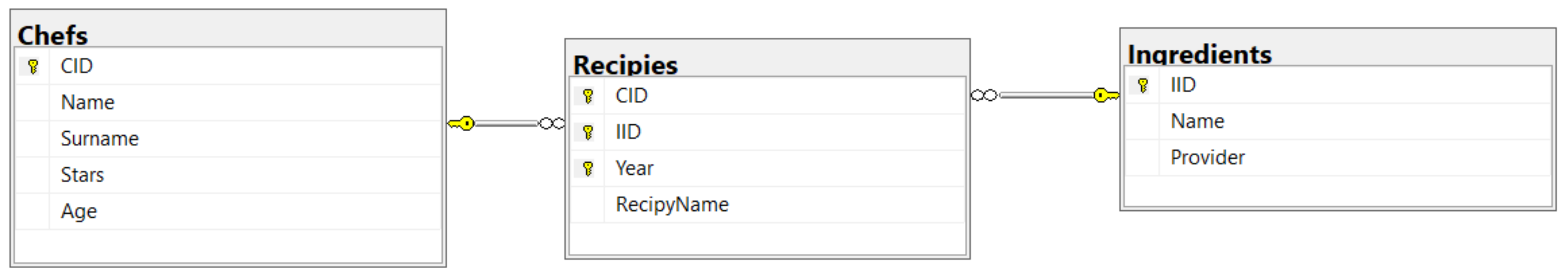
Lecture 4

SQL Queries (II)

SQL Queries

Consider the relational schema of the following database:



```
create table Chefs(
CID INT primary key identity,
Name varchar(50),
Surname varchar(50),
Stars int,
Age int)

create table Ingredients(
IID int primary key identity,
Name varchar(50),
Provider varchar(50))
```

```
create table Recipies(
CID int foreign key references Chefs(CID),
IID int foreign key references Ingredients(IID),
Year int,
RecipyName varchar(50),
constraint pk_Recepies primary key(CID, IID, Year)

Select * from Chefs
select * from Ingredients
select * from Recipies
```

SQL Queries

Basic Queries

Find the name and the age of all chefs. Eliminate duplicates.

```
SELECT DISTINCT C.Name, C. Age
FROM Chefs C
```

Find the chefs with the number of the stars>3 (all the data about chefs).

```
SELECT C.CID, C.Name, C.Surname, C.Stars, C.Age
FROM Chefs AS C
WHERE C.Stars>3
```

SQL Queries

Basic Queries

Find the name of the chefs who have used 'Moldovan' provider for their ingredients.

```
SELECT C.Name
FROM Chefs C, Recipies R, Ingredients I
WHERE C.CID=R.CID AND R.IID=I.IID AND I.Provider='Moldovan’
```

Find the ids of the chefs who have used 'Moldovan' provider for their ingredients.

```
SELECT R.CID
FROM Recipies R, Ingredients I
WHERE R.IID=I.IID AND I.Provider='Moldovan'
```

SQL Queries

Basic Queries

Find the name of the chefs who have used at least one ingredient.

```
SELECT C.Name
FROM Chefs C, Recipies R
WHERE C.CID=R.CID
```

Find the providers that used carrot's ingredient.

```
SELECT I.Provider
FROM Chefs C, Recipies R, Ingredients I
WHERE C.CID=R.CID AND R.IID=I.IID AND I.Name='carrot’
```

*obs. There can be more than one ingredient called carrot

SQL Queries

Expression in SELECT

Compute an incremented star for chefs who worked with two different ingredients in the same year.

```
SELECT C.Name, C.Stars+1 AS NewStars
FROM Chefs C, Recipies R1, Recipies R2
WHERE C.Cid=R1.Cid AND C.Cid=R2.Cid
AND R1.IId<>R2.IID
AND R1.Year=R2.Year
```

SQL Queries

Nested Queries

- **the WHERE clause**
- **IN**

Find the name of the chefs who have workes with the ingredient Iid=7.

```
SELECT C.Name
FROM Chefs C
WHERE C.Cid IN
            (SELECT C.Cid
            FROM Recipies R
            WHERE R.IID=7)
```

SQL Queries

Nested Queries

- **the WHERE clause**
- **IN**

Find the name of the chefs who **have** as a provider Moldovan.

```
SELECT C.Name
FROM Chefs C
WHERE C.Cid IN
            (SELECT R.Cid
            FROM Recipies R
            WHERE R.IId IN
                        (SELECT I.IId
                        FROM Ingredients I
                        WHERE I.Provider='Moldovan'))
```

SQL Queries

Nested Queries

- **the WHERE clause**
- **IN**

Find the name of the chefs who **haven't** as a provider Moldovan.

```
SELECT C.Name
FROM Chefs C
WHERE C.Cid IN
        (SELECT R.Cid
        FROM Recipies R
        WHERE R.IId NOT IN
                (SELECT I.IId
                FROM Ingredients I
                WHERE I.Provider='Moldovan'))
```

SQL Queries

Nested Queries

- **EXISTS**

Find the names of chefs who have worked with the ingredient with Iid=7.

```
SELECT C.Name
FROM Chefs C
WHERE EXISTS
            (SELECT *
            FROM Recipies R
            WHERE R.IId=7 AND C.Cid=R.Cid)
```

SQL Queries

Nested Queries

- **operators ANY and ALL**

Find the chefs whose stars is greater than the stars **of some** chefs called Paul.

```
SELECT C.Cid
FROM Chefs C
WHERE C.Stars>ANY
            (SELECT C2.Stars
            FROM Chefs C2
            WHERE C2.Name='Paul')
```

SQL Queries

Nested Queries

- **operators ANY and ALL**

Find the chefs whose stars is greater than the stars **of every (all)** chef called Paul.

```
SELECT C.Cid
FROM Chefs C
WHERE C.Stars>ALL
        (SELECT C2.Stars
        FROM Chefs C2
        WHERE C2.Name='Paul')
```

