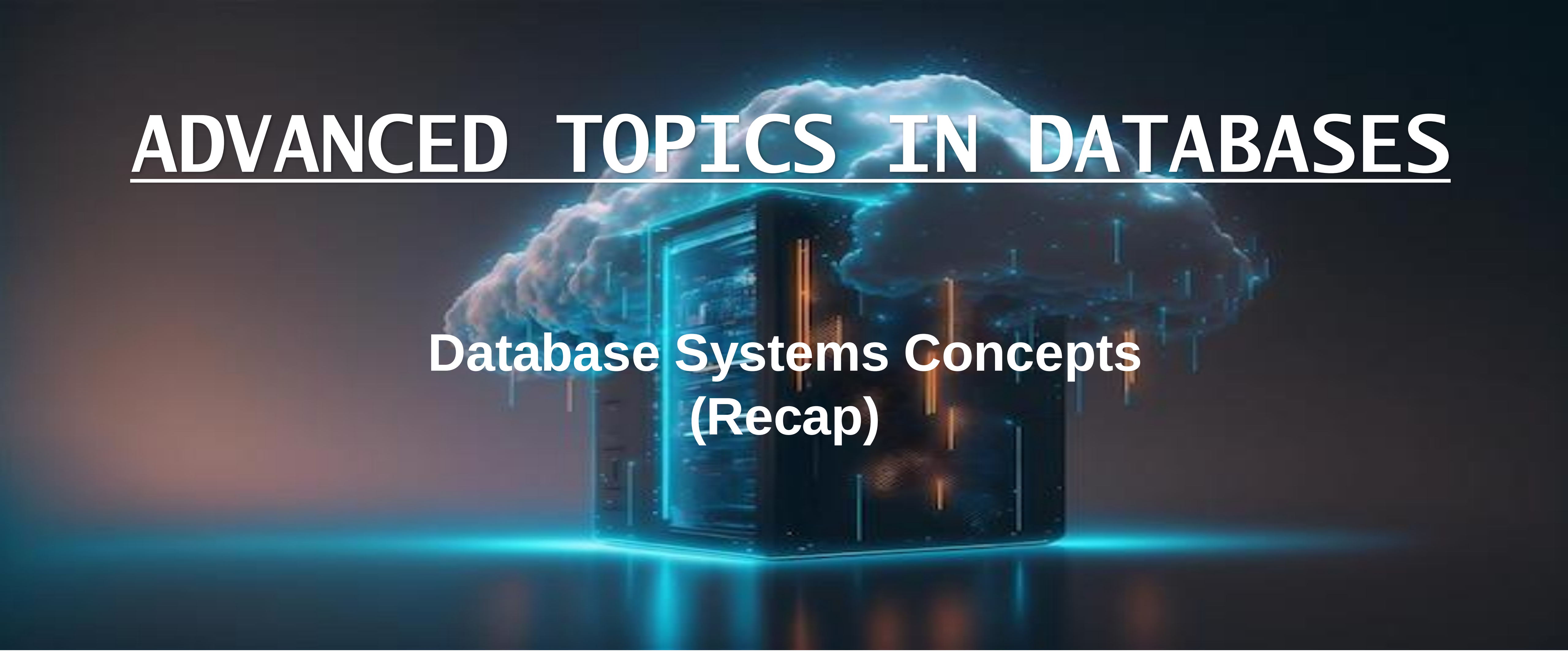


ADVANCED TOPICS IN DATABASES

Database Systems Concepts (Recap)

Master in Informatics Engineering
Data Engineering

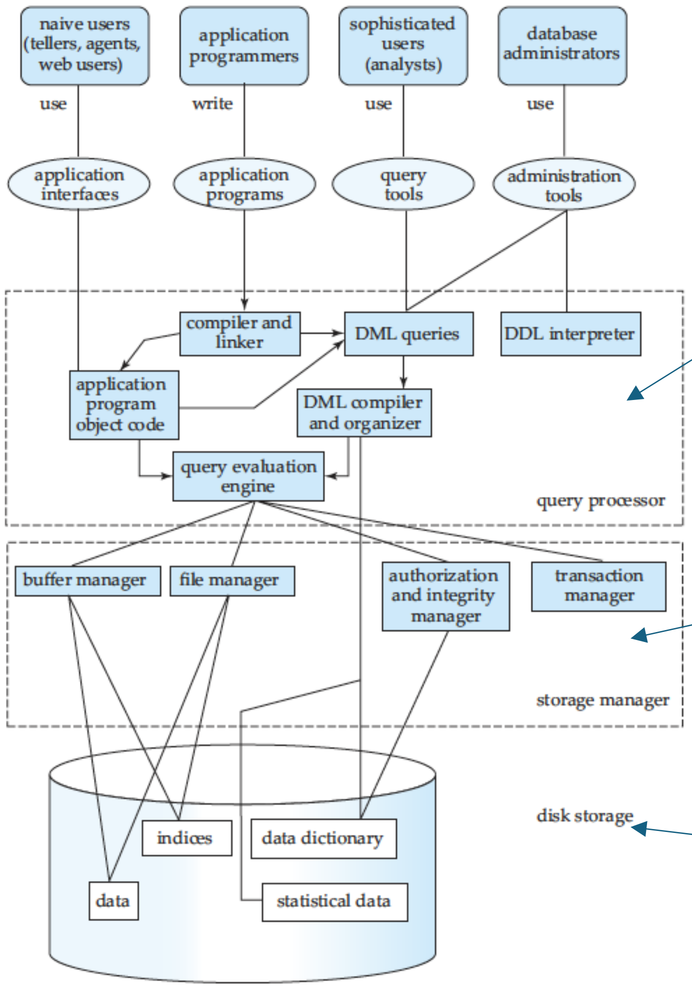
Informatics Engineering Department

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Database Management System (DBMS)

- **A Database Management System (DBMS) is a collection of interrelated data and various programs that are used to handle that data.**
- The **primary goal** of DBMS is **to provide a way to store and retrieve the required information from the database in convenient and efficient manner.**
- **For managing** the data in the database **two important tasks** are conducted:
  - (i) Define the structure for storage of information.
  - (ii) Provide mechanism for manipulation of information.

