DATABASES

Source: https://en.itpedia.nl/2017/11/26/wat-is-een-database/

Prof. Dr. Ulrike Herster
Hamburg University of Applied Sciences

HAW HAMBURG

COPYRIGHT

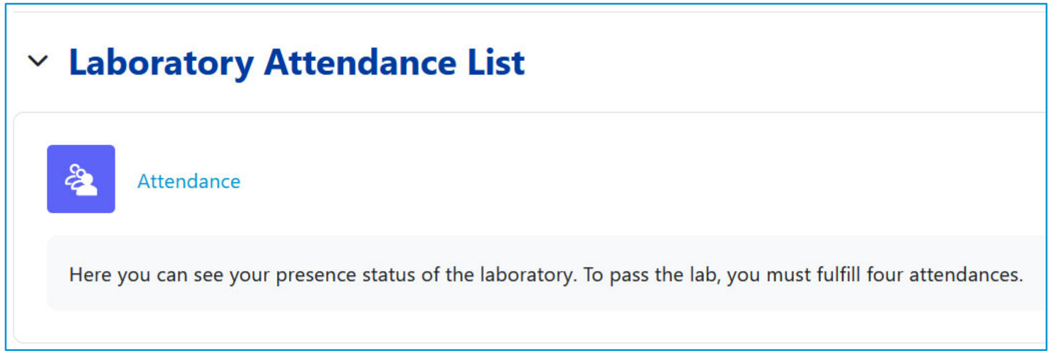
The publication and sharing of

slides, images and sound recordings of this

course is not permitted

LABORATORY ATTENDANCE LIST

- In our moodle room you find a *Laboratory Attendance List*



- This list documents
  - Your attendance for the four labs
  - Comments, e.g. about a presentation you did within a laboratory
- After each laboratory you have time until the following Friday to report incorrect comments / absences by e-mail to the lecturer of that lab (Moldenhauer, Yildirim, or Herster), e.g., for the first laboratory on 29.04.2024 you have time until Friday, 03.05.2024

ORGANIZATION
OUR JOURNEY IN THIS SEMESTER

Repetition

- Integrity, Trigger & Security
- Database Applications
- Transactions
- Subqueries & Views
- **More SQL**
- Notations & Guidelines
- Constraints
- Relationships
- Simple Entities and Attributes
- Basics

Source: Foto von Justin Kauffman auf Unsplash

MORE SQL
DATABASE DESIGN

Repetition

Conceptual Design
- Entity Relationsship Model (ERM)

Logical Design
- Relational Model (RM)

Implementation & Usage
- Implementation using SQL

MORE SQL
DATA DEFINITION

Repetition

- **CREATE** table
  - **DEFAULT, AUTO_INCREMENT, NOT NULL, PRIMARY KEY, …**
- **ALTER** table
  - **ADD, DROP, MODIFY**
  - Column, constraint
  - **RENAME** table
- **DROP**
  - **CASCADE, RESTRICT**

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

MORE SQL
DATA MANIPULATION

Repetition

- `INSERT`
  - Constant tuples
  - Tuples returned by a query
- `UPDATE`
- `DELETE`
- All modifications need to observe constraints
  → Domain constraints, Primary Key, Referential Integrity constraints, …

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

MORE SQL QUERYS

Repetition

SELECT – Basic form

```
SELECT  <attribute list>
FROM    <table list>
WHERE   <condition>
```

- `<attribute list>` is a list of attribute names (columns) whose values are to be retrieved by the query
- `<table list>` is a list of the relation names (e.g., tables) required to process the query
- `<condition>`: optional conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

HAW HAMBURG

MORE SQL QUERYS

Repetition

Syntax:

```
SELECT [ DISTINCT | ALL ] < attribute_list >
FROM < table list >
[ WHERE < condition > ]
[ <group by clause > ]
[ <having clause > ]
[ UNION [ ALL ] < query specification> ]
[ < order by clause > ]
```

MORE SQL QUERYS

Repetition

One big difference between Relational Model and SQL:

- SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values.
  → SQL table is not a set of tuples, it is a multiset

- Some SQL relations are constrained to be sets because
  - a key constraint has been declared or
  - the DISTINCT option has been used with the SELECT statement

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

MORE SQL
QUERYS: ATTRIBUTE LIST

Repetition

- SQL uses (mainly) multiset semantics
  - No elimination of duplicates
  - No duplicates wanted: use **`DISTINCT`**

- Ambiguity of attribute
  - The same name can be used for two (or more) attributes as long as the attributes are in different relations
  - If this is the case, and a multi-table query refers to two or more attributes with the same name, we must qualify the attribute name with the relation's name
  - This is done by prefixing the relation's name to the attribute name and separating the two by a period