SQL Queries

Nested Queries

expression=ANY(subsquery) equivalent expression IN (subquery)

```
SELECT C.Name
FROM Chefs C
WHERE C.Cid=ANY
        (SELECT R.Cid
        FROM Recipies R
        WHERE R.Cid=3)
```

```
SELECT C.Name
FROM Chefs C
WHERE C.Cid IN
        (SELECT R.Cid
        FROM Recipies R
        WHERE R.Cid=3)
```

SQL Queries

Nested Queries

expression<>ALL(subquery) equivalent expression NOT IN(subquery)

```
SELECT C.Name
FROM Chefs C
WHERE C.Cid<>ALL
        (SELECT R.Cid
        FROM Recipies R
        WHERE R.Cid=3)
```

```
SELECT C.Name
FROM Chefs C
WHERE C.Cid NOT IN
        (SELECT R.Cid
        FROM Recipies R
        WHERE R.Cid=3)
```

SQL Queries

union, intersection, set-difference

Find the name of chefs who have used the provider Moldovan OR Petri.

```
SELECT C.Name
FROM Chefs C, Recipies R, Ingredients I
WHERE C.Cid=R.Cid AND R.IId=I.IID AND
(I.Provider='Moldovan' OR I.Provider='Petri')
```

```
SELECT C.Name
FROM Chefs C, Recipies R, Ingredients I
WHERE C.Cid=R.Cid AND R.IId=I.IID AND I.Provider='Moldovan'
UNION
SELECT C.Name
FROM Chefs C, Recipies R, Ingredients I
WHERE C.Cid=R.Cid AND R.IId=I.IID AND I.Provider='Petri'
```

SQL Queries

Find the name of the chefs who have used as a provider both Moldovan and Petri.

~~SELECT C.Name~~
~~FROM Chefs C, Recipies R, Ingredients I~~
~~WHERE C.Cid=R.Cid AND R.IId=I.IID AND~~
~~(I.Provider='Moldovan' AND I.Provider='Petri’)~~

SQL Queries

Find the name of the chefs who have used as a provider both Moldovan and Petri.

```
SELECT C.Name
FROM Chefs C, Recipies R1, Ingredients I1, Recipies R2, Ingredients I2
WHERE C.Cid=R1.Cid AND R1.IId=I1.IID AND I1.Provider='Moldovan' AND
C.Cid=R2.Cid AND R2.IId=I2.IId AND I2.Provider='Petri'

SELECT C.Name
FROM Chefs C, Recipies R, Ingredients I
WHERE C.Cid=R.Cid AND R.IId=I.IID AND I.Provider='Moldovan'
INTERSECT
SELECT C.Name
FROM Chefs C, Recipies R, Ingredients I
WHERE C.Cid=R.Cid AND R.IId=I.IID AND I.Provider='Petri'
```

SQL Queries

Find the name of the chefs who have used as a provider Moldovan but have not used Petri.

```
SELECT C.Name
FROM Chefs C, Recipies R, Ingredients I
WHERE C.Cid=R.Cid AND R.IId=I.IID AND I.Provider='Moldovan' AND
R.IID  NOT IN (SELECT R2.IId
FROM Recipies R2, Ingredients I2
WHERE R2.IId=I2.IId AND I2.Provider='Petri')
```

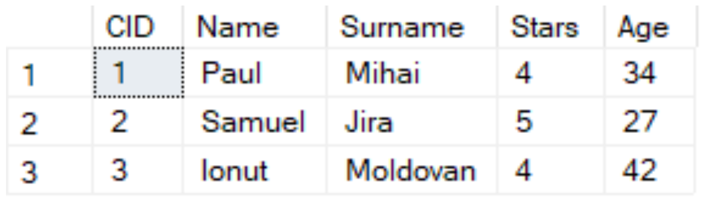
```
SELECT C.Name
FROM Chefs C, Recipies R, Ingredients I
WHERE C.Cid=R.Cid AND R.IId=I.IID AND I.Provider='Moldovan'
EXCEPT
SELECT C.Name
FROM Chefs C, Recipies R, Ingredients I
WHERE C.Cid=R.Cid AND R.IId=I.IID AND I.Provider='Petri'
```

SQL Queries

The JOIN operator

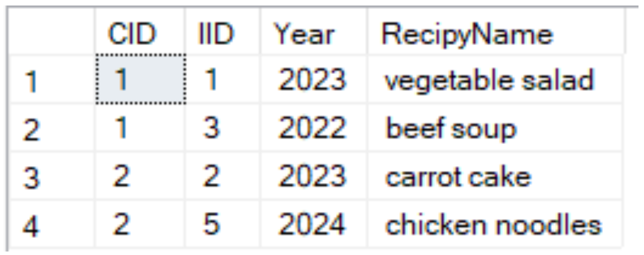
JOIN examples are presented on the following relational database

Chefs



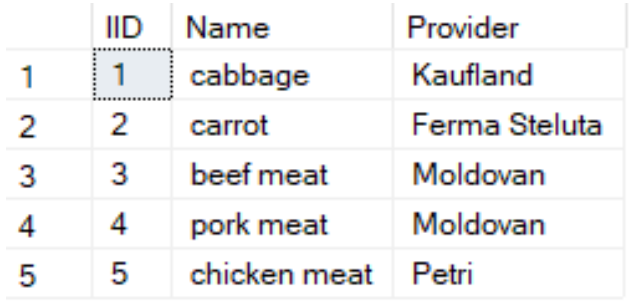
	CID	Name	Surname	Stars	Age
1	1	Paul	Mihai	4	34
2	2	Samuel	Jira	5	27
3	3	Ionut	Moldovan	4	42

Recipies



	CID	IID	Year	RecipyName
1	1	1	2023	vegetable salad
2	1	3	2022	beef soup
3	2	2	2023	carrot cake
4	2	5	2024	chicken noodles

Ingredients



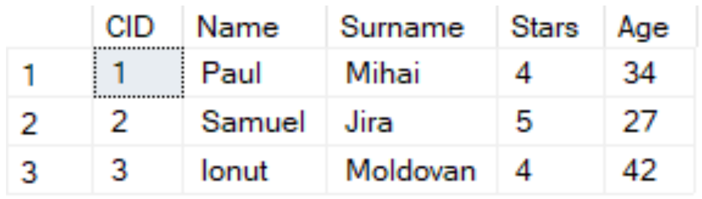
	IID	Name	Provider
1	1	cabbage	Kaufland
2	2	carrot	Ferma Steluta
3	3	beef meat	Moldovan
4	4	pork meat	Moldovan
5	5	chicken meat	Petri

