

## 10. Релационна алгебра

Лекционен курс "Бази от данни"

### Въведение

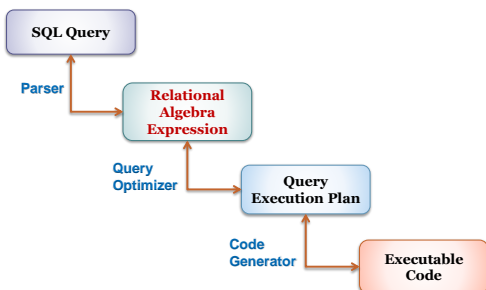
Третата и последна част на релационния модел (манипулативната част) се състои от множество от оператори, които образуват така наречената релационна алгебра.

Всеки оператор от релационната алгебра има една (унарен) или две (бинарен) релации като входни данни и връща като резултат една нова релация.

Codd дефинира 8 операции, разделени в две групи:

- ✓ множество на традиционните (класическите) операции;
- ✓ специални релационни операции.

## Къде точно в архитектурата?



## Затвореност

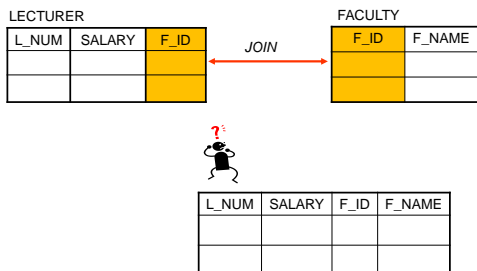
Резултатът на всяка релационна операция е друга релация - това свойство се нарича релационна затвореност.

- Така всеки изход от една операция може да бъде вход на друга операция;
- Ще е възможно да създаваме вградени изрази - т.е. операндите могат да бъдат представени посредством изрази;
- Затвореността има два аспекта:
  - Затвореност на заглавните части на релациите;
  - Затвореност на телата на релациите.

Релационната алгебра гарантира, че всички релации имат собствени заглавни части - т.е. заглавни части, в които всеки атрибут има собствено име, което е уникално в съдържащата го релация.

Заглавната част е добре дефинирана и позната на системата за базовите релации - но за релации, които се извеждат?

- Напр. LECTURER join FACULTY описва сливането на двете релации върху F\_ID – атрибута;
- Каква е заглавната част на резултата?



- Затвореността изисква тя отново да е заглавна част и системата трябва да знае каква е тя;
- Така системата трябва да има информация за такова множество от подходящи имена на атрибути, че да ни разреши да цитираме тези имена в последващи операции;
- Напр., не можем да запишем израза

```
(LECTURER join FACULTY)
where L_NUM > 30
```

ако не знаем, че в резултата от израза  
(LECTURER join FACULTY)  
има атрибут с име L\_NUM

Т.е. това, от което се нуждаем, е едно множество от правила за наследяване на имената на атрибути, създадено в алгебрата – такова, че ако знаем имената на атрибутите на входа на който да е релационен оператор, ние можем да предскажем имената на атрибутите на изхода на тази операция.

Като подготвяща стъпка да постигнем тази цел въвеждаме един нов оператор RENAME:

- Преименува атрибути в специфицираната релация;
- Връща едно ново копие на релацията, като някои атрибути са зададени с други имена;
- Напр., FACULTY rename L# as L\_NUM.

## Класически релационни оператори

Множеството на традиционните оператори включва:

- Обединение (UNION);
- Сечение (INTERSECTION);
- Разлика (DIFFERENCE);
- Произведение (CARTESIAN PRODUCT, CROSS-PRODUCT).

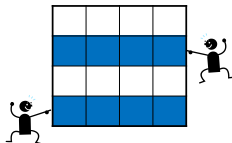
Всяка операция има два операнда, т.е. тези оператори са бинарни.

## Специални релационни оператори

Това са операторите:

- Ограничение (избор) – SELECTION (RESTRICTION) – унарен;
- Проекция (PROJECTION) – унарен;
- Естествено сливане (JOIN) – бинарен;
- Частно (DIVISION) – бинарен.

## SELECTION



**SELECT (RESTRICT)** – връща релация, която съдържа всички записи, отговарящи на специфицираните условия (предикати).

Операторът RESTRICT връща хоризонтално подмножество от релацията, т.е. редовете, които имат стойности на атрибутите си, отговарящи на зададените условия.

## SELECTION

Нека  $\Theta$  (тита) представлява някой валиден оператор за сравнение ( $=$ ,  $!=$ ,  $<$ ,  $>$ , ...). Тита-избор на релацията  $R$  върху атрибутите  $X$  и  $Y$

**$R \text{ WHERE } R.X \Theta R.Y$**

е релация със същата заглавна част като на  $R$  и тяло множеството на всички записи  $t$  на  $R$  така, че сравнението  $t.X \Theta t.Y$  е вярно за всички  $t$  (атрибутите  $X$  и  $Y$  са дефинирани върху един и същ домейн).

На мястото на всеки един атрибут може да се специфицира константна стойност.

**$R \text{ WHERE } R.X \Theta \text{constant}$**

По този начин тита-избор операторът произвежда "хоризонтално" подмножество на дадена релация, състоящо се от записите, отговарящи на специфицираното сравнение.

## SELECTION - пример

Нека имаме следната релация  $R$ , съдържаща преподаватели от различни факултети:

**$R$**

FName	LName	Title	Faculty
Стоян	Колев	Проф.	ФМИ
Петър	Иванов	Доц.	Право
Венета	Георгиева	Ст.н.с.	Икономика
Спас	Петров	Доц.	ФМИ

**$R \text{ WHERE FACULTY} = \text{'ФМИ'}$**

FName	LName	Title	Faculty
Стоян	Колев	Проф.	ФМИ
Спас	Петров	Доц.	ФМИ

**$R \text{ WHERE FACULTY} = \text{'ФМИ'} \text{ AND TITLE} = \text{'Проф.'}$**

FName	LName	Title	Faculty
Стоян	Колев	Проф.	ФМИ

## SELECTION - пример

SQL: >

```
select employee_id, first_name, salary
from employees
where salary > 11000
```

SQL Query2.sql - G:\rthelid (sa (330))

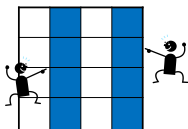
```
SELECT CustomerID, CompanyName,
Address, Country
FROM Customers
WHERE Country = 'Mexico'
```

Results

CustomerID	CompanyName	Address	Country
1	ANATOL	Ave Tule Escondido y Hidalgo	Mexico
2	ANTON	Antonio Moreno Taqueria	Mexico
3	CENTC	Centro comercial Modurama	Mexico
4	FBIIC	Ferdinand Comedias	Mexico
5	FORTU	Tortuga Restaurante	Mexico

Query: GCHOLAROV-IMPUSCULEXPRESS - sa (330) Northwind 00:00:00 5 rows

## PROJECTION



**PROJECT** - връща релация, която съдържа всички записи (без дубликати) със специфицирани атрибути от дадена релация.

Проекцията на релацията R върху атрибутите X, Y, ..., Z

$R[X, Y, \dots, Z]$

е релация със заглавна част  $\{X, Y, \dots, Z\}$  и тяло множеството от всички записи  $\{X:x, Y:y, \dots, Z:z\}$  така, че един запис  $t$  се появява в R с X-стойност  $x$ , Y-стойност  $y$ , ..., Z-стойност  $z$ .

