

Databases

-

Chapter II: SQL

References

Further study required

- "SQL - Structured Query Language", Luís Damas, 6th Edition, FCA, 2005
- "Beginning Oracle SQL", Lex de Haan et al., APress, 2009

Programming languages

Generations

- 1st gen. => Machine code (0001010101)
- 2nd gen. => Assembly
- 3rd gen. => Java, C, Pascal, **PL/SQL**
- 4th gen. => LISP, PROLOG, **SQL**

Accessing Databases

Universal programming language

- SQL, *Structured Query Language*

Available basic data operations

- Insert new rows
- Update old data
- Delete old data
- Select existing data

SQL - queries

SQL example

CLIENTS

id	name	city	phoneNr
1	Valdemar Freitas	Leiria	244000001
2	Manuel da Silva	Lisboa	210000001
...
99	Pedro Passos Coelho	Lisboa	961000001
100	Maria de Sousa	Porto	

SQL query

```
SELECT name, phoneNr
FROM clients
WHERE city = 'Lisboa'
ORDER BY name;
```

=>

Information to user

NAME	PHONENR
-----	-----
Manuel da Silva	210000001
Pedro Passos Coelho	961000001

SQL statements

Data insertion

- Inserting new rows

- `INSERT INTO clients (id,name,city)`
`VALUES (101,'Rui Oliveira','Coimbra');`

CLIENTS

id	name	city	phoneNr
1	Valdemar Freitas	Leiria	244000001
2	Manuel da Silva	Lisboa	210000001
...
99	Pedro Passos Coelho	Lisboa	961000001
100	Maria de Sousa	Porto	
101	Rui Oliveira	Coimbra	

SQL statements

Other SQL basic statements

- Updating old data

- `UPDATE` clients

- `SET` nome='Manuela da Silva'

- `WHERE` id=2;

CLIENTS

id	name	city	phoneNr
1	Valdemar Freitas	Leiria	244000001
2	Manuela da Silva	Lisboa	210000001
...
99	Pedro Passos Coelho	Lisboa	961000001
100	Maria de Sousa	Porto	
101	Rui Oliveira	Coimbra	