SQL Queries

The JOIN operator

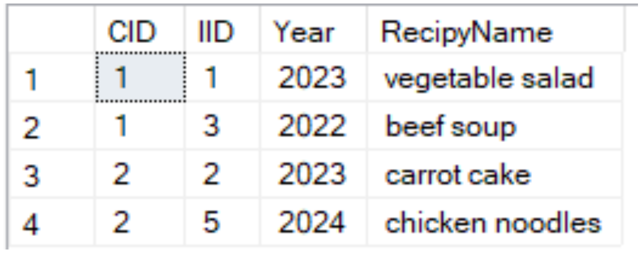
INNER JOIN: source1 [alias] [INNER] JOIN source2 [alias] ON condition

Chefs



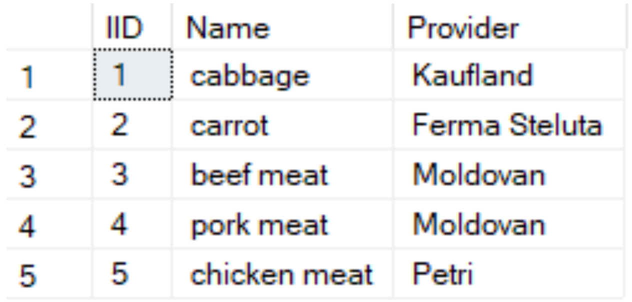
	CID	Name	Surname	Stars	Age
1	1	Paul	Mihai	4	34
2	2	Samuel	Jira	5	27
3	3	Ionut	Moldovan	4	42

Recipies



	CID	IID	Year	RecipyName
1	1	1	2023	vegetable salad
2	1	3	2022	beef soup
3	2	2	2023	carrot cake
4	2	5	2024	chicken noodles

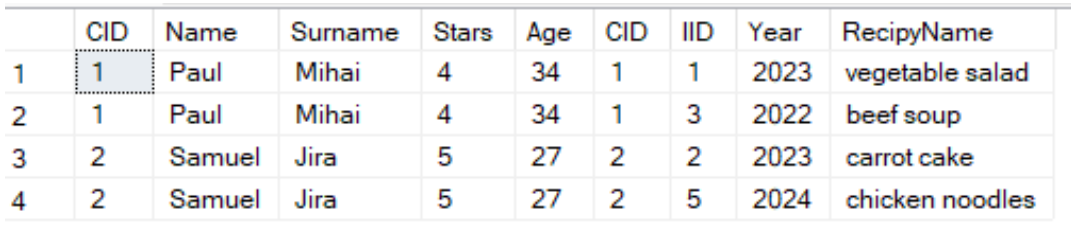
Ingredients



	IID	Name	Provider
1	1	cabbage	Kaufland
2	2	carrot	Ferma Steluta
3	3	beef meat	Moldovan
4	4	pork meat	Moldovan
5	5	chicken meat	Petri

Find all the chefs' recipies; include the chefs' name in the answer set.

```
SELECT *
FROM Chefs C INNER JOIN Recipies R ON C.Cid=R.CID
```



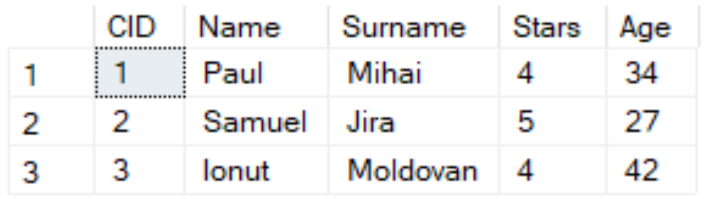
	CID	Name	Surname	Stars	Age	CID	IID	Year	RecipyName
1	1	Paul	Mihai	4	34	1	1	2023	vegetable salad
2	1	Paul	Mihai	4	34	1	3	2022	beef soup
3	2	Samuel	Jira	5	27	2	2	2023	carrot cake
4	2	Samuel	Jira	5	27	2	5	2024	chicken noodles

SQL Queries

The JOIN operator

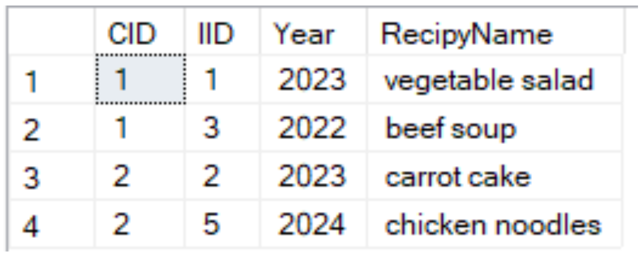
LEFT OUTER JOIN: source1 [alias] LEFT [OUTER] JOIN source2 [alias] ON condition

Chefs



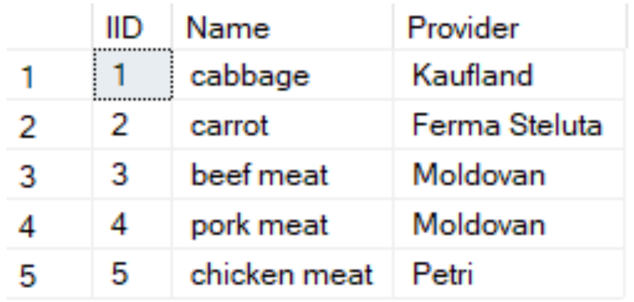
	CID	Name	Surname	Stars	Age
1	1	Paul	Mihai	4	34
2	2	Samuel	Jira	5	27
3	3	Ionut	Moldovan	4	42

Recipies



	CID	IID	Year	RecipyName
1	1	1	2023	vegetable salad
2	1	3	2022	beef soup
3	2	2	2023	carrot cake
4	2	5	2024	chicken noodles

