

Databases

Lecture 3

Querying Relational Databases Using SQL

- **basic SELECT query:**

```
SELECT [DISTINCT] select-list  
FROM from-list  
WHERE qualification
```

- **select-list**

- list of (expressions involving) attributes from relations in the **from-list**

- **from-list**

- list of relation names; each of them can be followed by a range variable

- **qualification**

- selection conditions on the data from the relations in the **from-list**
- conditions (*expr op expr*, where $op \in \{<, \leq, =, >, \geq, \neq\}$ and *expr* is an expression that can include attributes, constants, etc) combined with the logical operators AND, OR, NOT

- basic SELECT query:

```
SELECT [DISTINCT] select-list  
FROM from-list  
WHERE qualification
```

- the SELECT, FROM clauses - mandatory
- the WHERE clause - optional

- the conceptual evaluation strategy:

```
SELECT [DISTINCT] select-list  
FROM from-list  
WHERE qualification
```

- compute the cross product of tables in the **from-list**
- remove the rows that don't meet **qualification**
- eliminate unwanted columns, i.e., those that don't appear in the **select-list**
- if DISTINCT is specified, remove duplicates
 - by default, duplicates are not eliminated

- examples on the schema

Researchers(*RID*: integer, *Name*: string, *ImpactFactor*: integer, *Age**: integer)

Papers(*PID*: integer, *Title*: string, *Conference*: string)

AuthorContribution(*RID*: integer, *PID*: integer, *Year*: integer)

* we use the *Age* attribute for simplicity; it is preferable to store the date of birth, as it doesn't change every year

- Find the names of researchers who have worked on the paper with PID = 307.

```
SELECT R.Name
FROM Researchers R, AuthorContribution A
WHERE R.RID = A.RID AND A.PID = 307
```

Researchers

RID	Name	ImpactFactor	Age
1	Popescu	10	30
2	Ionescu	10	40
4	Andreescu	5	24

AuthorContribution

RID	PID	Year
1	307	2011
1	200	2012
2	307	2011

- compute the cross product of tables *Researchers* and *AuthorContribution*

RID	Name	ImpactFactor	Age	RID	PID	Year
1	Popescu	10	30	1	307	2011
1	Popescu	10	30	1	200	2012
1	Popescu	10	30	2	307	2011
2	Ionescu	10	40	1	307	2011
2	Ionescu	10	40	1	200	2012
2	Ionescu	10	40	2	307	2011
4	Andreescu	5	24	1	307	2011
4	Andreescu	5	24	1	200	2012
4	Andreescu	5	24	2	307	2011

- *RID* appears in both *Researchers* and *AuthorContribution* => it must be qualified (e.g., in the WHERE clause)

- remove the rows in the cross product that don't satisfy the condition
 $R.RID = A.RID$ AND $A.PID = 307$

RID	Name	ImpactFactor	Age	RID	PID	Year
1	Popescu	10	30	1	307	2011
1	Popescu	10	30	1	200	2012
1	Popescu	10	30	2	307	2011
2	Ionescu	10	40	1	307	2011
2	Ionescu	10	40	1	200	2012
2	Ionescu	10	40	2	307	2011
4	Andreescu	5	24	1	307	2011
4	Andreescu	5	24	1	200	2012
4	Andreescu	5	24	2	307	2011

- remove the rows in the cross product that don't satisfy the condition
 $R.RID = A.RID \text{ AND } A.PID = 307$

RID	Name	ImpactFactor	Age	RID	PID	Year
1	Popescu	10	30	1	307	2011
1	Popescu	10	30	1	200	2012
1	Popescu	10	30	2	307	2011
2	Ionescu	10	40	1	307	2011
2	Ionescu	10	40	1	200	2012
2	Ionescu	10	40	2	307	2011
4	Andreescu	5	24	1	307	2011
4	Andreescu	5	24	1	200	2012
4	Andreescu	5	24	2	307	2011

- remove the rows in the cross product that don't satisfy the condition
 $R.RID = A.RID \text{ AND } A.PID = 307$

