Databases

Lecture 3

The Relational Model (II)

Querying Relational Databases Using SQL

Sabina S. CS

Managing Relational Databases with SQL

- ALTER TABLE – changes the structure of a defined table

  ALTER TABLE table_name operation

```
ALTER TABLE Students
ADD FavSymphony VARCHAR(50)
```

- possible operations (differences among DBMSs)
  - add / change / remove a column
    - ADD column_definition
    - {ALTER COLUMN | MODIFY} column_definition
    - DROP COLUMN column_name

Managing Relational Databases with SQL

- add / remove a constraint
  - ADD [CONSTRAINT constraint_name] PRIMARY KEY(column_list)
  - ADD [CONSTRAINT constraint_name] UNIQUE(column_list)
  - ADD [CONSTRAINT constraint_name] FOREIGN KEY (column_list) REFERENCES table_name[(column_list)] [ON UPDATE action] [ON DELETE action]
  - DROP [CONSTRAINT] constraint_name

Managing Relational Databases with SQL

- DROP TABLE – removes a table

  DROP TABLE table_name

```
DROP TABLE Students
```

- **Data Definition Language** (DDL) - subset of SQL used to create / remove / change components (e.g., tables)

Managing Relational Databases with SQL

- changing data in a table
- the INSERT command – adding records

    INSERT INTO table_name[(column_list)] VALUES (value_list)

    INSERT INTO table_name[(column_list)] subquery,

where *subquery* refers to a set of records (generated with the SELECT statement)

```
INSERT INTO Students (sid, cnp, lastname, firstname, age)
 VALUES (1, '123456789012', 'Popescu', 'Maria', 20)
```

Managing Relational Databases with SQL

- changing data in a table
- the UPDATE command – changing records

  UPDATE table_name

  SET column_name=expression [, column_name=expression] ...

  [WHERE condition]

- the command changes the records in the table that satisfy the condition in the WHERE clause; if the WHERE clause is omitted, all the records in the table are changed; the values of the columns specified in SET are changed to the associated expressions' values

```
UPDATE Students
SET age = age + 1
WHERE cnp = '123456789012'
```

Managing Relational Databases with SQL

- changing data in a table
- the DELETE command – removing records

  DELETE FROM table_name

  [WHERE condition]

- the command deletes the records in the table that satisfy the condition in the WHERE clause; if the WHERE clause is omitted, all the table's records are deleted

```
DELETE
FROM Students
WHERE lastname = 'Popescu'
```

- **Data Manipulation Language** (DML) - subset of SQL used to pose queries, to add / update / remove data

Managing Relational Databases with SQL

- **filter conditions**
  - expression comparison_operator expression
  - expression [NOT] BETWEEN valmin AND valmax
  - expression [NOT] LIKE pattern ("%" - any substring, "_" - one character)
  - expression IS [NOT] NULL
  - expression [NOT] IN (value [, value] ...)
  - expression [NOT] IN (subquery)
  - expression comparison_operator {ALL | ANY} (subquery)
  - [NOT] EXISTS (subquery)

Managing Relational Databases with SQL

- **filter conditions**
  - elementary condition (previously described)
  - (condition)
  - NOT condition
  - $condition_1$ AND $condition_2$
  - $condition_1$ OR $condition_2$

Managing Relational Databases with SQL

- 3-valued logic (truth values: *true*, *false*, *unknown*)

	TRUE	FALSE	NULL
NOT	FALSE	TRUE	NULL

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

Querying Relational Databases Using SQL

Sabina S. CS

- 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 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 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 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 Ionescu'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 = 'Ionescu'
```

* obs. There can be more than one researcher named Ionescu.

- 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
```

- 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)
```

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'
          )
     )
```

- 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)
```

- 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) $\Longleftrightarrow$ 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) $\Longleftrightarrow$ 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)
```

- union, intersection, set-difference

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'
```

Find the names of researchers who have published in EDBT but not in IDEAS.

```
SELECT R.Name
FROM Researchers R, AuthorContribution A, Papers P
WHERE R.RID = A.RID AND A.PID = P.PID AND P.Conference = 'EDBT'
    AND A.RID NOT IN (SELECT A2.RID
                      FROM AuthorContribution A2, Papers P2
                      WHERE A2.PID = P2.PID AND P2.Conference = 'IDEAS')
```

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

Students

SID	Name	Group
135	Alexandra	922
82	Paul	926
294	Ștefania	925

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

Students

SID	Name	Group
135	Alexandra	922
82	Paul	926
294	Ștefania	925

Exams

StdId	CrsId	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	CrsId	Grade	Credits
135	Alexandra	922	135	BD	10	6
135	Alexandra	922	135	SGBD	10	6
82	Paul	926	82	SGBD	10	6

Students

SID	Name	Group
135	Alexandra	922
82	Paul	926
294	Ștefania	925

Exams

StdId	CrsId	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	CrsId	Grade	Credits
135	Alexandra	922	135	BD	10	6
135	Alexandra	922	135	SGBD	10	6
82	Paul	926	82	SGBD	10	6
294	Ștefania	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	Alexandra	922
82	Paul	926
294	Ștefania	925

Exams

StdId	CrsId	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	CrsId	Grade	Credits
135	Alexandra	922	135	BD	10	6
135	Alexandra	922	135	SGBD	10	6
82	Paul	926	82	SGBD	10	6
294	Ștefania	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

- aggregation operators

```
COUNT, AVG, SUM, MIN, MAX
```

- 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