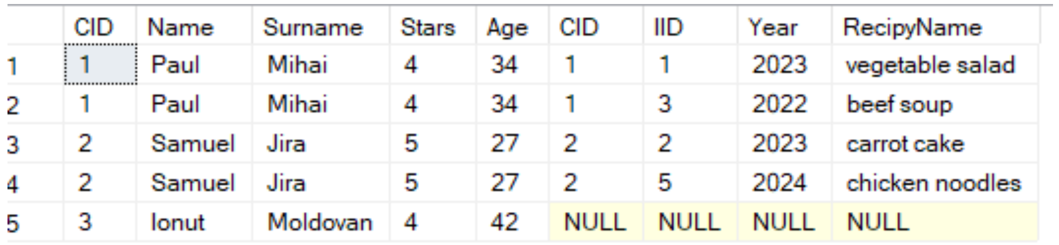
Ingredients



	IID	Name	Provider
1	1	cabbage	Kaufland
2	2	carrot	Ferma Steluta
3	3	beef meat	Moldovan
4	4	pork meat	Moldovan
5	5	chicken meat	Petri

Find all the chefs' recipies; include the chefs' with no recipies; the chefs' name has to be included in the answer set.

SELECT *
FROM Chefs C LEFT JOIN Recipies R ON C.Cid=R.CID



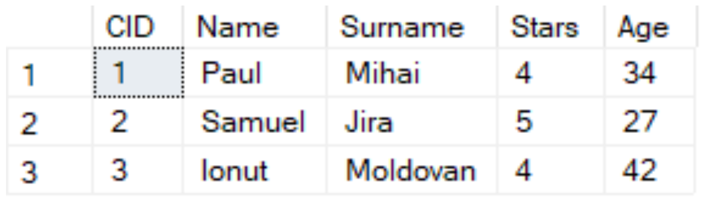
	CID	Name	Surname	Stars	Age	CID	IID	Year	RecipyName
1	1	Paul	Mihai	4	34	1	1	2023	vegetable salad
2	1	Paul	Mihai	4	34	1	3	2022	beef soup
3	2	Samuel	Jira	5	27	2	2	2023	carrot cake
4	2	Samuel	Jira	5	27	2	5	2024	chicken noodles
5	3	Ionut	Moldovan	4	42	NULL	NULL	NULL	NULL

SQL Queries

The JOIN operator

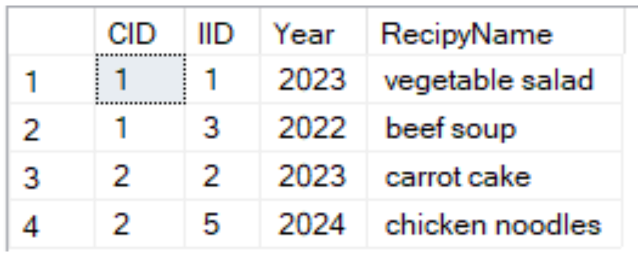
RIGHT OUTER JOIN: source1 [alias] RIGHT [OUTER] JOIN source2 [alias] ON condition

Chefs



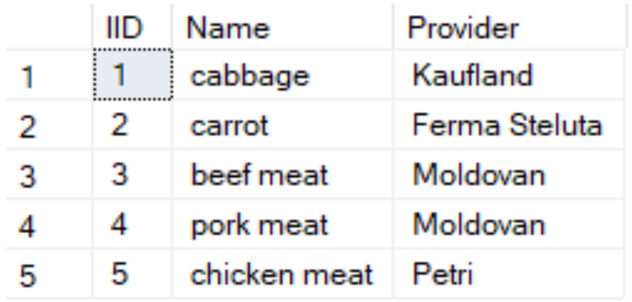
	CID	Name	Surname	Stars	Age
1	1	Paul	Mihai	4	34
2	2	Samuel	Jira	5	27
3	3	Ionut	Moldovan	4	42

Recipies



	CID	IID	Year	RecipyName
1	1	1	2023	vegetable salad
2	1	3	2022	beef soup
3	2	2	2023	carrot cake
4	2	5	2024	chicken noodles

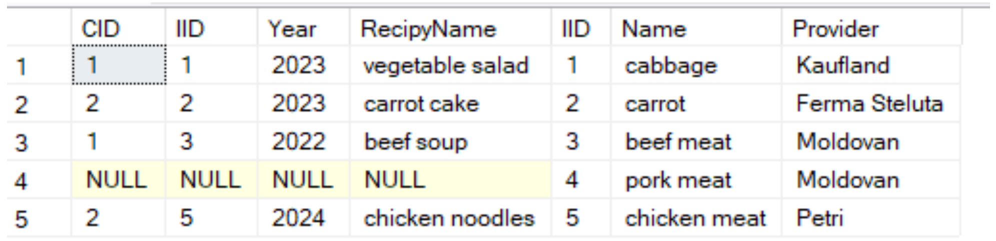
Ingredients



	IID	Name	Provider
1	1	cabbage	Kaufland
2	2	carrot	Ferma Steluta
3	3	beef meat	Moldovan
4	4	pork meat	Moldovan
5	5	chicken meat	Petri

Find all the recipies (including the name of the ingredients); include ingredients who haven't been used in the recipies.

SELECT *
FROM Recipies R RIGHT JOIN Ingredients I ON R.IId=I.IId



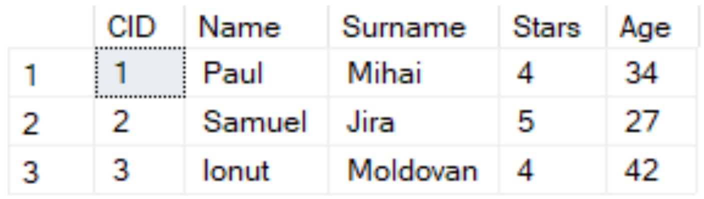
	CID	IID	Year	RecipyName	IID	Name	Provider
1	1	1	2023	vegetable salad	1	cabbage	Kaufland
2	2	2	2023	carrot cake	2	carrot	Ferma Steluta
3	1	3	2022	beef soup	3	beef meat	Moldovan
4	NULL	NULL	NULL	NULL	4	pork meat	Moldovan
5	2	5	2024	chicken noodles	5	chicken meat	Petri