SQL statements

Other SQL basic statements

- Deleting old rows

- **DELETE**

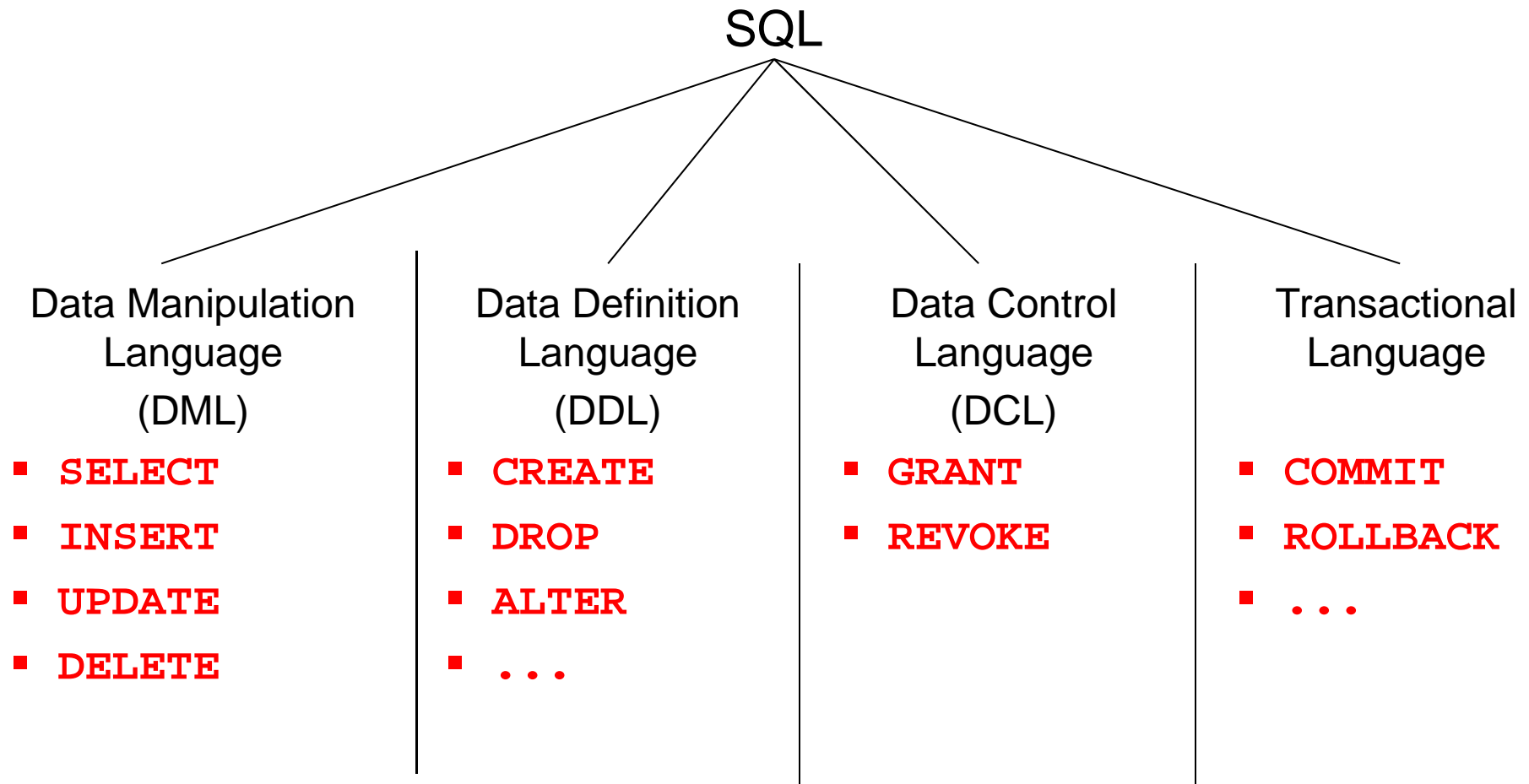
- FROM** clients

- WHERE** city='Leiria' ;

CLIENTS

id	name	city	phoneNr
1	Valdemar Freitas	Leiria	244000001
2	Manuela da Silva	Lisboa	210000001
...
99	Pedro Passos Coelho	Lisboa	961000001
100	Maria de Sousa	Porto	
101	Rui Oliveira	Coimbra	

SQL statements



SQL DML: SELECT statement

Basically... :(

```
SELECT [DISTINCT | ALL]  
        { * | <column name or expresion> [AS <new name>] [, ...] }  
FROM <table or view name> [new name] [, | JOIN [, ...] ]  
[WHERE <row selection conditions>]  
[GROUP BY <grouping criteria>  
[HAVING <group selection conditions>]  
[ORDER BY <column name or expression> [ASC | DESC] [, ...] ] ;
```

Reference: Oracle® Database SQL Language Reference 11g Release 1 (11.1) , pgs 19-4..19-52

SQL DML: SELECT statement

Meaning... :)

CLIENTS

id	name	city	phone_nr	birth_date	total_spent
1	António Freitas	Leiria	244244098	1980-04-06	1200
2	Rita Marujo	Lisboa	217769576	1983-01-06	1500
3	Carlos da Silva	Coimbra		1972-01-31	100
4	Ana Oliveira	LEIRIA	244987601	1978-11-09	5400

```
SELECT id,  
       name AS "Nome",  
       city  
FROM clients  
WHERE phone_nr IS NOT NULL  
ORDER BY name;
```

=

ID	Nome	CITY
---	-----	-----
4	Ana Oliveira	LEIRIA
1	António Freitas	Leiria
2	Rita Marujo	Lisboa
3 rows selected		

SQL DML: SELECT statement

Concepts

CLIENTS

id	name	city	phone_nr	birth_date	total_spent
1	António Freitas	Leiria	244244098	1980-04-06	1200
2	Rita Marujo	Lisboa	217769576	1983-01-06	1500
3	Carlos da Silva	Coimbra		1972-01-31	100
4	Ana Oliveira	LEIRIA	244987601	1978-11-09	5400

PROJECTION

```
SELECT id,  
       name AS "Nome",  
       city  
FROM clients  
WHERE phone_nr IS NOT NULL  
ORDER BY name;
```

=

ID	Nome	CITY
---	-----	-----
4	Ana Oliveira	LEIRIA
1	António Freitas	Leiria
2	Rita Marujo	Lisboa

3 rows selected

SQL DML: SELECT statement

Concepts

CLIENTS

id	name	city	phone_nr	birth_date	total_spent
1	António Freitas	Leiria	244244098	1980-04-06	1200
2	Rita Marujo	Lisboa	217769576	1983-01-06	1500
3	Carlos da Silva	Coimbra		1972-01-31	100
4	Ana Oliveira	LEIRIA	244987601	1978-11-09	5400

SELECTION

```
SELECT id,  
       name AS "Nome",  
       city  
FROM clients  
WHERE phone_nr IS NOT NULL  
ORDER BY name;
```

=

ID	Nome	CITY
---	-----	-----
4	Ana Oliveira	LEIRIA
1	António Freitas	Leiria
2	Rita Marujo	Lisboa

3 rows selected

SELECT statement: SELECT clause

Useful operators

- Mathematical: + , - , * , /
- Others: | |

Useful predicates

- DISTINCT

Exercises

- What is the amount spent by client with id 2? (in U.S. dollars)
Rita Marujo spent 1950 US dollars
- Which addresses have clients?

