

Data Wrangling and Data Analysis

Introduction

Relational Model, Schemas, Data Extraction in SQL and Python

Hakim Qahtan

Department of Information and Computing Sciences

Utrecht University



About the INFOMDWR Team

Hakim Qahtan



Faculty of Science

Information and Computing
Sciences Department

Daniel Oberski



Erik-Jan van Kesteren



Ayoub Bagheri



Faculty of Social and Behavioural Sciences
(Tentative)



About Hakim Qahtan



Cairo University
Egypt
BSc in CS



Saudi Arabia
MSc in ICS



Qatar
Postdoc



Yemen
TA for 3 Years



Saudi Arabia
PhD in CS



Utrecht University

Netherlands
Assistant Professor

<https://www.linkedin.com/in/hakimqahtan/>



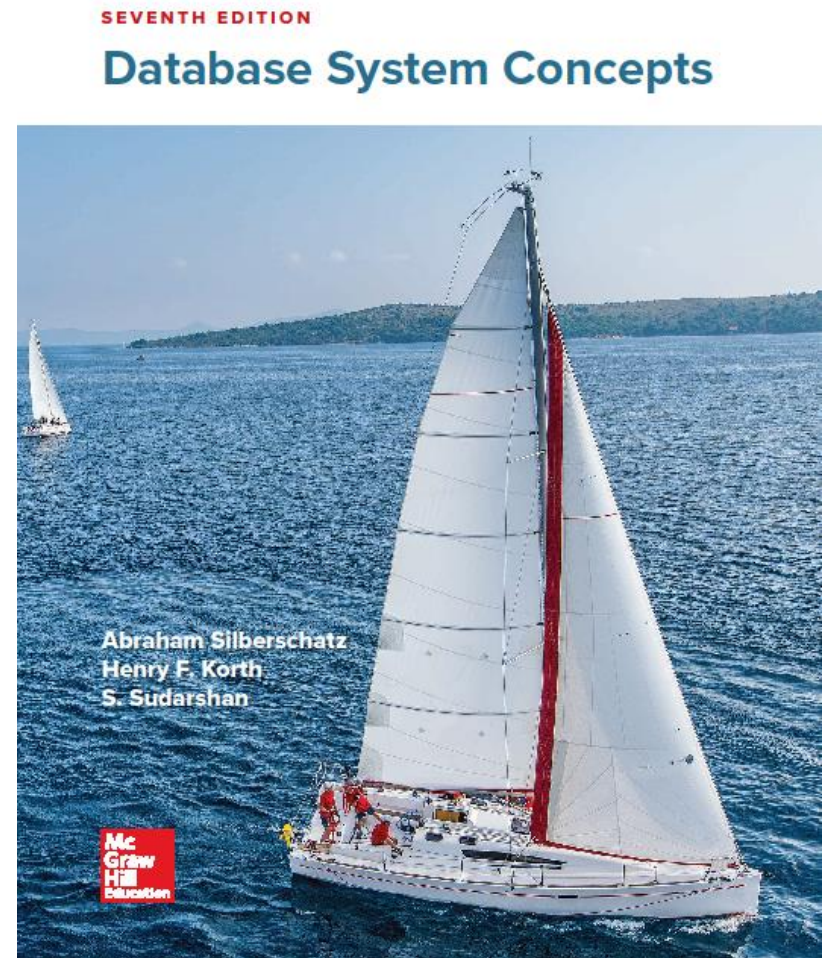
Content of the Data Wrangling and Data Analysis

- Main topics in this course include:
 - Data Extraction – Relational databases and SQL + Data extraction in Python
 - Data Integration and entity linkage
 - Data preprocessing
 - Consistency assurance + Cleaning + Transformation + Normalization
 - Data reduction + Data Imputation
 - Data Visualization
 - Data modeling
 - Classification + Clustering
 - Text Mining
 - Dashboards for Data Visualization



Reading Material for this Lecture

- Chapters 1-3.3 of the Database System Concepts Book



Passing the Course

- Weekly exams: every students should attend 2 weekly exams.
- Exams will be closed book on Remindo. You can take the exam anywhere (we rely on your professionalism and integrity).
- The exam on Thursday will be the same for all students.
- Students who didn't pass the exam on Thursday will be given a chance for a retake exam on Friday
- Students who pass the exam on Thursday will be given an honor exam to get better grade on Friday.



Course Objectives

Know, explain, and apply data retrieval from existing relational and nonrelational databases, including text, using queries build from primitives such as select, subset, and join both directly in, e.g., SQL and Python. (This Week)

Know, explain, and apply common data clean-up procedures, including missing data and the appropriate imputation methods and feature selection.

Know, explain, and apply methodology to properly set-up data analysis experiments, such as train, validate, and test and the bias/variance trade-off.

Know, explain, and apply supervised machine learning algorithms, both for classification and regression purposes as well as their related quality measures, such as AUC and Brier scores.

Know, explain, and apply non-supervised learning algorithms, such as clustering and (other) matrix factorization techniques that may or may not result in lower-dimensional data representations.

Be able to choose between the different techniques learned in the course and be able to explain why the chosen technique fits both the data and the research question best.

Note: We will not cover extracting data from text documents this week. It will be in week 10



Databases: Background and Terminology



Databases

- Used to be about boring stuff: employee records, bank records, etc.
- Today, the field covers all the largest sources of data, with many new ideas.
 - Web search.
 - Data mining.
 - Scientific and medical databases.
 - Graph databases.
 - Integrating information.



