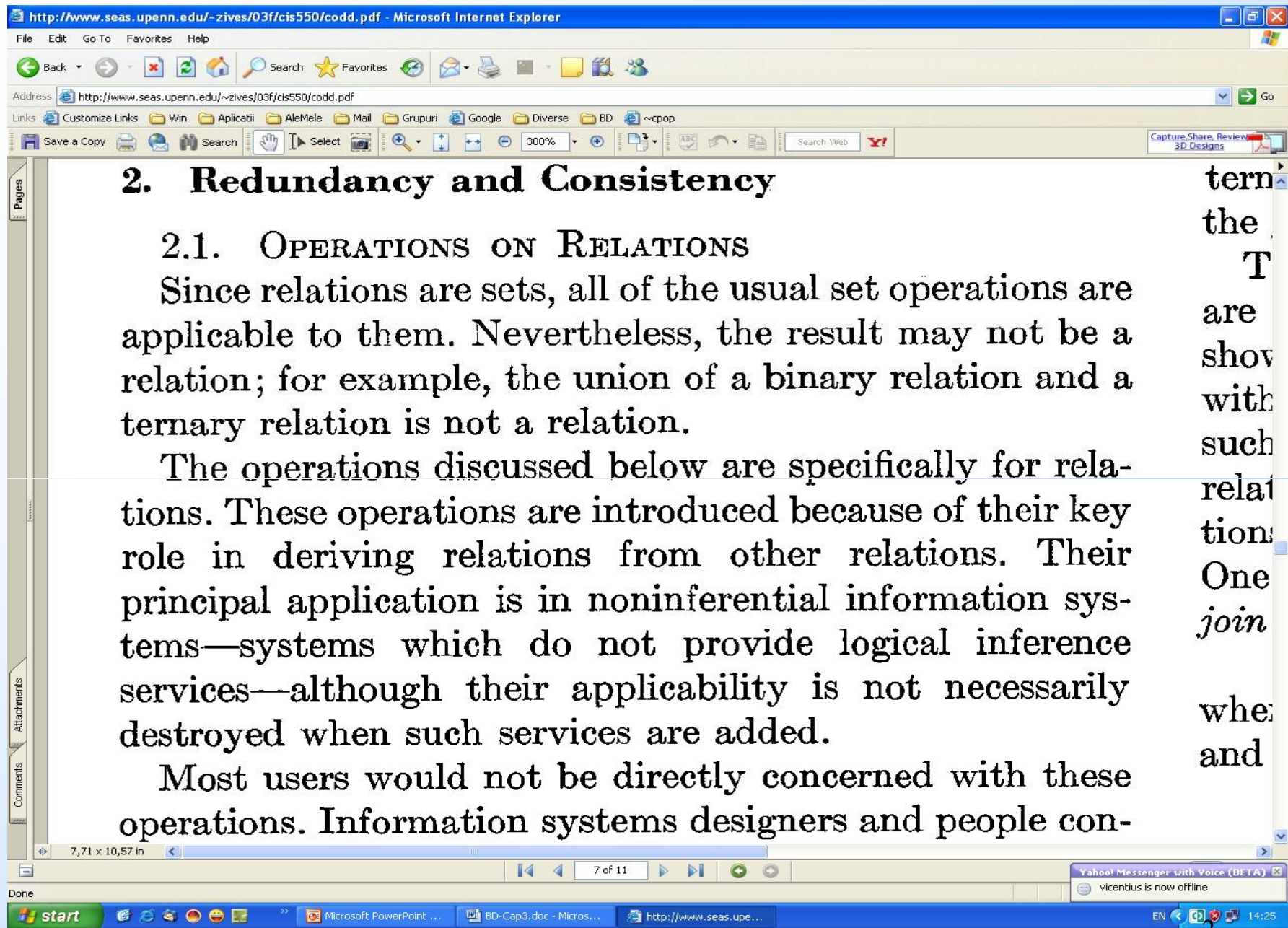


# ALGEBRA

- ◆ Înca din primul articol în care introduce modelul relational, E.F. Codd propune un set de operatori pentru lucrul cu relatii.
- ◆ O relatie este o multime de tupluri => o parte dintre acesti operatori provin direct din teoria multimilor.
- ◆ Ceilalti operatori, introdusi in aceasta algebra pentru relatii (numita in literature de specialitate *algebra relationala*) sunt specifici acesteia si au la baza operatii uzuale cu tabele – acestea fiind reprezentarea intuitiva pentru relatii.



tern  
the  
T  
are  
show  
with  
such  
relat  
tions  
One  
join  
  
whe  
and

# OPERATO RI

- ◆ Exista mai multi operatori in cadrul acestei algebre, unii dintre ei fiind derivati (se pot rescrie in functie de alti operatori). Putem imparti acesti operatori in doua categorii:
  - ◆ Operatori derivati din teoria multimilor.
  - ◆ Operatori specifici algebrei relationale