SELECT statement: WHERE clause

Purpose

- Perform a *selection*

Useful operators

- Mathematical: $>$, $<$, $>=$, $<=$, $<>$, \neq
- Logical: AND, OR, NOT
- Others: BETWEEN, IN, LIKE, IS NULL

SELECT statement: ORDER BY clause

Predicates

- ASC
- DESC

Single expression vs Multiple expression sorting

- ORDER BY <expression1> [ASC|DESC] [, ...]

Example

```
SELECT id,  
       name,  
       address  
FROM clients  
ORDER BY address ASC, name DESC;
```


SELECT statement: *line functions*

Overall advantages of functions

- They simplify programming
- They allow output formatting

Properties of line functions in SQL

- Each function is executed once per retrieved row
- Each function returnn one value per row

Example

```
SELECT id, UPPER(name),  
       LOWER(city) AS morada  
FROM clients  
ORDER BY id;
```

=

ID	NAME	MORADA
---	-----	-----
1	ANTÓNIO FREITAS	leiria
2	RITA MARUJO	lisboa
3	CARLOS DA SILVA	coimbra
4	ANA OLIVEIRA	leiria
4 rows selected		

SELECT statement: line functions

Examples of line functions

- TO_CHAR, TO_DATE, UPPER, NVL, DECODE, MONTHS_BETWEEN

Example

- *What is the address of clients born in 1975?*

```
SELECT id,  
       DECODE (address, 'Leiria', 'Cá da terra', 'De fora') AS "De onde"  
FROM clients  
WHERE TO_CHAR(birth_date, 'yyyy') = '1975'  
ORDER BY name;
```

Exercises

CLIENTS

id	name	city	phone_nr	birth_date	spent	nr_children	gender
1	António Freitas	Leiria	244244098	1980-04-06	1200	1	M
2	Rita Marujo	Lisboa	217769576	1983-01-06	1500	0	F
3	Carlos da Silva	Coimbra		1972-01-31	100	3	M
4	Ana Oliveira	LEIRIA	244987601	1978-11-09	5400	2	F
5	João Silva	Coimbra	239876098	1978-12-04	650	0	M

Exercises

- Phone number of clients who spent more than 1200
- Clients who spent more than 1200 and who have kids
- Clients who spent 500 or more and 2000 or less
- Each client's age
- Clients having more than 40 years of age
- Full list of clients with 2 or 3 kids

Databases - Chapter II: SQL

Grouping data

Grouping Data

Advantages

- Aggregate rows using specific criteria to generate summary information

Examples

- How many clients live per city?
- How much has been spent per gender?
- How much has been spent per city?
- How much has been spent per city per capita?

Grouping Data: Example ⁽¹⁾

“How many clients live per city?”

CLIENTS

id	name	city	phone_nr	birth_date	spent	nr_children	gender
1	António Freitas	LEIRIA	244244098	1980-04-06	1200	1	M
2	Rita Marujo	Lisboa	217769576	1983-01-06	1500	0	F
3	Carlos da Silva	Coimbra		1972-01-31	100	3	M
4	Ana Oliveira	LEIRIA	244987601	1978-11-09	5400	2	F
5	João Silva	Coimbra	239876098	1978-12-04	650	0	M

- How to solve this query mentally?
 - First, group rows based on the city name
 - Then, count how many rows exist in each group

Grouping Data: Example ⁽²⁾

So, first create groups

CLIENTS

id	name	city	phone_nr	birth_date	spent	nr_children	gender
1	António Freitas	LEIRIA	244244098	1980-04-06	1200	1	M
2	Rita Marujo	Lisboa	217769576	1983-01-06	1500	0	F
3	Carlos da Silva	Coimbra		1972-01-31	100	3	M
4	Ana Oliveira	LEIRIA	244987601	1978-11-09	5400	2	F
5	João Silva	Coimbra	239876098	1978-12-04	650	0	M

=

GROUP 1

1	António Freitas	LEIRIA	244244098	1980-04-06	1200	1	M
4	Ana Oliveira	LEIRIA	244987601	1978-11-09	5400	2	F

+

GROUP 2

3	Carlos da Silva	Coimbra		1972-01-31	100	3	M
5	João Silva	Coimbra	239876098	1978-12-04	650	0	M

+

GROUP 3

2	Rita Marujo	Lisboa	217769576	1983-01-06	1500	0	F
---	-------------	--------	-----------	------------	------	---	---

Grouping Data: Example ⁽³⁾

Then, count...

1	António Freitas	LEIRIA	244244098	1980-04-06	1200	1	M
4	Ana Oliveira	LEIRIA	244987601	1978-11-09	5400	2	F

=> 2 rows

3	Carlos da Silva	Coimbra		1972-01-31	100	3	M
5	João Silva	Coimbra	239876098	1978-12-04	650	0	M

=> 2 rows

2	Rita Marujo	Lisboa	217769576	1983-01-06	1500	0	F
---	-------------	--------	-----------	------------	------	---	---

=> 1 row

Grouping Data: Example ⁽³⁾

And in SQL it goes like...

```
SELECT city, COUNT(*)  
FROM clients  
GROUP BY city;
```

=

CITY	COUNT (*)
-----	-----
LEIRIA	2
Coimbra	2
Lisboa	1

Grouping Data

“How much has been spent per gender?”