SQL Queries

The JOIN operator

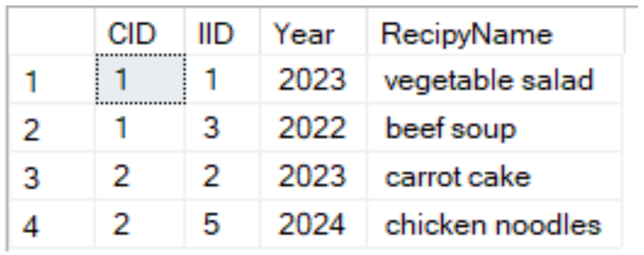
FULL OUTER JOIN: source1 [alias] FULL [OUTER] JOIN source2 [alias] ON condition

Chefs



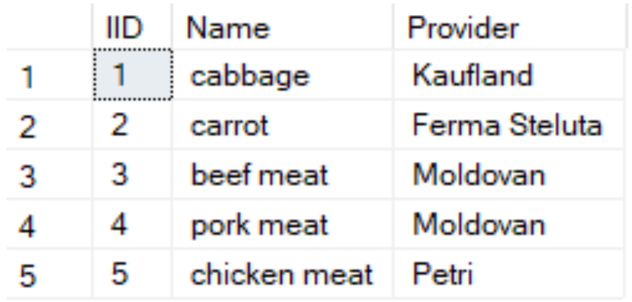
	CID	Name	Surname	Stars	Age
1	1	Paul	Mihai	4	34
2	2	Samuel	Jira	5	27
3	3	Ionut	Moldovan	4	42

Recipies



	CID	IID	Year	RecipyName
1	1	1	2023	vegetable salad
2	1	3	2022	beef soup
3	2	2	2023	carrot cake
4	2	5	2024	chicken noodles

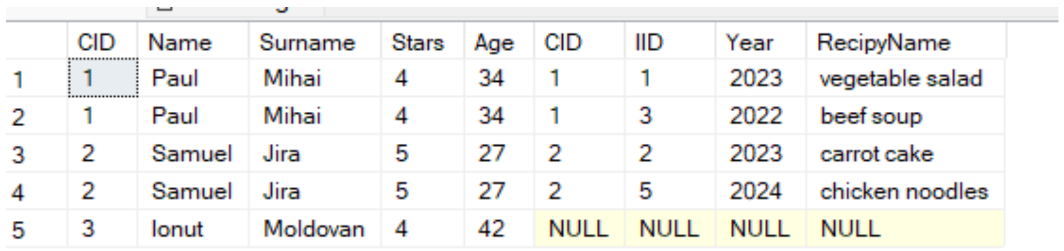
Ingredients



	IID	Name	Provider
1	1	cabbage	Kaufland
2	2	carrot	Ferma Steluta
3	3	beef meat	Moldovan
4	4	pork meat	Moldovan
5	5	chicken meat	Petri

Find all the recipies; include the chefs' with no recipies and the recipies given by mistace to nonexisting chefs; the chefs' name has to be included in the answer set.

SELECT *
FROM Chefs C FULL JOIN Recipies R ON C.Cid=R.CID



	CID	Name	Surname	Stars	Age	CID	IID	Year	RecipyName
1	1	Paul	Mihai	4	34	1	1	2023	vegetable salad
2	1	Paul	Mihai	4	34	1	3	2022	beef soup
3	2	Samuel	Jira	5	27	2	2	2023	carrot cake
4	2	Samuel	Jira	5	27	2	5	2024	chicken noodles
5	3	Ionut	Moldovan	4	42	NULL	NULL	NULL	NULL

SQL Queries

Other JOIN expressions

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

source1 [alias1] NATURAL JOIN source2 [alias2]

source1 [alias1] CROSS JOIN source2 [alias2]

Copy data from one table to another

```
INSERT INTO A2
SELECT * FROM A1
```

SQL Queries

Nested Queries

- **subquery in the FROM clause**

```
SELECT C.*
FROM Chefs C INNER JOIN
            (SELECT *
            FROM Recipies R
            WHERE R.Cid=400) A
ON C.Cid=A.Cid
```

SQL Queries

GROUP BY Queries

Find the age of the youngest chef for each star.

```
SELECT C.Stars, MIN(C.Age)
FROM Chefs C
GROUP BY C.Stars
```

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

Find the age of the youngest chef who is at least 18 years old for each star with at least 10 such chefs.

```
SELECT C.Stars, MIN(C.Age)
FROM Chefs C
WHERE C.Age>=18
GROUP BY C.Stars
HAVING COUNT(*)>=10
```

SQL Queries

GROUP BY Queries

Find the most prolific chef (those with the largest number of ingredients used in their recipies on the schema Recipies(Cid, IId, Year)).
Does it compute the same result on Recipies(Cid, IId, Year)? If not, change it so the intended result is computed on this schema as well.

Find the name and age of the oldest chef.

```
SELECT C.Name, MAX(C.Age)
FROM Chefs C
-- error: if the SELECT clause contains an aggregate operator, then it must contain ONLY aggregation
operators, unless the query has a GROUP BY clause

-- correct query
SELECT C.Name, C.Age
FROM Chefs C
WHERE C.Age=(SELECT MAX(C2.Age) FROM Chefs C2)
```

SQL Queries

ORDER BY, TOP

Sort the chefs by stars (in descending order) and age (in ascending order).

```
SELECT *
FROM Chefs C
ORDER BY C.Stars DESC, C.Age ASC
```

Retrieve the name and the age of the top 10 chefs ordered by name.

```
SELECT TOP 10 C.Name, C.Age
FROM Chefs C
ORDER BY C.Name
```

SQL Queries

ORDER BY, TOP

Find the top 25% chefs (all the data) ordered by age (descending).

```
SELECT TOP 25 PERCENT *
FROM Chefs C
ORDER BY C.Age DESC
```

Find the number of chefs for each star. Order the result by the number of the chefs.

```
SELECT C.Stars, COUNT(*) AS NoS
FROM Chefs C
GROUP BY C.Stars
ORDER BY NoS
```