RID	Name	ImpactFactor	Age	RID	PID	Year
1	Popescu	10	30	1	307	2011
1	Popescu	10	30	1	200	2012
1	Popescu	10	30	2	307	2011
2	Ionescu	10	40	1	307	2011
2	Ionescu	10	40	1	200	2012
2	Ionescu	10	40	2	307	2011
4	Andreescu	5	24	1	307	2011
4	Andreescu	5	24	1	200	2012
4	Andreescu	5	24	2	307	2011

- remove the rows in the cross product that don't satisfy the condition
 $R.RID = A.RID \text{ AND } A.PID = 307$

RID	Name	ImpactFactor	Age	RID	PID	Year
1	Popescu	10	30	1	307	2011
2	Ionescu	10	40	2	307	2011

- remove the columns that don't appear in R.Name

Name
Popescu
Ionescu

- basic queries

Find the names and ages of all researchers. Eliminate duplicates.

```
SELECT DISTINCT R.Name, R.Age  
FROM Researchers R
```

Find the researchers with an impact factor > 3 (all the data about researchers).

```
SELECT R.RID, R.Name, R.ImpactFactor, R.Age  
FROM Researchers AS R  
WHERE R.ImpactFactor > 3  
-- SELECT *
```

Find the names of researchers who have published in the EDBT conference.

```
SELECT R.Name  
FROM Researchers R, AuthorContribution A, Papers P  
WHERE R.RID = A.RID AND A.PID = P.PID AND P.Conference =  
'EDBT'
```

Find the ids of researchers who have published in the EDBT conference.

```
SELECT A.RID  
FROM AuthorContribution A, Papers P  
WHERE A.PID = P.PID AND P.Conference = 'EDBT'
```

Find the names of researchers who have published at least one paper.

```
SELECT R.Name  
FROM Researchers R, AuthorContribution A  
WHERE R.RID = A.RID
```

Find the conferences that published Pop's papers.

```
SELECT P.Conference  
FROM Researchers R, AuthorContribution A, Papers P  
WHERE R.RID = A.RID AND A.PID = P.PID AND R.Name = 'Pop'
```

*** lecture discussion - 2 researchers named Pop**

- expressions in SELECT

Compute an incremented impact factor for researchers who worked on two different papers in the same year.

```
SELECT R.Name, R.ImpactFactor + 1 AS NewIF
FROM Researchers R, AuthorContribution A1, AuthorContribution
A2
WHERE R.RID = A1.RID AND R.RID = A2.RID
      AND A1.PID <> A2.PID
      AND A1.Year = A2.Year
```


- union, intersection

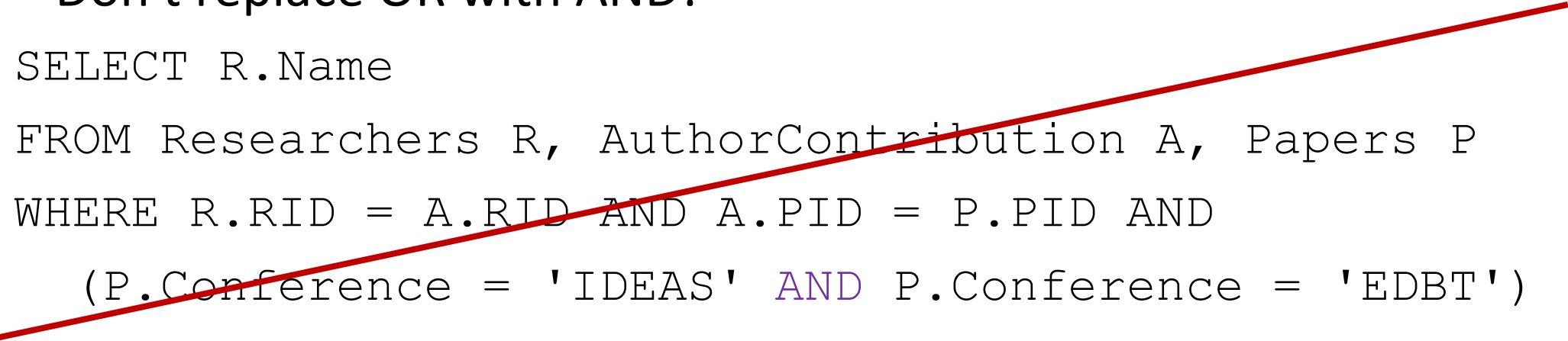
Find the names of researchers who have published in EDBT or IDEAS.

```
SELECT R.Name
FROM Researchers R, AuthorContribution A, Papers P
WHERE R.RID = A.RID AND A.PID = P.PID AND
      (P.Conference = 'IDEAS' OR P.Conference = 'EDBT')
```