Components of DBMS



Database System Concepts-Silberschatz, Korth and Sudarshan

The **Query Processor** transforms (or interprets) the user's application program-provided requests into instructions that a computer can understand.

The **storage manager** is responsible to the following tasks:

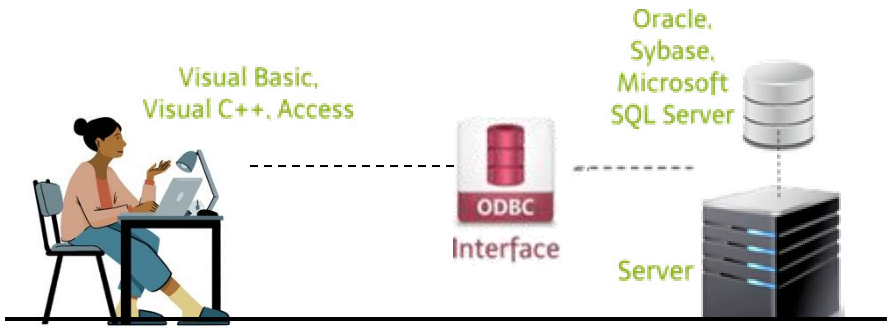
- Interaction with the OS file manager
- Efficient storing, retrieving and updating of data

Disk storage writes data to a physical disk.

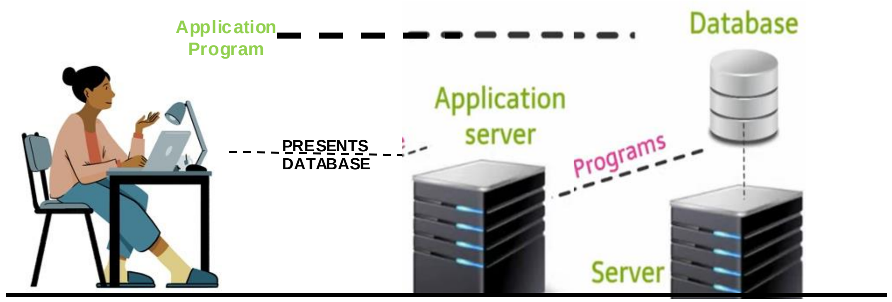
Architecture of a DBMS

- The term "database architecture" refers to the structural design and methodology of a database system, which forms the core of a Database Management System (DBMS).
- DBMS Architecture **helps users to get their requests d**one while connecting to the database.
- the choice database architecture depends on several factors:
  - size of the database,
  - number of users, and relationships between the users.

Types of DBMS Architecture



Two-tier architecture, clients are connecting directly to a database.

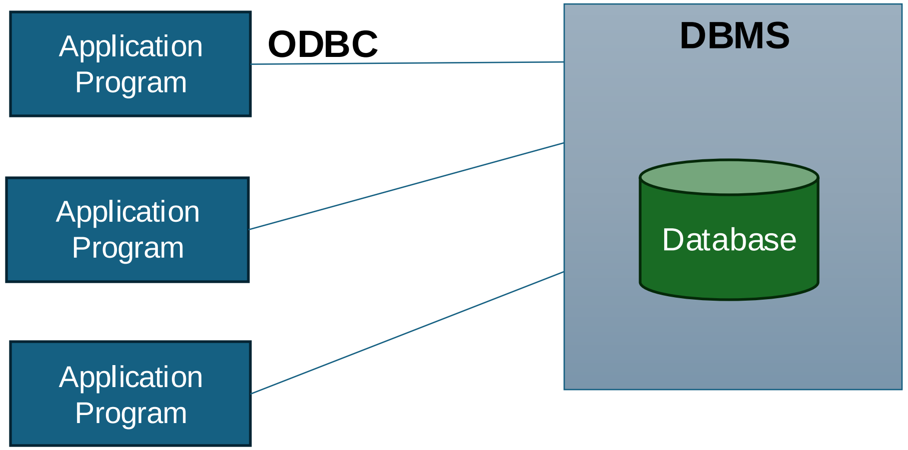


A three-tier architecture is composed of three layers: the data, the application, and the presentation.