Databases – More

- You may not notice it, but databases are behind almost everything you do on the Web.
 - Google searches.
 - Queries at Amazon, eBay, etc.
- DBMS contains information about a particular enterprise
 - Collection of interrelated data
 - Set of programs to access the data
 - An environment that is both *convenient* and *efficient* to use



Database – Example

- University Database Example:
 - Add new students, instructors, and courses
 - Register students for courses, and generate class rosters
 - Assign grades to students, compute grade point averages (GPA) and generate transcripts



File system as data storage

- Drawbacks of using file systems to store data include:
 - Data redundancy and inconsistency
 - Multiple file formats, duplication of information in different files
 - Difficulty in accessing data
 - Need to write a new program to carry out each new task
 - Data isolation
 - Multiple files and formats
 - Integrity problems
 - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
 - Hard to add new constraints or change existing ones



File system as data storage (Cont.)

- Drawbacks of using file systems to store data include:
 - Atomicity of updates
 - Failures may leave database in an inconsistent state with partial updates carried out
 - Example: Transfer of funds from one account to another should either complete or not happen at all
 - Concurrent access by multiple users
 - Concurrent access needed for performance
 - Uncontrolled concurrent accesses can lead to inconsistencies
 - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
 - Security problems
 - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems



Database levels of abstraction

- **Physical level:** describes how a record (e.g., instructor) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

```
type instructor = record
```

```
    ID : string;
```

```
    name : string;
```

```
    dept_name : string;
```

```
    salary : real;
```

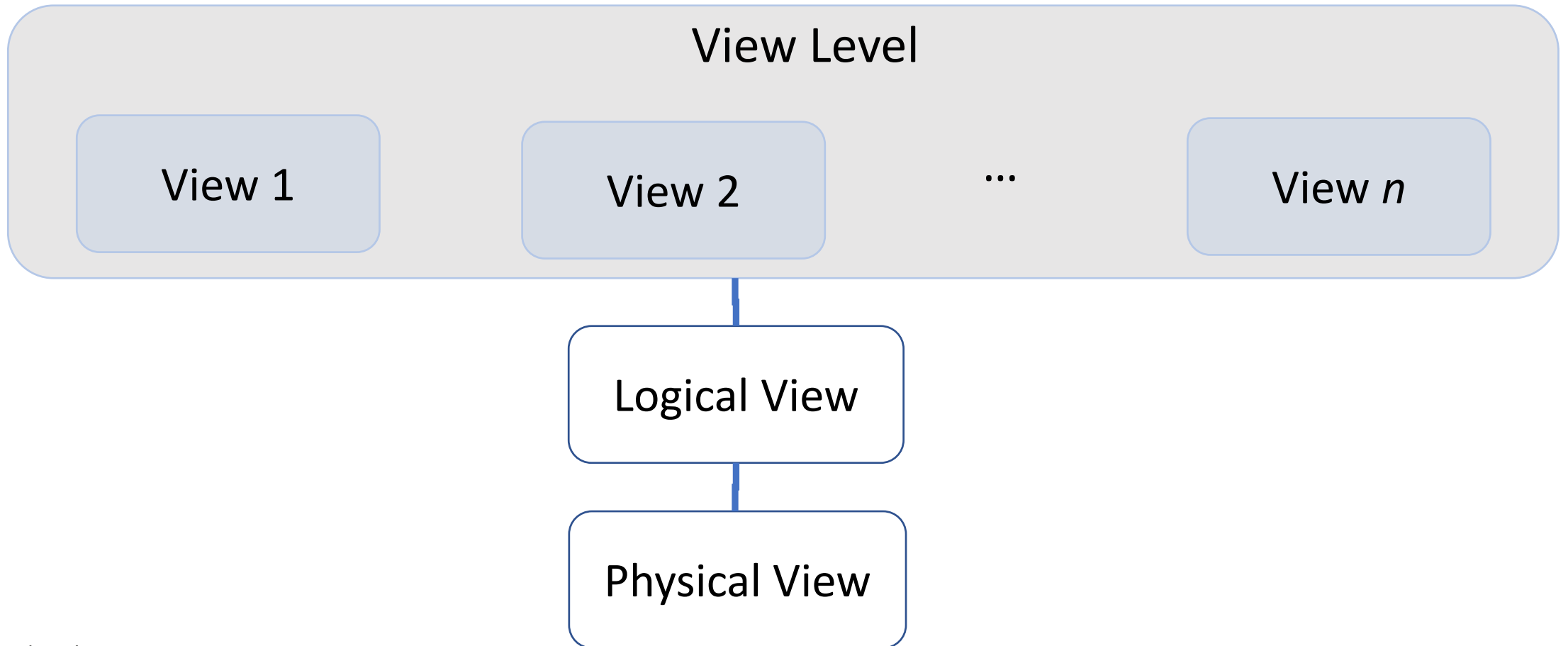
```
end;
```

- **View level:** application programs which hide details of data types. Views can also hide information for security/privacy purposes (such as an employee's salary).



Database levels of abstraction

- Architecture of a database system



Data Models

- **Data model:** mathematical representation of the data.
 - Relational model = tables
 - Semistructured model = trees/graphs
- Data models can be viewed as a collection of tools describing:
 - Data
 - Data relationships
 - Data semantics
 - Data constraints



Data Models

- Data models are said to contain three main components:
 - Structures
 - Rows and columns?
 - Nodes and edges?
 - Key-value pairs?
 - Constraints
 - All rows must have the same number of columns
 - Column A3 contains numerical values
 - The age must be positive
 - Operations
 - Return the values of record x
 - Find the rows where the column "lastname" is "Jordan"

