**SQL:**

* CHEIA PRIMARA:

->nu se repeta

->nu are voie sa lipseasca

->este UNICA

->nu poate fi NULL

* Cheia externa:

->se repeta => operatori (unique, distinct)

->nu este unica

->poate fi NULL

->daca se afla in tabel, se afla si in tabelul pe care il refera

* Cum vad tipul de date din tabel si cheile:

DESC nume\_tabel;

* Aliasuri:

->fara spatii decat cu “ “/un singur cuvant

->numele capului de tabel

* Concatenare: ||

-> sirurile de caractere se pun intre ‘ ‘

Ex: || ‘, ‘ ||

->din mai multe coloane facem una singura

* BETWEEN/ NOT BETWEEN: (interval inchis)

->val1 AND val1

* Apartenenta la o multime:

->IN(el ignora nul)

->NOT IN(el nu se impaca bine cu NULL)

* ORDER BY 1/n/alias/coloana;

ATENTIE: WHERE NU accepta virgula!

* Prelucrare data calendaristica/alte valori:

Select sysdate from DUAL;

* Operatorul LIKE:

-> \_ tine locul unui singur caracter

-> ‘%x%’(contine cuvantul x)

->’%’ sirul vid

ATENTIE: compararea cu NULL se face cu IS/IS NOT !

AND>OR !

* Functii simple: UPPER(), LOWER(), INITCAP(), LENGTH()(lucreaza de la 1)
* Conversii explicite :

->TO\_CHAR(data, ‘format’)

->TO\_NUMBER(‘nr char’, 99.999)

->TO\_DATE(‘data char’, ‘format’)

* Conversii implicite:

->varchar/char⬄ number

->varchar/char ⬄ date

* Alte functii single-row:

->SUBSTR(string, start, len)=> sir(incepe de la 1)

->INSTR(string1, string2, start, n)=> a n-a aparitie a lui string2 in string1 de la start

->ROUND(expresie, zecimale)=>numar

->LPAD/RPAD(string1, len, string2)=> adauga la stanga lui string1 atatia de string2 ca sa ajunga la dimensiunea len

->NVL(expresie1, expresi2)+NOT IN

Daca expresie1=NULL=>expresie2

Altfel =>expresie1

->NVL2(expresie1, expresie2, expresie3)

Daca expresie1 !=NULL =>expresie2

Altfel daca expresie1=NULL=>expresie 3

Toate expresiile trebuie sa aiba acelasi tip!

* Operatii cu nr de zile:

Data+-zile=>data

Data1-Data2=>nr zile+round

DECODE=CASE:

DECODE(valoare, if1, tehn 1,

If2, then 2,……

…then x) +alias

CASE valoare WHEN if1 TEHN ..

WHEN if2 THEN…

…………………..

ELSE ….

END alias

* JOIN INTRE TABELE:

2 metode:

1) conditia scrisa in WHERE

2) conditia scrisa in FROM (+join +on/using)

Produs cartezian:

Select x, y

From tabel1, tabel2;

(+) -> il punem unde avem deficit de informative (invers fata de left/roght join)

* Tipuri de join : self, LEFT, RIGHT, FULL
* Operatori pe multimi:

1) UNION

-> reuneste si elimina duplicatele

-> NU ignora NULL

2)UNION ALL

->reuneste si NU elimina duplicatele

-> NU ignora NULL

->NU se poate folosi cu DISTINCT/UNIQUE

3)INTERSECT

->intersecteaza si elimina duplicatele

->NU ignora NULL

4)MINUS

->ce nu apare in a 2 a dar apare in prima  
 ->ce se afla in WHERE se afla si in SELECT

->elimina duplicatele

* Subcereri:

->operatori single-row: =,<,>,<>,<=

->operatori multiple-row: IN, ALL, ANY

ATENTIE: La subcereri cand avem mai multe clause in WHERE punem intr() cu virgule in interior !

Functiile speciale (ex AVG, MIN, MAX ) NU pot fi puse in WHERE !