Two Tier Client/Server Architecture

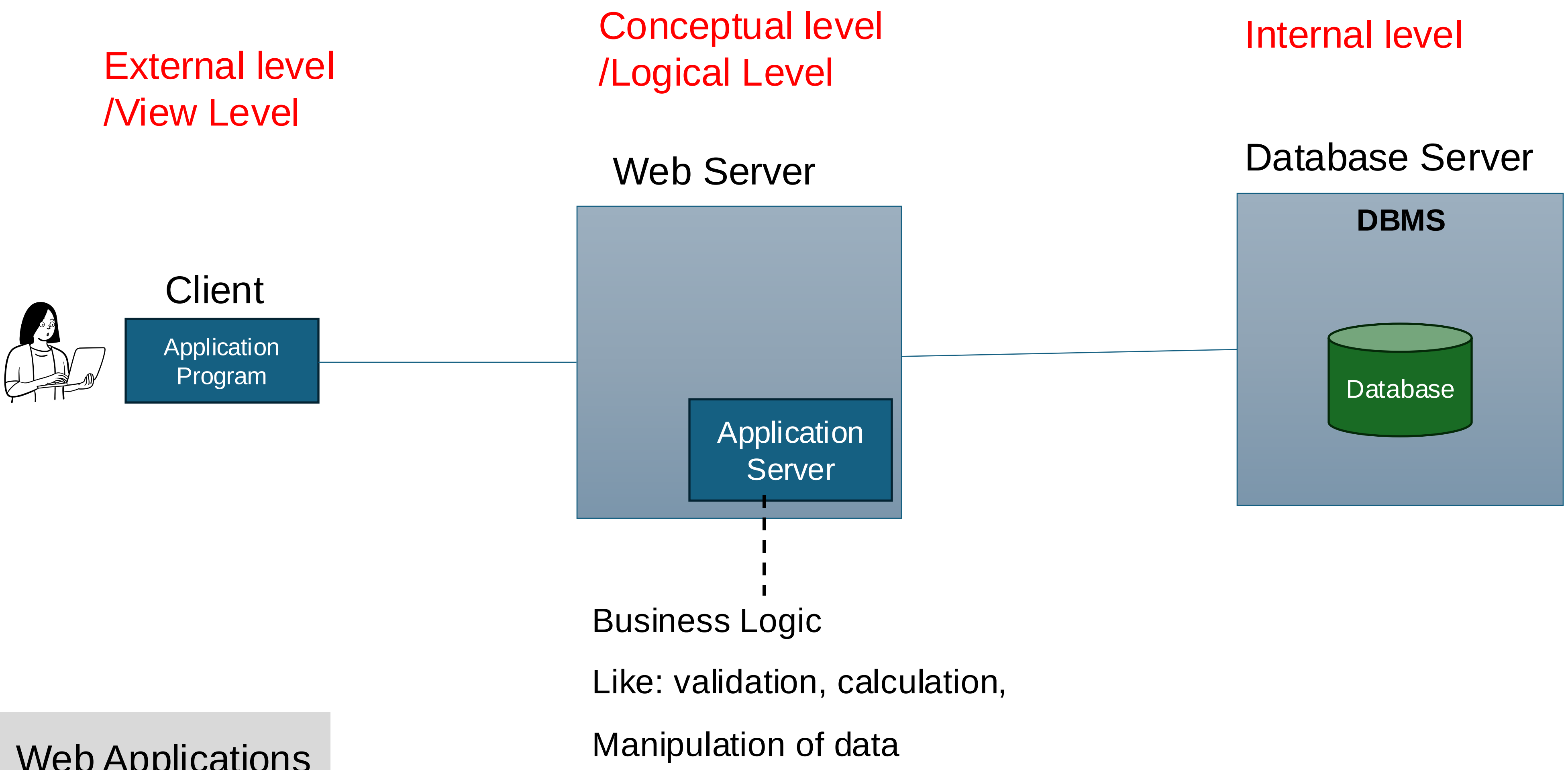
- A Key aspect of **modern database architecture is the** **client-server model**, particularly relevant in two-tier and three-tier architectures.

- **The client, usually a front-end application**, **interacts with the server**, requesting data and presenting it to the end user.
  - This separation **enhances data integrity, security, and management efficienc**y, allowing for **a more robust and scalable system.**



Used in application programs that run on the client- side. They in with the database directly.

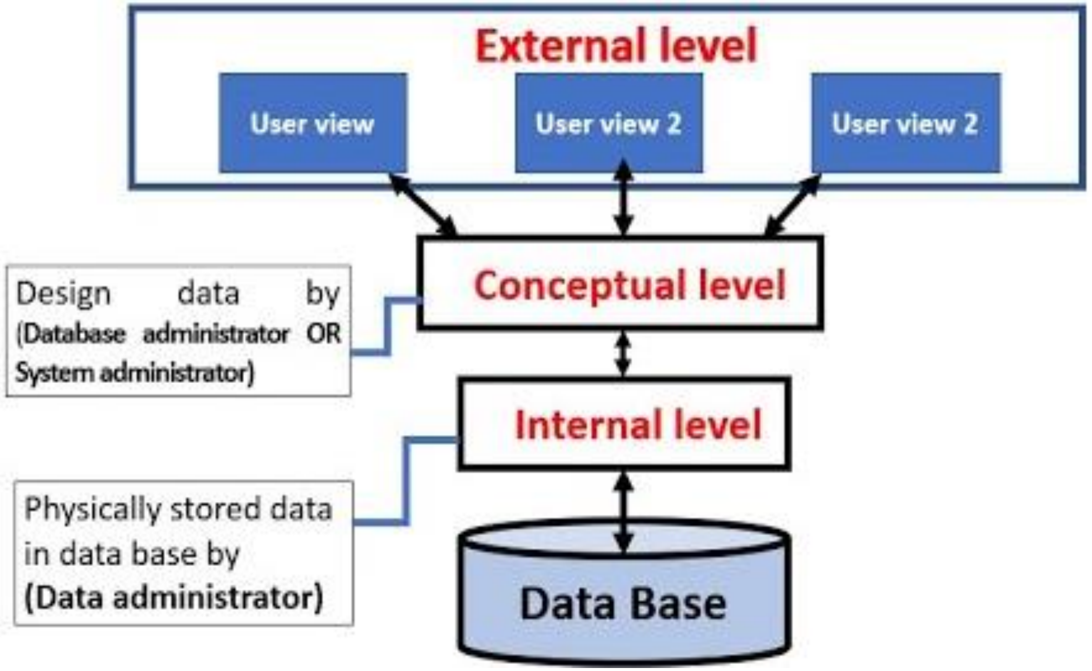
Three Tier Client/Server Architecture



Used for Web Applications

Three Tier Client/Server Architecture

Tier architecture divides the complete system into three inter-related but independent modules



Three Tier Client/Server Architecture

Advantages

- **Enhanced scalability:** Scalability is enhanced **due to the distributed deployment of application servers**. Now, **individual connections** need **not be made** between **the client and server.**
- **Maintains Data Integrity**. Since there **is a middle layer** between **the client and the server**, **data corruption can be avoided/removed**.
- **Improves Securi**ty. This type of model **prevents direct interaction** of the **client with** the **server** thus **reducing access to unauthorized data**.

Types of Database Models

- The design of database depends upon the database model being implemented
- **A database model** is “a type of database structure that determines the logical structured of database and fundamentally determines, the manner in which the data can be stored, organized and manipulated within a database”
- **The choice of a database model is often influenced** by the **specific requirements** of the application, the **nature of the data**, and the **desired performance characteristics**.