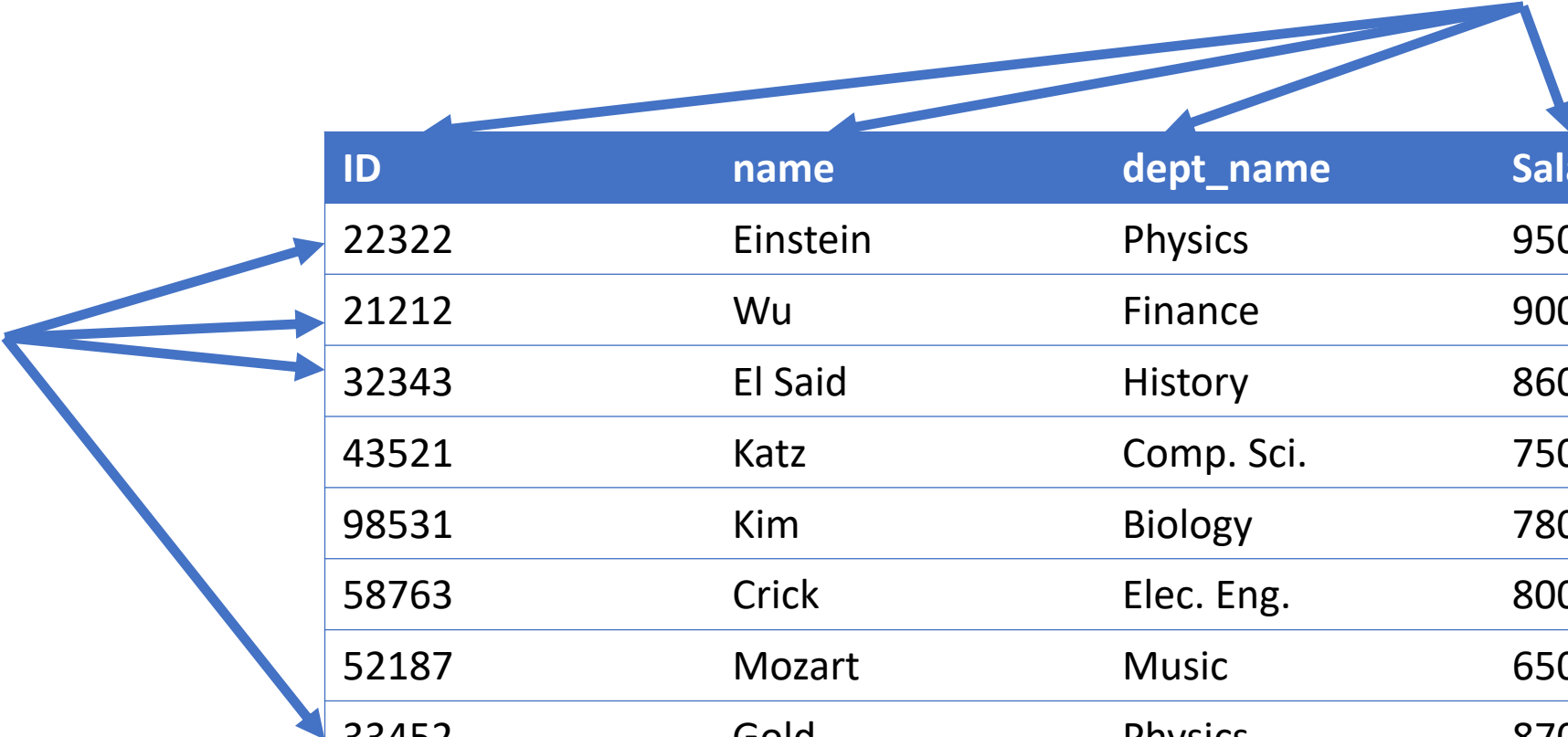


The Relational Model

- Data stored in different tables
- Example of tabular data

Columns (Attributes or Features)

Rows
(Records or
tuples)



ID	name	dept_name	Salary
22322	Einstein	Physics	95000
21212	Wu	Finance	90000
32343	El Said	History	86000
43521	Katz	Comp. Sci.	75000
98531	Kim	Biology	78000
58763	Crick	Elec. Eng.	80000
52187	Mozart	Music	65000
33452	Gold	Physics	87000



The Relational Model

ID	name	dept_name	Salary
22322	Einstein	Physics	95000
21212	Wu	Finance	90000
32343	El Said	History	86000
43521	Katz	Comp. Sci.	75000
98531	Kim	Biology	78000
58763	Crick	Elec. Eng.	80000
52187	Mozart	Music	65000
33452	Gold	Physics	87000

The *instructor* table

dept_name	building	Budget
Comp. Sci.	Turing	1000000
Physics	Watson	800000
Chemistry	Ibn-Hayyan	850000
Biology	AlRazi	780000
Elec. Eng.	Taylor	690000
Music	Packard	400000
Finance	Painter	1200000
History	Painter	500000

The *department* table



Schema

- Relation schema = relation name and attribute list.
 - Optionally: types of attributes.
 - Example: student(ID, name, dept_name, tot_cred) or student(ID: numeric, name: string, dept_name: string, tot_cred: numeric)
- Database = collection of relations.
- Database schema = set of all relation schemas in the database.



Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value **null** is a member of every domain. Indicated that the value is "unknown"
- The null value causes complications in the definition of many operations



Relation Schema and Instance

- Let A_1, A_2, \dots, A_n be the names of attributes
- $R = (A_1, A_2, \dots, A_n)$ is a relation schema
- Given sets D_1, D_2, \dots, D_n from which A_1, A_2, \dots, A_n take their values, we call $r \subseteq D_1 \times D_2 \times \dots \times D_n$ a relation instance or simply a relation
- An element t in r is a tuple and represented as a row in the table



Keys

- Let $K \subseteq R$
- K is a superkey of R if values for K are sufficient to identify a unique tuple of each possible relation $r(R)$
 - Example: $\{ID\}$ and $\{ID, name\}$ are both superkeys of instructor.
- Superkey K is a candidate key if K is minimal
Example: $\{ID\}$ is a candidate key for Instructor
- One of the candidate keys is selected to be the **primary key**.
 - Which one?
- **Foreign key**: Value in one relation must appear in another
 - **Referencing** relation
 - **Referenced** relation
 - Example – dept_name in instructor is a foreign key in the instructor relation referencing department



Data Definition Language (DDL)

- Specification notation for defining the database schema
- DDL is used to create and modify database objects such as tables, indexes and users

Example: create table instructor (
 ID **char**(5),
 name **varchar**(20),
 dept_name **varchar**(20),
 salary **real**);



Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
 - DML also known as query language
 - DML is used for adding (inserting), deleting, and modifying (updating) the records in a database
- Most common DML is the one in the structured query language (SQL)
- Other forms uses kind of ODBC/JDBC (Open/Java DataBase Connectivity) interfaces to connect and send queries to the DBMS.



Structured Query Language (SQL)

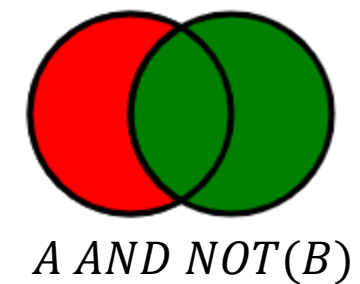
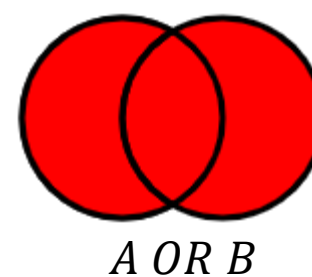
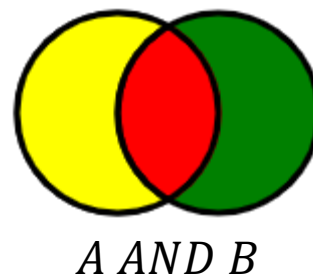
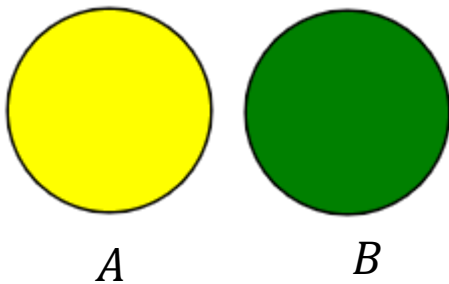
- High level language to manipulate relational data
 - Say “what to do” rather than “how to do it.”
 - Avoid a lot of data-manipulation details needed in procedural languages like C++ or Java.
- Database management system figures out “best” way to execute query.
 - Called “query optimization.”
- SQL has three categories based on the functionality involved
 - DDL
 - DML
 - DCL: used to grant and revoke authorization.



Boolean Operators

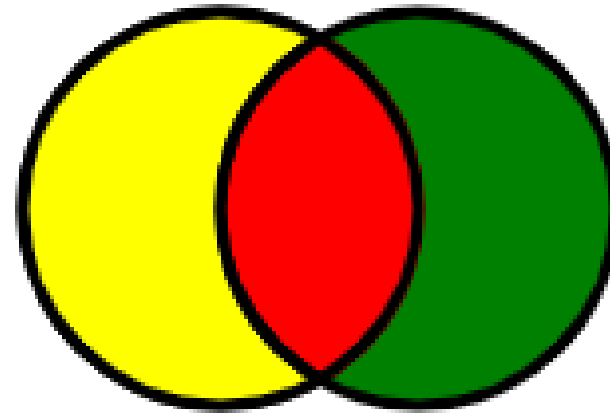
Boolean Operators

- Searching through a database or search engine can often be frustrating
- Boolean Operators create relationships between concepts and terms for better search results
- Most popular Boolean operators are **AND**, **OR** and **NOT**
 - The red areas represent the results of the operators



AND (\wedge)

A	B	$A \wedge B$
1	1	1
1	0	0
0	1	0
0	0	0

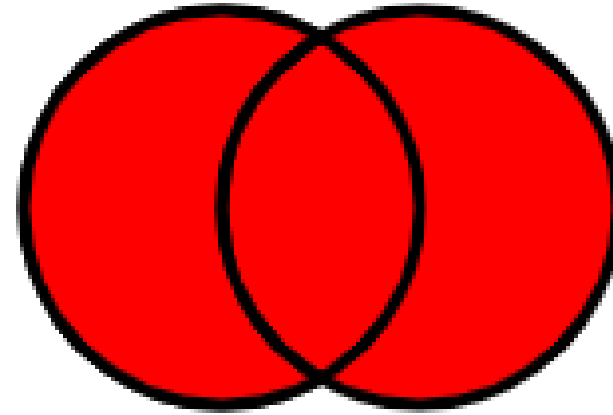


The red area – $A \text{ AND } B$

- Retrieves only records that satisfy both conditions
- Example:
name = “Taylor” AND dept_name = “Chem.”
Returns all instructors in the Chem. department whose name is Taylor

OR (\vee)

A	B	$A \vee B$
1	1	1
1	0	1
0	1	1
0	0	0

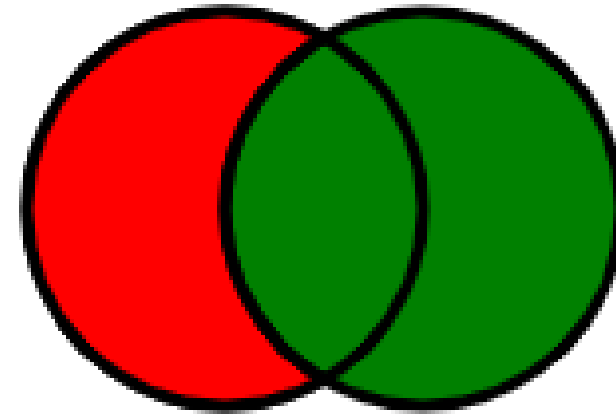


The red area – $A \text{ OR } B$

- Retrieves records that satisfy one of the conditions
- Example:
name = “Taylor” OR dept_name = “Chem.”
Returns all instructors with the name Taylor and all instructors of the Chem. department

