Suport de curs BD

**CURS 1-2-3**

De ce folosim baze de date?

 Explozie informațională fără precedent, seturi de date complexe, de dimensiuni foarte mari

 O organizație trebuie să își gestioneze datele eficient, să poată obține informații corecte și în timp util pentru o anumită întrebare

 Nevoia de sisteme de gestiune a datelor puternice și flexibile –simplificarea gestiunii datelor și a extragerii de informații utile în timp optim

Componentele unei aplicații:

 Date (memorate în fișiere sau baze de date)

 Algoritm de gestiune

 Interfața cu utilizatorul

Date. Informatii

 Date: simboluri care trebuie să fie interpretate și corelate pentru a oferi informații

Exemplu:

 date: „Savin Bianca” și 0721459607

 informaţie: rezultat al căutarii unui numar de telefon in lista de contacte

 Datele sunt mult mai stabile în timp decât procesele care le gestionează

 Exemplu: nu au existat variaţii în structura datelor aplicațiilor bancare de zeci de ani; procedurile care gestionează datele variază de la an la an

 Datele sunt o resursă importantă a organizației care le gestionează

Exemplu

 Se doreşte realizarea unui sistem ce necesită stocarea de date specifice unei universitati: date despre studenţi, profesori,cursurile predate de aceştia şi cursurile la care participă studenţii

 FĂRĂ o bază de date =>

 stocăm datele în fişiere text

 scriem aplicatii (C, Java) care să implementeze diferite operatii asupra datelor in vederea obtinerea informatiilor

 Ex. cursurile oferite de Universitate, informatii despre facultati,etc.

 Dezavantajele stocarii in sistem de fisiere:

 redundanta si inconsistenta datelor

 dependența de date

 formate de fișiere incompatibile

 separarea și izolarea datelor

 probleme de securitate a datelor

Dezavantajele stocarii in sistem de fisiere

 Redundanta

 duplicarea datelor: aceleași date sunt stocate de mai multe ori

 risipa de spațiu de stocare => cresterea costului sistemului

 pierderea integrității datelor: pp.că detaliile angajatului sunt stocate atât în fisierul cu departamente și în cel cu birouri; angajatul își schimbă adresa de email; modificarea este realizata doar intr-unul din fisiere => date inconsistente!

 Dependența de date: programul depinde de date! Modificarea datelor => modificarea programului

 Formate de fișiere incompatibile: structura fisierelor depinde de limbajul de programare a aplicatiei

 Separarea și izolarea datelor: datele sunt stocate în fișiere separate=> este dificil pentru a le accesa

Baza de date (database)

 In sens larg: o bază de date este o colecţie de date corelate din punct de vedere logic, care reflectă un anumit aspect al lumii reale şi este destinat unui anumit grup de utilizatori

In sens mai restrans:

 O bază de date este o colecţie de date centralizate, creată şi menţinută computerizat, în scopul prelucrării datelor în contextul unui set de aplicaţii. Prelucrarea datelor se referă la operaţiile de introducere, ştergere, actualizare şi interogare a datelor.

 colecţii de fişe (documente pe hârtie) sau fişiere de date care conţin date, dar nu permit operaţii de interogare NU sunt considerate baze de date

 De ex., datele memorate în fişiere pe disc într-o aplicaţie de calcul tabelar (Microsoft Excel) sau

documentele memorate de un editor de texte (ca Microsoft Word) NU sunt considerate baze de

date.

 Domenii in care se folosesc baze de date: Bănci (conturi, carduri, etc.), Gestionarea datelor clienților unei companii de asigurări, Platforme de E-learning, E-commerce Websites (Amazon, Emag, etc), Rețele sociale (Facebook, Instagram, etc)

Whether you know it or not, you’re using a database every day (J. Widom)

Baza de date (database)

Proprietati ale unei baze de date (BD):

 Baza de date este o colecţie logică coerentă de date.

 Este destinată, construită şi populată de date despre un domeniu bine precizat

 Baza de date are un grup de utilizatori şi se adresează unui anumit grup de aplicaţii

 O bază de date reprezintă câteva aspecte ale lumii reale creând subuniversul propriu.

 Schimbările subuniversului sunt reflectate în baza de date.

 Un ansamblu de componente care asigură crearea, utilizarea şi întreţinerea uneia sau mai multor baze de date

Componentele unui sistem de baze de date sunt:

 Hardware: PC, Cloud, telefon

 Software: Sisteme de Gestiune a Bazelor de Date (SGBD)

 Interpretor de cereri de acces la baza de date

Sistem de baze de date

 Utilizatori

 Proiectantul bazei de date: stabilește structura (schema) bazei de date

 Programatorul bazei de date: implementează schema

 Administratorul bazei de date: face update-uri, creaza conturi, verifica securitatea, etc

 Programatorii de aplicații: implementează aplicații ce folosesc date le din BD si le modifica

 Date persistente

 Datele memorate într-o bază de date sunt date persistente ,date care rămân memorate pe suportul de stocare al sistemului de BD, independent de execuţia programelor de aplicaţii.

 Scop: separarea aplicaţiilor utilizatorului de baza de date fizică

 Propusă prin standardul ANSI/X3/SPARC (1975) conţine trei

niveluri funcţionale

Sistem de baze de date - Arhitectura interna

Nivelul intern (schema interna): descrie structura de stocare fizică a datelor în BD

Nivelul conceptual (schema conceptuala): structura întregii BD, ascunzând detaliile legate de stocarea fizică, concentrându-se asupra descrierii entităţilor, tipurilor de date, relaţiilor şi a restricţiilor asociate

Nivelul extern (vizual): o colecţie de scheme externe ce descriu BD prin prisma diferiţilor utilizatori. Fiecare grup de utilizatori descrie baza de date prin prisma propriilor interese

 Posibilitatea modificării sistemului bazei de date la orice nivel fără a influenţa nivelurile superioare

 Independenţa logică

 capacitatea schimbării schemei conceptuale fără a atrage după sine schimbări în schema externă sau în programele de aplicaţie

 expandarea bazei de date ca urmare a adăugării de noi tipuri de înregistrări sau a datelor insăşi (adăugare a unei noi coloane la un tabel)

 reducerea bazei de date ca urmare a stergerii înregistrărilor

 Independenţa fizică

 capacitatea de schimbare a schemei interne fără schimbarea schemei conceptuale sau externe

 Ex. schimbare a schemei interne: reorganizărea fizica a unor fişiere,

prin crearea de noi structuri de acces menite să asigure accesul

eficient la date.

Sistem de baze de date - Independenţa datelor

 Controlul centralizat al datelor – administrator BD

 Acces multi-utilizator controlat

 Viteză mare de regăsire a informatiilor şi actualizare a datelor

 Flexibilitate: posibilitatea modificării structurii bazei de date fără a fi necesară modificarea programelor de aplicaţie

 Redundanţă scăzută a datelor memorate: se obţine prin partajarea datelor între mai mulţi utilizatori şi aplicaţii.

 Posibilitatea introducerii standardelor privind modul de stocare a datelor, ceea ce permite interschimbarea datelor între organizaţii

 Menţinerea integrităţii datelor prin constrangeri de integritate care sa nu fie ‘ascunse’ in cod, ci declarate explicit

 politica de securitate (drepturi de acces diferenţiate în funcţie de rolul utilizatorilor)

 prin gestionarea tranzacţiilor şi prin refacerea datelor în caz de funcţionare defectuoasă a diferitelor componente hardware sau software.

Sistem de baze de date - Avantaje

 Independenţa datelor faţă de suportul hardware utilizat

 Protecţia şi securitatea datelor

 totalitatea mijloacelor, metodelor şi a mecanismelor destinate prevenirii distrugerii, modificării sau folosirii neautorizate a informaţiei protejate.

 Autentificare, criptarea datelor, firewall

 responsabil: administratorul bazei de date

 concepte de bază:

 Securitatea datelor: măsurile de protecţie împotriva distrugerii accidentale sau intenţionate, a modificării neautorizate sau a divulgării acestora

 Caracterul secret: se aplică la un individ sau organizaţie şi constă în dreptul acestora de a decide ce informaţii se pot folosi în comun şi în ce condiţii

 Confidenţialitatea: se aplică la date şi se referă la statutul acordat

 Integritatea: sensul datelor trebuie să nu difere faţă de cel înscris pe documentul de cerinte, impunând totodată ca datele să nu fie alterate accidental sau voit.

Sistem de Gestiune a Bazelor de Date (SGBD)

 Un sistem de programe general ce facilitează procesul definirii, construcţiei şi manipulării datelor pentru diverse aplicaţii

 Definire: specificarea tipurilor de date ce vor fi stocate în baza de date şi descrierea detaliată a fiecărui tip de dată.

 Construcţie: procesul stocării datelor pe mediul de stocare

 Manipulare: o serie de funcţii ce facilitează implementarea cererilor pentru găsirea datelor specificate,adăugarea de noi date ce reflectă modificarea contextului, generarea de rapoarte pe baza conţinutului bd

Sisteme comerciale de baze de date, oferite de companiile:

 Oracle

 IBM (with DB2)

 Microsoft (SQL Server)

 Sybase

 Microsoft Access

 Sisteme free (Unix/Linux, dar nu neaparat):

 Postgres

 Mysql

 SQLite

 Apache Derby sau H2 (scrise in Java).

 Sisteme NoSQL

 MongoDB, Redis, Solr, ElasticSearch

 Amazon SimpleDB

Model de date

 Definitie: un model al datelor este o unealtă formală pentru descrierea informaţiilor de interes în diverse aplicaţii.

 Datele si semantica lor

 Relaţiile dintre date

 Constrângerile asupra datelor

 Modele cunoscute

 Modelul Entităţi Asociaţii (Entity-Relationship data model, Chen, 1976)

 Modelul Relaţional al Datelor (Relational Data Model, E.F. Codd, 1970)

 Modele de date orientat obiect / Modelul de date obiect-relaţional

 Modelul Matematic Elementar al Datelor (Mancaș, 1985, 2013)

 Modele semistructurate (e.g. XML, JSON)

 Modele mai vechi

 Modelul reţea (C.W. Bachman, 1969)

 Modelul ierarhic

Model de date

 Modelul Entităţi Asociaţii (Entity-Relationship data model,Chen, 1976)

 Reprezentare grafică sugestivă

 Modelează universul problemei ca o mulțime de entități și asociații (relatii) între acestea

 Independent de orice considerații de memorare ori eficiență a implementării

 Modelul Relaţional al Datelor (Relational Data Model, E.F. Codd,1970)

 se bazează pe noţiunea de relaţie din matematică, care corespunde unei entităţi

 are o reprezentare uşor de înţeles şi de manipulat: datele sunt percepute de utilizatori ca tabele

 fiecare linie din tabel reprezintă o entitate şi este compusă din mulţimea valorilor atributelor entităţii respective

 fiecare atribut corespunde unei coloane a tabelului

 asocierea dintre tabele se realizează prin intermediul egalităţii valorilor unor atribute comune

 beneficiază de un limbaj de programare recunoscut şi acceptat, limbajul SQL (Structured Query Language)

 SQL2, standardul pentru limbajul SQL, 1992

 propune teste pentru a depista proiectari care nu sunt bune

 Teoria normalizării oferă reguli de proiectare/descompunere

Modele de date orientat obiect

 Se bazează pe limbaje de programare orientate obiect

 Caracteristicile importante ale modelului orientat obiect sunt: abstractizarea, moştenirea, încapsularea

Modelul de date obiect-relaţional

 Extinde modelul relaţional prin adăugarea de obiecte şi construcţii noi pentru tipurile de date

 Păstrează structurarea datelor în relaţii (reprezentate ca tabele)

 Adaugă posibilitatea definirii unor noi tipuri de date, pentru domeniile de valori ale atributelor.

 Tipurile de date definite de utilizator pot fi extinse prin mecanismul de moştenire şi pentru fiecare tip sau subtip se pot defini metode pe care le pot executa obiectele de acel tip

Model de date

Modelul ierarhic

 primul model folosit pentru dezvoltatea bazelor de date.

 o BD se reprezintă printr-o structură ierarhică de înregistrări de date (records) conectate prin legături (links)

 o schemă ierarhică este un arbore direcţionat, reprezentat pe mai multe niveluri

 nodurile sunt tipurile de înregistări

 arcele sunt tipurile de legături.

 fiecare nod (cu excepţia nodului rădăcină) are o singură legătură către un nod de pe un nivel superior (nodul părinte) şi fiecare nod (cu excepţianodurilor frunză) are una sau mai multe legături către noduri de pe nivelul imediat inferior (noduri fii).

Modelul retea

 foloseşte o structură de graf pentru definirea schemei conceptuale a BD

 nodurile grafului sunt tipuri de entităţi (înregistrări), iar muchiile grafului reprezintă în mod explicit asocierile (links) dintre tipurile de entităţi

Tendinţe noi în practică şi cercetare

 In-memory database (Main Memory DB)

 Raspuns rapid si mai sigur la interogari

 Datorita procesoarelor multi-core ce pot adresa spaţii de memorie mai mari si datorita RAM ieftin

 Pt aplicaţii unde e necesar raspunsul foarte rapid (ex tt echipamente de retele de telecomunicaţii)

 XML/JSON

 Baze de date relaţionale cu suport XML

Când este util să folosim baze de date?

 Atunci când nevoile aplicației noastre implică:

 Persistență

 Cantitate mare de date

 Date structurate

 Acces distribuit si concurrent la date

 Integritate

 Securitate

 Partajarea datelor cu alte aplicații

Când NU utilizăm baze de date?

 Investiția inițială e prea mare

 Prea mult effort

 Dezvoltăm o aplicație foarte simplă, bine-definită și care nu presupune modificări ulterioare

 Nu este necesar accesul mai multor utilizatori la date.

Alternativa: fișiere

Proiectarea bazelor de date

 Proiectarea unei baze de date presupune definirea

 structurii

 caracteristicilor

 conţinutului

Procesul de proiectare a bazei de date

Ciclul de viaţă al unui sistem informational

 Proiectarea BD reprezinta doar una dintre multiplele etape de dezvoltare a unui sistem informational (SI)

 Trebuie prezentata in contextul larg al ciclului de viata al unui SI

Def: Un sistem informaţional este un ansamblu de elemente implicate în procesul de colectare, transmisie si prelucrare de informaţie. În cadrul SI se regăsesc : informaţia vehiculată, documentele purtătoare de informaţii, personalul, mijloace de comunicare, sistemele de prelucrare (de regulă, automată) a informaţiei, etc

Etapele ciclului de viata ale unui SI:

 Studiu de fezabilitate

 definirea cât mai precisă a costurilor diferitelor soluţii posibile

 stabilirea priorităţilor în crearea componentelor sistemului

 Extragerea şi analiza cerinţelor

 definirea şi studiul proprietăţilor şi funcţionalităţii sistemului informaţional

 presupune interacţiunea cu clienţii pentru extragerea cerinţelor

 rezultatul este o descriere completă a datelor care intervin şi a operaţiilor care acţionează asupra datelor respective

 se stabilesc de asemenea necesităţile hardware şi software ale sistemului

 Proiectarea

 proiectarea bazei de date - se stabilesc structura şi organizarea datelor

 proiectarea operaţională - se definesc caracteristicile programelor

 Implementarea

 crearea sistemului informaţional în concordanţă cu structura şi caracteristicile precizate în etapa de proiectare

 se construieşte baza de date şi se scrie codul programelor

 Validare și testare.

 verificarea funcţionării şi a calităţii sistemului informaţional

 testele trebuie să conţină, pe cât posibil, toate condiţiile de operare posibile

 Operare

 SI devine operational și execută sarcinile pentru care a fost proiectat inițial.

Metodologia de proiectare

 o descompunere a întregii activitati de proiectare în etape succesive, independente una de celalalta;

 o serie de strategii care trebuie urmate în diferitele etape si unele criterii dintre care sa alegem în cazul în care exista optiuni;

 câteva modele de referinta pentru a descrie intrarile si iesirile din diferite etape

Se bazeaza pe un principiu de inginerie simplu, dar extrem de eficient: separa deciziile referitoare la „ce” este de reprezentat în baza de date, de cele referitoare la „cum” sa o faca.

CURS 4- Fazele proiectării bazelor de date

Proiectarea conceptuala

 Reprezentarea cerințelor unei aplicații în termenii corespunzatori unui model de date conceptual.

 Rezultat: schema conceptuală – permite descrierea organizării datelor la un nivel înalt de abstractizare fără a lua în considerare aspectele de implementare

Proiectarea logica

 Translarea schemei conceptuale într-un model de date disponibil pentru SGBD

 Rezultat: schema logica - permite reprezentarea datelor într-o formă independentă încă de detaliile fizice, deşi SGBD-ul folosit pentru implementare poate fi unul ce suportă acest model de date.

 Proiectantul trebuie să ia în considerare criterii de optimizare.

 Se utilizează frecvent tehnici formale pentru verificarea calităţii schemei logice.

 În cazul unui model de date relaţional, tehnica folosită este aceea a normalizării.

Proiectarea fizica

 schema logică se completează cu detalii de implementare fizică (organizarea fişierelor şi indecşi) puse

la dispoziţie de SGBD.

 Rezultat: schema fizică - depinde de SGBD

Cerinţe ale aplicaţiei care sunt utilizate în cele trei faze ale proiectării BD

 cerinţele de date - conţinutul bazei de date

 cerinţele operaţionale - utilizarea bazei de date de clienţi sau programatori

În proiectarea conceptuală

 cerinţele de date furnizează majoritatea informaţiilor

 cerinţele operaţionale se folosesc doar pentru a verifica dacă schema conceptuală este completă

În proiectarea logică

 schema conceptuală (furnizată ca intrare), rezumă cerinţele de date

 cerinţele operaţionale se folosesc pentru a obţine schema logică

În proiectarea fizică

 schema logică şi cerinţele operaţionale se folosesc pentru optimizarea performanţelor sistemului

 trebuie luate în considerare caracteristicile SGBD-ului utilizat

Proiectarea bazelor de date

 Rezultatul procesului de proiectare a bazei de date constă în:

 schema conceptuală - o reprezentare de nivel înalt a bazei de date

 schema logică - o descriere a conţinutului bazei de date

 este utilă pentru realizarea interogărilor şi modificărilor bazei de date.

 schema fizică

Modelul Entitate –Relatie (E-R)

 Entity Relationship Data Model (ERDM)

 Un model de date conceptual, introdus de P.S. Chen in 1976

 Descrie datele necesare unei aplicaţii într-o manieră uşor de înţeles şi independentă de SGBD

 Elementele de bază: conceptele de entitate şi cel de relaţie

 Fiecare element are o reprezentare grafica care ne permite sa definim schema (diagrama) conceptuala – DEA (MEA) –Diagrama (Modelul) Entitate-Asociatii

 Exista mai multe reprez.grafice (ex. Chen, Oracle (Crow‟s Foot), Barker, IDEF, Bachman, Mancas), dintre care se alege cea preferata de proiectant

Exemplu de model E-R

 O entitate (entity) este "orice poate fi identificat în mod distinctiv“ in cadrul subuniversului ce tb modelat.

 o entitate se refera la un aspect al realitatii obiective care poate fi deosebit de restul subuniversului si poate reprezenta un obiect fizic, o activitate, un concept, etc

 toate entitatile similare, care pot fi descrise prin aceleasi proprietati, apartin unui acelasi tip de entitate (entity type)

 colectia tuturor entitatilor de acelasi tip dintr-o baza de date constitue o multime de entitati (entities set).

 in general, în modelul E-R se foloseste aceeasi denumire atât pentru un entitate, tip de entitate cât si pentru multimea entitatilor de acel tip

 reprezentare: prin dreptunghi

 o instanta a unei entităţi este un obiect al clasei reprezentate de entitatea respectivă. Ex. oraşele Iaşi şi Bucureşti sunt instantei ale entităţii ORAS

 nu pot exista, în aceeaşi diagramă, doua entitati cu acelaşi nume, sau o aceeasi entitate cu nume diferite.

Entitati

Relatie (asociatie, legatura)

 reprezintă o legătura logica între două sau mai multe entităţi

 REZIDENŢĂ este un exemplu de relaţie care poate exista între entităţile ORAŞ şi ANGAJAT

 fiecare relatie are o denumire unica

 pentru numele relaţiilor se pot folosi verbe sau substantive; prin folosirea substantivelor în locul verbelor se evita sugerarea unei „direcţii”

 între aceleaşi entităţi pot exista mai multe relaţii

Relatii

 Dpdv matematic, relatiile sunt submulțimi ale produsului cartezian\

 o instanta a unei relaţii este un n-tuplu format din instante ale entităţilor, câte o instanta pentru fiecare entitate implicată

 este asigurat faptul ca niciun n-tuplu nu va apărea de două ori pentru instantele unei relaţii!

 sunt posibile relaţiile recursive care reprezintă relaţii între o entitate şi ea însăşi

 Ex. relaţia recursivă COLEG a entităţii ANGAJAT conectează perechi de oameni care lucrează împreună

 există relaţii care implică mai mult de două entităţi

 o instanta a relaţiei FURNIZEAZĂ descrie faptul că o anumită firmă furnizează un anumit produs unui departament

Atribute

 un atribut este o proprietate care descrie un anumit aspect al unei entitati sau al unei relatii

 Ex.: Nume, An sunt atribute pentru entitatea Curs, iar Dată, Notă sunt atribute pentru relaţia EXAMEN

 domeniile – sunt mulţimile de definiţie ale atributelor (multimile de valori admisibile)

 Ex.: domeniul pentru atributul Nume poate fi orice şir de caractere de lungime 20

(Dom(Nume)=CHAR(20)), iar domeniul pentru atributul Nota poate fi orice număr între 1 si 10 (Dom(Nota)={n∈ NN|1 ≤ nn ≤ 10}}

 domeniile nu sunt reprezentate grafic, ele fiind de obicei descrise în documentaţia asociată

 Dpdv matematic: atributele sunt functii ce asociaza entitatilorvalori de anumite tipuri

Gen: PERSOANE -> {„m‟, „f‟}

DataNasterii: PERSOANE -> [01/01/1900, 31/12/2012] Denumire: LOCALITĂȚI -> CHAR(64)

Atribute

Categorii de atribute:

 simple: o valoare atomica. Ex. Nume

 compuse: alcatuite din mai multe atribute simple. Ex. Adresa– compus din Strada, Numar si CodPostal

 multivaloare: au mai multe valori pentru aceeasi entitate. Ex. NumarTelefon poate avea mai multe valori pentru un student.

 derivate (calculate): deduse din alte atribute. Ex. Varsta – se calculeaza din DataNastere si data curenta.

Atributele derivate NU se introduc (in general) in model deoarece conduc la redundanta in baza de date!

Gradul unei relatii

 gradul unei relatii este dat de numarul de multimi de entitati asociate

 unare (recursive): aceeasi multime de entitati participa de mai multe ori intr-o asociere.

 Ex. Un sef este un angajat

 Ex. Un coleg este un angajat

 binare (de gradul 2, între 2 multimi de entitati)

 multiple (între k multimi de entitati, k> 2)

Cardinalitatea unei relatii

 este specificată pentru fiecare entitate participantă la relaţie

 descrie numarul maxim, respectiv minim, de entitati din cele doua multimi de entitati care participa la relatie

 reprezentare:

 cardinalitatea minimă este indicată în paranteze, iar cardinalitatea maximă se scrie fără paranteze. Ex. M(0)

 cardinalitatea minima se indica in stanga, urmata de doua puncte si cardinalitatea maxima. Ex. 0..M

 cardinalitatile se indica in paranteza, mai intai cea minima, apoi cea maxima. Ex. (0,M)

 în principiu este posibil să atribuim orice valoare întreagă mai mare ca 0 cardinalităţii unei relaţii, singura cerinţă fiind ca valoarea minimă a cardinalităţii să fie mai mică decât cea maximă

 în majoritatea cazurilor este suficientă utilizarea a trei valori: 0, 1 sau N (N este o valoare întreagă mai mare ca 1):

 cardinalitatea minimă 0 – participarea la relaţie este opţională

 cardinalitatea minimă 1 – participarea la relaţie este obligatorie

 cardinalitatea maximă 1 – fiecare entităte este asociată cel mult unei singure instante a relaţiei

 cardinalitatea maximă N – fiecare entitate este asociată unui număr arbitrar de instante ale relaţiei

Fie o relatie R între mulţimile de entităţi A şi B

 One-to-one:

 O entitate din A este asociată cel mult unei entităţi din B şi o entitate din B

este asociată cel mult unei entităţi din A.

 Dpdv matematic, asociațiile 1-to-1 sunt funcții

 One-to-Many:

 O entitate din A este asociată cu oricâte entităţi din B şi o entitate din B este asociată cel mult unei entităţi din A.

 Dpdv matematic, asociațiile 1-to-N sunt funcții

 Many-to-many

 O entitate din A este asociată cu oricâte entităţi din B ş i o entitate din B este asociată cu oricâte entităţi din A.

 Denumire alternativa: entitate asociativa

Tipuri de relatii

Observatii

 este rar cazul în care participarea este obligatorie pentru toate entităţile implicate

 când se adaugă o nouă instanta a entităţii, în general instantele corespunzătoare ale altor entităţi legate de aceasta nu sunt cunoscute încă sau nu există

 Exemplu: când se primeşte o nouă comandă, nu există încă o factură şi deci nu este posibilă construirea unei instante pentru relaţia ONORARE care conţine noua comandă

Pentru a determina cardinalitatea:

 Maxima: poate

 Minima: trebuie

Cardinalitatea unei relatii

Exemplu: fie functia Capitala : TARI -> LOCALITATI

Cardinalitate maxima: 1-1

 O tara, câte capitale poate avea ? => Una!

 O localitate, capitala câtor tari poate fi ? => Una!

 Relaţia Capitala : TARI -> LOCALITATI are cardinalitatea maxima one-one(1:1).

Cardinalitate minima:

 O tara, câte capitale trebuie sa aiba ? =>Una!

 O localitate, capitala câtor tari trebuie sa fie ? => Zero!

Relaţia Capitala : TARI -> LOCALITATI are cardinalitatea minimă one-zero (1:0).

TARI LOCALITATI 1(0) 1(1)

Cardinalitatea unei relatii

Exemplu: fie relatia LocNaştere: PERSOANE -> LOCALITĂŢI

Cardinalitata maxima: M-to-1

 Un angajat, in câte departamente poate lucra ? => Unul!

 Intr-un departament, cati angajati pot lucra? => Multi!

Cardinalitate minima:

 Un angajat, in câte departamente trebuie sa lucreze? => Unul!

 Intr-un departament, cati angajati trebuie sa lucreze? => Zero!

Obs. Prin conventie, cardinalitatea 1(1) este notata 1

ANGAJATI DEPARTAMENTE M(0) 1

Exemplu: fie relatia Task: SALARIAT -> PROIECT

Cardinalitate maxima: M-to-M

 Câţi salariaţi pot fi atasati la un proiect? Mulţi!

 La câte proiecte poate lucra un salariat? Multe!

Cardinalitate minima

 Câţi salariaţi trebuie sa fie atasati la un proiect? Zero!

 La câte proiecte trebuie sa lucreze un salariat? Zero!

 Dpdv matematic, reprezinta submultimi ale produsului cartezian al multimilor suport al asociatiei:

 Pot avea atribute care caracterizeaza leagatura dintre entitati. Ex. DataInceput, DataSfarsit

 Notația propusă pentru a reprezenta asociații many-to-one/one-to-many (1-N): notatia Mancas - sageata (->), indreptata spre entitatea cu cardinalitatie 1, etichetata cu numele functiei si cu cardinalitatile minime si maxime (ex. N(1))

 Notatia propusa pentru reprezentarea grafica a asociațiilor one-to-one (1-1): notatia Mancas - sageată cu capete duble, cu al doilea capat ne-tangent mulțimiii domeniu al funcției

Atribute multivaloare

 Atributele multivaloare trebuie folosite cu precauţie!

 reprezintă situaţii care pot fi modelate prin entităţi adiţionale legate prin relaţii unu-la-unu sau mai mulţi-la-mai mulţi cu entităţile la care se referă.

 Exemplu: atributul multivaloare Calificări pentru entitatea PERSOANĂ (o persoană poate avea mai multe calificări).

 Calificarea este un concept ce poate fi atribuit mai multor persoane ⇒ modelam acest concept cu ajutorul unei entităţi CALIFICARE legată de PERSOANĂ cu o relaţie mai mulţi-la-mai mulţi (M-M)

 sunt specifici fiecărei entităţi din schemă şi descriu conceptele schemei ce permit identificarea unică a instantelor (valorilor) acelei entităţi

 cunoscuti sub numele de chei (semantice)

 componentenele unei chei semantice:

 identificatori interni - unul sau mai multe atribute ale entităţii

 Ex. SALARIAT are cheia SerieCI \* NrCI

 INTERVALE\_ORARE(Id,Zi,OraInceput,OraSfarsit) are cheia Zi\*OraInceput\*OraSfarsit

 identificator extern - când atributele unei entităţi nu sunt suficiente pentru a identifica în mod unic instantele unei entităţii.

Identificatori

 Identificator extern.

Exemplu:

 studenţi înscrişi la diverse universităţi;

 doi studenţi din universităţi diferite pot avea acelaşi număr de înregistrare ⇒ pentru a putea identifica un student în mod unic avem nevoie atât de numărul său de înregistrare, cât şi de universitatea de care acesta aparţine.

 Identificator pentru entitatea STUDENT: atributul NrÎnreg şi entitatea UNIVERSITATE - identificator extern.

 Se observă că identificarea este posibilă prin relaţia obligatorie unu-la-mai mulţi dintre entităţile UNIVERSITATE şi STUDENT, care asociază fiecare student cu o singură universitate

Fiecare entitate din schema conceptuala trebuie să aibă cel putin o cheie semantica!

Proprietatile unei chei semantice:

 Unicitate:identifică unic o entitate

 Minimalitate: excluzând un atribut din cheie, combinaţia de atribute rămasă NU mai identifică unic instanţele entităţii respective

Ex. Cheie semantica: SerieCI \* NrCI ; daca eliminam atributul NrCI, atributul ramas, SerieCI, nu mai are proprietatea de unicitate!

 Supercheie: Un atribut sau o combinaţie de atribute care identifică unic o entitate.

 Supercheia are proprietatea de unicitate

 Supercheia NU are proprietatea de minimalitate.

Ex. SerieCI \* NrCI \* Nume este supercheie!

Ex. Entitatea ORAR(Id,Curs,Zi, OraInceput,OraSfarsit) are cheia semantica Curs\*Zi\*OraInceput. Daca adaugam OraSfarsit obtinem supercheie!

 In proiectarea bazelor de date, NU intereseaza determinarea supercheilor!

Cheia primara

 trebuie să fie unica şi cunoscuta la orice moment

 nu admite valori nule (necunoscute)

 trebuie sa nu conţina informaţii descriptive (fara semantica), să fie simplă, fără ambiguităţi;

 să fie stabilă (sa nu se modifice in timp)

 in general, se foloseste o cheie surogat (artificiala), de tip numeric. Ex.Id, Cod

 unica semnificație a cheii surogat este numerotarea entitatilor!

 ordinea dată de valorile cheii surogat nu are însă nici o semnificație (semantica) pentru obiectele reprezentate în baza de date!

 cheia primara = cheie sintactica

 Fiecare entitate din schema trebuie sa aiba o cheie primara surogat!

 Atributul ales cheie primară va apare subliniat în diagramă, daca se alege sa se reprezinte

Entitati dependente (weak)

Doua categorii de entitati:

 entitati normale (master, obisnuite -regular entities) si

 entitati dependente (dependente -weak entities): nu pot avea o existenta de sine statatoare

 existenta lor depinde de existenta altor tipuri de entitati

Exemplu: PROIECT – entitate normala, MODUL – entitate dependenta (un modul nu poate exista in afara proiectului)

 Intre cele doua entitati este o relatie one-to-many

 o cheie semantica a unei entitati dependente include un identificator extern, care provine de la entitata master (PROIECT), şi cel puţin un atribut al entităţii dependente (CodModul).