MORE SQL
QUERYS: SET OPERATIONS

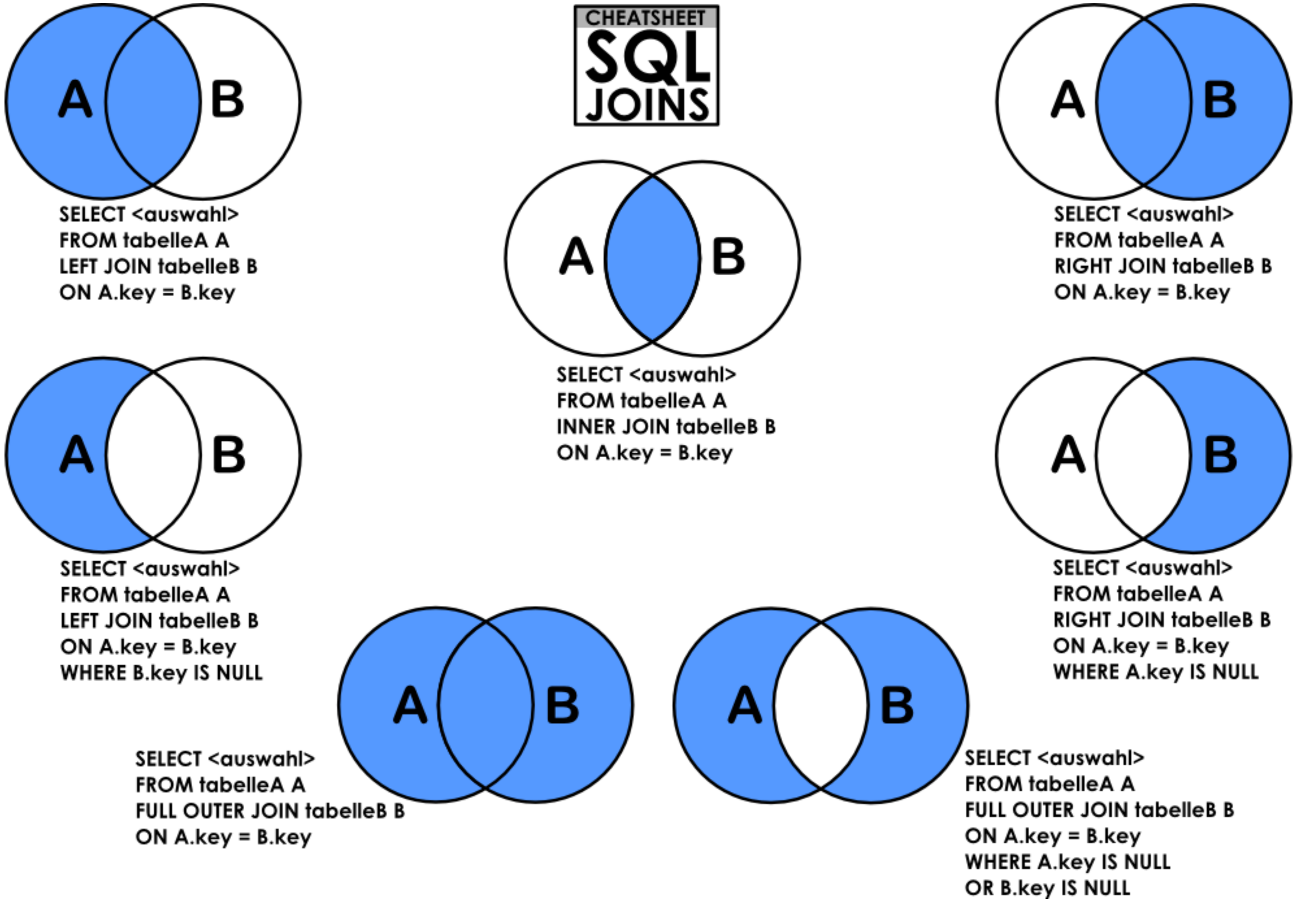
Repetition

- SQL has incorporated some of the set operations:
  - **UNION**
  - **EXCEPT**
  - **INTERSECT**

- SQL has also the corresponding multiset operations (keyword **ALL**):
  - **UNION ALL**
  - **EXCEPT ALL**
  - **INTERSECT ALL**

MORE SQL
JOIN OF TABLES: OVERVIEW

Repetition



Source: https://stackoverflow.com/questions/59590346/trying-to-do-a-left-join-but-ending-up-getting-empty-result

MORE SQL
SPECIAL FEATURES: WHERE CLAUSE

Repetition

- **BETWEEN**

```
... WHERE age >= 18 AND age <= 21 ;
... WHERE age BETWEEN 18 AND 21 ;
```

- **IN**

```
... WHERE DId = 4 OR DId = 5 OR Did = 7 ;
... WHERE DId IN ( 4 , 5 , 7 ) ;
```

- Searching for string patterns
  - Search for patterns using **LIKE** and wildcards
    - _(underscore): replaces a single character
    - % : replaces an arbitrary number of zero or more characters
  - Escape '\' for literals '%' and '_' in strings
    → E.g., 'AB_CD' represents the string "AB_CD"

MORE SQL
SPECIAL FEATURES: SORTING OF RESULTS

Repetition

- Results are (multi-)sets
  → No defined order!
- Order wanted: use ORDER BY
  - **ASC** (default): ascending order
  - **DESC**: descending order
  - Precondition: Datatype defines order
    - For **VARCHAR** it depends on locale
  - Ordering for more than one column is possible

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

MORE SQL
SPECIAL FEATURES: AGGREGATE FUNCTIONS

Repetition

- Summarize information from multiple tuples into a single-tuple summary
  - Analyze column values
  - Return one value for many rows (data reduction)
  - NULL values do not count!

  - **`COUNT`**, **`SUM`**, **`AVG`**, **`MAX`**, **`MIN`**
    - **`COUNT(*)`**: number of rows
    - **`COUNT(DISTINCT a)`**: count different values

  - Can be used in **`SELECT`** clause and **`HAVING`** clause

! - Attention: Not allowed in **`WHERE`** clause!

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

MORE SQL
SPECIAL FEATURES: GROUP BY

Repetition

- Grouping is used to create subgroups of tuples before summarization
  → partition the relation into nonoverlapping subsets (or groups) of tuples
  - Using a grouping attribute
  - Grouping attribute should appear in the **SELECT** clause
  - If NULLs exist in the grouping attribute, then a separate group is created for all tuples with a NULL value

- Example: For each department, retrieve the department number, the number of employees in the department, and their average salary

```
SELECT     Dno, COUNT(*), AVG(Salary)
FROM       Employee
GROUP BY   Dno ;
```

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

MORE SQL
SPECIAL FEATURES: HAVING

Repetition

Example: For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project

```
SELECT      Pnumber, Pname, COUNT(*)
FROM        Project, Works_on, Employee
WHERE       Pnumber = Pno AND SSN = ESSN AND Dno = 5
GROUP BY    Pnumber, Pname ;
```

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

MORE SQL
SPECIAL FEATURES: HAVING

Repetition

- **`HAVING`** provides a condition on the summary information regarding the group of tuples associated with each value of the grouping attributes
  - Only the groups that satisfy the condition are retrieved in the result of the query
  - **`HAVING`** clause appears in conjunction with **`GROUP BY`** clause

- Note:
  - Selection conditions in **`WHERE`** clause limit the tuples
  - **`HAVING`** clause serves to choose whole groups

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

MORE SQL
SPECIAL FEATURES: HAVING

Repetition

Example: For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project

```
SELECT      Pnumber, Pname, COUNT(*)
FROM        Project, Works_on
WHERE       Pnumber = Pno
GROUP BY    Pnumber, Pname
HAVING      COUNT(*) > 2;
```

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

MORE SQL
SPECIAL FEATURES: ASSIGNMENT

- How many students are studying CS?
- List all majors and the number of students which have at least 2 or more students with this major.
- List all course names and how often they have been taught.
- For each section taught by Professor Anderson, retrieve the course number, semester, year, and number of students who took the section.

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

MORE SQL
RELATIONAL ALGEBRA

- SQL → What!
  Relational Algebra → How!

- In mathematics an algebra is a values range combined with defined operations

- *Relational Algebra*: The values range is the content of the database; operations are functions to calculate the query results
  → a set of operations for the relational model

- *Relational Calculus*: Descriptive approach that is based on mathematical logic
  → higher-level declarative language for specifying relational queries, e.g., no order of operations, only what information the result should contain

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: OVERVIEW

- *Algebra operations* produce new relations
- These can be further manipulated using operations of the same algebra
- Sequence of relational algebra operations:
  relational algebra expression
- The result of a *relational algebra expression* is also a relation
- ... representing the result of a database query (retrieval request)

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: OVERVIEW

- *Algebra operations* can be divided into two groups
  - First group consists of operations developed specifically for relational databases
    → i.e., Selection, Projection, and Join
  - Second group includes set operations from mathematical set theory
    → i.e., Union, Intersection, Set Difference, and Cartesian Product

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: OVERVIEW

- Order of explanation
  - Selection
  - Projection
  - Renaming
  - Union, Intersection, Set Difference
  - Cartesian Product
  - Join (Equijoin, Natural Join)

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: QUERYS

SELECT – Basic form

```
SELECT <attribute list>      → Projection
FROM   <table list>
WHERE  <condition>           → Selection
```

