Università degli Studi di Trieste

Corso di Laurea Magistrale in
INGEGNERIA CLINICA

RICHIAMI DI BASI DI DATI

Corso di Informatica Medica

Docente Sara Renata Francesca MARCEGLIA

Dipartimento di Ingegneria e Architettura

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

DATABASE DEFINITION AND PROPERTIES

A **database** is a collection of **related data** with **an implicit meaning** that:

- Is **logically coherent** (random assortments are not a database)
- Is designed, built, and populated with a **specific purpose** and intended users
- Represents some aspects of the real world (**miniworld**)

EXAMPLE

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

PATIENT	Name	Surname	Address	Location	Unique_ID
	Jack	White	17	MI01	1
	Anna	Green	1	MI03	2
	Herbert	Brown	7	MI01	3

ADDRESS	Street	Number	City	ZIP	State	AddressNumber
	Rockville Pike	3800	New York	19003	NY	1
	Walnut Street	17	Albany	21187	NY	7
	Chestnut Road	17977	Cherry Hill	80087	NJ	17

OPERATIVE UNIT	Unit_Number	Name	Specialty	n.beds
	1	Cardiology 1	Cardiology	55
	2	G.Washington	Oncology	37
	3	M. Montessori	Pediatrics	47

DOCTORS	Name	Surname	Specialty	Unique_ID
	Jane	Smith	Surgery	1
	Anne	Powell	Neurology	2
	Henry	Doe	Pharmacology	3

PRESCRIPTIONS	Patient	Operative Unit	Doctor	Drug name
	3	1	3	Paracetamol
	3	2	1	Antibiotics
	1	3	2	Melatonin

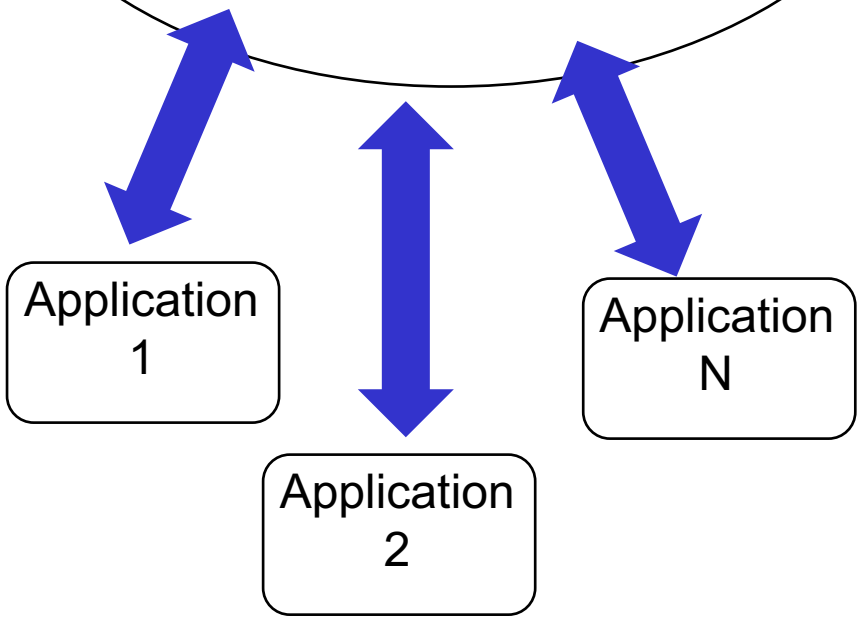
DATABASE APPROACH VS FILE PROCESSING APPROACH (1)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

PATIENT	Name	Surname	Address	Location	Unique_ID
	Jack	White	17	MI01	1
	Anna	Green	1	MI03	2
	Herbert	Brown	7	MI01	3

ADDRESS	Street	Number	City	ZIP	State	AddressNumber
	Rockville Pike	3800	New York	19003	NY	1
	Walnut Street	17	Albany	21187	NY	7
	Chestnut Road	17977	Cherry Hill	80087	NJ	17

OPERATIVE UNIT	Unit_Number	Name	Specialty	n.beds
	1	Cardiology 1	Cardiology	55
	2	G.Washington	Oncology	37
	3	M. Montessori	Pediatrics	47



Application 1

```
struct Patient {
   char Name[30];
   char Surname[30];
   address Address;
   char location[5];
   char uniqueID [21];
   ....
};
```

→ Patient FindPatient (char Name[30]);

Application N

```
struct Patient {
   char Name[30];
   char Surname[30];
   address Address;
   char location[5];
   char uniqueID [21];
   ....
};
```

→ CreateNewPrescription (Patient pat);

DATABASE APPROACH VS FILE PROCESSING APPROACH (2)

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

DATABASE

- A single repository of data is mainteined that is defined once and then accessed by various users
- The database system contains a complete definition or description of the database itself **(self-contained nature)**
- Database access programs are written independently of any specific file **(independence between programs and data)**
- Provide a conceptual representation of data **(data abstraction)**
- Has many users with different perspectives **(multiple views)**

FILE PROCESSING

- Each user defines and implements the files needed for a specific application **(redundancy)**
- Data definition is part of the application program **(data definitions embedded)**
- The structure of the data files is embedded in the application program **(dependency between programs and data)**
- Data are represented by the memory occupation/record length
- Each different user needs a different application **(unique view)**

SCHEMAS AND INSTANCES IN A DATABASE

•The **DATABASE SCHEMA** is the description of the database that is specified during database design and is not expected to change frequently.