SQL Queries

Remark: INTERSECTION queries can be expressed with IN

Find the names of chefs who have used as provider Moldovan AND Petri.

```
SELECT C.Name
FROM Chefs C INNER JOIN Recipies R ON C.Cid=R.Cid
INNER JOIN Ingredients I ON I.IId=R.IId
WHERE I.Provider='Moldovan' AND
C.Cid IN (SELECT R2.Cid
                FROM Recipies R2 INNER JOIN Ingredients I2 ON I2.IId=R2.IId
                WHERE I2.Provider='Petri')
```

SQL Queries

Remark: SET-DIFFERENCE queries can be expressed with NOT IN

Find the names of chefs who have used as provider Moldovan, but not Petri.

```
SELECT C.Name
FROM Chefs C INNER JOIN Recipies R ON C.Cid=R.Cid
INNER JOIN Ingredients I ON I.IId=R.IId
WHERE I.Provider='Moldovan' AND
C.Cid NOT IN (SELECT R2.Cid
                          FROM Recipies R2 INNER JOIN Ingredients I2 ON I2.IId=R2.IId
                          WHERE I2.Provider='Petri')
```

SQL Queries

expression>ANY(subquery) equivalent expression>(subquery MIN)

Find chefs whose stars is greater than the stars of some chefs called Paul.

```
SELECT C.Cid
FROM Chefs C
WHERE C.Stars>ANY(SELECT C2.Stars
                  FROM Chefs C2
                  WHERE C2.Name='Paul')
```

```
SELECT C.Cid
FROM Chefs C
WHERE C.Stars>(SELECT MIN(C2.Stars)
               FROM Chefs C2
               WHERE C2.Name='Paul')
```

SQL Queries

expression>ALL(subquery) equivalent expression>(subquery MAX)

Find chefs whose stars is greater than the stars of every (all) chefs called Paul.

```
SELECT C.Cid
FROM Chefs C
WHERE C.Stars>ALL(SELECT C2.Stars
                  FROM Chefs C2
                  WHERE C2.Name='Paul')
```

```
SELECT C.Cid
FROM Chefs C
WHERE C.Stars>(SELECT MAX(C2.Stars)
               FROM Chefs C2
               WHERE C2.Name='Paul')
```

The SELECT statement

```
SELECT [ALL / DISTINCT / TOP n [PERCENT]] * / column1, … / expr1 [AS col1], …
FROM source1 [alias1], … [JOIN / …]
[WHERE qualification]
[GROUP BY grouping_list]
[HAVING group_qualification]
[UNION [ALL] / INTERSECT / EXCEPT SELECT_statement]
[ORDER BY column1 / column1_number [ASC] / [DESC], …]
```

- non-procedural query
- **SELECT statement evaluation**: the result is a **relation (table)**
- data can be obtained from one or multiple data sources; a source can have an associated *alias*, used only in the SELECT statement
- various expressions are evaluated on the data (from the above-mentioned sources)
- a source column can be qualified with the source's name (or alias)

The SELECT statement

A data source can be:
1. table / view in the database
2. (SELECT_statement)
3. join_expression:
• source1 [alias1] join_operator source2 [alias2] ON join_condition
• (join_expression)

* a join condition can be of the form:
• elementary_cond
• (condition)
• NOT condition
• condition1 AND condition2 • condition1 OR condition2

* an elementary join condition (elementary_cond) among two data sources can be of the form:
• [source1_alias.]column1 relational_operator [source2_alias.]column2
• expression1 relational_operator expression2 (expression1 and expression2 use columns from different sources)

The SELECT statement

- the WHERE clause can contain filter and join conditions
- filter conditions:
  - expression relational_operator expression
  - expression [NOT] BETWEEN valmin AND valmax
  - expression [NOT] LIKE pattern
  - expression IS [NOT] NULL
  - expression [NOT] IN (value [, value] ...)
  - expression [NOT] IN (subquery)
  - expression relational_operator {ALL | ANY} (subquery) • [NOT] EXISTS (subquery)
- filter conditions can be:
  - elementary (described above)
  - composed with logical operators and parentheses