Generalizare (specializare) - de tip IS-A

 Pune în evidenta subgrupuri distincte, E1, E2, ...En, intr-o mulțime de entități de un anume tip, E

Ex. Mulțimea studenților străini este inclusă în mulțimea studenților

 O submultime de entitati distincte, Ei, se numeste subentitate

 Multimea de entitati parinte, E, se numeste superentitate

 Superentitatea E este mai generala, in sensul ca subentitatile sunt cazuri particulare ale superentitatii => E este o generalizare a lui E1, E2,...En, iar E1, E2,...,En sunt specializari ale entitatii E

Ex. PERSOANĂ este o generalizare pentru BĂRBAT şi FEMEIE

 Subentitatea

 mosteneste toate atributele supentitatii (inclusiv cheile semantice); poate avea, în plus, anumite atribute

 participă la relatii unde nu se potrivesc toate instanțele superentitatii

 În sens matematic: relația este de incluziune între mulțimi

Specializare (de tip IS-A)

 Specializarea se poate modela

 extinzând notația Mancas, folosind o săgeată care are un capăt notat cu semnul incluziunii (din algebră)

 subentitatea se desenează prin dreptunghi inclus în superentitate

Relatia dintre subentitate si superentitate este 1-1(0)

 prin sageata dubla

 prin sageata alba

 Specializarea este totala daca fiecare instanta a superentitatii se regaseste intr-o instanta a unei subentitati; altfel, ea este partiala

 Specializarea se numeste suprapusa daca o instanţă a unei superentităţi poate aparţine de una sau mai multe subentitati; altfel, ea este exclusiva

 Există generalizări cu mai multe niveluri, cunoscute sub numele de ierarhii.

 Generalizarea suprapusă poate fi transformată intr-o generalizare exclusivă prin adăugarea uneia sau mai multor subentitati, pentru reprezentarea entităţilor ce sunt „intersecţia” între entităţile ce se suprapun.

Exemplu:

Generalizarea PERSOANĂ pentru STUDENT şi ANGAJAT este parţială şi suprapusă (există studenţi care sunt şi angajaţi).Se poate adăuga entitatea STUDENTANGAJAT pentru a obţine o generalizare exclusivă.

Observatii finale asupra modelului E-R

 Modelul E-R este realizat pe baza a două construcţii de bază: entitate şi relaţie.

 o entitate poate participa în mai multe relaţii sau în nici una;

 o relaţie implică două sau mai multe entităţi;

 participarea unei entităţi într-o relaţie are o cardinalitate minimă şi una maximă.

 Modelul E-R mai are construcţiile atribut şi generalizare.

 un atribut are un nume şi aparţine unui concept de bază (entitate sau relaţie)

 atributele compuse sunt specializări ale atributelor şi sunt formate din unul sau mai multe atribute;

 o generalizare are o entitate părinte şi una sau mai multe entităţi copil

 este esenţială folosirea de nume diferite în cazul construcţiilor de bază pentru a evita ambiguităţile. Două atribute pot avea acelaşi nume dacă aparţin unor construcţii de bază diferite

Observatii finale asupra modelului E-R

 Pot exista mai multe diagrame pentru o problemă dată

 2 probleme majore pentru un design de bd

 Redundanţă: duce la informaţii inconsistente,

 Informaţii incomplete

Reteta succesului implică

 cunoaşterea profundă a teoriei

 experienţă

 o buna cunoastere si intelegere a subuniversului ce trebuie modelat

Modelul E-R– exemple DEA

 Modelarea procesului de inchiriere a unei masini

O companie ofera masini pentru inchiriat una sau mai multe zile. Compania incheie contracte cu clientii din diferite localitati.Prin intermediul unui contract, un client poate inchiria mai multe masini.Masinile sunt de mai multe tipuri, tariful de inchiriere fiind corespunzator tipului masinii respective.

 Modelarea participarii la cursuri

 Tipuri de asociaţii construite peste un alt tip de asociaţii

O companie ofera cursuri pentru instruire in diverse domenii.Cursurile se desfasoara in diverse localitati si sunt predate de catre instructori.La un curs pot participa mai multi cursanti,care pot urma mai multe cursuri oferite de companie. Fiecare curs se incheie cu un examen,evaluat cu o nota de absolvire.

 Modelarea acitivitatii dintr-o companie

 Obs. Specializarea (ISA) reprez. prin dreptunghiuri

 Modelarea acitivitatii unei firme de constructii

Reguli de operare (Business rules)

 O schemă E-R este adeseori insuficientă pentru a descrie toate aspectele unei aplicaţii în detaliu.

 intr-o schemă E-R sunt precizate doar numele conceptelor, acest fapt fiind insuficient pentru a explica semnificaţia acestora

 dacă schema este complexă, se poate întâmpla să nu poată fi reprezentate toate proprietăţile conceptelor care apar într-un mod inteligibil.

 este imposibilă reprezentarea unor proprietăţi ale datelor prin intermediul schemelor E-R

 Ex. un angajat nu poate avea un salariu mai mare decât managerul departamentului de care aparţine – constrangere de integritate

Modelul E-R nu furnizează mijloace potrivite pentru reprezentarea constrângerilor de integritate impuse datelor => O schemă E-R trebuie însoţită de o documentaţie care:

 facilitează interpretarea schemei

 permite descrierea proprietăţilor care nu pot fi exprimate direct prin construcţiile modelului E-R

Reguli de operare (Business rules)

 Unelte utilizate de analişti pentru a descrie proprietăţile unei aplicaţii care nu pot fi exprimate direct cu ajutorul modelului conceptual ⇒ permite specificarea regulilor unei aplicaţii.

Exemplu de regulă de operare: un angajat nu poate câştiga mai mult decât managerul său.

Clasificare:

 descriptive: descriere a unui concept relevant al aplicaţiei - o definire precisă a entităţilor, atributelor sau relaţiilor unui model E-R

 constrângere de integritate aplicată datelor aplicaţiei

 derivate - un concept care poate fi obţinut pe baza unei deducţii sau a unui calcul matematic din alte concepte ale schemei

Exemplu: atributul SalariuNet poate fi obţinut ca diferenta a atributelor SalariuBrut şi Taxe.

Este imperios necesara reprezentarea tuturor regulilor care descriu constrângeri ce nu sunt exprimate în schemă!

Reguli de operare (Business rules)

Regulile descriptive

 se exprimă în general în limbaj natural.

 pot fi organizate ca un dicționar de date.

 două tabele:

 primul tabel descrie entităţile din schemă: numele lor, definiţii informale în limbaj natural, lista tuturor atributelor şi cheile semantice

 al doilea tabel descrie relaţiile: numele lor, descriere informală, lista atributelor (cu descrieri posibile) şi lista entităţilor implicate împreună cu cardinalităţile de participare la relaţie.

Reguli de operare (Business rules)

Constrangeri de integritate:

 pot fi relationale si non-relationale

 folosesc definiţii formale si structuri de forma

<concept> trebuie / nu trebuie <expresie>

<concept> poate/ nu poate <expresie>

unde conceptele pot corespunde:

- unui concept a schemei E-R la care se referă

- unui concept derivat.

Exemplu:

(RO1) managerul departamentului trebuie să aparţină departamentului;

(RO2) un angajat nu trebuie să aibă salariul mai mare decât managerul departamentului căruia îi aparţine;

(RO3) un departament din filială Constanta trebuie administrat de un angajat cu mai mult de 10 ani vechime în companie

 Regulile care descriu instrumentele derivate pot fi exprimate

 prin specificarea operaţiilor (matematice sau de alt fel) care permit obţinerea conceptelor derivate.

 cu următoarea formă:

<concept> este obținut prin <operații pe concepte>

Exemplu:

(RO4) Numărul angajaţilor dintr-un departament este obţinut prin numărarea angajaţilor care aparţin departamentului respectiv

Reguli de operare pentru tratarea buclelor

 În orice graf orientat există următoarele trei tipuri de cicluri (bucle)

 Comutative: au un nod sursă, unul destinație, și două căi de a ajunge de la sursă la destinație

 Circulare: fiecare nod este atât sursă, cât și destinație

 Oarecare: cu lungime minim 4, cu minim două surse și două destinații unde prin nod sursă se înțelege unul din care pleacă săgeți iar prin nod destinație se înțelege unul în care sosesc săgeți

 Foarte multe asemenea cicluri au atașate constrangeri (non-relationale)

 există și cicluri neinteresante din punctul de vedere al proiectării bd (in general, ciclurile oarecare), atunci când ele nu au nici o constrângere atașată

Reguli de operare pentru tratarea buclelor

Bucle comutative

 Problema: Un angajat face parte dintr-un departament (divizie); fiecare departament are repartizate masini. Angajatii fiecarui departament pot conduce masinile departamentului respectiv.

 Soluție: Introducerea unei relatii MASINI\_FOLOSITE => bucla

MASINI\_FOLOSITE => ANGAJATI =>DIVIZIE <= MASINI <=MASINI\_FOLOSITE

 (RO5): Masina folosita de un angajat trebuie sa fie o masina repartizata departamentului din care face parte angajatul

Reguli de operare pentru tratarea buclelor

Bucle circulare

 (RO): Capitala unei tari trebuie sa fie un oras din tara respectiva.

Bucle de lungime 1 (recursive)

 (RO): O persoana nu poate fi casatorita cu ea insasi.

Reguli business - Exemplu

Considerăm subuniversul unui “Post TV”, în care interesează doar

următoarele mulţimi de obiecte şi proprietăţi ale acestora: meserii

(meseriile angajaţilor postului, precum şi cele ale colaboratorilor care

participă la emisiuni), persoane (nume, prenume, CNP, meserie,

localitate domiciliu) (angajaţii postului şi colaboratorii la emisiuni),

colaborări (colaboratorii fiecărei emisiuni în parte), tipuri de emisiuni

(tipurile de emisiuni difuzate de post), difuzari (emisiune, zi,

oraÎnceput, oraSfârşit) (orarul difuzării emisiunilor), sponsorizari

(sponsorii emisiunilor şi sumele plătite de aceştia pentru fiecare

emisiune în parte), firme (firmele ce sponsorizează emisiunile

postului), studiouri (studiourile din/în care se transmit/înregistreaza

emisiunile postului).

Sursa: C. Mancas, curs BD

Reguli business - Exemplu

Acest subunivers este supus următoarelor restricţii:

R1: nu pot exista două emisiuni cu acelaşi nume;

R2: nu pot exista două persoane cu acelaşi CNP;

R3: o persoană poate colabora la mai multe emisiuni;

R4: o emisiune poate avea mai mulţi colaboratori;

R5: o emisiune poate fi de un singur tip;

R6: o emisiune se transmite/înregistreaza dintr-un singur studio;

R7: o emisiune poate fi sponsorizată de mai multe firme;

R8: o firmă poate sponsoriza mai multe emisiuni;

R9: în orice moment, nu se poate difuza decât o singură emisiune;

R10: nici o emisiune nu se poate termina înainte de a începe;

R11: orice emisiune are un singur coordonator şef;

R12: un coordonator şef al unei emisiuni nu poate fi şi colaborator la acea emisiune (bucla: PERSOANE, COLABORARI şi EMISIUNI)

R13: o persoană are o singură meserie;

R14: o persoană are domiciliul într-o singură localitate;

R15: nu există două localităţi cu acelaşi nume;

R16: nu există două tipuri de emisiuni cu acelaşi nume;

R17: nu există două meserii cu acelaşi nume;

R18: nu există două studiouri cu acelaşi nume;

R19: nu există două firme cu acelaşi nume;

R20: orice emisiune poate fi difuzată zero sau de mai multe ori (chiar şi în aceeaşi zi, dar, desigur, la ore diferite).

Unelte de modelare

• DIA

• Diagramo

• ER Assistant

Proiectarea conceptuala a BD

 Proiectarea conceptuală a bazelor de date constă în construirea

unei scheme E-R (diagrama EA) care furnizează o descriere

optimă a cerinţelor clienţilor.

 Construcţia schemei este un proces iterativ, aceasta suferind o

serie de transformări şi corecţii

 Activitatea care precede etapa de proiectare conceptuala este cea

de Colectare si Analiza a Cerintelor.

Colectarea si analiza cerintelor

 Colectarea cerintelor: identificarea completă a problemelor pe care aplicaţia trebuie să le rezolve şi a caracteristicilor aplicaţiei

 Cerinţele pot proveni din mai multe surse, cum ar fi:

 Utilizatori ai aplicaţiei - informaţia este obţinută prin interviuri

 Documentaţie existentă referitoare la problema de rezolvat - reguli interne, proceduri de operare etc. Sunt necesare colectarea şi selecţia. Responsabilitatea revine proiectantului.

 Posibile aplicaţii anterioare care trebuie să fie înlocuite sau cu care noua aplicaţie trebuie să interacţioneze

 Cerinţele sunt transformate în specificaţii care în general sunt exprimate în limbaj natural şi din acest motiv pot fi ambigue şi dezorganizate.

 Analiza cerinţelor - clarificarea şi organizarea specificatiilor

Exemplu de cerinte date in limbaj natural

Se cere proiectarea unei baze de date pentru o companie de training pentru care s-au colectat specificaţiile prezentate mai jos. Datele au fost extrase prin interviuri cu angajaţii companiei. Dorim să creăm o bază de date pentru o companie care ofera cursuri de instruire.Pentru aceasta, trebuie să stocam date despre cursanți și instructori. Pentru fiecare participant la curs (aproximativ 5000), identificat printr-un cod, dorim să stocam cnp-ul, numele si prenumele, vârsta, genul, locul nașterii, numele, angajatorii anteriori (și perioada angajarii), adresa și numărul de telefon, cursurile urmate (există aproximativ 200 de cursuri) precum și evaluarea finală a fiecărui curs. Avem nevoie, de asemenea, să reprezentam seminariile la care fiecare cursant participă în prezent și, pentru fiecare zi, locurile și orele în sunt organizate clasele. Fiecare curs are un cod și un titlu și orice curs poate fi organizat de mai multe ori. Fiecărei organizări a unui curs particular îi spunem „ediţie” a cursului, iar fiecare editie este compusa din 8 intalniri. La o editie e unui curs pot participa maxim 10 cursanti. Pentru fiecare ediție vom reprezenta data de început, data de sfarsit, precum și numărul de participanți.

Exemplu de cerinte date in limbaj natural

În cazul în care un cursant este liber profesionist,trebuie cunoscut domeniul de expertiză și, dacă este necesar, titlul său. Pentru oricine care lucrează la companie, vom stoca nivelul şi poziţia deţinută.Pentru fiecare instructor (aproximativ 300), vom pastra numele, vârsta, locul nașterii, ediția cursului predat, editiile predate în trecut si cursurile pe care un titular este calificat să le predea. Numerele de telefon ale instructorilor sunt de asemenea stocate. Un instructor poate fi angajat permanent de către compania de training sau poate poate fi angajat temporar.

Este evident că cerinţele au ambiguităţi. Spre exemplu există:

 participant sau cursant

 titulari sau instructori

 cursuri sau seminarii

Reguli pentru scrierea specificaţiilor mai precis şi fără ambiguităţi

 Se alege un nivel potrivit de abstractizare. Se evită termenii prea generali sau prea specifici.

Exemplu:

 “și perioada angajarii “ => perioadă - dată start şi dată sfârşit

 titlu => titlu profesional

 evaluare => notă

 Se standardizează structura propoziţiei.

Exemplu:

 „pentru <concept> păstrăm <proprietăţi>”

 Se evită frazele complexe

Exemplu:

 angajat este preferat lui oricine care lucrează pentru companie

Reguli pentru scrierea specificaţiilor mai precis şi fără ambiguităţi

 Se identifică sinonimele şi omonimele

 Pentru sinonime se foloseşte un singur termen iar pentru omonime se caută alţi termeni

Exemplu:

 titular şi instructor, participant curs şi cursant

 loc care înseamnă locul de naştere cât şi locul unde se ţin orele

 Se marchează explicit referinţele. Absenţa referinţelor dintre termeni duce la concepte ambigue.

Exemplu:

 adresa şi numărul de telefon se referă la angajat sau angajator ?

Reguli pentru scrierea specificaţiilor mai precis şi fără ambiguităţi

 se construieste un Glosar de Termeni (vocabular)

 Pentru fiecare termen, vocabularul conţine:

 o scurtă descriere

 sinonime posibile

 referinţe la alţi termeni conţinuţi de vocabular cu care este în legătură logică

Termen Descriere Sinonim Referinte Cursant Participant la curs. Poate fi liber profesionist sau angajat

Participant Curs, Angajator (Companie)

Instructor Profesor de curs. Poate fi angajat temporar sau permanent

Titular Curs

Curs Curs oferit de companie. Poate avea mai multe editii

Seminar Instructor, Cursant

Companie Compania la care cursantul este sau a fost angajat

Angajator Cursant

Editie Organizare a unui curs. Poate avea mai multe intalniri, care se desfasoara in clase

Curs, Clasa

Reguli pentru scrierea specificaţiilor mai precis şi fără ambiguităţi

 se grupeaza cerintele in functie de termenii identificati in glosarul de termeni

Expresii de natură generală

Dorim să creăm o bază de date pentru o companie care ofera cursuri de instruire. Pentru aceasta, trebuie să stocam date despre cursanți și instructori.Expresii referitoare la cursanți.

Pentru fiecare cursant (aproximativ 5000), identificat printr-un cod, dorim să stocam cnp-ul, numele si prenumele, vârsta, genul, locul nașterii, numele, angajatorii anteriori (cu data de inceput si sfarsit), adresa și numărul de telefon, cursurile urmate (există aproximativ 200 de cursuri) precum și evaluarea finală a fiecărui curs.

Expresii referitoare la tipul de cursant

În cazul în care un cursant este liber profesionist,trebuie cunoscut domeniul de expertiză și, dacă este necesar, titlul său profesional. Pentru cursantii care au fost angajati vom stoca nivelul şi poziţia deţinută. Pentru fiecare angajator se vor stoca numele, adresa si numarul de telefon.

Rescrierea si structurarea cerintelor

Expresii referitoare la cursuri

Avem nevoie să reprezentam cursurile la care fiecare cursant participă în prezent, si, pentru fiecare zi, locurile și orele în sunt organizate clasele. Fiecare curs are un cod și un titlu și orice curs poate fi organizat de mai multe ori.

Expresii referitoare le editii

Fiecărei organizări a unui curs particular îi spunem „ediţie” a cursului, iar fiecare editie este compusa din 8 intalniri. La o editie e unui curs pot participa maxim 10 cursanti. Pentru fiecare ediție vom reprezenta data de început, data de sfarsit, precum și numărul de participanți.

Expresii referitoare la instructori

Pentru fiecare instructor (aproximativ 300), vom pastra numele, vârsta, locul nașterii, ediția cursului predat, editiile predate în trecut si cursurile pe care un titular este calificat să le predea. Numerele de telefon ale instructorilor sunt de asemenea stocate. Un instructor poate fi angajat permanent de către compania de training sau poate poate fi angajat temporar.

Cerinte operationale

După specificarea datelor trebuie specificate operaţiile care trebuie executate asupra acestor date.

Op1. se introduce un nou cursant, cu toate datele sale (operatia urmează să fie efectuata de aproximativ 40 de ori pe zi)

Op2. se inscrie un cursant la o ediție a unui curs (de 50 de ori pe zi)

Op3. se introduce un nou instructor, cu toate datele sale și cursurile pe care acesta este calificat sa le predea (de două ori pe zi)

Op4. se inscrie un instructor calificat la o ediție a unui curs (de 15 ori pe zi)

Op5. se afișează toate informațiile cu privire la edițiile anterioare ale unui curs cu titlu, orarul intalnirilor si numărul de cursanți (de 10 ori pe zi)

Op6. se afișeaza toate cursurile oferite, cu informații despre instructorii care au calificare pentru a le preda (de 20 de ori pe zi)

Op7. pentru fiecare instructor, se afiseaza cursanții din toate cursurile pe care acesta le preda in prezent sau le-a predat de-a lungul timpului (de 5 ori pe săptămână)

Op8. se efectueaza o analiză statistică a tuturor cursanților cu toate informațiile despre ei, despre edițiile cursurilor la care au participat și notele obținute (de 10 ori pe lună)

Reguli generale de proiectare

1) Dacă un concept are proprietăți semnificative și / sau descrie clase de obiecte cu o existență autonomă, este indicat să îl reprezentati ca o entitate

2) Dacă un concept are o structură simplă și nu are proprietăți relevante, este indicat să-l reprezentați printr-un atribut al unui concept la care se referă

3) Dacă cerințele conțin un concept care furnizează o legătură logică între două (sau mai multe) entități, este indicat să il reprezentați printr-o relație

4) Dacă unul sau mai multe concepte sunt cazuri particulare ale unui alt concept, este indicat sa il reprezentati prin generalizare/specializare

Strategii de proiectare conceptuala

Dezvoltarea unei scheme conceptuale pe baza specificațiilor sale trebuie să fie considerată un proces de proiectare, prin urmare, se pot aplica strategiile utilizate la alte discipline

• Top-down

• Bottom-up

• Inside-out

• Mixed

Strategia top-down

• Schema conceptuală este obţinută printr-o serie de rafinări succesive ale schemei iniţiale ce descrie toate cerinţele prin intermediul câtorva concepte abstracte.

• Fiecare nivel reprezentat conţine o schemă ce descrie informaţii diverse la diferite grade de detaliu.

• Trecerea de la un nivel la altul se face cu ajutorul unor transformări numite primitive de transformare de sus în jos

• operează pe un singur concept al schemei

• îl transformă într-o structură de complexitate mai ridicată, capabilă să descrie conceptul iniţial în detaliu

• sunt disponibile 6 primitive de transformare

• se aplică atunci când o entitate descrie două concepte logice diferite legate unele de altele

Exemplu

• în aplicaţia descrisă în secţiunea anterioară se poate începe cu entitatea CURS

• acest concept pare prea abstract, putând face deosebirea între:

• TIPCURS (care are un cod şi un titlu)

• EDIŢIECURS (care are o dată de start şi una de sfârşit)

• aceste două entităţi pot fi legate prin relaţia TIP

• este aplicată atunci când o entitate este alcătuită din subentităţi

Exemplu

• în aplicaţia noastră această transformare are loc atunci când ne dămseama că printre cursanti se poate distinge între:

• ANGAJAT

• LIBERPROFESIONIST

• se aplică atunci când o relaţie descrie două sau mai multe relatii diferite legând aceleaşi entităţi

Exemplu

• în relaţia PREDARE între instructori şi cursuri, PREDARECURENTĂ poate fi separată de PREDAREANTERIOARĂ

• se aplică atunci când o relaţie descrie un concept cu existenţă autonomă

Exemplu

• dacă relaţia CONTRACT între o entitate CONSULTANT şi o entitate COMPANIE are multe atribute, atunci ea este mai bine reprezentată printr-o entitate (asociativa) legată de altele prin intermediul unor relaţii binare

• se aplică pentru adăugarea unor proprietăţi (atribute) entităţilor.

Exemplu

• atunci când rafinăm entitatea CURSANT prin adăugarea atributelor:

• CNP

• Nume

• Vârstă

• Gen

• OraşDeNaştere

• se aplică atunci când se adaugă proprietăţi la o relaţie, într-o manieră similară transformării T5

Avantajul strategiei top – down:

• proiectantul poate începe cu o reprezentare completă a cerinţelor, chiar dacă lipsesc unele detalii

Dezavantajul:

• este necesară o viziune globală asupra tuturor conceptelor, ceea ce este dificil de realizat în cazul aplicaţiilor complexe

Strategia bottom-up

• specificaţiile iniţiale sunt descompuse în componente până când fiecare componentă descrie un fragment

elementar al specificaţiilor

• în acest punct, componentele sunt reprezentate prin scheme conceptuale simple

• aceste scheme vor fi combinate pentru a se obţine schema finală

• şi în acest caz se utilizează transformări elementare - primitive de transformare de jos în sus

• aceste primitive introduc în schemă concepte noi care nu au fost prezente până în acel moment, capabile să descrie aspecte ale aplicaţiei care nu au fost luate în considerare

• se aplică atunci când se identifică în specificaţii o clasă de obiecte cu proprietăţi comune

Exemplu

• în aplicaţia descrisă anterior se poate identifica entitatea CLASĂ (ce păstrează o anumită clasă la un anumit moment)

• se aplică atunci când se identifică în specificaţii o legătură logică între două entităţi

Exemplu

• în aplicaţia noastră se poate identifica relaţia CALIFICARE între entităţile INSTRUCTOR şi CURS

• se aplică atunci când se identifică în specificaţii o generalizare între entităţi

Exemplu

• entitatea INSTRUCTOR este o generalizare a entităţilor PERMANENT şi TEMPORAR

• se aplică atunci când se identifică în specificaţii o entitate care poate fi privită ca o agregare a unor serii de atribute

Exemplu

• se identifică entitatea CURSANT cu proprietăţile

• CNP

• Nume

• Vârstă

• Gen

• OraşDeNaştere

• se aplică atunci când o relaţie poate fi privită ca o agregare a unor atribute

Exemplu

• se identifică pentru relatia ANGAJARI\_ANTERIOARE proprietăţile DataStart si DataSfarsit

Avantajul strategiei bottom – up

• permite descompunerea problemei în componente simple care pot fi uşor identificate şi astfel procesul de proiectare poate fi atribuit mai multor proiectanţi dacă este necesar

Dezavantajul

• este necesară integrarea mai multor scheme conceptuale

Strategia inside-out

• caz particular al strategiei bottom-up

• se începe cu câteva concepte importante şi apoi pe baza acestora, proiectarea se extinde radial

• cu alte cuvinte se reprezintă mai întâi conceptele cele mai apropiate de conceptele iniţiale şi apoi procesul de proiectare se mută spre conceptele mai depărtate prin intermediul navigării prin specificaţii

• este necesară examinarea, din timp în timp, a tuturor specificaţiilor căutând concepte ce nu au fost reprezentate încă

• Avantaj: eliminarea paşilor de integrare din strategia de jos în sus

Ariile indicate prezintă o dezvoltare cronologică posibilă a schemei.

Strategia mixta

• Cerintele problemei se descompun intr-un numar de componente, precum in stategia bottom-up, dar nu se continua pana la determinarea componentelor elementare

• In acelasi timp, se defineste o schema initiala cu componentele principale ale aplicatiei; aceasta schema ofera o viziune unitara asupra intregului design si ajuta la integrarea schemelor realizate separat.

• Astfel, proiectantul poate examina conceptele principale si continua cu rafinarea lor, urmand o strategie top-down, sau poate extinde o latura a schemei cu concepte inca neindentificate, urmand o strategie bottom-up.

Proprietati ale schemei conceptuale

• Corectitudine

• Completitudine

• Accesibilitate

• Minimalitate

Proprietati ale schemei conceptuale

• Corectitudine

• O schemă conceptuală este corectă dacă utilizează corect construcţiile puse la dispoziţie de modelul conceptual.

• Se pot defini două tipuri de erori:

• erorile sintactice, marchează utilizarea ilegală a unei construcţii (ex.: generalizarea dintre relaţii în detrimentul entităţilor)

• erorile semantice marchează utilizarea unei construcţii care nu işi urmăreşte definiţia (ex.: utilizarea unei relaţii pentru a descrie faptul că o entitate este o specializare a altei entităţi)

Proprietati ale schemei conceptuale

• Caracterul complet al schemei

• O schemă conceptuală este completă dacă include concepte ce reprezintă toate cerinţele de date şi care permit execuţia tuturor operaţiilor incluse în cerinţele operaţionale

• Accesibilitatea schemei

• O schemă conceptuală este accesibilă când reprezintă cerinţele într-un mod natural şi uşor de înţeles.

• Minimalitatea schemei

• schema este minimală când toate specificaţiile datelor sunt reprezentate doar o singură dată în schemă

• schema nu este minimală când apar redundanţele – concepte derivate din altele

• o sursă tipică de redundanţe este prezenţa ciclurilor determinate de prezenţa relaţiilor şi/sau generalizărilor

• câteodată redundanţele sunt necesare din motive de proiectare, aceste situaţii fiind precizate în documentaţie

Metodă de abordare a proiectării conceptuale

În practică se aplică foarte rar o singură strategie de proiectare conceptuală.

Etapele ce trebuie parcurse pentru realizarea unei scheme conceptuale sunt:

1. Analiza cerințelor

(A) se construieste un glosar de termeni.

(B) se analizeaza cerințele și se elimina orice ambiguitate

(C) se grupeaza cerințele

2. Etapa de bază

(A) se determina conceptele cele mai relevante și se reprezinta intr-un dictionar de date initial, apoi intr-o schema initiala

3. Etapa de descompunere (care urmează să fie utilizată dacă este cazul sau este necesar)

(A) se descompun cerințele cu referire la conceptele prezente în dictionarul de date initial / schema initiala

4.Etapa iterativa (este repetată pentru toate schemele până când fiecare specificatie este reprezentata)

(A) se rafineaza conceptele din schema, pe baza cerințelor.

(B) se adauga noi concepte la dictionar/schema pentru a descrie orice părți ale cerințelor care nu sunt încă reprezentate.

5. Etapa de integrare

(A) se integreaza într-o diagrama generală (DEA) schemele rezultate in etapa 4

6. Analiza calitatii

• Verificarea corectitudinii şi realizarea restructurărilor necesare

• Verificarea caracterului complet al schemei şi realizarea restructurărilor necesare

• Verificarea minimalităţii, listarea redundanţelor şi dacă este necesar realizarea restructurărilor necesare

• Verificarea accesibilităţii şi realizarea restructurărilor necesare dacă este necesar

Exemplu de proiectare conceptuala

Compania de cursuri de formare

Strategie mixta cu bottom-up

• Schema initiala

• Schema rafinata (I)

Exemplu: compania de cursuri deformare

• Schema rafinata (II)

• Schema rafinata (III)

Se observă cu nu se poate stabili un

identificator pe baza acestor atribute ⇒

se decide folosirea CNP-ului

instructorului chiar dacă nu este cerut în

specificaţii.

• Schema rafinata (IV)

Exemplu: compania de cursuri de formare

• În acest moment se începe verificarea schemei obţinute.

• Se verifică dacă schema este completă prin întoarcerea la specificaţii şi

verificarea dacă toate datele sunt reprezentate şi toate operaţiile pot fi

efectuate.

Exemplu

• Să considerăm operaţia 7, în care se cer cursantii pentru toate cursurile

ţinute de un instructor.

• Datele pentru această operaţie se găsesc pe schemă în felul următor:

• se pleacă de la entitatea INSTRUCTOR

• se trece prin relaţiile PREDARECURENTĂ şi

PREDAREANTERIOARĂ, entitatea EDIŢIECURS, relaţiile

PREZENŢĂCURENTĂ şi PREZENŢĂANTERIOARĂ şi apoi se

ajunge la entitatea CURSANT

Exemplu: compania de cursuri deformare

Cu privire la minimalitate, să notăm că există o redundanţă în schemă:

• atributul NrParticipanţi al entităţii EDIŢIECURS se poate obţine prin

numărarea numărului de instanţe ale entităţii CURSANT care sunt

legate de ediţia respectivă.

• se va discuta despre eliminarea sau menţinerea acestei redundanţe într-

unul din cursurile viitoare, referitor la proiectarea logică

Schema trebuie să aibă o documentaţie potrivită!

• este importantă descrierea restricţiilor posibile care nu sunt exprimate în

schemă, sub forma regulilor de operare

Exemplu: un instructor predă un curs doar dacă este calificat să o facă

CURS 5

Modelul Relational al Datelor (MRD)

 Introdus în 1970 de matematicianul englez E.F.Codd

(cercetător IBM)

 Scopul: asigurarea independenței datelor (de nivelul fizic) și

depașirea restricțiilor impuse de modelele de până atunci

(ierarhic, rețea)

 Primul model matematic, cu fundamente pur

algebrice și de calcul predicativ de ordinul I

 Noțiunea centrală este cea de relație

Structura modelului relaţional

 Conceptele ce stau la baza modelului relaţional:

 relaţie

 tabel

 Relatie –Tabel – diferă ca noţiuni dar sunt strâns legate

 Relaţie - noţiune formală - domeniul teoriei mulţimilor

 Tabel - noţiune simplă şi intuitivă

Structura modelului relaţional

Definiţii

Fie mulţimile D1 şi D2.

