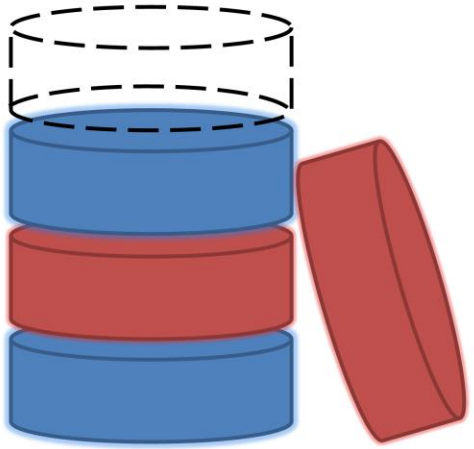
SQL DML (cont)

GROUP BY, HAVING
ODER BY



Valoarea NULL

- În anumite situații valorile particulare ale unor attribute (câmpuri) pot fi *necunoscute* sau *inaplicabile* temporar.
  - SQL permite utilizarea unei valori speciale *null* pentru astfel de situații.

- Prezența valorii *null* implică unele probleme suplimentare:
  - E necesară implementarea unei logici cu 3 valori: *true*, *false* și *null* (de exemplu o condiție de tipul *rating>8* va fi intotdeauna evaluată cu *false* daca valoarea câmpului rating este *null*)
  - E necesară adăugarea unui operator special IS NULL / IS NOT NULL.

Operatori de agregare

COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A)
MIN (A)

atribut

```
SELECT  COUNT (*)
FROM  Students S
```

```
SELECT  AVG (S.age)
FROM  Students S
WHERE  S.gr=921
```

```
SELECT COUNT (DISTINCT S.gr)
FROM  Students S
WHERE S.name='Bob'
```

```
SELECT S.name
FROM Students S
WHERE S.age = ANY
      (SELECT MAX(S2.age)
       FROM  Students S2)
```

GROUP BY / HAVING

For *i* = 221,222,223,224...:

```
SELECT MIN(S.age)
FROM   Students S
WHERE  S.gr = i
```

GROUP BY / HAVING

```
SELECT [DISTINCT] target-list
FROM   relation-list
WHERE  qualification
GROUP BY  grouping-list
HAVING    group-qualification
```

GROUP BY / HAVING

```
SELECT [DISTINCT] target-list
FROM   relation-list
WHERE  qualification
GROUP BY  grouping-list
HAVING    group-qualification
```

- Un *grup* este o mulțime de tupluri care au aceeași valoare pentru toate atributele din *grouping-list*.

- *target-list* conține (i) nume de atribute sau (ii) termeni ce utilizează operatori de agregare (e.g., MIN (*S.age*)).
  - numele de atribute (i) trebuie să fie o submulțime a *grouping-list*.
  - Intuitiv, fiecare tuplu din rezultat corespunde unui *grup*, și toate atributele vor avea o singură valoare per grup.

Numarul studentilor cu nota la cursurile cu 6 credite si media notelor acestora

```
SELECT  C.cid,  COUNT (*) AS scount, AVG(grade)
FROM  Enrolled E, Courses C
WHERE E.cid=C.cid AND C.credits=6
GROUP BY  C.cid
```

Courses

cid	*cname*	*credits*
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

Students

sid	*name*	*email*	*age*	*gr*
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Enrolled

sid	***cid***	*grade*
1234	Alg1	9
1235	Alg1	10
1234	DB1	10
1234	DB2	9
1236	DB1	7

Enrolled

sid	*cid*	*grade*
1234	Alg1	9
1234	Alg1	9
1234	Alg1	9
1235	Alg1	10
1235	Alg1	10
1235	Alg1	10
1234	DB1	10
1234	DB1	10
1234	DB1	10
1234	DB2	9
1234	DB2	9
1234	DB2	9
1236	DB1	7
1236	DB1	7
1236	DB1	7

Courses

cid	*cname*	*credits*
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

```
SELECT C.cid,
COUNT(*)AS scount,
AVG(grade)AS average
FROM   Enrolled E,
       Courses C
WHERE
       E.cid=C.cid
       AND
       C.credits=6
GROUP BY C.cid
```