По този начин проекцията създава "вертикално" подмножество на дадена релация – т.е. това подмножество се получава чрез избор на специфични атрибути в специфициран ред.

Проекцията предоставя начин за пренаареждане на атрибутите в една релация.

## PROJECTION - пример

Нека имаме следната релация R, съдържаща преподаватели от различни факултети:

R

FName	LName	Title	Faculty
Стоян	Колев	Проф.	ФМИ
Петър	Иванов	Доц.	Право
Венета	Георгиева	Ст.н.с.	Икономика
Спас	Петров	Доц.	ФМИ

R [FName, LName]

FName	LName
Стоян	Колев
Петър	Иванов
Венета	Георгиева
Спас	Петров

R [Faculty, Title, FName]

Faculty	Title	FName
ФМИ	Проф.	Стоян
Право	Доц.	Петър
Икономика	Ст.н.с.	Венета
ФМИ	Доц.	Спас

## PROJECTION - пример

SQL Window - select last\_name, first\_name from employees

	LAST_NAME	FIRST_NAME
1	Abel	Ellen
2	Ade	Bunster
3	Adams	Mozhe
4	Austin	David
5	Baer	Hermann
6	Baida	Shelli
7	Banda	Amel
8	Baer	Elizabeth
9	Beil	Sarah
10	Bernstein	David
11	Bost	Laura
12	Bloom	Harrison
13	Bull	Alexis
14	Calero	Anthony
15	Cambraut	Gerald
16	Cambraut	Nanette
17	Chen	John

SQL Query Window - SELECT Country, CompanyName, Phone, ContactName FROM Customers

	Country	CompanyName	Phone	ContactName
1	Germany	Alfreds Futterkiste	030-0074321	Maria Anders
2	Mexico	Ana Trujillo Emparedados y helados	(5) 555-4729	Ana Trujillo
3	Mexico	Antonio Moreno Taqueria	(5) 555-3892	Antonio Moreno
4	UK	Around the Horn	(171) 555-7788	Thomas Hardy
5	Sweden	Berglunds snabbköp	0821-12 34 56	Christina Berglund
6	Germany	Blaug's Spezialitäten	0621-0940	Hanna Christ
7	France	Bonaparte et fils	08 62 15 31	Fredérique Citeaux
8	Spain	Bonito Comidas preparadas	91 555 22 82	Martin Sommer
9	France	Bon app	91 24 45 40	Laurence Labarre
10	Canada	Bonito-Dollar Markets	(504) 555-4729	Elizabeth Linch
11	UK	B's Beverages	(171) 555-1212	Victoria Ashworth
12	Argentina	Cactus Comidas para llevar	(1) 135-5555	Pedro Simpson
13	Mexico	Centro comercial Moduruna	(5) 555-3392	Rafael Chang
14	Switzerland	Chop-suey Chinese	(043) 076545	Yang Wang
15	Brazil	Comércio Mineiro	(11) 555-7647	Pedro Alonso

## PROJECTION

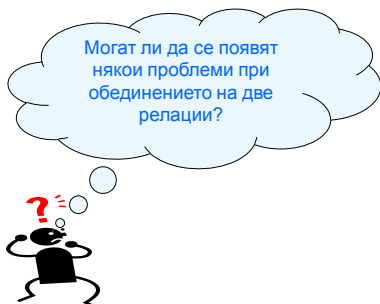
Забележки:

- Не можем да специфицираме един атрибут повече от веднъж;
- Когато всички атрибути на релацията бъдат указани, проекцията се нарича *идентична*;
- Проекция във формат  $R \bowtie$  се нарича *нулева проекция*.

## Съвместимост на типове

Нека разгледаме обединението:

- В математиката обединението на две множества е множеството на всички елементи, принадлежащи към едно от двете оригинални множества;
- Понеже релацията е множество от записи – възможно е да конструираме обединението на две релации.



FName	LName	Title	Faculty
Стоян	Копев	Проф.	ФМИ
Петър	Иванов	Доц.	Право
Венета	Георгиева	Ст.н.с.	Икономика
Спас	Петров	Доц.	ФМИ

Обединение

Subject	Horarium	Hall
Бази от данни	30	422
Изкуствен интелект	20	424

Резултат:

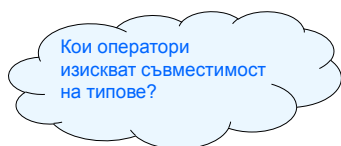
?	?	?	?
Стоян	Копев	Проф.	ФМИ
Петър	Иванов	Доц.	Право
Изкуствен интелект	20	424	
Венета	Георгиева	Ст.н.с.	Икономика
Спас	Петров	Доц.	ФМИ
Изкуствен интелект	20	424	

- Въпреки, че резултатът е множество от редове, той не е **релация**!
- Релацията не може да съдържа смесени типове записи!
- Искаме резултатът да е релация, за да запазим свойството **затвореност**;
- Следователно, обединението в релационната алгебра не е идентично с математическото обединение;
- По-скоро то е специален случай, при който изискваме двете входни релации да бъдат от **един и същ тип**.



Съвместимост на типове: двете релации да имат идентични заглавни части - т.е.:

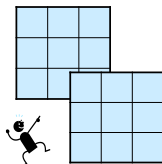
- Да имат еднакви множества от имена на атрибутите;
- Кореспондиращите атрибути да са дефинирани върху еднакви области.



Съвместимост по типове се изисква за операторите:

- Обединение
- Сечение
- Разлика

## UNION



**UNION** - създава релация, която се състои от всички записи, които се появяват във всяка една или и в двете релации.

Обединението на две релации А и В със съвместими типове поражда трета релация със:

- заглавна част като на А и В (трябва да бъдат съвместими);
- тяло – множеството от всички записи, принадлежащи на А, В или на двете, като дубликатите се елиминират.