- `<attribute list>` is a list of attribute names (columns) whose values are to be retrieved by the query
- `<table list>` is a list of the relation names (e.g., tables) required to process the query
- `<condition>`: optional conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

MORE SQL
RELATIONAL ALGEBRA: SELECTION

- **Selection** (σ): mask out rows
  - Specify, which rows should remain (subset of the tuple)
    - Usage of selection: Specify, which tuples are interesting
    - Selection condition is a Boolean expression (condition)
    - The condition may contain complex expressions (combinations)
  - Specify, which relation is meant
    - Notice that R is generally a relational algebra expression whose result is a relation, e.g., a relation
  - Syntax: $\sigma_{selection\ condition}(R)$
  - Example: $\sigma_{Salary>30{,}000}(Employee)$

$$\sigma_{(DNr=4\ AND\ Salary>30{,}000) OR (DNr=5\ AND\ Salary>25{,}000)}(Employee)$$

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: SELECTION

- Example:

$Person$

<u>SSN</u>	Last Name	First Name	Mobile	…
123456789	Miller	Jane	0044 7701 123456	…
234567891	Miller	Steven	0044 7701 123457	…
345678912	Smith	Maria		…
…	…	…	…	…

$\sigma_{LastName="Miller"}\ (Person)$

<u>SSN</u>	Last Name	First Name	Mobile	…
123456789	Miller	Jane	0044 7701 123456	…
234567891	Miller	Steven	0044 7701 123457	…

MORE SQL
RELATIONAL ALGEBRA: SELECTION

Note:

- Selection is unary (apply to a single relation)
- The degree of the relation resulting from a Selection is the same as the degree of R
- The number of tuples in the resulting relation is always less than or equal to the number of tuples in R

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: SELECTION

Selection condition is typically specified in the **WHERE** clause of a SQL query

Example: $\sigma_{Salary>30{,}000}(Employee)$

```
SELECT *
FROM Employee
WHERE Salary > 30,000
```

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: PROJECTION

- **Projection** (π): mask out columns
  - Specify, which columns should remain
  - Specify, which relation is meant
  - Syntax: $\pi_{attribute\ list}(R)$
  - Example: $\pi_{SSN,LastName}(Person)$

$Person$

SSN	Last Name	First Name	Mobile	…
123456789	Miller	Jane	0044 7701 123456	…
234567891	Miller	Steven	0044 7701 123457	…
345678912	Smith	Maria		…
…	…	…	…	…

$\pi_{SSN,LastName}(Person)$

SSN	Last Name
123456789	Miller
234567891	Miller
345678912	Smith
…	…

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: PROJECTION

Note:

- The degree of the result is equal to the number of attributes in attribute list
- If the attribute list includes only non-key attributes of R, duplicate tuples are likely to occur
  → The Projection removes any duplicate tuples,
  so, the result of the Projection is a set of distinct tuples, and hence a valid relation
- The number of tuples in a relation resulting from a Projection is always less than or equal to the number of tuples in R

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: PROJECTION

The Projection attribute list is specified in the **SELECT** clause of a SQL query

Example: $\pi_{LastName}(Person)$

```
SELECT LastName
FROM Person
```

Person

SSN	Last Name	First Name	Mobile	…
123456789	Miller	Jane	0044 7701 123456	…
234567891	Miller	Steven	0044 7701 123457	…
345678912	Smith	Maria		…
…	…	…	…	…

Use DISTINCT!!

$\pi_{LastName}(Person)$

Last Name
Miller
Miller
Smith
…

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: RENAMING

- **Renaming** (ρ): Column gets new name
  - Specify, which column
  - Specify, which new name
  - Specify, which relation
  - Set theory: Union (∪), Intersection (∩) and Set Difference (−) are only defined for the same relation schema
    → To achieve similar relation schema use projection and renaming
  - Renaming allows the renaming of attributes and relations

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: RENAMING

- Renaming in SQL is accomplished by aliasing using **AS**
- Example: $\rho_{surname \leftarrow LastName}(Person)$

```
SELECT    SSN, LastName AS Surname
FROM      Person
```

Person

SSN	Last Name	First Name	Mobile	…
123456789	Miller	Jane	0044 7701 123456	…
234567891	Miller	Steven	0044 7701 123457	…
345678912	Smith	Maria		…
…	…	…	…	…

$\rho_{ID \leftarrow GenreID}(Genre)$

SSN	Surname
123456789	Miller
234567891	Miller
345678912	Smith
…	…

MORE SQL
RELATIONAL ALGEBRA: UNION, INTERSECTION AND SET DIFFERENCE

- Union, intersection, and set difference can only be applied on two relations that are *union compatible*
  - Union compatible means that the two relations have the same number of attributes and
  - each corresponding pair of attributes has the same domain

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: UNION, INTERSECTION AND SET DIFFERENCE

- **Union** ∪
  - Example: Retrieve the Social Security numbers of all employees who
    - either work in department 5
    - or directly supervise an employee who works in department 5

$$DEP5\_EMPS \longleftarrow \sigma_{DNr=5}(Employee)$$
$$\mathrm{RESULT1} \longleftarrow \pi_{SSN}(DEP5\_EMPS)$$
$$\mathrm{RESULT2} \longleftarrow \pi_{Superssn}(DEP5\_EMPS)$$
$$\mathrm{RESULT} \longleftarrow RESULT1 \cup RESULT2$$

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: UNION, INTERSECTION AND SET DIFFERENCE

- **Union** ∪
  - Example: Retrieve the Social Security numbers of all employees who
    - either work in department 5
    - or directly supervise an employee who works in department 5

$$DEP5\_EMPS \longleftarrow \sigma_{DNr=5}(Employee)$$
$$\mathrm{RESULT1} \longleftarrow \pi_{SSN}(DEP5\_EMPS)$$
$$\mathrm{RESULT2} \longleftarrow \pi_{Superssn}(DEP5\_EMPS)$$
$$\mathrm{RESULT} \longleftarrow RESULT1 \cup RESULT2$$

```
(SELECT     ssn
FROM        Employee
WHERE       Dno = 5)
UNION
(SELECT     super_ssn
FROM        EMPLOYEE
WHERE       dno = 5);
```

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: UNION, INTERSECTION AND SET DIFFERENCE

Student

FirstName	LastName
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wand
Ernest	Gilbert

Instructor

FirstName	LastName
John	Smith
Ricardo	Brown
Susan	Yao
Francis	Johnson
Ramesh	Shah

Student ∪ Instructor

FirstName	LastName
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wand
Ernest	Gilbert
John	Smith
Ricardo	Brown
Francis	Johnson

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: UNION, INTERSECTION AND SET DIFFERENCE

- **Intersection ∩**
  - The result $(R \cap S)$ is a relation that includes all tuples that are in both R and S
  - Commutative
    → $R \cap S = S \cap R$
  - Duplicate tuples are eliminated

- Example:

```
SELECT supplier_id FROM Suppliers
INTERSECT
SELECT supplier_id FROM Orders ;
```