Enrolled

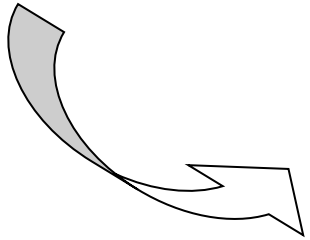
sid	*cid*	*grade*
1234	Alg1	9
1235	Alg1	10
1234	DB1	10
1234	DB2	9
1236	DB1	7

Courses

cid	*cname*	*credits*
Alg1	Algoritmics 1	7
Alg1	Algoritmics 1	7
DB1	Databases1	6
DB2	Databases2	6
DB1	Databases1	6

```
SELECT C.cid,
COUNT(*)AS scount,
AVG(grade)AS average
FROM   Enrolled E,
       Courses C
WHERE  E.cid=C.cid
       AND
       C.credits=6
GROUP BY C.cid
```

sid	*cid*	*grade*	*cid*	*cname*	*credits*
1234	DB1	10	DB1	Databases1	6
1234	DB2	9	DB2	Databases2	6
1236	DB1	7	DB1	Databases1	6



cid	*scount*	*average*
DB1	2	8.5
DB2	1	9

```
SELECT C.cid
COUNT(*)AS scount,
AVG(grade)AS average
FROM   Enrolled E,
       Courses C
WHERE  E.cid=C.cid
       AND
       C.credits=6
GROUP BY C.cid
HAVING MAX(grade) = 10
```

Sortarea rezultatului interogărilor

- ORDER BY *column* [ASC | DESC] [, ...]

```
SELECT cname, sname, grade
FROM Courses C
      INNER JOIN Enrolled E ON C.cid = E.cid
      INNER JOIN Students S ON E.sid = S.sid
ORDER BY cname, grade DESC , sname
```

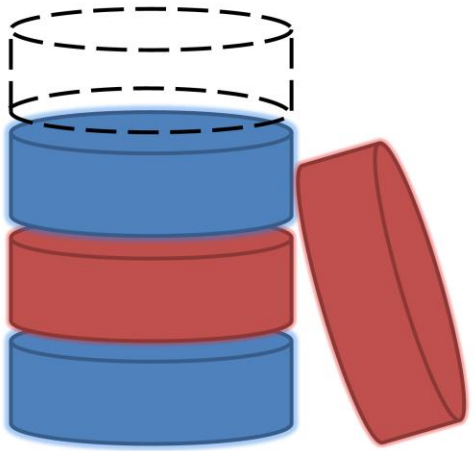
Sortarea rezultatului interogărilor

- Rezultatul e sortat după orice câmp din clauza SELECT, inclusiv expresii sau agregări:

```
SELECT gr, Count(*) as StudNo
FROM Students C
GROUP BY gr
ORDER BY StudNo
```

Rafinarea structurii bazelor de date

(Dependențe funcționale)



3

Structura bazei de date

=

Structura relațiilor

+

Constrângeri

Exemplu: relația *MovieList*

Title	*Director*	*Cinema*	*Phone*	*Time*
The Hobbit	Jackson	Florin Piersic	441111	11:30
The Lord of the Rings 3	Jackson	Florin Piersic	441111	14:30
Adventures of Tintin	Spielberg	Victoria	442222	11:30
The Lord of the Rings 3	Jackson	Victoria	442222	14:00
War Horse	Spielberg	Victoria	442222	16:30

Constrângeri:

- Fiecare film are un regizor
- Fiecare cinematograf are un număr de telefon
- Fiecare cinematograf începe proiecția unui singur film al un moment dat

Proiectare defectuoasă!