## UNION - пример 1

Нека имаме следните релации А и В, съдържащи преподавателите във факултети на ПУ и на СУ:

**A**

Name	Faculty
Стоян Копев	ФМИ
Петър Иванов	Филология
Венета Георгиева	Право

**B**

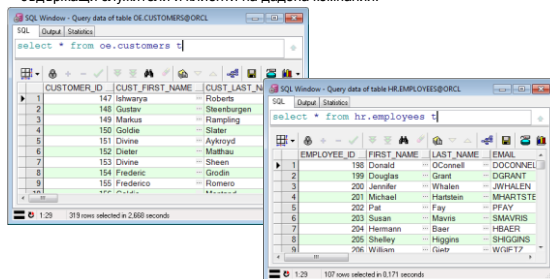
Name	Faculty
Десислава Янева	ФМИ
Стоян Копев	ФМИ

**A UNION B**

Name	Faculty
Стоян Копев	ФМИ
Петър Иванов	Филология
Десислава Янева	ФМИ
Венета Георгиева	Право

## UNION - пример 2

Нека имаме следните релации EMPLOYEES и CUSTOMERS, съдържащи служители и клиенти на дадена компания.



## UNION - пример 2

CUSTOMERS UNION EMPLOYEES

SQL Window - select c.customer\_id as id, c.cust\_first\_name as fname, c.cust\_last\_name from oe.customers c uni...

```

select c.customer_id as id, c.cust_first_name as fname, c.cust_last_name
from oe.customers c
union
select e.employee_id, e.first_name, e.last_name
from hr.employees e
  
```

ID	FNAME	CUST_LAST_NAME
1	Steven	King
2	Constantin	Wallace
3	Neena	Kochhar
4	Hermann	Baer
5	Lex	De Haan
6	Alexander	Hunold
7	Martha	Taylor
8	Bruce	Ernst
9	Hermann	Schneider
10	David	Austin
11	Mathias	MacGraw
12	Mathias	Hartmann
13	Valli	Pataballa
14	Dora	Reade

426 rows selected in 0.243 seconds

## UNION ALL - пример 3

CUSTOMERS UNION EMPLOYEES

SQL Window - select c.cust\_first\_name as fname fr...

```

select c.cust_first_name as fname
from oe.customers c
union
select e.first_name
from hr.employees e
order by 1

```

FNAME	
1	Adam
2	Ajay
3	Alan
4	Alan
5	Alana
6	Albert
7	Alberto
8	Alec
9	Alexander
10	Alicia

240 rows selected in 0.14 seconds

CUSTOMERS UNION ALL EMPLOYEES

SQL Window - select c.cust\_first\_name as fname fr...

```
 select c.cust_first_name as fname from oe.customers c union all select e.first_name from hr.employees e order by 1   
```

FNAME	
1	Adam
2	Ajay
3	Ajay
4	Alan
5	Alan
6	Alan
7	Alan
8	Alan
9	Alana
10	Albert

426 rows selected in 0.203 seconds

## UNION ALL - пример с T-SQL

SQL Query - 6\_rhatched (sa (S))

```

SELECT Country FROM Customers
UNION ALL
SELECT Country FROM Employees

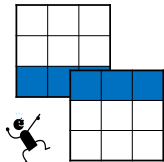
```

Results Messages

	Country
1	Germany
2	Mexico
3	Mexico
4	UK
5	Sweden
6	Germany
7	France
8	Spain
9	France
10	Canada
11	UK
12	Argentina
13	Mexico
14	Switzerland
15	Brazil

Query ex... | GCHOLAKOV-UNPC\SQLEXPRESS... | sa (S) | Northwind | 00:00:00 | 390 rows

## INTERSECTION



**INTERSECTION** – създава нова релация, състояща се от всички записи, които се появяват в двете релации едновременно.

Сечението на две релации А и В със съвместими типове поражда трета релация със:

- заглавна част като на А и В (трябва да бъдат съвместими);
- тяло – множеството от всички записи, принадлежащи едновременно на А и В, т.е. общите за двете релации n-торки (записи).

## INTERSECTION - пример 1

Нека имаме следните релации А и В, съдържащи преподавателите във факултетите на ПУ и на СУ:

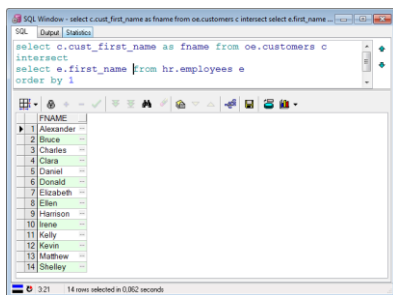
A	Name	Faculty	B	Name	Faculty
	Стоян Копев	ФМИ		Десислава Янева	ФМИ
	Петър Иванов	Филология		Стоян Копев	ФМИ
	Венета Георгиева	Право			

**A INTERSECTION B**

Name	Faculty
Стоян Копев	ФМИ

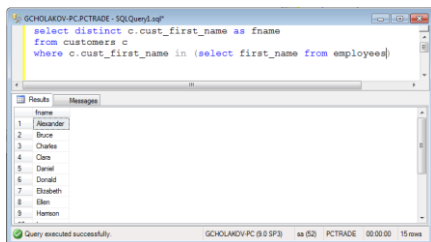
## INTERSECTION - пример 2

**CUSTOMERS INTERSECT EMPLOYEES**



## INTERSECTION - пример 3

Някои СУБД не поддържат оператор INTERSECT. При тях може да се използва алтернативен код.



```

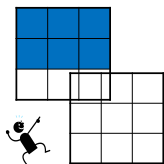
GOCHOLANDVPC\CTRADE - SQLQuery1.sql
select distinct c.cust_first_name as fname
from customers c
where c.cust_first_name in (select first_name from employees)
  
```

Results

Name
1. Alexander
2. Bruce
3. Charles
4. Chris
5. Daniel
6. Donald
7. Elisabeth
8. Ellen
9. Hartson

Query executed successfully. GOCHOLANDVPC\SS SP3 ss (5) CTRADE 00:00:00 15 rows

## DIFFERENCE (MINUS)



**DIFFERENCE** – създава релация, състояща се от всички записи, които се появяват в първата, но не и във втората релация.

Разликата на две релации A и B със съвместими типове поражда трета релация със:

- заглавна част като на A и B (трябва да бъдат съвместими);
- тяло – множеството от всички записи, принадлежащи на A и не принадлежащи на B.