Produsul cartezian dintre D1 şi D2 (notat

fiind mulţimea perechilor ordonate .

O relaţie pe mulţimile D1 şi D2 (numite domeniile relaţiei) este o

submulţime a produsului cartezian D1 × D2

 Mulţimile D1 şi D2 sunt finite sau nu ?

 Este de dorit ca domeniile să aibă dimensiuni infinite, a.î. să putem presupune

existenţa unei valori care nu e prezentă în baza de date.

 În practică relaţiile trebuie să fie finite deoarece bazele de date trebuie stocate în

sisteme computerizate de dimensiuni finite.

⇓

Vom presupune că bazele de date sunt alcătuite din relaţii finite, definite

pe domenii infinite

D1 × D2) se defineşte ca

(v1,v2 ), v1 ∈ D1, v2 ∈ D2

Definiţiile anterioare pot fi generalizate

Fie n > 0 mulţimi D1, D2 ,...,Dn , nu neapărat distincte

Produsul cartezian D1 × D2 × ...× Dn

este dat de mulţimea n-tuplurilor

(v1, v2 ,...,vn ), unde

O relaţie matematică pe domeniile D1, D2 ,...,Dn

produsului cartezian D1 × D2 × ...× Dn

este o submulţime a

 n - gradul (aritatea) produsului cartezian şi al relaţiei (dat de numarul

componentelor produsului cartezian)

 cardinalitatea relaţiei este dată de numărul n-tuplurilor

 analogia relaţie – mulţime implică:

• între n-tupluri nu există o ordine predefinită

• n-tuplurile unei relaţii sunt distincte unele de celelalte  nu exista dubluri

ale tuplurilor => un tabel reprezintă o relaţie doar dacă liniile sale sunt diferite

între ele

vi ∈ Di , i = 1,n

Structura modelului relaţional

• Notaţia non-poziţională – se asociaza (atribute) domeniilor unei

relaţii

• Atributele descriu rolurile jucate de domenii

EchipaGazda EchipaOaspete GoluriGazde GoluriOaspeti

Real Madrid Liverpool 3 1

Liverpool Milan 2 0

Real Madrid Roma 1 2

Roma Milan 0 1

Fig. 1 Relatie cu atribute

• fiecare atribut are nume unic

• toate valorile unui atribut apartin aceluiasi domeniu

• ordinea atributelor nu are importanta

Structura modelului relaţional

Pentru formalizarea conceptelor introduse anterior, definim:

D - mulţimea domeniilor

X - mulţimea atributelor

funcţia DOM : X → D care asociază fiecărui atribut un domeniu

Definiţie. Un tuplu definit pe o mulţime de atribute este o funcţie t care

asociază fiecărui atribut A∈ X o valoare din domeniul DOM (A)

Definiţie. O relaţie pe o mulţime de atribute X este o mulţime de tupluri

definite pe X

Notaţie. Fie t un tuplu definit pe o mulţime de atribute X şi A∈ Xun atribut.

Prin t[A] se notează valoarea tuplului t în domeniul DOM (A)

Exemplu: Fie t primul tuplu din relaţia prezentată în figura 1

t[EchipaOaspete]=Liverpool

t[EchipaOaspete, GoluriOaspeti]=Liverpool,1

Structura modelului relaţional

 BD sunt formate din mai multe relaţii

 tuplurile acestor relaţii conţin valori comune atunci când acest lucru

este necesar pentru stabilirea unor corespondenţe

EXAMENE

CURSURI

Fig. 2 Exemplu de bază de date relaţională

NrInreg Nume Prenume DataNastere

276545 Ionescu Maria 25/11/1980

485745 Popescu Ana 23/04/1981

200768 Georgescu Paul 12/02/1981

587614 Luca Radu 10/10/1980

937653 Maftei Lucian 01/12/1980

Student Nota Curs

276545 8 01

276545 9 04

937653 9 01

200768 9 04

Cod Denumire Titular

01 Fizica Melinte

03 Chimie Mardare

04 Chimie Dascalu

STUDENT

Structura modelului relaţional

 Schema unei relaţii este formată din numele relaţiei R şi o mulţime de

atribute X = {A1 , A2 ,..., An} şi se notează R(X); fiecărui atribut i se asociază un

domeniu. Ex. R(STUDENTI)= (NrInreg, Nume, Prenume, DataNastere)

 Schema bazei de date este formată dintr-o mulţime de scheme de relaţii:

R ={R1 (X1 ),R2 (X2 ),...,Rn (Xn )}

 Instanţa unei relaţii (pe scurt – relaţia) având schema R(X) este dată de

mulţimea r a tuplurilor definite pe mulţimea de atribute X

 Instanţa bazei de date (pe scurt – baza de date) având schema

R ={R1 (X1 ),R2 (X2 ),...,Rn (Xn )} este mulţimea r ={r1,r2 ,...,rn}de

relaţii în care fiecare ri , i = 1,n este o relaţie definită pe schema Ri(Xi)

Exemplu

Schema bazei de date din figura 2 este

R = {STUDENTI (NrInreg, Nume, Prenume, DataNastere),

CURSURI( Cod, Denumire, Titular),

EXAMENE (Student, Nota, Curs)}

Modelul Relaţional al Datelor (MRD)

 O relatie  un tabel in baza de date

 Un tuplu al relatiei  un rand/linie/inregistrare din tabel

 Un atribut al relatiei  o coloana/câmp din tabel

 Domeniu atribut  tip de date

 Baza de date  multime (colectie) finita de tabele

Modelul Relaţional al Datelor (MRD)

Modelul Relaţional al Datelor (MRD)

 Baza de date  multime (colectie) finita de tabele

Fig. 3 Reprezentarea tabulară a unei bd

Modelul Relaţional al Datelor (MRD)

• Schema bd din Fig.3 conţine patru scheme de relaţie:

AUTORI (IdAutor, Prenume, Nume)

VÂNZĂTORI (IdVânzător, Vânzător)

CĂRŢI (IdCarte, Autor, Titlu, Preţ)

VÂNZĂRI (IdVânzare, Vânzător, Carte, Cantitate)

• Instantele relaţiilor corespunzătoare sunt:

autori = {t1, t2, t3, t4}

vânzători = {t5, t6, t7}

cărţi = {t8, t9, t10, t11} şi

vânzări = {t12, t13, t14, t15, t16, t17, t18},

• Valorile tuplilor:

t1[#Autor] = 1, t1[Prenume] = “Emil”, t1[Nume] = “Cioran”,...,

t8[Titlu] = “Silogismele amărăciunii”, t8[Preţ] = 6.000 ş.a.m.d.

Valoarea NULL

 Informaţiile trebuie reprezentate sub forma unor tupluri omogene de date

 În particular, putem reprezenta în cadrul unei relaţii doar tupluri ce

corespund schemei relaţiei

 În practică există cazuri în care datele disponibile nu corespund cu

exactitate formatului ales

Exemplu: se consideră schema PERSOANĂ (Nume, Prenume,Adresă,

Telefon)

 Problemă: Pot exista tupluri pentru care valoarea atributului Telefon nu

este disponibilă

 Rezolvarea problemei indisponibilităţii valorilor: s-a inclus posibilitatea ca

fiecare atribut al unui tuplu să poată lua fie valori din domeniul asociat,

fie o valoare specială, denumită valoare null

 Valoarea null indică absenţa informaţiei din cauza ca NU SE CUNOASTE

 este o valoare suplimentară, ce nu este conţinută de domeniu;

 la definirea unei relaţii se pot specifica acele atribute care acceptă

valori null

Valoarea NULL

 Tipuri de valoare null

1. Valoare inexistentă (inaplicabilă)

2. Valoare temporar necunoscută

3. Valoare lipsită de informație

 Nu se știe dacă există sau nu, iar în caz că ar exista, nu se știe

nimic despre natura ei

IdPersoana Personalitate Engleză Franceză Rusă

1 Decebal φ φ φ

2 Constantin

Brâncoveanu

φ φ φ

3 Dimitrie

Cantemir

Da Da Da

4 Mihai I Da Da φ

5 φ φ φ φ

tip1

tip2

tip 3

Prima formă normală (FN1)

 Definiţie: o schemă de relaţie se zice că este în prima formă normală

(FN1) (sau "plată") dacă toate atributele ei sunt simple (atomice). Altfel,

relaţia se zice "încuibată" ("nested" în engleză).

 În modelul relaţional al datelor sunt considerate doar

relaţii plate

 Orice relaţie încuibată se poate transforma într-o relaţie plată

echivalentă!

Constrângeri de integritate

 Ideal, calitatea maximă a datelor este asigurată de corectitudinea lor.

Practic, calitatea maximă relativă ce poate fi asigurată datelor unei bd se

numește plauzibilitate

 Ex. înălțimea în cm a unei persoane este un număr între 30 și 225, în timp ce orice alt

număr este o valoare neplauzibilă pentru această proprietate

 Plauzibilitatea datelor din bd se poate garanta specificând și impunând

toate constrângerile de integritate

 Constrângerea de integritate - condiție ce trebuie respectata de toti

tuplii unei relații dintr-o baza de date, astfel încât interpretarea tuplilor să

fie întotdeauna acceptabilă

 O constrângere poate fi privită ca un predicat ce asociază valoarea

adevărat sau fals fiecărei instanţe

 Se pot defini mai multe constrângeri pentru o bază de date

 O bază de date este consistentă (validă, legală) dacă satisface toate

constrângerile impuse asupra tuplilor ei

 Altfel, se zice inconsistentă (ilegală).

Constrângeri de integritate

Clasificare:

• constrângeri intra-relaţionale - sunt definite pe o singură relaţie a bazei

de date. Ex. in relatia ANGAJAT, DataNasterii <= DataAngajarii-18

1. constrângeri la nivel de domeniu (la nivel de valoare) – impun

restricţii asupra domeniului unui atribut (ce valori sunt plauzibile

pentru un atribut)

2. constrangeri de obligativitate (de existenta, NOT NULL) – impun

restrictia ca atributul sa nu aiba valoarea null (atributul este

obligatoriu de completat)

3. constrangeri de unicitate – chei primare, chei semantice (atribute

(multimi de atribute) care trebuie sa identifice unic entitatile)

4. constrângeri la nivel de tuplu (CHECK) – pot fi evaluate pe fiecare

tuplu, independent de celelalte tupluri

• constrângeri inter-relaţionale – implică mai multe relaţii

• integritate referentiala, dependente functionale

Constrângeri de integritate

Exemplu. Fie baza de date din figura următoare

 constrângeri intra-relaţionale

 constrângeri la nivel de domeniu

• pentru atributul Notă din relaţia EXAMENE, numai valorile

cuprinse între 1 şi 10 sunt permise

• pentru data de nastere a studentilor sunt permise valori doar dintr-

un anumit interval

• sunt permise doar doua valori pentru atributul Promovat (DA sau

NU)

Fig.4 Bază de date cu informaţii incorecte

Constrângeri de integritate

 constrângeri intra-relaţionale

 constrângeri la nivel de domeniu

Dom(AnNastere)= {n∈N |1965< n<1978)}

Dom(NrMatricol)=NAT(5)

Dom(Student)=CHAR(64)

Dom(DataAngajare)= {d∈ Date |01.02.2020<=d<=data\_curenta )}

Dom(Promovat)={‘Da’,’Nu’} sau {True,False}

Regula de buna practica: Se vor da constrangeri de domeniu pentru

toate atributele unei relatii, specificand intervalul de valori

admisibile!

Constrângeri de integritate

Exemplu. Fie baza de date din figura următoare

Fig.4 Bază de date cu informaţii incorecte

 constrângeri intra-relaţionale

 constrângeri de existenta

 In ultimul tuplu al relatiei EXAMENE, atributul Nota are valoare

null => se impune restrictia ca atributul Nota sa NU admita valori

null. Exemplu: Nota NOT NULL

 aceeasi restrictie se poate impune pe atributele NrInreg, Nume,

Student, DataNastere, Curs, Cod, Denumire

Constrângeri de integritate

 constrângeri intra-relaţionale

 constrângeri de existenta

NrMatricol, Student NOT NULL

Regula de buna practica: Orice relatie trebuie să aibă cel putin un

atribut cu constrângere de existenţă, pentru a evita cazul unei

relatii cu toti tuplii cu valoare null (tabel lipsit de date!)

Constrângeri de integritate

Exemplu. Fie baza de date din figura următoare

Fig.4 Bază de date cu informaţii incorecte

 constrângeri intra-relaţionale

 constrângeri de unicitate (chei primare, chei semantice)

 ultimele două tupluri ale relaţiei STUDENTI conţin informaţii

pentru doi studenţi diferiţi dar cu acelaşi număr de înregistrare,

identificarea studenţilor fiind astfel ambiguă => se impune

constrangerea ca atributul NrInreg sa NU admita duplicate

 aceeasi restrictie se poate impune atributului Denumire, sau perechii

de atribute (Student,Curs) a relatiei EXAMENE

Regula de buna practica: Orice relație are o cheie primara SI cel putin o

cheie semantica!

Constrângeri de integritate

Exemplu. Fie baza de date din figura următoare

Fig.4 Bază de date cu informaţii incorecte

 constrângeri intra-relaţionale

 constrângeri la nivel de tuplu:

• în al doilea tuplu al relaţiei EXAMENE un student este considerat

promovat la examen deşi nota sa este 4

 constrângeri inter-relaţionale

 integritate referentiala

 al patrulea tuplu al relaţiei EXAMENE are, pentru atributul Student,

o valoare care nu apare printre numerele de înregistrare din relaţia

STUDENTI

• primul tuplu al relaţiei EXAMENE are, pentru atributul Curs, o

valoare care nu apare printre codurile cursurilor din relaţia CURSURI

Constrângeri de integritate

Constrângerile de tuplu - condiţii impuse valorilor fiecărui tuplu,

independent de celelalte tupluri

 sintaxă - expresii booleene (folosind conectorii AND, OR şi NOT) ale căror

atomi compară valorile atributelor implicate în constrângeri sau

expresii aritmetice folosind valorile atributelor

 singura condiţie este ca aceste expresii să utilizeze valorile unui singur tuplu

(din aceesi relatie sau relatii diferite)

Exemple

1) Expresia ce descrie constrângerea intra-relaţionala încălcate în exemplul

prezentat în figura 3:

(Promovat = 'DA’) AND (Nota ≥ 5)

2)Fie dată schema

ANGAJAT(Nume, CNP, Salariu, Deduceri)

Se poate defini o constrângere care impune condiţia ca salariul să fie mai

mare decat valoarea deducerilor: Salariu > Deduceri

Exprimarea formală: (∀t∈ANGAJAT)(t[Salariu] > t[Deduceri])

Constrângeri de integritate

Constrângeri de unicitate

 O mulţime de atribute K este o cheie (semantica) a relaţiei r daca are doua

proprietati:

 Identificare unica: ajută la identificarea în mod unic a tuplurilor unei relaţii

 r nu conţine nici o pereche de tupli care să aibă aceleaşi valori pentru

toate atributele din K i.e. ((oricare ar fi t1 !=t2), t1[K] != t2[K])

 Minimalitate:nici o submulţime proprie a lui K nu satisface proprietatea de

identificare unică  eliminând un atribut din multime, se pierde

proprietatea de identificare unică

 Dpdv matematic, o cheie este o funcţie injectivă , cu prop,că nici o funcţie

obţinută prin înlăturarea vreuneia din componentele sale nu mai este injectivă.

 Daca multimea K nu este minimala, atunci multimea este supercheie

 Obs. Pentru baza de date NU intereseaza idetificarea supercheilor!

 Orice cheie este supercheie, iar orice supercheie formată dintr-un singur atribut

este cheie.

 Deoarece relaţiile sunt mulţimi şi nu pot conţine duplicate, mulţimea tuturor

atributelor oricărei relaţii este supercheie => orice relaţie are cel puţin o cheie!

Constrângeri de integritate

Constrângeri de unicitate

 Cheile sunt cele mai importante constrângeri de integritate din schema

fiecărei relaţii.

 Identificarea şi includerea lor în schema bd este crucială pentru modelarea

oricărei aplicaţii.

Regula de buna practica: pentru orice relatie a unei baze de date

trebuie sa se stabileasca cel putin o cheie semantica!

Chei semantice: CodPersonal, NrMatricol

Constrângeri de integritate

Constrângeri de chei – Exemplu

 Chei semantice: NrMatricol

 Mulţimile {NrMatricol, Student}, {NrMatricol,AnNastere) sunt superchei!

 Fie relatiaANGAJAT={IdAng,Nume, CNP, SerieNrCI, DataNastere}

 Chei semantice: CNP, SerieNrCI

Constrangeri de integritate

Chei semantice –Algoritmul de asistență în proiectarea cheilor

unei relații

Pașii de parcurs presupun următoarele:

• Pentru fiecare atribut al relatiei, ne punem problema dacă este cheie

• După ce au fost identificate toate cheile formate dintr-un singur atribut, se trece la

combinații de câte două atribute

• Vom lua în considerare combinații care NU cuprind deja o cheie, deci nu vom

considera combinații cu atributele identificate ca chei la punctul 1

• Dupa ce au fost identificate toate cheile formate din 2 atribute, se trece la combinații

de câte 3 atribute

• Vom lua în considerare doar combinații care NU cuprind deja o cheie

(identificată la pașii anteriori)

• Se continuă cu combinații de câte 3, ..., n atribute până se identifică toate cheile

posibile

• Dacă la primii n-1 pași nu este identificată vreo cheie, automat combinația celor n

atribute este cheie (deci orice relatie are cel putin o cheie semantica)

Observație. Cheia surogat NU va fi niciodată interogată în algoritm.

Algoritmul se adresează doar atributelor semantice ale relației.

Constrangeri de integritate

Algoritmul de asistență în proiectarea cheilor unei relații

Exemplu:

Fie relația STUDENT, peste atributele IdStudent, NrMatricol, Nume, Prenume, AnNastere.

i=1

NrMatricol–cheie?Da, pentru că nu pot exista doi studenți cu același număr matricol

Nume–cheie? Nu, pentru ca pot exista doi sau mai mulți studenți cu același nume de

familie.

Prenume–cheie? Nu, pentru ca pot exista doi sau mai mulți studenți cu același prenume.

AnNaștere–cheie? Nu, pentru ca pot exista doi sau mai mulți studenți născuți în același an.

i=2

Nume ∙ Prenume –cheie? Nu, pentru că pot să existe mai mulți studenți cu același nume și

același prenume

Nume ∙ AnNaștere–cheie? Nu, pentru că pot să existe mai mulți studenți cu același nume

născuți în același an.

Prenume ∙ AnNaștere–cheie? Nu, pentru că pot să existe mai mulți studenți cu același

prenume născuți în același an.

i=3

Nume ∙ Prenume ∙ AnNaștere–cheie? Nu, pentru că pot să existe mai mulți studenți cu

același nume și același prenume, născuți în același an.

Constrângeri de integritate

Constrângeri de chei – Exemplu

 Fie relatia Carti (autor, titlu, editura, an\_aparitie)

 Ce ați alege ca si cheie semantica?

 Fie relatia Orar(zi, ora, sala, profesor, formatie, disciplina)

 Ce ați alege ca si cheie semantica?

Constrângeri de integritate

 Cheia primară (Primary Key – PK) - cheia prin intermediul căreia se realizează

referinţe între relaţii

 Constrângere: este interzisă utilizarea valorilor null pentru cheile primare!

 Valorile null sunt permise pentru celelalte chei

 Cheia primară NU poate fi reactualizată !

 Regula de buna practica: pentru a eficientiza executia interogarilor ⇒ se va

introduce un atribut suplimentar, numeric, care va fi generat şi asociat

fiecărui tuplu în momentul inserării în relaţia corespunzătoare

 Atributul se numeste cheie primara surogat (cheie sintactica)

 De obicei, un camp cu autonumărare (Autonumber)

 Domeniul acestui atribut este mulțimea numerelor naturale.

 Denumit cu Id

 Atributele care reprezintă cheia primară sunt fie subliniate, fie

precedate de semnul #

Constrângeri de integritate

 Atributul se numeste cheie primara surogat (cheie sintactica)

 De obicei, un camp cu autonumărare (Autonumber)

 Domeniul acestui atribut este mulțimea numerelor naturale.

 Denumit cu Id

 Folosește cel mai puțin spațiu posibil, în contextul memorării

legăturilor dintre tabele.

 In plus, operaţiile de join ce implica chei primare de acest tip sunt cele

mai rapide.

 Valorile ei nu se pot modifica, asa incat nu implica niciodata

propagarea modificarilor

Obs. Cheia primara poate si o combinatie de atribute, dar nu este recomandat!

STUDENTI (IdSt, NrMatricolm Numre, Prenume, DataNastere) CURSURI

IdSt NrMatricol Nume Prenume DataNastere

1 276545 Ionescu Maria 25/11/1980

2 485745 Popescu Ana 23/04/1981

3 200768 Georgescu Paul 12/02/1981

4 587614 Luca Radu 10/10/1980

5 937653 Maftei Lucian 01/12/1980

IdCs Cod Denumire Titular

1 01 Fizica Melinte

2 03 Chimie Mardare

3 04 Chimie Dascalu

Constrângeri de referinţă – integritate referentiala

 O constrângere de referinţă (sau cheie străină – Foreign Key - FK)

între o mulţime de atribute X ale relaţiei R1 şi altă relaţie R2 este

satisfăcută dacă valorile fiecărui tuplu din R1 corespunzătoare mulţimii

X se regăsesc printre valorile cheii primare a relaţiei R2.

(a) domeniul sau de definitie este compatibil cu cel al cheii primare PK a

relatiei R2

(b) valoarea sa într-un tuplu din relatia R1, fie este identica cu valoarea

cheii primare PK a unui tuplu oarecare din starea curenta a relatiei R2,

fie ia valoarea NULL

Constrângeri de integritate

Constrângeri de referinţă – integritate referentiala

 Cheia straina provine din relatiile 1:N din modelul E-R

 Atributul cheie straina este introdus in relatia (tabelul) care provine

din entitatea E-R etichetata cu N (notatie Mancas: entitatea din care

pleaca sageata!)

Ex. Relatiile LOCALITATI (R1 )

JUDETE (R2 ),

cheia straina: Judet

Constrângeri de integritate

LOCALITATI JUDETE

1,N 1,1

SGBD Access

Constrângeri de referinţă – integritate referentiala

• Cheile străine furnizează principalul mijloc de a "lega" între ei tupli din relaţii

diferite!

• Integritatea referenţială este proprietatea bazei de date care garantează că oricare

valoare a unei chei străine, FK, din relatia R1 se regăseşte printre valorile cheii

primare, PK, din relaţia referită, R2, sau cheia străină are valoarea NULL

• Reprezentare: FK ⊆ PK

• Obs:

• Este posibil ca o cheie straina sa fie compusa din mai multe campuri (dar nu este

optim!)

• Este posibil ca o cheie straina sa fie legata la o alta cheie decat cheia primara (dar

nu este recomant!)

• In unele SGBD este posibil ca o cheie straina sa fie legata la un camp care nu este

cheie, dar nu e recomandat!

Constrângeri de integritate

STUDENTI

EXAMENE CURSURI

IdSt NrInreg Nume Prenume DataNastere

1 276545 Ionescu Maria 25/11/1980

2 485745 Popescu Ana 23/04/1981

3 200768 Georgescu Paul 12/02/1981

4 587614 Luca Radu 10/10/1980

5 937653 Maftei Lucian 01/12/1980

IdEx Student Nota Curs

1 1 8 1

2 1 9 3

3 5 9 1

4 3 9 3

IdCs Cod Denumire Titular

1 01 Fizica Melinte

2 03 Chimie Mardare

3 04 Chimie Dascalu

STUDENTI CURSURI EXAMENE

N(1) N(1)

Constrângeri de referinţă

 Dangling pointer: o valoare a unei chei străine care nu are corespondent în cheia

primara din relatia referita!

 La operații de inserare: trebuie să ne asigurăm, la inserarea unei valori într-un

câmp cheie străină, că există valoarea respectivă în câmpul cheie corespunzător

 La operații de ștergere: la ștergerea unei linii dintr-o tabelă, trebuie sa verifică

ca respectiva înregistrare nu este referită în altă tabelă prin intermediul vreunei

chei străine

Constrângeri de integritate

STUDENTI

EXAMENE

CURSURI

IdSt NrInreg Nume Prenume DataNastere

1 276545 Ionescu Maria 25/11/1980

2 485745 Popescu Ana 23/04/1981

3 200768 Georgescu Paul 12/02/1981

4 587614 Luca Radu 10/10/1980

5 937653 Maftei Lucian 01/12/1980

IdEx Student Nota Curs

1 1 8 1

2 1 9 3

3 7 9 1

4 3 9 5

IdCs Cod Denumire Titular

1 01 Fizica Melinte

2 03 Chimie Mardare

3 04 Chimie Dascalu

Dangling pointer

Constrângeri de referinţă

 Dacă o înregistrare din STUDENTI este ştearsă, dar ea este referită din

relatia EXAMENE, exista mai multe soluții:

 se şterg toate înregistrările ce o refera din EXAMENE (NU E

RECOMANDAT!)

 nu se permite ştergerea înregistrării din STUDENTI

 Student din EXAMENE va avea asignată o valoare implicită.

 Student din EXAMENE va avea asignată valoarea null

Constrângeri de integritate

STUDENTI

EXAMENE

IdSt NrInreg Nume Prenume DataNastere

1 276545 Ionescu Maria 25/11/1980

2 485745 Popescu Ana 23/04/1981

3 200768 Georgescu Paul 12/02/1981

4 587614 Luca Radu 10/10/1980

5 937653 Maftei Lucian 01/12/1980

IdEx Student Nota Curs

1 1 8 1

2 1 9 3

3 4 9 1

4 3 9 4

Constrângeri de chei – Exemplu

• In cazul relatiilor care provind din entitati asociative, o cheie semantica este

formata din multimea cheilor straine + alte atribute, daca este cazul

• Exemplu: pentru relatia EXAMENE, o cheie semantica este

• Caz 1: Student \* Curs

• Caz 2: Student \* Curs \* DataExamen

Constrângeri de integritate

STUDENTI EXAMENE CURSURI

IdSt NrInreg Nume Prenume DataNastere

1 276545 Ionescu Maria 25/11/1980

2 485745 Popescu Ana 23/04/1981

3 200768 Georgescu Paul 12/02/1981

4 587614 Luca Radu 10/10/1980

5 937653 Maftei Lucian 01/12/1980

IdCs Cod Denumire Titular

1 01 Fizica Melinte

2 03 Chimie Mardare

3 04 Chimie Dascalu

STUDENTI EXAMENE CURSURI N(1) N(1)

IdEx Student Nota Curs

1 1 8 1

2 1 9 3

3 5 9 1

4 3 9 3

IdEx Student Nota Curs DataExamen

1 1 4 1 12/02/2022

2 1 9 3 16/02/2022

3 5 9 1 20/07/2022

4 1 9 1 21/07/2022

EXAMENE

Caz 1:

Caz 2:

Constrangeri de integritate

• Sarcina principală a proiectantului de bd este de a depista şi

formaliza constrângerile necesare universului modelat pentru

a le include în schema proiectată (zeci sau chiar sute ori mii

de constrângeri).

• Este absolut necesar ca aplicațiile software să fie gândite,

proiectate și implementate astfel încât să garanteze

plauzibilitatea datelor gestionate, adică cea mai înaltă

formă posibilă de calitate a lor!

Modelul Relational al Datelor

• Avantajele modelului relaţional:

• fundamentare matematică riguroasă;

• independenţă fizică a datelor;

• existenţa unor structuri de date simple;

• realizarea unei redundanţe minime;

• supleţe în comunicarea cu utilizatorul neinformatician.

• Ca limite ale modelului relaţional putem menţiona:

• rămâne totuşi redundanţă,

• ocupă spaţiu

• apar fenomene de inconsistenţă,

• nu lucrează cu obiecte complexe

• nu există mijloace perfecţionate pentru exprimarea constrângerilor

de integritate

• nu realizează gestiunea totala a datelor distribuite

• nu realizează gestiunea cunoştinţelor

MRD - Anomalii

• Supraîncărcarea unei relații cu prea multe semnificații

• Redundanță

• Dependență mare a datelor

• Solutie: regândirea schemei bazei de date (re-design) în scopul

eliminării anomaliilor

• Adăugările, modificările, ștergerile de informații ar trebui să se

facă în câte un singur tabel, și apoi să se propage în restul bd prin

legăturile definite între tabele.

• Metode

• Modelul Entități Asociații

• Teoria formală a normalizării (discutie in cursurile viitoare)

• Forme normale: FN1, FN3, FN4, FNBC, FNDK (discutie in

cursurile viitoare)

MRD - Anomalii

 Un tabel prezintă anomalie de modificare (update) dacă o

aceeaşi informaţie trebuie actualizată în înregistrări diferite.

 Un tabel prezintă anomalie de inserare dacă nu se pot insera

înregistrări decât cu adăugarea de valori null pentru unele

dintre coloane

 Anomalie de stergere: pierdere neintenționată de date din

cauza stergerii unor alte date.

MRD - Anomalii

Exemplu

 Anomalie de modificare: daca vrem sa modificam bugetul proiectului

Physics, trebuie sa facem modificarea in toate inregistrarile acestui proiect

 Anomalie de inserare: vrem sa adaugam un nou departament, fara a

cunoaste cine lucreaza in acesta (deci vom avea null pt name,salary)

 Anomalie de stergere: daca stergem pe Mozart, vom sterge si

departamentul Music

Anomalii de modificare (actualizare)

• Exemplu

• Redundata!

• Dependenta

de date!

• Anomalie de modificare: daca vrem sa modificam bugetul proiectului

Physics, trebuie sa facem modificarea in toate inregistrarile acestui

proiect

• Solutie: identificarea corectă a tipurilor de entități implicate,

modelarea DEPARTAMENTE si PROFESORI ca entitati/tabele

(relații) separată și reprezentarea Departament ca cheie străină în

PROFESORI cu legătură către noua entitate DEPARTAMENTE

Anomalii de inserare

• Exemplul 1

• Redundata!

• Dependenta

de date!

• Anomalie de inserare: vrem sa adaugam un nou departament, fara a

cunoaste cine lucreaza in acesta (deci vom avea null pt name,salary)

• Exemplul 2: fie relatia ANGAJATI(IdAng, Nume, CNP, Sef); dacă Sef

ar fi obligatoriu de introdus (not null), nu am putea introduce un

angajat al cărui șef nu îl cunoaștem (nu am putea introduce nici măcar

primul angajat, am fi într-un cerc vicios)

• Soluție, eliminarea constrângerii not null pentru atributul Sef!

Anomalii de steregere

• Exemplul 1

• Redundata!

• Dependenta

de date!

• Anomalie de stergere: daca stergem pe Mozart, vom sterge si departamentul Music

• Exemplul 2: fie relatia PERSOANE(IdP, Cod, Nume, Religie) - dacă s-ar șterge toate persoanele de o anumită religie, ar dispărea complet informațiile despre religia respectivă.

• Soluţia: tabel separat cu RELIGII și cheie străină în PERSOANE care semnifică religia unei persoane

CURS 6-7

Proiectarea logica

 Scop: construirea unei scheme logice ce

reprezintă corect şi eficient toate informaţiile

descrise într-o schemă E-R

 Etape:

 Restructurarea schemei E-R

 fază independentă de modelul logic ales

 se bazează pe criterii de optimizare a

schemei şi de simplificare a următoarei

etape;

 Translarea în modelul logic

 ţine cont de un anumit model logic (ex.

modelul relaţional)

 poate include alte optimizări, bazate pe

caracteristicile modelului logic.