Title	*Director*	*Cinema*	*Phone*	*Time*
The Hobbit	Jackson	Florin Piersic	441111	11:30
The Lord of the Rings 3	Jackson	Florin Piersic	441111	14:30
Adventures of Tintin	Spielberg	Victoria	442222	11:30
The Lord of the Rings 3	Jackson	Victoria	442222	14:00
War Horse	Spielberg	Victoria	442222	16:30

Anomalie de inserare

Anomalie de ştergere

Anomalie de actualizare

Rafinarea unei structuri defectuoase prin *descompunerea* în mai multe structuri "bune"

Movies

Title	*Director*
The Hobbit	Jackson
The Lord of the Rings 3	Jackson
Adventures of Tintin	Spielberg
War Horse	Spielberg

Screens

Cinema	*Time*	*Title*
Florin Piersic	11:30	The Hobbit
Florin Piersic	14:30	The Lord of the Rings 3
Victoria	11:30	Adventures of Tintin
Victoria	14:00	The Lord of the Rings 3
Victoria	16:30	War Horse

Cinema

Cinema	*Phone*
Florin Piersic	441111
Victoria	442222

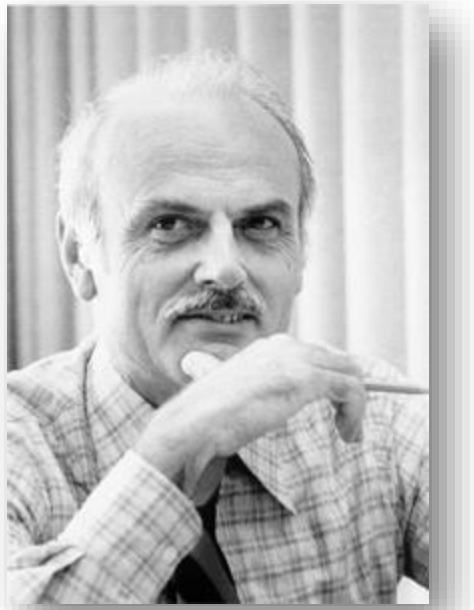
- ✓ Anomalie de inserare
- ✓ Anomalie de ştergere
- ✓ Anomalie de actualizare

Cum determinăm
dacă o structură este
"bună" sau *"defectuoasă"*?

Cum transformăm
o structură ***defectuoasă***
într-una ***bună***?

Teoria *dependențelor funcționale* furnizează o abordare sistematică a celor două întrebări

Introdusă de
Edgar Frank Codd în:



"A relational model for large shared data banks", Com. of the ACM, 13(6), 1970, pp.377-387.

Dependențe funcționale

$$\alpha \rightarrow \beta$$

α, β sunt submulțimi de atribute ale R

"**α** determină funcțional **β**"

sau

"**β** depinde functional de **α**"

Definiție dependențe funcționale

Dependența funcțională $\alpha \rightarrow \beta$ este satisfăcută de R dacă şi numai dacă

pentru *orice* instanță a lui R,

oricare două tupluri t_1 şi t_2 pentru care valorile lui α sunt identice

vor avea de asemenea valori identice pentru β.

O dependență funcțională

$$\alpha \to \beta$$

este **trivială** dacă

$$a \supseteq \beta.$$

Title	*Director*	*Cinema*	*Phone*	*Time*
The Hobbit	Jackson	Florin Piersic	441111	11:30
The Lord of the Rings 3	Jackson	Florin Piersic	441111	14:30
Adventures of Tintin	Spielberg	Victoria	442222	11:30
The Lord of the Rings 3	Jackson	Victoria	442222	14:00
War Horse	Spielberg	Victoria	442222	16:30

Dependențe funcționale pentru relația *MovieList*:

1. Title → Director
2. Cinema → Phone
3. Cinema, Time → Title

Fie r instanța unei relații R

- Spunem că r **satisface DF** $\alpha \rightarrow \beta$ dacă
  pentru orice pereche de tupluri
  t_1 şi t_2 din r astfel încât $\pi_\alpha(t_1) = \pi_\alpha(t_2)$,
  este de asemenea adevărat că $\pi_\beta(t_1) = \pi_\beta(t_2)$.

sau

$$\forall\ t_1, t_2 \in r$$
$$\pi_\alpha(t_1) = \pi_\alpha(t_2) \Rightarrow \pi_\beta(t_1) = \pi_\beta(t_2)\ *$$

* $\pi_\alpha(t)$ este proiecția atributelor α pentru tuplul t

Fie *r* instanța unei relații *R*

- o **DF *f* este satisfăcută pe** *R* dacă şi numai dacă orice instanță *r* a lui *R* satisface *f*

- *r* **nu respectă** o DF *f* dacă *r* nu satisface *f*.

