Relational Model

COURSE 3: Databases

Relational model

Relational model

- Codd rules 1985 → Is DBMS relational? If yes, to what degree?

- **Regula 0.** Un SGBD Relaţional trebuie să fie capabil să gestioneze BD exclusiv pe baza caracteristicilor sale relaţionale.

Relational Integrity constraints

RELATIONS

OPERATORS

Relational model

- Codd rules 1985 → Is DBMS relational? If yes, to what degree?

- Regula 1. Regula reprezentării logice a datelor: Într-o BD relaţională, toate datele sunt reprezentate la nivel logic într-un singur mod, sub formă de **valori atomice** în tabele.

- Regula 2. Regula accesului la date: Toate datele individuale din tabele trebuie să fie accesibile prin furnizarea numelui tabelului, numelui coloanei şi valorii cheii primare. Nu există tupluri identice.
  - ROWID in Oracle

Relational model

- Regula 3. Regula reprezentării valorilor necunoscute: Un sistem relaţional trebuie să permită declararea şi manipularea sistematică a valorilor Null, cu semnificaţia unor valori necunoscute sau inaplicabile.

- Regula 4. Regula dicţionarului de date: Descrierea BD (dicţionarul de date) trebuie să fie reprezentată la nivel logic tot sub formă de tabele, astfel încât asupra acesteia să se poată aplica aceleaşi operaţii ca şi asupra datelor propriu-zise.

Relational model

- Regula 5. Regula limbajului de acces: Într-un sistem relaţional trebuie să existe cel puţin un limbaj de accesare a datelor, care să asigure următoarele operaţii:
  - definirea tabelelor de bază şi a tabelelor virtuale (vederilor) CREATE, ALTER, DROP,
  - manipularea şi interogarea datelor (atât interactiv cât şi prin program) INSERT UPDATE, DELETE, SELECT
  - definirea restricţiilor de integritate, CONSTRAINT
  - autorizarea accesului la date, ROLES, PRIVILEGES
  - delimitarea tranzacţiilor. COMMIT, ROLLBACK

Relational model

- Regula 6. Regula de actualizare a tabelelor virtuale (vederilor): Un SGBD trebuie să poată determina dacă o vedere poate să fie actualizată sau nu.

- Regula 7. Regula manipulării datelor: Un sistem relaţional trebuie să ofere posibilitatea procesării tabelelor (de bază sau virtuale) nu numai în operaţiile de interogare a datelor cât şi în cele de inserare, actualizare şi ştergere. INSERT folosind subcerere.

-

Relational model

- Regula 8. Regula independenţei fizice a datelor: Programele de aplicaţie nu trebuie să depindă de modul de stocare şi accesare fizică a datelor.

- Regula 9. Regula independenţei logice a datelor: Programele de aplicaţie nu trebuie să fie afectate de nici o restructurare logică a tabelelor BD care conservă datele.

- Regula 10. Regula independenţei datelor din punctul de vedere al integrităţii: **Regulile de integritate a BD** trebuie să fie definite în limbajul utilizat de sistem pentru definirea datelor şi nu în cadrul aplicaţiilor individuale; în plus, aceste reguli de integritate trebuie stocate în dicţionarul de date.

Relational model

- Regula 11. Regula independenţei datelor din punctul de vedere al distribuirii: Programele de aplicaţie nu trebuie să fie afectate de distribuirea pe mai multe calculatoare a BD.

- Regula 12. Regula privind prelucrarea datelor de către un limbaj de nivel inferior: Orice limbaj nerelaţional folosit pentru accesarea datelor trebuie să respecte aceleaşi condiţii de integritate ca şi limbajul relaţional de acces.

Relational model

- Codd rules 1985 → Is DBMS relational? If yes, to what degree?

Relational Integrity constraints

RELATIONS

OPERATORS

Relational model

- Database = collection of RELATIONS
  - relation in relational model ≠ relationship in ERD.
  - relation in relation model < -- > table with lines and columns

- Relation Schema: A relation schema represents the name of the relation with its attributes.

- Attribute domain – Each attribute has some pre-defined values.