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: UNION, INTERSECTION AND SET DIFFERENCE

Student

FirstName	LastName
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wand
Ernest	Gilbert

Instructor

FirstName	LastName
John	Smith
Ricardo	Brown
Susan	Yao
Francis	Johnson
Ramesh	Shah

Student ∩ Instructor

FirstName	LastName
Susan	Yao
Ramesh	Shah

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: UNION, INTERSECTION AND SET DIFFERENCE

- **Set Difference (-)**
  - The result $(R - S)$ is a relation that includes all tuples that are in R but not in S
  - Not commutative
    → $R - S \neq S - R$
  - Duplicate tuples are eliminated

- Example SQL:
  `Student **EXCEPT** Instructor`
- Example Oracle:
  `Student **MINUS** Instructor`

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: UNION, INTERSECTION AND SET DIFFERENCE

Student

FirstName	LastName
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wand
Ernest	Gilbert

Instructor

FirstName	LastName
John	Smith
Ricardo	Brown
Susan	Yao
Francis	Johnson
Ramesh	Shah

Student - Instructor

FirstName	LastName
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wand
Ernest	Gilbert

Instructor - Student

FirstName	LastName
John	Smith
Ricardo	Brown
Francis	Johnson

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: UNION, INTERSECTION AND SET DIFFERENCE

Student

FirstName	LastName
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wand
Ernest	Gilbert

Instructor

FirstName	LastName
John	Smith
Ricardo	Brown
Susan	Yao
Francis	Johnson
Ramesh	Shah

Student ∪ Instructor

FirstName	LastName
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wand
Ernest	Gilbert
John	Smith
Ricardo	Brown
Francis	Johnson

Student - Instructor

FirstName	LastName
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wand
Ernest	Gilbert

Instructor - Student

FirstName	LastName
John	Smith
Ricardo	Brown
Francis	Johnson

Student ∩ Instructor

FirstName	LastName
Susan	Yao
Ramesh	Shah

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: UNION, INTERSECTION AND SET DIFFERENCE

- In SQL, there are three operations — **`UNION`**, **`INTERSECT`**, and **`EXCEPT`** — that correspond to the set operations described here

- In addition, there are multiset operations — **`UNION ALL`** , **`INTERSECT ALL`** , and **`EXCEPT ALL`** — that do not eliminate duplicates

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

RELATIONAL ALGEBRA: CARTESIAN PRODUCT

- **Cartesian Product**
  - This is also a binary set operation
  - Relations do not have to be union compatible
  - The result $(\mathrm{A} \times B)$ is the combination of each tuple of the first relation A with each tuple of the second one B
  - In general, the result of $R(A_1, A_2, \dots A_n) \times S(B_1, B_2, \dots B_m)$ is a relation Q with degree $\mathrm{n} + \mathrm{m}$ attributes $\mathrm{Q}(A_1, A_2, \dots A_n, B_1, B_2, \dots B_m)$
  - If R has n_R tuples, and S has n_S tuples, then Q will have $\mathrm{n_R} * \mathrm{n_S}$ tuples
  - In SQL, the Cartesian Product can be realized by using the Cross Join option in joined tables

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: CARTESIAN PRODUCT

- Example: Person $\boldsymbol{Person} \times \boldsymbol{House}$

Person

SSN	Last Name	First Name	Mobile
123456789	Miller	Jane	0044 7701 123456
234567891	Miller	Steven	0044 7701 123457
…	…	…	…

House

Address	Phone
221 Baker Street, 1NW London	0044 20 7946 0000
112 Baker Street, 1NW London	0044 20 7946 1000
…	…

$\boldsymbol{Person} \times \boldsymbol{House}$

SSN	Last Name	First Name	Mobile	Address	Phone
123456789	Miller	Jane	0044 7701 123456	221 Baker Street, 1NW London	0044 20 7946 0000
123456789	Miller	Jane	0044 7701 123456	112 Baker Street, 1NW London	0044 20 7946 1000
234567891	Miller	Steven	0044 7701 123457	221 Baker Street, 1NW London	0044 20 7946 0000
234567891	Miller	Steven	0044 7701 123457	112 Baker Street, 1NW London	0044 20 7946 1000
…	…	…	…	…	…

MORE SQL
RELATIONAL ALGEBRA: CARTESIAN PRODUCT

- Example: retrieve a list of names of each female employee and her dependents

$$\text{FEMALE\_}EMPS \leftarrow \sigma_{Sex='F'}(Employee)$$
$$\text{EMPNAMES} \leftarrow \pi_{Fname,Lname,\,SSN}(FEMALE\_EMP)$$
$$\text{EMP\_DEPENDEN}TS \leftarrow EMPNAMES \times DEPENDENT$$
$$\text{ACTUAL\_DEPENDENTS} \leftarrow \sigma_{SSN=ESSN}(EMP\_DEPENDENTS)$$
$$\text{RESULT} \leftarrow \pi_{Fname,\,Lname,\,Dependent\_name}(ACTUAL\_DEPENDENTS)$$

FEMALE_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	1968-07-19	3321Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291Berry, Bellaire, TX	F	43000	888665555	4
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

EMPNAMES

Fname	Lname	Ssn
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Joyce	English	453453453

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: CARTESIAN PRODUCT

- Example:

EMP_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	...
Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	...
Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	...
Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	...
Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	...
Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	...
Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	...
Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	...
Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	...
Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...
Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	...
Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	...
Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	...
Joyce	English	453453453	333445555	Alice	F	1986-04-05	...
Joyce	English	453453453	333445555	Theodore	M	1983-10-25	...
Joyce	English	453453453	333445555	Joy	F	1958-05-03	...
Joyce	English	453453453	987654321	Abner	M	1942-02-28	...
Joyce	English	453453453	123456789	Michael	M	1988-01-04	...
Joyce	English	453453453	123456789	Alice	F	1988-12-30	...
Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	...

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

HAW HAMBURG

MORE SQL
RELATIONAL ALGEBRA: CARTESIAN PRODUCT

- Example:

$$\mathrm{FEMALE\_}EMPS \leftarrow \sigma_{Sex='F'}(Employee)$$
$$\mathrm{EMPNAMES} \leftarrow \pi_{Fname,\,Lname,\,SSN}(FEMALE\_EMP)$$
$$\mathrm{EMP\_DEPENDEN}TS \leftarrow EMPNAMES \times DEPENDENT$$
$$\mathrm{ACTUAL\_DEPENDENTS} \leftarrow \sigma_{SSN=ESSN}(EMP\_DEPENDENTS)$$
$$\mathrm{RESULT} \leftarrow \pi_{Fname,\,Lname,\,Dependent\_name}(ACTUAL\_DEPENDENTS)$$

ACTUAL_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...