Find the names of researchers who have published in EDBT and IDEAS.

*** Don't replace OR with AND!**

```
SELECT R.Name
FROM Researchers R, AuthorContribution A, Papers P
WHERE R.RID = A.RID AND A.PID = P.PID AND
      (P.Conference = 'IDEAS' AND P.Conference = 'EDBT')
```



Find the names of researchers who have published in EDBT and IDEAS.

```
SELECT R.Name
FROM Researchers R, AuthorContribution A1, Papers P1,
AuthorContribution A2, Papers P2
WHERE R.RID = A1.RID AND A1.PID = P1.PID AND
      P1.Conference = 'IDEAS' AND
      R.RID = A2.RID AND A2.PID = P2.PID AND
      P2.Conference = 'EDBT'
```

- nested queries
 - the WHERE clause
- IN

Find the names of researchers who have worked on the paper with PID = 307.

```
SELECT R.Name
FROM Researchers R
WHERE R.RID IN
    (SELECT A.RID
     FROM AuthorContribution A
     WHERE A.PID = 307)
```

- EXISTS

Find the names of researchers who have worked on the paper with PID = 307.

```
SELECT R.Name
FROM Researchers R
WHERE EXISTS (SELECT *
              FROM AuthorContribution A
              WHERE A.PID = 307 AND A.RID = R.RID)
```

Find the names of researchers who have published in EDBT.

```
SELECT R.Name
FROM Researchers R
WHERE R.RID IN
    (SELECT A.RID
     FROM AuthorContribution A
     WHERE A.PID IN
         (SELECT P.PID
          FROM Papers P
          WHERE P.Conference = 'EDBT'
         )
    )
```

Find the names of researchers who haven't published in EDBT.

```
SELECT R.Name
FROM Researchers R
WHERE R.RID NOT IN
    (SELECT A.RID
     FROM AuthorContribution A
     WHERE A.PID IN
         (SELECT P.PID
          FROM Papers P
          WHERE P.Conference = 'EDBT'
         )
    )
```

- operators ANY and ALL

Find researchers whose IF is greater than the IF of some researcher called *Ionescu*.

```
SELECT R.RID
FROM Researchers R
WHERE R.ImpactFactor > ANY
      (SELECT R2.ImpactFactor
       FROM Researchers R2
       WHERE R2.Name = 'Ionescu')
```

expression = ANY(subquery) \iff expression IN(subquery)

```
SELECT R.Name
FROM Researchers R
WHERE R.RID = ANY
      (SELECT A.RID
       FROM AuthorContribution A
       WHERE A.PID = 300)
```

```
SELECT R.Name
FROM Researchers R
WHERE R.RID IN
      (SELECT A.RID
       FROM AuthorContribution A
       WHERE A.PID = 300)
```


Find researchers whose IF is greater than the IF of every researcher called *Ionescu*.

```
SELECT R.RID
FROM Researchers R
WHERE R.ImpactFactor > ALL
      (SELECT R2.ImpactFactor
       FROM Researchers R2
       WHERE R2.Name = 'Ionescu')
```

expression <> ALL(subquery) \iff expression NOT IN(subquery)

```
SELECT R.Name
FROM Researchers R
WHERE R.RID <> ALL
      (SELECT A.RID
       FROM AuthorContribution A
       WHERE A.PID = 300)
```

```
SELECT R.Name
FROM Researchers R
WHERE R.RID NOT IN
      (SELECT A.RID
       FROM AuthorContribution A
       WHERE A.PID = 300)
```