- *r* este o **instanță legală a lui R** dacă *r* satisface toate dependențele funcționale definite pentru *R*.

Exemplu: ***Movie***(*Title, Director, Composer*)

Title	*Director*	*Composer*
Schindler's List	Spielberg	Williams
Saving Private Ryan	Spielberg	Williams
North by Northwest	Hitchcock	Herrmann
Angela's Ashes	Parker	Williams
Vertigo	Hitchcock	Herrmann

- DF *composer* → *director* nu este respectată de relația ***Movie***
- *r* satisface DF *director* → *composer*

  Acest lucru nu înseamnă că *director*→*composer* e respectat de ***Movie***!

Problema implicației

Putem deduce că o DF f e respectată de R pe baza unei mulțimi de DF F ?

Exemplu: în ***MovieList***, avem

F = { Title → Director
Cinema → Phone
Cinema, Time → Title }

- *Time* → *Director* este respectată?
- Dar *Cinema, Time* → *Director*?

F **implică logic** pe f

notat prin

$$\mathbf{F \Rightarrow f}$$

daca fiecare instanță r a relației R

ce satisface F

satisfice şi f

F & G : mulțimi de dependențe funcționale

f : dependeța funcțională

F implică logic G

notat prin

$$\boldsymbol{F \Rightarrow G}$$

dacă $F \Rightarrow g$ pentru fiecare $g \in G$

Închiderea lui **F**

(notată prin F^+)

este mulţimea tuturor DF implicate de F

$$F^+ = \{ f \mid F \Rightarrow f \}$$

F şi G sunt **echivalente**

(notat prin $F \equiv G$)

dacă

$F^+ = G^+$

(adică $F \Rightarrow G$ şi $G \Rightarrow F$)

Axiomele lui Armstrong

Fie $\alpha, \beta, \gamma \subseteq R$

Reflexivitate: Dacă $\beta \subseteq \alpha$, atunci $\alpha \rightarrow \beta$

Augmentare: Dacă $\alpha \rightarrow \beta$, atunci $\alpha\gamma \rightarrow \beta\gamma$

Tranzitivitate: Dacă $\alpha \rightarrow \beta$ şi $\beta \rightarrow \gamma$, atunci $\alpha \rightarrow \gamma$

Sistemul axiomelor lui Armstrong este

Corect

(Orice FD derivată este implicată de F)

&

Complet

(Toate DF din F^+ pot fi derivate)

Exemplu: Fie R(A, B, C, D, E) cu mulțimea

$$F = \{A \rightarrow C;\ B \rightarrow C;\ CD \rightarrow E\}.$$

Arătați că $F \Rightarrow AD \rightarrow E$

Soluție:

1. $A \rightarrow C$ (dat)
2. $AD \rightarrow CD$ (augumentare cu (1))
3. $CD \rightarrow E$ (dat)
4. $AD \rightarrow E$ (tranzitivitate cu (2) si (3))

Reguli de inferență adiționale

Reuniunea:

Dacă $\alpha \to \beta$ şi $\alpha \to \gamma$, atunci $\alpha \to \beta\gamma$

Descompunerea:

Dacă $\alpha \to \beta$, atunci $\alpha \to \beta'$ pentru orice $\beta' \subseteq \beta$

Exemplu: Aratati ca $\{A \rightarrow BCD\} \equiv \{A \rightarrow B;\ A \rightarrow C;\ A \rightarrow D\}$