RESULT

Fname	Lname	Dependent_name
Jennifer	Wallace	Abner

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: JOIN

- **Join** ($\bowtie$):
  - Combine related tuples from two relations into single "*longer*" tuples
  - Very important!!!
  - Specify, which tables should be combined
  - The same attribute name merges
  - Without same attributes: the join is the cartesian product
  - There are different types of joins, which are presented later in more detail
  - Comparison to Cartesian Product: The result has one tuple for each combination of tuples of the two relations whenever the combination satisfies the join condition

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: JOIN

- Example: retrieve all attributes of the managers of each department

$$\text{DEPT\_MGR} \leftarrow DEPARTMENT \bowtie_{Mgr\_SSN=\ SSN} (Employee)$$

DEPT_MGR

Dname	Dnumber	Mgr_ssn	. . .	Fname	Minit	Lname	Ssn	. . .
Research	5	333445555	. . .	Franklin	T	Wong	333445555	. . .
Administration	4	987654321	. . .	Jennifer	S	Wallace	987654321	. . .
Headquarters	1	888665555	. . .	James	E	Borg	888665555	. . .

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: JOIN

- Example
  - With Cartesian Product:

$$\text{EMP\_DEPENDENTS} \longleftarrow EMPNAMES \times DEPENDENT$$
$$\text{ACTUAL\_DEPENDENTS} \longleftarrow \sigma_{SSN=ESSN}(EMP\_DEPENDENTS)$$

  - With Equijoin:

$$\text{ACTUAL\_DEPENDENTS} \longleftarrow EMPNAMES \bowtie_{SSN=\ ESSN} (DEPENDENT)$$

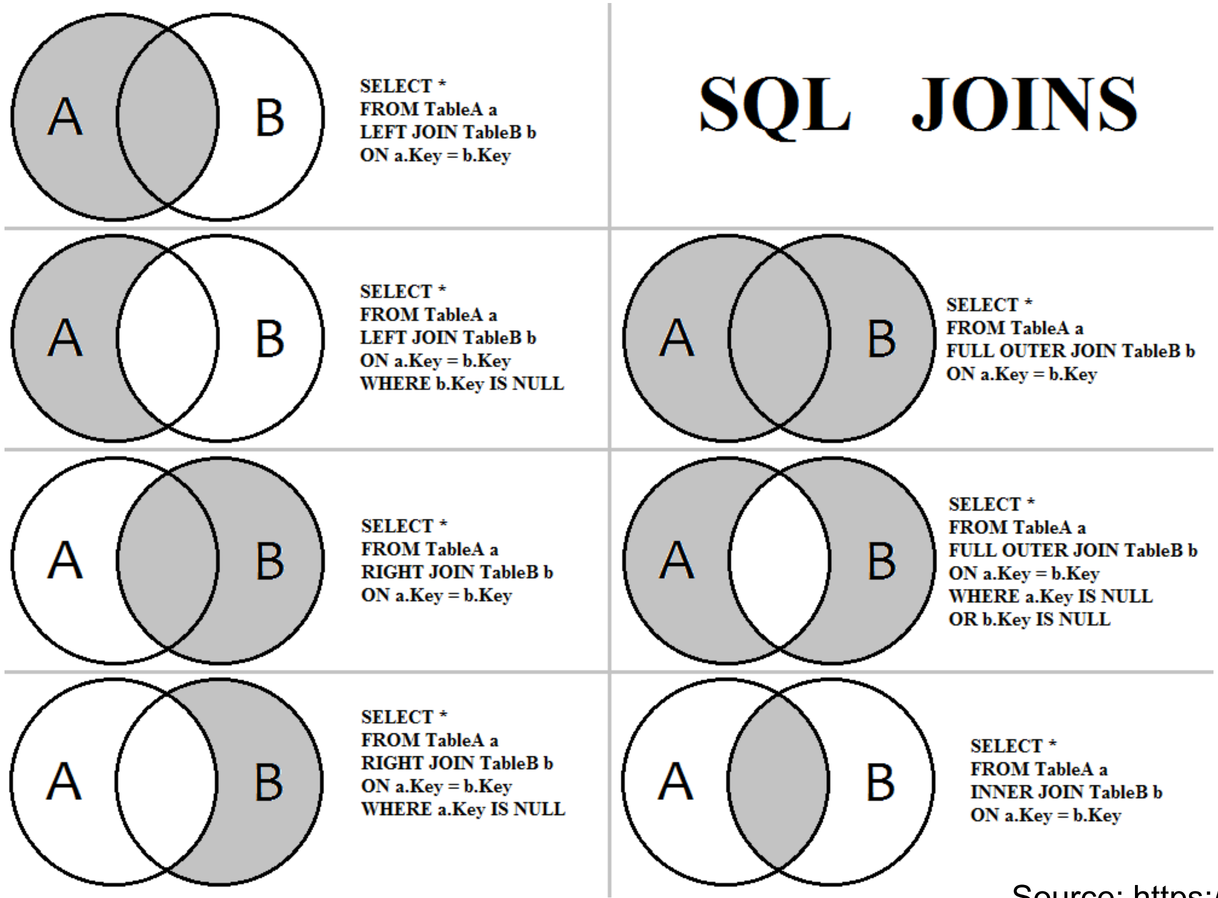
Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: JOIN

- Variations of JOIN:
  - Equijoin
    - Used comparison operator is = only
      → two attributes requires the values to be identical in every tuple in the result
  - Natural Join
    - Two join attributes (or each pair of join attributes) have the same name in both relations.
      → If this is not the case, a renaming operation is applied first

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: JOIN



Source: https://huklee.github.io/2017/01/28/021.SQL-all-kinds-of-join-queries/

MORE SQL
RELATIONAL ALGEBRA: SIZE OF RESULT

- How many tuples are in my result set (cardinality)?
  - Interesting question for end user
    ("I'll just print it!")
  - Interesting question for programmer
    ("Program is running forever?!")
  - Interesting question for DBMS creator
    ("I'll start with operation 1 and do operation 2 afterwards")

- The answer to this question depends on involved operations...

MORE SQL
RELATIONAL ALGEBRA: SIZE OF RESULT

- The answer to this question depends on involved operations:

- Projection
  - Upper bounds: number of tuples in the projected relation
  - Lower bounds: 1 (for not empty original relation)
  - Rule: If the projected attribute contains a key candidate, then the cardinality of the result is equal to the amount of tuples
  - This rule also applies if the attributes of the current database state are coincidentally a key candidate

MORE SQL
RELATIONAL ALGEBRA: SIZE OF RESULT

- The answer to this question depends on involved operations:

- Selection
  - The cardinality of the selection depends on the selection conditions
  - Upper bounds: amount of tuples
  - Lower bounds: 0
  - Selection is used to restrict the number of tuples,
    thus, the upper bounds is rarely present in practice

MORE SQL
RELATIONAL ALGEBRA: SIZE OF RESULT

- The answer to this question depends on involved operations:

- Cartesian Product
  - Cardinality is the product of the cardinalities of participating relations
  - Thus, the cartesian product is always an ”expensive” operation

- Join
  - Upper bounds: Product of cardinalities of participating relations
  - Lower bounds: 0
  - Thus, the join operation may be an ”expensive” operation

MORE SQL
RELATIONAL ALGEBRA: OPERATIONS

BONUS

- Minimal set of operations
  - It‘s sufficient if a language provides the operations ρ, σ, π, $\cup$, $-$, $\times$
    - The language is then ”relational complete”, meaning ”everything” is requestable
    - The operations are also independent, therefore none of it are dispensable
  - Other operations are representable by these operations:
    - Example: $\mathrm{R} \cap S \Leftrightarrow (R \cup S) - ((R - S) \cup (S - R))$
  - Important for the implementation of a DBMS and for the optimization of queries

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: OPERATIONS

BONUS

- Selection (σ (sigma))
- Projection (π (pi))
- Renaming (ρ (rho))
- Union (∪)
- Set Difference (or Except, Minus, −)
- Cartesian Product (x)
- All other operations can be built from these!