- the JOIN operators
- JOIN examples are described on the following relational database:

Students

SID	Name	Group
135	Breje Mihai Paul	922
82	Radu Ana-Maria	926
294	Nourescu Oana	925

Courses

CID	Name
BD	Baze de date
SGBD	Sisteme de Gestiune a Bazelor de Date
DMBD	Data Mining in Big Data

Exams

StdId	CrslId	Grade	Credits
135	BD	10	6
82	SGBD	10	6
135	SGBD	10	6

Students

SID	Name	Group
135	Breje Mihai Paul	922
82	Radu Ana-Maria	926
294	Nourescu Oana	925

Exams

StdId	CrslId	Grade	Credits
135	BD	10	6
82	SGBD	10	6
135	SGBD	10	6

- find all the students' grades; include the students' names in the answer set

1. inner join: source1 [alias] **[INNER] JOIN** source2 [alias] **ON** condition

```
SELECT *
```

```
FROM Students S INNER JOIN Exams E ON S.SID = E.StdId
```

SID	Name	Group	StdId	CrslId	Grade	Credits
135	Breje Mihai Paul	922	135	BD	10	6
135	Breje Mihai Paul	922	135	SGBD	10	6
82	Radu Ana-Maria	926	82	SGBD	10	6

Students

SID	Name	Group
135	Breje Mihai Paul	922
82	Radu Ana-Maria	926
294	Nourescu Oana	925

Exams

StdId	CrslId	Grade	Credits
135	BD	10	6
82	SGBD	10	6
135	SGBD	10	6

- find all the students' grades; include students with no exams; the students' names must appear in the answer set

2. left outer join: source1 [alias] **LEFT [OUTER] JOIN** source2 [alias] **ON** condition

```
SELECT *
```

```
FROM Students S LEFT JOIN Exams E ON S.SID = E.StdId
```

SID	Name	Group	StdId	CrslId	Grade	Credits
135	Breje Mihai Paul	922	135	BD	10	6
135	Breje Mihai Paul	922	135	SGBD	10	6
82	Radu Ana-Maria	926	82	SGBD	10	6
294	Nourescu Oana	925	null	null	null	null

Courses

CID	Name
BD	Baze de date
SGBD	Sisteme de Gestiune a Bazelor de Date
DMBD	Data Mining in Big Data

Exams

StdId	CrsId	Grade	Credits
135	BD	10	6
82	SGBD	10	6
135	SGBD	10	6

find all the exams (including the names of the courses); include courses with no exams

3. right outer join: source1 [alias] **RIGHT [OUTER] JOIN** source2 [alias] **ON** condition

```
SELECT *
```

```
FROM Exams E RIGHT JOIN Courses C ON E.CrsId = C.CID
```

StdId	CrsId	Grade	Credits	CID	Name
135	BD	10	6	BD	Baze de date
135	SGBD	10	6	SGBD	Sisteme de Gestiune a Bazelor de Date
82	SGBD	10	6	SGBD	Sisteme de Gestiune a Bazelor de Date
null	null	null	null	DMBD	Data Mining in Big Data

Students

SID	Name	Group
135	Breje Mihai Paul	922
82	Radu Ana-Maria	926
294	Nourescu Oana	925

Exams

StdId	CrslId	Grade	Credits
135	BD	10	6
82	SGBD	10	6
135	SGBD	10	6
737	SGBD	9	6

find all the exams; include students with no exams and grades given by mistake to nonexistent students; the result should also contain students' names

4. full outer join: source1 [alias] **FULL [OUTER] JOIN** source2 [alias] **ON** condition

```
SELECT *
```

```
FROM Students S FULL JOIN Exams E ON S.SID = E.StdId
```

SID	Name	Group	StdId	CrslId	Grade	Credits
135	Breje Mihai Paul	922	135	BD	10	6
135	Breje Mihai Paul	922	135	SGBD	10	6
82	Radu Ana-Maria	926	82	SGBD	10	6
294	Nourescu Oana	925	null	null	null	null
null	null	null	737	SGBD	9	6