Types of DataBase Models

- Relational ← **Most DBMS**
- Key/Value
- Graph
- Document / Object
- Wide-Column / Column-family

← **NoSQL**

- Array / Matrix / Vectors ← **Machine Learning**
- Hierarchical
- Network
- Multi-Value

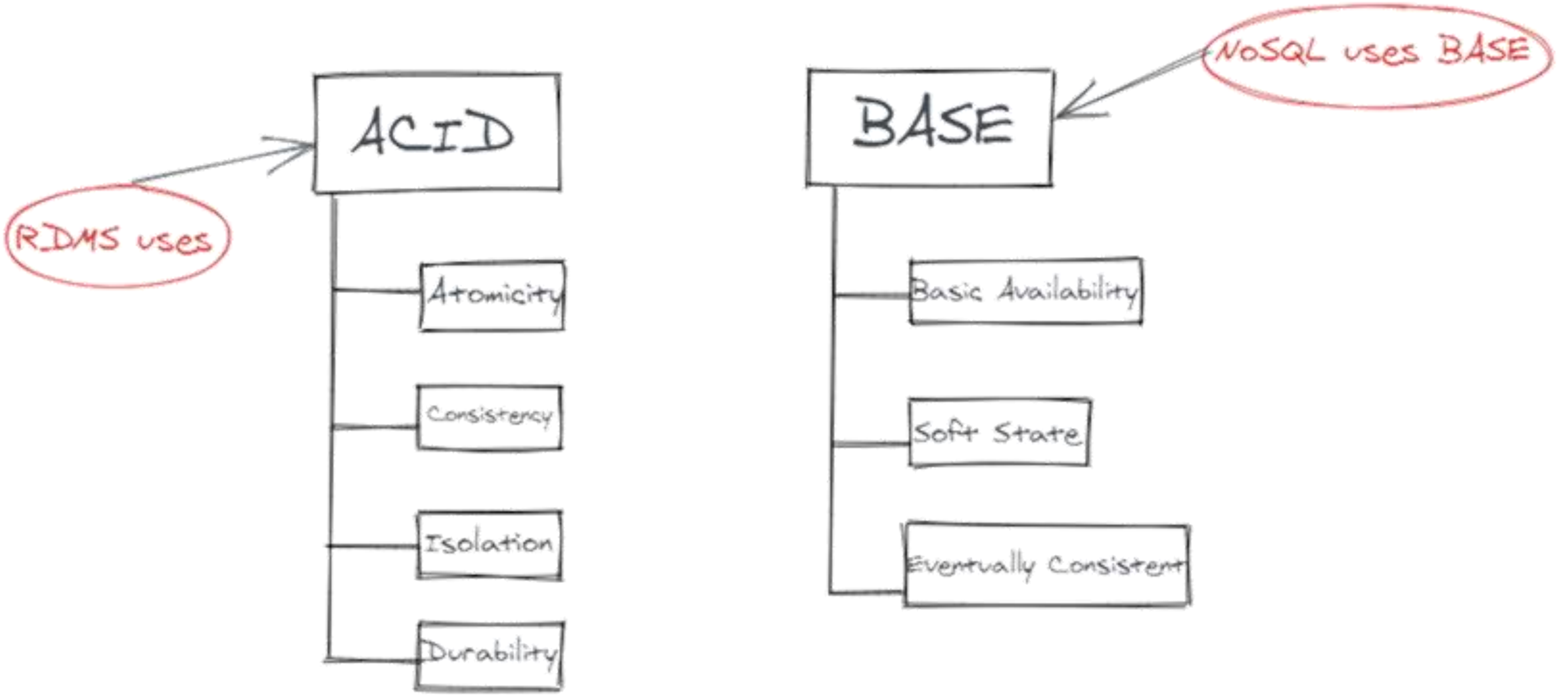
← **Legacy**

- ……

Transaction

A **transaction** is a unit of program execution that accesses and possibly updates various data items.

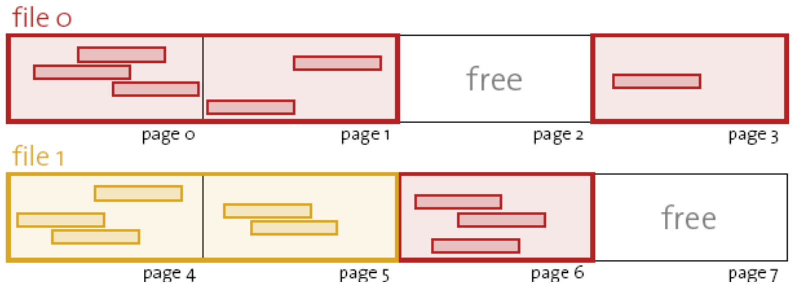
To preserve the integrity of data the database system must ensure:



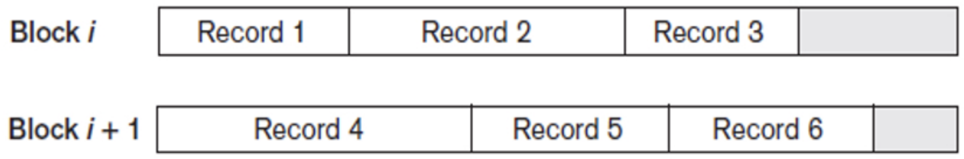
Source: AlgoDaily

Database Data Storage

- The data in a BD is normally stored in files of records:
  - A **file** consists of **one or more pages**.
  - Each **page** contains **one or more records**.
  - Each **record** corresponds to **one tuple**.



- To store data from a BD on disk:
  - **Blocks - are** units of both storage allocation and data transfer.
  - Given that the unit of transfer to the disk is the block, to store data from a BD on disk we have to associate records with **blocks on disk**.



