toate.

Operatorul SELECT

Primeste o relatie si o conditie si pastreaza doar elementele (liniile din tabel) ce indeplinesc acea conditie. Acest operator este util atunci cand dorim sa filtram datele (ca intr-un WHERE) .

Sintaxa:

nume_relatie = SELECT(relatie, conditie)

Sa exemplificam acest operator afisand toti angajatii din EMPLOYEES ce au un salariu mai mare de 7500.
SQL:

```
1. SELECT *
2. FROM employees
3. WHERE salary > 7500;
```

Expresie algebrica:

Rezultat = SELECT(employees, salary > 7500)

Un lucru important la SELECT este ca el intoarce **TOATE** atributele relatiei primite ca parametru. Daca dorim sa afisam numai anumite atribute acestea trebuie izolate folosind PROJECT.

Operatorul UNION

Asemanator operatorului UNION din SQL. Reuneste elementele a 2 relatii. Pe langa alte utilizari operatorul ne ajuta sa modelam cuvantul cheie OR din clauza WHERE intr-o expresie algebrica

Sintaxa:

Nume_relatie = UNION(relatie1, relatie2)

Sa exemplificam acest operator afisand toti angajatii ce au prenumele de „John” sau „Steve”

SQL:

```sql
SELECT *
FROM employees
WHERE first_name = 'John'
    OR first_name = 'Steve';
```

Expresie algebrica:

R1 = SELECT(employees, first_name = „John”)

R2 = SELECT(employees, first_name = „Steve”)

Rez = UNION(R1,R2)

Operatorul DIFFERENCE

Asemanator cu UNION, este echivalentul operatorului MINUS din SQL si face diferenta dintre 2 relatii: elementele ce se afla in prima relatie dar nu si in a doua (ordinea conteaza).

Sintaxa:

nume_relatie = DIFFERENCE(relatie1, relatie2);

Ca exemplu sa afisam toti angajatii cu employee_id par ce primesc comision

SQL:

```sql
SELECT *
FROM employees
WHERE MOD(employee_id , 2) = 0
MINUS
SELECT *
FROM employees
WHERE commission_pct IS NULL;
```

Expresie algebrica:

R1 = SELECT(employees, MOD(employee_id , 2) = 0)

R2 = SELECT(employees, commission_pct IS NULL)

Rez = DIFFERENCE(R1,R2)

Operatorul INTERSECT

Asemenator operatorului INTERSECT din SQL, acesta face intersectia dintre 2 relatii. Pastreaza ce elemente se afla atat in prima cat si in a doua relatie. Operatorul INTERSECT ajuta la modelarea intr-o expresie algebrica a cuvantului cheie AND din clauza WHERE.

Sintaxa:

Nume_relatie = INTERSECT(relatie1,relatie2)

Ca exemplu sa afisam toti angajatii ce lucreaza ca si „IT_PROG” in departamentul 60.

SQL:

```
1. SELECT *
2. FROM employees
3. WHERE job_id = 'IT_PROG'
4.   AND department_id = 60;
```

Expresie algebrica:

R1 = SELECT(employees, job_id = „IT_PROG”)

R2 = SELECT(employees, department_id = 60)

Rez = INTERSECT(R1,R2)

Operatorul PRODUCT

Face produsul cartezian al tuturor elementelor din prima relatie cu toate elementele din a doua (grupeaza fiecare element din prima relatie cu fiecare element din a doau relatie). Are acelasi efect ca un SELECT din 2 sau mai multe tabele fara join.

Sintaxa:

Nume_relatie = PRODUCT(relatie1, relatie2)

Ca si exemplu sa afisam pentru fiecare angajat toate departamentele in care poate lucra. Sa se afisaze pentru angajat first_name si last_name si pentru departamente doar numele acestora.

SQL:

```sql
SELECT e.first_name, e.last_name, d.department_name
FROM employees e, departments d;
```

Expresie algebrica:

R1 = PROJECT(employees, first_name, last_name)

R2 = PROJECT(departments, department_name)

Rez = PRODUCT(R1,R2)

Operatorul NATURAL JOIN

Asemanator cu join-ul clasic din SQL, uneste elementele a 2 relatii dupa coloanele comune daca au aceleasi valori (nu este obligatoriu ca coloanele sa aiba acelasi nume, se presupune ca se stiu coloanele dupa care se va face join).

Sintaxa:

Nume_relatie = JOIN(realtie1,relatie2)

Ca exemplu sa afisam datele fiecarui angajat si ale departamentului in care lucreaza.

SQL:

```sql
SELECT *
FROM employees e, departments d
WHERE e.department_id = d.department_id;
```

Expresie algebrica:

R = JOIN(employees, departments)

Operatorul θ-JOIN

Uneste elementele din 2 relatii dupa o conditie specificata.

Sintaxa:

Nume_relatie = JOIN(relatie1, relatie2, conditie)

Sa exemplifiam acest lucru afisand toti angajatii cu toate departamentele mai putin departamentul in care lucreza.

SQL:

```
1. SELECT *
2. FROM employees e, departments d
3. WHERE e.department_id <> d.department_id;
```

Expresie algebrica:

Rez = JOIN(employees, departments, employees.department_id<>departments.department_id)

Operatorul OUTERJOIN

Asemanator cu OUTER JOIN-ul din SQL acesta poate fi LEFT,RIGHT sau FULL in functie de ce dorim sa afisam.