Restructurarea schemei E-R

Scop: optimizarea a doi parametri:

 costul unei operatii (evaluat în funcție de numărul de instante de entități și

relații care sunt vizitate pentru a executa o operație în baza de date)

 cerința de stocare (evaluată în termeni de număr de octeți necesari pentru

stocarea datelor descrise de schemă).

Pentru a studia acesti parametri, trebuie sa stim incarcarea BD, adica:

 volumul de date

 numărul de instante ale fiecărei entităţi şi relaţii din schemă

 dimensiunea fiecărui atribut

 caracteristicile operației (de interogare, inserare date, stergere, modificare)

 tipul operaţiei (interactivă sau batch)

 frecvenţa operaţiei (numărul mediu de execuţii într-un anumit interval

de timp)

 datele implicate (entităţi şi/sau relaţii)

 operaţia 1: atribuie un angajat unui proiect;

 operaţia 2: găseşte datele pentru un angajat, pentru departamentul în care

lucrează acesta şi pentru proiectele în care este implicat;

 operaţia 3: găseşte datele pentru toţi angajaţii unui anumit departament;

 operaţia 4: pentru fiecare filială, găseşte departamentele, cu numele

managerilor şi lista angajaţilor din fiecare departament.

Restructurarea schemei E-R

În tabelul volumului de date, numărul apariţiilor unei relaţii depinde de doi parametri:

 volumul entităţilor implicate în relaţie

 de câte ori o instanta a acestor entităţi participă, în medie, într-o instanta a relaţiei; acest

parametru depinde de cardinalităţile relaţiei

Exemplu:

 numărul de instante ale relaţiei COMPOZIŢIE = cu numărul departamentelor

(cardinalitatea relaţiei indică faptul că fiecare departament aparţine doar unei filiale)

 numărul de instante ale relaţiei MEMBRU < decât numărul angajaţilor (exista angajaţi

care nu aparţin nici unui departament)

 dacă un angajat este implicat în medie în trei proiect, avem 6000 de instante ale relaţiei

PARTICIPARE (=> 6000/500=12 angajaţi în medie pentru fiecare proiect)

Restructurarea schemei E-R

 Schema de navigare -> Tabelul accesarilor:

 fragmente ale schemei E-R relevante pentru operaţie

 este utilă desenarea „căilor logice” care trebuie urmate pentru a accesa informaţiile

cerute.

Exemplu: operaţia 2: găseşte datele pentru un angajat, pentru departamentul în care lucrează

acesta şi pentru proiectele în care este implicat; Estimarea costului operaţiei 2:

• trebuie accesată o instanta a entităţii

ANGAJAT pentru a accesa o instanta

a relaţiei MEMBRU şi, prin

intermediul acesteia, a unei instante a

entităţii DEPARTAMENT

• pentru a obţine datele referitoare la

un proiect la care angajatul lucrează

trebuie să accesăm în medie trei

instante ale relaţiei PARTICIPARE

(deoarece am presupus că un angajat

lucrează în medie la trei proiecte);

apoi, prin această relaţie accesăm în

medie trei instanteale entităţii

PROIECT

Restructurarea schemei E-R

 Tabelul accesărilor

 În ultima coloană a acestui tabel se menţionează tipul accesului (R

pentru citire, W pentru scriere)

Restructurarea schemei E-R

 decide ştergerea sau păstrarea unor redundanţe

din schema E-R;

 înlocuieşte toate generalizările cu alte construcţii

 decide dacă este convenabilă partiţionarea unui

concept în mai multe sau unirea mai multor

concepte separate în unul singur

Analiza redundantei

 Redundanta într-o schemă conceptuală corespunde unei informatii care

poate fi derivata (obtinuta de o serie de operațiuni de regăsire) din alte

date.

 O schema conceptuala poate conține diferite forme redundanţă

 Cele mai frecvente exemple sunt:

 atribute ale căror valori pot fi derivate, pentru fiecare apariţie a unei

entităţi/relaţii, din valorile altor atribute pentru aceeaşi instanta

Ex. fiecare atribut poate fi dedus din celelalte două prin operaţii

aritmetic

Analiza redundantei

 Cele mai frecvente exemple sunt:

 atribute ce pot fi derivate din alte atribute aparţinând altor

entităţi/relaţii, de obicei prin intermediul funcţiilor agregat

atributul ValoareTotala a entităţii ACHIZITIE poate fi calculat din valorile atributului

Pret a entităţii PRODUS prin insumarea preţurilor produselor achiziţionate, după

cum se specifică în relaţia COMPUNERE

 atribute ce pot fi derivate din operaţii de numărare a apariţiilor

atributul NumarLocuitori poate fi obţinut prin numărarea apariţiilor relaţiei REZIDENŢĂ în

care apare un anumit oraş.

Analiza redundantei

 Cele mai frecvente exemple sunt:

 relaţii ce pot fi derivate din alte relaţii în prezenţa ciclurilor

Relaţia INVATARE dintre studenţi şi profesori poate fi derivată din relaţiile PREZENŢĂ şi

ATRIBUIRE.

Observaţie. Prezenţa ciclurilor nu generează în mod automat redundanţe

 Spre exemplu, dacă se înlocuieşte relaţia INVATARE cu relaţia

SUPERVIZARE reprezentând legătura dintre studenţi şi supervizori, atunci

schema nu mai este redundantă.

Analiza redundantei

Redundanta:

 avantaj: reducerea numărului de accesari necesare obținerii

informatiilor derivate;

 dezavantaje:

 o cerință de stocare mai mare (care are adesea un cost neglijabil)

 necesitatea de a efectua suplimentar operatii pentru a menține

datele derivate la zi.

 Decizia de a menține sau sterge o redundanta este luată de

compararea costului operatiilor care implică informatia derivata si a

spatiului de stocare necesar, în cazul prezentei sau absentei

redundantei.

Analiza redundantei

Exemplu

 operaţia 1: adaugă o persoană nouă cu oraşul de rezidenţă al

acesteia;

 operaţia 2: afişează toate datele referitoare la oraş (inclusiv numărul

de locuitori).

 Presupunem că încărcarea bazei de date este dată în tabelele de mai

jos

Analiza redundantei

1. cazul prezenţei redundanţei (atributul NumarLocuitori)

 Pp. că pentru stocarea numărului de locuitori dintr-un oraş sunt necesari

4 octeţi ⇒ datele redundante necesită 4x200=800 octeţi (nesemnificativ)

 Costul operaţiilor

 Pentru operaţia 1 (adaugă o persoană nouă cu oraşul de rezidenţă al

acesteia):

 1 operaţie de scriere în entitatea PERSOANĂ (adăugare persoană);

 1 operaţie de scriere în relaţia REZIDENŢĂ (adăugare pereche

persoană-oraş)

 1 operaţie de citire din entitatea ORAŞ (găsire oraş);

 1 operaţie de scriere în entitatea ORAŞ ( actualizare număr de

locuitori).

Analiza redundantei

1. cazul prezenţei redundanţei (atributul NumarLocuitori)

 Operaţia 1 necesită

 3 x 500 = 1500 accesări în scriere / zi

 1 x 500 = 500 accesări în citire / zi

 Pentru operaţia 2 (afişează toate datele referitoare la oraş):

 1 operaţie de citire din entitatea ORAŞ

 Operaţia 2 necesită 1 x 2 = 2 accesări în citire / zi (neglijabil)

 Pp. că accesul în scriere costă de două ori mai mult decât accesul în citire =>

numărul de accesări / zi în prezenţa redundanţelor este 2 x 1500 + 500 = 3500.

Analiza redundantei

2. cazul absenţei redundanţei

Costul operaţiilor

 Pentru operaţia 1 (adaugă o persoană nouă cu oraşul de rezidenţă al

acesteia):

 1 operaţie de scriere în PERSOANA;

 1 operaţie de scriere în REZIDENŢĂ.

Operaţia 1 necesită 2 x 500 = 1000 accesări în scriere / zi

Analiza redundantei

2. cazul absenţei redundanţei

Costul operaţiilor

 Pentru operaţia 2 (afişează toate datele referitoare la oraş):

 1 operaţie de citire din ORAŞ (neglijabil)

 în medie 5000 de operaţii de citire din REZIDENŢĂ pentru a determina

numărul de locuitori (5000 = 1000000 persoane / 200 oraşe)

 Operaţia 2 necesită 2 x 5000 = 10000 accesări în citire / zi

 Numărul de accesări / zi în absenţa redundanţelor este de 2 x 1000 + 10000

= 12000.

 Obs. că în al doilea caz sunt necesare 8500 accesări / zi în plus pentru a „salva”

mai puţin de un kilo-octet.

=> este mai convenabil să menţinem redundanţa în cazul acestei

probleme.

Eliminarea generalizărilor (specializarilor)

 Modelul relaţional nu permite reprezentarea directă a generalizărilor din

modelul E-R => transformărea acestor construcţii în alte construcţii ce pot

fi translatate cu uşurinţă.

 Trei moduri de reprezentare a unei generalizări cu ajutorul

entităţilor şi relaţiilor.

 Exemplu:fie schema din figura de mai jos

Eliminarea generalizărilor (specializarilor)

1. Înglobarea entităţilor copil (subentitati) în entitatea părinte (superentitate)

 Se elimină E1 şi E2 şi proprietăţile acestora sunt adăugate entităţii E0.

 Se adaugă atributul Atype lui E0 pentru a distinge tipul (E1 sau E2) unei instante a lui

E0.

 A11 şi A21 pot avea valoarea NULL pentru unele instante ale entităţii E0.

 Relaţia R2 va avea cardinalitatea minimă 0 pentru E0 deoarece instantele lui E2 formează

o submulţime a instantelor entităţii E0.

 Se foloseşte când operaţiile implică instante şi atribute ale E0, E1 şi E2 într-

un mod asemănător.

 Deşi se stochează valori NULL => mai puţine accesări în comparaţie cu celelalte

variante în care apariţiile şi atributele sunt distribuite celorlalte entităţi

Eliminarea generalizărilor (specializarilor)

2. Înglobarea entităţii părinte în entităţile copil

 Entitatea părinte E0 este eliminată & prop.de moştenire=>atributele sale, identificatorii şi

relaţiile în care era implicată sunt adăugate entităţilor copil E1 şi E2.

 Relaţiile R11 şi R12 reprez. restricţia relaţiei R1 pe instantele entităţii E1, respectiv E2.

 Doar dacă generalizarea este totală, altfel instantele lui E0 care nu sunt instantei nici ale

lui E1 nici ale lui E2 nu vor putea fi reprezentate.

 Utilă când subentitatile au multe atribute proprii si există operaţii care se referă

doar la instantele lui E1 sau ale lui E2 şi astfel se face distincţia între aceste

entităţi.

 Nu mai există în principiu atribute ce pot lua valori NULL.

 Numărul de accesări este mai redus în comparaţie cu a treia metodă, deoarece nu este

necesară accesarea lui E0 pentru a accesa atribute ale E1 şi E2.

Eliminarea generalizărilor (specializarilor)

3. Substituirea unei generalizări cu relaţii

 Generalizarea este transformată în două relaţii „una la unu”

 Nu există transfer de atribute sau relaţii,şi entităţile E1 şi E2 sunt identificate extern prin

entitatea E0 => cheia primara a entitatii E0 va fi cheie primara a entitatilor E1 si E2

 În noua schemă apar constrângeri în plus: nici o apariţie a lui E0 nu poate participa în

ambele relaţii RG1 şi RG2.; mai mult, dacă generalizarea este completă, fiecare instanta

a lui E0 trebuie să participe în exact una din relaţiile RG1 şi RG2.

 Utilă atunci când generalizarea nu este totală, subentitatile au atribute proprii

şi operaţiile se referă fie la instantele şi atributele lui E1(E2) fie ale lui E0

 Stocarea este mai mică în raport cu prima metodă datorită absenţei valorilor NULL.

 Creşte numărul accesărilor pentru a păstra consistenţa instantelor

Eliminarea generalizărilor (specializarilor)

Opţiunile prezentate nu sunt singurele posibile.

O variantă ar fi cea din figura următoare.

Partiţionarea / combinarea entităţilor şi relaţiilor

 Creşterea eficienţei accesului la datele dintr-o bază de date se poate realiza

conform următorului principiu:

Operaţiile de acces sunt reduse

 prin separarea atributelor aceluiaşi concept (entitate sau relaţie) ce sunt

accesate de operaţii diferite

 prin combinarea atributelor aparţinând unor concepte diferite ce sunt

accesate de aceleaşi operaţii.

Partiţionarea entităţilor

 una din entităţi descrie informaţii despre statutul unui angajat

 cealaltă descrie informaţii personale ale unui angajat.

 Utilă doar dacă operaţiile ce implică frecvent entitatea originală necesită, pentru

un angajat, ori numai informaţii personale, ori numai informaţii legate de

angajare.

 partiţionare verticală - un concept este divizat ţinând cont de atributele sale.

 Avantajul partiţionării verticale este generarea unor noi entităţi cu număr

mai mic de atribute decât

entitatea originală ⇒ noile entităţi pot

fi translate în structuri fizice din care

se poate extrage un volum mare de date

printr-un singur acces.

Partiţionarea entităţilor

 Partiţionarea orizontală - divizarea se face în raport cu instantele entităţii.

Exemplu:

 Pot exista operaţii asociate analiştilor şi operaţii asociate angajaţilor din

departamentul vânzări.

 În acest caz se dovedeşte utilă partiţionarea entităţii ANGAJAT în două

entităţi distincte, ANALIST şi VÂNZARE, având aceleaşi atribute ca

entitatea originală.

 Partiţionarea orizontală are ca efect secundar introducerea de duplicate

pentru relaţiile în care participă entitatea originală ⇒ impact negativ

asupra performanţelor bazei de date.

Ştergerea atributelor multi-valoare – introducerea entitatilor

dependente

 Necesara deoarece modelul relaţional nu permite reprezentarea directă a atributelor

multi-valoare

 Entitatea AGENTIE este separată în două entităţi: una având acelaşi nume şi aceleaşi

atribute cu entitatea originală, în afară de atributul multi-valoare (Telefon) şi o entitate

TELEFON cu atributul Numar.

 Entitatea TELEFON este entitate dependenta

 Entităţile sunt legate de o relaţie 1:N.

 Dacă atributul este opţional, atunci cardinalitatea minimă pentru entitatea TELEFON

trebuie să fie 0.

 O cheie semantica a entităţilor dependente conţine cheia primară a entităţii de care

depinde (cheia straina) plus unul sau mai multe atribute adiţionale.Ex. Agentie \* Numar

Combinarea entităţilor

 Operaţia inversă partiţionării

 Entităţile PERSOANA şi APARTAMENT aflate în relaţia PROPRIETAR sunt

combinate într-o singură entitate ce deţine atributele ambelor entităţi.

 Necesara deoarece majoritatea operaţiilor frecvente asupra entităţii

PERSOANA necesită informaţii în legătură cu apartamentul deţinut de

persoana respectivă ⇒ se doreşte evitarea accesărilor necesare extragerii

datelor prin intermediul relaţiei PROPRIETAR

Combinarea entităţilor

 Dezavantaj: posibilitatea apariţiei valorilor NULL în noua entitate

PERSOANA deoarece cardinalitatea entităţii originale PERSOANA indică

faptul că există persoane ce nu deţin un apartament.

 Operaţia de combinare se efectuează în general asupra relaţiilor 1:1 şi mai

rar asupra relaţiilor 1:N sau N:N.

 Motiv: apariţia redundanţelor în atributele non-cheie ale entităţii având

cardinalitatea N.

Alte tipuri de partiţionare şi combinare

 pentru a separa apariţiile unei relaţii ce sunt întotdeauna accesate separat;

 pentru a uni două (sau mai multe) relaţii între aceleaşi entităţi într-o

singură relaţie, atunci când apariţiile lor sunt accesate întotdeauna

împreună.

Combinarea entităţilor

 se face distincţie între jucătorii prezenţi ai unei echipe de fotbal şi cei din

trecut

Translarea (traducerea) în Modelul Relaţional (MR)

 Translarea în modelul relaţional este a doua etapă a proiectării logice.

 Plecând de la o schemă E-R se construieşte schema relaţională echivalentă

(dpdv.al capacităţii de a reprezenta informaţii echivalente).

Translarea în Modelul Relaţional (MR) Entităţi şi relaţii N:N

 Fiecare entitate (dreptunghi) va fi translată într-o relaţie MR (tabel) cu

acelaşi nume şi având aceleaşi atribute ca entitatea

 Cheia primara a fiecărei relaţii este dată de cheia primara (surogat) a

entităţii

 Fiecare relaţie E-R de tip N:N (entitate asociativa – romb incadrat de

dreptunghi) va avea drept corespondent o relaţie MR (tabel) cu acelaşi

nume, avand ca atribute atributele relaţiei E-R şi cheile primare ale

entităţilor implicate

 O cheie semantica a relaţiei MR va conţine cheile primare ale entităţilor

implicate, si posibil alte atribute

 In noua relatie MR, cheile primare ale entitatilor implicate poarta

denumirea de chei straine

 Relatia MR va avea un o cheie primara surogat

 Dacă entităţile sau relaţiile E-R au atribute opţionale, atributele

corespunzătoare din modelul relaţional pot avea valoarea NULL.

Translarea în modelul relaţional

Entităţi şi relaţii N:N

ANGAJAT(IdAng, Numar, Nume, Salariu)

PROIECT( IdProiect, Cod, Nume, Buget)

PARTICIPARE(IdPart, IdAng, IdProiect, DataStart)

 Obs. Domeniul atributului Numar este o mulţime de numere de înregistrare pentru

angajaţi, iar domeniul pentru atributul Cod este o mulţime de coduri de proiecte.

 În cazul în care se doreşte mărirea clarităţii schemei obţinute se pot opera redenumiri de

atribute:

PARTICIPARE(IdPart, Angajat, Proiect, DataStart)

 Trebuie impuse constrângeri de referinţă între cheile straine Angajat si Proiect, şi

relaţiile ANGAJAT, respectiv PROIECT.

Translarea în modelul relaţional

 Operaţia de redenumire este esenţială în cazul relaţiilor recursive

 Prin translare se obţine:

PRODUS(IdProdus, Cod, Nume, Pret)

COMPUNERE(IdComp, Componenta, Subcomponenta, Cantitate)

 Atributele Componenta şi Subcomponenta (chei straine) au ca domeniu mulţimea de

id-uri ale produselor.

 Trebuie impuse constrângeri de referinţă între Componenta şi PRODUS,şi

Subcomponentă şi PRODUS.

Translarea relaţiilor E-R ce implică mai mult de două entităţi

FURNIZOR(IdFurnizor, NumeFurnizor)

PRODUS(IdProd, Cod, Tip)

DEPARTAMENT(IdDept, Nume, Telefon)

LIVRARE(IdLivr, Furnizor, Produs, Departament, Cantitate)

 Se introduc constrangeri de referinta, de ex. PPPPPPPPPPPP ⊆ PPPPPPPPPPPP.II

 Se verifică dacă cheile straine, luate împreună, formează o cheie semantica, sau mai

sunt necesare alte atribute

Translarea în modelul relaţional

Translarea relaţiilor E-R dependente

AGENTIE(IdAgentie, Adresa,Oras)

TELEFON(IdTel, Agentie, Numar)

 Relatia TELEFON

 cheie semantica : Agentie\*Numar

 Se introduce constrangere de referinta petru atributul Agentie:

AAAAAAAAAAAAAA ⊆ AAAAAAAAAA AA.II

Translarea în modelul relaţional

Relaţii 1:N

 Fiecare entitate E-R etichetata cu cardinalitatea N se traduce intr-o relatie MR care va

avea atributele entitatii E-R si, in plus, identificatorul extern (cheia straina)

Exemplu: Relatia STUDENT va avea cheia straina Universitate.

 Schema este translatată sub forma:

STUDENT(IdStud, NrInregistrare, Universitate, Nume, AnInregistrare)

UNIVERSITATE(IdUniv, Nume, Oras, Adresa)

 Se impun constrângeri de referinţă între atributul Universitate al relaţiei STUDENT şi

relaţia UNIVERSITATE

Translarea în modelul relaţional

Relaţii 1:1

 Regulă: relatia 1-1 (sageata dubla, in notatia Mancas) devine atribut (cheie straina) in relatia

MR corespunzatoare entitatii cu participare obligatorie la relatie(entitatea din care pleaca

sageata).

 Ex1. participare obligatorie din partea celor două entităţi implicate.

 Există două posibilităţi valide de translare:

SEF(IdSef, Numar, Nume, Salariu, Departament, DataStart)

DEPARTAMENT(IdDept, Nume, Telefon, Filiala)

cu constrângere de referinţă între atributul Departament al relaţiei SEF şi relaţia DEPARTAMENT,

sau

SEF(IdSef, Numar, Nume, Salariu)

DEPARTAMENT(IdDept, Nume, Telefon, Filiala, Sef, DataStart)

cu constrângere de referinţă între atributul Sef al relaţiei DEPARTAMENT şi relaţia SEF.

Translarea în modelul relaţional

Relaţii 1:1

 Ex2. participarea opţională a uneia dintre entităţ

 Soluţia este:

ANGAJAT(IdAngajat, Numar, Nume, Salariu)

DEPARTAMENT(IdDept, Nume, Telefon, Filiala, Sef, DataStart)

cu constrângere de referinţă între atributul Sef al relaţiei DEPARTAMENT şi relaţia

ANGAJAT.

 Ex3. participarea opţională a ambelor enitati (pot exista departamente fără şef)

ANGAJAT(IdAngajat, Numar, Nume, Salariu)

DEPARTAMENT(IdDept, Nume, Telefon, Filiala, Sef, DataStart)

SAU

ANGAJAT(IdAngajat, Numar, Nume, Departament, Salariu)

DEPARTAMENT(IdDept, Nume, Telefon, Filiala, DataStart)

Translarea în modelul relaţional

 Translarea entităţilor

 Entităţile independente devin relatii (tabele) independente

 Entităţile dependente devin tabele dependente.

 Cheia primară a entităţilor dependente conţine cheia primară a entităţii

de care depinde (cheie straina) plus unul sau mai multe atribute

adiţionale.

 Subentităţile devin tabele, in anumite cazuri (vezi eliminarea specializarilor)

 Translarea asociatiilor (relatiilor)

 Relaţiile 1:1 şi 1:n devin chei straine.

 Relaţia 1:1 plasează cheia straina în tabelul cu participare obligatorie/mai

putine linii

 Relaţia m:n devine un tabel special, numit tabel asociativ, care are două chei

straine pentru cele două tabele asociate.

 O cheie semantica este compunerea acestor două chei straine plus

eventuale atribute adiţionale.

Translarea în modelul relaţional - Rezumat

 Translarea atributelor

 Un atribut singular (atomic) devine o coloană

 Un atribut compus se traduce in mai multe coloane

 Atributele multiple devin tabele dependendente ce conţin cheia primară a

entităţii (cheie straina) şi atributul multiplu.

 O cheie semantica este formata din cheia straina , plus una sau mai multe

coloane adiţionale.

 Pentru relaţii 1:1 şi 1:n, atributele relaţiilor vor aparţine tabelului care conţine

cheia straina, iar pentru relaţii m:n, atributele vor fi plasate în tabelele

asociative.

Translarea în modelul relaţional - Rezumat

Exemplu – compania cu cursuri de formare

Exemplu – compania cu cursuri de formare Tratarea specializarilor

• Instructor

• operatiile relevante nu fac nicio diferență între subentități

• subentitatile nu au atribute proprii in problema considerata

• putem elimina subentitatile si adauga atributul Tip (cazul 1: contopirea

• Cursant:

• operatiile relevante nu fac nicio diferență între subentități, dar aceste entități au

atribute specifice;

• putem deci pastra toate entitățile și adăuga două relații: în acest fel, nu vom avea

atribute cu posibile valori nule pentru entitatea părinte, iar dimensiunea relaţiilor va fi

redusă.(cazul 3: substituirea unei generalizări cu relaţii)

Separarea si fuziunea entitatilor si relatiilor

• Relațiile PREZENTA\_CURE NTA si PREZENTA\_ANTERIOARA să fie contopite, deoarece ele

descriu concepte similare între care operatiile nu fac nicio diferenta

• In acelasi mod tratam si relatiile PREDARE\_CURE NTA si PREDARE\_ANTERIOARA

• Atributul multivaloareTelefon (al entitatii Instructor) poate fi eliminat prin introducerea

unei noi entitati TELEFON legata printr-o relație 1-N cu entitatea INSTRUCTOR.

Exemplu – compania cu cursuri de formare

Analiza redundantei

Exemplu – compania cu cursuri de formare

Analiza redundantei

 Entitatea EDITIE\_CURS are un atribut NrParticipanti.

 Vom analiza costul a doua operatii in cazurile in care exista sau nu acest atribut.

Op2: se inscrie un cursant la o ediție a unui curs (de 50 de ori pe zi)

Din tabelele de acces obținem urmatoarele informatii:

 cu redundanta:

 Op2: avem 100 de accesari de tip Read si 100 de accesari de tip W pe zi

 fara redundanță:

 Op2: avem 50 de accesari de tip Read si 50 de accesari de tip W pe zi

Exemplu – compania cu cursuri de formare

Analiza redundantei

 Entitatea EDITIE\_CURS are un atribut NrParticipanti.

 Vom analiza costul a doua operatii in cazurile in care exista sau nu acest atribut.

Op5: se afișează toate informațiile cu privire la edițiile anterioare ale unui curs cu titlu, orarul

intalnirilor si numărul de cursanți (de 10 ori pe zi)

obs. O editie poate avea maxim 8 clase! obs. O editie poate avea maxim 10 cursanti!

Din tabelele de acces obținem urmatoarele informatii:

 cu redundanta:

Op5: avem 190 de accesari de tip Read pe zi

 fără redundanță:

Op5: avem 290 de accesari de tip Read pe zi

Exemplu – compania cu cursuri de formare

Analiza redundantei

 Entitatea EDITIE\_CURS are un atribut NrParticipanti.

 Trebuie avut in vedere ca operatia de scriere (W) este mai mare consumatoare de resurse

decat cea de citire (R). Astfel, atunci când redundanța este prezenta, avem dezavantaje în

ceea ce privește timpul de acces.

 Prin urmare, putem șterge atributul NrParticipanti al tipului de entitati

EDITIE\_CURS.

Exemplu – compania cu cursuri de formare

Exemplu – compania cu cursuri de formare

Modelul Relational al Datelor (MRD)

Schema logica

CURSANT (IdCursant, Cod, CNP, Nume, Varsta, OrasDeNastere, Gen)

LIBER\_PROFESIONST (IdCursant, Expertiza, Titlu\_Profesional)

ANGAJAT(IdCursant, Nivel, Pozitie, AngajatorActual, DataStart)

EDITIE\_CURS (IdEd,Cod, DataStart, DataSfarsit, Curs, Instructor)

CLASA (IdClasa, Timp, Sala, Data, Editie)

INSTRUCTOR (IdInstr, CNP, Nume, OrasDeNastere, Tip)

TELEFON (IdTel, Numar, Instructor)

CURS (IdCurs, Cod, Nume)

CALIFICARE (IdCalif, Curs, Instructor)

PREZENTA (IdPrez, Cursant, Editie, Nota)

ANGAJATOR (IdAng, Nume, StradaNr, Oras, Telefon)

LOCALITATE (IdLoc, Nume, Judet)

JUDET (IdJud, Nume)

ANGAJARI\_ANTERIOARE(IdAngAnt, Cursant, Angajator, DataStart, DataSfarsit)

Exemplu – compania cu cursuri de formare

Modelul Relational al Datelor (MRD) Constrangeri de integritate:

CURSANT (IdCursant, Cod, CNP, Nume, Varsta, OrasDeNastere, Gen)

Chei semantice: CNP

Constrangeri de domeniu:

Dom(Cod)=CHAR(6)

Dom (CNP)=CHAR(13)

Dom(Nume)=CHAR(100)

Dom(Varsta)= n ∈ /18 ≤ ≤ 65

Dom(Gen)={“F”,“M”}

Constrangeri de existenta: Cod, CNP, Nume, Varsta

Constrangeri de integritate referentiala:

OO OO ⊆ LL LL LL .II II

LIBER\_PROFESIONST (IdCursant, Expertiza, Titlu\_Profesional)

Constrangeri de domeniu:

Dom(Expertiza)=CHAR(200)

Dom(Titlu\_Profesional) = CHAR(200)

Constrangeri de existenta: Expertiza, Titlu\_Profesional

Constrangeri de integritate referentiala:

Id ⊆ .II

Exemplu – compania cu cursuri de formare

Modelul Relational al Datelor (MRD) ANGAJAT(IdCursant, Nivel, Pozitie, AngajatorActual, DataStart)

Constrangeri de domeniu:

Dom(Nivel)=CHAR(200)

Dom(Pozitie) = CHAR(200)

Dom(DataStart)= d ∈ /01.01.1980 ≤ ≤ 31.12.2100

Constrangeri de existenta: AngajatorActual, DataStart

Constrangeri de integritate referentiala:

Id ⊆ .II

AAAAAA AA AA AA AA AA ⊆ AAAAAAAAAAAAAA .II

EDITIE\_CURS (IdEd,Cod, DataStart, DataSfarsit, Curs, Instructor)

Chei semantice: Cod, Curs\*Instructor\*DataStart, Curs\*Instructor\*DataSfarsit

Constrangeri de domeniu:

Dom(Cod)=CHAR(10)

Dom(DataStart)= d ∈ /01.01.2020 ≤ ≤ 31.12.2100

Dom(DataSfarsit)= d ∈ /01.01.2020 < ≤ 31.12.2100

Constrangeri de existenta: Cod, DataStart, Curs, Instructor

Constrangeri de tuplu: DataStart<DataSfarsit

Constrangeri de integritate referentiala:

Curs ⊆ .II

II II ⊆ II .II II

Exemplu – compania cu cursuri de formare

Modelul Relational al Datelor (MRD)

CLASA (IdClasa, Timp, Sala, Data, Editie)

Chei semantice: Editie\*Data\*Timp, Sala\*Data\*Timp

Constrangeri de domeniu:

Dom(Timp)=TIME

Dom(Sala) = CHAR(100)

Dom(Data)= d ∈ /01.01.1980 ≤ ≤ 31.12.2100

Constrangeri de existenta: Editie, Data, Timp

Constrangeri de integritate referentiala:

Editie ⊆ .II

INSTRUCTOR (IdInstr, CNP, Nume, OrasDeNastere, Tip)

Chei semantice: CNP

Constrangeri de domeniu:

Dom(Nume)=CHAR(100)

Dom(Tip)={“Temporar”, “Permanent”}

Constrangeri de existenta: CNP, Tip

Constrangeri de integritate referentiala:

OO OO ⊆ LL LL LL .II II

Exemplu – compania cu cursuri de formare

Modelul Relational al Datelor (MRD) TELEFON (IdTel, Numar, Instructor)

Chei semantice: Instructor\*Numar

Constrangeri de domeniu:

Dom(Numar)=CHAR(15)

Constrangeri de existenta: Numar, Instructor

Constrangeri de integritate referentiala:

Instruct ⊆ II .II II

CURS (IdCurs, Cod, Nume)

Chei semantice: Cod, Nume

Constrangeri de domeniu:

Dom(Cod)=CHAR(6)

Dom(Nume) = CHAR(200)

Constrangeri de existenta: Cod, Nume

CALIFICARE (IdCalif, Curs, Instructor)

Chei semantice: Curs\*Instructor

Constrangeri de existenta: Curs, Instructor

Constrangeri de integritate referentiala:

Curs ⊆ .II

Instruct ⊆ II .II II

Exemplu – compania cu cursuri de formare

Modelul Relational al Datelor (MRD) PREZENTA (IdPrez, Cursant, Editie, Nota)

Chei semantice: Cursant\*Editie

Constrangeri de domeniu:

Dom(Nota)= n ∈ /4 ≤ ≤ 10

Constrangeri de existenta: Cursant, Editie

Constrangeri de integritate referentiala:

Cursant ⊆ .II

Editie ⊆ .II

ANGAJATOR (IdAng, Nume, StradaNr,Localitate, Telefon)

Chei semantice: Nume

Constrangeri de domeniu:

Dom(Nume)= CHAR(200)

Dom(StradaNr)=CHAR(200)

Dom(Telefon)=CHAR(15)

Constrangeri de existenta: Nume, Telefon

Constrangeri de integritate referentiala:

LL ⊆ LL LL LL .II II

Exemplu – compania cu cursuri de formare

Modelul Relational al Datelor (MRD)

LOCALITATE(IdLoc, Nume, Judet)

Chei semantice: Nume \* Judet

Constrangeri de domeniu:

Dom(Nume)= CHAR(200)

Constrangeri de existenta: Nume, Judet

Constrangeri de integritate referentiala:

⊆ .II

JUDET IdJud, Nume)

Chei semantice: Nume

Constrangeri de domeniu:

Dom(Nume)= CHAR(200)

Constrangeri de existenta: Nume

Exemplu – compania cu cursuri de formare

Modelul Relational al Datelor (MRD)

ANGAJARI\_ANTERIOARE(IdAngAnt, Cursant, Angajator, DataStart, DataSfarsit)

Chei semantice: Cursant \* Angajator \* DataStart, Cursant \* Angajator\*DataSfarsit

Constrangeri de domeniu:

Dom(DataStart)= d ∈ /01.01.1980 < < \_

Dom(DataSfarsit)= d ∈ /01.01.1980 < ≤ \_

Constrangeri de existenta: Cursant, Angajator, DataStart

Constrangeri de tuplu: DataStart<=DataSfarsit

Constrangeri de integritate referentiala:

Cursant ⊆ .II

Angajator ⊆ AAAAAA AA AA .II

Observaţii şi concluzii referitoare la

PROIECTAREA BAZELOR DE DATE

 Proiectarea conceptuală a bazei de date  construirea schemei

conceptuale (Diagramei Entitati-Asociatii) a bazei de date, care include

identificarea tipurilor importante de entităţi, relaţii, atribute.

 identificarea tipurilor de entităţi (E)

 specializarea /generalizarea tipurilor de entităţi;

 identificarea tipurilor de relaţii (R);

 identificarea şi asocierea atributelor (A) cu tipurile de E sau R;

 determinarea domeniilor atributelor;

 determinarea atributelor chei semantice;

 desenarea diagramei E/R;

 revizuirea modelului de date conceptual, împreună cu utilizatorul,

pentru a garanta că modelul este o reprezentare corectă a punctului de

vedere al utilizatorului asupra sistemului real analizat.