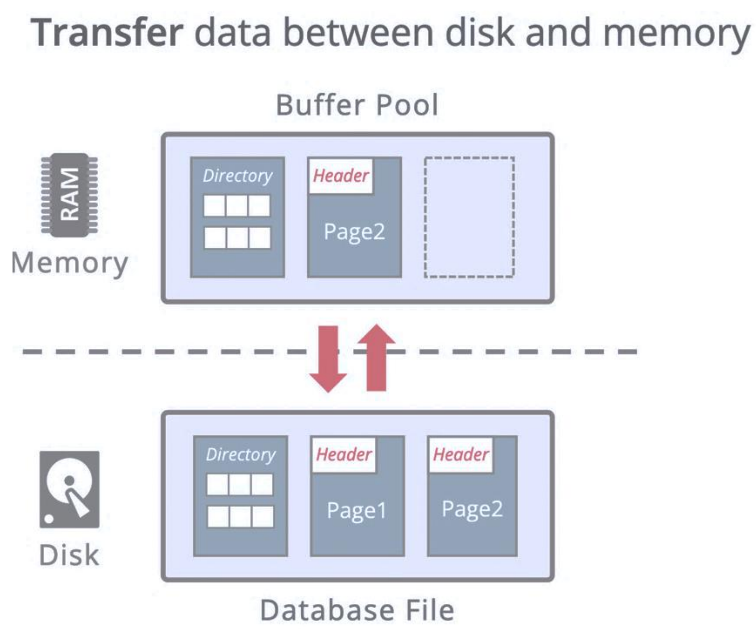
- If the **block size** is B bytes and the **record size is** R bytes (assuming B ≥ R) then the number of records per block **(blocking factor or bfr) is B/R.**

Database Data Storage

Objective:

- Database system seeks to minimize the number of block transfers between the disk and memory. We can **reduce the number of disk accesses** by keeping as **many blocks as possible in main memory.**

- **Buffer manager** – subsystem responsible for allocating buffer space in main memory



Buffer – portion of main memory available to store copies of disk blocks.

Database Data Storage

- **Association** between **records and blocks on di**sk, it is important to know the **organization of the record file**, that is, **how records are organized on disk**
  - Unordered files (heap files)
  - Sorted files
  - Direct access files (hash files)

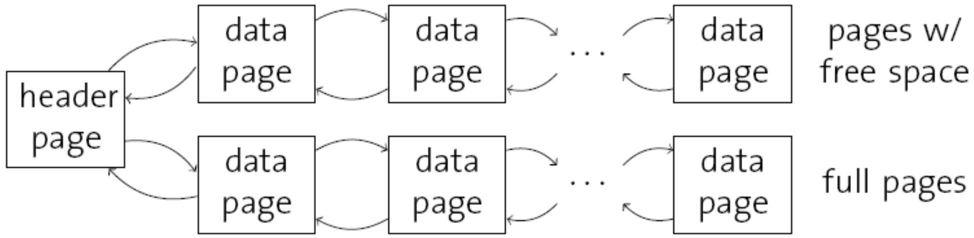
good organization is to minimize the number of block transfers from disk to memory that are required to locate a given record.

Database Data Storage

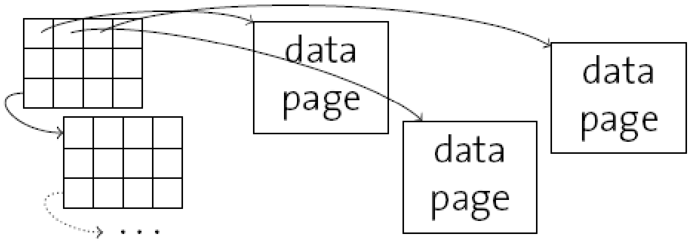
666666	MGT123	F1994	4.0	page 0
123456	CS305	S1996	4.0	
987654	CS305	F1995	2.0	
717171	CS315	S1997	4.0	page 1
666666	EE101	S1998	3.0	
765432	MAT123	S1996	2.0	
515151	EE101	F1995	3.0	
234567	CS305	S1999	4.0	page 2
878787	MGT123	S1996	3.0	

- **Heap files**
  - It stores records in **no particular order** (in line with, e.g., SQL)
  - In this method records are **inserted at the end** of the file, **into the data blocks.**
  - No Sorting or Ordering is required in this method.
  - It is the responsibility of DBMS to store and manage the new records.
  - Requires **a linear seach** if we want search one record

Linked list of pages

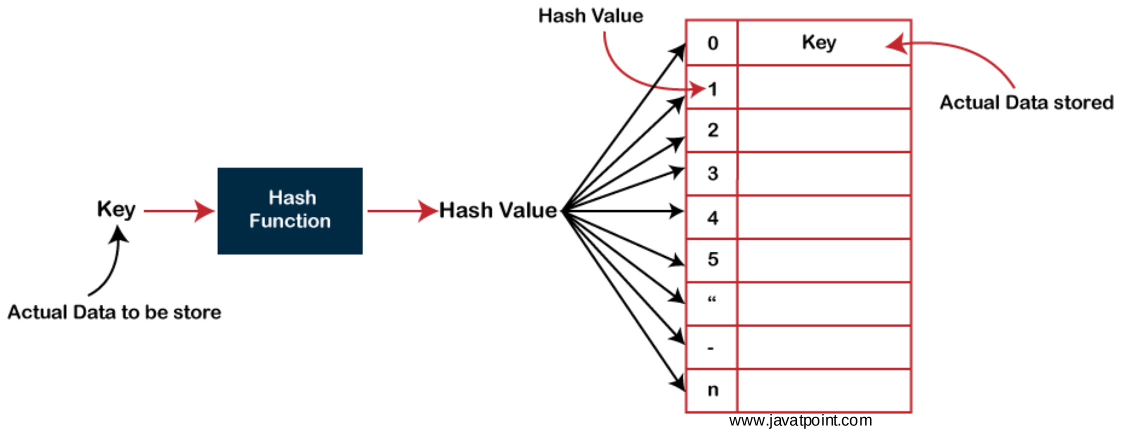


Directory of pages



- **Sorted/ Ordered File**
  - store records sorted in order, based on the value of search key of each record
  - need **external sort** or **an index to keep sorted**

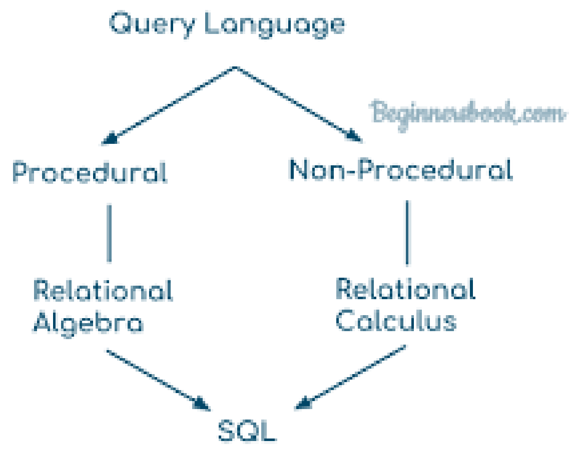
- **Hashing**
  - in DBMS **is a technique for quickly finding a data record** in a database, regardless of the size of the database.
  - Uses a **Hash table to store data** records by **using a Hash function**



Query Languages

- Allow manipulation and retrieval of data from a database
- Two mathematical Query Languages form the basis for “real” languages (e.g. SQL), and for implementation:
    - **Relational Algebra**: More operational, very useful for representing execution plans.
    - **Relational Calculus**: Lets users describe what they want, rather than how to compute it.