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: OPERATIONS

Operation	Purpose	Notation
Selection	Selects all tuples that satisfy the selection condition from a relation R	$\sigma_{<selection\ condition>}(R)$
Projection	Produces a new relation with only some of the attributes of R, and removes duplicate tuples	$\pi_{<attribute\ list>}(R)$
Renaming	Column in the result relation gets new name	$\rho_{new\ name \leftarrow attribute\ name}(R)$
Join	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition	$R_1 \bowtie_{<join\ condition>} R_2$
Equijoin	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons	$R_1 *_{<join\ condition>} R_2$

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: OPERATIONS

Operation	Purpose	Notation
Union	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible	$R_1 \cup R_2$
Intersection	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible	$R_1 \cap R_2$
Set Difference	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible	$R_1 - R_2$
Cartesian Product	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2	$R_1 \times R_2$

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL CALCULUS

- The relational algebra constructs the query result by applying operations and an order (project on X, select the Y and combine that with R_2,. . .)
- In contrast, the relational calculus are using a descriptive approach
- Calculus are logic-based approaches like the predicate logic
- Therefore, sets are characterized that correspond with the query result
- Calculus has variables, constants, comparison operations, logical operations and quantifier

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL CALCULUS

- Two types of relational calculus
  - Tuple relational calculus: variables declare tuples (are bounded to them)
  - Domain relational calculus: variables declare domain elements (thus, values range of attributes)
- Expressions in the calculus are called formula
- A result tuple is more or less an assignment of constants to variables, so that the formula is evaluated as TRUE

MORE SQL
RELATIONAL CALCULUS

Source: https://www.youtube.com/watch?v=ekF4qQBsk18

MORE SQL
RELATIONAL CALCULUS

- Example:
  - In mathematics:
    - $\{x^2 \mid x \in N \wedge x^3 < 1000 \wedge x^3 > 0\}$
    - This defines the set of all square numbers that cube number is between 0 and 1000
  - Relational calculus:
    - $A = \{x \mid Person(x, y) \wedge y = 'Jones'\} = \{2\}$
    - By usage of complex expressions (formula), the calculus has the same expressiveness as the relational algebra

Person

PID	Name
1	Kohler
2	Jones
3	Ford
4	Jones

MORE SQL
RELATIONAL CALCULUS

- Query languages for relational database schemas are mathematical substantiated
- The mathematical basis are the relational algebra and the relational calculus
- The relational algebra defines few operations, with that every request is expressible: Projection, Selection, Join, Renaming, Union, Set difference
- The relational calculus characterizes sets, which corresponds with the query result
- The relational calculus is descriptive, because it doesn't have to define an order of operations that construct the result
- Relational algebra and relational calculus have the same expressiveness

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

ORGANIZATION
OUR JOURNEY IN THIS SEMESTER

- Integrity, Trigger & Security
- Database Applications
- Transactions
- **Subqueries & Views**
- More SQL
- Notations & Guidelines
- Constraints
- Relationships
- Simple Entities and Attributes
- Basics

Source: Foto von Justin Kauffman auf Unsplash

HAW HAMBURG

SUBQUERIES AND VIEWS
SUBQUERIES

- **`SELECT`** returns relation: a (multi-)set
- Result of **`SELECT`** can be included in query
  - **`WHERE`** clause
    → also, for **`UPDATE`**, **`DELETE`**
  - **`HAVING`** clause
  - **`FROM`** clause
  - **`SELECT`** clause (in column list)
- So, we have two (or more) **`SELECT`**s:
  - Outer **`SELECT`**
  - Nested (or inner) **`SELECT`**: *subquery*

SUBQUERIES AND VIEWS
SUBQUERIES

Example from before:

```
INSERT INTO  Underpaid ( lname , fname )
       SELECT lname , fname
       FROM Employee
       WHERE salary < 1000 ;
```

→ **WHERE** clause belongs to **SELECT**

SUBQUERIES AND VIEWS
SUBQUERIES - `WHERE .. IN`

- In general, the nested query will return a **table** (relation), which is a set or multiset of tuples

- Check if value is a member of set
  → ... `WHERE a IN (1,4,9)`

- Set can be result of query
  → … `WHERE a IN (SELECT x FROM y)`

SUBQUERIES AND VIEWS
SUBQUERIES - WHERE .. IN

Example:

What does this statement mean???

```
SELECT DISTINCT Essn
FROM   WORKS_ON
WHERE  (Pno)  IN    (SELECT       Pno
                     FROM         WORKS_ON
                     WHERE        Essn='123456789');
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - `WHERE .. IN`

Example:

```
SELECT title
FROM   books
WHERE  isbn IN      (SELECT isbn
                     FROM recommended_books)
```

→ The subqueries are independent of each other, because they do not access the same tables

SUBQUERIES AND VIEWS
SUBQUERIES - WHERE .. IN

Example:

```
SELECT lastname
FROM   person
WHERE  1.0 IN (SELECT grade
               FROM exam
               WHERE person.no = exam.no)
```

→ In this example both queries depend on each other,
because the second query references a part of the first relation ("person").

SUBQUERIES AND VIEWS
SUBQUERIES - `WHERE .. IN`

Example:

```
SELECT E.Fname, E.Lname
FROM   EMPLOYEE AS E
WHERE  E.Ssn IN (   SELECT Essn
                    FROM   DEPENDENT AS D
                    WHERE  E.Sex = D.Sex );
```

What does this statement mean???

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - `WHERE .. IN`

- The operator **`IN`** can also be used for explicit enumerations

- Example:

```
… WHERE value IN ( value1 , value2 , value3 , … )
… WHERE colour IN ( 'red' , 'blue’ )
```

```
SELECT     DISTINCT Essn
FROM       WORKS_ON
WHERE      Pno IN (1, 2, 3);
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES

Example from last chapter (about set operations):

```
(SELECT     DISTINCT Pnumber
FROM        PROJECT, DEPARTMENT, EMPLOYEE
WHERE       Dnum=Dno AND Mgr_ssn=Ssn AND Lname="Wong" )
UNION
(SELECT     DISTINCT Pnumber
FROM        PROJECT, WORKS_ON, EMPLOYEE
WHERE       Pnumber= Pno AND Essn= Ssn AND Lname="Wong" );
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES

Example from last chapter (about set operations):

Alternative statement using subqueries