Observaţii şi concluzii referitoare la

PROIECTAREA BAZELOR DE DATE

 Proiectarea logică a bazei de date  transpunerea reprezentării conceptuale în

structura logică a BD, care include proiectarea relaţiilor.

 transpunerea modelului conceptual în modelul de date logic:

 Eliminarea entitatilor si atributelor simple si compuse (inlocuirea cu tabele,

respectiv coloane in tabele)

 Eliminarea relatiilor 1:N si 1:1 (inlocuirea cu chei straine)

 eliminarea relaţiilor M:N (inlocuirea cu tabele asociative);

 eliminarea atributelor multiple (înlocuirea prin relaţii dependente)

 normalizarea relaţiilor

 validarea modelului conform tranzacţiilor utilizatorului pentru a garanta că acesta

acceptă şi rezolvă operaţiile cerute de către model (se verifică faptul că modelul

furnizează toate informaţiile cerute de fiecare tranzacţie şi se reprezintă schematic

calea urmată de fiecare tranzacţie, direct în diagrama E/R)

 definirea constrîngerilor de integritate

 validarea modelului conform tranzacţiilor utilizatorului;

 revizuirea modelului de date conceptual împreună cu utilizatorii.

Observaţii şi concluzii referitoare la

PROIECTAREA BAZELOR DE DATE  Proiectarea fizică a bazei de date  implementarea fizică a schemei logice într-o

capacitate de stocare secundară corespunzătoare SGBD-ului ţintă.

 Transformarea relaţiilor extrase din modelul de date logic într-o formă care să poată fi

implementată în SGBD-ul relaţional ţintă (cu limbajul LDD).

 Proiectarea (definirea) constrângerilor sistemului real modelat în SGBD-ul ţintă

(CONSTRAINT, declanşatori).

 Proiectarea reprezentării fizice - determinarea organizării fişierelor şi a metodelor de

acces (indecşi) care vor fi utilizate pentru a stoca relaţiile de bază (modul în care

relaţiile şi tuplurile vor fi păstrate în cadrul capacităţii de stocare secundare).

 Introducerea unei redundanţe controlate (denormalizarea).

 Estimarea necesarului de spaţiu pe disc cerut de baza de date.

 Proiectarea mecanismelor de securitate:

 proiectarea vizualizărilor (VIEW);

 proiectarea regulilor de acces la relaţiile de bază şi la vizualizări (privilegii, role-uri).

 Monitorizarea neîntreruptă şi reglarea sistemului operaţional pentru obţinerea unor

performanţe maxime (micşorarea configuraţiei hardware, timpi de răspuns mai scăzuţi,

transfer eficient etc.)

Observaţii şi concluzii referitoare la

PROIECTAREA BAZELOR DE DATE

Observatii:

 Proiectarea reprezentării fizice:

 scopul este de a stoca datele în mod eficient. Pentru măsurarea eficienţei

poate fi analizată:

 capacitatea de stocare pe disc;

 timpul de răspuns;

 transferul de tranzacţii (numărul de tranzacţii care pot fi efectuate într-

un anumit interval de timp).

 Principiile de bază ale distribuirii datelor pe unităţile de disc:

 fişierele sistemului de operare să fie separate de cele ale BD;

 fişierele principale ale BD să fie separate de fişierele de indexare;

 fişierul jurnalului de recuperare să fie separat de restul BD.

Observaţii şi concluzii referitoare la

PROIECTAREA BAZELOR DE DATE Observatii:

 Proiectarea reprezentării fizice:

 analizarea tranzacţiilor  cunoaşterea funcţionalităţii acestora şi analizarea celor mai

importante dintre acestea. Pentru fiecare tranzacţie trebuie determinat:

 frecvenţa estimată cu care va fi rulată;

 relaţiile şi atributele accesate;

 tipul de acces (inserare, interogare, ştergere sau rectualizare);

 atributele care apar în predicate (condiţii);

 atributele care apar în join-uri;

 constrângerile de timp impuse tranzacţiilor

 Introducerea unei redundanţe controlate (denormalizarea)

 considerarea datelor derivate (calculate)

 Atributul NrParticipanti al relatiei EDITIE\_CURS

 considerarea dublării atributelor

 are ca scop reducerea numărului de join-uri necesare pentru efectuarea unei interogări.

CURS 8-9

Limbaje de interogare pentru BD

relaţionale

 SQL (Structured Query Language)

SELECT Nume FROM Studenti WHERE Varsta > 20

 Algebră relațională

 Calcul relational pe domenii

 Calcul relational pe tupluri

SQL

 Acronim pentru Structured Query Language

 Dezvoltat pentru sistemul de gestiune a bazelor de date System R, creat

de IBM Research Laboratory, San Jose, California, la sfârşitul anilor ’70.

 Standard, limbaj de referinţă pentru bazele de date relaţionale

 Standarde ANSI și ISO

 SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2008

 Limbaj declarativ (neprocedural)

 Fundamente teoretice: Algebra relaţională

SQL

 Sublimbaje:

 DDL – Data Definition Language - limbaj de definire a datelor

(comenzi pentru definirea unei scheme a unei baze de date relaţionale)

 CREATE, ALTER, DROP

 DML – Data Manipulation Language - limbaj de manipulare a datelor

(comenzi pentru modificarea şi interogarea unei instanţe a unei baze

de date relaţionale)

 INSERT, UPDATE, DELETE, SELECT

 DCL – Data Control Language – limbaj pentru controlul datelor

 GRANT, REVOKE

 Comenzi pentru gestiunea tranzacțiilor

 START TRANSACTION, COMMIT, ROLLBACK

Domenii elementare

 SQL pune la dispoziţie mai multe familii de domenii elementare, care pot

fi utilizate pentru definirea domeniilor asociate atributelor schemei.

 Caractere (Text) - permite reprezentarea caracterelor sau a şirurilor

de caractere

 Domenii numerice - permit reprezentarea valorilor de tip întreg

sau real

 Date calendaristice - oferă suport pentru gestiunea informaţiilor

temporale

 Valori logice – ofera suport pentru reprezentarea valorilor logice

(True, False)

Interogări în SQL

 SQL urmăreşte principiile calculului relaţional

 Cererile de interogare exprimate în SQL prezintă un aspect declarativ

sunt specificate proprietăţile rezultatului şi nu modul de obţinere

 Cererile SQL sunt pasate pentru execuţie optimizatorului de cereri.

 Optimizatorul de cereri este o componentă a SGBD care

 analizează cererea;

 selectează o strategie de execuţie;

 formulează o cerere echivalentă în limbajul procedural intern al

SGBD-ului

Interogări simple

 Instrucţiunea SELECT, are sintaxa:

SELECT ExprAtribut [[AS] Alias ] {, ExprAtribut [[AS] Alias]}

FROM NumeTabel [[AS] Alias ]

[WHERE Conditie]

[ORDER BY ExprAtribut [asc|desc]{,ExprAtribut [asc|desc]}

 O cerere SQL va lua în considerare doar liniile ce apartin tabelului listat în

clauza FROM şi va selecta liniile ce satisfac condiţia exprimată în clauza

WHERE.

 Rezultatul execuţiei unei cereri SQL este o relatie (tabel), având câte o linie

pentru fiecare linie selectată de clauza WHERE şi ale cărui coloane rezultă din

evaluarea expresiilor ExprAtribut ce apar în clauza SELECT (lista ţintă)

 Fiecare coloană poate fi redenumită cu ajutorul unui Alias ce urmează imediat

după expresie.

 Tabelele pot fi de asemenea redenumite prin intermediul unui Alias.

 În general, rezultatul unei interogări conţine linii, aranjate într-o ordine

oarecare. Dacă se doreşte impunerea unei ordonări după un anumit criteriu

asupra liniilor returnate de o interogare se va utiliza clauza ORDER BY

Interogări simple

 Instrucţiunea SELECT, are sintaxa:

SELECT ExprAtribut [[AS] Alias ] {, ExprAtribut [[AS] Alias]}

FROM NumeTabel [[AS] Alias ]

[WHERE Conditie]

[ORDER BY ExprAtribut [asc|desc]{,ExprAtribut [asc|desc]}

 Expresiile din SELECT pot fi

 constante,

 coloane din tabelele incluse in clauza FROM,

 expresii algebrice arbitrare bazate pe aceste coloane

 chiar alte (sub)interogari

Interogări simple

 Exemplu: se consideră o bază de date care conţine tabelele

ANGAJATI (IdA,Nume, Prenume, IdDept, Birou, Salariu)

DEPARTAMENTE (IdDept, Dept, Adresa, Oras)

cu precizarile ca: 1) salariul înregistrat este anual; 2) ANGAJATI.IdDept este cheie straina

ce refera atributul IdDept din Departamente

DEPARTAMENTE

ANGAJATI

IdA Nume Prenume IdDept Birou Salariu

1 Ionescu Maria 1 10 45

2 Popescu Ion 2 20 36

3 Popa Stefan 1 20 40

4 Dumitrescu Vasile 3 16 45

5 Ionescu Ion 4 14 80

6 Manole Radu 4 7 73

7 Luca Doru 1 75 40

8 Vasile Alina 2 20 46

IdDept Dept Adresa Oras

1

2

3

4

5

Administratie

Productie

Distributie

Planificare

Cercetare

Independentei

Primaverii

Central

Nicolina

Trandafirului

Iasi

Bucuresti

Focsani

Iasi

Cluj

Interogări simple

Interogarea 1: Să se găsească salariile angajaţilor cu numele Ionescu

SELECT Nume, Salariu AS SalariuAnual

FROM Angajati

WHERE Nume = “Ionescu”

 SQL este case Insensitive (ex. Nume ≡ NUME ≡ nume )

 Lista ţintă - specifică elementele schemei tabelului rezultat.

 Caracterul special \* poate să apară în lista ţintă şi reprezintă selecţia

tuturor atributelor tabelelor precizate în clauza FROM

Interogarea 2: Extrageti toate informaţiile referitoare la angajatul cu numele

Ionescu in ordine descendenta dupa Salariu.

SELECT \*

FROM Angajati

WHERE Nume = “Ionescu”

ORDER BY Salariu DESC

Nume SalariuAnual

Ionescu 45

Ionescu 80

IdA Nume Prenume Dept Birou Salariu

1 Ionescu Ion 4 14 80

5 Ionescu Maria 1 10 45

Interogări simple

 Lista ţintă poate conţine expresii aritmetice ce utilizează valorile

atributelor din fiecare linie selectată.

Interogarea 3: Găsiţi salariul lunar al angajaţilor cu numele Popescu.

SELECT Nume, Salariu/12 AS SalariuLunar

FROM Angajati

WHERE Nume = “Popescu”

Nume SalariuLunar

Popescu 3

Interogări simple

Clauza WHERE

 Condiţia din clauza WHERE este o expresie booleană: combinarea

predicatelor simple cu operatorii AND, OR şi NOT.

 Fiecare predicat simplu utilizează operatorii de comparaţie (=, >, >=, <,

<=, <>) şi are

 într-un membru, o expresie formată din valori ale atributelor dintr-o

linie, operatori aritmetici (+, -, \*, /, ^, mod ) si functii SQL

 în celălalt membru o valoarea constantă sau o altă expresie

 Prioritar este operatorul NOT, dar NU se introduce o precedenţă între

AND şi OR.

 Dacă într-o expresie se folosesc ambii operatori AND şi OR este

indicată specificarea precedenţei prin utilizarea parantezelor.

Clauza WHERE. Operatorul IN

Interogarea 4: Să se găsească prenumele angajaţilor cu numele Ionescu care

lucrează în departamentele 1 si 2

Varianta 1

SELECT Nume, Prenume , Dept FROM Angajati

WHERE Nume = “Ionescu” AND (IdDept = 1 OR IdDept = 2)

 Operatorul IN - utilizat pentru a specifica lista de valori

 Returneaza TRUE daca primul operand se regaseste in lista de valori

 Ex. IN (valoare1, valoare2, ...)

Varianta 2

SELECT Nume, Prenume , IdDept FROM Angajati

WHERE Nume = “Ionescu” AND IdDept IN (1, 2)

Interogări simple

Nume Prenume IdDept

Ionescu Maria 1

Clauza WHERE. Operatorul NOT IN

 Returnează FALSE dacă valoarea primului operand face parte din lista de

valori specificată de al doilea operand

Interogarea 5: Să se găsească angajatii al caror nume este diferit de Ionescu si

Popescu.

SELECT Nume

FROM Angajati

WHERE Nume NOT IN (“Ionescu”, “Popescu”)

Interogări simple

Clauza WHERE. Operatorul BETWEEN...AND

 Operatorul BETWEEN...AND – utilizat pentru a specifica un interval de valori

Exemplu

Interogarea 6: Să se găsească angajatii cu salariul intre 40 si 70. Ordonati descendent

dupa salariu si alfabetic dupa Nume

Varianta1

SELECT Nume, Prenume , Salariu

FROM Angajati

WHERE Salariu >=40 AND Salariu <=70

ORDER BY Salariu DESC, Nume

Varianta 2

SELECT Nume, Prenume , Salariu

FROM Angajati

WHERE Salariu BETWEEN 40 AND 70

ORDER BY Salariu DESC, Nume

Interogări simple

Nume Prenume Salariu

Vasile Alina 46

Dumitrescu Vasile 45

Ionescu Maria 45

Luca Doru 40

Popa Stefan 40

Nume Prenume Salariu

Ionescu Maria 45

Popescu Ion 36

Popa Stefan 40

Dumitrescu Vasile 45

Ionescu Ion 80

Manole Radu 73

Luca Doru 40

Vasile Alina 46

Clauza WHERE. Concatenarea sirurilor de caractere

 In MS Access – operatorul &

 In Oracle: ||

Interogarea 7. Sa se afiseze urmatoarele informatii despre fiecare

angajat: Nume Prenume are biroul nr. Birou

SELECT Nume & “ “ & Prenume & “ are biroul nr. “ & Birou AS Angajat

FROM Angajati

Interogări simple

Angajat

Ionescu Maria are biroul nr. 10

Popescu Ion are biroul nr. 20

Popa Stefan are biroul nr. 20

Dumitrescu Vasile are biroul nr. 16

Ionescu Ion are biroul nr. 14

Manole Radu are biroul nr. 7

Luca Doru are biroul nr. 75

Vasile Alina are biroul nr. 20

Clauza WHERE. Operatorul LIKE

 pentru compararea şirurilor de caractere

 compară un şir cu alt şir, specificat parţial cu ajutorul caracterelor

speciale

 In MS Access:

 Caracterul ’?’ substituie un caracter oarecare. Ex. b?ll => ball, bell,

si bill.

 Caracterul ’\*’ substituie un şir oarecare de caractere, posibil vid.

Ex. wh\* => what, white, si why

 Caracterul ’#’ substituie un caracter numeric. Ex. 1#3 => 103,

113, si 123.

Exemplu: comparaţia like ’ab\*ba?’ va fi satisfăcută de toate şirurile de

caractere ce încep cu secvenţa ab şi au în componenţă perechea ba

înainte de ultimul caracter

Interogări simple

Clauza WHERE. Gestiunea valorilor NULL – operatorul IS NULL

 Predicatul IS NULL este adevărat doar dacă atributul are valoarea NULL.

 Predicatul IS NOT NULL este adevărat în caz contrar celui prezentat

anterior.

 Sintaxa: Atribut IS [NOT] NULL

Clauza WHERE. Duplicate – predicatul DISTINCT

 Un tabel poate avea mai multe linii ce conţin aceleaşi valori pentru toate

atributele (duplicate)

 SQL admite duplicate în rezultatele interogării

 Sintaxa pentru eliminarea duplicatelor se foloseste predicatul DISTINCT:

SELECT [DISTICT] atribute

 Dacă în clauza SELECT sunt listate mai multe atribute, atunci combinația

valorilor acestor atribute trebuie să fie unică pentru ca o înregistrare să fie

inclusă în rezultatul interogării.

 Duplicatele sunt eliminate la sfarsit, inainte de executarea ordonarii

Interogari simple

Interogarea 8: Să se găsească departamentele în care lucreaza persoanele cu

numele Ionescu si cu salariul nenul.

SELECT IdDept

FROM Angajati

WHERE Nume = ’Ionescu’

AND salariu IS NOT NULL

Interogarea 9. Să se găsească departamentele în care lucreaza persoanele cu

numele Ionescu, fiecare departament apărând o singură dată.

SELECT DISTINCT IdDept

FROM Angajati

WHERE Nume = ’Ionescu’

Interogari simple

IdDept

1

4

1

IdDept

1

4

2

Nume IdDept Salariu

Ionescu 1 45

Popescu 2 36

Popa 1 40

Dumitrescu 3

Ionescu 4 80

Manole 4

Ionescu 1 40

Ionescu 2

Clauza WHERE. TOP n

 Se obtin doar primele n (procente) rezultate din interogare (în ordinea

dată de ORDER BY)

 Nu face distinctie intre valori egale (TOP 25 returneaza 26 de randuri

daca a 26+a este egala cu a 25-a)

 Alte variante:

 MySQL: LIMIT n

 ORACLE: WHERE Rownum <n

Interogarea 10: Sa se gaseasca primii 3 angajati, in ordinea salariilor.

SELECT TOP 3 Nume, Salariu

FROM Angajati

ORDER BY Salariu DESC

Interogari simple

Clauza ORDER BY

 In clauza ORDER BY se listează atributele după care se doreşte efectuarea ordonării

rezultatelor interogării

 Implicit ordonarea este in ordine crescătoare

 Se pot folosi mai multe atribute în această clauză (implică sortarea după mai multe

criterii)

Interogarea 11: Sa se afiseze angajatii in ordine alfabetica dupa nume si prenume, si

descrescator dupa salariu

SELECT Nume, Prenume, Salariu

FROM Angajati

ORDER BY Nume, Prenume, Salariu DESC

 Daca exista mai multe persoane cu același nume,

acestea ar fi ordonate alfabetic în ordine lexicografică

(De la „a‟ la „z‟) dupa prenume

 Dacă ar exista mai multe persoane cu același nume și același prenume, atunci acestea ar

fi ordonate în ordine descrescătoare a salariului

Interogari simple

Nume Prenume Salariu

Dumitrescu Vasile 45

Ionescu Ion 80

Ionescu Maria 45

Luca Doru 40

Manole Radu 73

Popescu Stefan 40

Popescu Stefan 36

Vasile Alina 46

Sunt dependente de SGBD

 Pentru tipuri de date numerice

 Sqr(n) - radacina patrata din n.

 Int(n) – aproximare la intreg a lui n.

 Round(n,m) – rotunjire la m zecimale a numarului n.

 Str(n) – conversie in sir de caractere

 Val(sir) – conversie in valoare numerica

 Pentru siruri de caractere

 Left(str,n) – extrage subsirul stang din str cu lungimea de n

 Right(str,n) – extrage subsirul drept din str cu lungimea de n

 Mid(str,n,m) – extrage subsirul din str ce incepe din pozitia n cu lungimea m

 Ucase(str) - converteste in majuscule str

 Lcase - converteste la litere mici

 Trim(str) – extrage spatiile din marginile lui str

 Len(str) - Intoarce lungimea sirului str

 Space(n) - Intoarce un sir cu n spatii

 InStr(str1, str2) - Intoarce pozitia str2 in str1

SGBD MS Access. Functii SQL

Interogarea 11: Afisaţi numele angajatilor cu majuscule si salariul semestrial, cu

2 zecimale.

SELECT UCASE(Nume), ROUND(Salariu/2,2) AS SalariuLunar

FROM Angajati

Interogarea 12: Sa se gaseasca angajatii al caror nume are mai mult de 6 litere.

Afisati numele si 3 litere din prenume, incepand cu pozitia 2.

SELECT Nume, MID(Prenume,2,3) AS

Prenume

FROM Angajati

WHERE LEN(Nume)>6

SGBD MS Access. Functii SQL

Nume Salariu

IONESCU 22.50

POPESCU 18.00

POPA 20.00

DUMITRESCU 22.50

IONESCU 40.00

MANOLE 36.50

LUCA 20.00

VASILE 23.00

Nume Prenume

Ionescu Maria

Popescu Ionela

Dumitrescu Vasile

Ionescu Mirela

Nume Prenume

Ionescu ari

Popescu one

Dumitrescu asi

Ionescu ire

 Pentru date calendaristice

 Date() – întoarce data curenta a sistemului

 Day(d) – întoarce ziua, ca valoare numerica, pentru data (d) specificata ca argument

 Month(d) – întoarce luna, ca valoare numerica, pentru data (d) specificata ca

argument

 WeekDay(d) - întoarce ziua din saptamâna, ca valoare numerica, pentru data (d)

specificata ca argument

 Year(d) - întoarce anul, ca valoare numerica, pentru data (d) specificata ca argument

 Now() - întoarce data si ora curenta a sistemului

 MonthName(d) - întoarce numele lunii, ca sir de caractere, pentru data (d)

specificata ca argument

 DateAdd(interval, numar,d) – întoarce data corespunzatoare adaugarii numarului

specificat de intervale din argumentul functiei; pentru interval se pot folosi urmatoarele

sabloane pentru a specifica perioada:

 “yyyy” – an

 “q” – sfert de an

 “m” – luna

 “y” – zi a anului

SGBD MS Access. Functii SQL

 “d “- zi

 “ww” – saptamana

 numar reprezinta numarul de

perioade;

 d reprezinta data calendaristica.

Se consideră tabelul PERSOANA (IdP, DataNastere Nume, Prenume, Oras)

Interogarea 13: Afisaţi lista persoanelor cu anul nasterii fiecareia

SELECT Nume, YEAR (DataNastere) AS AnulNasterii

FROM Angajati

Interogarea 14: Afisati lista persoanelor si varsta lor in ani

SELECT Nume, (Now() – DataNastere)/365 AS Varsta

FROM Angajati

Interogarea 15: Afisati lista persoanelor cu varsta peste 18 ani

SELECT Nume, (Now() – DataNastere)/365 AS Varsta

FROM Angajati

WHERE (Now() – DataNastere)/365 >=18

 Obs. Standardul SQL nu admite folosirea alias-urilor pentru coloane in clauza WHERE

SGBD MS Access. Functii SQL

IdP DataNastere Nume Prenume Oras

1 02/23/2000 Ionescu Maria Iasi

2 03/14/2010 Popescu Ion Bucuresti

3 05/16/1995 Luca Vasile

4 06/22/1999 Pastrascu Radu Suceava

Nume AnulNasterii

Ionescu 2000

Popescu 2010

Luca 1995

Pastrascu 1999

Nume Varsta

Ionescu 23

Popescu 13

Luca 28

Pastrascu 24

Nume Varsta

Ionescu 23

Luca 28

Pastrascu 24

 Pentru tipul de date logic

IIf(expr, valAdev, valFals)

unde:

 expr = expresia logica care se evalueaza

 valAdev = expresie a carei valoare se întoarce daca expr este adevarata

 valFals = expresia a carei valoare se întoarce daca expr este falsa

SGBD MS Access. Functii SQL

 Pentru tipul de date logic

Interogarea 16. Să se afiseze numele angajatilor, initiala prenumelui, salariul actual si salariul

majorat cu 10% daca lucreaza in departamentele nr. 1 si 3. Daca nu este eligibil pentru

majorare, se va afisa un mesaj corespunzator.

SELECT Nume & “ “ & Left(Prenume, 1) & “.” AS Angajat, IdDept, Salariu AS

SalariuActual, IIF(IdDept IN (1,3), Salariu \* 1.1, “Nu este eligibil pentru majorare”)

AS SalariuMajorat

FROM Angajati

SGBD MS Access. Functii SQL

Angajat IdDept SalariuActual SalariuMajorat

Ionescu M. 1 45 49.5

Popescu I. 2 36 Nu este eligibil pentru majorare

Popa S. 1 40 44

Dumitrescu V. 3 45 49.5

Ionescu I. 4 80 Nu este eligibil pentru majorare

Manole R. 4 73 Nu este eligibil pentru majorare

Luca D. 1 40 44

Vasile A. 2 46 Nu este eligibil pentru majorare

 Pentru valoarea NULL

NZ(expr,val) – pentru valoarea NULL a expresiei expr intoarce valoarea val specificata.

Exemplu: se consideră tabelul PERSOANA (IdP, CNP, Nume, Prenume, Oras)

Interogarea 17: Să se gaseasca numele angajatilor si orasul in care locuiesc, daca

se cunoaste, altfel sa se afiseze mesajul “Resedinta necunoscuta!”

SELECT Nume, NZ(Oras, “Resedinta necunoscuta!”) AS Resedinta

FROM Persoana

SGBD MS Access. Functii SQL

IdP CNP Nume Prenume Oras

1 2783456784351 Ionescu Maria Iasi

2 1983498784761 Popescu Ion Bucuresti

3 1745456734321 Ionescu Vasile

4 1883457883511 Ionescu Radu Suceava

Nume Resedinta

Ionescu Iasi

Popescu Bucuresti

Ionescu Resedinta necunoscuta!

Ionescu Suceava

 Dacă o interogare implică înregistrări din mai multe tabele, argumentul din clauza

FROM va reprezenta o listă de tabele.

 Fiind date relatiile R si S, jonctiunea theta, R ⋈θ S, este o relatie in care fiecare

tuplu este o asociere a doua tupluri, unul din R, celalalt din S, unde asocierea

indeplineste conditia de jonctiune θ.

 Conditia de jonctiune se defineste prin intermediul unei expresii logice care

contine operatori logici (and, or, not) sau de comparare (>, <, =, >=, <=, <>,

!=), nume de atribute ale relatiilor initiale (de obicei, chei primare şi străine) sau

constante din domeniul atributelor relatiilor initiale.

 Un caz particular al jonctiunii theta il reprezinta echi-jonctiunea (equi join), caz in

care conditia de jonctiune este implementata folosind doar operatorul de egalitate

(=).

 Jonctiunile theta in care conditia de jonctiune este implementata folosind un alt

operator decat cel de egalitate, se numeste non-echi-jonctiune (non-equi join)

 Suplimentar condiţiei de joncţiune, interogarea poate conţine şi alte criterii

necesare operaţiei de selecţie a datelor

Jonctiuni. Jonctiunea THETA

Interogare: Să se găsească numele angajaţilor şi oraşele în care aceştia lucrează

Obs. valorile din coloana IdDept din ambele tabele trebuie să coincidă.

SELECT Angajati.Nume, Angajati.Prenume, Departament.Oras

FROM Angajati, Departament

WHERE Angajati.IdDept = Departament.IdDept

ANGAJATI

DEPARTAMENTE

Jonctiunea THETA – Echi-jonctiune

Id Nume Prenume IdDept Birou Salariu

1 Ionescu Maria 1 10 45

2 Popescu Ion 2 20 36

3 Popa Stefan 1 20 40

4 Dumitrescu Vasile 3 16 45

5 Ionescu Ion 4 14 80

6 Manole Radu 4 7 73

7 Luca Doru 1 75 40

8 Vasile Alina 2 20 46

IdDept Dept Adresa Oras

1

2

3

4

5

Administratie

Productie

Distributie

Planificare

Cercetare

Independentei

Primaverii

Central

Nicolina

Trandafirului

Iasi

Bucuresti

Focsani

Iasi

Cluj

Cheie straina!

Cheie

primara!

 În interogarea precedentă s-a folosit operatorul punct (’.’) pentru identificarea tabelului

din care se extrag atributele.

 Folosirea acestei construcţii este necesară în cazul în care tabelele din clauza FROM

au atribute cu acelaşi nume, pentru a distinge între referinţele la atribute omonime.

 În cazul în care nu există posibilitatea apariţiei unei ambiguităţi se poate specifica

atributul fără a preciza tabelul căruia îi aparţine.

 Într-o interogare se pot utiliza alias-uri pentru tabele cu scopul de a scurta referinţa la

acestea (recomandat!)

 Alias-urile de tabel pot avea lungimea maximă de 30 caractere

 dacă în clauza FROM se introduce un alias pentru tabel, atunci acel alias trebuie să

înlocuiască numele tabelului în toată instrucţiunea SELECT

 alias-urile de tabel ar trebui sa fie semnificative

 alias-ul unui tabel este valid numai pentru SELECT-ul curent

Interogare: Să se găsească numele angajaţilor şi oraşele în care aceştia lucrează

SELECT Nume, Prenume, Oras

FROM Angajati AS a, Departamente AS d

WHERE a.IdDept = d.IdDept

Jonctiunea THETA

 Relaţia dintre tabelul ANGAJAT si BAREMSALARIU este o non-echi-joncţiune, în sensul

că nici o coloana din tabelul ANGAJATI nu corespunde direct unei coloane din tabelul

BAREMSALARIU

 Legătura dintre cele două tabele este următoarea: valorile coloanei Salariu din ANGAJATI

sunt cuprinse între valorile coloanelor sal\_min şi sal\_max din tabelul BAREMSALARIU

 Legătura se obţine folosind un operator, altul decât operatorul egal (‘=’).

 Interogare: determinati gradele salariilor angajatilor

SELECT Nume & “ “ & Prenume AS Nume, Salariu, Grad

FROM Angajati, BaremSalariu

WHERE Salariu BETWEEN sal\_min AND sal\_max

Jonctiunea THETA– Non-echi-jonctiune

IdA Nume Prenume IdDept Birou Salariu

1 Ionescu Maria 1 10 45

2 Popescu Ion 2 20 36

3 Popa Stefan 1 20 40

4 Dumitrescu Vasile 3 16 45

5 Ionescu Ion 4 14 80

6 Manole Radu 4 7 73

7 Luca Doru 1 75 40

8 Vasile Alina 2 20 46

Nume Salariu Grad

Ionescu Maria 45 3

Ionescu Ion 80 4

 suplimentar condiţiei de joncţiune, clauza WHERE poate conţine şi alte

criterii necesare operaţiei de selecţie a datelor.