CLIENTS

id	name	city	phone_nr	birth_date	spent	nr_children	gender
1	António Freitas	Leiria	244244098	1980-04-06	1200	1	M
2	Rita Marujo	Lisboa	217769576	1983-01-06	1500	0	F
3	Carlos da Silva	Coimbra		1972-01-31	100	3	M
4	Ana Oliveira	LEIRIA	244987601	1978-11-09	5400	2	F
5	João Silva	Coimbra	239876098	1978-12-04	650	0	M

=

GROUP 1

2	Rita Marujo	Lisboa	217769576	1983-01-06	1500	0	F
4	Ana Oliveira	LEIRIA	244987601	1978-11-09	5400	2	F

+

GROUP 2

1	António Freitas	Leiria	244244098	1980-04-06	1200	1	M
3	Carlos da Silva	Coimbra		1972-01-31	100	3	M
5	João Silva	Coimbra	239876098	1978-12-04	650	0	M

GROUP BY clause

Syntax

- SELECT ...
FROM ...
GROUP BY <expression1> [, ...]

Group functions

Advantages

- Generate information from data stored in grouped rows

Rules

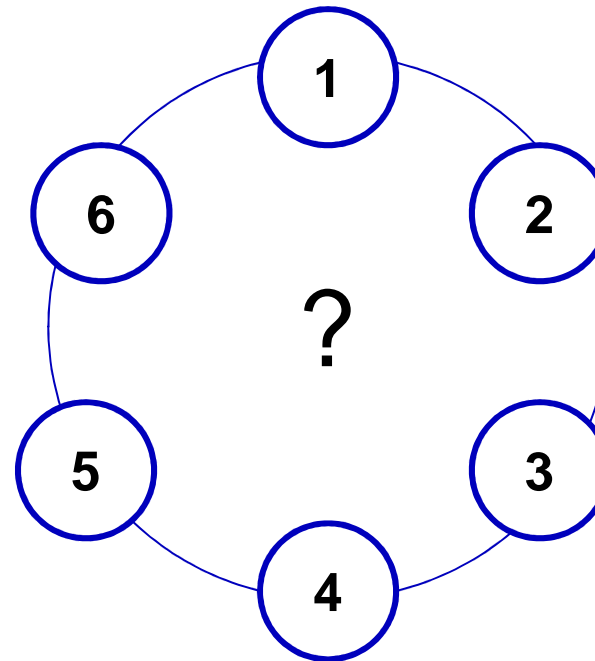
- Applied *per group* basis
- Each function returns one value per group