Relational Integrity constraints

RELATIONS

OPERATORS

- Relational schema $R(A_1, A_2, \ldots, A_n)$
- $R \subset D_1 \times D_2 \times \cdots \times D_n$, D_i $domain$
- Example
  - Participant(participant_id, last_name, first_name)
  - A1 - - participant_id D1 - - integer size 6
  - A2 - - last_name D2 - - string, length 20
  - A3 - - first_name D3 - - string, length 20

Relational Integrity constraints

RELATIONS

OPERATORS

- Domain constraints
  - “the value of each attribute must be unique”, specifies data types: integers, real numbers, characters, Booleans; variable length for strings, numbers etc.
- Key constraint
  - Unique + not null -- PK
- Referential integrity constraints
  - the value of a FK is null or it corresponds to the value of a PK.

Relational Integrity constraints

RELATIONS

OPERATORS

- UNION, INTERSECT, PRODUCT, DIFFERENCE - - *next lecture*

- PROJECT
- SELECT
- JOIN
- DIVISION

Converting ERD into RM

Rules for entities

- Strong entities → independent tables
  - PK doesn’t contain foreign keys.

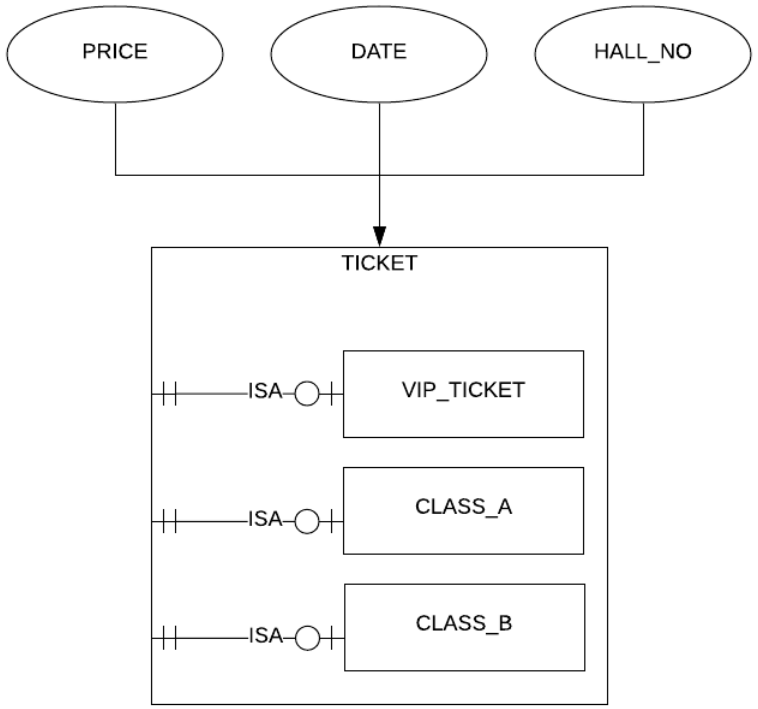
- Weak entities → table
  - PK contains the key of the related strong entity and one or more key attributes.

- Sub-entities → one ore more tables/ Boolean attribute, /type_attribute
  - PK of a subentity may also represent a FK.

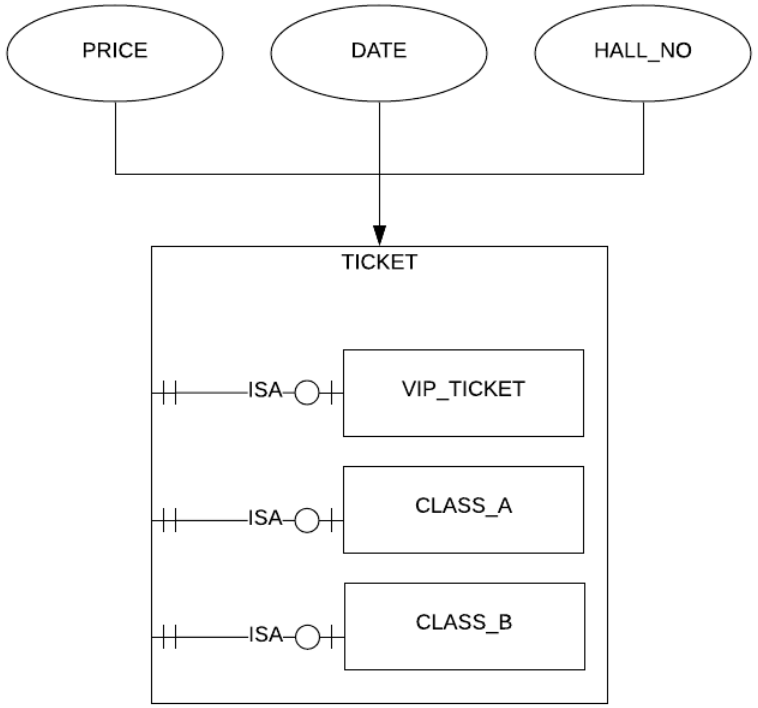
Rules for entities strong – weak entity

AIRPLANE ($airplane\_id$, …)
SEAT ($airplane\_id, seat\_id$, …, observations)