•When we define a new database, we **define the schema**.

ADDRESS	Street	Number	City	ZIP	State	AddressNumber

OPERATIVE UNIT	Unit_Number	Name	Specialty	n.beds

•The **DATABASE INSTANCE** is composed by the data in a database at a certain time point (occurrence or state)

•A new database is an "empty istance". When we populate the database, we **load data**

OPERATIVE UNIT	Unit_Number	Name	Specialty	n.beds
	1	Cardiology 1	Cardiology	55
	2	G.Washington	Oncology	37
	3	M. Montessori	Pediatrics	47

DATA BASE MANAGEMENT SYSTEM

•A DBMS is a collection of programs that enables users to create and maintain database.

•It is a generalized software package for implementing maintaining a computerized database:

–DEFINING a database involves specifying the data types, structures and constraints for the data to be stored in the database;

–CONSTRUCTING the database is the process of storing the data itself on some storage medium that is controlled by the DBMS

–MANIPULATING a database includes such functions as query the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.

DBMS PROPERTIES

1. Management of huge amount of data
2. Providing persistent storage for program objects and data Structures
3. Sharing of Data (and Concurrency control)
4. Controlling redundancy
5. Providing Backup & Recovery
6. Restricting Unauthorized Access
7. Providing multiple interfaces
8. Representing complex relationships among data
9. Enforcing integrity constraints

HUGE AMOUNT OF DATA

- A DBMS can manage **Gbyte or Tbyte** of data
- It provides **data organization** → facilitates **data retrieval**

- Examples:
  - Hospital radiology
  - Patient demographics

DATA PERSISTENCY

- Data in a database created thorugh a DBMS are **persistent:** their lifetime goes beyond the execution time of the applications that use the database.

- The file processing approach is persistent too, but is strictly dependent on the programmed application

DATA SHARING

- **Multiple users** can access the database at the same time:
  - Clinicians
  - Nurses
  - Administration personnel
  - IT people
  - …
- **Concurrency control:** the contemporary update by several users is controlled so that no conflicts occur.

CONTROLLING REDUNDANCY

- The database approach integrates the views of different user groups → each logical item is stored only **once and in one place**

- Example:

PATIENT	Name	Surname	Address	Location	Unique_ID
	Jack	White	17	MI01	1

DOCTORS	Name	Surname	Specialty	Unique_ID
	Jane	Smith	Surgery	1

Information on the patient and the doctor are not repeated in the prescription

PRESCRIPTIONS	Patient	Operative Unit	Doctor	Drug name
	1	3	1	Paracetamol

BACKUP AND RECOVERY

- The DBMS should be able to preserve data in the case of hardware or software **failures**;

- Backup and recovery operations make sure that the database is restored to its last consistent state (the state before the program that caused the failure or during which the failure occurred started)

- The system is hence **reliable**

RESTRICTING UNAUTHORIZED ACCESS

- **Multiple users** may not be authorized to access all the information in a database (e.g., clinical vs administrative information)

- The DBMS provides a a security and authorization subsystems that allows creating accounts and specifying their restrictions

- The authorized operations are known as "**privileges**"

PROVIDING MULTIPLE INTERFACES

- Database have different types of users → the DBMS should provide a variety of user interfaces

- They include:
  - different data views
  - different languages → for naive users/for expert users
  - different interfaces → menu-based / programming language based

REPRESENTING COMPLEX RELATIONSHIPS AMONG DATA

- The DBMS should represent all the possible relationships among data

- Example: the patient is related to →
  - Address
  - Doctor
  - Prescription
  - Operative unit
  - Drug therapy
  - …

ENFORCING INTEGRITY CONSTRAINTS

- Data should be consistent → the same data has to be represented by the same datatype
- Relationships can be constraints → a certain record has to be related to another record
- Unique values are constraints → these can be checked directly by the system
- Semantic constraints are more difficult to be automatically checked.

DBMS ACTORS

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Database administrator (DBA)

- responsible of the management of the database and if the DBMS

Database designer

- responsible of choosing the data to be saved in the database and the appropriate structure to manage them

End users

- casual end users: occasional users who access the database and may have different needs
- naive or parametric end users: users who mainly query and update the database
- sophisticated end users: use the database but also understand the complex requirements behind

System analysts and application programmers

- determine the requirements of the end users and develop specification transactions to meet such requirements
- implement the specifications

Other behind the scene

- DBMS designers and implementers
- tool developers
- operators and mainteinance personnel

DBMS ARCHITECTURE

- The advantages of a DBMS instead of a non-specific database system are:

1. Independency between data and applications
2. Multiple views
3. Use of a catalogue to store tha database schema independent from actual data

IMPLEMENTED THORUGH A THREE-LEVEL ARCHITECTURE

THE THREE LEVELS

- **INTERNAL LEVEL**

It has an internal schema that describes the physical storage structure of the database. The Internal schema includes the complete details of data storage and access paths of the database.