Examples

- COUNT
- SUM
- MIN, MAX, AVG

SELECT statement: *clause order*

5 SELECT ...
1 FROM ...
2 WHERE ...
3 GROUP BY ...
4 HAVING ...
6 ORDER BY ...



Filtering Grouped Data

Advantages

- Choose which of the groups can be used

Examples

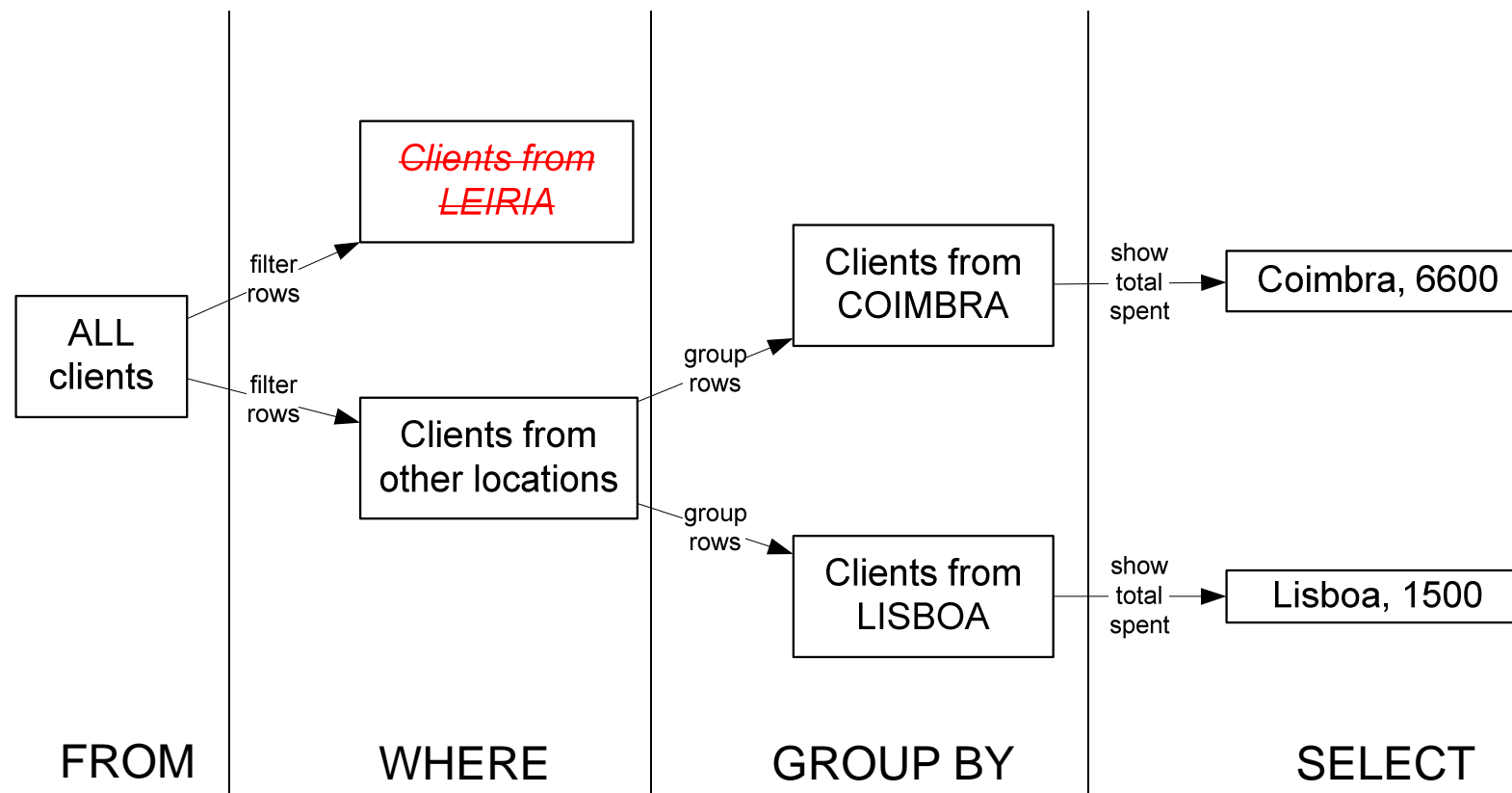
- How many clients live per city,
but only in cities with 2 or more clients ?
- How much has been spent per city,
except for the city of Leiria ?
- How much has been spent per city,
but only in cities with more than 500 spent per capita ?