Interogare: afisam codul angajatului Popa, numele, biroul şi locaţia

departamentului

SELECT Nume, Prenume, Oras

FROM Angajati AS a, Departamente AS d

WHERE a.IdDept = d.IdDept AND Nume = “Popa”

Jonctiunea THETA

Nume Prenume Oras

Popa Stefan Iasi

 Atunci când o condiţie de joncţiune este invalidă sau complet omisă,

rezultatul este un produs cartezian în care vor fi afişate toate combinaţiile de

linii din tabelele implicate.

 Un produs cartezian tinde să genereze un număr mare de linii, iar

rezultatul său este în general nefolositor => trebuie inclusă întotdeauna o

condiţie de joncţiune validă în clauza WHERE, cu excepţia cazului când se

doreşte în mod explicit combinarea tuturor liniilor din tabele implicate în

relaţie.

 Interogare: afişează numele fiecărui angajat şi numele departamentului în

care lucrează. Deoarece nu a fost specificată clauza WHERE, toate liniile

din tabelulANGAJATI (8) sunt combinate cu toate liniile din tabelul

DEPARTAMENTE (5) generând astfel 40 de linii în tabelul interogarii.

SELECT Nume, Prenume, Oras

FROM Angajati, Departamente

Jonctiuni

 SQL-92 introduce o sintaxă alternativă (RECOMANDATA!) pentru specificarea

joncţiunilor (in afara non-echi jonctiunilor) => se realizeaza distincţia între

condiţiile ce reprezintă condiţii de joncţiune şi cele ce reprezintă selecţii de linii.

 Sintaxa este:

SELECT ExprAtribut [[as] Alias] {, ExprAtribut [[as] Alias]}

FROM NumeTabel [[as] Alias] {[TipJonctiune] JOIN

NumeTabel [[as] Alias] ON CondJonctiune}

WHERE AlteConditii]

 În acest fel, condiţia de joncţiune este mutată din clauza WHERE în clauza FROM

 Parametrul TipJonctiune specifică tipul joncţiunii:

 Echi-jonctiune: INNER

 Jonctiune externa: LEFT (stanga), RIGHT (dreapta), FULL (completa)

Interogare: Să se găsească numele angajaţilor şi oraşele în care aceştia lucrează

SELECT a.Nume, a.Prenume, d.Oras

FROM Angajati AS a INNER JOIN Departament AS d ON a.IdDept = d.IdDept

Jonctiuni – operatorul JOIN

 În cazul unei echi-joncţiuni (INNER JOIN), liniile dintr-un tabel ce NU

au linii corespondente în celălalt tabel vor fi eliminate din rezultat

 Pentru a forţa apariţia unor astfel de linii în rezultat se poate apela la

joncţiunea externă,cu cele trei variante:

 LEFT JOIN – furnizează acelaşi rezultat ca şi INNER JOIN, dar

include şi liniile tabelului ce apare în stânga joncţiunii pentru care nu

există linii corespondente în tabelul din dreapta;

 RIGHT JOIN – păstrează liniile tabelului din dreapta ce nu au

corespondent în tabelul din stânga;

 FULL JOIN (completa) – furnizează acelaşi rezultat ca şi INNER

JOIN, suplimentat cu liniile excluse din ambele tabele

 Nu este definit în toate SGBD-urile

 Se poate implementa ca un UNION intre rezultatul unui LEFT

JOIN și al unui RIGHT JOIN

Jonctiuni externe

 Exemplu: se consideră o bază de date care conţine tabelele prezentate în

figură

SOFERI AUTOVEHICULE

Interogare: Să se găsească şoferii ce deţin autovehicul (echi-jonctiune)

SELECT s.IdS AS IdSofer, Nume, Prenume, Permis, NrInreg, Marca, Model

FROM Soferi AS s INNER JOIN Autovehicule AS a ON s.IdS =a.IdS

Jonctiuni externe

IdS Nume Prenume Permis

1

2

3

Ionescu

Popescu

Popa

Maria

Ion

Stefan

VR 001Y

PZ 111B

AP 222C

IdA NrInreg Marca Model IdS

1 IS01AAA BMW 323 1

2 SV02BBB BMW Z3 1

3 IS02CCC Lancia Delta 2

4 IS01EFD BMW 316 -

IdSofer Nume Prenume Permis NrInreg Marca Model

1 Ionescu Maria VR 001Y IS01AAA BMW 323

1 Ionescu Maria VR 001Y SV02BBB BMW Z3

2 Popescu Ion PZ 111B IS02CCC Lancia Delta

 Exemplu: se consideră o bază de date care conţine tabelele prezentate în

figură

SOFERI AUTOVEHICULE

Interogare: Să se găsească şoferii ce deţin autovehicule, incluzând şi şoferii

fără autovehicule (jonctiune externa)

SELECT s.IdS AS IdSofer, Nume, Prenume, Permis, NrInreg, Marca, Model

FROM Soferi AS s LEFT JOIN Autovehicule AS a ON s.IdS =a.IdS

Jonctiuni externe

IdS Nume Prenume Permis

1

2

3

Ionescu

Popescu

Popa

Maria

Ion

Stefan

VR 001Y

PZ 111B

AP 222C

IdA NrInreg Marca Model IdS

1 IS01AAA BMW 323 1

2 SV02BBB BMW Z3 1

3 IS02CCC Lancia Delta 2

4 IS01EFD BMW 316 -

IdSofer Nume Prenume Permis NrInreg Marca Model

1 Ionescu Maria VR 001Y IS01AAA BMW 323

1 Ionescu Maria VR 001Y SV02BBB BMW Z3

2 Popescu Ion PZ 111B IS02CCC Lancia Delta

3 Popa Stefan AP 222C - - -

 Exemplu: se consideră o bază de date care conţine tabelele prezentate în

figură

SOFERI AUTOVEHICULE

Interogare: Să se găsească şoferii ce deţin autovehicule, incluzând

autovehiculele fara sofer (jonctiune externa)

SELECT s.IdS AS IdSofer, Nume, Prenume, Permis, NrInreg, Marca, Model

FROM Soferi AS s RIGHT JOIN Autovehicule AS a ON s.IdS =a.IdS

Jonctiuni externe

IdS Nume Prenume Permis

1

2

3

Ionescu

Popescu

Popa

Maria

Ion

Stefan

VR 001Y

PZ 111B

AP 222C

IdA NrInreg Marca Model IdS

1 IS01AAA BMW 323 1

2 SV02BBB BMW Z3 1

3 IS02CCC Lancia Delta 2

4 IS01EFD BMW 316 -

IdSofer Nume Prenume Permis NrInreg Marca Model

1 Ionescu Maria VR 001Y IS01AAA BMW 323

1 Ionescu Maria VR 001Y SV02BBB BMW Z3

2 Popescu Ion PZ 111B IS02CCC Lancia Delta

- - - - IS01EFD BMW 316

 Exemplu: se consideră o bază de date care conţine tabelele prezentate în

figură

SOFERI AUTOVEHICULE

Interogare: Să se găsească toţi şoferii şi toate maşinile împreună cu

posibilele relaţii între ele

SELECT s.IdS AS IdSofer, Nume, Prenume, Permis, NrInreg, Marca, Model

FROM Soferi AS s FULL JOIN Autovehicule AS a ON s.IdS =a.IdS

Jonctiuni externe

IdS Nume Prenume Permis

1

2

3

Ionescu

Popescu

Popa

Maria

Ion

Stefan

VR 001Y

PZ 111B

AP 222C

IdA NrInreg Marca Model IdS

1 IS01AAA BMW 323 1

2 SV02BBB BMW Z3 1

3 IS02CCC Lancia Delta 2

4 IS01EFD BMW 316

IdSofer Nume Prenume ID NrInreg Marca Model

1 Ionescu Maria VR 001Y IS01AAA BMW 323

1 Ionescu Maria VR 001Y SV02BBB BMW Z3

2 Popescu Ion PZ 111B IS02CCC Lancia Delta

3 Popa Stefan AP 222C - - -

- - - - IS01EFD BMW 316

 Unele implementări SQL (ex. ORACLE) specifică joncţiunea externă prin

adăugarea unui caracter special sau a unei secvenţe de caractere (\* sau (+))

la atributele implicate în condiţia de joncţiune din tabelul deficitar in date

Exemplu: Să se găsească şoferii ce deţin autovehicule, incluzând şi şoferii

fără autovehicule (jonctiune externa)

=> tabelul deficitar in date este Autovehicule!

SELECT s.IdS IdSofer, Nume, Prenume, Permis, NrInreg, Marca, Model

FROM Autovehicule a, Soferi s

WHERE a.IdS (+) = s.IdS

Jonctiuni externe

IdSofer Nume Prenume Permis NrInreg Marca Model

1 Ionescu Maria VR 001Y IS01AAA BMW 323

1 Ionescu Maria VR 001Y SV02BBB BMW Z3

2 Popescu Ion PZ 111B IS02CCC Lancia Delta

3 Popa Stefan AP 222C - - -

 Unele aplicaţii impun realizarea unei joncţiuni între un tabel şi el însuşi.

 Când tabelul apare de mai multe ori in interogare, este obligatoriu sa utilizam

alias-uri

 Prin folosirea alias-urilor se poate referi un tabel de mai multe ori

 Introducerea unui alias are semnificaţia declarării unei variabile de tip tabel, ce

are acelaşi conţinut cu tabelul al cărui alias este

Interogarea: Se consideră baza de date de mai jos. Să se găsească

toţi angajaţii (id, nume si prenume) si sefii lor (id, nume si prenume).

Autojonctiuni. Utilizarea variabilelor

IdA Nume Prenume IdDept Birou Salariu Sef

1 Ionescu Maria 1 10 45 7

2 Popescu Ion 2 20 36 2

3 Popa Stefan 1 20 40 7

4 Dumitrescu Vasile 3 16 45 4

5 Ionescu Ion 4 14 80 5

6 Manole Radu 4 7 73 5

7 Luca Doru 1 75 40 7

8 Vasile Alina 2 20 46 2

Exemplu

Interogarea: Se consideră baza de date de mai jos. Să se găsească

toţi angajaţii (id, nume si prenume) si sefii lor (id, nume si prenume).

SELECT a.IdA AS IdAngajat, a.Nume & “ “ & a.Prenume AS Angajat,

s.IdA AS IdSef, s.Nume & “ “ & s.Prenume AS Sef

FROM Angajati as a INNER JOIN Angajati as s ON a.Sef = s.IdA

Autojonctiune!

Autojonctiuni

IdA Nume Prenume IdDept Birou Salariu Sef

1 Ionescu Maria 1 10 45 7

2 Popescu Ion 2 20 36 2

3 Popa Stefan 1 20 40 7

4 Dumitrescu Vasile 3 16 45 4

5 Ionescu Ion 4 14 80 5

6 Manole Radu 4 7 73 5

7 Luca Doru 1 75 40 7

8 Vasile Alina 2 20 46 2

 Functiile de grup (sau functii agregat) sunt functii care opereaza pe un set

(grup) de randuri pentru a da un rezultat (o valoare) pe acel set.

 Functiile de grup sunt:

 COUNT

 Sintaxă: COUNT ( < \* | [DISTINCT | ALL] ListaAtribute>)

 Opţiunea \* returnează numărul de linii din rezultat.

 Opţiunea DISTINCT returnează numărul valorilor distincte pentru

lista de atribute din rezultat.

 Opţiunea ALL returnează numărul liniilor ce conţin valori diferite de

NULL pentru lista de atribute.

 Dacă se specifică un atribut fără DISTINCT sau ALL se consideră

opţiunea implicităALL

Interogari agregate

 Functiile de grup sunt:

 SUM, MAX, MIN, AVG

 Sintaxă: <SUM | MAX | MIN | AVG> ([DISTINCT | ALL] ExprAtribut)

 SUM returnează suma valorilor deţinute de expresia atribut;

 MAX si MIN returnează valoarea maximă, respectiv minimă;

 AVG returnează media valorilor expresiei atribut.

 expresia ExprAtribut poate fi un atribut sau o expresie.

 operatorii SUM şi AVG acceptă ca argument expresii ce reprezintă

valori numerice sau intervale de timp.

 funcţiile MIN si MAX necesită definirea unei ordini în expresia atribut,

fiind aplicabile şi asupra şirurilor de caractere şi momentelor de timp.

Interogari agregate

Interogari agregate

Dacă colecția conține doar valori nule, atunci

• COUNT returnează 0

• Toate celelalte returnează NULL

 Caz 1: Functiile de grup se aplica unui singur set, fie tabelul insusi, fie un

subtabel obtinut prin selectie dupa anumite expresii logice

 In acest caz, clauza SELECT contine doar functiile grup cerute de problema

de rezolvat

 Caz 1.1: grupul este toata multimea de randuri selectata prin clauza FROM

SELECT ListaFnctiiGrup(ExprAtribut) AS Alias

FROM ListaTabele

 Caz 1.2: grupul este o submultime de randuri selectata prin clauza WHERE

SELECT ListaFunctiiGrup(ExprAtribut) AS Alias

FROM ListaTabele

WHERE conditii

Exemple: Se consideră baza de date cu relaţiile ANGAJATI (IdA,Nume, Prenume, IdDept,

Birou, Salariu), DEPARTAMENT (IdDept,Dept, Adresa, Oras)

Interogari agregate

CAZ 1.1. - Exemple

 Interogare: Sa se calculeze totalul si media salariilor din firma

SELECT SUM(Salariu) AS TotalSalarii, AVG(Salariu) AS MedieSalarii

FROM Angajati

 Interogare: Să se găsească numărul valorilor distincte pentru atributul

Salariu pentru toţi angajaţii din tabela ANGAJAŢI.

SELECT COUNT (DISTINCT Salariu)

FROM Angajati

 Interogare: Să se găsească numărul de linii din tabelul ANGAJATI care au

valori diferite de NULL pentru atributul Salariu.

SELECT COUNT (Salariu)

FROM Angajati

Interogari agregate

TotalSalarii MedieSalarii

405 50.625

CAZ 1.2 - Exemple

 Interogare: Sa se calculeze totalul si media salariilor din departamentul 1

SELECT SUM(Salariu) AS TotalSalarii, AVG(Salariu) AS MedieSalarii

FROM Angajati

WHERE IdDept = 1

 Interogare: Să se găsească salariul maxim pentru angajaţii care lucrează într-

un departament din Iasi.

SELECT MAX(Salariu)

FROM Angajati AS a INNER JOIN Departament AS d ON a.IdDept =

d.IdDept

WHERE Oras = “Iasi”

Interogari agregate

TotalSalarii MedieSalarii

125 41.66

IdA Nume Prenume IdDept Birou Salariu

1 Ionescu Maria 1 10 45

3 Popa Stefan 1 20 40

7 Luca Doru 1 75 40

 Caz 2: Functiile de grup se aplica mai multor seturi, care reprezinta grupuri

de randuri din tabel, obtinute prin aplicarea unor criterii de grupare

 Clauza GROUP BY specifică modul în care va fi împărţit tabelul în

submulţimi de randuri

 Clauza acceptă ca argument o mulţime X de atribute (criterii de grupare),

iar interogarea va opera separat pe fiecare set ce posedă aceleaşi valori

pentru X.

 In acest caz, atributele ce pot apărea în clauza SELECT trebuie să fie o

submulţime a criteriilor de grupare ( a atributelor din clauza GROUP

BY)

SELECT ListaCriteriiGrupare, ListaFunctiiGrup(ExprAtribut) AS Alias

FROM ListaTabele

GROUP BY ListaCriteriiGrupare

Interogari agregate – Clauza GROUP BY

 Caz 2 - Exemple

 Se consideră tabela ANGAJATI (IdA,Nume, Prenume, IdDept, Birou, Salariu)

 Interogare: Să se găsească suma salariilor angajaţilor din fiecare departament.

SELECT IdDept, SUM(Salariu) AS TotalSalarii

FROM Angajati

GROUP BY IdDept

Pas 1. Într-o primă fază interogarea

este executată fără a ţine cont

de clauza group by. De fapt se

execută interogarea (proiectia)

SELECT IdDept, Salariu

FROM Angajati

Interogari agregate – Clauza GROUP BY

IdDept Salariu

1 45

2 36

1 40

3 45

4 80

4 73

1 40

2 46

 Caz 2 - Exemple

 Interogare: Să se găsească suma salariilor angajaţilor din fiecare departament.

SELECT IdDept, SUM(Salariu) AS TotalSalarii

FROM Angajati

GROUP BY IdDept

Pas 2. Tabelul rezultat este apoi împărţit

în seturi ce au aceeaşi valoare

pentru atributele listate în clauza GROUP BY

Interogari agregate – Clauza GROUP BY

IdDept Salariu

1 45

1 40

1 40

2 36

2 46

3 45

4 80

4 73

 Caz 2 - Exemple

 Se consideră tabela ANGAJATI (Nume, Prenume, Dept, Birou, Salariu)

 Interogare: Să se găsească suma salariilor angajaţilor din fiecare departament.

SELECT IdDept, SUM(Salariu) AS TotalSalarii

FROM Angajati

GROUP BY IdDept

Pas 3. Odată stabilite grupurile de linii,

funcţia agregat se aplică fiecărui grup

în parte.

Rezultatul final al interogării:

Interogari agregate – Clauza GROUP BY

IdDept Salariu

1 125

2 82

3 45

4 153

IdDept Salariu

1 45

1 40

1 40

2 36

2 46

3 45

4 80

4 73

 Caz 2 – Exemple - mai multe criterii de grupare

 Interogare: Să se găsească suma salariilor angajaţilor din fiecare departament

si fiecare birou.

SELECT IdDept, Birou, SUM(Salariu) AS TotalSalarii

FROM Angajati

GROUP BY IdDept, Birou Rezultat:

Interogari agregate – Clauza GROUP BY

IdDept Birou TotalSa

larii

1 10 85

1 75 40

2 20 82

3 16 45

4 14 80

4 7 73

IdA Nume Prenume IdDept Birou Salariu

1 Ionescu Maria 1 10 45

2 Popescu Ion 2 20 36

3 Popa Stefan 1 10 40

4 Dumitrescu Vasile 3 16 45

5 Ionescu Ion 4 14 80

6 Manole Radu 4 7 73

7 Luca Doru 1 75 40

8 Vasile Alina 2 20 46

Id

De

pt

Birou Salariu

1 10 45

1 10 40

1 75 40

2 20 36

2 20 46

3 16 45

4 14 80

4 7 73

 Caz 2 – clauza HAVING

 Pentru a lua în considerare doar grupurile de linii ce satisfac anumite condiţii

trebuie utilizată clauza HAVING

 Clauza HAVING conţine condiţii ce trebuie aplicate DUPA grupare (dupa

aplicarea clauzei GROUP BY)

 Fiecare subgrup va participa la formarea rezultatului doar dacă satisface

condiţia din clauza having.

 Conditii din clauza HAVING: expresii booleene, formate din predicate

simple şi operatori booleeni.

 Regula de buna practica: in clauza HAVING trebuie sa apară doar predicatele

ce implică o funcţie agregat!

 În standardul SQL se precizează că în HAVING se pot folosi aliasuri de

coloane, definite în clauza SELECT

 În WHERE nu se pot folosi aliasuri de coloane

 În MsAccess NU se pot folosi aliasuri in clauza HAVING

Interogari agregate – Selectia de grupuri – Clauza HAVING

 Caz 2 – clauza HAVING - Exemple

 Interogare: Să se găsească departamentele în care salariul mediu al angajaţilor

este mai mare ca 44

SELECT IdDept, AVG(Salariu) AS MedieSalarii

FROM Angajati

GROUP BY IdDept

HAVING AVG(Salariu)>44

Rezultat:

Interogari agregate – Selectia de grupuri – Clauza HAVING

IdDept MedieSalarii

1 41.66

2 41

3 45

4 76.5

IdA Nume Prenume IdDept Birou Salariu

1 Ionescu Maria 1 10 45

2 Popescu Ion 2 20 36

3 Popa Stefan 1 20 40

4 Dumitrescu Vasile 3 16 45

5 Ionescu Ion 4 14 80

6 Manole Radu 4 7 73

7 Luca Doru 1 75 40

8 Vasile Alina 2 20 46

Id

De

pt

Salariu

1 45

1 40

1 40

2 36

2 46

3 45

4 80

4 73 IdDept MedieSalarii

3 45

4 76.5

 Caz 2 – selectie de randuri INAINTE de grupare

 Dacă exista condiţii ce trebuie verificate la nivel de linie, atunci trebuie

sa se utilizeze predicatele corespunzătoare ca argument al clauzei

WHERE

 Folosind clauzaWHERE se pot selecta randuri INAINTE de

impartirea lor in grupuri

Interogari agregate – Selectia de grupuri – Clauza HAVING

 Caz 2 – clauza HAVING - Exemple

 Interogare: Să se găsească departamentele în care salariul mediu al angajaţilor

din biroul 20 este mai mare ca 25

SELECT IdDept, AVG(Salariu) AS MedieSalarii

FROM Angajati

WHERE Birou = 20

GROUP BY IdDept

Rezultat:

Interogari agregate – Selectia de grupuri – Clauza HAVING

IdA Nume Prenume IdDept Birou Salariu

1 Ionescu Maria 1 10 45

2 Popescu Ion 2 20 36

3 Popa Stefan 1 20 40

4 Dumitrescu Vasile 3 16 45

5 Ionescu Ion 4 14 80

6 Manole Radu 4 7 73

7 Luca Doru 1 75 40

8 Vasile Alina 2 20 46

Id

De

pt

Birou Salariu

2 20 36

1 20 40

2 20 46

IdDept Birou Salariu

1 20 40

2 20 36

2 20 46

IdDept MedieSalarii

1 40

2 41

Ordinea de executie in cadrul unei interogari agregate

 Se realizeaza:

1. Selectia randurilor ce respecta clauza WHERE

2. Gruparea randurilor obtinute, respectand clauza GROUP BY

3. Calcularea rezultatelor functiilor de grup pentru fiecare grup in parte

4. Selectia grupurilor ce respecta clauza HAVING

5. Ordonarea rezultatelor respectand clauza ORDER BY.

 Ordinea de executie are o importanta foarte mare, deoarece are un

impact direct asupra vitezei de executie a interogarii.

 Cu cat mai multe inregistrari pot fi eliminate utilizand clauza

WHERE, cu atat mai putin va dura gruparea si operatiile ce urmeaza

 Se face astfel economie de timp şi memorie

Interogari agregate

Forma completă a unei instrucţiuni SELECT devine

SELECT ListaTinta

FROM ListaTabele

[WHERE Conditie]

[GROUP BY ListaAtributeGrupare]

[HAVING ConditieAgregata]

[ORDER BY ListaAtributeOrdonare]

Instructiunea SELECT

 SQL pune la dispoziţie operatori din teoria mulţimilor, cum ar fi operatorii de

reuniune (union), intersecţie (intersect) şi diferenţă (except sau minus).

 Orice interogare ce utilizează operatorii de intersecţie şi diferenţă poate fi

exprimată cu ajutorul interogărilor imbricate (subinterogari)

 Sintaxa pentru utilizarea operatorilor din teoria mulţimilor este:

InterogareSQL {<UNION | INTERSECT | EXCEPT> [ALL] InterogareSQL}

 IMPORTANT În rezultatele operaţiilor cu mulţimi sunt eliminate (implicit)

duplicatele!

 Dacă se doreşte utilizarea acestor operatori cu menţinerea duplicatelor este suficientă

specificarea opţiunii ALL

 SQL nu impune ca schemele pe care se execută operaţiile să fie identice (spre

deosebire de algebra relaţională), ci doar ca atributele să aibă domenii

compatibile.

 Corespondenţa între atribute nu se bazează pe nume, ci pe poziţia atributelor.

Dacă atributele au nume diferite, rezultatul va prelua numele de atribute din

primul operand.

Interogări cu operatori din teoria mulţimilor

 În MsAccess este implementat doar UNION

Exemple: se consideră tabelul ANGAJATI (Nume, Prenume, Dept, Birou,

Salariu)

Interogare: să se găsească numele şi prenumele tuturor angajaţilor.

SELECT Prenume AS NumeAngajat

FROM Angajati

UNION

SELECT Nume

FROM Angajati

Interogări cu operatori din teoria mulţimilor

IdA Nume Prenume IdDept Birou Salariu

1 Ionescu Maria 1 10 45

2 Popescu Ion 2 20 36

3 Popa Stefan 1 20 40

4 Dumitrescu Vasile 3 16 45

5 Ionescu Ion 4 14 80

6 Manole Radu 4 7 73

7 Luca Doru 1 75 40

8 Vasile Alina 2 20 46

 Exemple: se consideră tabelul ANGAJATI (IdA,Nume, Prenume, IdDept, Birou, Salariu)

Interogare: Să se găsească numele de angajaţi care sunt şi prenume

SELECT Prenume AS NumeAngajat

FROM Angajati

INTERSECT

SELECT Nume

FROM Angajati

Interogări cu operatori din teoria mulţimilor

IdA Nume Prenume IdDept Birou Salariu

1 Ionescu Maria 1 10 45

2 Popescu Ion 2 20 36

3 Popa Stefan 1 20 40

4 Dumitrescu Vasile 3 16 45

5 Ionescu Ion 4 14 80

6 Manole Radu 4 7 73

7 Luca Doru 1 75 40

8 Vasile Alina 2 20 46

 Exemple: se consideră tabelul ANGAJATI (IdA,Nume, Prenume, IdDept, Birou, Salariu)

Interogare: Să se găsească numele de angajaţi care NU sunt şi prenume

SELECT Nume AS NumeAngajat

FROM Angajati

EXCEPT

SELECT Prenume

FROM Angajati

Interogări cu operatori din teoria mulţimilor

IdA Nume Prenume IdDept Birou Salariu

1 Ionescu Maria 1 10 45

2 Popescu Ion 2 20 36

3 Popa Stefan 1 20 40

4 Dumitrescu Vasile 3 16 45

5 Ionescu Ion 4 14 80

6 Manole Radu 4 7 73

7 Luca Doru 1 75 40

8 Vasile Alina 2 20 46

 In cadrul clauzei WHERE se pot defini predicate cu structură complexă, în care o

expresie valorică poate fi comparată cu rezultatul execuţiei unei interogări SQL.

 Interogarea utilizată în comparaţie se numeşte interogare imbricată

(subinterogare), iar interogarea in care se defineste subinterogarea se numeste

interogare externa.

 Pentru a înţelege mecanismul rezolvării interogărilor ce conţin subinterogari se

pleacă de la presupunerea că subinterogarea se execută înaintea analizei liniilor

din interogarea externă

 O subinterogare poate contine clauzele WHERE, GROUP BY, HAVING

 Subinterogarile nu pot contine clauza ORDER BY

 Recomandare: a se evita, daca este posibil, realizarea jonctiunilor in

subinterogare!

Interogări imbricate (Subinterogari)

 Primul operand al unei comparaţii de genul celei amintite anterior este un atribut, în

timp ce în celălalt membru avem o valoare sau o mulţime de valori (rezultatul

subinterogării).

 Sintaxa cere ca domeniul elementelor returnate de interogarea imbricată să fie compatibil

cu atributul cu care se face comparaţia

 Daca rezultatul subinterogarii este o singura valoare, se vor utiliza operatorii

de comparare (<, <=, =, >, >=, <>)

 Daca rezultatul subinterogarii este o multime de valori, SQL pune la dispoziţie

cuvintele cheie ANY, ALL, IN pentru a extinde operatorii de comparaţie

 ANY - linia este validă dacă valoarea atributului se află în relaţie (<,<=,>,<=)

cu cel puţin o valoare returnată de subinterogare

 ALL - linia este validă dacă valoarea atributului se află în relaţie

(<,<=,>,<=,<>) cu toate valorile returnate de subinterogare

 IN - linia este validă dacă valoarea atributului apartine multimii de valori

returnate de subinterogare (echivalent cu =ANY)

 NOT IN - linia este validă dacă valoarea atributului nu apartine multimii de

valori returnate de subinterogare (echivalent cu <>ALL)

Interogări imbricate (Subinterogari)

 < ANY (...) înseamnă mai mic decât maximul

 > ANY (...) înseamnă mai mare decât minimul

 < ALL (...) înseamnă mai mic decât minimul

 <=ALL (...) înseamnă egal cu minimul

 > ALL (...) înseamnă mai mare decât maximul

 > =ALL (...) înseamnă egal cu maximul

Interogări imbricate (Subinterogari)

 Exemplu: se consideră tabelele

ANGAJATI (IdA,Nume, Prenume, IdDept, Birou, Salariu),

DEPARTAMENTE (IdDept, Dept, Adresa, Oras).

Interogare: Să se găsească angajaţii care

lucrează în departamentele din Iaşi.

Obs.

- Necunoscuta: id-ul departamentelor din Iasi => subinterogare!

- Subinterogarea poate returna mai multe valori!

SELECT Nume, Prenume

FROM Angajati

WHERE IdDept IN (SELECT IdDept FROM Departament eWHERE Oras = “Iasi”)

Obs. Această interogare poate fi rezolvată prin realizarea unei joncţiuni între cele

două tabele.

SELECT Nume, Prenume

FROM Angajati AS a INNER JOIN Departamente AS d ON a.IdDept=d.IdDept WHERE

Oras = “Iasi”

Interogări imbricate (Subinterogari)

IdDept Dept Adresa Oras

1

2

3

4

5

Administrati

Productie

Distributie

Planificare

Cercetare

Independent

Primaverii

Central

Nicolina

Trandafirului

Iasi

Bucuresti

Focsani

Iasi

Cluj

Interogare: Să se găsească departamentele în care nu lucrează nici un angajat cu

numele Ionescu

Obs. Subinterogarea poate returna o multime de valori!

Varianta 1

SELECT Dept

FROM Departamente

WHERE IdDept NOT IN (SELECT IdDept FROM Angajati WHERE

Nume=“Ionescu”)

Varianta 2

SELECT IdDept

FROM Departamente

EXCEPT

SELECT IdDept FROM Angajati WHERE Nume=“Ionescu”

Interogări imbricate (Subinterogari)

IdDept Dept Adresa Oras

1

2

3

4

5

Administratie

Productie

Distributie

Planificare

Cercetare

Independentei

Primaverii

Central

Nicolina

Trandafirului

Iasi

Bucuresti

Focsani

Iasi

Cluj

IdA Nume Prenume IdDept Birou Salariu

1 Ionescu Maria 1 10 45

2 Popescu Ion 2 20 36

3 Popa Stefan 1 20 40

4 Dumitrescu Vasile 3 16 45

5 Ionescu Ion 4 14 80

6 Manole Radu 4 7 73

7 Luca Doru 1 75 40

8 Vasile Alina 2 20 46

Dept

Productie

Distributie

Cercetare

 Atentie! Valoarea pe care o comparați cu mulțimea de valori din

subinterogare trebuie sa fie de același tip atât ca și tip de date, dar și semantic

ca să aibă sens.

SELECT Dept

FROM Departamente

WHERE IdDept NOT IN (SELECT IdA FROM Angajati WHERE

Nume=“Ionescu”)

Interogări imbricate (Subinterogari)

Interogare: Să se găsească angajatii din departamentele 3 si 4, cu salariul mai mare decat

salariul unui angajat din departamentele 1 si 2.

Obs. Subinterogarea poate returna o multime de valori!

SELECT Nume, Salariu

FROM Angajati

WHERE Salariu >ANY (SELECT Salariu

FROM Angajati

WHERE IdDept IN (1,2))

AND IdDept IN (3,4)

Interogare: Să se găsească angajatii din departamentele 3 si 4, cu salariul mai mare decat

TOATE salariile angajatilor din departamentele 1 si 2.