## DIFFERENCE - пример 1

Нека имаме следните релации A и B, съдържащи преподавателите във факултетите на ПУ и на СУ:

**A**

Name	Faculty
Стоян Колев	ФМИ
Петър Иванов	Филология
Венета Георгиева	Право

**B**

Name	Faculty
Десислава Янева	ФМИ
Стоян Колев	ФМИ

**A MINUS B**

Name	Faculty
Венета Георгиева	Право
Петър Иванов	Филология

**B MINUS A**

Name	Faculty
Десислава Янева	ФМИ

## DIFFERENCE - пример 2

CUSTOMERS MINUS EMPLOYEES

SQL Window - select c.cust\_first\_name as fname from oe.customers c minus select e.first\_name from hr.employees e

FNAME
1 Ajay
2 Alan
3 Alan
4 Albert
5 Alec
6 Alfred
7 Ali
8 Alice
9 Aly
10 Alonso
11 Amanda
12 Amish
13 Ben
14 Ben

5:20 157 rows selected in 0.109 seconds

EMPLOYEES MINUS CUSTOMERS

SQL Window - select e.first\_name from hr.employees e minus select c.cust\_first\_name as fname from oe.customers c

FIRST_NAME
1 Adam
2 Ales
3 Alberto
4 Ales
5 Alan
6 Alyssa
7 Ann
8 Anthony
9 Briany
10 Christopher
11 Curtis
12 Danielle
13 David
14 P...

5:20 77 rows selected in 0.109 seconds

## DIFFERENCE - пример 3

Пример за алтернативен код:

SQL Window - select distinct c.cust\_first\_name as fname from oe.customers c where c.cust\_first\_name not in (select first\_name from hr.employees) order by c.cust\_first\_name

FNAME
1 Ajay
2 Alan
3 Alan
4 Albert
5 Alec
6 Alfred
7 Ali
8 Alice
9 Aly
10 Alonso
11 Amanda
12 Amish

4:27 157 rows selected in 0.124 seconds

## DIFFERENCE - пример T-SQL

Някои СУБД не поддържат оператор MINUS или използват друга ключова дума – EXCEPT в T-SQL:

SQL Window - SELECT CUST\_FIRST\_NAME FROM CUSTOMERS EXCEPT SELECT FIRST\_NAME FROM EMPLOYEES

CUST_FIRST_NAME
1 Ajay
2 Alan
3 Alan
4 Albert
5 Alec
6 Alfred
7 Ali
8 Alice
9 Aly
10 Alonso
11 Amanda
12 Amish
13 Ben

SQL Window - SELECT FIRST\_NAME FROM EMPLOYEES EXCEPT SELECT CUST\_FIRST\_NAME FROM CUSTOMERS

FIRST_NAME
1 Adam
2 Ales
3 Alberto
4 Ales
5 Alan
6 Alyssa
7 Ann
8 Anthony
9 Briany
10 Christopher
11 Curtis
12 Danielle
13 David

## SELECTION - ревизирана

Дефинираната по-рано операция тита-избор (избор) разрешава само прости сравнение в WHERE клаузата, но е възможно дефиницията да се разшири с описаните след това оператори, както е показано чрез следните равенства:

- $R \text{ WHERE } c1 \text{ AND } c2$

е дефинирана като еквивалентна на

$(R \text{ WHERE } c1) \text{ INTERSECTION } (R \text{ WHERE } c2)$

- $R \text{ WHERE } c1 \text{ OR } c2$

е еквивалентна на

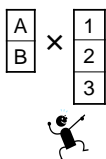
$(R \text{ WHERE } c1) \text{ UNION } (R \text{ WHERE } c2)$

- $R \text{ WHERE NOT } c$

е еквивалентна на

$R \text{ MINUS } (R \text{ WHERE } c)$

## PRODUCT



A	1
B	2
	3

PRODUCT - създава нова релация от две дадени релации, която се състои от всички възможни конкатенирани двойки от записи от двете релации.

Произведението на две релации A и B поражда трета релация със:

- заглавна част обединението на заглавните части на A и B;
- тяло – множеството от всички записи t, където t е обединението на всеки запис от A с всеки от B.

## PRODUCT

Произведението на две релации често се нарича Декартово, защото резултатът е релация с тяло множеството на всички подредени двойки записи от двете релации.

Но за да се запази свойството затвореност, т.е. резултатът от операцията отново да е релация, трябва резултатната релация да съдържа записи, а не подредени двойки от записи.

В релационната алгебра Декартовото произведение е една разширена форма на операцията, където всяка подредена двойка от записи се заменя от отделен запис, който представлява обединение (конкатенация) на двата записа.

Т.е., ако са дадени два записа  $\{A_i; a_1, \dots, A_m; a_m\}$  и  $\{B_j; b_1, \dots, B_n; b_n\}$  обединението има вида  $\{A_i; a_1, \dots, A_m; a_m, B_j; b_1, \dots, B_n; b_n\}$

**Степен на резултата** – сумата от степените на A и B.

**Кардиналност на резултата** – произведението от кардиналностите на A и B.

## PRODUCT - пример 1

Нека имаме следните релации А и В, съдържащи съответно преподаватели и университети, в които те преподават:

**A**

UNI_ID	FName	LName	Title
101	Стоян	Копев	Проф.
102	Петър	Иванов	Доц.
101	Венета	Георгиева	Ст.н.с.

**B**

UNI_ID	Name	Short
101	Пловдивски университет	ПУ
102	Софийски университет	СУ

**A × B**

UNI_ID	FName	LName	Title	UNI_ID	Name	Short
101	Стоян	Копев	Проф.	101	Пловдивски университет	ПУ
101	Стоян	Копев	Проф.	102	Софийски университет	СУ
102	Петър	Иванов	Доц.	101	Пловдивски университет	ПУ
102	Петър	Иванов	Доц.	102	Софийски университет	СУ
101	Венета	Георгиева	Ст.н.с.	101	Пловдивски университет	ПУ
101	Венета	Георгиева	Ст.н.с.	102	Софийски университет	СУ

## PRODUCT - пример 2

SQL Window - Query data of table regions t

```
select * from regions t
```

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa

×

SQL Window - select \* from countries t

```
select * from countries t
```

COUNTRY_ID	COUNTRY_NAME	REGION_ID
1	AR Argentina	2
2	AU Australia	3
3	BE Belgium	1
4	BR Brazil	2
5	CA Canada	2
6	CH Switzerland	1
7	CN China	3
8	DE Germany	1
9	DK Denmark	1
10	EG Egypt	4
11	FR France	1
12	IL Israel	4
13	IN India	3
14	IT Italy	1
15	JP Japan	3
16	KW Kuwait	4

## PRODUCT - пример 2

=

SQL Window - select \* from countries, regions order by country\_id

```
select *
from countries, regions
order by country_id
```