Attention

- WHERE \neq HAVING

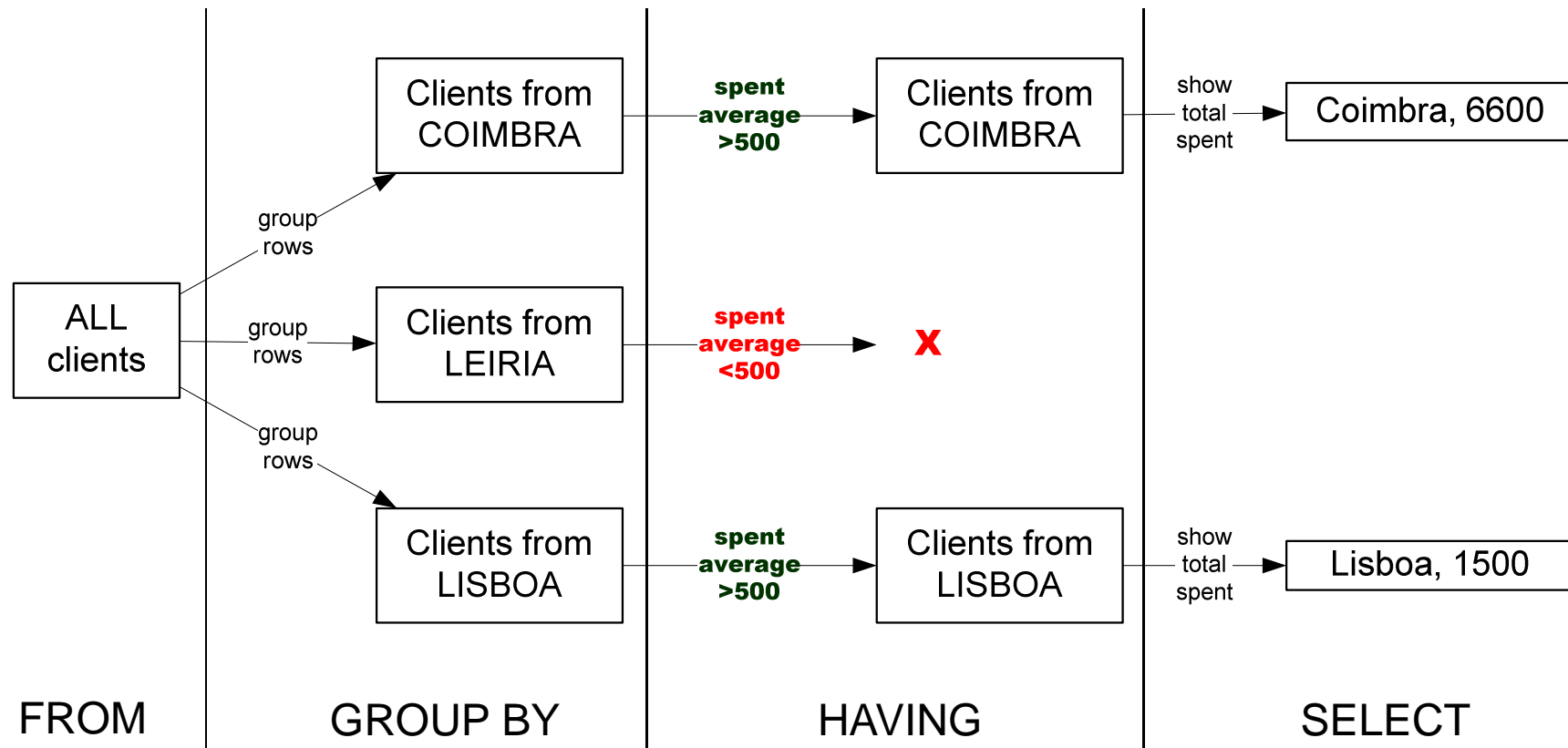
Filtering Grouped Data... visually

How much has been spent per city,
except for the city of Leiria ?



Filtering Grouped Data... visually

How much has been spent per city,
but only in cities with more than 500 spent per capita ?



Exercises

CLIENTS

id	name	city	phone_nr	birth_date	spent	nr_children	gender
1	António Freitas	Leiria	244244098	1980-04-06	1200	1	M
2	Rita Marujo	Lisboa	217769576	1983-01-06	1500	0	F
3	Carlos da Silva	Coimbra		1972-01-31	100	3	M
4	Ana Oliveira	LEIRIA	244987601	1978-11-09	5400	2	F
5	João Silva	Coimbra	239876098	1978-12-04	650	0	M

Exercises

- What is the highest spent value?
- What is the highest spent value per city?
- Who spends more, women or men?
- Who spends more, clients *with* children or *without* children?
- What is the average age of clients?
- How many clients do not possess a phone number?

Retrieving data from multiple tables

Retrieving data from multiple tables

New scenario

CLIENTS

id	name	city_id	phone_nr	birth_date	spent	nr_children	gender
1	António Freitas	1	244244098	1980-04-06	1200	1	M
2	Rita Marujo	2	217769576	1983-01-06	1500	0	F
3	Carlos da Silva	3		1972-01-31	100	3	M
4	Ana Oliveira	1	244987601	1978-11-09	5400	2	F
5	João Silva	3	239876098	1978-12-04	650	0	M

CITIES

id	name
1	Leiria
2	Lisboa
3	Coimbra
4	Guarda

Retrieving data from multiple tables

Example: *Where does João Silva lives?*

CLIENTS

id	name	city_id	phone_nr	birth_date	spent	nr_children	gender
1	António Freitas	1	244244098	1980-04-06	1200	1	M
2	Rita Marujo	2	217769576	1983-01-06	1500	0	F
3	Carlos da Silva	3		1972-01-31	100	3	M
4	Ana Oliveira	1	244987601	1978-11-09	5400	2	F
5	João Silva	3	239876098	1978-12-04	650	0	M

CITIES

id	name
1	Leiria
2	Lisboa
3	Coimbra
4	Guarda

Retrieving data from multiple tables

Example: *Where does João Silva lives?*

How it happens?