NOT (\sim)

A	$\sim A$
1	0
0	1



The red area – $A \text{ AND } \text{NOT}(B)$

- Retrieves only records that satisfy the first condition and doesn't satisfy the second
- Example:
name = "Taylor" AND
NOT dept_name = "Chem."
Returns all instructors with the name Taylor who do not work in the Chem. department

Boolean Equivalence

- Equivalence of two Boolean operations can be easily proven using truth tables
- Equivalence of Boolean operations is useful for optimizing the Boolean queries
- Examples:

- $A - B \equiv A \wedge \sim B$ 

A	B	$\sim B$	$A \wedge \sim B$	$A - B$
1	1	0	0	0
1	0	1	1	1
0	1	0	0	0
0	0	1	0	0

 \equiv 

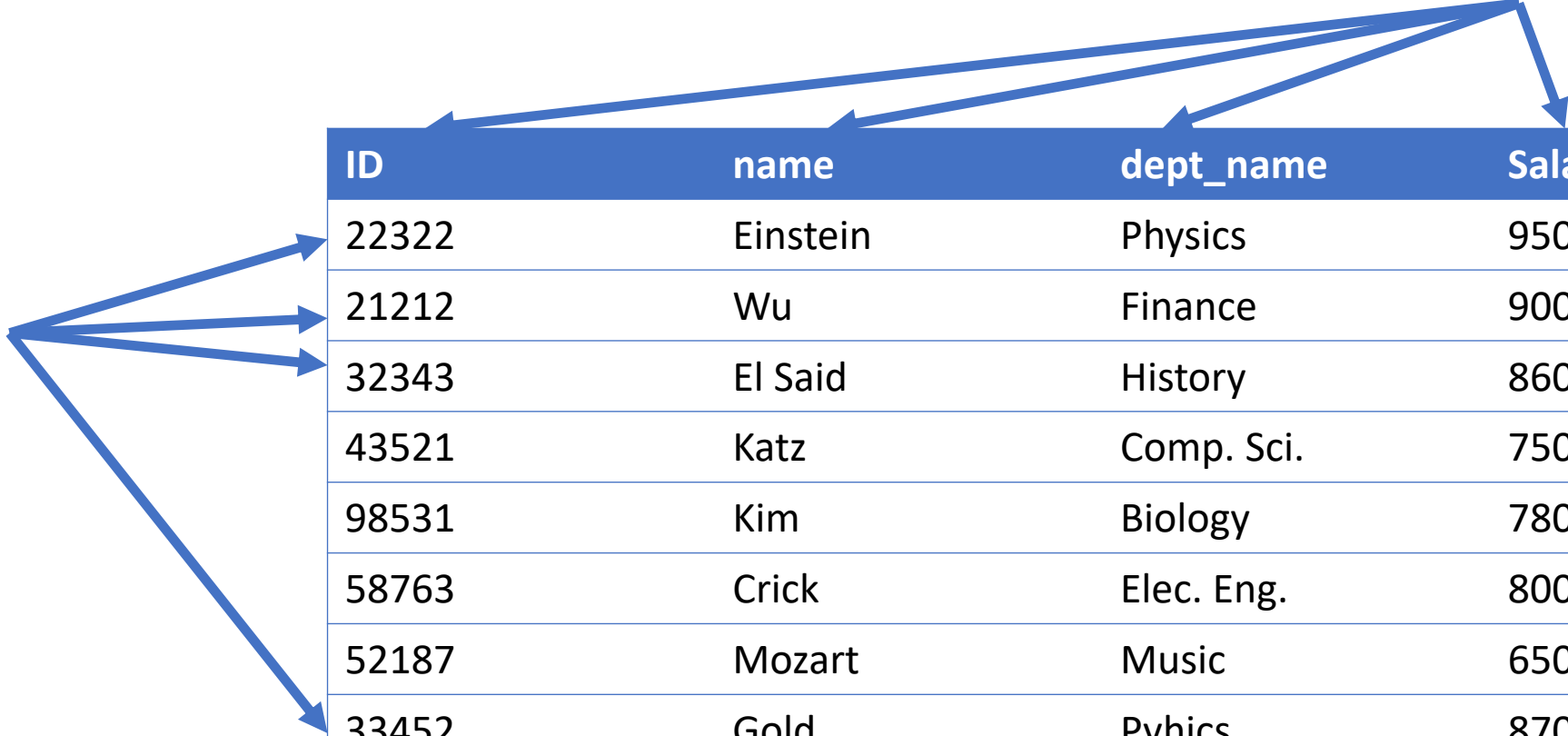
- $(A \wedge B) \wedge \sim B \equiv A \wedge (B \wedge \sim B) \equiv \text{false}$
 - $(A \wedge B) \vee (\sim B \wedge C) \vee (A \wedge C) \vee \sim (A \wedge C) \equiv (A \wedge C) \vee \sim (A \wedge C) \equiv \text{true}$

Relational Algebra

Example of a Relation

Columns (Attributes or Features)

Rows
(Records or
tuples)