Rules for entities ISA

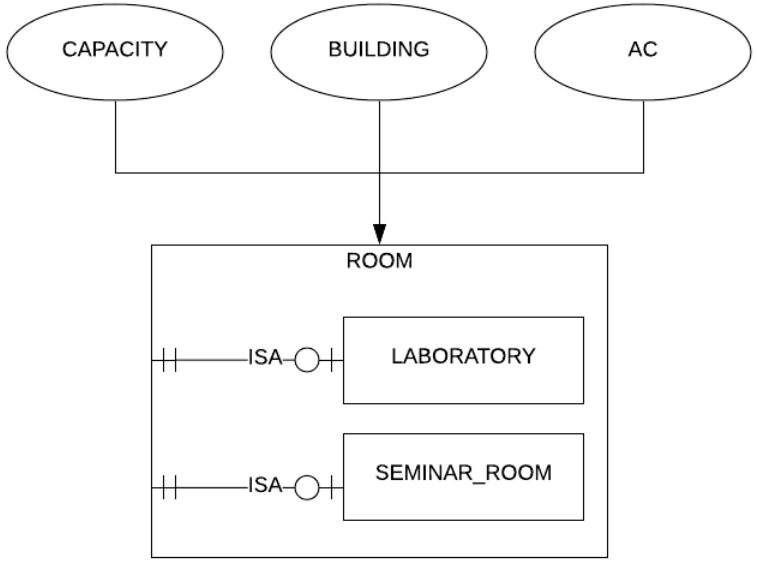


Rules for entities ISA

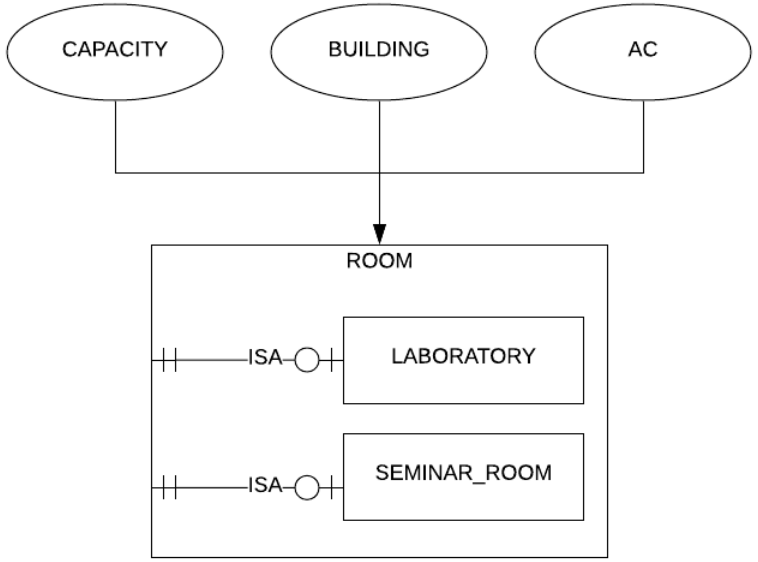


TICKET_ID	PRICE	HALL_NO	DATE	TYPE
1	200	Coliseum	08/03/20	VIP
2	150	Lyttelton	14/04/20	A
3	140	Olivier	01/05/20	A
4	90	Coliseum	04/06/20	B
5	220	Lyttelton	08/03/20	VIP
6	95	Olivier	14/04/20	B
7	210	Coliseum	20/03/20	VIP