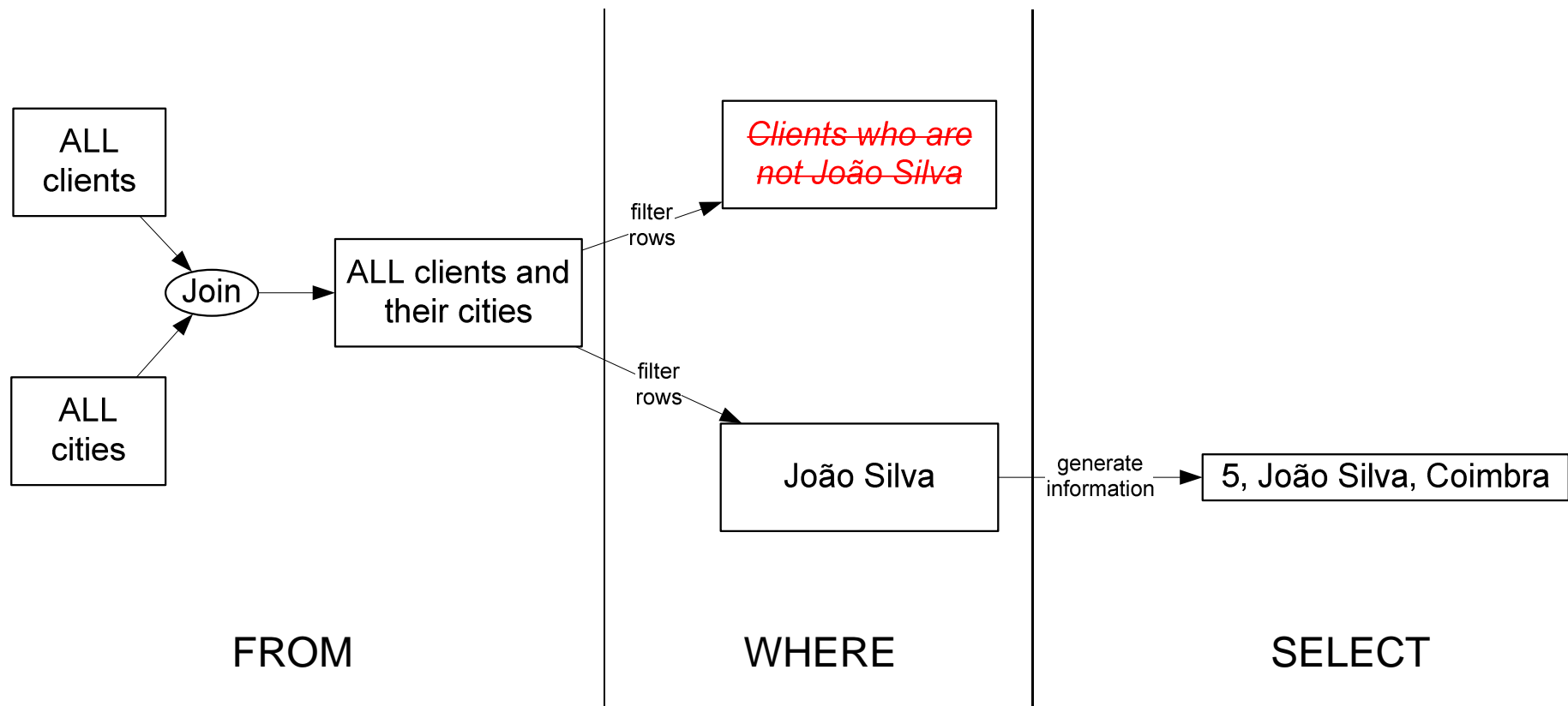
CLIENTS

id	name	...	gender	city_id	Join	cities.id	cities.name
1	António Freitas	...	M	1	→	1	Leiria
2	Rita Marujo	...	F	2	→	2	Lisboa
3	Carlos da Silva	...	M	3	→	3	Coimbra
4	Ana Oliveira	...	F	1	→	1	Leiria
5	João Silva	...	M	3	→	3	Coimbra

1. Join
2. *Filter*

Joining Data... visually

Where does 'João Silva' lives?



Retrieving data from multiple tables

Example: *Where does João Silva lives?*

```
SELECT clients.id, clients.name, cities.name  
FROM clients JOIN cities ON clients.city_id = cities.id  
WHERE name = 'João Silva';
```

OR

```
SELECT clients.id, clients.name, cities.name  
FROM clients, cities  
WHERE clients.city_id = cities.id  
AND name = 'João Silva';
```

clients.id	clients.name	cities.name
5	João Silva	Coimbra

Retrieving data from multiple tables

Advantages

- Generate information from data stored in more than one table

Rules

- Available only in FROM and WHERE clauses
- A join condition should always be used, or else...

Retrieving data from multiple tables: *horizontal join*

Definition

- Find the line(s) of table *t2* that are *connected* to a specific line in table *t1* as long as a *connection condition* exists

Types of horizontal joins

- Equijoin (=)
- Non-equijoin (BETWEEN, >, <)
- Self join
- Outer join
 - LEFT, RIGHT, FULL

Exercises

CLIENTS

id	name	city_id	phone_nr	birth_date	spent	nr_children	gender
1	António Freitas	1	244244098	1980-04-06	1200	1	M
2	Rita Marujo	2	217769576	1983-01-06	1500	0	F
3	Carlos da Silva	3		1972-01-31	100	3	M
4	Ana Oliveira	1	244987601	1978-11-09	5400	2	F
5	João Silva	3	239876098	1978-12-04	650	0	M

SALES

id	date	client_id
4000	2014-05-09	3
4001	2014-05-09	3
4002	2014-05-09	2
4003	2014-05-09	1

CITIES

id	name
1	Leiria
2	Lisboa
3	Coimbra
4	Guarda

Exercises

- Which sales belong to clients living in *Lisboa*?
- Total sales made last year in each city (show city name)
- How many clients exist per city (show city name)
- Which cities have no clients?
- Are there any clients without related sales?

