Capitolul 1

MODELUL RELAȚIONAL

Modelul relațional ca şi orice alt model de date utilizat în proiectarea logică a bazelor de date eliberează utilizatorul de cunoaşterea detaliilor despre structura fizică şi metodele de acces la date. În afară de aceasta, el are două avantaje suplimentare: e simplu şi elegant. Simplitatea sa constă în structurile de date omogene în formă de relații tabelare. Iar eleganța modelului se explică prin temelia sa ştiințifică. El este riguros din punct de vedere matematic grație faptului că se sprijină pe bine puse la punct teoriile matematica relațiilor şi logica de ordinul unu.

Modelul relațional a fost primul exemplu de model de date formal şi a fost propus de E. Codd în 1970. Prin model datele utilizatorului sunt reprezentate şi manipulate în mod abstract. Modelul de asemenea presupune tehnici ce ajută administratorul de a detecta şi corecta posibilele probleme de proiectare ce pot apărea o dată cu pregătirea datelor pentru implementare într-un SGBD concret.

Orice model de date, conform unei sugestii a lui Codd, trebuie să se bazeze pe trei componente: structurile de date, constrângerile de integritate şi operatorii de manipulare a datelor.

- **Structurile de date.** Structurile sunt definite de un limbaj de definire a datelor (data definition language). Datele în modelul relațional sunt structurate în relații bidimensionale. Elementele principale ale structurii relaționale sunt relațiile, tuplurile, atributele, domeniile.
- **Constrângerile de integritate.** Prin integritatea datelor se subînțelege că datele rămân stabile, în siguranță şi corecte. Integritatea în modelul relațional este menținută de constrângeri interne care nu sunt cunoscute utilizatorului.
- **Manipularea datelor.** Relațiile pot fi manipulate utilizând un limbaj de manipulare a datelor (data manipulation language). În modelul relațional, limbajul foloseşte operatorii relaționali bazați pe conceptul algebrei relaționale. În afară de aceasta, există limbaje echivalente algebrei relaționale, cum ar fi calculul relațional orientat pe tuplu şi calculul relațional orientat pe domeniu.

1.1. Structura relațională a datelor

Unul din avantajele modelului relațional rezidă în omogenitatea lui. Toate datele sunt structurate în tabele, fiecare linie ale căror are acelaşi format. Linia într-un tabel reprezintă un obiect (sau o relație dintre obiecte) din lumea înconjurătoare.

1.1.1. Atribute şi domenii

În sistemele obişnuite de gestionare a fişierelor câmpul este cea mai mică unitate accesibilă de date. Se presupune că fiecare câmp poate conține un anumit tip de date

(integer, real, character, string etc.), pentru care se specifică numărul necesar de octeți de memorie. Câmpul, bineînțeles, are şi un nume. Făcând analogie, în modelul relațional fiecare coloană a unei linii dintr-un tabel corespunde noțiunii de câmp din fişiere.

Definiția 1.1. Fie U o mulțime nevidă de elemente $A_1,A_2,...,A_n$, numite *nume de atribute* sau simplu *atribute.* Mulțimea $U = \{A_1,A_2,...,A_n\}$ se numeşte *universul* unei baze de date relaționale sau *mulțime universală*.

Definiția 1.2. *Domeniul* unui atribut A_i din U, $1 \leq i \leq n$, notat cu $dom(A_i)$, este o mulțime finită de valori de acelaşi tip care le poate primi atributul A_i.

Domeniul este *simplu*, dacă elementele sale sunt atomice (adică nu pot fi descompuse din punctul de vedere al SGBD-ului). Atributul ce are un domeniu de valori simplu se numeşte *atribut atomic*. Domeniul unei submulțimi R a universului U , se notează dom(R), este uniunea tuturor domeniilor atributelor din R, adică $dom(R) = \cup_{Ai \in R} dom(A_i)$, unde $dom(\varnothing) = \varnothing$, dacă $R = \varnothing$.

Remarcă. Cu toate că elementele unui domeniu trebuie să fie de acelaşi tip, această restricție nu se extinde asupra elementelor din dom(R).