COUNTRY_ID	COUNTRY_NAME	REGION_ID	REGION_ID	REGION_NAME
1	AR Argentina	2	1	Europe
2	AR Argentina	2	2	Americas
3	AR Argentina	2	3	Asia
4	AR Argentina	2	4	Middle East and Africa
5	AU Australia	3	1	Europe
6	AU Australia	3	2	Americas
7	AU Australia	3	3	Asia
8	AU Australia	3	4	Middle East and Africa
9	BE Belgium	1	2	Americas
10	BE Belgium	1	3	Asia
11	BE Belgium	1	4	Middle East and Africa
12	BE Belgium	1	1	Europe
13	BR Brazil	2	3	Asia
14	BR Brazil	2	4	Middle East and Africa
15	BR Brazil	2	1	Europe
16	BR Brazil	2	2	Americas
17	CA Canada	2	4	Middle East and Africa
18	CA Canada	2	1	Europe
19	CA Canada	2	2	Americas
20	CA Canada	2	3	Asia



## PRODUCT

Забележки:

- Възможно е да възникне проблем – ако заглавните части на двете релации имат атрибути с еднакви имена – тогава трябва да използваме оператора RENAME за преименуване на тези атрибути;
- От примера се вижда, че резултатната релация съдържа доста неверни данни и ще се наложи да се изпълнят допълнителни операции, за да се извлекат само значещите данни.

## Алгебрични свойства на релационните оператори

Асоциативност:

◦ UNION (обединение)

$$(A \cup B) \cup C \Leftrightarrow A \cup (B \cup C)$$

◦ INTERSECTION (сечение)

$$(A \cap B) \cap C \Leftrightarrow A \cap (B \cap C)$$

◦ PRODUCT (произведение)

$$(A \times B) \times C \Leftrightarrow A \times (B \times C)$$

Комутативност:

◦ UNION (обединение):  $A \cup B = B \cup A$

◦ INTERSECTION (сечение):  $A \cap B = B \cap A$

◦ PRODUCT (произведение):  $A \times B \Leftrightarrow B \times A$

## JOIN

Операцията join е една от най-полезните в релационната алгебра и е най-често използвания способ за комбиниране на данни от две или повече релации.

Въпреки че join операцията може да бъде представена като Декартово произведение, последвано от операциите selection и projection, нейното използване в практиката се среща много често.

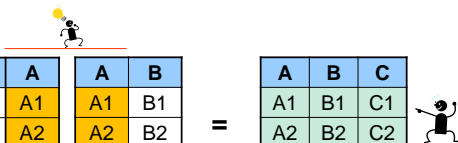
Още повече, резултатът от Декартовото произведение обикновено е много по-голям от резултата, получен от join. Но все пак относно въпроси, касаещи ефективността на извличане на данни, е трудно да се отговори еднозначно кой от двата подхода би бил по-ефективен – това зависи много от конкретната приложна област, т.е. от данните, техните типове, наличието на индекси и т.н.

## JOIN

Ще разгледаме следните видове JOIN операции:

- Natural join
- Theta ( $\Theta$ ) join
- Equi-join
- Semi-join
- Anti-join
- Outer join

## NATURAL JOIN



C	A	A	B
C1	A1	A1	B1
C2	A2	A2	B2
C3	A3	A3	B3

=

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

**NATURAL JOIN** (естествено сливане) - създава нова релация от двете дадени релации, която съдържа всички възможни свързани двойки от записи, (по един от всяка релация), така че всяка двойка да има равенство на стойностите във **всички атрибути с еднакви имена** от двете релации.

## NATURAL JOIN

Нека са дадени релациите A и B със заглавни части  $\{X_1, \dots, X_m, Y_1, \dots, Y_n\}$  и  $\{Y_1, \dots, Y_n, Z_1, \dots, Z_p\}$ , т.е. атрибутите  $\{Y_1, \dots, Y_n\}$  са общи за двете релации. Допускаме, че общите атрибути са дефинирани върху общи домейни.

По-нататък ще разглеждаме:

- $\{X_1, \dots, X_m\}$  – като X;
- $\{Y_1, \dots, Y_n\}$  – като Y;
- $\{Z_1, \dots, Z_p\}$  – като Z.

## NATURAL JOIN

Естественото сливане (A NATURAL JOIN B) е релация със:

- ✓ заглавна част {X, Y, Z};
- ✓ тяло – множеството на всички записи {X:x, Y:y, Z:z} с равенство в общите атрибути така, че един запис се появява в A с X-стойностите и с Y-стойностите, а в B – с Y-стойностите и Z-стойностите.

JOIN операторът е:

- ✓ асоциативен:  
(A JOIN B) JOIN C  $\Leftrightarrow$  A JOIN (B JOIN C)  $\Leftrightarrow$  A JOIN B JOIN C;
- ✓ комутативен: A JOIN B  $\Leftrightarrow$  B JOIN A

## NATURAL JOIN - пример

SQL Window - select \* from countries natural join regions order by country\_id

```
select *
from countries natural join regions
order by country_id
```

REGION_ID	COUNTRY_ID	COUNTRY_NAME	REGION_NAME
1	2 AR	Argentina	Americas
2	3 AU	Australia	Asia
3	1 BE	Belgium	Europe
4	2 BR	Brazil	Americas
5	2 CA	Canada	Americas
6	1 CH	Switzerland	Europe
7	3 CN	China	Asia
8	1 DE	Germany	Europe
9	1 DK	Denmark	Europe
10	4 EG	Egypt	Middle East and Africa
11	1 FR	France	Europe
12	4 IL	Israel	Middle East and Africa
13	3 IN	India	Asia

region\_id, number, optional: Region ID for the country. Foreign key to region\_id column in R

Забележки:

- Не всички SQL езици поддържат синтаксиса за natural join;
- От фигурата се вижда, че атрибутът REGION\_ID участва само веднъж в резултата, за разлика от примера с PRODUCT.

## Θ-JOIN

Нека релациите A и B нямат общи атрибути и нека  $\Theta$  е валиден оператор за сравнение (>, <, >=, <=, <>). Тогава  $\Theta$ -join на релацията A върху атрибута X с релацията B върху атрибута Y е резултатът от изпълнението на израза

**(A  $\times$  B) WHERE X  $\Theta$  Y**

т.е. резултатната релация е със:

- заглавна част – като на Декартовото произведение на A и B, т.е. обединението на заглавните части на A и B;
- тяло – множеството на всички записи, принадлежащи на Декартовото произведение, за които X  $\Theta$  Y е вярно.