# REUNIUNE

- ◆ Reuniunea: Fie date doua relatii  $R$  si  $S$ , reuniunea lor, notata  $R \cup S$  este o relatie care contine tuplurile care sunt fie in  $R$ , fie in  $S$  fie in ambele relatii. In rezultatul reuniunii nu apar tupluri duplicat.
- ◆ Pentru ca aceasta operatie sa poata fi executata cele doua relatii care se reunesc trebuie sa aiba scheme compatibile (acelasi numar de coloane provenind din aceleasi domenii (deci cu acelasi tip de date).
- ◆ Echivalent SQL: operatorul UNION prin care se pot reuni rezultatele a doua cereri SQL de tip SELECT.

# REUNIUNEA (2)

A	B	C
1	1	2
2	1	3
1	3	2

Rela  
tia R

A	B	C
4	1	2
2	1	3
1	3	2
5	1	7

Rela  
tia S

A	B	C
1	1	2
2	1	3
1	3	2
4	1	2
5	1	7

Relatia  
 $R \cup S$

# DIFERENT

- ◆ Diferenta: Fiind date doua relatii R si S, diferenta lor, notata  $R - S$  este o relatie care contine tuplurile care sunt in R si nu sunt in S.
- ◆ Si in cazul diferentei cele doua relatii care se reunesc trebuie sa aiba scheme compatibile.
- ◆ Echivalent SQL: operatorul MINUS prin care se poate face diferenta intre rezultatele a doua cereri SQL de tip SELECT.

# DIFERENTA (2)

A	B	C
1	1	2
2	1	3
1	3	2

Rela  
tia R

A	B	C
4	1	2
2	1	3
1	3	2
5	1	7

Rela  
tia S

A	B	C
1	1	2

Relati  
a R -  
S

# INTERSECT

- ◆ Intersectia: ~~Fiind~~ Fiind date doua relatii R si S, intersectia lor, notata  $R \cap S$  este o relatie care contine tuplurile care sunt si in R si in S. De asemenea cele doua relatii care se reunesc trebuie sa aiba scheme compatibile.
- ◆ Echivalent SQL: operatorul INTERSECT prin care se poate calcula intersectia rezultatelor a doua cereri SQL de tip SELECT.



# INTERSECTIA (2)

A	B	C
1	1	2
2	1	3
1	3	2

Relatia R

A	B	C
4	1	2
2	1	3
1	3	2
5	1	7

Relatia S

A	B	C
2	1	3
1	3	2

Relatia  $R \cap S$

# INTERSECTIA

- ◆ Observatie: Intersectia este un operator derivat. Putem rescrie orice intersectie astfel:

$$R \cap S = R - (R - S)$$

# PRODUS

## CARTEZIAN

- ◆ Produsul cartezian. Fie date doua relatii  $R$  si  $S$ , produsul lor cartezian, notata  $R \times S$  este o relatie ale carei tupluri sunt formate prin concatenarea fiecărei linii a relatiei  $R$  cu fiecare linie a relatiei  $S$ .
- ◆ Rezulta de aici urmatoarele:
  - ◆ Numarul de attribute (coloane) ale lui  $R \times S$  este egal cu suma numerelor de attribute ale lui  $R$  si  $S$
  - ◆ Numarul de tupluri (linii) ale lui  $R \times S$  este egal cu produsul numerelor de tupluri ale lui  $R$  si  $S$

# PRODUS CARTEZIAN

(2)

- Dacă în R și S avem attribute (coloane) cu același nume, în produsul cartezian R × S vom avea attribute care au același nume.
- Pentru a le deosebi se prefixează numele atributului cu cel al relației din care provine (ex.: R.A și S.A, ca în exemplul următor)

# PRODUS CARTEZIAN

- ◆ (3) Echivalent SQL:
- ◆ In clauza FROM a unei cereri SELECT apar doua (sau mai multe) tabele
- ◆ In cazul standardului SQL-3, se poate folosi clauza CROSS JOIN a unei cereri de regasire de date de tip SELECT prin care se poate efectua produsul cartezian a doua tabele.