ID	name	dept_name	Salary
22322	Einstein	Physics	95000
21212	Wu	Finance	90000
32343	El Said	History	86000
43521	Katz	Comp. Sci.	75000
98531	Kim	Biology	78000
58763	Crick	Elec. Eng.	80000
52187	Mozart	Music	65000
33452	Gold	Pyhics	87000



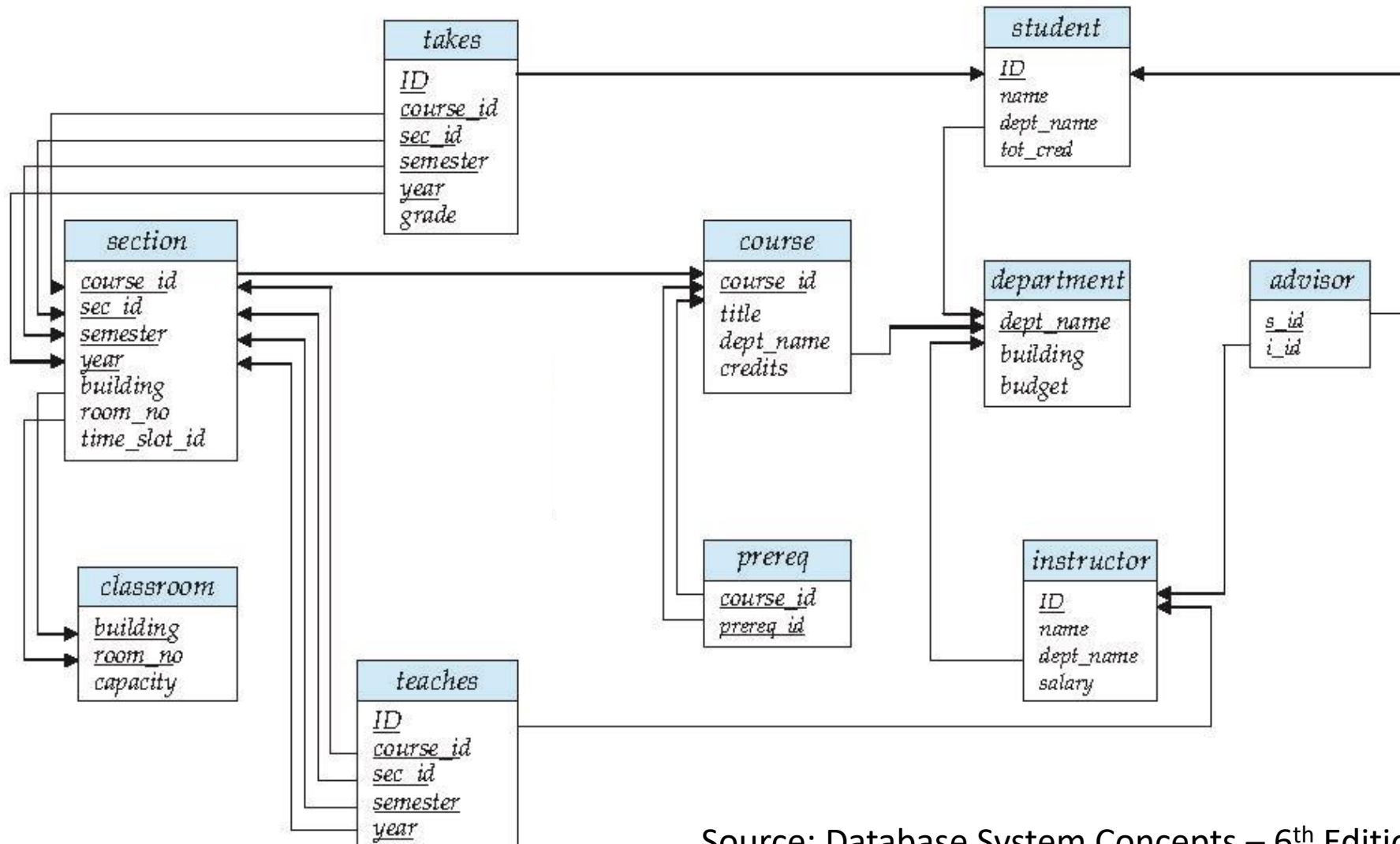
Relations are Unordered

- Order of tuples is not considered (tuples might be stored in an arbitrary order)
- The same is also true in the case of attributes
 - Example: putting “salary” before/after “dept_name” will have no effect on the relation (table)

ID	name	dept_name	salary
22322	Einstein	Physics	95000
21212	Wu	Finance	90000
32343	El Said	History	86000
43521	Katz	Comp. Sci.	75000
98531	Kim	Biology	78000
58763	Crick	Elec. Eng.	80000
52187	Mozart	Music	65000
33452	Gold	Physics	87000



Example: Schema Diagram for University Database



Source: Database System Concepts – 6th Edition



Relational Algebra Operators

- Union \cup , intersection \cap , difference $-$
- Selection σ
- Projection π
- Join \bowtie
- Duplicate elimination δ
- Grouping and aggregation γ
- Sorting τ

RA

Extended RA



Union

$r \cup s$

r

A	B
α	3
β	2

s

A	B
α	1
β	2
γ	3

$r \cup s$

A	B
α	1
β	2
γ	3
α	3

Difference

$$r - s$$

r

A	B
α	3
β	2

s

A	B
α	1
β	2
γ	3

r - s

A	B
α	3



Intersection

$$r \cap s$$

r

A	B
α	3
β	2

s

A	B
α	1
β	2
γ	3

$r \cap s$

A	B
β	2



Selection

r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\sigma_{(A=B) \wedge (D>5)} (r)$