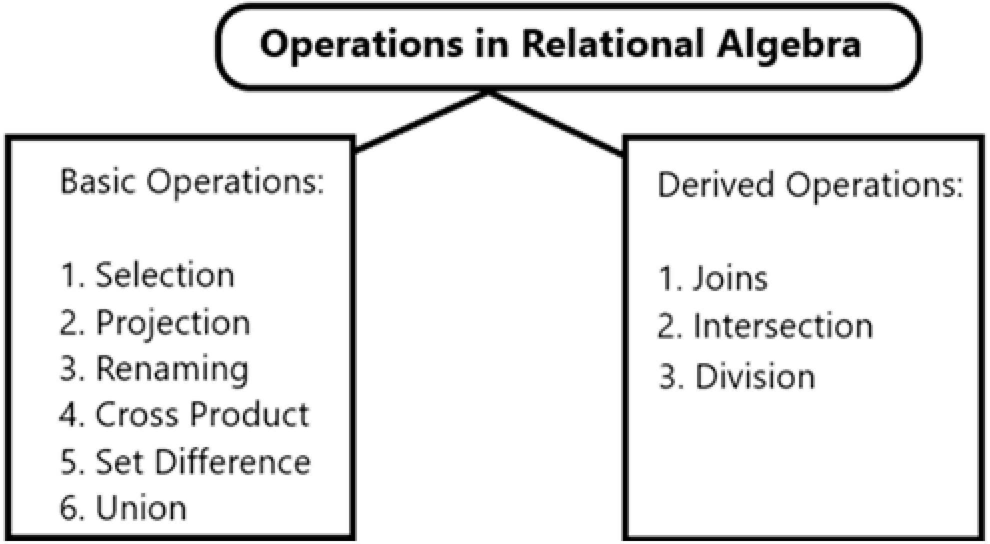
      (Non-operational, declarative)



- ♦ **Understanding Algebra & Calculus** is key to **understanding SQL, query processing!**

Relation ALgebra

- Is a procedural query language, which takes instances of relations as input and yields instances of relations as output.
- Every operator of the algebra accepts as arguments instances of relations (one or more) and returns a relation as a result.
- Allows you to compose operators to generate a complex research
- Basic operators of algebra:
  - Selection
  - Projection
  - Union
  - Cross Product
  - Difference



Operations from set theory

Operator	Notation	Result Degre	Result Cardinality	Restritions
Union	r ∪ s	gr(r)= gr(s)	=n(r)+ n(s)	1,2
Difference	r - s	gr(r)= gr(s)	<= n(r)	1,2
Cartesian product	r x s	gr(r) + gr(s)	n(r)* n(s)	
Intersect	r ∩ s	gr(r)= gr(s)	<=min(n(r), n(s))	1,2

Restrictions

1- The r and s relations must have the same degree

2- Attributes must be compatible

Specificied operations of relations

Operator	Notation	Description
Selection	σ_{P} (r)	tuples of r that satisfy the selection predicate P
Projection	$\pi_{A1 \,..\, An}(r)$	projection of tuples of r into a list of attributes
renaming	$\rho_{S[\ B1..Bn]}$ (e)	renames the result of the expression e and how S [B1,....Bk]
Join	r $\bowtie_{P}$ s	concatenation of tuples of r and s that satisfy the join predicate P
EquiJoin	r ⋈ s	concatenation of tuples of r and s that match in common attributes, eliminating duplicates

Example

- Example to follow throughout the presentation:

  Sailors(sid: integer, sname: string, rating: integer, age: real)

  Boats(bid: integer, bname: string, color: string)

  Reserves(sid: integer, bid: integer, day: date)

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Selection and Projection

- **Selection and Projection**
  - Relational algebra has operators for selecting tuples from a relation (**σ**) and for projecting columns (**π**).
  - These operations allow you to manipulate data in a single relation.

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

❖ **Selection Operator**

Select sid, sname, rating, age
From S2
Where rating >8

Relation 2 = σ $_{\text{<condition>}}$ (Relation 1)

- where <condition> is a Boolean combination of terms in the form:
  - Constant or Attribute
  - the operator can be <, <=, =, , >= or >

$\sigma_{rating>8}(S2)$

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

Selection and Projection

❖ **Projection Operator**

Relation 2= π <list of attributes> (Relation 1)

- The Projection takes an initial relation and originates another relation whose scheme is reduced to set of attributes present in the <attribute list>

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Select sname, rating
From S2

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$$\pi_{sname,rating}(S2)$$

Selection and Projection

The records repeated are eliminated.

Select age
From S2

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

age
35.0
55.5

$$\pi_{age}(S2)$$

Selection and Projection

Since the result of a relational algebra expression **is a relation,** we **can replace the relation by an expression**

S2

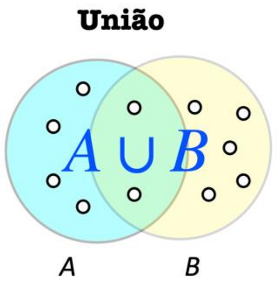
sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

sname	rating
yuppy	9
rusty	10

Union Operator

Relation 3 = (Relation 1 ∪ Relation 2)



- ➢ Returns a Relation that contains all the tuples that occur in the Relation 1 or Relation 2 or both.
- ➢ There must be compatibility between the Relation 1 and the Relation 2:
  - Must have the same number of attributes
  - Corresponding fields with the same domain

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

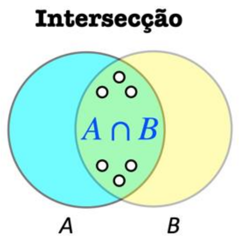
sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S1 U S2

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

Intersection Operator

Relation 3 = (Relation 1 ∩ Relation 2)



- Returns a relation that contains all the tuples that occur in both relations.
- Just as the union must be compatible between relation 1 and relation 2.