# PRODUS CARTEZIAN

## (4)

◆ Exemplu:  
Fie  
relatiile:

A	B	C
1	1	2
2	1	3
1	3	2

Rela  
tia R

A	C	D	E
4	1	2	5
2	1	3	1

Rela  
tia S

# PRODUS CARTEZIAN

◆ Rezultat:  
(4)

R.	R.	R.	S.	S.	S.	S.
A	B	C	A	C	D	E
1	1	2	4	1	2	5
1	1	2	2	1	3	1
2	1	3	4	1	2	5
2	1	3	2	1	3	1
1	3	2	4	1	2	5
1	3	2	2	1	3	1

# ALGEBRA RELATIONALA CLASICA

- ◆ Exista mai multi operatori in cadrul acestei algebre, unii dintre ei fiind derivati (se pot rescrie in functie de alti operatori). Putem imparti acesti operatori in doua categorii:
  - ◆ Operatori derivati din teoria multimilor.
  - ◆ Operatori specifici algebrei relationale



# PROIECTII

- ◆ Proiectia: Fie data o relatie  $R$  si o multime de attribute ale acesteia  $X=A_1, A_2, \dots A_n$ , proiectia lui  $R$  pe multimea de attribute  $X$  este o relatie care se obtine din  $R$  luand doar coloanele din  $X$  (in aceasta ordine) si eliminand eventualele tupluri duplicat.
- ◆ Notatia pentru selectie este urmatoarea:

$$\pi_X(R) \text{ sau } \pi_{A_1 A_2 \dots A_n}(R)$$

# PROIECTIA

- ◆ Echivalență SQL: Clauza SELECT a unei cereri de regasire de date în care este specificată lista de expresii care dă structura de coloane a rezultatului.
- ◆ Exemplu: din relația R de mai jos dorim să calculăm  $\pi_{B, C, E}(R)$

# PROIECTIA

(3)

A	B	C	D	E
1	1	2	1	3
2	1	2	1	3
2	7	4	4	1
2	3	9	2	1
1	3	7	4	1
1	3	9	2	1

Relati  
a R

B	C	E
1	2	3
7	4	1
3	9	1
3	7	1

Rezultatul proiectiei  $\pi_{B, C, E}(R)$  Observam ca s-au eliminat doua linii duplicat din rezultat (cele provenite din liniile 2 si 6).

# PROIECTIA

- ◆ Nota: in multimea de attribute pentru o proiectie poate sa apara toate attributele relatiei. In acest caz se obtine o relatie cu acelasi continut cu cea initiala dar in care coloanele sunt permutate:

$$\pi_{B, C, A, E, D}(R)$$

# PROIECTIA

(5)

A	B	C	D	E
1	1	2	1	3
2	1	2	1	3
2	7	4	4	1
2	3	9	2	1
1	3	7	4	1
1	3	9	2	1

Relati  
a R

B	C	A	E	D
1	2	1	3	1
1	2	2	3	1
7	4	2	1	4
3	9	2	1	2
3	7	1	1	4
3	9	1	1	2

Rezultatul proiectiei  $\pi_{B, C, A, E, D}^{(R)}$

# SELECTI

- ◆ Selectia (numita uneori restrictia):  
Fiind data o relatie R si o expresie logica F (o conditie), selectia lui R in raport cu F este o relatie care se obtine din R luand doar liniile care verifica expresia logica F.
- ◆ Notatia pentru selectie este urmatoarea:

$$\sigma_F(R)$$

# SELECTIA

- ◆ Echivalent SQL: Clauza WHERE a unei cereri de regasire de date de tip SELECT pe care se scrie conditia pe care trebuie sa o indeplineasca liniile pentru a trece mai departe spre rezultat.
- ◆ Exemplu: din relatia R de mai jos dorim sa calculam  $\sigma_{B+1 > A+C}(R)$ :

# SELECTIA

(3)

A	B	C	D	E
1	1	2	1	3
2	1	2	1	3
2	7	4	4	1
2	3	9	2	1
1	3	7	4	1
1	3	9	2	1

Relatia R

A	B	C	D	E
2	7	4	4	1

Rezultatul selectiei

$$\sigma_{B+1 > A+C}(R)$$



# JOI

- ◆ Joinul general (numit si theta-join sau  $\theta$ -join): fiind date doua relatii R si S, joinul lor (notat  $R \bowtie_F S$ ) se obtine din produsul cartezian al relatiilor R si S urmat de o selectie dupa conditia F (numita si *conditie de join*).
- ◆ Denumirea de theta-join este folosita din motive istorice, simbolul  $\theta$  fiind folosit initial pentru a desemna o conditie.
- ◆ Rezulta ca:

$$R \bowtie_F S = \sigma_F(R \times S)$$

# JOIN

Sa luam un (ex)emplu concret pentru exemplificarea acestui operator: Sa consideram ca avem doua relatii, STUD si SPEC avand schemele:

- ◆ STUD(Mat, Nume, CodSpec, Media)
- ◆ SPEC(CodS, NumeS)

# JOIN

(3)

Mat r	Nume	CodSp ec	Medi a
101	Ionescu Ion	10	8
102	Popescu Maria	11	9
302	Georges cu Vasile	10	9,50

Relatia  
STUD

Cod S	NumeS
10	Calculato si Tehnolog are ia Informati ei
11	Automati si Informati ca SPEC ca Industrial a

# JOIN

- ◆ Sa consideram urmatoarele joinuri:
- ◆  $STUD \bowtie_{STUD.CodSpec=SPEC.CodS} SPEC$
- ◆ Rezultatul celor doua joinuri este urmatorul:

# JOIN

- ◆ In cazul in care (5) conditia de join este una de egalitate, joinul se mai numeste si echijoin (ca in cazul joinului precedent).
- ◆ In restul cazurilor se foloseste sintagma non-echijoin (joinul urmator).