## Θ-join - пример 1

Нека имаме следните релации А и В, съдържащи съответно преподаватели и университети, в които те преподават и нека в случая Θ е операторът за сравнение '='. Тогава резултатната релация ще съдържа само записите от Декартовото произведение, оцветени в жълто:

**A**

TEACH_ID	FName	LName	Title
101	Стоян	Колев	Проф.
102	Петър	Иванов	Доц.
101	Венета	Георгиева	Ст.н.с.

**B**

UNI_ID	Name	Short
101	Пловдивски университет	ПУ
102	Софийски университет	СУ

(A × B) WHERE A.TEACH\_ID = B.UNI\_ID

TEACH_ID	FName	LName	Title	UNI_ID	Name	Short
101	Стоян	Колев	Проф.	101	Пловдивски университет	ПУ
101	Стоян	Колев	Проф.	102	Софийски университет	СУ
102	Петър	Иванов	Доц.	101	Пловдивски университет	ПУ
102	Петър	Иванов	Доц.	102	Софийски университет	СУ
101	Венета	Георгиева	Ст.н.с.	101	Пловдивски университет	ПУ
101	Венета	Георгиева	Ст.н.с.	102	Софийски университет	СУ

## Θ-join - пример 2

The screenshot shows two SQL queries in a window. The top query is:

```
select *
from countries c join regions r on c.region_id = r.region_id
order by 1
```

The bottom query is:

```
select *
from countries c, regions r
where c.region_id = r.region_id
order by 1
```

Both queries result in a table with columns COUNTRY\_ID, COUNTRY\_NAME, REGION\_ID, and REGION\_NAME. The bottom query's result is highlighted in yellow.

## EQUI-JOIN

Това е частен случай на Θ-join, в който операторът Θ е само операторът за сравнение =.

## SEMI-JOIN

- Връща редовете от първата релация, за които има поне един съвпадащ от втората релация;
- Разликата между него и досега описаните е, че редовете от първата релация ще участват в резултата най-много по веднъж;
- Дори втората релация да има два съвпадащи за ред от първата, само едно копие на реда ще бъде върнато в резултата;
- Реализира се с предикатите EXISTS или IN.

## SEMI-JOIN - пример

Да предположим, че имаме таблиците DEPT и EMP, които съдържат отдели и служители и нека във всеки отдел има поне един служител.

SQL\*Plus - Query data of table DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SQL\*Plus - Query data of table SCOTT.EMP@ORCL

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-12-1980	800.00		20
7499	ALLEN	SALESMAN	7698	20-2-1981	1600.00	300.00	30
7521	WARD	SALESMAN	7698	22-2-1981	1200.00	500.00	30
7566	JONES	MANAGER	7539	2-4-1981	2975.00		20
7654	MARTIN	SALESMAN	7698	25-9-1981	1200.00	1400.00	30
7698	BLAKE	MANAGER	7539	1-5-1981	2850.00		30
7782	CLARK	MANAGER	7539	9-6-1981	2450.00		10
7788	SCOTT	ANALYST	7566	19-4-1987	3000.00		20
7839	KING	PRESIDENT	17-11-1981	5000.00		10	
7844	TURNER	SALESMAN	7698	8-9-1981	1500.00	0.00	30
7876	ADAMS	CLERK	7788	23-5-1987	1100.00		20
7900	JAMES	CLERK	7698	3-12-1981	950.00		30
7902	FORD	ANALYST	7566	3-12-1981	3000.00		20
7934	MILLER	CLERK	7782	23-1-1982	1300.00		10

## SEMI-JOIN - пример

Да покажем всички отдели, които имат поне един служител.

SQL\*Plus - SELECT D.deptno, D.dname FROM dept D, emp E WHERE E.deptno = D.deptno ORDER BY D.deptno

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

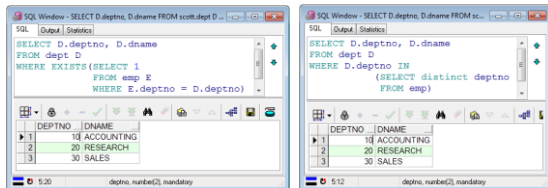
Ясно се вижда, че редовете за отделите се повтарят толкова пъти, колкото служители има в отдел.

Също се вижда, че отдел с DEPTNO = 40 не участва в резултата, защото няма служители в този отдел.

Дубликатите могат да бъдат елиминирани с ключовата дума DISTINCT, но това ще повлияе само на визуализацията им, не и на ефективността на извличане на данните.

## SEMI-JOIN - пример

Следната заявка ще извлече същите данни, но без дубликати и по-ефективно.



Тук дали отделаг има 1 или 1000 служителя няма значение, защото системата ще спира да търси служители за отдела още щом намери първия, вместо да търси всички служители.

В тази заявка се проверява стойността на всеки ред в deptno дали е измежду тези, върнати от вложената заявка.

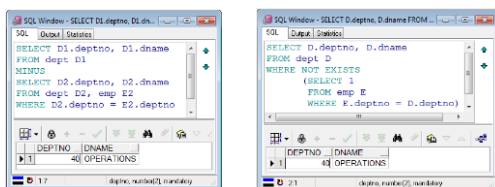
## ANTI-JOIN

Операторът “anti-join” между две релации върши обратното на semi-join: връща редовете от първата релация, които нямат съпадения във втората.

По своята природа това е операторът за разлика (minus), но може да бъде реализиран и с предикатите NOT EXISTS или NOT IN.

## ANTI-JOIN

Следните две заявки извличат по двата начина отделите, които нямат служители.



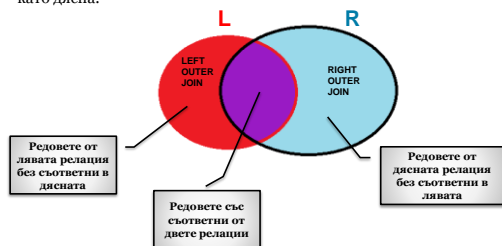
## INNER JOIN

Изброените дотук видове join операции (natural join,  $\Theta$ -join, equi join) реализират т.нар. вътрешни съединения, характерни с това, че в резултатът участват само редовете от двете релации, които имат съпадения.

За случаите, в които ще се налага от една от двете или и от двете релации да бъдат запазени всички редове в резултатната релация, се използват външни съединения.

## OUTER JOIN

Външното съединение генерира релация, в която записите, които нямат съпадения в двете релации, могат също да бъдат запазени в резултата. Нека за примерите използваме релацията L като лява, а R като дясна.



## OUTER JOIN

Видове:

1. **Left Outer Join:** съединение, в което записите от L, които нямат съответни в R (сравнение в общите атрибути), също ще участват в резултатната релация.
2. **Right Outer Join:** съединение, в което записите от R, които нямат съответни в L, също ще участват в резултатната релация.
3. **Full Outer Join:** съединение, в което записите от L, които нямат съответни в R, ще участват в резултатната релация, както и тези от R, които нямат съответни в L, също ще участват в резултатната релация.