The SELECT statement

obs: not all DBMSs support TOP

- MySQL: SELECT ... LIMIT n
- Oracle: SELECT ... WHERE ROWNUM <= n

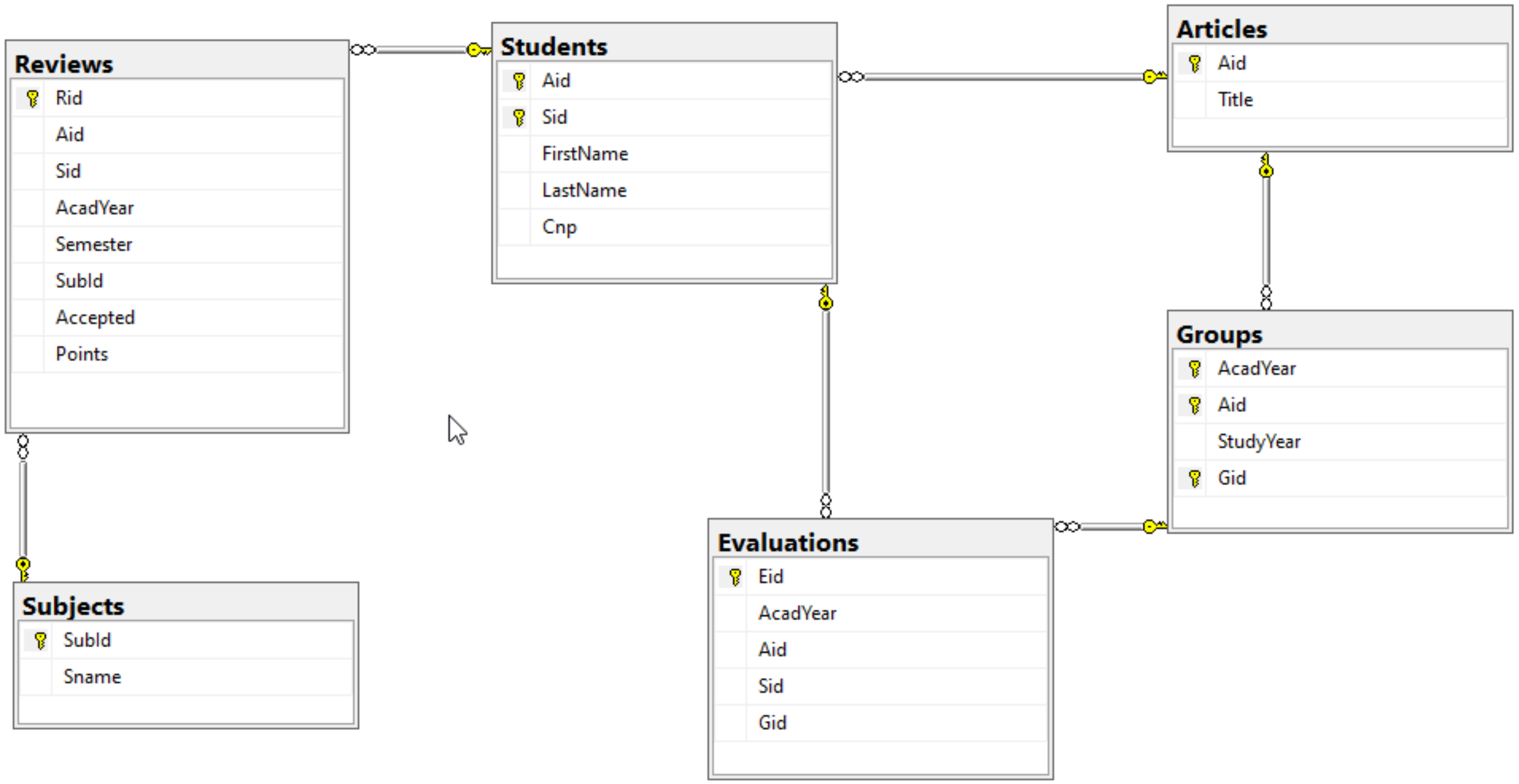
- rules for building expressions:
- operands: constants, columns, system functions, user functions • operators: corresponding to operands
- ordering records: the ORDER BY clause

- the SELECT statement - logical processing (Transact-SQL)

FROM
WHERE
GROUP BY
HAVING
SELECT
DISTINCT
ORDER BY
TOP

Appendix - More SQL Queries

Consider the following database:



More SQL Queries

```
create database Course7_Appendix; go
use Course7_Appendix; go

CREATE TABLE Articles
  (Aid SMALLINT PRIMARY KEY,
  Title VARCHAR(100))

CREATE TABLE Students
(Aid SMALLINT REFERENCES Articles(Aid),
Sid CHAR(10),
FirstName CHAR(50),
LastName CHAR(50),
Cnp CHAR(13) UNIQUE,
PRIMARY KEY(Aid, Sid))

CREATE TABLE Groups
(AcadYear SMALLINT,
Aid SMALLINT REFERENCES Articles(Aid),
StudyYear SMALLINT,
Gid CHAR(10),
PRIMARY KEY (AcadYear, Aid, Gid))
```

```
CREATE TABLE Subjects
(SubId CHAR(10) PRIMARY KEY,
Sname VARCHAR(70))

CREATE TABLE Evaluations
  (Eid INT PRIMARY KEY IDENTITY(1,1),
  AcadYear SMALLINT,
  Aid SMALLINT,
  Sid CHAR(10),
  Gid CHAR(10),
  FOREIGN KEY(Aid, Sid) REFERENCES Students(Aid, Sid),
  FOREIGN KEY(AcadYear, Aid, Gid) REFERENCES Groups(AcadYear, Aid,
Gid))

CREATE TABLE Reviews
(Rid INT PRIMARY KEY IDENTITY(1,1),
Aid SMALLINT,
Sid CHAR(10),
AcadYear SMALLINT,
Semester SMALLINT,
SubId CHAR(10) REFERENCES Subjects(SubId),
Accepted INT, -- 1=yes, 0=not, 2=yes with small corrections, 3 = yes with
corrections
Points INT,
FOREIGN KEY(Aid, Sid) REFERENCES Students(Aid, Sid))
```

More SQL Queries

```
-- a student's evaluation (academic year and group)

SELECT AcadYear, Gid
FROM Evaluations
WHERE Aid = 2 AND Sid = '1214’



-- student's points

SELECT AcadYear, FirstName, LastName, Accepted, Points
FROM Reviews r INNER JOIN Students s ON r.Aid=s.Aid and r.Sid=s.Sid
WHERE r.Aid = 2 AND r.Sid = '7654’
```

More SQL Queries

```
-- Students who belonged to group 822 in the academic year 2020-2021.

SELECT FirstName, LastName, s.Sid
FROM Students s INNER JOIN Evaluations e ON s.Aid=e.Aid AND s.Sid=e.Sid
WHERE AcadYear=2020 AND Gid='822'

SELECT FirstName, LastName, s.Sid
FROM Students s INNER JOIN
            (SELECT *
            FROM Evaluations
            WHERE AcadYear=2020 AND Gid='822’) e
ON s.Sid=e.Sid AND s.Aid=e.Aid
```

More SQL Queries

```
-- Students who belong to a group, but have no points in the academic year 2020-2021.

SELECT FirstName, LastName
FROM Students AS s
WHERE EXISTS
        (SELECT *
        FROM Evaluations e
        WHERE AcadYear=2020 AND e.Aid=s.Aid AND e.Sid=s.Sid)
AND NOT EXISTS
        (SELECT *
        FROM Reviews r
        WHERE AcadYear=2020 AND s.Aid=r.Aid AND s.Sid=r.Sid)
ORDER BY FirstName, LastName
```

More SQL Queries

```
-- Students who belong to a group, but have no points - in the academic year 2020-2021.

SELECT FirstName, LastName
FROM (Students s INNER JOIN
            (SELECT *
            FROM Evaluations
            WHERE AcadYear=2020) t
ON s.Aid=t.Aid AND s.Sid=t.Sid)
LEFT JOIN
            (SELECT *
            FROM Reviews
            WHERE AcadYear=2020) r
ON s.Aid=r.Aid AND s.Sid=r.Sid
WHERE Points IS NULL
```