A	B	C	D
α	α	1	7
β	β	23	10

Projection

r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\pi_{A,C}(r)$

A	C
α	1
α	5
β	12
β	23

Join

Cartesian Product

r

A	B
α	3
β	2

s

A	B
α	1
β	2
γ	3

r \times *s*

r.A	r.B	s.A	s.B
α	3	α	1
α	3	β	2
α	3	γ	3
β	2	α	1
β	2	β	2
β	2	γ	3

Natural Join

- Let r and s be relations on schemas R and S respectively.
Then, the “natural join” of relations R and S is a relation on schema $R \cup S$ obtained as follows:
 - Consider each pair of tuples t_r from r and t_s from s .
 - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where
 - t has the same value as t_r on r
 - t has the same value as t_s on s



Natural Join – Example

r

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

s

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

$r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

$$r \bowtie s \equiv \pi_{A,r.B,C,r.D,E}(\sigma_{r.B=s.B \wedge r.D=s.D}(r \times s))$$

Composition of Operations

r

A	B
α	3
β	2

s

A	B
α	1
β	2
γ	3

$r \times s$

r.A	r.B	s.A	s.B
α	3	α	1
α	3	β	2
α	3	γ	3
β	2	α	1
β	2	β	2
β	2	γ	3

$\sigma_{r.A=s.A}(r \times s)$

r.A	r.B	s.A	s.B
α	3	α	1
β	2	β	2

Summary of Relational Algebra Operators

Symbol (name)	Description
σ (Selection)	Return rows of the input relation that satisfy the predicate.
π (Projection)	Return specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
\times (Cartesian Product)	Return pairs of rows from the <i>two</i> input relations that have the same value on all attributes that have the same name.
\cup (Union)	Return the union of tuples from the <i>two</i> input relations.
$-$ (Difference)	Return the set difference of tuples from the <i>two</i> input relations.
\cap (Intersection)	Return the intersection of tuples from the <i>two</i> input relations.
\bowtie (Natural Join)	Return pairs of rows from the <i>two</i> input relations that have the same value on all attributes that have the same name.



Algebraic Optimization

- Let $y = \frac{(x*2)+((x*3)+0)}{1}$
- Algebraic laws:
 1. (+) identity : $a + 0 = a$
 2. (/) identity: $a/1 = a$
 3. (*) distributes: $a * (b + c) = (a * b) + (a * c)$
 4. (*) commutes: $a * b = b * a$

By applying rules 1, 3, 4, 2 we get:

$$y = (2 + 3) * x$$

Instead of having 5 operations, we have only 2 operations and no division

Same idea can be applied for Query Optimization



Data Extraction Using Structured Query Language (SQL)