Fie $F = \{A \rightarrow BCD\}$

Fie $G = \{A \rightarrow B;\ A \rightarrow C;\ A \rightarrow D\}$

Prin regula de descompunere avem

$$F \Rightarrow A \rightarrow B,$$

$$F \Rightarrow A \rightarrow C, \text{ si}$$

$$F \Rightarrow A \rightarrow D$$

Prin urmare $F \Rightarrow G$

Din regula reuniunii avem

$$\{A \rightarrow B;\ A \rightarrow C\} \Rightarrow A \rightarrow BC \text{ si}$$

$$\{A \rightarrow BC;\ A \rightarrow D\} \Rightarrow A \rightarrow BCD$$

Prin urmare $G \Rightarrow F$, deci $F \equiv G$

Superchei, chei & atribute prime

- O mulțime de atribute α reprezintă o supercheie a relației R (având mulțimea de DF F) dacă

$$F \Rightarrow \alpha \rightarrow R.$$

- O mulțime de atribute α e o cheie a relației R dacă

  (1) α este o supercheie, şi

  (2) nici o submulțime a lui α nu e supercheie

  (adică, pentru fiecare $\beta \subset \alpha$, $\beta \rightarrow R \notin F^+$)

- Un atribut $A \in R$ se numeşte atribut prim dacă A face parte dintr-o cheie a lui R; în caz contrar, A se numeşte atribut neprim.

- Considerăm din nou relația

  MovieList (Title, Director, Cinema, Phone, Time)

cu DF

(1) Cinema, Time → Title

(2) Cinema → Phone

(3) Title → Director

- {*Cinema, Time*} este singura cheie a relației ***MovieList***.
- *Cinema* şi *Time* sunt singurele atribute prime din ***MovieList***.
- Orice mulțime ce include {*Cinema; Time*} e supercheie a ***MovieList***.

Închiderea atributelor

Fie $\alpha \subseteq R$ şi F o mulţime de DF satisfăcute pe R

- Închiderea lui α (cu respectarea mulțimii F de DF), notată cu α^+, este mulțimea de atribute ce sunt determinate funcțional din α pe baza dependențelor funcționale din F; adică

$$\alpha^+ = \{A \in R \mid F \Rightarrow \alpha \rightarrow A\}$$

- Se observă că $F \Rightarrow \alpha \rightarrow \beta$ dacă şi numai dacă
$\beta \subseteq \alpha^+$(cu respectarea DF din F)

Algoritm pt deteminarea închiderii atributelor

Input: α, F

Output: α^+ (w.r.t. F)

Compute a sequence of sets of attrs α_0, α_1,... α_k, α_{k+1} as follows:

$\alpha_0 = \alpha$

$\alpha_{i+1} = \alpha_i \cup \gamma$ such that there is some FD $\beta \rightarrow \gamma \in F$ and $\beta \subseteq \alpha_i$

Terminate the computation once $\alpha_{k+1} = \alpha_k$ for some k

Return α_k

Input: α, F
Output: α^+ (w.r.t. F)
Compute a sequence of sets of attrs α_0, α_1,... α_k, α_{k+1} as follows:

$\alpha_0 = \alpha$

$\alpha_{i+1} = \alpha_i \cup \gamma$ such that there is some FD $\beta \rightarrow \gamma \in F$ and $\beta \subseteq \alpha_i$

Terminate the computation once $\alpha_{k+1} = \alpha_k$ for some k

Return α_k

Exemple: Fie F = {A → C;B → C;CD → E}, aratati ca F ⇒ AD→E

i	α_i	*FD folosit*
0	AD	dat
1	ACD	A→C
2	ACDE	CD → E
3	ACDE	-

Deci AD^+ = ACDE. Deoarece E ∈AD^+, rezulta ca F ⇒ AD→E

Descompunerea relațiilor

Descompunerea unei relații R

este o mulțime de (sub)relații

$\{R_1, R_2, ..., R_n\}$

astfel încât fiecare $R_i \subseteq R$ si $R = \cup R_i$

Dacă *r* este o instanță din R,

atunci *r* se descompune în

$\{r_1, r_2, ..., r_n\}$,

unde fiecare $r_i = \pi_{Ri}(r)$

Proprietățile descompunerii relațiilor

1. Descompunerea trebuie să păstreze informațiile
   - Datele din relația originală ≡ Datele din relațiile descompunerii
   - Crucial pentru păstrarea consistenței datelor!