În modelul relațional fiecare tabel se spune că reprezintă o relație. Atributele sunt nişte identificatori pentru a diferenția şi marca coloanele tabelului. Deci, câmpul sau numele de coloană e şi atribut. Toate atributele ce apar într-un tabel trebuie să fie distincte şi să fie incluse în universul U. Atributele au un caracter global în baza de date: dacă un nume denotă două coloane în tabele distincte în aceeaşi bază de date, atunci el reprezintă acelaşi atribut.

Relația tabelară cu i linii şi j coloane are $i*j$ elemente. Fiecare element este o valoare dintr-un domeniu simplu. Cu toate că atributele în universul U trebuie să fie distincte, domeniile acestor atribute nu trebuie neapărat să fie disjuncte. De exemplu, managerul în acelaşi timp poate fi funcționar. Deci domeniile atributelor MANAGER şi FUNCȚIONAR nu sunt disjuncte, adică MANAGER şi FUNCȚIONAR sunt definite pe acelaşi domeniu (cu toate că atributele respective pot avea şi valori distincte).

Convenție. Mai departe vom utiliza următoarele notații. Universul $U=\{A_1,A_2,...,A_n\}$ şi orice submulțime a lui, $R=\{A_{i1},...,A_{ik}\}$, vor fi reprezentate ca string-uri, adică

$$U = A_1...A_n,$$
$$R = A_{i1}...A_{ik}.$$

Vom folosi o notație mai simplă $A_i \subseteq U$, în loc de $\{A_i\} \subseteq U$ pentru "A_i este o submulțime a mulțimii U". Reuniunea $Y \cup Z$ a două mulțimi Y şi Z va fi reprezentată de simbolul YZ, unde operația binară uniunea, "$\cup$", este omisă. Mulțimile Y şi Z pot fi şi mulțimi vide, fiindcă $\varnothing Z=Z$, $Y\varnothing=Y$ şi $\varnothing\varnothing=\varnothing$.

Cu litere majuscule de la începutul alfabetului latin vom nota atributele singulare, iar cu cele de la sfârşitul alfabetului latin - mulțimi de atribute.

1.1.2. Tupluri

În sistemele cu fişiere o mulțime de câmpuri ce e concepută ca o unitate de salvare şi căutare se numeşte înregistrare. Înregistrarea are un format specific şi depinde de tipurile de date ale câmpurilor. O linie dintr-o relație tabelară corespunde înregistrării din fişiere şi în teoria relațională se numeşte tuplu.

Definiția 1.3. Fie R o submulțime a universului U, $R \subseteq U$, unde $R \neq \emptyset$ şi fie dom(R) domeniul mulțimii R. *Tuplu* se numeşte o funcție, $t:R \rightarrow dom(R)$, din R în dom (R), adică

$$t = \{(A_{i1},a_1),..., (A_{ik}, a_k)\},$$

unde orice A_{ij}, $1 \leq j \leq k$, este un atribut în R şi un argument al lui t, iar orice a_j, $1 \leq j \leq k$, este o valoare în $dom(A_{ij})$.

Considerăm o restricție asupra tuplului t.

Definiția 1.4. Fie $X = B_1...B_m$ o submulțime proprie a mulțimii R, $X \subset R$, unde $X \neq \emptyset$. *X-valoare* a tuplului t, notată cu t[X], este $t[X] = \{(B_j, b_j)|b_j=t(B_j)=t(A_{ip}), 1 \leq j \leq m, p \in \{1,...,k\}\}$. Dacă $X=A_{ij}$, $j \in \{1,...,k\}$, atunci A_{ij}-valoarea tuplului t se mai numeşte A_{ij}–componentă a tuplului t.

Ultima definiție ne spune, că $t(A_{ij})=t[A_{ij}]=a_j$ pentru $a_j \in dom(A_j)$. Deci nu vom diferența simbolurile $t(A_{ij})$ şi $t[A_{ij}]$ pentru un atribut singular A_{ij} din U.

Pentru comoditate tuplul t şi X-valoarea tuplului t vor fi notate

$$t = <a_1...a_k|A_{i1}...A_{ik}> \text{ şi}$$
$$t[X] = <b_1...b_m|B_1...B_m>,$$

respectiv. Însă, dacă coloanele tabelului ce corespund mulțimilor R şi X sunt marcate cu atribute din R şi X, iar ordinea atributelor ce marchează corespund ordinii atributelor în R şi X, atunci notațiile tuplului t şi X-valorii tuplului t pot fi simplificate respectiv

$$t = < a_1...a_k > \text{şi}$$
$$t[X] = < b_1...b_m >.$$

Deci putem reprezenta printr-un string nu numai o mulțime de atribute, dar şi o mulțime de valori. Dar permutarea atributelor într-un tabel va trebui reflectată în tupluri, permutându-le componentele. Cu toate că string-urile ce reprezintă tuplurile inițial şi final vor fi diferite, vom considera că aceste tupluri sunt identice.