- other JOIN expressions

source1 [alias1] JOIN source2 [alias2] USING (column_list)

source1 [alias1] NATURAL JOIN source2 [alias2]

source1 [alias1] CROSS JOIN source2 [alias2]

- subquery in the FROM clause

```
SELECT R.*  
FROM Researchers R INNER JOIN  
    (SELECT *  
     FROM AuthorContribution A  
     WHERE A.PID = 400) t  
ON R.RID = t.RID
```

- copy data from one table to another

```
INSERT INTO T2
```

```
SELECT * FROM T1
```

- **GROUP BY, HAVING**

```
SELECT [DISTINCT] select-list  
FROM from-list  
WHERE qualification  
GROUP BY grouping-list  
HAVING group-qualification
```

- optional GROUP BY clause
 - list of (expressions involving) columns used for grouping
- optional HAVING clause
 - group qualification conditions

- **GROUP BY, HAVING**

```
SELECT [DISTINCT] select-list  
FROM from-list  
WHERE qualification  
GROUP BY grouping-list  
HAVING group-qualification
```

- **group**

- a collection of rows with identical values for the columns in **grouping-list**
- every row in the result of the query corresponds to a group

- GROUP BY, HAVING

```
SELECT [DISTINCT] select-list  
FROM from-list  
WHERE qualification  
GROUP BY grouping-list  
HAVING group-qualification
```

- **select-list**

- columns (that must appear in **grouping-list**)
- terms of the form *aggop(column) [AS NewName]*
 - e.g., MAX(R.ImpactFactor) AS MaxImpactFactor
 - *NewName* assigns a name to the column in the result table

- **GROUP BY, HAVING**

```
SELECT [DISTINCT] select-list  
FROM from-list  
WHERE qualification  
GROUP BY grouping-list  
HAVING group-qualification
```

- **group-qualification**

- expressions with a single value / group
- a column in **group-qualification** appears in **grouping-list** or as an argument to an aggregation operator
- records that meet **qualification** are partitioned into groups based on the values of the columns in **grouping-list**
- an answer row is generated for every group that meets **group-qualification**

Find the age of the youngest researcher for each impact factor.

```
SELECT R.ImpactFactor, MIN(R.Age)
FROM Researchers R
GROUP BY R.ImpactFactor
```

* discussion: using the GROUP BY clause vs writing n queries, one for each of the n values of the impact factor, where n depends on the relation instance

Find the age of the youngest researcher who is at least 18 years old for each impact factor with at least 10 such researchers.

```
SELECT R.ImpactFactor, MIN(R.Age) AS MinAge
FROM Researchers R
WHERE R.Age >= 18
GROUP BY R.ImpactFactor
HAVING COUNT(*) >= 10
```

See seminar 2:

- range variables
- the LIKE operator
- the UNION [ALL], INTERSECT, EXCEPT operators
- joins with more than 2 tables
- aggregation operators

References

- [Ta13] ȚÂMBULEA, L., Curs Baze de date, Facultatea de Matematică și Informatică, UBB, 2013-2014
- [Ra00] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems (2nd Edition), McGraw-Hill, 2000
- [Da03] DATE, C.J., An Introduction to Database Systems (8th Edition), Addison-Wesley, 2003
- [Ga08] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., Database Systems: The Complete Book, Prentice Hall Press, 2008
- [Ha96] HANSEN, G., HANSEN, J., Database Management And Design (2nd Edition), Prentice Hall, 1996
- [Ra07] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems, McGraw-Hill, 2007,
<http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html>
- [Si10] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts, McGraw-Hill, 2010, <http://codex.cs.yale.edu/avi/db-book/>
- [Ul11] ULLMAN, J., WIDOM, J., A First Course in Database Systems,
<http://infolab.stanford.edu/~ullman/fcdb.html>
- [Ta03] ȚÂMBULEA, L., Baze de date, Litografiat Cluj-Napoca 2003