```
SELECT   DISTINCT Pnumber
FROM     PROJECT
WHERE    Pnumber IN
         (SELECT Pnumber
         FROM            PROJECT, DEPARTMENT, EMPLOYEE
         WHERE           Dnum=Dnumber
                         AND Mgr_ssn=Ssn
                         AND Lname="Wong")
         OR
         Pnumber IN
         (SELECT Pno
         FROM            WORKS_ON, EMPLOYEE
         WHERE           Essn=Ssn
                         AND Lname="Wong" );
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - `WHERE .. IN`

- Special case: Nested query returns only one value
  → In such cases, it is permissible to use = instead of **`IN`** for the comparison operator

- Example:

```
SELECT *
FROM y
WHERE x = ( SELECT MAX(x) FROM y ) ;
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - `WHERE .. IN`

- In general, a query written with nested **`SELECT-FROM-WHERE`** blocks and using the **`=`** or **`IN`** comparison operators can *always* be expressed as a single block query

- Example from before:

```
SELECT E.Fname, E.Lname
FROM   EMPLOYEE AS E
WHERE  E.Ssn IN (   SELECT Essn
                    FROM   DEPENDENT AS D
                    WHERE  E.Sex=D.Sex );
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - WHERE .. IN

Example from last slide:

Alternative statement without a subquery

```
SELECT E.Fname, E.Lname
FROM   EMPLOYEE AS E, DEPENDENT AS D
WHERE  E.Ssn=D.Essn
       AND E.Sex=D.Sex;
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - OTHER COMPARISON OPERATORS

- **=ANY** (**=SOME**)
  - operator returns $TRUE$ if the value v is equal to *some value* in the set V
  - is equivalent to **IN**
  - Other operations can be combined with **ANY**, e.g., `>`, `>=`, `<`, `<=`, and `<>`

  - Example: Persons who have borrowed a book:

```
SELECT name
FROM   Person
WHERE  PNr = ANY (SELECT PNr FROM book ) ;
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - OTHER COMPARISON OPERATORS

- **ALL**
  - Comparison operations can be combined with **ALL**, e.g., >, >=, <, <=, and <>
  - Example: Employees earning more money as the employees of department 5

```
SELECT Lname, Fname
FROM   EMPLOYEE
WHERE  Salary > ALL
       (SELECT       Salary
       FROM          EMPLOYEE
       WHERE         Dno=5) ;
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - OTHER COMPARISON OPERATORS

BONUS

- Often several queries give the same result
  → but might have difference in the performance!

```
SELECT * FROM book WHERE price <= ALL (SELECT price FROM book ) ;
SELECT * FROM book WHERE price = (SELECT MIN(price) FROM book ) ;

SELECT * FROM book WHERE price >= ALL (SELECT price FROM book ) ;
SELECT * FROM book WHERE price = (SELECT MAX(price) FROM book ) ;

SELECT * FROM book WHERE price > ANY (SELECT price FROM book ) ;
SELECT * FROM book WHERE price > (SELECT MIN(price) FROM book ) ;
```

SUBQUERIES AND VIEWS
SUBQUERIES - OTHER COMPARISON OPERATORS

- Often several queries give the same result
  → but might have difference in the performance!

- Strategy depends on DBMS, probably equivalent if no index on price, otherwise, the second version will be (much) faster

SUBQUERIES AND VIEWS
SUBQUERIES - CORRELATED NESTED QUERIES

- *Uncorrelated*
  - Outer and nested query are independent

  → Nested query must be computed only once

- *Correlated*
  - Nested query depends on columns of outer query
  - Result of a correlated nested query is different for each tuple of the relation(s) outer query
    → *A nested query is evaluated once for each tuple (or combination of tuples) in the outer query*
  - Performance?

SUBQUERIES AND VIEWS
SUBQUERIES - CORRELATED NESTED QUERIES

Source: https://www.youtube.com/watch?v=0ETfzIAQqBQ

SUBQUERIES AND VIEWS
SUBQUERIES - CORRELATED NESTED QUERIES

BONUS

- Example: Who has borrowd books for 9.99$?

```
SELECT name
FROM   pers
WHERE  9.99 IN
            (SELECT     price
             FROM       book
             WHERE      pers.PNr = book.PNr ) ;

SELECT name
FROM   pers, book
WHERE  pers.PNr = book.PNr
AND    book.price = 9.99;
```

SUBQUERIES AND VIEWS
SUBQUERIES - CORRELATED NESTED QUERIES

- Example from before:

```
SELECT E.Fname, E.Lname
FROM   EMPLOYEE AS E
WHERE  E.Ssn IN (   SELECT Essn
                    FROM   DEPENDENT AS D
                    WHERE  E.Sex=D.Sex );
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - AMBIGUITY OF ATTRIBUTES

- In general, we can have several levels of nested queries
  → possible ambiguity among attribute names if attributes of the same name exist:
  - one in a relation in the **`FROM`** clause of the *outer query,* and
  - another in a relation in the **`FROM`** clause of the *nested query*
- The rule is that a reference to an *unqualified attribute* refers to the relation declared in the **innermost nested query**

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - AMBIGUITY OF ATTRIBUTES

- Example: Retrieve the name of each employee who has a dependent with the same sex as the employee

```
SELECT E.Fname, E.Lname
FROM   EMPLOYEE AS E
WHERE  E.Ssn IN (   SELECT Essn
                    FROM   DEPENDENT AS D
                    WHERE  E.Sex=D.Sex );
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES - AMBIGUITY OF ATTRIBUTES

It is generally advisable to create
tuple variables (aliases)
for *all the tables referenced*
in an SQL query
to avoid potential errors and ambiguities!!!

SUBQUERIES AND VIEWS
SUBQUERIES - EXISTS

- The operator **EXISTS** (**NOT EXISTS**) provides the possibility to check if the result of another query is empty ($FALSE$) or not ($TRUE$)

- Example:

```
SELECT isbn FROM book WHERE EXISTS
   (SELECT * FROM borrowed WHERE book.libraryno = borrowed.libraryno )
```

- This example provides as result a set of all borrowed books
- Typically, the usage is
  `... EXISTS (SELECT *...`
  → so that the DBMS may decide, which column should be examined

SUBQUERIES AND VIEWS
SUBQUERIES - EXISTS

Example from before:

```
SELECT E.Fname, E.Lname
FROM   EMPLOYEE AS E
WHERE  E.Ssn IN (    SELECT Essn
                     FROM   DEPENDENT AS D
                     WHERE  E.Sex=D.Sex );
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - EXISTS

Example from before:
Alternative SQL-statement

```
SELECT E.Fname, E.Lname
FROM   EMPLOYEE AS E
WHERE  EXISTS
             (SELECT *
             FROM   DEPENDENT AS D
             WHERE  E.Ssn = D.Essn
                    AND E.Sex = D.Sex);
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - EXISTS

- **EXISTS** and **NOT EXISTS** are typically used in conjunction with a correlated nested query

- Example: Retrieve the names of employees who have no dependents

```
SELECT Fname, Lname
FROM   EMPLOYEE
WHERE  NOT EXISTS ( SELECT *
                    FROM   DEPENDENT
                    WHERE  Ssn=Essn );
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES - EXISTS