## LEFT OUTER JOIN - пример

SQL Window - select \* from countries c left join regions r on c.region\_id = r.region\_id order by 2

```

select *
from countries c left outer join regions r
on c.region_id = r.region_id
order by 2

```

COUNTRY_ID	COUNTRY_NAME	REGION_ID	REGION_ID	REGION_NAME
1	AD		2	Americas
2	AU		3	Asia
3	BE		1	Europe
4	BR		2	Americas
5	BG			
6	CA		2	Americas
7	CH		3	Asia
8	DK		1	Europe
9	EG		4	Middle East and Africa
10	FR		1	Europe
11	DE		1	Europe
12	IN		3	Asia
13	IL		4	Middle East and Africa
14	IT		1	Europe

26 rows selected in 0.032 seconds

## RIGHT OUTER JOIN - пример

SQL Window - select \* from countries c right outer join regions r on c.region\_id = r.region\_id order by 2 desc

```

select *
from countries c right outer join regions r
on c.region_id = r.region_id
order by 2 desc

```

COUNTRY_ID	COUNTRY_NAME	REGION_ID	REGION_ID	REGION_NAME
1			5	North Pole
2	ZW		4	Middle East and Africa
3	ZM		4	Middle East and Africa
4	US		2	Americas
5	UK		1	Europe
6	CH		1	Europe
7	SG		3	Asia
8	NG		4	Middle East and Africa
9	NL		1	Europe
10	MX		2	Americas
11	MY		3	Asia
12	KW		4	Middle East and Africa
13	JP		3	Asia
14	IT		1	Europe

country\_id, country\_name: Primary key of countries table

## FULL OUTER JOIN - пример

SQL Window - select \* from countries c full outer join regions r on c.region\_id = r.region\_id order by c.region\_id

```

select *
from countries c full outer join regions r
on c.region_id = r.region_id
order by c.region_id desc

```

COUNTRY_ID	COUNTRY_NAME	REGION_ID	REGION_ID	REGION_NAME
1	BG			
2			5	North Pole
3	EG		4	Middle East and Africa
4	ZM		4	Middle East and Africa
5	NG		4	Middle East and Africa
6	KW		4	Middle East and Africa
7	IL		4	Middle East and Africa
8	ZW		4	Middle East and Africa
9	CH		3	Asia
10	AU		3	Asia
11	SG		3	Asia
12	MY		3	Asia
13	JP		3	Asia
14	IN		3	Asia

27 rows selected in 0.032 seconds



## Частно

A	B	B	A
A1	X	X	A1
A1	Y	Z	
A1	Z		
A2	X		
A2	Y		

DIVIDE – от две релации генерира нова релация, която съдържа всички стойности на атрибута A от първата релация, които съответстват (равни в другия атрибут B) на всички стойности на атрибута B от втората релация.

## Частно

Нека релациите A и B имат заглавни части  $\{X_1, \dots, X_m, Y_1, \dots, Y_n\}$  и  $\{Y_1, \dots, Y_n\}$ , т.е. атрибутите  $\{Y_1, \dots, Y_n\}$  са общи за двете релации, а B няма други атрибути. Допускаме, че общите атрибути са дефинирани върху общи домейни. Нека разглеждаме  $\{X_1, \dots, X_m\}$  – като X,  $\{Y_1, \dots, Y_n\}$  – като Y.

A DIVIDE B е релация със:

- заглавна част {X};
- тяло – множеството на всички записи {X:x} така, че един запис {X:x, Y:y} се появява в A за всички записи {Y:y}, появяващи се в B.

Накратко:

- операторът за деление дели една релация A от степен  $m+n$  на друга релация B от степен  $n$  и създава нова релация от степен  $m$ ;
- $(m+i)$ -тият атрибут на A и  $i$ -тият атрибут на B трябва да са дефинирани върху един и същ домейн.

## Частно - пример 1

Искаме да извлечем всички студенти, които са завършили поставените им задачи по Базис от данни и ще бъдат допуснати до изпит.

Completed

Student	Task
Стоян Копев	БД - самостоятелна работа
Стоян Копев	БД – упражнения
Венета Георгиева	БД – упражнения
Венета Георгиева	Компилятор
Венета Георгиева	БД - самостоятелна работа
Иван Пенев	БД – упражнения
Иван Пенев	Компютърна графика

Projects

Task
БД - самостоятелна работа
БД – упражнения

÷

=

Student
Стоян Копев
Венета Георгиева

## Частно - пример 2

Да извлечем всички преподаватели, които изнасят лекции едновременно в ПУ и СУ.

Lecturers

LEC	UNI
Стоян Копев	ПУ
Петър Иванов	СУ
Венета Георгиева	ТУ
Стоян Копев	СУ
Петър Иванов	ПУ
Петър Иванов	ТУ
Венета Георгиева	СУ

÷

Universities

UNI
ПУ
СУ

=

LEC
Стоян Копев
Петър Иванов

## Частно - пример 2-1

Table: lec

LEC	UNI
Стоян Копев	ПУ
Петър Иванов	СУ
Венета Георгиева	ТУ
Стоян Копев	СУ
Петър Иванов	ПУ
Петър Иванов	ТУ
Венета Георгиева	СУ

Table: uni

UNI
ПУ
СУ

```

GCHOLAKOV-PC-Division - SQLQuery3.asp
SELECT DISTINCT lec
FROM lecturers AS L1
WHERE NOT EXISTS
  (SELECT *
   FROM universities
   WHERE NOT EXISTS
     (SELECT *
      FROM lecturers AS L2
      WHERE L1.lec = L2.lec
      AND L2.uni = universities.uni))

```

Results

lec
Петър Иванов
Стоян Копев

Query executed: GCHOLAKOV-PC (0.0 SP3) sa (55) Division 00:00:00 2 rows

## Частно - пример 2-2

Table: lec

LEC	UNI
Стоян Копев	ПУ
Петър Иванов	СУ
Венета Георгиева	ТУ
Стоян Копев	СУ
Петър Иванов	ПУ
Петър Иванов	ТУ
Венета Георгиева	СУ

Table: uni

UNI
ПУ
СУ

```

GCHOLAKOV-PC-Division - SQLQuery4.asp
SELECT L1.lec
FROM lecturers AS L1, universities AS U
WHERE L1.uni = U.uni
GROUP BY L1.lec
HAVING COUNT(L1.uni) = (SELECT COUNT(uni) FROM universities);

```

Results

lec
Петър Иванов
Стоян Копев

Query executed successfully: GCHOLAKOV-PC (0.0 SP3) sa (55) Division 00:00:00 2 rows

26

### Частно - пример 3

Искаме да извлечем всички доставчици, които доставят частите P1 и P3 едновременно.