SELECT Nume, Salariu

FROM Angajati

WHERE Salariu > ALL (SELECT Salariu

FROM Angajati

WHERE IdDept IN (1,2))

AND IdDept IN (3,4)

Interogări imbricate (Subinterogari)

 funcţiile agregat MAX şi MIN pot fi utilizate în interogările imbricate.

Interogare: Să se găsească angajatii cu cel mai mare salariu din companie.

Necunoscuta: cel mai mare salariu din companie =>subinterogare!

SELECT Nume, Salariu

FROM Angajati

WHERE Salariu = (SELECT MAX(Salariu) FROM Angajati)

Interogări imbricate (Subinterogari)

Interogare: Să se găsească angajatii din departamentele 3 si 4, cu salariul egal

cu salariul maxim din departamentele 1 si 2.

SELECT Nume, Salariu

FROM Angajati

WHERE Salariu > =ALL (SELECT Salariu

FROM Angajati

WHERE IdDept IN (1,2))

AND IdDept IN (3,4)

Obs.

atr >=ALL(multime\_valori) -> atributul este egal cu valoarea maxima

din multimea de valori

atr <=ALL(multime\_valori)-> atributul este egal cu valoarea minima

din multimea de valori

Interogări imbricate (Subinterogari)

 Uneori subinterogarea face referire la contextul interogării externe, în

care este imbricată

 Un astfel de mecanism este cunoscut sub numele de transferul legăturilor

(corelare) dintr-un context în altul

 Acest lucru are loc prin intermediul unui alias definit în interogarea

externă şi utilizată în subinterogare

 În acest caz, noua interpretare pentru subinterogarile corelate este

următoarea: pentru fiecare linie din interogarea externă se evaluează mai

întâi subinterogarea corelata şi apoi se evaluează predicatul din interogarea

externă

 Deci subinterogarea corelata se executa pentru fiecare linie a interogarii

externe!

Subinterogari corelate

Interogare: Să se găsească angajatii din firma cu salariul mai mare decat

salariul mediu din departamentul lor.

SELECT Nume, Salariu

FROM Angajati AS a

WHERE Salariu >(SELECT AVG(Salariu)

FROM Angajati

WHERE IdDept = a.IdDept)

Subinterogari corelate

 Operatorul acceptă ca parametru o interogare imbricată şi returnează valoarea

adevărat doar dacă interogarea nu produce un rezultat vid.

Exemplu: se consideră relaţia PERSOANA (Cod, Nume, Prenume, Oras).

Interogare: Să se găsească persoanele care au acelaşi nume şi prenume, dar coduri

diferite.

SELECT \*

FROM Persoana AS p

WHERE EXISTS (SELECT \*

FROM Persoana AS p1

WHERE p1.Nume = p.Nume

AND p1.Prenume = p.Prenume

AND p1.Cod <> p.Cod )

Subinterogari - Operatorului logic exists

Interogare: Să se găsească persoanele care au acelaşi nume şi prenume, dar coduri diferite.

SELECT \*

FROM Persoana AS p

WHERE EXISTS (SELECT \*

FROM Persoana AS p1

WHERE p1.Nume = p.Nume

AND p1.Prenume = p.Prenume

AND p1.Cod <> p.Cod )

 Subinterogarea nu se poate executa înaintea evaluării interogării externe,

dat fiind că subinterogarea nu este definită până când nu se asignează o

valoare variabilei P.

 Subinterogarea este evaluata pentru fiecare linie produsă în cadrul

interogării externe.

 vor fi examinate mai întâi liniile variabilei P una câte una

 pentru fiecare din aceste linii va fi executată subinterogarea

Subinterogari - Operatorului logic exists

 O altă cale de a formula interogarea din exemplul anterior este prin

folosirea constructorului de tuplu, reprezentat de o pereche de paranteze

rotunde care marchează lista de atribute.

SELECT \*

FROM Persoana AS p

WHERE (Nume, Prenume) IN (SELECT Nume, Prenume

FROM Persoana

WHERE Cod <> p.Cod )

Obs. SGBD comerciale nu rezolvă întotdeauna subinterogarile corelate prin

scanarea tabelului extern şi producerea unei interogări pentru fiecare linie

din relaţie, se încearcă procesarea cât mai multor interogări într-o manieră

orientată pe mulţimi, cu scopul de a manevra cantităţi mari de date prin cât

mai puţine operaţii posibile.

CURS 10-11

SQL

 Sublimbaje SQL

 DDL – Data Definition Language – instructiuni pentru definirea

unei scheme de baze de date

 DML – Data Manipulation Language – instructiuni pentru

introducerea/actualizarea/stergerea datelor dintr-o baza de date

 DCL – Data Control Language – instructiuni pentru controlul

accesului la date

SQL - DDL

 permite specificarea informațiilor despre relații

 Schema fiecărei relații

 Domeniul de valori asociat fiecărui atribut

 Constrângeri de integritate (primitive)

SQL - DDL

 SQL permite definirea unei scheme de baze de date ca o colecţie de

obiecte.

 Fiecare schemă conţine o mulţime de domenii, tabele, indici, vederi şi

privilegii şi este definită cu ajutorul următoarei sintaxe:

create schema [NumeSchemă] [[authorization] Autorizare]

{DefiniţieElementeDinSchemă}

 Autorizare - numele utilizatorului proprietar al schemei - dacă este omis se

consideră că utilizatorul care a executat comanda este proprietarul schemei.

 Dacă NumeSchemă este omis va fi adoptat ca nume al schemei numele

utilizatorului ce a executat comanda.

 După comanda create schema se pot defini obiectele din schema

respectivă.

DDL - Definirea tabelelor

 Un tabel SQL este format dintr-o mulţime ordonată de atribute şi o

mulţime, posibil vidă, de constrângeri.

CREATE TABLE NumeTabel

(NumeAtribut Domeniu [ValoareImplicită] [Constrângeri] {,

NumeAtribut Domeniu [ValoareImplicită] [Constrângeri]}

[,AlteConstrângeri])

 După ce au fost definite toate atributele, se pot defini alte constrângeri ce

implică mai multe atribute.

 Iniţial tabelul nu conţine înregistrări, proprietarul deţinând toate

privilegiile asupra tabelului, adică drepturi de a accesa şi de a modifica

datele din tabel.

 Exemplu (MS Access)

CREATE TABLE Departamente (IdDept AUTOINCREMENT PRIMARY KEY,

NumeDept TEXT(20) UNIQUE, Adresa TEXT(50), Oraş TEXT(20),Buget

SINGLE)

DDL - Definirea tabelelor

 Definire atribut:

Sintaxa: Tip\_date DEFAULT <ValoareImplicita | NULL>

 Domeniu atribut

 Tipuri de date in MS Access: Autoincrement, Byte, Smallint (Integer), Long,

Single, Double, Number (implicit, tipul de date va fi Double), Currency,

Text, DateTime, YesNo, OleObject

 Termenul ValoareImplicită indică valoarea ce va fi considerată pentru atributul

asociat în cazul inserării unei linii ce nu specifică o valoare pentru acel atribut.

 Dacă este omisă specificarea unei valori implicite, atunci se va utiliza valoarea NULL

ca valoare implicită.

 ValoareImplicita este o valoare compatibilă cu domeniul asociat;

Exemplu (MS Access)

NumărCopii SMALLINT DEFAULT 0 – dacă se inserează o linie şi nu se specifică

valoarea pentru acest atribut, atunci acestui atribut i se va atribui valoarea 0.

DDL - Definirea tabelelor – SELECT..INTO

Instructiunea SELECT...INTO

 Se defineste un tabel cu structura si datele corespunzatoare listei de atribute din lista de

tabele specificate in clauza FROM

 Atentie: constrangerile de integritate (in afara constrangerii de existenta – NOT

NULL) nu sunt mostenite de noul tabel!

 Sintaxa:

SELECT \*/lista\_coloane INTO nume\_tabel [IN

baza\_date\_externa]

FROM lista\_tabele

[WHERE ...]

[GROUP BY ...]

[HAVING...]

[ORDER BY...]

 Exemplu (MS Access)

SELECT IdDept, NumeDept, Oras INTO DepartamenteDobrogea

FROM Departamente

WHERE Oras IN (“Constanta”, “Mangalia”, “Tulcea”)

DDL - Constrângeri intra-relaţionale

 Constrângerile sunt proprietăţi ce trebuie verificate de fiecare instanţă

a bazei de date

 constrângeri intra-relaţionale (implică o singură relaţie)

 NOT NULL

 UNIQUE

 PRIMARY KEY

 CHECK

 constrângeri inter-relaţionale (iau în considerare mai multe relaţii)

 FOREIGN KEY...REFERENCES

DDL - Constrângeri intra-relaţionale

Not Null

 indică faptul că valoarea NULL nu este admisă ca valoare pentru

atributul afectat de constrângere.

 În acest caz valoarea atributului trebuie să fie specificată la inserare.

 Este posibilă inserarea unei linii fără a specifica valoarea unui atribut

cu constrângere not NULL doar dacă pentru atributul respectiv s-a

definit o valoare implicită diferită de valoarea NULL.

 Specificarea acestei constrângeri se face prin adăugarea cuvintelor

cheie NOT NULL la definirea atributului.

 Exemplu (MS Access)

Nume TEXT(20) NOT NULL

DDL - Constrângeri intra-relaţionale

Unique

 impune ca un atribut sau o mulţime de atribute să formeze o cheie semantica

 se impune astfel ca linii diferite să nu conţină aceleaşi valori.

 Excepţie face valoarea NULL, care poate apărea în diverse linii fără a încălca

constrângerea.

 Deoarce fiecare valoare NULL reprezintă o valoare necunoscută diferită de a

altei valori NULL.

 Există două moduri de definire a acestei constrângeri.

 constrângerea implică un singur atribut şi constă în adăugarea cuvântului cheie

UNIQUE la definirea atributului.

 Exemplu (MS Access): NrMatricol TEXT(6) UNIQUE

 constrângerea implică mai multe atribute şi constă în utilizarea clauzei

CONSTRAINT numeconstrangere UNIQUE (Atribut{, Atribut}) după

definirea atributelor.

 Exemplu (MS Access): CREATE TABLE Departamente (IdDept AUTOINCREMENT

PRIMARY KEY, NumeDept TEXT(20) NOT NULL, Oras TEXT(20),Buget SINGLe,

CONSTRAINT cs1 UNIQUE(NumeDept, Oras))

DDL - Constrângeri intra-relaţionale

Primary Key

 Poate fi specificată o singură dată pentru un tabel şi poate fi definită pe un singur

atribut (cheia primara surogat) sau pe o mulţime de atribute (nerecomandat!)

 Definiţia unei astfel de constrângeri implică definirea implicită a unor

constrângeri NOT NULL pentru atributul (atributele) cheii primare.

 Exemplu

 MS Access

CREATE TABLE Departamente (IdDept AUTOINCREMENT PRIMARY KEY,

NumeDept TEXT(20) NOT NULL UNIQUE, Oras TEXT(20))

 MS SQL Server

CREATE TABLE Departamente (IdDept IDENTITY(1,1) PRIMARY KEY, NumeDept

CHAR(20) NOT NULL UNIQUE, Oras CHAR(20))

 Oracle

CREATE TABLE Departamente (IdDept Number GENERATED ALWAYS AS

IDENTITY, NumeDept CHAR(20) NOT NULL UNIQUE, Oras CHAR(20), PRIMARY

KEY(IdDept))

DDL - Constrângeri intra-relaţionale

CHECK

 Pot fi specificate constrângerile pe tuplu, deoarece condiţia din constrângerea

CHECKpoate face referire la alte atribute

 Sintaxă:

CHECK (Condiţie)

 Condiţiile ce pot fi utilizate sunt cele ce pot apărea în clauza WHERE a unei

interogări SQL.

 Condiţia impusă trebuie verificată întotdeauna pentru a menţine corectitudinea

bazei de date.

 Exemplu (Oracle)

CREATE TABLE Angajati

(IdA Number Primary Key, Nume VARCAHR2(20) NOT NULL, Salariu Number,

CONSTRAINT cksalang CHECK (Salariu>=0))

DDL - Constrângeri inter-relaţionale

 Cele mai importante constrângeri inter-relaţionale sunt constrângerile de

integritate referenţială (de chei straine)

 Constrângerea de tip FOREIGN KEY impune ca pentru fiecare linie dintr-

un tabel (numit tabel intern), valoarea corespunzătoare atributului cheie

straina, diferită de NULL, să se regăsească printre valorile cheii primare a

unui alt tabel (numit tabel extern sau referit)

 Domeniul cheii straine trebuie sa fie acelasi cu cel al cheii primare!

 In MS Access, domeniul cheii primare este Long!

 Constrângerile de referinţă pot fi definite în două moduri

1. Cheie primara surogat: se utilizează construcţia sintactică

REFERENCES, care indică tabelul extern şi atributul asociat.

 Exemplu (MS Access)

CREATE TABLE Angajati(IdA AUTOINCREMENT PRIMARY KEY,

Nume TEXT(20) NOT NULL, CNP TEXT(13) NOT NULL UNIQUE,

Dept LONG REFERENCES Departament (IdDept), Salariu NUMBER

DEFAULT 0)

DDL - Constrângeri inter-relaţionale

2. Cheie primara compusa (nerecomandat!): se va utiliza construcţia

CONSTRAINT numeconstrangere FOREIGN KEY,

plasată după definirea tuturor atributelor

 Exemplu (MS Access). Pp. ca cheia primare in DEPARTAMENT este

(NumeDept, Oras)

CREATE TABLE Angajati(IdA AUTOINCREMENT PRIMARY KEY,

Nume TEXT(20) NOT NULL, CNP TEXT(13) NOT NULL UNIQUE,

Dept TEXT(20), DeptOras TEXT(20), Salariu NUMBER DEFAULT 0,

CONSTRAINT fk1 FOREIGN KEY (Dept,DeptOras) REFERENCES

Departament(NumeDept,Oras))

DDL - Constrângeri inter-relaţionale

 În cazul constrângerilor discutate până acum sistemul va rejecta (generând

un mesaj de eroare) orice operaţie de actualizare ce violează constrângerea.

 Pentru constrângerea de referinţă (FOREIGN KEY), SQL permite

utilizatorului selecţia acţiunii ce va fi executată în cazul violării constrângerii.

 Exemplu. Să considerăm constrângerea de tip cheie straina asupra

atributului Dept în tabelul ANGAJATI.

 Constrângerea poate fi violată operând atât asupra liniilor din tabelul

internANGAJATI cât şi asupra liniilor din tabelul extern

DEPARTAMENT.

 Modificarea conţinutului tabelului intern (cel cu cheia straina)

 Doua cazuri:

o inserarea unei linii invalide;

o modificarea atributului Dept.

 In aceste cazuri nu este oferit un suport special, operaţiile fiind pur şi

simplu rejectate.

DDL - Constrângeri inter-relaţionale

 Modificarea conţinutului tabelului extern

 se oferă opţiuni variate de reacţie la încălcarea constrângerii de

referinţă

 Doua cazuri:

o Modificarea (actualizarea) valorilor atributelor referite (cheilor

primare)

o stergerea de înregistrări

 Tipul reacţiei la astfel de încălcări ale constrângerii diferă în funcţie

de comanda ce a generat violarea constrângerii

DDL - Constrângeri inter-relaţionale

 În cazul operaţiilor de actualizare, reactiile posibile sunt:

 CASCADE: noua valoare a atributului din tabelul extern va fi atribuită

tuturor liniilor corespunzătoare din tabelul intern

 SET NULL: valoarea NULL este asignată cheii straine din tabelul intern în

locul valorii modificate din tabelul extern;

 SET DEFAULT: cheii straine, din tabelul intern, îi va fi asignată valoarea

implicită în locul valorii modificate în tabelul extern

 NO ACTION: operaţia de actualizare este rejectată

 În cazul operaţiilor de stergere, reactiile posibile sunt:

 CASCADE: vor fi şterse toate liniile din tabelul intern corespunzătoare liniei

şterse din tabelul extern (NERECOMANDAT!)

 SET NULL: se asignează valoarea NULL cheii straine din tabelul intern în

locul valorii şterse din tabelul extern;

 SET DEFAULT: cheia straina din tabelul intern va primi valoarea implicită în

locul valorii şterse din tabelul extern;

 NO ACTION: operaţia de ştergere este rejectată

DDL - Constrângeri inter-relaţionale

 Specificarea modului de reacţie în cazul violării unei constrângeri de referinţă se

face imediat după definirea constrângerii prin sintaxa:

ON <DELETE | UPDATE>

<CASCADE | SET NULL | SET DEFAULT | NO ACTION>

Exemplu (MS Access)

CREATE TABLE Angajati(IdA AUTOINCREMENT PRIMARY KEY,

Nume TEXT(20) NOT NULL, CNP TEXT(13) NOT NULL UNIQUE,

Dept LONG REFERENCES Departament (IdDept) ON UPDATE

CASCADE,Salariu NUMBER DEFAULT 0)

DDL - Actualizarea schemei unei relaţii

 Comenzi pentru manipularea schemelor unei BD: ALTER,DROP

 Comanda ALTER - permite modificarea domeniilor şi schemelor de tabele

 adăugare / eliminare constrângeri

 În momentul definirii unei noi constrângeri, datele din tabel trebuie să

satisfacă acea constrângere. În caz contrar definiţia constrângerii nu va fi

validată.

 modificare valoare implicită

 adăugare / eliminare atribute din schema unui tabel.

ALTER TABLE NumeTabel

<ALTER COLUMN NumeAtribut

<SET DEFAULT ValoareImplicită | DROP DEFAULT> |

ADD CONSTRAINT NumeConstrangere constrângere |

DROP CONSTRAINT NumeConstrângere |

ADD COLUMN DefiniţieAtribut |

DROP COLUMN NumeAtribut>

DDL - Actualizarea schemei unei relaţii

 Exemplu (MS Access)

ALTER TABLE Departamente ALTER COLUMN NumeDept TEXT(40)

ALTER TABLE Departamente ADD COLUMN Telefon TEXT(10)

Impunerea constrangerii NOT NULL pe atributul Telefon:

ALTER table Departamente ATLER COLUMN Telefon TEXT(10) NOT NULL

Impunerea constrangerii de cheie semantica pe atributul Telefon:

ALTER TABLE Departamente ADD CONSTRAINT tel\_unic UNIQUE(Telefon)

ALTER TABLE Departamente DROP COLUMN Telefon

DDL – Comanda DROP

 Permite eliminarea datelor de tip schemă, tabel, vedere

 Sintaxa este:

DROP <schema|domain|table|view> NumeComponentă

 schemă nu va fi eliminată dacă ea conţine tabele sau alte componente;

 stergerea unui tabel folosind această opţiune implică ştergerea schemei si a

tuturor liniilor din tabel

 Exemplu (MS Access)

DROP TABLE Departament

DDL– Comanda TRUNCATE

 Permite eliminarea datelor dintr-un tabel

Sintaxa este:

TRUNCATE TABLE NumeTabel

 trunchierea unui tabel implică ştergerea tuturor liniilor din tabel (schema

tabelului NU se modifica)

 Exemplu (Oracle)

TRUNCATE TABLE Departament

Vederi

 Tipuri de relaţii într-o bază de date relaţională

 relaţii de bază - conţinutul este autonom şi stocat în baza de date

 relaţii derivate - conţinutul este definit pe baza conţinutului altor relaţii

 Există două tipuri principale de relaţii derivate:

 vederi materiale - relaţii derivate stocate în baza de date;

 vederi (relaţii virtuale) – relaţii definite prin intermediul unor funcţii (expresii

ale limbajului de interogare) care nu sunt stocate în baza de date, dar pot fi

folosite în interogări ca şi cum ar exista fizic

 Vederile materiale oferă un avantaj când numărul cererilor de interogare (citire

date) este mai mare decât operaţiile de actualizare (scriere date) ale relaţiei pe care

se bazează vederea.

 Deoarece nu se pot specifica tehnici generale de păstrare a consistenţei între

relaţiile de bază şi vederile materiale, majoritatea sistemelor comerciale oferă

mecanisme numai pentru relaţiile virtuale (vederi).

 Vederile sunt definite în sistemele relaţionale ca fiind expresii ale unui limbaj de

interogare

 Un utilizator interesat numai de o porţiune din baza de date poate evita contactul

cu componentele ce nu prezintă interes => securitatea datelor!

Vederi (relatii virtuale)

 Tabele „virtuale” al căror conţinut depinde de conţinutul altor tabele din baza

de date.

 În SQL vederile sunt definite prin asocierea unui nume şi a unei liste de

atribute,cu rezultatul execuţiei unei interogări.

 Sintaxa:

CREATE VIEW NumeVedere [(ListaAtribute)] AS InterogareSQL

[WITH [<LOCAL | CASCADED>] CHECK OPTION]

 Interogarea SQL şi schema vederii trebuie să aibă acelaşi număr de atribute

 Exemplu: Să se definească vederea AngajatiAdmin care va conţine toţi angajaţii

din departamentul Administraţie care au salariul mai mare decat 10.

CREATE VIEW AngajatiAdmin (NrInreg, Nume, Prenume, Salariu) AS

SELECT IdA, Nume, Prenume, Salariu

FROM Angajati AS a INNER JOIN Departament AS d ON a.IdDept=d.IdDept

WHERE DEPT = “Administratie” and Salariu > 10

Vederi

 În cazul anumitor vederi se pot efectua operaţii de actualizare, dar aceste

operaţii trebuie translate în instrucţiuni de modificare a tabelelor ce stau la

baza vederii.

 Nu întotdeauna se pot găsi soluţii de modificare a tabelelor de bază, mai ales în

situaţiile în care vederea se defineşte pe baza unei joncţiuni între mai multe

tabele.

 În general sistemele comerciale permit modificarea unei vederi doar dacă este

definită pe un singur tabel;alte sisteme cer ca atributele vederii să conţină

măcar o cheie primară a tabelului de bază.

 Clauza CHECK OPTION specifică faptul că operaţiile de actualizare se pot

face doar daca noile valori respecta conditiile clauzei WHERE

 Când o vedere este definită pe baza altor vederi, opţiunile LOCAL sau

CASCADED specifică dacă actualizarea/ştergerea liniilor se face la nivel local

sau trebuie propagată la toate vederile de care depinde vederea în cauză.

 Opţiunea implicită este CASCADED.

Vederi

 Exemplu: Să se definească vederea AngajatiAdmin1, bazată pe vederea

AngajatiAdmin, care va conţine toţi angajaţii din departamentul

Administraţie care au salariul între 10 şi 50.

CREATE VIEW AngajatiAdmin1 AS

SELECT \*

FROM AngajatiAdmin

WHERE Salariu < 50

WITH CHECK OPTION

 Încercarea de a da valoarea 8 atributului Salariu nu va fi acceptată de

definiţia curentă a vederii, dar ar fi fost validată dacă check option ar fi fost

definită ca local.

 Încercarea de a modifica valoarea atributului Salariu pentru o linie din

vedere la valoarea 60 nu ar fi validată nici cu opţiunea local.

Vederi

 Vederile pot fi utilizate în SQL pentru formularea unor interogări care altfel ar fi dificil

(uneori, imposibil) de exprimat.

 În general, vederile pot fi considerate ca fiind unelte ce măresc posibilitatea creării

subinterogarilor

 Exemplu: Să se găsească departamentele cu cel mai mare buget alocat salariilor.

Pas 1: CREATE VIEW BugetSalarii (Dept, SalariuTotal) AS

SELECT Dept, sum(Salariu)

FROM Angajati AS a INNER JOIN Departament AS d on a.IdDept=d.IdDept

GROUP BY Dept

Pas 2: SELECT Dept, SalariuTotal

FROM BugetSalarii

WHERE SalariuTotal = (SELECT MAX(SalariuTotal) FROM BugetSalarii)

Obs. Vederea BugetSalarii poate fi reutilizata la constructia mai multor interogari

Subinterogare: SELECT Dept, Sum(Salariu) as TotalSalarii

FROM Angajati AS a INNER JOIN Departament AS d on a.IdDept=d.IdDept

GROUP BY Dept

HAVING Sum(Salariu)>= ALL(SELECT Sum(Salariu) FROM Angajati GROUP BY IdDept)

DML - Modificarea datelor in SQL

 Pentru modificarea conţinutului unei baze de date SQL-DML pune la

dispoziţie instrucţiunile INSERT, DELETE ŞI UPDATE

Inserări în baza de date

 Sintaxa instrucţiunii INSERT este:

INSERT INTO NumeTabel [(ListaAtribute)] VALUES (ListaVal)

INSERT INTO NumeTabel [(ListaAtribute)] (Subinterogare)

 Prima variantă permite adaugarea unei singure linii în tabele.

 Argumentul clauzei VALUES reprezintă valorile atributelor pentru linia inserată.

 Exemplu: INSERT INTO Departament (NumeDept, Oras) VALUES (’Producţie’,

’Suceava’)

 A doua variantă permite adăugarea unei mulţimi de linii, care sunt extrase mai întâi din baza

de date

 Exemplu: INSERT INTO DepartamenteIasi (SELECT NumeDept, Adresa

FROM Departament WHERE Oras = ’Iasi’)

 Comanda anterioară inserează în tabelul DepartamenteIasi liniile din tabelul

Departament care au Iasi ca valoare a atributului Oras.

DML - Inserări în baza de date

 Dacă valorile pentru anumite atribute nu sunt specificate în momentul

inserării, se vor asigna valori implicite sau valori NULL în cazul în care nu

sunt definite valori implicite.

 Corespondenţa între atributele tabelului şi valorile ce urmează a fi inserate

este dictată de ordinea în care termenii apar în definiţia tabelului.

 primul element din ListaValori (în cazul primei variante) sau primul

element din lista ţintă (în cazul celei de-a doua variante) trebuie să

corespundă primului element ce apare în ListaAtribute (sau în definiţia

tabelului dacă ListaAtribute este omisă) şi aşa mai departe pentru

celelalte atribute

 In cazul adaugarii unei multimi de linii, daca tabelul T in care se introduc

liniile se afla intr-o alta baza de date in raport cu tabelul din care se preiau

liniile, trebuie specificata baza de date a tabelului T (bd\_externa) .

INSERT INTO NumeTabel [(ListaAtribute)] IN

BD\_externa(Subinterogare)

Obs. Interogarea se va rula in tabelul din care se preiau liniile (tabel intern)

DML - Inserări în baza de date

 Exemplu: adaugarea unui departament

INSERT INTO Departament (NumeDept, Adresa, Oras) VALUES

(’Producţie’, ‘Str. ABC’, ’Suceava’)

Sau

INSERT INTO Departament VALUES (’Producţie’, ‘Str. ABC’, ’Suceava’)

 Exemplu: adaugarea unui departament cu adresa null

INSERT INTO Departament (NumeDept, Adresa, Oras) VALUES

(’Producţie’, null, ’Suceava’)

Sau

INSERT INTO Departament (NumeDept, Oras) VALUES (’Producţie’,

’Suceava’)

DML - Ştergerea înregistrărilor

 Sintaxa instrucţiunii DELETE este:

DELETE FROM NumeTabel [WHERE Conditie]

 ATENTIE: Dacă NU se specifică clauza WHERE se vor şterge toate

înregistrările din tabel.

 În cazul în care instrucţiunea DELETE conţine clauza WHERE se vor şterge

doar acele înregistrări ce satisfac condiţia precizată.

 Condiţia poate conţine şi subinterogari ce fac referire la conţinutul altor

tabele.

 Exemplu: şterge departamentele fără angajaţi

DELETE FROM Departamente WHERE IdDept NOT IN (SELECT Dept FROM

Angajati)

DML - Ştergerea înregistrărilor

 Exemplu: stergeti toți angajatii care au salariul mai mic decât salariul mediu

al angajatilor

DELETE FROM Angajati

WHERE Salariu < (SELECT AVG(salariu) FROM Angajati)

 Obs: la fiecare înregistrare ștearsă, salariul mediu se recalculează și nu mai

are aceeași valoare

 Soluție:

Q1:

SELECT AVG(salariu) AS SalariuMediu

FROM Angajati

Apoi Q2:

DELETE FROM Angajati

WHERE Salariu < (SELECT salariumediu FROM Q1)

DML - Ştergerea înregistrărilor

 Obs. diferenţa dintre comanda DELETE (DML) şi comanda DROP (DDL)

 O comandă de genul DELETE FROM Departament şterge toate liniile din

tabelul DEPARTAMENT; schema tabelului Departament rămâne

neschimbată (tabelul ramane in bd, fara niciun rand)

 Comanda DROP TABLE Departament sterge tabelul Departament (schema

si date)

DML – Actualizarea înregistrărilor

 Sintaxa instrucţiunii UPDATE este

UPDATE NumeTabel

SET Atribut = <Expresie | subinterogare | NULL |

DEFAULT>

{, Atribut = <Expresie | subinterogare | NULL |

DEFAULT>}

[WHERE Conditie]

 Instrucţiunea UPDATE face posibilă actualizarea unuia sau a mai multor atribute din

liniile tabelului NumeTabel ce satisfac o posibilă Conditie.

 Noua valoare ce va fi asignată unui atribut poate fi:

 rezultatul evaluării unei expresii, definită pe atributele din tabel;

 rezultatul unei subinterogări

 valoarea NULL;

 valoarea implicită a domeniului de definiţie

 Condiție poate conține subinterogări

DML - Actualizarea / stergerea înregistrărilor

 Atentie: In instructiunile UPDATE/DELETE, dacă NU apare clauza

WHERE se vor actualiza, respectiv, sterge, toate liniile din tabel.

 Condiții de filtrare a datelor

 Expresie operator\_relational expresie (ex. age > 20, grupa != 711)

 Expresie [NOT] BETWEEN min AND max (ex. age BETWEEN 18 AND

30)

 Expresie[NOT] LIKE 'sablon' ("%" –orice număr de caractere, "\_" – un

caracter) Ex. nume LIKE 'P%‘

 Expresie IS [NOT] NULL (ex. email IS NOT NULL)

 Expresie[NOT] IN (valoare[, valoare] ...) (ex. age NOT IN (64, 65) )

 Expresie[NOT] IN (subselectie)

 Expresie operator\_relational {ALL | ANY} (subselectie)

 [NOT] EXISTS (subselectie)

 Condițiile de filtrare pot fi elementare (descrise mai sus) sau compuse cu

operatorii logici NOT, AND, OR și paranteze

DML - Actualizarea înregistrărilor

 Exemplu

UPDATE Angajati SET Salariu = Salariu \* 1.2

WHERE IdDept = 1

 produce o creştere cu 20% a salariilor angajaţilor din departamentul cu id

egal cu 1

Jonctiune:

UPDATE Angajati AS a,Departamente AS d

SET Salariu=Salariu\*1.2

WHERE a.Dept=d.IdDept AND NumeDept=”Productie”

 produce o creştere cu 20% a salariilor angajaţilor din departamentul

Productie

 Observati jonctiunea relatiilor Angajati si Departamente

DML - Actualizarea înregistrărilor

 Exemplu: se doreşte modificarea salariilor angajaţilor astfel:

 creşterea salariilor sub 30 mii cu 10% şi a salariilor peste 30 mii cu 15%.

 o cale de a face actualizarea este execuţia următoarelor comenzi:

UPDATE Angajati SET Salariu = Salariu \* 1.1

WHERE Salariu <= 30

UPDATE Angajati SET Salariu = Salariu \* 1.15

WHERE Salariu > 30

 Atenție! Ordinea în care se vor efectua măririle este importantă!

Pp. că avem un angajat care câştigă 28 mii =>satisface condiţia din prima comanda de

actualizare =>Salariu va fi setat la 30.8 mii. În acest moment, linia satisface de

asemenea şi condiţia celei de-a doua actualizări =>salariul va fi modificat din nou =>

creşterea pentru salariatul respectiv va fi de 26.5%

 Această problemă particulară poate fi rezolvată prin inversarea celor două operaţii de

actualizare.