Rules for entities ISA

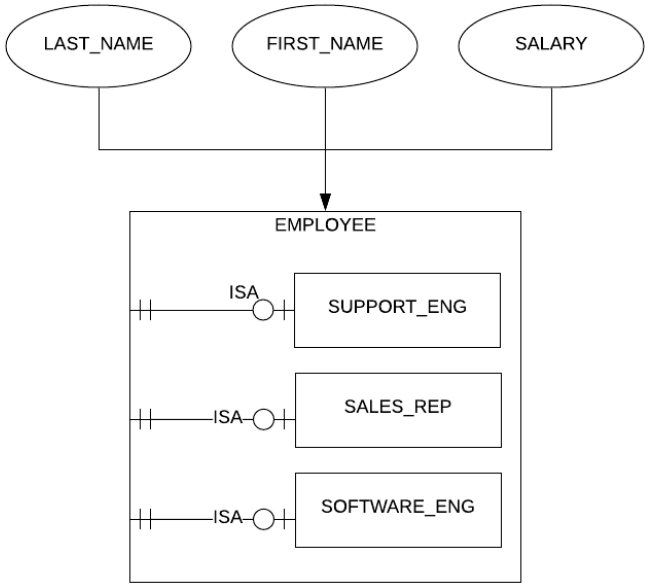


Rules for entities ISA

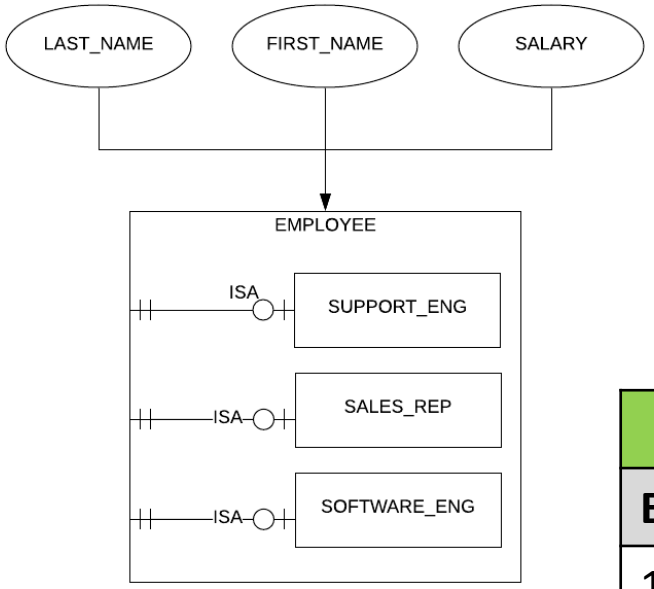


ROOM_ID	CAPACITY	BUILDING	LAB	SEM
1	40	FMI	1	1
2	45	Magurele	1	0
3	30	Geografie	0	0
4	90	FMI	1	0
5	80	FMI	1	0
6	95	Drept	0	1
7	20	FMI	1	1

Rules for entities ISA



Rules for entities ISA



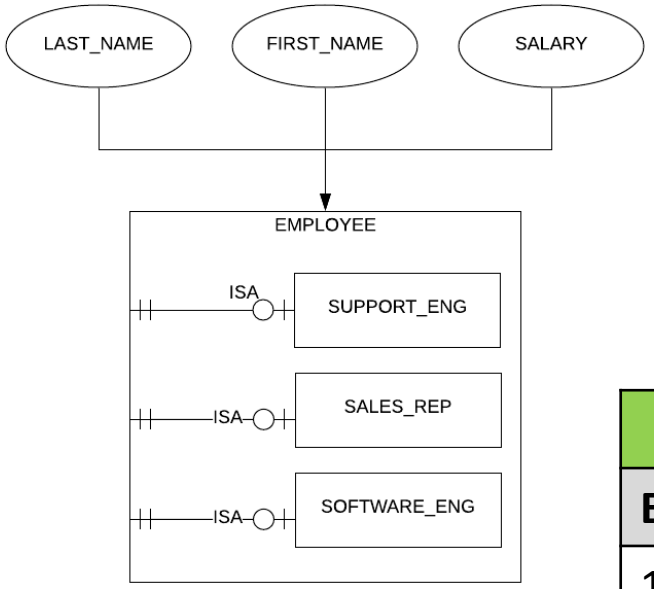
EMPLOYEES			
EMP_ID	**LAST_NAME**	**FIRST_NAME**	**SALARY**
1	Smith	John	2500
2	Grant	Anne	2700
3	Brown	Gregory	2300
…			