# JOIN

- ◆ Echivalent ~~SQL~~ (6):
- ◆ In clauza FROM a unei cereri de regasire de tip SELECT apar tabelele care participa la join +
- ◆ In clauza WHERE se pune conditia de join, conectata cu AND de celelalte conditii care eventual sunt necesare in cererea respectiva.

# JOIN

## NATURAL

- ◆ Join natural - Join natural pentru doua relatii R si S (notat  $R \bowtie S$ ) se obtine:
- ◆ facand joinul celor doua relatii dupa conditia: "coloanele cu aceeasi semnificatie au valori egale" +
- ◆ eliminand prin proiectie coloanele duplicat (cele dupa care s-a facut joinul).

# JOIN NATURAL

- ◆ Echivalent SQL: Clauza NATURAL JOIN din sintaxa SQL-3.
- ◆ Observatie: deoarece SGBD-ul nu cunoaste semnificatia coloanelor, conditia de join implicita in acest caz este "coloanele cu acelasi nume au valori egale"



# JOIN NATURAL

- ◆ Exemplu (B): In cazul celor doua tabele de mai sus, STUD si SPEC, joinul lor natural va fi asemanator cu echijoinul anterior, lipsind insa coloana duplicat SPEC.CodS (care are aceleasi valori ca si coloana STUD.CodSpec)
- ◆ Obs: In cazul folosirii clauzei NATURAL JOIN cele doua coloane trebuie sa aiba acelasi nume

# JOIN NATURAL

## (4)

Mat r	Nume	CodS pec	Med ia	NumeS
101	Ionescu Ion	10	8	Calculatoare si Tehnologia Informatiei
102	Popescu Maria	11	9	Automatica si Informatica Industriala
302	Georgescu Vasile	10	9,50	Calculatoare si Tehnologia Informatiei

# JOIN

## EXTERN

- ◆ Join extern. Așa cum s-a văzut din nonechijoinul anterior, în cazul în care o linie a unei tabele, oricare ar fi concatenarea ei cu o altă linie din cealaltă tabelă, nu îndeplinește condiția de join, **linia respectiva nu are corespondent în rezultat.**
- ◆ Este cazul liniilor studenților de la specializarea 10 și al liniei specializării 11.

# JOIN EXTERN

- ◆ In unele cazuri se doreste insa ca aceste linii sa apara in rezultat, cu valori nule pe coloanele din cealalta tabela.
- ◆ Aceasta operatie poarta numele de **join extern** (in engleza outer join).
- ◆ Cum la un join participa doua tabele, pot exista trei tipuri de join extern:

# JOIN EXTERN

- ◆ **Join extern stanga** (left outer join), in care in rezultat apar toate liniile tabelului din stanga operatorului. Notatia este:  $R \Join_L S$ .
- ◆ **Join extern dreapta** (right outer join), in care in rezultat apar toate liniile tabelului din dreapta operatorului. Notatia este:  $R \Join_R S$ .
- ◆ **Join extern complet** (full outer join), in care in rezultat apar toate liniile tabelurilor din stanga si din dreapta operatorului. Notatia este:  $R \Join S$ .
- ◆ De notat ca in rezultatul joinului extern sunt **intotdeauna** continute tuplurile (liniile) din rezultatul joinului general dupa aceeasi conditie

# SEMIJOIN

- ◆ Semijoin: Fie două relatii  $R$  si  $S$ . Atunci semijoinul lui  $R$  in raport cu  $S$  ( notat  $R \bowtie S$  ) este o relatie care contine multimea tuplurilor lui  $R$  care participa la joinul natural cu  $S$ .
- ◆ Semijoinul este un operator derivat. Putem scrie ca:
- ◆  $R \bowtie S = \pi_R (R \Join S)$
- ◆ Semijoinurile pot fi folosite in optimizarea cererilor de regasire in baze de date distribuite.

# Bibliografie

1. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer D. Widom: *Database Systems: The Complete Book*, Prentice-Hall, Englewood Cliffs, NJ, 2002.
2. F. Rădulescu : *Oracle SQL, PL/SQL*, Editura Printech,