În tuplul $t = <a_1...a_k|A_{i1}...A_{ik}>$ distingem două componente – string-ul de atribute $A_{i1}...A_{ik}$ care este invariant în timp şi string-ul de valori $a_1...a_k$, care este foarte dinamic. Partea invariantă a tuplului vom numi-o schema tuplului (uneori se notează sch(t)). Îndată ce am definit schema tuplului, expresia "tuplul asupra R" devine clară şi este echivalentă expresiei "tuplul t cu schema R".

Pentru comoditate notațională, un tuplu cu numele t şi schema R se va nota uneori

$$t(R) = t(A_{i1})\ t(A_{i2})\ ...\ t(A_{ik}).$$

Deci putem concepe tuplul t(R) ca un *tuplu variabilă* asupra R şi fiecare componentă $t(A_{ij})$, $1 \leq j \leq k$, ca un *domeniu variabilă*. Dacă tuplul t(R) are o formă constantă, adică string-ul lui de valori este $<c_1...c_k>$ şi aceste valori sunt în dom(R), el se numeşte *tuplu constantă* asupra R.

1.1.3. Relații şi scheme

Definiția 1.5. Fie R o submulțime a universului U. *Relația* r asupra R este o mulțime finită de tupluri cu schema R. *Aritatea* relației r este egală cu cardinalitatea mulțimii R. *Cardinalitatea* relației r este numărul de tupluri în ea.

Definiția 1.6. Fie $R \subseteq U$ şi relația r asupra R. Mulțimea de atribute R se numeşte *schema* relației r (notată cu sch(r) = R).

Definiția 1.7. *Baza de date* relațională (sau simplu bază de date) este o mulțime finită de relații, db = $\{r_1,...,r_m\}$, unde r_i este o relație cu schema R_i, $1 \le i \le m$.

Definiția 1.8. Fie baza de date db = $\{r_1,...,r_m\}$. *Schema bazei de date* este mulțimea schemelor relațiilor ce formează baza de date, Db = $\{R_1,...,R_m\}$, unde R_i = sch(r_i).

Deci schema unei relații este o expresie a proprietăților comune şi invariante ale tuplurilor ce compun relația. Schema unei relații mai este cunoscută sub denumirea de *intensia* unei relații. Relația se mai numeşte *extensie*. Extensia reprezintă mulțimea tuplurilor care compun la un moment dat relația, mulțime care este variabilă în timp.

Din definițiile de mai sus putem conchide următoarele:

(1) Într-o relație nu există coloane cu nume duplicate, fiindcă atributele A_{ij}, $1 \le j \le k$, sunt elemente ale mulțimii R_i.
(2) Relația r_i nu are tupluri identice, fiindcă r_i este o mulțime de tupluri.
(3) Ordinea tuplurilor în r_i este nesemnificativă, fiindcă r_i este o mulțime.
(4) Ordinea coloanelor e nesemnificativă.
(5) Valorile atributelor în r_i sunt atomice fiindcă domeniile sunt simple.

Relațiile ce se stochează fizic şi formează baza de date se numesc *relații de bază*. Există, însă, şi situații în care extensia nu se memorează în baza de date. Este cazul aşa-numitelor *relații virtuale*, cunoscute şi sub numele de *relații derivate* sau *viziuni*. Relația virtuală nu este definită explicit ca relația de bază, prin mulțimea tuplurilor componente, ci implicit pe baza altor relații. Relațiile de bază sunt proiectate de administratorul bazei de date, în timp ce viziunile sunt definite de utilizatorii bazei de date.