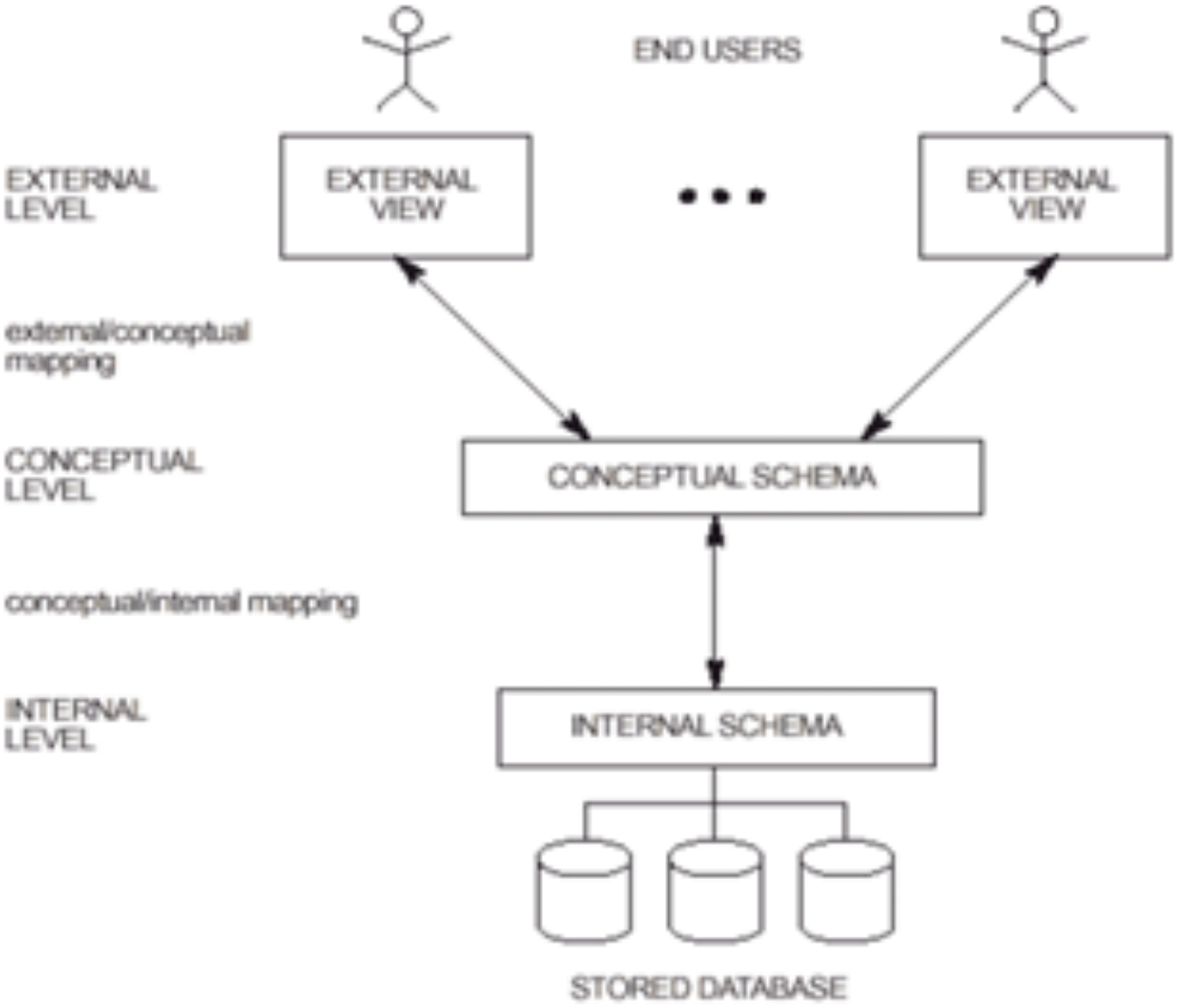
- **CONCEPTUAL LEVEL**

It has a conceptual schema that describes the structure of the whole database for a community of users, hiding the detais of the physical storage. A high-level data model can be used as conceptual schema.

- **EXTERNAL (VIEW) LEVEL**

Each external schema or user view describes the database view for a specific user group, hiding what is not interesting to the particular users.

THE THREE-SCHEMA ARCHITECTURE



DATA INDEPENDENCE

•Data independence can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

•Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs.

•Physical data independence is the capacity to change the internal schema without having to change the conceptual (or external) schemas.

DBMS LANGUAGES

- **Data Definition Language (DDL).** Language for **data defnition, the DDL** is used to specify conceptual schema for the database.
- **Storage Definition Language (SDL).** Language for **data storage, the SDL** defines the internal schema (physical storage of the data)
- **View Definition Language (VDL).** Language for **view defininition, the VDL** specifies user views and their mappings
- **Data Manipulation Language (DML).** Language for **data manipulation, the MDL** can be: high-level non procedural (does not define how the result is obtained but what is wanted) or low-level procedural (defines the procedure to obtain the result)

DEFINITION OF DATA MODEL

A data model is a collection of concepts that can be used to describe the structure of a database

1.Conceptual data models → used to describe data independently from the logical model (e.g. entity-relationship model)

2.Representational data models → used to represent the data in a DBMS

3.Physical data models → describe how data are physically stored in the computer memory (how many files, their size, …)

DATA MODELS IN DBMS

Representational Data models are used to provide the conceptual representation of data in a DBMS.