SUPPORT_ENG	
EMP_ID	**LEVEL**
1	3
…	…

SALES_REP	
EMP_ID	**TARGET**
2	25
…	…

SOFTWARE_ENG	
EMP_ID	**TEEM**
3	
…	…

Rules for entities ISA



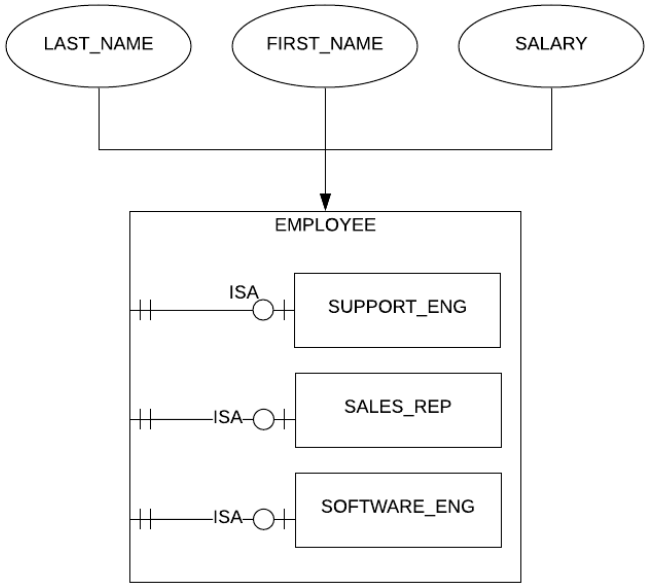
EMPLOYEES			
EMP_ID	**LAST_NAME**	**FIRST_NAME**	**SALARY**
1	Smith	John	2500
2	Grant	Anne	2700
3	Brown	Gregory	2300
…			

SUPPORT_ENG	
EMP_ID	**LEVEL**
1	3
…	…

SALES_REP	
EMP_ID	**TARGET**
2	25
…	…

SOFTWARE_ENG	
EMP_ID	**TEEM**
3	
…	…

Rules for entities ISA



SUPPORT_ENG				
EMP_ID	**LEVEL**	**LAST_NAME**	**FIRST_NAME**	**SALARY**
1	3	Smith	John	2500
…	…			

SALES_REP				
EMP_ID	**TARGET**	**LAST_NAME**	**FIRST_NAME**	**SALARY**
2	25	Grant	Anee	2700
…	…			

SOFTWARE_ENG				
EMP_ID	**TEEM**	**LAST_NAME**	**FIRST_NAME**	**SALARY**
3	3	Brown	Gregory	2300
…	…			

Rules for relationships

- 1 to 1 & 1 to M → foreign keys.
  - 1 (PK) to M (FK)
  - Usually, in 1 to 1 relationship, the FK is placed in the tables with fewer rows.
  - in 1 to many relationship, the PK is places on the ‘M’ side of relationship.

- M to M → associative table.
  - PK contains FKs and additional column.

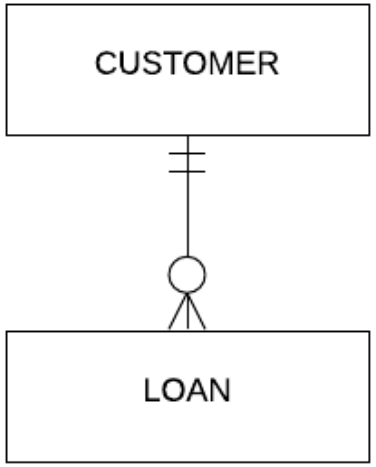
- Ternary relationships → associative table.
  - PK contains FKs and additional column.

One to One

ACCOUNT			
ACCOUNT_ID	**LAST_NAME**	**FIRST_NAME**	**DATE**
10	Snow	John	08/03/20
22	Grant	Anee	14/04/20
300	Brown	Gregory	01/05/20
…	…	…	…

CARD			
CARD_ID	**ACCOUNT_ID**	**CVN**	**DATE**
16897	10	125	18/04/21
24789	22	987	14/04/22
34597	300	875	03/05/21
…	…	…	…

One to Many

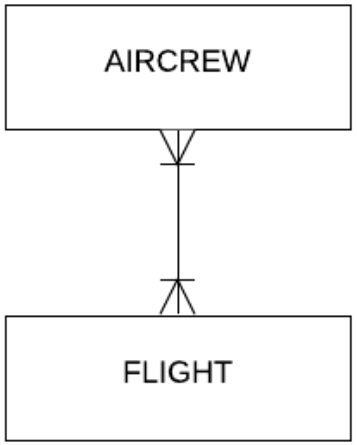


CUSTOMER			
CUSTOMER_ID	**LAST_NAME**	**FIRST_NAME**	**….**
10	Snow	John	….
22	Grant	Anee	….
300	Brown	Gregory	….
…	…	…	…

LOAN			
LOAN_ID	**CUSTOMER_ID**	**VALUES**	**DATE**
16897	10	125000	18/04/21
24789	22	987000	14/04/22
34597	300	87500	03/05/21
…	…	…	…