Relațiile asupra unei mulțimi de atribute pot avea un nume, sau pot să nu aibă, dacă ele sunt identificate în mod unic de schemele sale. Numele relației, de obicei, se scrie cu minuscule, de exemplu, relația r.

studenți	NUME	NOTĂ_MED	FACULTATE	DECAN
	Vasilache	7.8	Cibernetică	Popovici

discipline	FACULTATE	DISCIPLINĂ

corp_didac	DISCIPLINA	PROFESOR

şarjă	DISCIPLINA	TIP	ORE

Fig.1.1. Baza de date Universitatea

Exemplul 1.1. Fie baza de date "Universitatea" constă din patru relații *studenți*, *discipline*, *corp_didactic* şi *şarjă* (vezi fig.1.1).

Primul tabel, reprezentând relația *studenți*, stochează numele, nota medie, facultatea şi decanul asociate fiecărui student. Relația are patru atribute: NUME, NOTĂ_MED, FACULTATE, DECAN. Relația *discipline* afişează disciplinele ce se predau la diverse facultăți. Ea are două atribute FACULTATE şi DISCIPLINĂ. Relația *corp_didac* specifică disciplinele predate de diferiți profesori. Ea are două atribute: DISCIPLINĂ şi PROFESOR. Relația *şarjă* descrie disciplinele cu formele sale de predare şi numărul de ore. Ea antrenează trei atribute: DISCIPLINĂ, TIP şi ORE.

Datele în fiecare relație sunt atomice şi sunt luate din domeniile (simple) atributelor corespunzătoare. FACULTATE în relațiile *studenți* şi *discipline* reprezintă acelaşi atribut. De asemenea atributul DISCIPINĂ figurează în trei relații: *discipline*, *corp_didac* şi *şarjă*. Cele opt atribute prezente în relațiile descrise constituie universul: U=NUME NOTĂ_MED FACULTATE DECAN DISCIPLINĂ PROFESOR TIP ORE. Aşadar, atributele oricărei relații formează o submulțime a mulțimii universale U.

Domeniul atributului NUME constă din eventualele nume de familii, dar trebuie să conțină numaidecât şi valori active, adică numele studenților ce actualmente îşi fac studiile la facultate. Domeniul atributului NOTĂ_MED conține numere pozitive. Dat fiind faptul că mulțimea de numere pozitive e infinită, fiecare NOTĂ_MED-valoare nu poate depăşi valoarea maximă 10. Deci dom(NOTĂ_MEDIE) poate fi menținut finit. Celelalte domenii se definesc similar. Nu e exclus faptul că un decan să fie student la o altă facultate. În cazul acesta domeniile active ale atributelor NUME şi DECAN nu vor fi disjuncte.

Tuplurile relației *studenți* sunt definite pe mulțimea de atribute R= NUME NOTA_MED FACULTATE DECAN. Ele sunt concepute ca tupluri constante-valori ale tuplului variabilă t(R) =t(NUME) t(NOTA_MED) t(FACULTATE) t(DECAN). De exemplu, tuplul constantă <Vasilache 7.8. Cibernetică Popovici> arată că Vasilache este student la facultatea Cibernetică a cărei decan este Popovici şi are nota medie 7.8. Considerăm X=NUME DECAN. Tuplul variabilă va fi t[X]=t(NUME) t(DECAN). Tuplul constantă definit pe schema X este derivat din tuplul <Vasilache 7.8. Cibernetică Popovici> al relației *studenți* şi este <Vasilache Popovoci>.

În baza de date "Universitatea", *studenți* este nume de relație. Schema relației *studenți* e NUME NOTĂ_MED FACULTATE DECAN.

Baza de date "Universitatea" constă din patru relații db={*studenți, discipline, corp_didactic, şarjă*}. Schema bazei de date este mulțimea schemelor celor patru relații Db={NUME NOTĂ_MED FACULTATE DECAN, FACULTATE DISCIPLINĂ, DISCIPLINĂ PROFESOR, DISCIPLINĂ TIP ORE}.

Relațiile s*tudenți*, *discipline*, *corp_didactic*, *şarjă* au aritatea 4, 2, 2 şi 3 respectiv. Ele formează relațiile de bază. Relația definită pe atributele X= NUME DECAN e o relație virtuală.

1.2. Constrângeri de integritate

1.2.1. Tipuri de constrângeri

Constrângerile de integritate, numite şi restricţii de integritate definesc cerinţele pe care trebuie să le satisfacă datele din baza de date pentru a putea fi considerate corecte, coerente în raport cu lumea reală pe care o reflectă.