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

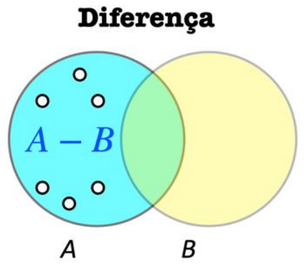
$S1 \cap S2$

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

Difference operator

Relation 3 = (Relation 1 - Relation 2)

- Returns a relation with all tuples that occur in relation 1 and do not occur in Relation 2.
- Again, there must be compatibility between relation 1 and relation 2.



S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

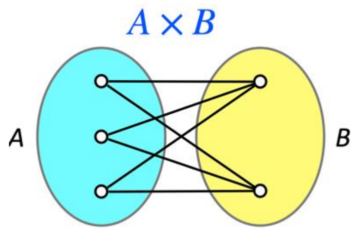
S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S1 - S2

sid	sname	rating	age
22	dustin	7	45.0

Cartesian Product Operator



Relation 3 = (Relation 1 X Relation 2)

- It produces a relation with all possible combinations between the tuples of relation 1 with 2, that is, its cardinality is the multiplication of the cardinalities of the initial relations.
- The scheme produced is the "sum" of the schemes of th initial relations (1 and 2)

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1 X R1

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Join Operator

- is one of the most widely used operations in relational algebra and the most common form to combine information from two or more relationships.

Relation 3 = (Relation 1 $\bowtie_{<condição>}$ Relation 2)

- Produces a relation with all combinations possible between the records of the relationship 1 with the 2 conditioned by <condição>
- The scheme produced is the "sum" of the schemes of the initial relations (1 and 2).

Join Operator

Conditional Join example

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

Equi Join Example

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$$S1 \bowtie_{S1.sid = R1.sid} R1$$

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

Natural Join $R \bowtie S$

Special case of Join where equalities are specified in all fields that have the same name in both relations.In the following example, the following two expressions are equal:

$$R \bowtie S$$

$$S1 \bowtie_{R.sid=S.sid} R1$$

Division Operator

- is important for expressing certain types of searches, such as ”Find the name of the sailors who booked all the boats.”
- The word **ALL** is usually associated with division.
    - Let's define the division through an example:
    - Considering two relations A and B, where A has two fields (x and y) and B have only one (y) with the same domain in A. We define the A/B division as the set of all values x, such that for **all** y existing in B has a tuple in A.

Division Operator

Example

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

B2

pno
p2
p4

B3

pno
p1
p2
p4

A/B1

sno
s1
s2
s3
s4

A/B2

sno
s1
s4

A/B3

sno
s1

Thus, we can define **A/B** as

$$\pi_x(A) - \pi_x((\pi_x(A) \times B) - A)$$

Division Operator

Example

Specify expressions like:

"What are the students enrolled in **all** subjects"

Relação INSCRICAO

ID_ALUNO	ID_DISCIPLINA
1090	PT
1090	IN
1080	PT
1070	PT
1060	PT
1060	IN

÷

Relação DISCIPLINA

ID_DISCIPLINA	DESIGNACAO
PT	PORTUGUÊS
IN	INGLÊS

=

Relação resultante

ID_ALUNO	ID_DISCIPLINA
1090	PT
1090	IN
1060	PT
1060	IN

Division Operator

- ✓ Derived Operator (from Primitive Operators)

Equivalent expression

$$R1 \div R2 \equiv \pi_C(R1) - \pi_C(\,[\pi_C(R1) \times R2] - R1)$$

Cartesian product: All tuples of R1 combined with all tuples of R2.

All unrelated tuples of R1 with tuples of R2.

All tuples of R1 related to all tuples of R2

Division Operator

Who are the students enrolled in all subjects?

- algebraic solution 1

$$INSCRICAO \div DISCIPLINA$$

- algebraic solution 2

$$\pi_{ID\_ALUNO}(INSCRICAO) - \pi_{ID\ ALUNO}(\,[\Pi_{ID\ ALUNO}(INSCRICAO) \times DISCIPLINA] - INSCRICAO)$$

- algebraic solution 3

$T1 = \pi_{ID\_ALUNO}(INSCRICAO)$ → ID_ALUNO of all enrolled students

$T2 = T1 \times DISCIPLINA$ → all Id_ALUNO combined with all All disciplines

$T3 = \pi_{ID\_ALUNO}(T2 - INSCRICAO)$ → all Id_ALUNO without enrolled for all subjects

$T = T1 - T3$ → all ID_ALUNO enrolled in all subjects

Renaming Operator

- **Renaming - ρ**

Helper operator, does not derive new result, just renames relations and fields

The renaming operation is represented by the expressions

$\rho S(R)$ or $\rho(B1, B2, ..., Bn)(R)$ or $\rho S(B1, B2, ..., Bn)(R)$

where ρ is the renaming operator, S is the new relation name and B1, B2, ..., Bn are the new attribute names

$$DEP4\_SAL2000 \leftarrow \sigma_{NumDep = 4 \text{ AND } Salário > 2000}(EMPREGADO)$$

$$RESULT \leftarrow \pi_{NumBI, NomeP, NomeF}(DEP4\_SAL2000)$$

$$\rho_{DEP4\_SAL2000}(\sigma_{NumDep = 4 \text{ AND } Salário > 2000}(EMPREGADO))$$