1. Hierarchical data model
2. Network data model
3. Relational data model
4. Object-oriented data model

DBMS CLASSIFICATIONS

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Data model

- Relational
- Network
- Hierarchical
- Object oriented

Number of contemporary users

- Single-user
- Multi-user

Number of sites where the database is distributed

- Centralized DBMS (the database is stored in a single site)
- Distributed DBMS (the database is distributed over many sites)
- Federated DBMS (local databases have a degree of autonomy)

Cost

THE RELATIONAL DATA MODEL

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Representational Data models are used to provide the conceptual representation of data in a DBMS.

1. Hierarchical data model
2. Network data model
3. **Relational data model**
4. Object-oriented data model

BASIC CONCEPTS

- The relational model represents the data in a database as a collection of **relations**

- A relation resembles a **table** → we can introduce an informal definition

Table = rows, columns

Relation = tuples, attributes

RELATIONS, TUPLES, AND ATTRIBUTES

Relational Model

Query Language

- Relation ←→ Table
- Tuple ←→ Row
- Attribute ←→ Column

Relation

Attribute

Tuple

PATIENT	PHID	FirstName	LastName	Encounter Date	Therapy
	000ZZ000	John	Smith	2003-03-12	Flutamide
	111AA222	Mary	Brown	2004-10-14	Penicillin
	000EE999	Kevin	Green	2001-09-23	Leuprolide
	123XX456	Ann	Black	2002-05-11	Epinephrine

RELATIONS: PROPERTIES

1. There is no order among the relationships between relations (tables) in a relational database.

2. There cannot be two identical tuples in a relation (non-redundancy)

3. Attributes in a relation are not ordered

4. A relation is characterized by:

   relation schema + relation instance

RELATION SCHEMA

It represents the **Relation intension:**

- Relation Name (eg, PATIENT)
- Relation attributes (eg, PHID, FirstName, LastName, EncounterDate, Therapy)

PATIENT	PHID	FirstName	LastName	Encounter Date	Therapy
	000ZZ000	John	Smith	2003-03-12	Flutamide
	111AA222	Mary	Brown	2004-10-14	Penicillin
	000EE999	Kevin	Green	2001-09-23	Leuprolide
	123XX456	Ann	Black	2002-05-11	Epinephrine

RELATION INSTANCE

It represents the **Relation extension:**

•Tuples (=rows) containing actual data are the instance of the relation.

PATIENT	PHID	FirstName	LastName	Encounter Date	Therapy
	000ZZ000	John	Smith	2003-03-12	Flutamide
	111AA222	Mary	Brown	2004-10-14	Penicillin
	000EE999	Kevin	Green	2001-09-23	Leuprolide
	123XX456	Ann	Black	2002-05-11	Epinephrine

KEYS

- **A Relation is a set of tuples in which two tuples cannot be identical (each tuple is unique)**

- This property has to be valid for at least a subset of attributes →
- **There cannot be two or more tuples with the same combination of values for this subset**

PATIENT	Name	Surname	Address	Location	Unique_ID
	Jack	White	17	MI01	1
	Anna	Green	1	MI03	2
	Herbert	Brown	7	MI01	3
	Jack	White	17	MI02	4

SUPERKEYS

PATIENT	PHID	FirstName	LastName	BirthDate	BirthPlace	GP	Diagnosis
	000ZZ000	John	Smith	1980-03-12	NewYork	Parker	Diabetes
	080JJ333	John	Smith	1945-11-08	Los Angeles	Jackson	Hepatitis
	111AA222	Mary	Brown	1955-10-14	San Antonio	Hart	Hypertension
	000EE999	Kevin	Green	1974-09-23	Sydney	Goldman	Cold
	123XX456	Ann	Black	1963-05-11	Frankfurt	O'Neill	Miocarditis

Superkey =

a subset of attributes in a relation that is unique for each tuple.

A **Superkey** univocally identifies a tuyple

KEYS

key = minimum superkey

(superkey for which it is not possible to identify a subset of attributes satisfying the unicity property)

- Example:

{FirstName, LastName, BirthDate, BirthPlace, GP} → it is not a key (it is a superkey)

{FirstName, LastName, BirthDate} → it is a key (it is minimal → I cannot exclude any of the attributes, otherwise the tuples are not unique)

{PHID} → superkey and minimal → it is another key of the same relation.

PRIMARY KEYS

- There is more than one possible key in a Relation
- **Primary Key = key chosen to identify the tuples in a relation**
- Notation → the attributes that constitute the primary key are followed by the **%** symbol

Primary key - examples

1) Primary key =

{FirstName, LastName, BirthDate}

For the PATIENT relation

PATIENT (PHID, FirstName%, LastName%, BirthDate%, BirthPlace, GP, Diagnosis)

2) Primary key =

{PHID}

For the PATIENT relation

PATIENT (PHID%, FirstName, LastName, BirthDate, BirthPlace, GP, Diagnosis)

Integrity constraints

The concept of integrity constraints derives from the observation that not all value combinations are able to represent the information correctly → the introduction of integrity constraints ensures that the information represented are correct