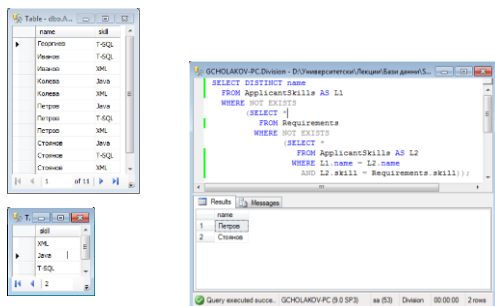
Suppliers_Parts		Parts		=	Supp
Supp	Part	Part			Supp
S1	P1	P1			S1
S2	P1	P3			S4
S4	P1				
S1	P3				
S3	P4				
S4	P4				
S4	P3				

### Частно - пример 4

A	Supp	Part	B1	Part	A / B1	Supp
	S1	P1		P2		S1
	S1	P2				S2
	S1	P3	B2	Part		S3
	S1	P4		P2		S4
	S2	P1		P4	A / B2	Supp
	S2	P2				S1
	S3	P2	B3	Part		S4
	S4	P2		P1		
	S4	P4		P2	A / B3	Supp
				P4		S1

### Частно - пример 5-1

## Частно - пример 5-2



## За какво ни е релационната алгебра?

Демонстрираните примери бяха предимно за извличане на данни, но това не означава, че релационната алгебра е приложима само при извличане. Нейната основна цел е да позволи **писането на изрази**, които да послужат за:

- Дефиниране на обхват за извличани данни – задаване на условия, на които да отговарят резултатите;
- Дефиниране на обхват за промяна на данни – при въвеждане, промяна и изтриване;
- Дефиниране на (именувани) виртуални релации – изгледи, напр.;
- Дефиниране на правила за сигурност;
- Дефиниране на правила за цялостност;
- И др.

## Разширение и сумиране

Много изследователи предлагат нови алгебрични оператори, като допълнение на тези на д-р Codd.

Ще разгледаме два такива:

- **EXTEND**
- **SUMMARIZE**

## EXTEND

В много случаи искаме да включим в резултатната релация нови атрибути, стойностите на които са резултат от някакви изчисления.

Например: да включим атрибут, който съдържа данъка –

**EXTEND LECTURER ADD (SALARY \* 0.1) AS TAX**

Резултатът е релация със:

- заглавна част - заглавната част на оригиналната релация, разширена с новия атрибут;
- тяло - всички записи от оригиналната релация, разширени с изчислената стойност на новия атрибут.

## EXTEND - пример

**LECTURER**

ID	FName	LName	Title	Salary
101	Стоян	Колев	Проф.	550.00
102	Петър	Иванов	Доц.	432.76
103	Венета	Георгиева	Ст.н.с.	389.23

**EXTEND LECTURER ADD (SALARY \* 0.1) AS TAX**

ID	FName	LName	Title	Salary	Tax
101	Стоян	Колев	Проф.	550.00	55.00
102	Петър	Иванов	Доц.	432.76	43.28
103	Венета	Георгиева	Ст.н.с.	389.23	38.92

## SUMMARIZE

Този оператор извършва вертикално изчисление.

Резултатът е релация със:

- заглавна част -  $\{A_1, \dots, A_n, Z\}$ ;
- тяло - всички записи  $t$  така, че  $t$  е един запис на проекция на  $A$  върху  $A_1, \dots, A_n$ , разширен с една стойност за новия атрибут  $Z$ ;
- новата  $Z$ -стойност е получена чрез изчисляване на израз за всички записи от  $A$ , които имат едни и същи стойности за  $A_1, \dots, A_n$ .

Степен на резултатната релация - степен на проекцията на оригиналната релация + 1.

Кардиналност на резултатната релация - кардиналност на проекцията на оригиналната релация.

## SUMMARIZE

LECTURER

L_ID	SALARY	F_ID	F_NAME
L1	340		
L1	380		
L2	320		
L2	290		

SUMMARIZE LECTURER  
BY (L\_ID)  
ADD SUM(SALARY) AS S\_TOTAL

L_ID	S_TOTAL
L1	720
L2	610

LECTURER

ID	Name	Title	Faculty	Salary
101	Стоян Копев	Проф.	ФМИ	550.00
102	Петър Иванов	Проф.	ФМИ	522.76
103	Венета Георгиева	Проф.	Икономика	730.23
104	Мария Михайлова	Проф.	ФМИ	609.70
105	Стоил Караджов	Проф.	ФМИ	590.00
106	Милен Петров	Проф.	Икономика	670.00
107	Васил Ганчев	Доц.	ФМИ	415.20
108	Надя Тодорова	Доц.	ФМИ	453.00

SUMMARIZE LECTURER  
BY (TITLE, FACULTY)  
ADD AVG(SALARY) AS AVGSAL

Title	Faculty	AvgSal
Проф.	ФМИ	568.12
Проф.	Икономика	700.16
Доц.	ФМИ	434.10

• AvgSal е атрибутът Z;

• Атрибутите с еднакви стойности  $A_1, \dots, A_n$  са Title, Faculty, т.е.  $n = 2$ .

## Релационни сравнения

***expression*  $\Theta$  *expression***

където *expression* са изрази, които се изчисляват до типове съвместими релации, а  $\Theta$  е някой от следните оператори за сравнение:

- = равно
- $\neq$  различно
- $\leq$  подмножество на
- $<$  истинско подмножество на
- $\geq$  супермножество на
- $>$  истинско супермножество


Примери:

1.  $S[CITY] = P[CITY]$ , означаващ проекцията на доставчиците върху атрибута CITY същата ли е като тази на частите върху CITY?
2.  $S[Supp] > SP[Supp]$ , означаващ има ли доставчици, които не доставят никакви части?

В практиката често се налага да се провери дали дадена релация е празна, т.е. не съдържа записи. Нека тогава дефинираме логическа функция:

**`IS_EMPTY(expression)`**

която връща `true`, ако резултатът от изчислението на израза е празен, и `false` в противен случай.



Друг чест случай е да се налага да проверим дали даден запис  $t$  се среща в дадена релация  $R$ .

$$\{t\} \leq R$$

Друго изразяване на това условие, познато от SQL, е

$$t \text{ IN } R$$

където IN е оператор за проверка на принадлежност към множество.

---

---

---

---

---

---

---