$$\rho_{RESULT(BI, Nome, Apelido)}(\pi_{NumBI, NomeP, NomeF}(DEP4\_SAL2000$$

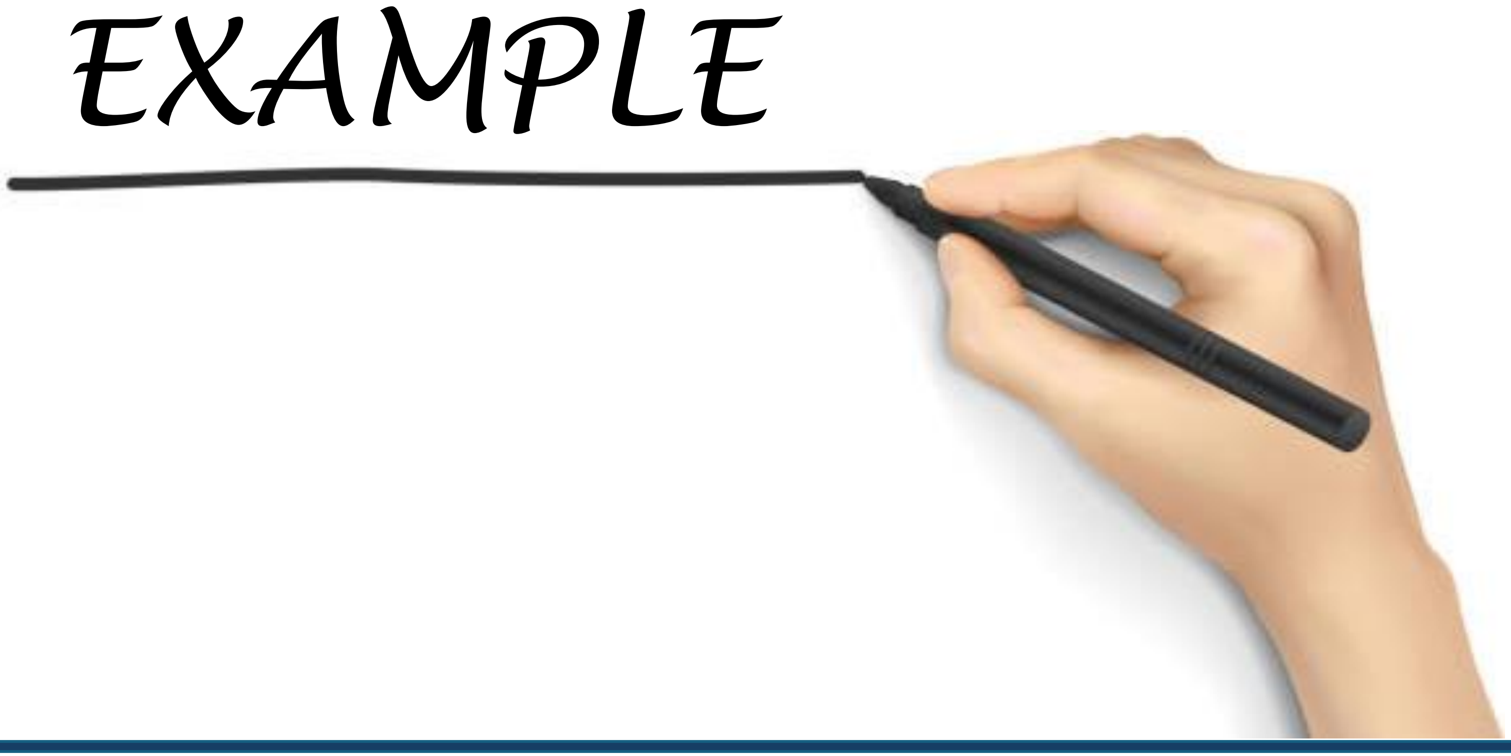
RESULT	BI	Nome	Apelido
	798764544	João	Santos
	342342324	Ana	Feio

Operations of Relational Algebra

Table 6.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{(\text{<join attributes 1>}),\ (\text{<join attributes 2>})} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{(\text{<join attributes 1>}),\ (\text{<join attributes 2>})} R_2$ OR $R_1 * R_2$

Relational Algebra



Queries Formulation

Instance $S3$ of Sailors

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Instance $R2$ of Reserves

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Instance $B1$ of Boats

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Find the name of Sailors who have reserved Bid 103.

$$\pi_{sname}(\sigma_{bid\,=\,103}\ (Reserves \infty Sailors))$$

Find the name of Sailors who have reserved a Boat red.

$$\pi_{sname}((\sigma_{color\,=\,\text{'red'}}\ Boats) \infty Reserves \infty Sailors\ )$$

Queries Formulation

Find names of sailors who've reserved a red or a green boat

$$\pi_{sname}(\sigma_{color='red' \text{ or } color = 'green'} \text{ Boats} \infty \text{ Reserves} \infty \text{ Sailors})$$

Find the names of sailors who have reserved a red and a green boat

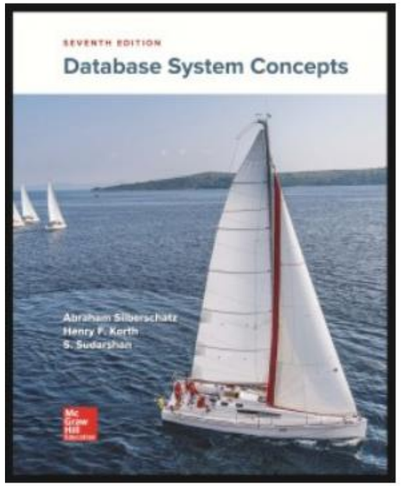
- Solution:

$$\cancel{\pi_{sname}(\sigma_{color='red' \text{ and } color = 'green'} \text{ Boats} \infty \text{ Reserves} \infty \text{ Sailors})}$$

A ship cannot have TWO colors at the same time

$$\pi_{sname}(\sigma_{color='red'} \text{ Boats} \infty \text{ Reserves} \infty \text{ Sailors})$$
$$\cap$$
$$\pi_{sname}(\sigma_{color = 'green'} \text{ Boats} \infty \text{ Reserves} \infty \text{ Sailors})$$

Readings



Chapter 1 – Overview

Chapter 2 – Intro to Relational Model

Chapter 13 – Data Storage Structure

Chapter 17 – Transactions

https://www.javatpoint.com/structure-of-dbms