- Intra-relational constraints → within a relation
  - Tuple constraints → constraints on the values of each tuple (NOT NULL, valid interval,…) independent from the others (es. university marks are in the interval [18, 30] e 30 e lode); attribute NOT NULL.
  - Key constraints → no primary key value can be null.
- Inter-relational constraints → between relations
  - Referential integrity constraint → used to mantain the consistency among tuples of two relations

REFERENTIAL INTEGRITY CONSTRAINTS

- Based on the concept of "foreign key"

- A set of attributes FK in the relation R1 is foreign key of R1 if:
  1. The attributes in FK have the same domain as the primary key attributes PK of another relation R2
  2. A value of FK in a tuple t1 of R1 either occurrs as a value of PK in some tuple t2 in R2 or is null t1[FK] = t2[PK]

REFERENTIAL INTEGRITY CONSTRAINT: EXAMPLE

Foreign key of PRESCRIPTIONS (FK)

PRESCRIPTIONS	Patient	Operative Unit	Doctor	Drug name
	1	3	1	Paracetamol
	3	2	1	Antibiotics
	1	3	2	Melatonin

1. The two attributes have the same domain
2. The values occurring in Operative Unit occur in Unit_Number and are Primary Keys

OPERATIVE UNIT	Unit_Number	Name	Specialty	n.beds
	1	Cardiology 1	Cardiology	55
	2	G.Washington	Oncology	37
	3	M. Montessori	Pediatrics	47

Primary key of OPERATIVE UNIT (PK)

THE NULL VALUE: MULTIPLE MEANINGS

1. Not valid for the current instance (Husband surname for a male)
2. Valid but not yet existing (Husband surname for a non-martried woman)
3. Existing but it cannot be saved (patient's religion in some Countries cannot be stored to avoid discrimination)
4. Existing but unknown
5. Existing but not yet saved (patint's history noyt collected yet)
6. Stored and then deleted (erroneous information)
7. Available but in an updating phase (patient's therapy under modification)
8. Available but not reliable (a non final diagnosis)
9. Available but not valid (a blood parameter abovce the threshold of valid range)
10. Calculted from anothe NULL value (BMI if the weight is not present).

DATABASE QUERYING: RELATIONAL ALGEBRA OPERATIONS

- SELECT
  - From all the rows in a table, this operation selects only those that satisfy a certain condition
- PROJECT
  - From all the columns in a table, this operation selects only a subset
- CARTESIAN PRODUCT
  - Given two tables, it creates all the possible combinations of the rows in each table
- JOIN
  - Selects only some rows satisfying a certain condition after a cartesian product

SELECT OPERATION - σ

- Is used to select a specific subset of tuples in a relation
- The selected tuples must satisfy a selection condition
- The result of the selection operation is a new relation with the same attributes as the starting relation and only the selected tuples

$$\text{New\_Relation} \leftarrow \sigma_{(\text{condition})}(\text{Relation\_Name})$$

Condition: usually a comparison operation with constant values and Boolean operators AND, OR, NOT

SELECT OPERATION – σ
Example (1/3)

We want to select the Cardiology patients

Patient (ID%, family_Name, Diagnosis_Date%,Diagnosis Physician_Name, Operative_Unit)

PATIENT	ID%	Family Name	Diagnosis Date%	Diagnosis	Physician Name	Operative Unit
	1123	White	12/11/85	Stroke	Ackerman	Cardoiology
	1123	White	4/4/87	Ventricular arrythmia	Fontelo	Emergency
	1763	Green	3/31/79	Stroke	Reds	Cardiology
	1763	Green	4/25/99	Angina	Grey	Medicine I
	1763	Green	11/18/03	Angina	Rome	Medicine II
	2156	Brown	2/27/01	SA nodal block	Hanna	Cardiology

ID and diagnosis date are the primary key

SELECT OPERATION – σ
Example (2/3)

$$\text{New\_Patient} \leftarrow \sigma_{(\text{Operative\_Unit} = \text{“Cardiology”})}(\text{Patient})$$

SELECT OPERATION – σ
Example (3/3)

NEW_PATIENT	ID%	Family Name	Diagnosis Date%	Diagnosis	Physician Name	Operative Unit
	1123	White	12/11/85	Stroke	Ackerman	Cardoiology
	1763	Green	3/31/79	Stroke	Reds	Cardiology
	2156	Brown	2/27/01	SA nodal block	Hanna	Cardiology

New relation with:

- The **SAME ATTRIBUTES**
- Only the **TUPLES SATISFYING THE CONDITION**

PROJECT OPERATION - π

- Is used to select certain columns from the table and discard the other columns
- Used when you are interest only in a subset of attributes
- The result is a new relation with the same tuples but different attributes

$$New\_Relation \leftarrow \pi_{(attribute\_list\ )}(Relation\_Name)$$

PROJECT OPERATION – π
Example (1/3)

We want to select the Name, Surname, and Date of Birth

Patient (Name, Surname, Birthdate,Gender)

PATIENT	Name	Surname	Birthdate	Gender
	Jack	White	11/5/61	M
	Anna	Green	7/9/25	F
	Mary	Brown	3/16/80	F
	Jack	Reds	9/15/73	M

PROJECT OPERATION – π
Example (2/3)

$$\text{New\_Patient} \leftarrow \pi_{(\text{Surname},\text{Name},\text{Birthdate})}(\text{Patient})$$