Constrângerile sunt principalul mod de integrare a semanticii datelor în cadrul modelului relațional. Mecanismele de definire şi verificare ale acestor restricţii reprezintă instrumentele principale de control al semanticii datelor. În modelul relațional, constrângerile sunt studiate mai ales sub aspectul puterii lor de modelare şi al posibilității de verificare a respectării lor.

Constrângerile de integritate pot fi divizate în linii mari în două grupuri: *constrângeri de comportament* şi *dependențe* între date.

Constrângerile de comportament specifică caracteristicile independente ale unui atribut (sau domeniu). Ele exprimă semantica elementelor domeniilor. De exemplu, toate valorile atributului NOTĂ_MED trebuie să fie mai mare decât zero, dar nu poate depăşi zece. Sau nici o persoană de vârsta 25 ani nu poate avea o vechime în muncă de 37 ani. Deci, conform acestei restricţii, valorile atributului trebuie să se încadreze între anumite limite.

Al doilea tip de constrângeri specifică legătura dintre atribute (sau domenii). Aici putem identifica aşa-numita dependență de submulțime. Considerăm, de exemplu, relația *personal* definită pe mulțimea de atribute {ANGAJAT SALARIU DEPARTAMENT MANAGER}. În relația *personal*, un manager este în acelaşi timp un angajat, dar nu orice angajat este manager. Deci avem că dom(MANAGER) $\subseteq$ dom(ANGAJAT).

Dacă presupunem că angajatul are un singur salariu, se subordonează unui singur manager direct, lucrează deci într-un singur departament, atunci ANGAJAT-valorile determină în mod unic toate tuplurile în relația *personal*. Această constrângere este numită dependență funcțională. Dependențele funcționale vor fi studiate detaliat în capitolul 3. Alte tipuri de dependențe cum ar fi cele multivaloare şi de joncțiune vor fi studiate în capitolul 4.

1.2.2. Chei

Aici vom examina constrângerile legate de noțiunea de cheie a relațiilor. Întrucât relația reprezintă o mulțime de tupluri, iar o mulțime nu poate conține elementele duplicate, relația nu poate prezenta tupluri identice. Deci tuplurile sunt unice şi trebuie să existe posibilitatea identificării lor în cadrul unei relații. Identificarea unui tuplu fără a consulta toate componentele tuplului a dus la apariția noțiunii de cheie.

Definiția 1.9. Fie U mulțimea universală de atribute, $R \subseteq U$ şi $R \neq \emptyset$. Mulțimea K de atribute, unde $K \subseteq R$, se numeşte *cheie* pentru schema R (sau pentru relația r cu schema R), dacă ea posedă următoarele proprietăți:

(1) pentru orice două tupluri t_1 şi t_2 din r avem $t_1[K] \neq t_2[K]$;

(2) nici o submulțime K^1 proprie a lui K nu posedă proprietatea (1)

Proprietatea (1), numită restricția de unicitate a cheii, permite K-valorilor să identifice în mod unic toate tuplurile dintr-o relație. Însă respectarea proprietății de

unicitate poate fi complicată, dacă însăşi K conține o cheie K^1 şi $K \neq K^1$. În acest caz, cu toate că atributele din K sunt suficiente de a atinge scopul, unele din ele nu sunt necesare, deci pot fi eliminate din cheie fără a se afecta unicitatea. Dacă, însă, K este o submulțime proprie a unei chei, atunci utilizarea a astfel de K-valori pentru căutarea datelor va descoperi tupluri ce coincid pe toate valorile atributelor din K.

Proprietatea (2) ne asigură că o cheie K constituie numai acele atribute ce sunt necesare şi suficiente pentru a determina univoc pe celelalte. Cu alte cuvinte K-valorile întotdeauna asigură un grad exact de informație nici mai mult, nici mai puțin, pentru a găsi un tuplu unic într-o relație.

Definiția 1.10. Mulțimea de atribute ce posedă proprietatea (1) se numeşte *supercheie*.

Deci cheia este o supercheie minimală. Orice cheie e şi supercheie. Afirmația inversă nu e corectă.

Este evident că o mulțime vidă nu poate servi drept cheie a unei relații ce conține mai mult de un tuplu. Orice relație are cel puțin o cheie. La limită cheia este constituită fie dintr-un singur atribut, fie din totalitatea atributelor din schema relației respective.