Many to Many

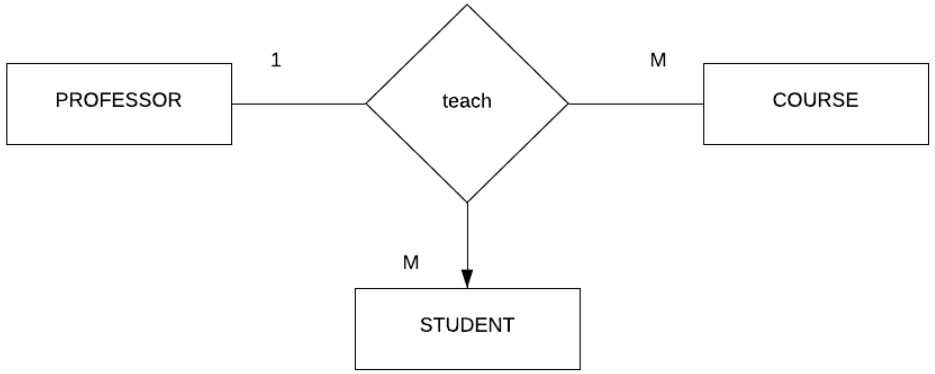
FLIGHT			
FLIGHT_ID	**DEP_AIRPORT**	**DATE**	**….**
1	Gatwick Airport	20/04/21	….
2	Grant	14/05/20	….
…	…	…	…



FLIGHT_CREW		
CREW_ID	**FLIGHT_ID**	**OBSERVATIONS**
10	1	…
22	1	…
10	2	…

AIRCREW			
CREW_ID	**LAST_NAME**	**FIRST_NAME**	**JOB_ID**
10	Snow	John	captain
22	Grant	Anee	first_officer
…	…	…	…

Ternary Relationships

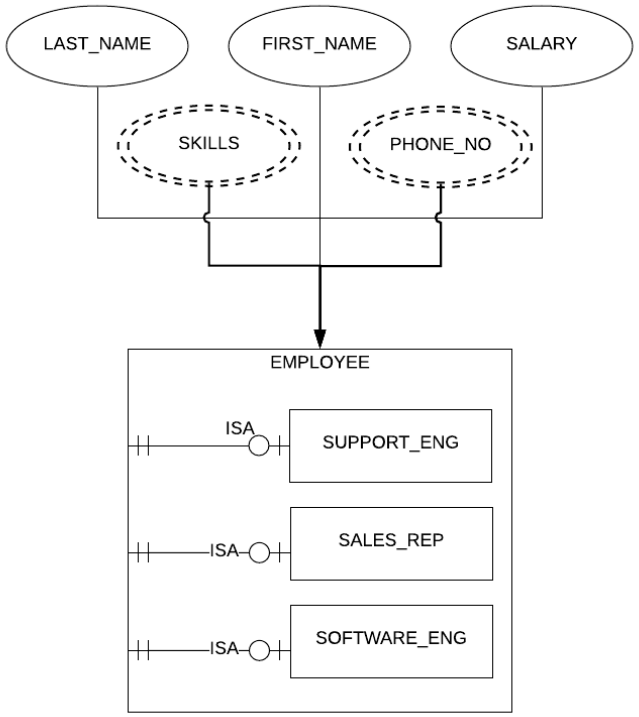


TEACH			
PROFESSOR_ID	**COURSE_ID**	**STUDENT_ID**	**GRADE**
1	BD	1001	9
1	SGBD	1002	10
1	BD	1002	8
2	TAP	1001	8
2	TAP	1002	10
2	AG	1001	5
….	….	….	….

Rules for attributes

- Simple attribute → column
- Multivalued attributes → weak entity → table
  → set of columns

Rules for entities ISA



EMPLOYEES					
EMP_ID	**LAST_NAME**	**FIRST_NAME**	**SALARY**	**PHONE1**	**PHONE2**
1	Smith	John	2500	0745…	0720…
2	Grant	Anne	2700	07497…	NULL
3	Brown	Gregory	2300	NULL	07458..
…	…	…	…	…	…