PROJECT OPERATION – π
Example (3/3)

NEW_PATIENT	Name	Surname	Birthdate
	Jack	White	11/5/61
	Anna	Green	7/9/25
	Mary	Brown	3/16/80
	Jack	Reds	9/15/73

New relation

- **SAME TUPLES**
- Only the **ATTRIBUTES LISTED** in the selection criterion

Set Theoretic Operations (1/5)

- **UNION OPERATION**
- It operates over two relations (R1 and R2)
- The result of this operation is a relation that includes all the tuples that are either in R1 or in R2
- Duplicate tuples are eliminated

$$\text{New\_Relation} \leftarrow R1 \cup R2$$

UNION OPERATION
Example (1/3)

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

PATIENT_UO1	ID%	Family Name	Diagnosis Date%	Diagnosis	Physician Name
	1123	White	12/11/85	Stroke	Ackerman
	1763	Green	3/31/79	Stroke	Reds
	2156	Brown	2/27/01	SA nodal block	Hanna

PATIENT_UO2	ID%	Family Name	Diagnosis Date%	Diagnosis	Physician Name
	1123	White	12/11/85	Stroke	Ackerman
	1763	Green	4/25/99	Angina	Grey
	1763	Green	11/18/03	Angina	Rome

We want to obtain the union of these two relations

UNION OPERATION
Example (2/3)

Patient_Union ← $Patient\_UO1 \cup Patient\_UO2$

UNION OPERATION
Example (3/3)

PATIENT_UNION	ID%	Family Name	Diagnosis Date%	Diagnosis	Physician Name
	1123	White	12/11/85	Stroke	Ackerman
	1763	Green	3/31/79	Stroke	Reds
	2156	Brown	2/27/01	SA nodal block	Hanna
	1763	Green	4/25/99	Angina	Grey
	1763	Green	11/18/03	Angina	Rome

New relation

- **SAME ATTRIBUTES**
- **ALL THE TUPLES**
- **THE DUPLICATE PATIENT WAS DELETED**

Set Theoretic Operations (2/5)

INTERSECTION OPERATION

- It operates over two relations (R1 and R2)
- The result of this operation is a relation that includes all the tuples that are both in R1 and in R2

New_Relation ← $R1 \cap R2$

INTERSECTION OPERATION
Example (1/3)

PATIENT_UO1	ID%	Family Name	Diagnosis Date%	Diagnosis	Physician Name
	1123	White	12/11/85	Stroke	Ackerman
	1763	Green	3/31/79	Stroke	Reds
	2156	Brown	2/27/01	SA nodal block	Hanna

PATIENT_UO2	ID%	Family Name	Diagnosis Date%	Diagnosis	Physician Name
	1123	White	12/11/85	Stroke	Ackerman
	1763	Green	4/25/99	Angina	Grey
	1763	Green	11/18/03	Angina	Rome

We want to obtain the intersection of these two relations

INTERSECTION OPERATION
Example (2/3)

$$\text{Patient\_Inters} \leftarrow \mathit{Patient\_UO1} \cap \mathit{Patient\_UO2}$$

INTERSECTION OPERATION
Example (3/3)

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

PATIENT_INTER	ID%	Family Name	Diagnosis Date%	Diagnosis	Physician Name
	1123	White	12/11/85	Stroke	Ackerman

New relation

- **SAME ATTRIBUTES**
- Only the tuple that was in **BOTH RELATIONS**

Set Theoretic Operations (3/5)

SET DIFFERENCE OPERATION

- It operates over two relations (R1 and R2)
- The result of this operation is a relation that includes all tuples that are in R1 but NOT in R2

New_Relation ← $R1 - R2$

SET DIFFERENCE OPERATION
Example (1/3)

PATIENT_UO1	ID%	Family Name	Diagnosis Date%	Diagnosis	Physician Name
	1123	White	12/11/85	Stroke	Ackerman
	1763	Green	3/31/79	Stroke	Reds
	2156	Brown	2/27/01	SA nodal block	Hanna

PATIENT_UO2	ID%	Family Name	Diagnosis Date%	Diagnosis	Physician Name
	1123	White	12/11/85	Stroke	Ackerman
	1763	Green	4/25/99	Angina	Grey
	1763	Green	11/18/03	Angina	Rome

We want to obtain the set difference of these two relations

SET DIFFERENCE OPERATION
Example (2/3)

Patient_Diff ← $Patient\_UO1 - Patient\_UO2$

SET DIFFERENCE OPERATION
Example (3/3)

PATIENT_DIFF	ID%	Family Name	Diagnosis Date%	Diagnosis	Physician Name
	1763	Green	3/31/79	Stroke	Reds
	2156	Brown	2/27/01	SA nodal block	Hanna

New relation

- **SAME ATTRIBUTES**
- Only the tuples that were in **UO1 BUT NOT in UO2**
- **The ORDER** of the relations in the operation is **RELEVANT (not COMMUTATIVE)**

Set Theoretic Operations (4/5)