Într-o relație pot exista mai multe chei. Se spune în acest caz că relația posedă mai multe *chei candidate*. În această situație administratorul bazei de date va stabili una din cheile candidate să servească în mod efectiv la identificarea unică a tuplurilor. Ea va primi numele de *cheie primară*. Primare se vor numi şi domeniile atributelor ce formează o cheie primară.

Definiția 1.11. Cheia primară a unei relații se numeşte *cheie simplă*, dacă este constituită dintr-un singur atribut, iar atunci când este formată din mai multe atribute este denumită *cheie compusă.*

Remarcă. Nu toate atributele unei chei compuse pot fi definite pe domenii primare.

Definiția 1.12. O *cheie externă* reprezintă un atribut (grup de atribute) dintr-o schemă R_i definit (definite) pe acelaşi (aceleaşi) domeniu (domenii) ca şi cheia primară a altei scheme R_j. Relația r_i se numeşte *relație care referă*, iar r_j poartă numele de *relație referită*.

Unele atribute pot avea aşa numitele valori nedefinite sau necunoscute notate cu ”null”. Însă sunt bine cunoscute constrângerile formulate prin următoarele reguli numite regulile de actualizare (inserare, modificare şi eliminare) a relațiilor.

(1) **Constrângerea entității:** cheia primară a unei relații de bază nu poate conține valori “null”;
(2) **Constrângerea referirii:** dacă atributul A al unei chei compuse a relației r_i este definit pe un domeniu primar, atunci trebuie să existe o relație de bază r_j cu o cheie primară B încât orice A-valoare din r_i să apară în calitate de B-valoare în r_j.

Constrângerea entității impune ca la înserarea unui tuplu, valoarea cheii să fie cunoscută, pentru a se putea verifica faptul că această valoare nu este deja încărcată (respectarea constrângerii de unicitate a cheii). Cu valori “null” cheia îşi pierde rolul de identificator de tuplu.

Constrângerea referențială impune ca într-o relație r_i, care referă o relație r_j, valorile cheii compuse să figureze printre valorile cheii primare din relația r_j pentru atributele compatibile.

Exemplul 1.2. Considerăm relațiile *studenți*, *discipline*, *corp_didac*., *şarjă*.

În relația *studenți* singura cheie candidat este NUME, deci NUME e şi cheia primară, iar dom(NUME) este domeniu primar pentru baza de date "Universitatea".

Considerăm relațiile *discipline*(FACULTATE DISCIPLINĂ) şi *corp_didac*(DISCIPLINĂ PROFESOR). Fiindcă la orice facultate se predă cel puțin o disciplină şi orice profesor predă cel puțin o disciplină, şi similar, orice disciplină se predă măcar la o facultate şi se predă cel puțin de un profesor, cheile primare ale acestor relații sunt compuse şi constau din toate atributele corespunzătoare fiecărei relații.

În relația *şarjă*(DISCIPLINĂ TIP ORE), orice disciplină poate fi de trei tipuri (prelegeri, practică, laborator) şi poate avea diferite ore de predare; unele discipline pot avea acelaşi tip şi acelaşi număr de ore. E puțin probabil ca cheia relației *şarjă* să fie simplă. Putem presupune că cheia ei este compusă.

În acest exemplu, domeniul dom(NUME) este primar. Cheile compuse ale relațiilor *discipline*, *corp_didac*, *şarjă* nu sunt definite pe acest domeniu primar.

Conform regulii (1) atributele celor patru chei nu pot avea valori "null". Dat fiind faptul că nici o cheie din cele trei compuse nu sunt definite pe domeniul primar dom(NUME), exemplul dat nu ne demonstrează regula (2).

Exemplul 1.3. Considerăm relațiile *studenți* şi *facultăți* din fig.1.2.

Presupunem că la o facultate îşi fac studiile mai mulți studenți şi un student poate studia la mai multe facultăți concomitent. Cheia primară a relației *studenți* este compusă şi constă din atributele NUME FACULTATE. Relația *facultăți* posedă două chei candidate: FACULTATE şi DECAN. Fie FACULTATE cheia primară. Atunci atributul FACULTATE al relației *studenți* este cheie externă. Conform regulii (2) toate valorile atributului FACULTATE al relației care referă trebuie să se conțină în relația referită.

studenți	NUME	FACULTATE	AN
	n_1	f_1	a_1
	n_2	f_1	a_2
	n_3	f_2	a_3

facultăți	FACULTATE	DECAN
	f_1	d_1
	f_2	d_2

Fig.1.2.