Retrieving data from multiple tables: *vertical join*

Definition

- Merge the lines of queries $q1$ and $q2$ using a set operation

Set operations

- UNION
- UNION ALL
- INTERSECT
- MINUS

Retrieving data from multiple tables: *vertical join*

Example

CLIENTS_OLD

id	name	phone_nr	birth_date	spent	gender	years_as_client
12	António Costa	210009999	1971-08-06	130000	M	11
61	Celine Dion	239000111	1970-01-16	600000	F	12

- Show name and birth date of current and old clients

```
SELECT name, birth_date  
FROM clients
```

UNION

```
SELECT name, birth_date  
FROM clients_old
```

```
ORDER BY 1;
```

=

NAME	BIRTH_DATE
-----	-----
ANA OLIVEIRA	1978-11-09
ANTÓNIO COSTA	1971-08-06
ANTÓNIO FREITAS	1980-04-06
CARLOS DA SILVA	1972-01-31
CELINE DION	1970-01-16
JOÃO SILVA	1978-12-04
RITA MARUJO	1983-01-06

Retrieving data from multiple tables: *vertical join*

Exercises

- Which of the current clients have been clients before?
- Which of the old clients are not returning clients?

Databases - Chapter II: SQL

Subqueries

Subqueries

Definition

- SELECT statement *inside* another SELECT statement (*query inside query*)

Advantages

- Compute the result of an *inner* query and use that result to compute the result of an outer query

Example

② outer

```
SELECT id, name
FROM clients
WHERE city_id NOT IN (SELECT id
                      FROM cities
                      WHERE name IN ('Leiria', 'Lisboa')) ;
```

① inner

In-line Subqueries

Advantages

- Compute the result of an *inner* query and use that result to compute the result of an outer query

Example

② outer

```
SELECT id, name
FROM clients
WHERE city_id NOT IN (SELECT id
                      FROM cities
                      WHERE name IN ('Leiria', 'Lisboa')) ;
```

① inner

Subqueries

Special operators

- IN
- NOT IN
- ALL (with >, <)
- ANY (with >, <)

Exercise

- List the name and id of all clients who bought something in the day with the highest ammount of sales in the previous year

In-line Subqueries

Example

- List the clients who spent more than the overall average of spent money.

```
SELECT *  
FROM clients  
WHERE spent > (SELECT AVG(spent)  
                FROM clients);
```

In-line Subqueries - Exercises

CLIENTS

id	name	city_id	phone_nr	birth_date	spent	nr_children	gender
1	António Freitas	1	244244098	1980-04-06	1200	1	M
2	Rita Marujo	2	217769576	1983-01-06	1500	0	F
3	Carlos da Silva	3		1972-01-31	100	3	M
4	Ana Oliveira	1	244987601	1978-11-09	5400	2	F
5	João Silva	3	239876098	1978-12-04	650	0	M

SALES

id	date	client_id
4000	2014-05-09	3
4001	2014-05-09	3
4002	2014-05-09	2
4003	2014-05-09	1

CITIES

id	name
1	Leiria
2	Lisboa
3	Coimbra
4	Guarda

Exercises

- Show the clients who spent the most
- Show the name of the cities with no clients (3 ≠ ways)
- Show clients who never bought products
- Show the city with the highest amount of spent money

Correlated Subqueries

Definition

- Subquery in which the outer query is executed once for each execution of the inner subquery (\neq linear subqueries)

Advantages

- Compute the result of the *inner* query **once for each** row given by the *outer* query

Correlated Subqueries

Show the clients who spent the most in their city

```
SELECT id, name
FROM clients cl1
WHERE spent = (SELECT MAX(spent)
               FROM clients cl2
               WHERE cl2.city_id = cl1.city_id) ;
```

① outer

② inner

CLIENTS

id	name	city_id	phone_nr	birth_date	spent	nr_children	gender
1	António Freitas	1	244244098	1980-04-06	1200	1	M
2	Rita Marujo	2	217769576	1983-01-06	1500	0	F
3	Carlos da Silva	3		1972-01-31	100	3	M
4	Ana Oliveira	1	244987601	1978-11-09	5400	2	F
5	João Silva	3	239876098	1978-12-04	650	0	M

Correlated Subqueries

Exercise: *<guess what this query returns>*

```
SELECT id, name, phone_nr
FROM clients c11
WHERE EXISTS (SELECT id
              FROM clients c12
              WHERE c12.phone_nr = c11.phone_nr
                 AND c12.id != c11.id);
```

Correlated Subqueries

Avoiding correlation of queries

- Substitute a correlated subquery by linear subqueries
- Correlated subqueries *can* harm performance

Exercises

- Remove correlation from the queries of the previous slides