CARTESIAN PRODUCT OPERATION

- It operates over two relations (R1 and R2)
- The two relations do not need to be UNION COMPATIBLE (= the two relations have the same number of attributes and each attribute pair has the same domain)
- The result of this operation is a relation that (1) combines all the tuples from R1 and R2 and (2) has all the attributes of both R1 and R2

$$\text{New\_Relation} \leftarrow R1 X R2$$

CARTESIAN PRODUCT
Example (1/3)

PATIENT	Name	Surname	Birthdate
	Jack	White	11/5/61
	Anna	Green	7/9/25
	Mary	Brown	3/16/80

DIAGNOSIS	Pat_Name	System	Ref_Operative_Unit
	Stroke	Cardiovascular	Cardiology
	Asthma	Respiratory	Pneumology
	Parkinson's Disease	Nervous	Neurology
	Angina	Cardiovascular	Cardiology

We want to create the cartesian product between the patients and the diagnoses

CARTESIAN PRODUCT
Example (2/3)

Pat_Dia ← *PatientXDiagnosis*

CARTESIAN PRODUCT
Example (3/3)

PAT_DIA	Name	Surname	Birthdate	Pat_Name	System	Ref_Operative_Unit
	Jack	White	11/5/61	Stroke	Cardiovascular	Cardiology
	Jack	White	11/6/61	Asthma	Respiratory	Pneumology
	Jack	White	11/7/61	Parkinson's Disease	Nervous	Neurology
	Jack	White	11/8/61	Angina	Cardiovascular	Cardiology
	Anna	Green	7/9/25	Stroke	Cardiovascular	Cardiology
	Anna	Green	7/10/25	Asthma	Respiratory	Pneumology
	Anna	Green	7/11/25	Parkinson's Disease	Nervous	Neurology
	Anna	Green	7/12/25	Angina	Cardiovascular	Cardiology
	Mary	Brown	3/16/80	Stroke	Cardiovascular	Cardiology
	Mary	Brown	3/17/80	Asthma	Respiratory	Pneumology
	Mary	Brown	3/18/80	Parkinson's Disease	Nervous	Neurology
	Mary	Brown	3/19/80	Angina	Cardiovascular	Cardiology

New relation

- **ALL ATTRIBUTES**
- **ALL TUPLES**
- The result per se does **not have a real meaning** but it is a matematical tool

Set Theoretic Operations (5/5)

JOIN

- It operates over two relations (R1 and R2)
- The result of this operation is a relation that (1) combines related tuples from R1 and R2 into single tuples and (2) it is a cartesian product followed by a selection

$$\text{New\_Relation} \leftarrow R1 \bowtie_{(\text{condition})} R2$$

JOIN
Example (1/3)

PATIENT	ID	Name	Surname	Birthdate
	1	Jack	White	11/5/61
	2	Anna	Green	7/9/25
	3	Mary	Brown	3/16/80
	4	Jack	Reds	9/15/73

DIAGNOSIS	Patient_ID	Diagnosis	Dia_Date
	1	Stroke	12/11/85
	2	Asthma	3/31/79
	3	Parkinson's Disease	2/27/01
	4	Angina	4/25/99

We want to calculate the result of the JOIN operation between these two relations

JOIN
Example (2/3)

$$\text{Pat\_Dia} \leftarrow \text{Patient} \bowtie_{(\text{ID}=\text{Pat\_ID})} \text{Diagnosis}$$

JOIN
Example (3/3)

PAT_DIA	ID	Name	Surname	Birthdate	Patient_ID	Diagnosis	Dia_Date
	1	Jack	White	11/5/61	1	Stroke	12/11/85
	2	Anna	Green	7/9/25	2	Asthma	3/31/79
	3	Mary	Brown	3/16/80	3	Parkinson's Disease	2/27/01
	4	Jack	Reds	9/15/73	4	Angina	4/25/99

New relation

- **ALL ATTRIBUTES**
- **ALL TUPLES satisfying the condition**
- It is the same result as the selection of the tuples where ID=Pat_ID after a cartesian product of Patient and Diagnosis

$$Pat\_Dia \leftarrow \sigma_{(ID = Pat\_ID)} (Patient \times Diagnosis)$$

NATURAL JOIN

- It operates over two relations (R1 and R2)

- The result of this operation is a relation that (1) combines related tuples from R1 and R2 into single tuples, but the set of common attribute is unique (2) it is a cartesian product followed by a selection and a projection

$$New\_Relation \leftarrow R1 \bowtie R2$$

NATURAL JOIN
Example (1/3)

PATIENT	ID	Name	Surname	Birthdate
	1	Jack	White	11/5/61
	2	Anna	Green	7/9/25
	3	Mary	Brown	3/16/80
	4	Jack	Reds	9/15/73

DIAGNOSIS	Patient_ID	Diagnosis	Dia_Date
	1	Stroke	12/11/85
	2	Asthma	3/31/79
	3	Parkinson's Disease	2/27/01
	4	Angina	4/25/99

We want to calculate the result of the NATURAL JOIN operation between these two relations

NATURAL JOIN
Example (2/3)

$$Pat\_Dia \leftarrow Patient \bowtie Diagnosis$$

NATURAL JOIN
Example (3/3)

PAT_DIA	ID	Name	Surname	Birthdate	Diagnosis	Dia_Date
	1	Jack	White	11/5/61	Stroke	12/11/85
	2	Anna	Green	7/9/25	Asthma	3/31/79
	3	Mary	Brown	3/16/80	arkinson's Diseas	2/27/01
	4	Jack	Reds	9/15/73	Angina	4/25/99