Example: List the names of managers who have at least one dependent

```
SELECT Fname, Lname
FROM   EMPLOYEE
WHERE  EXISTS (      SELECT *
                     FROM   DEPENDENT
                     WHERE  Ssn=Essn )
       AND
       EXISTS (      SELECT *
                     FROM   DEPARTMENT
                     WHERE  Ssn=Mgr_ssn );
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
SUBQUERIES – IN FROM

- **SELECT** returns a new relation
- ... so, we can select values from it
- Necessary: give a name to the relation

- Example: Alias's name

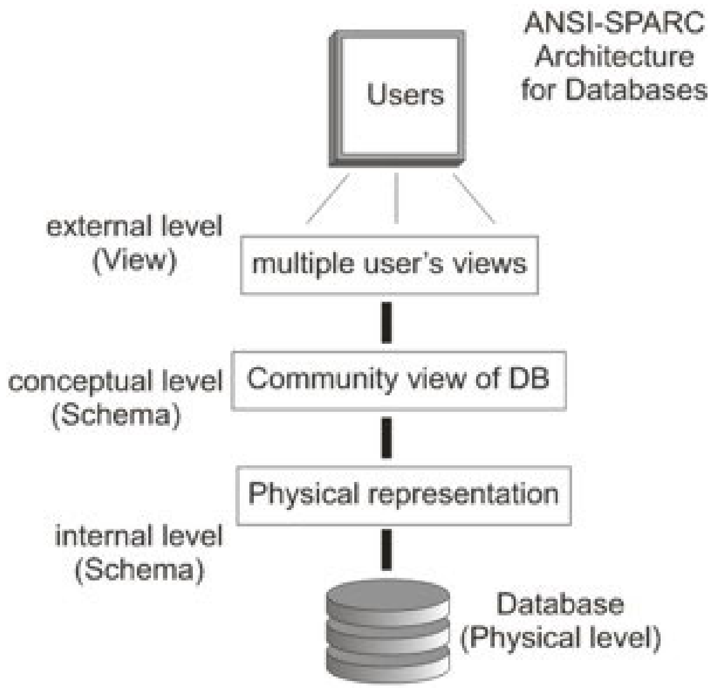
```
SELECT tab_a.x , newtab_b.y
FROM   tab_a , (SELECT v1, v2 FROM tab_b) AS newtab_b ;
```

SUBQUERIES AND VIEWS
VIEWS – RECAP: THE ANSI-SPARC ARCHITECTURE

External Schema

Conceptual Schema

Internal Schema



Source: www.wikipedia.org

SUBQUERIES AND VIEWS
VIEWS – BASICS

- User or application specific views on data
- Only relevant portions of the data

- A *view* in SQL terminology is a single table that is derived from other tables
  → Other tables can be *base tables* or previously defined views

- A view is considered to be a *virtual table*
  → In contrast to base tables
  → Limits the possible update operations
  → No limitations on querying a view

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
VIEWS – USE CASES

- Hide some information
  - Example: Salary not viewable for colleagues
  - Can see only employees of same department?

- Convert data for different users
  - Example: Price in $, EUR, ...

- Backward Compatibility
  - Example: Add some columns, but old applications do "`SELECT *`"

- Simplication: Hide away complex queries
  - Example: Data Dictionary Views (all tables)

SUBQUERIES AND VIEWS
VIEWS – CREATE

- Syntax:

```
CREATE [OR REPLACE] VIEW <vname> AS <query> ;
```

- `<query>` is an arbitrary **SELECT** statement

SUBQUERIES AND VIEWS
VIEWS – CREATE

- Example:

```
CREATE VIEW vPerson AS
   SELECT Name , Id , BirthDate FROM person ;
```

vPerson	Name	Id	BirthDate

Can rename columns in view:

```
CREATE VIEW vPerson ( lname , pnr , bd ) AS
   SELECT Name , Id , BirthDate FROM person
```

vPerson	lname	pnr	bd

SUBQUERIES AND VIEWS
VIEWS – CREATE

Example:

```
CREATE VIEW v_WORKS_ON1 AS
       SELECT Fname, Lname, Pname, Hours
       FROM   EMPLOYEE, PROJECT, WORKS_ON
       WHERE  Ssn=Essn AND Pno=Pnumber;
```

WORKS_ON1	Fname	Lname	Pname	Hours

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
VIEWS – CREATE

Example:

```
CREATE VIEW v_DEPT_INFO
             (Dept_name, No_of_emps, Total_sal) AS
             SELECT Dname, COUNT(*), SUM(Salary)
             FROM   DEPARTMENT, EMPLOYEE
             WHERE  Dnumber=Dno
             GROUP BY Dname;
```

DEPT_INFO	Dept_name	No_of_emps	Total_sal

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
VIEWS – QUERY

- A view is supposed to be *always up-to-date*
  → If we modify the tuples in the base tables on which the view is defined, the view must automatically reflect these changes
  → View realized at the time when we *specify a query* on the view

- Example:

```
SELECT Fname, Lname
FROM   v_WORKS_ON1
WHERE  Pname="ProductX";
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
VIEWS – DROP

- Views can be dropped

- Example:

```
DROP VIEW v_WORKS_ON1;
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
VIEWS – IMPLEMENTATION

- Two strategies:
  1. *Query Modification*
     → Transforming the view query into a query on the underlying base tables

```
CREATE VIEW v_WORKS_ON1 AS
       SELECT Fname, Lname, Pname, Hours
       FROM   EMPLOYEE, PROJECT, WORKS_ON
       WHERE  Ssn=Essn AND Pno=Pnumber;
```

Example:

```
SELECT Fname, Lname
FROM   EMPLOYEE, PROJECT, WORKS_ON
WHERE  Ssn=Essn
       AND Pno=Pnumber
       AND Pname= "ProductX";
```

```
SELECT Fname, Lname
FROM   v_WORKS_ON1
WHERE  Pname="ProductX";
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
VIEWS – IMPLEMENTATION

not supported by mySQL!

- Two strategies:
  2. *View Materialization*
     → Physically creating a temporary view
     → Incremental update of materialized view
     → If the view is not queried for a certain period of time, the system may then automatically remove the physical table and recompute it from scratch on new queries

```
CREATE MATERIALIZED VIEW v_WORKS_ON1 AS
       SELECT Fname, Lname, Pname, Hours
       FROM   EMPLOYEE, PROJECT, WORKS_ON
       WHERE  Ssn=Essn AND Pno=Pnumber;
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
VIEWS – MATERIALIZED VIEWS

not supported by mySQL!

- Syntax:
  `CREATE MATERIALIZED VIEW <name> AS SELECT …`

- Traditional views
  - Select is performed when needed
  - Performance penalty

- Materialized view
  - Store select statement and selected data
  - Problems
    - Store data twice
    - When to update selected data?
  - Rules for updating: event vs. time triggered
  - Selected data can be updated
    - manually
    - on a regular basis (every night)
    - event triggered (update to base table)