>ANY=> mai mare ca minimul

>ALL=> mai mare ca maximul

<ANY=> mai mic ca minimul

<ALL=> mai mic ca maximul

=ANY⬄ IN

!=ALL ⬄ NOT IN(atentie la null)!!--> nvl

* COUNT(Expresie)

->ignora NULL

-> expresie !=NULL

->functiile grup nu se pot folosit fara GROUP BY

-> nu accepta f1(f2(x)) ci doar o subcerere f1(x)

-> COUNT (\*) NU ignora NULL

GROUP BY => grupare 🡪 ignora valorile null cu exc lui count(\*)

Atentie la valorile care nu-s distincte le numara de mai multe ori -> avem nevoie de distinct in functia grup

+HAVING => pentru formule, comparatii=> criteriu->accepta functii grup

Functiile grup count, max, min, avg se pot folosi si fara group by

ATENTIE LA ORDINE :

1) WHERE

2) GROUP BY

3) HAVING

* COLACESE-> afiseaza primul element diferit de NULL
* Subcerere in FROM:

->stocata temporar=>alt tabel temporar

->de preferat sa-I dau alias

->se poate face JOIN cu el

-> se pot prelua elemente din el

* Subcerere in WHERE:

->folosesc aliasul de la FROM

-> e un fel de join

ROWNUM:

-> numeroteaza coloanele in ordinea in care le gaseste in tabel

Ex: SELECT ROWNUM

FROM …….

* WITH:

-> se scrie o singura data

->putem sa punem oricat blocuri de cod

->dupa subcereri se pune ,

-> inainte de SELECT

Ex: WITH tabel1 AS (subcerere),

tabel2 AS(subcerer)

……………………………….

SELECT ………

“toti”, “toate” -> COUNT x 2+GROUP BY+HAVING

CURSURI:

DIAGRAME ENTITATE-RELAȚIE

**poate**(cardinalitatemaximă) **trebuie**(cardinalitateminima)

Câţi salariaţi **pot** lucra într-un departament? Mulţi!

În câte departamente **poate** lucra un salariat? In cel mult unul!

Relaţia SALARIAT\_lucreaza\_in\_DEPARTAMENT are cardinalitatea maximă ***many-one***(n:1).

Câţi salariaţi **trebuie** să conducă un departament? Cel puţin unul!

Câte departamente **trebuie** să conducă un salariat? Zero!

Relaţia SALARIAT\_conduce\_DEPARTAMENT are cardinalitatea minimă ***one-zero***(1:0).

**Atribut**: **proprietate** descriptivăa unei **entităţi** sau a unei **relaţii**.

Trebuie făcută distincţia între atribut (devine coloană în modelele relaţionale) şi valoarea acestuia (devine valoare încoloane).

Atributele sunt substantive, dar nu orice substantive este atribut.

Fiecărui atribut trebuie să I se dea o descriere complete (exemple, contraexemple, caracteristici).

Pentru fiecare atribut trebuie specificat numele, tipul fizic (*integer*, *float*, *char*etc.),valori posibile, valori implicite, reguli de validare

**Moştenirea atributelor.**

**Subentitate**(subclasă) –submulţimea unei alte entităţi, numită **superentitate**(superclasă) (SALARIAT < ––> PROGRAMATOR).

Subentitatea se desenează prin dreptunghiuri incluse în superentitate.

Existăo relaţie între o subentitate şi o superentitate, numită **ISA** , care are cardinalitatea maximă 1:1 şi minimă1:0.

Cheile primare, atributele şi relaţiile unei superentităţi sunt valabile pentru orice subentitate. Afirmaţia reciprocă este falsă.

Din entităţi similar care au mai multe attribute commune se pot crea **superentităţi**.

Aceste superentităţi conţin atributele comune, iar atributele special sunt asignate la subentităţi. Pentru noile superentităţi se introduce chei primare artificiale.

După valorile unor attribute clasificatoare se pot determina **clase**.

Un grupde subentităţi reciproc exclusive defineşte o clasă.