SELECT-FROM-WHERE Statements

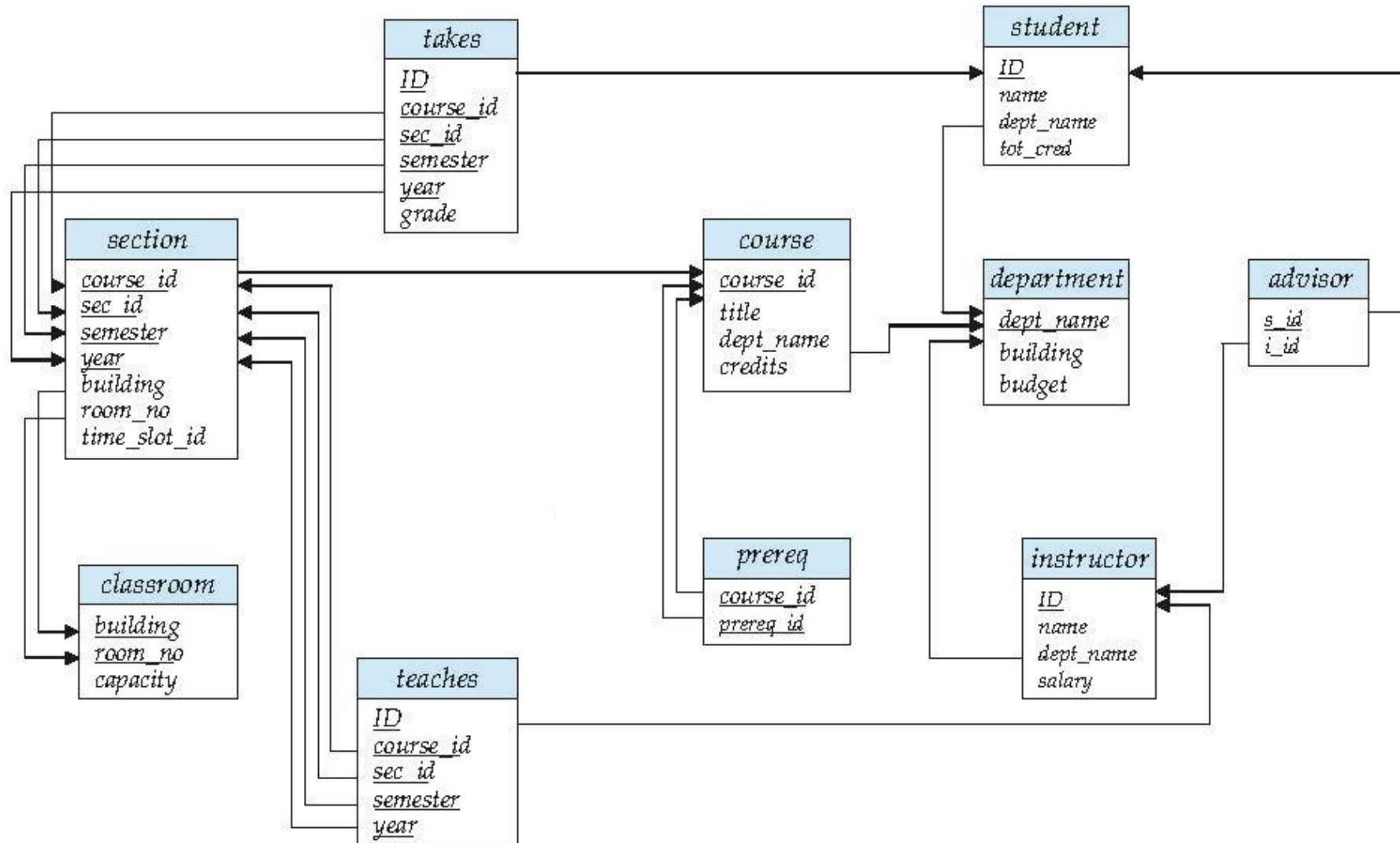
SELECT desired attributes

FROM one or more tables

WHERE condition about tuples of the tables



Remember Our Main Database



Example: The SELECT Clause

- Get the IDs, names and total credits of students who completed at least 24 credits

```
SELECT ID, name, tot_cred  
FROM student  
WHERE tot_cred > 24
```

- The answer is a relation with three attributes (ID, name, tot_cred)
- What if we need only the names of the students?
- Note that: the SQL operator names are case insensitive SELEECT \equiv Select \equiv select



The **SELECT** Clause

- Lists the desired attributes in the result of the query
 - Corresponds to the projection operator of the relational algebra
- SQL allows duplicates in relations as well as in query results
- To force the elimination of duplicates, use the keyword **DISTINCT** after select
- Example: find the department names of all instructors whose salary is strictly greater than 60000 without showing the department name more than once

```
SELECT DISTINCT dept_name
```

```
FROM instructor
```

```
WHERE salary > 60000
```



The **SELECT** Clause (Cont.)

- The keyword **ALL** specifies that duplicates should not be removed

```
SELECT ALL dept_name  
FROM instructor  
WHERE salary > 60000
```

- An asterisk in the **SELECT** clause denotes “all attributes”

```
SELECT * FROM instructor
```

will return all the records from table “instructor”



The **SELECT** Clause (Cont.)

- An attribute could be literal with no **FROM** clause

```
SELECT '542'
```

Results in a relation with one column and one row with value "542"

We can also give the column a name using

```
SELECT '542' AS V1
```

- An attribute could be a literal with no **FROM** clause

```
SELECT 'A' FROM instructor
```

will return a relation with one column and N rows (the number of tuples in the 'instructor' relation) where each row will contain the value "A"



The **SELECT** Clause (Cont.)

- **SELECT** clause can contain arithmetic expressions involving the operations *, +, -, and /.

```
SELECT ID, name, salary/12.0
```

```
FROM instructor
```

This query would return a relation with the same number of records as the ‘instructor’ relation and the (ID, name, salary/12.0) where the yearly salary is replaced by the monthly salary

We can rename the “salary/12.0” using the **AS** clause

```
SELECT ID, name, salary/12.0 as monthly_salary
```

```
FROM instructor
```



The WHERE Clause

- Specifies conditions that the result must satisfy
 - Corresponds to the selection predicate of the relational algebra
- To find all students enrolled in the 'Math.' department

```
SELECT ID, name FROM student
```

```
WHERE dept_name = 'Math.'
```

- Conditions can be also combined using logical operators (AND, OR, NOT)
 - Find all students in the 'Math.' department who completed a minimum of 24 credits

```
SELECT ID, name FROM student
```

```
WHERE dept_name = 'Math.' AND tot_cred >= 24
```

- Comparisons =, <>, <, >, <=, >=



The FROM Clause

- Lists the relations involved in the query
 - Corresponds to the Cartesian product operation of the relational algebra
- Find the Cartesian product '*instructor*' X '*teaches*'

SELECT * **FROM** instructor, teaches

- Generates every possible instructor – teaches pair, with all attributes from both relations.
- For common attributes (e.g., *ID*), the attributes in the resulting table are renamed using the relation name (e.g., *instructor.ID*)
- Cartesian product not very useful directly, but useful combined with where-clause condition (selection operation in relational algebra).



SELECT-FROM-WHERE Examples

```
SELECT name, course_id  
FROM instructor, teaches  
WHERE instructor.ID = teaches.ID
```

Returns the names of all instructors who have taught any courses and the course_id

```
SELECT name, course_id  
FROM instructor, teaches  
WHERE instructor.ID = teaches.ID AND instructor.dept_name = 'Art'
```

Returns the names of all instructors in the Art department who have taught any courses and the course_id



The Rename Operation

- SQL allows renaming relations and attributes using the **AS** clause:

old-name **AS** *new-name*

- Find the names of all instructors who have a higher salary than some instructor in 'Comp. Sci'.

```
SELECT DISTINCT T.name  
FROM instructor AS T, instructor AS S  
WHERE T.salary >= 75000 AND S.dept_name = 'Comp. Sci.'
```

Returns the names of instructors in the 'Comp. Sci.' department and whose salary is greater than or equal 75000

- Keyword **AS** is optional and may be omitted
instructor **AS** *T* \equiv *instructor T*



Renaming Example: Self-Join

- Sometimes, a query needs to use two copies of the same relation.
- Distinguish copies by renaming the relations.
- Example self-Join

```
SELECT T.name, S.name
```

```
FROM instructor T, instructor S
```

```
WHERE T.salary = S.salary AND T.name < S.name
```

- Returns the names of instructors who has the same salary
 - Do not produce pairs like (Miller, Miller)
 - Produces pairs in alphabetic order, e.g. (Adison, Miller), not (Miller, Adison)
- Note that we omit AS when renaming the relations



Demo

- https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_in
- <https://www.postgresqltutorial.com/load-postgresql-sample-database>