Spuneam mai sus că extensiile relațiilor se schimbă pe parcursul timpului. S-ar părea că pentru fiecare instanță a relației pot fi determinate cheile şi supercheile. Dar schemele relațiilor, adică intensiile, trebuie să fie invariante şi e de dorit ca cheile pe parcursul timpului să nu se schimbe. Cheile trebuie să rămână chei pentru orice eventuale extensii. Prin urmare determinarea cheii unei relații necesită cunoaşterea semanticii relației respective, nu numai celei din momentul în care se stabileşte cheia.

Convenție. Dacă relația posedă o singură cheie sau dorim să evidențiem numai cheia primară mai departe vom sublinia atributele ce formează această cheie. De exemplu, relația r cu schema ABCD şi cheia AC se scrie r(A B C D). În cazul că relația posedă mai multe chei, atunci le vom scrie explicit: relația r(ABCD) are două chei candidate K_1=AC, K_2=B.

1.3. Operații de actualizare

Regulile de actualizare a bazei de date fac parte din cele trei componente ale modelului relațional de date. Vom examina cele trei operații de actualizare a datelor: *inserarea* datelor, *ştergerea* datelor şi *modificarea* datelor.

Fie că în relația r(A_1 A_2 ...A_n) vrem să introducem date. Operația de inserție, a unui tuplu în relația r poate avea forma:

Add (r;<$a_1a_2...a_n$|A_1 A_2...A_n>).

În cazul că ordinea atributelor în relație e cunoscută, e acceptabilă o formă mai scurtă a operației:

Add (r;<$a_1a_2...a_n$>).

Scopul acestei operații constă în adăugarea unui tuplu într-o relație concretă. Rezultatul operației poate să eşueze din următoarele cauze:

(1) tuplul de inserție e definit pe o mulțime de atribute ce nu corespunde schemei relației;
(2) valorile componentelor tuplului nu sunt luate din domeniile corespunzătoare;
(3) în relație deja se găseşte un tuplu cu asemenea componente cheie.

În toate aceste cazuri operația Add păstrează relația r intactă.

Operația de ştergere a datelor se utilizează pentru eliminarea conținutului relațiilor. Pentru relația r de mai sus, operația de ştergere se reprezintă:

Del (r; <$a_1a_2...a_n$|A_1 A_2...A_n>).

În cazul când numele de atribute sunt sortate, poate fi utilizată următoarea notație scurtă:

Del (r; <$a_1a_2...a_n$>).

În realitate, o parte de date din operația de mai sus poate fi redundantă pentru determinarea tuplului destinat ştergerii. E suficientă definiția valorilor atributelor cheie. Dacă K=$B_1B_2...B_m$ este cheia relației r, atunci e utilă următoarea formă a operației Del:

Del (r; <$b_1b_2...b_m$|B_1 B_2...B_m>).

Rezultatul operației de ştergere a tuplurilor nu se lasă mult aşteptat. Tuplul e eliminat, dacă el este relație. În cazul că tuplul lipseşte - relația rămâne intactă. Nu se pune nici o restricție asupra eliminării ultimului tuplu în relație: relația vidă se admite.

Uneori, în loc de eliminarea unui tuplu din relație şi includerea unui alt tuplu e mai efectivă schimbarea unei părți a tuplului. Schimbarea se face cu operația de modificare. Dacă $C_1C_2...C_k \subseteq A_1A_2...A_n$, atunci operația de modificare poate avea forma:

Ch (r; <$a_1a_2...a_n$|A_1 A_2...A_n>; <$c_1c_2...c_k$| $C_1C_2...C_k$>).

Dacă mulțimea K=$B_1B_2...B_m$ este cheia relației r, atunci expresia de mai sus poate fi redusă la:

Ch (r; <$b_1b_2...b_n$|B_1 B_2...B_m>; <$c_1c_2...c_k$| $C_1C_2...C_k$>).

Operația de modificare este foarte utilă. Acelaşi rezultat poate fi obținut prin intermediul operațiilor de inserare şi ştergere. Prin urmare, toate eşecurile operațiilor inserare şi ştergere sunt specifice şi operației modificare.