 În situaţiile mai complexe soluţia ar putea să necesite introducerea unor actualizări

intermediare sau utilizarea unui limbaj de programare de înalt nivel care foloseşte

cursori (Oracle - PL/SQL)

DML - Actualizarea înregistrărilor

 Exemplu: setati salariul angajatilor din departamentul Productie la 5 de ori

valoarea salariului mediu din firma

UPDATE Angajati SET Salariu = (SELECT 5\*AVG(Salariu) FROM Angajati)

WHERE IdDept=(SELECT IdDept FROM Departamente

WHERE NumeDept =“Productie”)

 Setaţi la 200.000 bugetul departamentelor care au nr maxim de angajati.

UPDATE Departamente SET Buget = 200000

WHERE IdDept IN

(SELECT Dept

FROM Angajati

GROUP BY Dept

HAVING COUNT(IdA) >=ALL (SELECT COUNT(IdA)

FROM Angajati

GROUP BY Dept))

SQL -DCL

 DCL – Data Control Language – instructiuni pentru controlul

accesului la date

 SQL a fost proiectat astfel încât fiecare utilizator poate fi

identificat în două moduri:

 utilizator al sistemului de operare;

 utilizator al bazei de date.

Resurse şi privilegii

 Resurse protejate de sistem: tabele, atributele unor tabele, vederi

 Regulă generală: utilizatorul care creează o resursă este

proprietarul ei şi este autorizat să efectueze orice operaţie asupra

acelei resurse.

 SQL pune la dispoziţie mecanisme de organizare ce permit

administratorului să specifice acele resurse la care utilizatorii au

acces şi cele la care nu au acces => utilizatorii dispun de privilegii

de acces la resursele sistemului.

 Fiecare privilegiu de acces este caracterizat de:

 resursa la care face referire;

 utilizatorul ce acordă privilegiul;

 utilizatorul ce primeşte privilegiul;

 operaţia permisă asupra resursei;

 posibilitatea acordării privilegiului altor utilizatori

Resurse şi privilegii

 În momentul creării unei resurse, sistemul acordă, în mod automat,

toate privilegiile asupra resursei creatorului său.

 Există un utilizator predefinit, \_system, asociat administratorului bazei

de date,ce deţine toate privilegiile asupra tuturor resurselor.

 Tipurile de privilegii disponibile sunt:

 1)insert – permite inserarea unui obiect nou în resursă (aplicabil numai

tabelelor şi vederilor)

 2) update – permite modificarea valorii unui obiect (poate fi utilizat cu

tabele, vederi şi atribute);

 3) delete – permite eliminarea unui obiect din resursă (doar tabele sau

vederi);

 4)select – permite utilizatorului să citească resursa cu scopul de a o

utiliza în interogări (tabele, vederi sau atribute);

Resurse şi privilegii

 5) references – permite crearea unei referinţe către o resursă în

contextul definirii unui tabel. Poate fi asociat cu tabele sau atribute.

 Acordarea acestui privilegiu asupra unei resurse poate conduce la limitarea

posibilităţii de modificare a resursei.

 Ex. Să considerăm că utilizatorul Paul este proprietarul tabelului

DEPARTAMENT, iar utilizatorul Ştefan deţine privilegiul de referinţă. Ştefan are

posibilitatea să definească o constrângere de tip foreign key pe tabelul său

ANGAJATI, referind resursa indicată de privilegiu (de exemplu cheia tabelului

DEPARTAMENT). Dacă Ştefan adoptă o politică no action la definirea

constrângerii, Paul va fi pus în situaţia de a nu putea şterge sau modifica linii din

tabelul său dacă aceste operaţii au ca efect încălcarea constrângerii.

 6)usage – se aplică domeniilor şi permite utilizarea lor, spre exemplu,

pentru definirea schemei unui tabel.

 Privilegiul de a efectua operaţiile de drop sau alter nu poate fi acordat.

Acest tip de privilegiu este deţinut doar de proprietarul resursei.

 Privilegiile se acordă sau se revocă cu ajutorul instrucţiunilor GRANT şi

REVOKE.

SQL - DCL

 Sintaxa comenzii de acordare de privilegii este:

GRANT Privilegii ON Resursă TO Utilizatori [WITH GRANT OPTION]

 Această instrucţiune permite acordarea de Privilegii asupra Resursei

către Utilizatori.

 Exemplu

GRANT select ON Departament TO Stefan

 Clauza WITH GRANT OPTION indică posibilitatea propagării

privilegiului către alţi utilizatori.

 Se pot utiliza cuvintele cheie ALL PRIVILEGES pentru acordarea

tuturor privilegiilor.

 Exemplu

GRANT ALL PRIVILEGES ON Departament to Stefan, Paul

SQL - DCL

 Sintaxa comenzii de revocare de privilegii este:

REVOKE Privilegii ON Resursă FROM Utilizatori [RESTRICT |

CASCADE]

 Printre privilegiile ce pot fi revocate unui utilizator se găseşte şi privilegiul

GRANT OPTION, ce derivă din utilizarea opţiunii WITH GRANT

OPTION.

 Revocarea privilegiilor poate fi făcută doar de utilizatorul care, într-o

primă fază, a acordat aceste privilegii.

 Opţiunea RESTRICT împiedică execuţia instrucţiunii REVOKE dacă

retragerea privilegiului are ca efect o retragere în lanţ de privilegii.

 O astfel de comportare poate apărea în situaţia în care utilizatorul a

primit privilegiul cu opţiunea WITH GRANT OPTION şi a propagat

privilegiul către alţi utilizatori.

 Opţiunea CASCADE va avea ca rezultat revocarea tuturor privilegiilor din

lanţ şi în plus va elimina toate obiectele din baza de date ce au fost

construite pe baza acestor privilegii.

SQL - Exemple

 Fie urmatoarele relatii

IdS NrMatricol Nume Email Varsta Grupa CNP

1 111111 Popa Maria maria@fmi.ro 20 1 6200513014576

2 222222 Ionescu Ioana ioana@fmi.ro 23 2 6200513011723

3 333333 Moraru Andrei andrei@fmi.ro 25 2 5200513013724

4 444444 Dumitru Anca anca@fmi.ro 21 3 6200513015667

STUDENTI

IdC Titlu Credite

1 Baze de date 5

2 SGBD 6

3 POO 4

4 Programare Web 5

CURSURI IdEx IdS IdC Nota

1 1 1 10

2 1 2 8

3 2 1 9

4 3 3 10

EXAMENE

SQL - Exercitii

 Găsiți toți studenții din grupele 1 si 2

SELECT Nume, NrMatricol

FROM Studenti

WHERE Grupa IN (1,2)

 Găsiți toți studenții a căror nume începe cu

“Po”

SELECT Nume, NrMatricol

FROM Studenti

WHERE Nume LIKE “Po\*”

 Găsiți toți studenții mai în vârstă decât orice alt student din grupa 2

SELECT Nume, NrMatricol, Varsta

FROM Studenti

WHERE Varsta > ALL (SELECT Varsta

FROM Studenti

WHERE Grupa = 2)

SQL - Exemple

 Găsiți email-urile studenților, notele și cursurile la care au luat studenții aceste note

SELECT Nume, Email, Titlu, Nota

FROM (Studenti as S INNER JOIN Examene as E ON S.IdS = E.IdS) INNER JOIN

Cursuri C ON C.IdC=E.IdC

 Găsiți email-urile studenților înscriși la cursul de Baze de date

Varianta 1

SELECT Nume, Email

FROM (Studenti as S INNER JOIN Examene as E ON S.IdS = E.IdS) INNER JOIN

Cursuri C ON C.IdC=E.IdC

WHERE Titlu = “Baze de date”

Varianta 2

SELECT Nume, Email

FROM Studenti

WHERE IdS IN (SELECT IdS FROM Examene as E INNER JOIN Cursuri as C on

C.IdC=E.IdC WHERE Titlu = “Baze de date”)

SQL - Exemple

 Găsiți numărul studenților din grupa 2 al căror nume începe cu litera “I”

SELECT Count(IdS) as NrStudenti

FROM Studenti

WHERE Grupa =2 and Nume Like “I\*”

 Pentru fiecare student înscris la cel puțin două cursuri afișați numărul de cursuri la care el e

înscris

SELECT NrMatricol, COUNT(IdE) AS NrCursuri

FROM Examene as E INNER JOIN Studenti as S ON E.IdS = S.IdS

GROUP BY NrMatricol

HAVING COUNT(IdE) >= 2

 Găsiți cea mai mică vârsta peste 20 ani din fiecare grupă de studenți

SELECT Grupa, MIN(Varsta) as VarstaMin

FROM Studenti

WHERE varsta >= 20

GROUP BY Grupa

SQL - Exemple

 Găsiți pentru fiecare curs cu mai mult de 4 credite numărul de studenți înscriși la acel curs -

doar pentru cursurile cu cel putin 2 studenti

SELECT Titlu, COUNT(IdS) AS NrStudenti

FROM Examene as E INNER JOIN Cursuri as C ON E.IdC = C.IdC

WHERE Credite >4

GROUP BY Titlu

HAVING COUNT(IdS) > 2

 Găsiți numărul de studenți înscriși și media notelor pentru toate cursurile de 6 credite

SELECT Titlu, COUNT(IdS) as Nr, AVG(Nota) as Medie

FROM Cursuri as C INNER JOIN Examene as E ON C.IdC = E.IdC

WHERE Credite = 6

GROUP BY Titlu

 Găsiți numărul de studenți din fiecare grupă cu mai puțini studenți decât grupa 2

SELECT Grupa, COUNT(IdS) as NrStudenti

FROM Studenti

GROUP BY Grupa

HAVING COUNT(IdS) < (SELECT COUNT(IdS) FROM Studenti WHERE Grupa = 2)

SQL - Exemple

 Găsiți studenții cu nota cea mai mare la Baze de date (dacă sunt mai mulți, atunci

îi afișăm pe toți)

SELECT TOP 1 Nume, Nota

FROM Studenti as S INNER JOIN Examene as E ON S.IdS = E.IdS

ORDER BY Nota DESC

Obs. Access SQL selectează toti studentii cu aceeasi nota! Daca rezultatul trebuie sa includa o singura

inregistrare, atunci se va ordona si dupa un atribut cheie (NrMatricol sau IdS)

 Afisati studentii si notele la cursuri. Gasiti inclusiv studentii care nu au note.

SELECT Nume as Student, Titlu as Curs, Nota

FROM (Studenti as S LEFT JOIN Examene as E ON S.IdS=E.IdS) LEFTJOIN

Cursuri as C ON E.IdC = C.IdC

CURS 12

MRD - Anomalii

• Supraîncărcarea unei relații cu prea multe semnificații => anomalii

• Redundanță

• Dependență mare a datelor

• Solutie: regândirea schemei bazei de date (re-design) în scopul

eliminării anomaliilor

• Adăugările, modificările, ștergerile de informații ar trebui să se

facă în câte un singur tabel, și apoi să se propage în restul bd prin

legăturile definite între tabele.

• Metode

• Modelul Entități-Asociații, Modelul Relational

• Teoria formală a normalizării

• Forme normale: FN1, FN2, FN3, FN4, BCFN

MRD - Anomalii

 Un tabel prezintă anomalie de modificare (update) dacă o

aceeaşi informaţie trebuie actualizată în înregistrări diferite.

 Un tabel prezintă anomalie de inserare dacă nu se pot insera

înregistrări decât cu adăugarea de valori null pentru unele

dintre coloane

 Anomalie de stergere: pierdere neintenționată de date din

cauza stergerii unor alte date.

MRD - Anomalii

Exemplu

 Anomalie de modificare: daca vrem sa modificam bugetul proiectului Physics,

trebuie sa facem modificarea in toate inregistrarile acestui proiect

 Anomalie de inserare: vrem sa adaugam un nou departament, fara a cunoaste cine

lucreaza in acesta (deci vom avea null pt name,salary)

 Anomalie de stergere: daca stergem pe Mozart, vom sterge si departamentul Music

Solutie: identificarea corectă a tipurilor de entități implicate, modelarea

DEPARTAMENTE si PROFESORI ca entitati/tabele (relații) separată și reprezentarea

Departament ca cheie străină în PROFESORI cu legătură către noua entitate

DEPARTAMENTE

• Redundata!

• Dependenta

de date!

Anomalii de inserare

Exemplu : fie relatia ANGAJATI(IdAng, Nume, CNP, Sef); dacă Sef ar fi

obligatoriu de introdus (not null), nu am putea introduce un angajat al

cărui șef nu îl cunoaștem (nu am putea introduce nici măcar primul

angajat, am fi într-un cerc vicios)

• Soluție, eliminarea constrângerii not null pentru atributul Sef!

Exemplu: fie relatia PERSOANE(IdP, Cod, Nume, Religie) - dacă s-ar

șterge toate persoanele de o anumită religie, ar dispărea complet

informațiile despre religia respectivă.

• Soluţia: tabel separat cu RELIGII și cheie străină în PERSOANE

care semnifică religia unei persoane

Anomalii de stergere

Scopul proiectării unei baze de date

 Evitarea redundanței și a anomaliilor:

 Pot insera date la fiecare entitate în parte

 Pot șterge date pentru o entitate (fără a afecta alte entități)

 Modificările care țin cont de cheile primare si străine nu pot

duce la inconsistențe

Rafinarea unei structuri

defectuoase

• Cum determinăm dacă o structură este “bună” sau ”defectuoasă”?

• Cum transformăm o structură defectuoasă într-una bună?

• Teoria dependenţelor funcţionale furnizează o abordare

sistematică a celor două întrebări

Rafinarea unei structuri defectuoase

Exemplu

 Constrângeri:

 Fiecare film are un regizor

 Fiecare cinematograf are un număr de telefon

 Fiecare cinematograf începe proiecţia unui singur film al un moment dat

Rafinarea unei structuri defectuoase

Exemplu

 Anomalie de inserare: Se poate insera următorul film?

INSERT INTO MovieList (Title,Director)

VALUES ('I.T.', 'Moore')

 Anomalie de stergere: Apare vreo problemă la următoarea operație de ștergere?

DELETE FROM MovieList

WHERE Director = 'Jackson'

 Anomalie de modificare: Apare vreo problemă la următoarea operație de actualizare?

UPDATE MovieList

SET Phone = 441133

WHERE Title = 'The Hobbit' and Cinema = 'Cinema City'

Title Director Cinema Phone Time

The Hobbit Jackson Cinema City 441111 11:30

The Lord of the Rings3 Jackson Cinema City 441111 14:30

Adventures of Tintin Spielberg Odeon 442222 11:00

The Lord of the Rings3 Jackson Odeon 442222 14:00

War Horse Spielberg Odeon 442222 16:30

Rafinarea unei structuri defectuoase prin

descompunerea în mai multe structuri “bune”

Movies

Cinema

Screens

Title Director

The Hobbit Jackson

The Lord of the

Rings3

Jackson

Adventures of

Tintin

Spielberg

War Horse Spielberg

Cinema Phone

Cinema City 441111

Odeon 442222

Cinema Time Title

Cinema City 11:30 The Hobbit

Cinema City 14:30 The Lord of the Rings3

Odeon 11:00 Adventures of Tintin

Odeon 14:00 The Lord of the Rings3

Odeon 16:30 War Horse

Dependenţe funcţionale

A → B

A, B sunt submulţimi de atribute ale R

“A determină funcţional B”

sau

“B depinde functional de A”

Definiţie dependenţe funcţionale

• Dependenţa funcţionalăA → B este satisfăcută de R dacă şi

numai dacă pentru orice instanţă a lui R, oricare două tupluri t1

şi t2 pentru care valorile luiA sunt identice vor avea de

asemenea valori identice pentru B

• FieA și B două submulțimi de atribute pentru relația R,tt1,tt2

tupluri din instanța r(R) și πA (tt1) proiecția înregistrării tt1 pe

mulțimea de atributeA.

A → B⇔ (∀tt1,tt2 ∈ rr(RR):πA (t1) = πA (tt2) ⇒ πB (tt1) = πB(tt2))

Definiţie dependenţe funcţionale

• O dependenţă funcţională A → B este trivială dacă B ⊆A

• O dependenţă funcţională A → B este totală/completă dacă

eliminarea oricărui atribut dinA are ca rezultat anularea

dependenței

• Adică nu există nici o submulțime C ⊆A, astfel încât să aibă loc C → B

• O dependenţă funcţională A → B este parțială dacă există un atribut

care poate fi eliminat dinA şi totuşi dependenţa să se menţină.

Definiţie dependenţe funcţionale

• Ce dependențe funcționale puteți identifica?

Definiţie dependenţe funcţionale

• Ce dependențe funcționale puteți identifica?

Title → Director

Cinema → Phone

Cinema, Time →Title (doar daca un cinema are o singură sală)

Dependenţe funcţionale

• Atenție. Dependențele funcționale depind de semantica schemei

bazei de date, nu de instanța bazei de date!

• Au loc următoarele dependențe funcționale?

• Composer → Director ?

• Director → Composer ?

Dependenţe funcţionale

• Atenție. Dependențele funcționale depind de semantica schemei

bazei de date, nu de instanța bazei de date!

• Au loc următoarele dependențe funcționale?

• Composer → Director ? – NU + ar fi incalcata de instanta bazei de date

• Director → Composer ? – instanța dată satisface dependența, dar asta NU

înseamnă că dependența are loc în relația Movie!

Exemplu

Distribuitor ( NumeProdus, NumeDistribuitor,AdresaDistribuitor, Pret)

• Cine ar putea fi cheie?

• Ce dependențe funcționale identificați?

Exemplu

Distribuitor ( NumeProdus, NumeDistribuitor,AdresaDistribuitor, Pret)

• Cine ar putea fi cheie? NumeProdus\* NumeDistribuitor

• Ce dependențe funcționale identificați?

• NumeDistribuitor →AdresaDistribuitor

• NumeDistribuitor, NumeProdus → Pret

• NumeDistribuitor,NumeProdus → NumeProdus (trivială)

• NumeDistribuitor,NumeProdus →AdresaDistribuitor (parțială)

• NumeDistribuitor → NumeDistribuitor (trivială)

Superchei, chei & atribute prime

• Dacă C este o cheie pentru RR(AA1,AA2,...,AAnn), atunci CC→ ββ,

∀ββ⊂{AA1,AA2,...,AAnn}

• Deoarece o astfel de dependență este valabilă tot

timpul, ea nu se va elimina prin descompuneri.

• Un atributA se numeşte atribut prim dacăA face parte dintr-o cheie

a lui R; în caz contrar,A se numeşte atribut neprim.

Descompunerea relaţiilor

• Descompunerea unei relaţii R este o mulţime de (sub)relaţii {R1, R2,

..., Rn} a.i.

Ri⊆ R și

R = ∪ Ri

• Dacă r este o instanţă din R, atunci r se descompune în {r1,r2,...,rn},

unde fiecare ri

= ππRi

(r).

• Exemplu: MovieList(Title, Director, Cinema, Phone,Time) se descopume

în:

• MOVIES = (Title, Director)

• CINEMA = (Cinema, Phone)

• SCREENS = (Cinema,Time,Title)

Proprietăţile descompunerii relaţiilor

1. Descompunerea trebuie să păstreze informaţiile – Descompunere

cu joncțiune fără pierderi (Lossless-Join Decomposition)

• Datele din relaţia originală = Datele din relaţiile descompunerii

• Crucial pentru păstrarea consistenţei datelor!

• Putem reconstrui r prin jonctiunea (join) proiecțiilor sale {r1,r2, ... ,rn}

• Descompunere cu joncțiune făra pierderi:

r = πRR1(r)⋈... ⋈ πRRnn(r)

2. Descompunerea trebuie să respecte toate dependențele funcționale (DF)

• Dependenţele funcţionale din relaţia originală = reuniunea

dependenţelor funcţionale din relaţiile descompunerii

• Facilitează verificarea violărilor DF

Descompunere cu joncțiune fără pierderi

• Întrebarea 1: Cum determinăm dacă {R1,R2} este o descompunere cu

joncţiuni fără pierderi a lui R?

• Descompunerea lui R în {R1, R2} este cu joncţiuni fără pierderi

dacă şi numai dacă are loc una din dependențele funcționale:

R1 ∩ R2 → R1 sau

R1 ∩ R2 → R2

• Întrebarea 2: Cum descompunem R în {R1,R2 }astfel încât aceasta e cu

joncţiuni fără pierderi?

Corolar: Dacă α → β este satisfăcută pe R şi α ∩ β = ∅, atunci

descompunerea lui R în {R – β, α ∪ β } este o descompunere cu

joncţiuni fără pierderi.

Descompunere cu joncțiune fără pierderi -

Exemplu

• R(A,B,C) cu dependențele funcționale: F = {A → B}

• Descompunerea {AB,AC} e cu joncțiuni fără pierderi deoarece

• AB ∩AC=A și A →AB

• Descompunerea {AB, BC} nu e cu joncțiuni fără pierderi deoarece

AB ∩ BC= B și nici una din dependenţete B →AB sau B → BC nu sunt

respectate de R

Descompunere cu joncțiune fără pierderi -

Exemplu

Descompunere cu joncțiune fără pierderi

• Tranzitivitate:

Dacă {R1, R2} este o descompunere cu joncţiuni fără pierderi a lui R,şi dacă

{R1\_1, R1\_2} e o descompunere cu joncţiuni fără pierderi a lui R1, atunci {R1\_1,

R1\_2, R2} e o descompunere cu joncţiuni fără pierderi a R:

Descompunere cu joncțiune fără pierderi

Forme Normale

• Dependenţele funcţionale pot fi utilizate pentru identificarea problemelor de proiectare

şi sugerează posibile îmbunătăţiri

• Nicio DF (în afară de faptul ca orice atribut depinde de cheie): nu avem

Redundanţa!

• O formă normală este o proprietate a unei baze de date relaționale.

• Dacă o relaţie se află într-o formă normală particulară avem certitudinea că

anumite categorii de probleme sunt eliminate/minimizate → ne ajută să

decidem daca descompunerea unei relaţii este necesară sau nu.

• Normalizare:

• Normalizarea este o procedură care permite schemelor ne-normalizate să fie

transformate în noi scheme pentru care satisfacerea unei forme normale este

garantata

• Fiecare relație corespunde unei singure entități (sau unei relații dintre două

entități în cazul modelării unui M:N)

• Pe cât posibil nu apar redundanțe

• Nu apar anomalii la operațiile de inserare, modificare,ștergere

• Sunt definite și se respectă constrângerile (cheile primare,cheile străine)

Obs. Soluția pentru normalizare este de obicei descompunerea relației, dar această

descompunere trebuie obligatoriu să fie cu joncțiune fără pierderi

Forme Normale

Formele normale bazate pe DF sunt:

• prima formănormală (1FN),

• a doua formă normală (2FN),

• a treia formă normală (3FN),

• a patra forma norma a(4FN)

• forma normală Boyce-Codd (BCNF).

1FN

• Definiţie. O relaţie se află în

Prima Formă Normală (1FN) dacă

fiecare atribut al relaţiei poate

avea doar valori atomice (deci

listele şi mulţimile sunt excluse)

• altfel, relația este încuibată

(imbricată, eng. Nested).

• În MRD, relațiile sunt

in FN1!

2FN

• Un atributA se numeşte atribut prim dacăA face parte dintr-o cheie

a lui R; în caz contrar,A se numeşte atribut neprim.

• Spunem că avem o dependenţă funcţională parţială într-o relaţie atunci când un

atribut neprim este dependent funcţional de o parte a cheii primare a relaţiei

(dar nu de întreaga cheie).

• Definiţie. O relaţie se află în a Doua Formă Normală(2FN) dacă este 1FN şi

nu există un atribut neprim care depinde parţial de cheia primara a relatiei

• Adică orice atribut neprim depinde total de cheia primara a relației.

• Daca tabela are o cheie primara formata dintr-un singur atribut,

atunci ea este automat in 2FN!

• cheie primara surogat!

2FN

• Exemplu:

EXAMEN(Student,Curs, Profesor, Nota)

• Cheie: Student \* Curs

• Student, Curs → Profesor e dependență parțiala

(Curs → Profesor)

=> Descompunere:

• INROLARI(Student, Curs, Nota)

• CURSURI(Curs, Profesor)

2FN

Exemplu

Cheia (de interes) este compusă din Produs \* Depozit

 Probleme: Adresa\_depozit contine (neprim) contine info despre Depozit (parte din

cheie)

 Adresa depozitului este repetata pentru fiecare produs din depozit

 Daca se modifica adresa, tb modificata peste tot

 Daca intr-un loc este scrisa gresit, datele pot deveni inconsistente

 Daca nu mai exista produse intr-un depozit, se pot pierde informatiile despre depozit

(eg, adresa), cand se va sterge ultimul produs din depozit

 Cauza: tabelul conţine informaţii despre 2 tipuri de entităţi: DEPOZITE şi PRODUSE,

cu stocurile acestora in depozite

2FN

Exemplu

PRODUS

 Solutie: descompunerea relaţiei in 2 relaţii

PRODUSE va conţine un câmp Depozit, cheie străină către DEPOZITE

3FN

• Definitie 1. O relaţie R ce satisface dependenţele funcţionale F se află în A

Treia Formă Normală (3FN) dacă, pentru toate dependențele funcționale A

→ B din F+ (nu doar cele date, ci și cele care se pot deduce) are loc una

din următoarele:

• B ⊆A (trivială), sau

• A este o supercheie pentru R, sau

• B este un atribut (sau o mulțime de atribute) prim.

• Definiție 2. O relaţie R se află în A Treia Formă Normală (3FN) dacă este

în 2NF și nici un atribut neprim nu depinde de un atribut neprim.

• Definitie 3: Toate atributele neprime ale unei relatii depind numai de

chei candidate ale acelei relatii

Obs. FN3 nu este satisfacuta cand un atribut neprim contine informatii despre

un alt camp neprim!

• Descompunerea cu joncţiune fără pierderi & cu păstrarea dependenţelor

relaţiei R într-o mulţime de relaţii 3FN este întotdeauna posibilă.

3FN

Exemplu

 O cheie în acest tabel ar CNP

 Departament ->Locatie

 Probleme: Locatia (neprim) contine informaţie despre Departament (neprim)

 Locatia unui departament este copiata pentru fiecare student din acel departament

 Daca locatia se modifica, trebuie modificat în toate înregistrările

 Datele redundante lasă loc pentru inconsistenţe (pot apare locaţii diferite pentru

acelaşi departament)

 Dacă se sterg toţi studenţii dintr-un departament, dispar informaţiile despre

departament cu totul

 Nu se pot insera informaţii despre un departament fără să existe studenţi înregistraţi

în acel departament

 Cauza: tabelul conţine informaţii despre 2 tipuri de entităţi, de fapt: STUDENŢI şi

DEPARTAMENTE

3FN

Exemplu

 Solutie: descompunerea în 2 relaţii

 STUDENTI va contine un camp Departament, cheie straina catre tabela

DEPARTAMENTE

3FN - Exemplu

• DISERTATIE(Student,Titlu, Profesor,Departament)

• Profesor → Departament nu respectă nici una din condiții:

• Profesor – nu e cheie!

• Dependența nu e trivială

• Descompunere:

• DISERTATIE (Student,Titlu, Profesor)

• PROFESORI (Profesor, Department)

Bill Kent: the relation is based on the key, the whole key and nothing

but the key", la care unii adauga: "so help me Codd“ (“relatia depinde

de cheie, de intreaga cheie si de nimic altceva decat de cheie”, la care

unii adauga “asa sa ne ajute Codd” ☺)

BCNF

• Boyce-Codd e o versiune mai restrictiva decat FN3

• R este în BCNF dacă singurele dependenţe funcţionale satisfăcute de R sunt

cele corespunzătoare constrângerilor de cheie.

• Dacă R este în BCNF, evident este şi în 3NF.

BCNF –Exemplu

• Tabelul este in 3NF pentru ca toate atributele depind de o cheie si numai de o cheie; cu

toate acestea, exista o redundanta, deoarece perechile (cod\_sofer, cod\_autobuz) (S1, A1) ,

(S2, A2) apar de cate doua ori in tabel

• Se pot elimina astfel de redundante astfel:

• din tabelul initial se elimina coloana “cod\_autobuz”;

• se creeaza un nou tabel, cu atributele “cod\_sofer” si “cod\_autobuz”, cu doar 2

coloane, constituind impreuna cheia primara a noului tabel.

• tabelele rezultate sunt in BCNF

Forme Normale bazate pe DF

1NF – toate valorile atributelor sunt atomice

2NF – toate atributele neprime depind de întreaga cheie (nu sunt

dependenţe parţiale)

3NF – tabele în 2NF şi toate atributele neprime depind doar de cheie (nu sunt

depedenţe tranzitive)

BCNF – Toate dependenţele sunt date de chei

4FN

 Are legătură cu atribute multi-valoare

 Pentru ca o relaţie să fie în 4FN

 Trebuie să fie în 3FN

 Nicio înregistrare nu poate conţine doua sau mai multe

atribute cu valori independente despre o entitate

4FN

Exemplu: Culoare şi LimbaVorbita sunt atribute cu valori independente

despre o persoana

4FN

Anomalii de actualizare, insertie, stergere

 Pot apare ambiguităţi, de ex: dacă într-o înregistrare lipseşte culoarea,

asta inseamna ca persoanei nu-i place nicio culoare?

 Null inseamna, in acest caz, ca informatiile se gasesc in alte inregistrari

 Informatiile apar repetate: de ex, persoanei cu CNP124 ii place

culoareaVerde

 Repetitiile inseamna probleme de actualizare (inconsistente datorate nevoii de

actualizari multiple)

 Ca sa se stearga informatia ca o persoana stie o anumita limba, trebuie

sterse mai multe inregistrari

 Si pot dispare si informatii legate de culori, in acelasi timp

 Adaugarea uneii informatii, gen unei persoane ii place o anumita culoare:

trebuie fie cautata o inregistrare in care culoarea este goala, fie creata o

inregistrare noua, dar care va ramane cu LimbaVorbita goala

4FN

 Solutia: descompunerea in 2, 3 relatii

"Strategia" de normalizare

• BCNF prin descompunere cu joncţiune fără pierderi şi păstrarea

dependenţelor (prima alegere)

• 3NF prin descompunere cu joncţiune fără pierderi şi păstrarea

dependenţelor (a doua alegere)

Obs. Uneori dependenţele nu pot fi păstrate pentru a obţine BCNF.

Descompunerea în BCNF

• Fie relaţia R cu dependenţele funcţionale F. Dacă α → β nu respectă

BCNF, descompunem R în:

R – β și α ∪ β.

• Aplicarea repetată a acestei idei va conduce la o colecţie de relaţii

care

• Sunt în BCNF;

• Conduc la joncţiune fără pierderi;

Descompunerea în BCNF

• R(C, S , J, D, P, Q, V), C – cheie primară

• F ={ JP → C, SD → P, J → S}

• SD → P nu respectă BCNF ⇒descompunem în (S,D,P) și (C,S,J,D,Q,V)

• J→ S nu respectă BCNF ⇒(C,S,J,D,Q,V) se descompune în (J, S) și (C, J,

D, Q, V)

• Descompunerea finală a lui R este (S,D,P) , (J, S) și (C, J, D, Q, V) – cu

joncțiune fără pierderi, dar fără a păstra dependențele

• Adăugând (J,P,C) la mulţimea de relaţii obţinem descompunere cu

păstrarea dependenţelor.

BCNF & redundanţă

• Obs. În general, mai multe dependenţe pot cauza nerespectarea BCNF. Ordinea în care le “abordăm

”

poate conduce la decompuneri de relaţii

complet diferite!

Normalizare si Proiectare conceptuala/logica

 Teoria normalizării poate fi folosita pentru controlul calitatii

schemelor conceptuale si logice

 Ideile pe care se bazeaza normalizarea pot fi, de asemenea, utilizate în

timpul fazei de proiectare conceptuala pentru controlul calitatii fiecarui

element al schemei conceptuale

 Analiza relatiilor obtinute în timpul proiectarii logice

poate identifica locurile unde a fost proiectarea conceptuala

a fost inexacta: aceasta verificare a proiectarii este adesea relativ usoara