More SQL Queries

```
-- The number of students in the database.

SELECT COUNT(*) AS NoS
FROM Students

-- The number of students born on the same day, regardless of year and month.

SELECT SUBSTRING(Cnp,6,2) AS DayOfBirth, COUNT(*) AS NoStudents
FROM Students
GROUP BY SUBSTRING(Cnp,6,2)

-- The points of a given student (only the maximum number of points is required for each subject).

SELECT SubId, Points, MAX(Points) AS MaxPoints
FROM Reviews
WHERE Aid = 3 AND Sid='1232'
GROUP BY SubId, Points
ORDER BY SubId
```

More SQL Queries

```
-- The points of a given student (only the maximum number of points is required for each subject).
-- Include the name of the article in the answer set.

SELECT s.SubId, Sname, Points, MaxPoints
FROM Subjects s INNER JOIN
        (SELECT SubId, Points, MAX(Points) AS MaxPoints
        FROM Reviews
        WHERE Aid = 3 AND Sid='1232’
        GROUP BY SubId, Points) r
ON s.SubId = r.SubId
ORDER BY Sname
```

More SQL Queries

```
-- For each student name that appears at least 3 times, retrieve all students with that name.

SELECT *
FROM Students
WHERE LastName IN
            (SELECT LastName
            FROM Students
            GROUP BY LastName
            HAVING COUNT(*)>=3)
ORDER BY LastName, FirstName

-- rewrite the query without IN
```

More SQL Queries

```
-- The number of students in each article (Aid) that are in the study year 2020.

SELECT g.Aid, g.StudyYear, COUNT(*) AS NoStudents
FROM Evaluations t INNER JOIN Groups g
        ON t.Aid=g.Aid AND t.Gid=g.Gid AND t.AcadYear=g.AcadYear
WHERE t.AcadYear=2020
GROUP BY g.Aid, g.StudyYear
```

More SQL Queries

```
-- The last name, first name, Aid (article id), Sid (student id), number of points>5,
-- number of points, and calc for each student with at least 3 points in Subjects at the end of 2020.

SELECT LastName, FirstName, s.Aid, s.Sid, COUNT(*) AS NoofPoints, SUM(Points) AS SumPoints,
SUM(MaxPoints*Points)/SUM(Points) AS calc
FROM Students s INNER JOIN
        (SELECT Aid, Sid, SubId, Points, MAX(Points) AS MaxPoints
        FROM Reviews
        WHERE AcadYear=2020 AND Points>=5
        GROUP BY Aid, Sid, SubId, Points) r
ON s.Aid=r.Aid AND s.Sid=r.Sid
GROUP BY s.Aid, s.Sid, LastName, FirstName
HAVING SUM(Points)>=3
ORDER BY 3,1,2
```

More SQL Queries

```
-- views -- The maximum points and the points >5 in 2020.
CREATE VIEW StudentsPoints AS
SELECT Aid, Sid, SubId, Points, MAX(Points) AS MaxPoints
FROM Reviews
WHERE AcadYear=2020 AND Points>=5
GROUP BY Aid, Sid, SubId, Points

SELECT * FROM StudentsPoints

-- Students' points average in 2020.
CREATE VIEW StudentsAverage AS
SELECT Aid, Sid,
SUM(Points*MaxPoints)/SUM(Points) AS PointsAverage
FROM StudentsPoints
GROUP BY Aid, Sid
HAVING SUM(Points) >= 30

SELECT * FROM StudentsAverage
```

More SQL Queries

```
-- average of the points
SELECT Gid, AVG(PointsAverage) AS AvgPoints
FROM
            (SELECT Aid, Sid, Gid
            FROM Evaluations
            WHERE AcadYear=2020) t
INNER JOIN StudentsAverage AS s ON t.Sid=s.Sid AND t.Aid=s.Aid
GROUP BY Gid

-- The average points for each article Aid in 2020
CREATE VIEW AidAverage AS
SELECT p.Aid, AVG(PointsAverage) AS PointsAvg
FROM StudentsPoints p INNER JOIN StudentsAverage s ON p.Aid=s.Aid
GROUP BY p.Aid

SELECT * FROM AidAverage
```

More SQL Queries

```
-- For every subject, the number of points and the number of points>5 in 2020.

-- a. MySQL

SELECT Sname, COUNT(*) AS NoPoints, SUM(IF(r.Points >= 5,1,0)) AS NoPoints5
FROM Subjects s INNER JOIN
        (SELECT *
        FROM Reviews
        WHERE AcadYear = 2020) r
ON s.SubId = r.SubId
GROUP BY Sname
ORDER BY 1
```

More SQL Queries

- For every subject, the number of points and the number of points>5 in 2020.

```
-- b. Oracle

SELECT Sname, COUNT(*) AS NoPoints, SUM(CASE WHEN (r.Points>= 5) THEN 1 ELSE 0 END) AS
NoPoints5
FROM Subjects s INNER JOIN
		(SELECT *
		FROM Reviews
		WHERE AcadYear = 2020) r
ON s.SubId = r.SubId
GROUP BY Sname
ORDER BY 1
```

References:

- C.J. Date, *An Introduction to Database Systems (8th Edition)*, Addison-Wesley, 2003.
- H. Garcia-Molina, J. Ullman, J. Widom, *Database Systems: The Complete Book, Prentice Hall Press*, 2008.
- G. Hansen, J. Hansen, *Database Management And Design (2nd Edition),* Prentice Hall, 1996.
- R. Ramakrishnan, J. Gehrke, *Database Management Systems*, McGraw- Hill, 2007. http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- R. Ramakrishnan, J. Gehrke, *Database Management Systems (2nd Edition),* McGraw-Hill, 2000.
- A. Silberschatz, H. Korth, S. Sudarshan, *Database System Concepts*, McGraw-Hill, 2010. http://codex.cs.yale.edu/avi/db-book/
- L. Țâmbulea, *Curs Baze de date*, Facultatea de Matematică și Informatică, UBB, 2013-2014.
- J. Ullman, J. Widom, *A First Course in Database Systems*, http://infolab.stanford.edu/~ullman/fcdb.html