Sintaxa:

Nume_relatie = OUTERJOIN LEFT/RIGHT/FULL(relatie1, relatie2)

Ca exemplu sa afisam: toti angajatii si departamentele lor, inclusiv angajatii fara departament; toate departamentele si angajatii ce lucreaza in ele, inclusiv departamentele fara angajati; toti angajatii(inclusiv cei fara departament) cu toate departamentele(inclusiv cele fara angajati).

SQL:

```
1. --RIGHT OUTER JOIN
2. SELECT *
3. FROM employees e, departments d
4. WHERE e.department_id = d.department_id(+);
5.
6. --LEFT OUTER JOIN
7. SELECT *
8. FROM employees e, departments d
9. WHERE e.department_id(+) = d.department_id;
10.
11. --FULL OUTER JOIN
12. SELECT *
13. FROM employees e, departments d
14. WHERE e.department_id(+) = d.department_id(+);
```

Expresie algebrica:

Rez1 = OUTERJOIN RIGHT(employees,departments)

Rez2 = OUTERJOIN LEFT(employees,departments)

Rez1 = OUTERJOIN FULL(employees,departments)

Operatorul DIVISION

Operator ce returneaza toate elementele primei relatii ale caror valori se afla in toate valorile atributelor celei de a doua relatii.

Sintaxa:

Nume_relatie = DIVISION(relatie1, relatie2)

Sa obtinenm codurile salariatilor atasati tuturor proiectelor pentru care s-a alocat un buget de 1000.

SQL: Tutoriat 5

Expresie algebrica:

R = PROJECT(atasat_la, cod_salariat, nr_proiect)

S1 = SELECT(proiect, buget = 1000)

S2 = PROJECT(S1, nr_proiect)

Rez = DIVISION(R,S2)

Rez o sa contina toate elementele din R al caror atribut „nr_proiect” se afla in **TOATE** elementele din S2.

IMPORTANT: La fel cum pentru o cerere exista mai multe rezolvari si expresiile algebrice se pot scrie in diferite feluri, in functie de ce operatori alegeti sa folositi.

Un alt lucru important pentru expresiile algebrice este ca mai multi operatori poti fi folositi, unul in interiorul celuilalt, pentru aceasi relatie daca dorim o scriere mai compacta.

Pentru a exemplifica acest lucru sa afisam numele si prenumele angajatilor ce il au ca manager pe „Steven King” si au fost angajati in anul 2000.

Varianta 1:

R1 = SELECT(employees, first_name = ’Steven’)

R2 = SELECT(employees, last_name = ’King’)

R3 = INTERSECT(R1,R2)

R4 = PROJECT(R3, employee_id)

S1 = JOIN(employees, R4, employees.manager_id = R4.employee_id)

S2 = SELECT(S1, TO_CHAR(hire_date,’YYYY’) = ’2000’)

Rez = PROJECT(S2, first_name, last_name)

Pentru aceasta rezolvare am folosit un singur operator in fiecare relatie. Prima data, in R1 si R2, iau toti angajatii cu prenumele „Steven”(R1) si numele „King”(R2). Dupa aceea intersectez cele 2 rezultate in R3, ce o sa ramana o sa fie angajatul Steven King.In final ii iau numai id-ul in R4. In S1 fac θ-JOIN pentru a afla toti angajatii ce il au ca manager pe Steven King, in S2 aflu toti angajatii subordonati lui Steven King ce au fost angajati in anul 2000 iar in Rez afisez numai numele si prenumele lor.

Varianta 2:

```
R = PROJECT(
            INTERSECT(
                        ( SELECT(employees, first_name = 'Steven') ),
                        ( SELECT(employees, last_name = 'King') )
                      ),
            employee_id)
S = JOIN(
          ( SELECT(employees, TO_CHAR(hire_date,'YYYY') = '2000') ),
           R,
           employees.manager_id = R.employee_id)
Rez = PROJECT(S, first_name, last_name)
```

Pentru aceasta varianta micsorez numarul de relatii si folosesc mai multi operatori pentru fiecare relatie. De asemenea scriu fiecare parametru al unei relatii pe un rand nou pentru a fi mai usor de citit (acest lucru este optional). In relatia R aflu id-ul lui Steven King unind relatiile R1, R2, R3 si R4 din exemplul anterior. In relatia S aflu toti angajatii subordonati lui Steven King din anul 2000 unind relatiile S1 si S2 si in Rez afisez numele lor.

Varianta 3:

Rez = PROJECT(JOIN((SELECT(employees, TO_CHAR(hire_date,'YYYY') = '2000')), INTERSECT((SELECT(employees, first_name = 'Steven')),(SELECT(employees, last_name = 'King')), employees.manager_id = INTERSECT((SELECT(employees, first_name = 'Steven')), (SELECT(employees, last_name = 'King')).employee_id), first_name, last_name)

Pentru aceasta varianta fac totul intr-o relatie (nu este recomandat).

Observatii

Pentru creearea unei expresii algebrice se recomanda o scriere asemanatoare celei din varianta 1. Aceasta ofera o claritate mai mare si este mai simplu de scris. Daca, din diverse motive, se doreste o scriere mai scurta se poate folosi o scriere asemanatoare celei din varianta 2 unde operatorii sunt grupati in mod logic(mai intai aflu id-ul lui Steven King apoi subalternii lui din anul 2000 si in final afisez numele lor). Nu recomand folosirea variantei 3 deoarece este greu de citit, de scris si de verificat.