2. Descompunerea trebuie să respecte toate DF
   - Dependențele funcționale din relația originală ≡ reuniunea dependențelor funcționale din relațiile descompunerii
   - Facilitează verificarea violărilor DF

1. Descompunerea trebuie să păstreze informațiile

Cu alte cuvinte:

putem reconstrui *r*

prin jonctiunea proiectiilor sale

$\{r_1, r_2, \ldots, r_n\}$

Observație: dacă $\{R_1, R_2, \ldots, R_n\}$ e o descompunere a R, atunci pentru orice instanță *r* din R, avem

$$r \subseteq \pi_{R1}(r) \otimes \pi_{R2}(r) \otimes \ldots \otimes \pi_{Rn}(r)$$

Descompunerea relațiilor

$\{ M_1 = (\text{Cinema, Time})$

$M_2 = (\text{Time, Title}),$

$M_3 = (\text{Title, Director}),$

$M_4 = (\text{Cinema, Phone})\}$

e o descompunere a:

MovieList(Title, Director, Cinema, Phone, Time)

Exemplu: relația *MovieList*

Title	*Director*	*Cinema*	*Phone*	*Time*
The Hobbit	Jackson	Florin Piersic	441111	11:30
The Lord of the Rings 3	Jackson	Florin Piersic	441111	14:30
Adventures of Tintin	Spielberg	Victoria	442222	11:30
The Lord of the Rings 3	Jackson	Victoria	442222	14:00
War Horse	Spielberg	Victoria	442222	16:30

Constrângeri:

- Fiecare film are un regizor
- Fiecare cinematograf are un număr de telefon
- Fiecare cinematograf începe proiecția unui singur film al un moment dat

MovieList(Title, Director, Cinema, Phone, Time)

M1

Cinema	*Time*
Florin Piersic	11:30
Florin Piersic	14:30
Victoria	11:30
Victoria	14:00
Victoria	16:30

M2

Time	*Title*
11:30	The Hobbit
14:30	The Lord of the Rings 3
11:30	Adventures of Tintin
14:00	The Lord of the Rings 3
16:30	War Horse

M3

Title	*Director*
The Hobbit	Jackson
The Lord of the Rings 3	Jackson
Adventures of Tintin	Spielberg
War Horse	Spielberg

M4

Cinema	*Phone*
Florin Piersic	441111
Victoria	442222

Title	*Director*	*Cinema*	*Phone*	*Time*
The Hobbit	Jackson	Florin Piersic	441111	11:30
The Hobbit	Jackson	Victoria	442222	11:30
The Lord of the Rings 3	Jackson	Florin Piersic	441111	14:30
Adventures of Tintin	Spielberg	Florin Piersic	441111	11:30
Adventures of Tintin	Spielberg	Victoria	442222	11:30
The Lord of the Rings 3	Jackson	Victoria	442222	14:00
War Horse	Spielberg	Victoria	442222	16:30

Descompunere cu joncțiune fără pierderi
(Lossless - Join Decomposition)

O descompunere a R (având DF *F*) în

$\{R_1, R_2, \ldots, R_n\}$

este o

descompunere cu joncțiuni fără pierderi

cu respectarea mulțimii *F*

dacă

$\pi_{R1}(r) \otimes \pi_{R2}(r) \otimes \ldots \otimes \pi_{Rn}(r) = r$

pentru orice instanță r din R ce satisface F.

Fie descompunere lui R(A,B,C) in $\{R_1(AC), R_2(BC)\}$

r

A	B	C
a_1	b_1	c
a_1	b_2	c
a_2	b_1	c

r_1

A	*C*
a_1	c
a_2	c

r_2

B	*C*
b_1	c
b_2	c

$r_1 \otimes r_2$

A	B	C
a_1	b_1	c
a_1	b_2	c
a_2	b_1	c
a_2	b_2	c

- Deoarece $r \subset r_1 \otimes r_2$, descompunerea nu este cu joncțiuni fără pierderi (*lossy decomposition*)

Întrebarea 1

Cum determinăm dacă $\{R_1, R_2\}$
este o descompunere cu joncţiuni fără
pierderi a lui R?

Întrebarea 2

Cum descompunem R *în* $\{R_1, R_2\}$ *astfel*
încât aceasta e
cu joncţiuni fără pierderi?