(Claselese aliniază în desen vertical. )

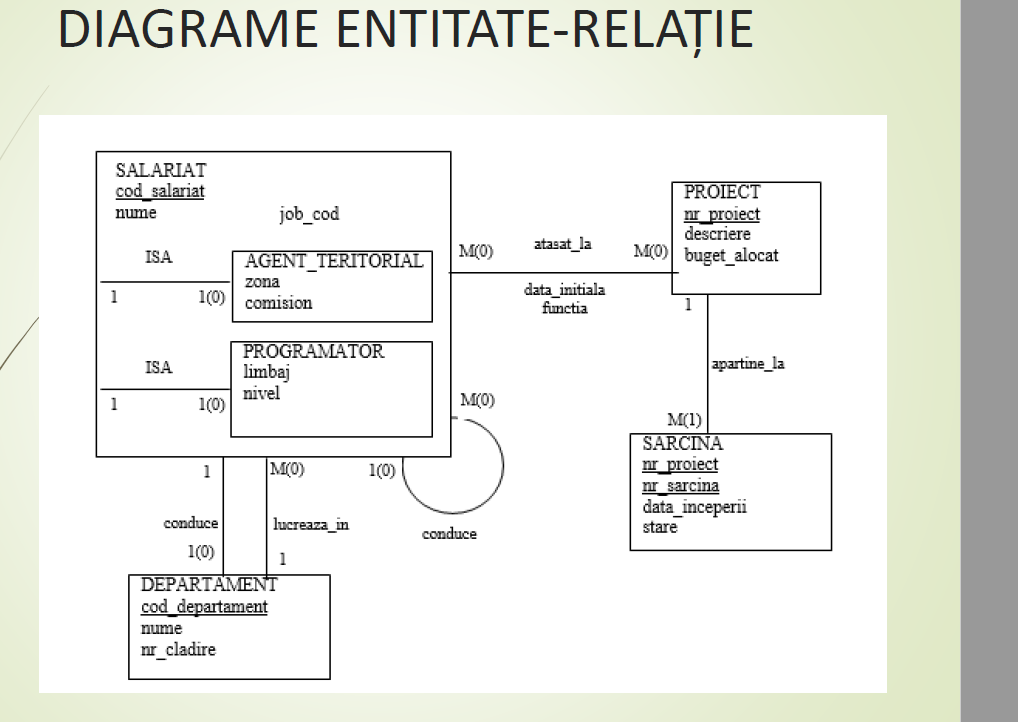
Într-o diagramăE/R se pot define **relaţii recursive**.

Unele relaţii sunt relative la două entităţi şi le numim de **tip 2**, iar dacă relaţiile implică mai mult de două entităţi, le vom numi de **tip 3**.

Trei relaţii de tip 2 sunt diferite de o relaţie de tip 3.

Rupând o relaţie de tip 3 în trei relaţii de tip 2, pot apărea informaţii incorecte

Dacă un atribut al unei entităţi reprezintă cheia primară a unei alte entităţi, atunci el referă o relaţie(*cod\_departament* în tabelul SALARIAT).



Observații temă

1. Cardinalitatile se pun la ambele capete ale relatiei (nu doar la mijloc)

2. Relatiile diferite trebuie sa fie semantic diferite. De exemplu: CAMIN se afla in LOCATIE, LOCATIE are CAMIN este o singura relatie, nu o dublez in diagrama

3. In anumite reprezentari, exista "roluri" ale relatiei=> regasim verbe la ambele capete=> OK

4. Nu punem attribute ce au valori multiple. Exemplu: FACULTATE nu are attribute precum specializari, profesori

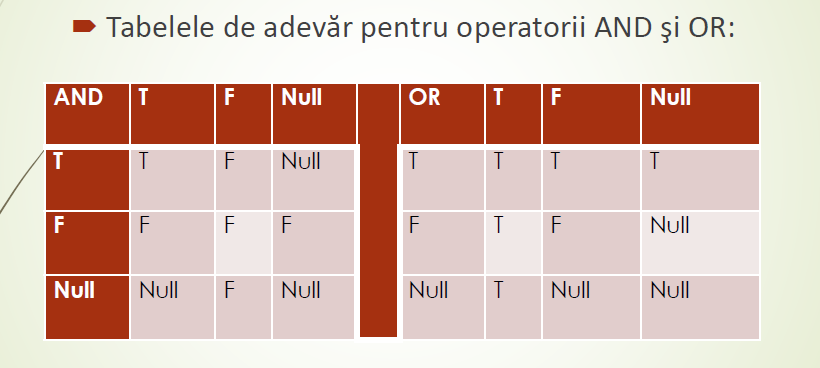
5. Gardian, administrator, personal curatenie-o entitate cu atribut job si eventual subentitati. Cand defines subentitati?