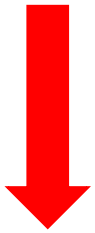
New relation

- **The same as the one obtained by the JOIN but the patient ID is repeated only once**
- It is the same result as the selection of the tuples where ID=Pat_ID after a cartesian product of Patient and Diagnosis and projecting all the attributes except Pat_ID

NATURAL JOIN vs JOIN

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

PAT_DIA	ID	Name	Surname	Birthdate	Patient_ID	Diagnosis	Dia_Date
	1	Jack	White	11/5/61	1	Stroke	12/11/85
	2	Anna	Green	7/9/25		Asthma	3/31/79
	3	Mary	Brown	3/16/80		Parkinson's Disease	2/27/01
	4	Jack	Reds	9/15/73	4	Angina	4/25/99



PAT_DIA	ID	Name	Surname	Birthdate	Diagnosis	Dia_Date
	1	Jack	White	11/5/61	Stroke	12/11/85
	2	Anna	Green	7/9/25	Asthma	3/31/79
	3	Mary	Brown	3/16/80	arkinson's Diseas	2/27/01
	4	Jack	Reds	9/15/73	Angina	4/25/99

RENAME OPERATION

- It operates over SINGLE RELATIONS
- It allows giving new names to the attributes of a relation
- The result of this operation is a relation that has the same attributes as the original one but with different names

$$New\_Relation \leftarrow \rho_{(renaming\_criteria)} (Relation\_Name)$$

(renaming_criteria) are in the form:
B1,B2,…Bn ← A1,A2,…,An

RENAME OPERATION
Example

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

PATIENT	Name	Surname	Birthdate	Gender
	Jack	White	11/5/61	M
	Anna	Green	7/9/25	F
	Mary	Brown	3/16/80	F
	Jack	Reds	9/15/73	M

$$New\_Pat \leftarrow \rho_{(Name,Surname \leftarrow First\_Name,Family\_Name)}(Patient)$$

PATIENT	First Name	Family Name	Birthdate	Gender
	Jack	White	11/5/61	M
	Anna	Green	7/9/25	F
	Mary	Brown	3/16/80	F
	Jack	Reds	9/15/73	M

Operating on relational databases: The SQL

- SQL (Structured Query Language): query language for relational databases;
- SQL is an ISO standard: independent from the implementation system;
- It specifies the characteristics of the results (Declarative Languge) instead of the operations needed to obtain the results (as in procedural languages);
- SQL uses the terms Table, Row, Column that correspond to Relation, Tuple, Attribute in the relational model.

SQL FUNCTIONS

Main SQL functions
(*SQL: Structured Query Language*)

- Data definition
- Data update
- Query

The general query model

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

The SELECT command syntax

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

select *Column_Name* [[**as**] *New_Column_Name*]
{, *Column_Name* [[**as**] *New_Column_Name*
}

from *Table_Name* [[**as**] *alias*]
{, *Table_Name* [[**as**] *alias*] }

[**where** *Condition*]

Example

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Surname	Name	BirthDate	Gender
Bianchi	Luca	1962-05-08	M
Mascheroni	Marinella	1965-12-02	F
Strozzi	Giulia	1964-02-11	F
Aldobrandi	Enrico	1960-02-29	M

1 Retrieve names and surnames of the male patients;

2 Retrieve all data of the patients whose surname is “Bianchi”;

3 Retrieve names, surnames, and birth year of all patients.

Query 1

Retrieve names and surnames of the male patients;

Surname	Name	BirthDate	Gender
Bianchi	Luca	1962-05-08	M
Mascheroni	Marinella	1965-12-02	F
Strozzi	Giulia	1964-02-11	F
Aldobrandi	Enrico	1960-02-29	M

SELECT *Surname*,*Name*

FROM *Demographic*

WHERE (*Gender*=`*M*` **OR** *Gender*=`*m*`);

RESULT:

Surname	Name
Bianchi	Luca
Aldobrandi	Enrico

Query 2

Retrieve all data of the patients whise surname is “Bianchi”;

SELECT *

FROM *Demographics*

WHERE (*Surname=`Bianchi`*);

Surname	Name	BirthDate	Gender
Bianchi	Luca	1962-05-08	M
Mascheroni	Marinella	1965-12-02	F
Strozzi	Giulia	1964-02-11	F
Aldobrandi	Enrico	1960-02-29	M

RESULT:

Surname	Name	BirthDate	Gender
Bianchi	Luca	1962-05-08	M

Query 3

Retrieve names, surnames, and birth year of all patients.

SELECT *Surname, Name,* **year**(*Birth_Date*) **AS** *Year*

FROM *Demographics*

Surname	Name	BirthDate	Gender
Bianchi	Luca	1962-05-08	M
Mascheroni	Marinella	1965-12-02	F
Strozzi	Giulia	1964-02-11	F
Aldobrandi	Enrico	1960-02-29	M

RESULT:

Surname	Name	Year
Bianchi	Luca	1962
Mascheroni	Marinella	1965
Strozzi	Giulia	1964
Aldobrandi	Enrico	1960