6. Greseli la cardinalitate(valorile au fost inversate intre capete).

7. Deoarece universitatea este sistemul al carui aspect il modelez=> NU este entitate in model! Ar fi fost daca modelam evidenta caminelor din universitatile din Romania sau dintr-un oras anume

8. Unele modele au presupus(restrictiv) ca un student poate urma o singura facultate si un camin apartine in intregime unei singure facultati.

9. Mai multe modele care au presupus ca un student poate urma mai multe facultati nu raspund la intrebarea: pe locurile carei facultati este cazat studentul?

Rezultatul operatorilor aritmetici sau logici este *null* când unul din argumente este *null*

Fie schemele relaţionale *R*1(*P*1, *S*1) şi *R*2(*S*1, *S*2), unde *P*1 este cheie primară pentru *R*1, *S*1 este cheie secundară pentru*R*1, iar *S*1 este cheie primară pentru *R*2. În acest caz, vom spune că *S*1 este **cheie externă**(cheie străină) pentru *R*1

Modelul relational respect **trei reguli de integritate structurală**.

**Regula 1** –**unicitate acheii**. Cheia primară trebuie să fie **unică** şi **minimală**.

**Regula 2**–**integritatea entităţii**. Atributele cheii primare trebuie să fie **diferite de valoarea *null***.

**Regula 3–integritatea referirii**. O cheie externă trebuie să fie ori ***null* în întregime**, ori să corespundă unei **valori a cheii primare asociate**.

**Transformarea relaţiilor**

Relaţiile 1:1 şi1:*n* devin **chei externe**.

Simbolul„X“ indică plasamentul cheii externe, iar simbolul„X subliniat“ exprimă faptul că această cheie externă este conţinută în cheia primară. Relaţia1:1 plasează cheia externă în tabelul cu mai puţine linii.

Relaţia *m*:*n* devine un tabel special, numit **tabel asociativ**, care are două chei externe pentru cele două tabele asociate.

Cheia primară este compunerea acestor două chei externe plus eventuale coloane adiţionale.

Tabelul se desenează punctat.

Relaţiile de tip trei devin **tabele asociative**.

Cheia primară este compunerea a trei chei externe plus eventuale coloane adiţionale.

**Operatorii algebrei relaţionale** sunt:

operatori tradiţionali pe mulţimi(**UNION, INTERSECT, PRODUCT, DIFFERENCE**);

operatori relaţionali speciali(**PROJECT, SELECT, JOIN, DIVISION**).

Operatoti:

**Notaţii**: PROJECT

**Π*A*1, ..., *Am*(*R*)**

**PROJECT (*R*, *A*1, ..., *Am*)**

*R*[*A*1, ..., *Am*]

Unde *A*1, *A*2, ..., *Am* sunt parametrii proiecţiei relativ la relaţia *R*.

**Notaţii**: SELECT

**σcondiţie(*R*)**

*R*[condiţie]

**SELECT(*R*, condiţie)**

RESTRICT(*R*, condiţie).

**Notaţii**: UNION

***R*** U ***S***

**UNION(*R*, *S*)**

OR(*R*, *S*)

APPEND(*R*, *S*).

**Notaţii**:DIFFERENCE

***R*–*S***

**DIFFERENCE(*R*, *S*)**

REMOVE(*R*, *S*)

MINUS(*R*, *S*).

**Notaţii**: INTERSECT

**INTERSECT(*R*, *S*)**

***R U intors S***

AND(*R*, *S*).

**Notaţii:PRODUCT**

***R*** x ***S***

**PRODUCT(*R*, *S*)**

TIMES(*R*, *S*).

**Notaţii**: DIVISION

DIVIDE(*R*, *S*)

**DIVISION(*R*, *S*)**

***R*** punct | orzinontala punct ***S*.**