EMP_SKILL		
EMP_ID	**SKILL**	**LEVEL**
1	Python	3
1	C++	2
1	NoSql	3
2	SQL	1

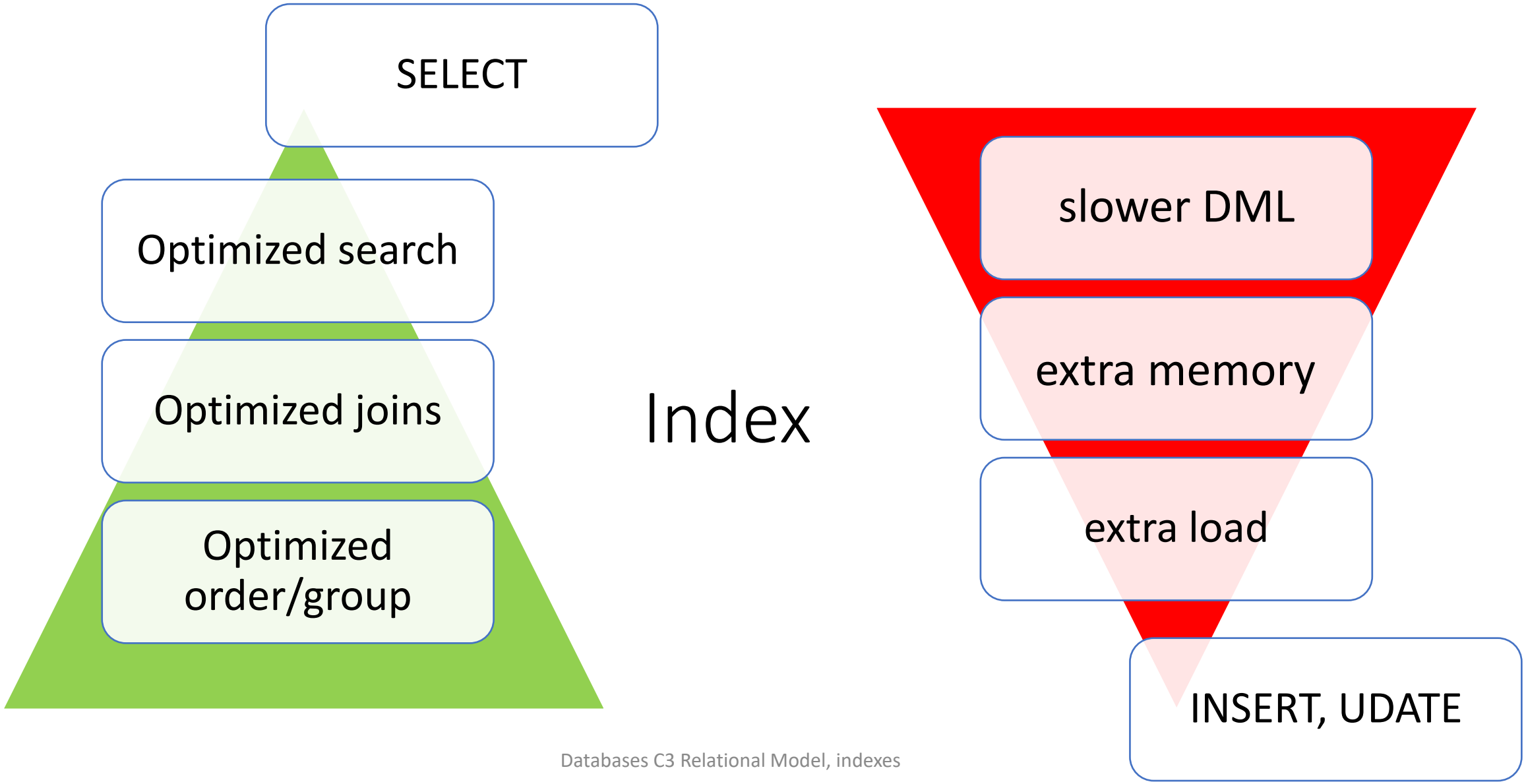
Indexes

Indexes

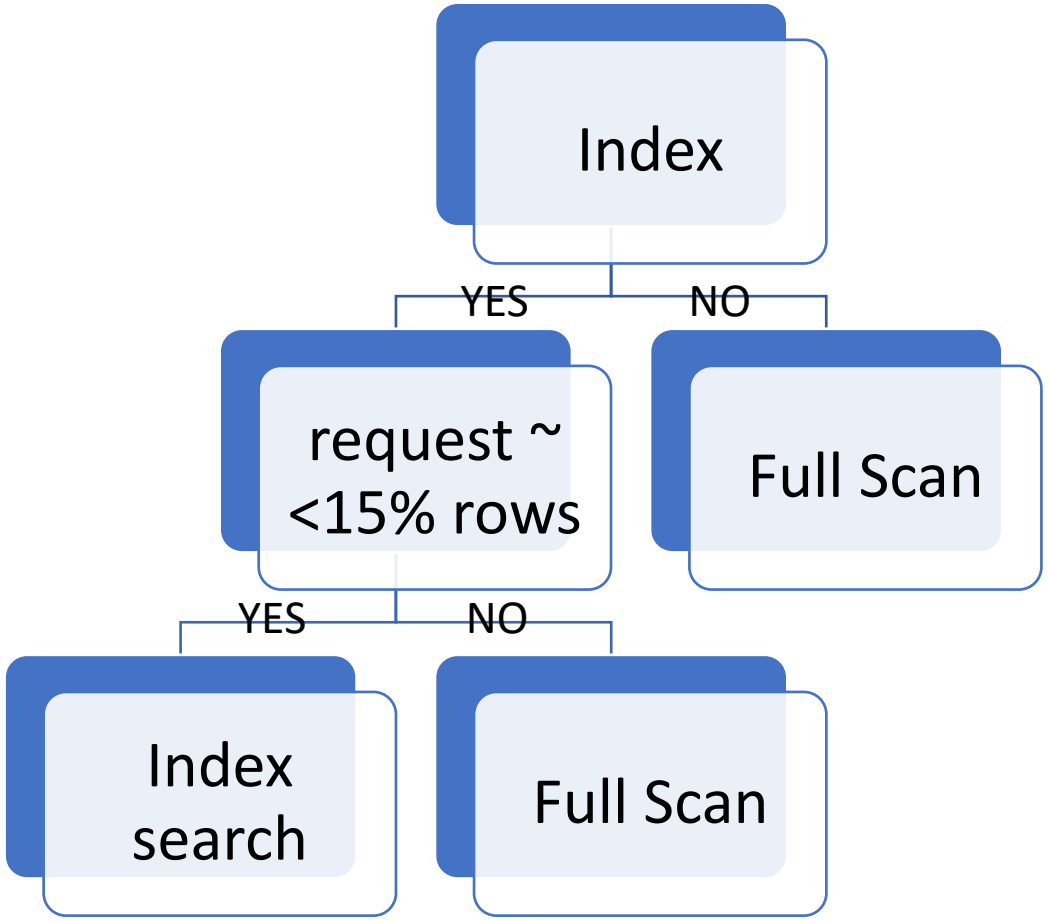
- Maps search key to data using specific data structures.

- Optimized search.
- Optimized joins (lookup in more than one table)
- Optimized order/group

- slower DML (insert and update operations).
- extra memory


SELECT
Optimized search
Optimized joins
Optimized order/group
Index
slower DML
extra memory
extra load
INSERT, UDATE

Sql Optimizer



Autogenerated columns

- MySQL auto-generated index (key):
  - DB_ROW_ID increases monotonically as new rows are inserted.
  - DB_ROLL_PTR roll pointer, points to log record.
  - DB_TRX_ID last transaction that updated or inserted the row.

- Oracle rowid:
  - Pseudo column 18 characters = 10 + 4 + 4 (block, row, file).
  - Store and return row address in hexadecimal format (string).
  - Unique identifier for each row.
  - Immutable.

Autogenerated columns

- Oracle rowid:
  - Used in where clause to select/update/delete a row.
- Oracle rownum:
  - Sequential number in which oracle has fetched the row, before ordering the result
  - Temporary generated along with a select statement.
- Mongo
  - ObjectID (timestamp 4Bytes + random 5Bytes + Count 3Bytes. (GUID)

Index

- Data structure that optimizes search.
- Automatically created when a PK/unique constraint is defined.

Primay key

- Constraint imposed on insert/update behavior.
- NotNull & Unique.

MySQL

```
SHOW EXTENDED INDEX FROM with_index;
```

MySQL

```
select * from  information_schema.statistics
where table_name = 'with_index'
        and index_name = 'primary';
```

Oracle

```
select * from user_indexes
where table_name = 'WITH_INDEX';
```

Oracle

```
select * from user_constraints
where table_name = 'WITH_INDEX';
```

Index types

Clustered index (SqlServer, MySql)

- Defines the order in which data is physically stored in a table. (index on column semester)
- Only one clustered index on a table (data can be stored in only one order)
- A cluster index is created automatically when a primary key is defined.
- *No second data structure for the table.*

- Oracle: IOT index organized tables. Table is stored in a B-tree structure. (key and non-keys column are stored in leafs)

Clustered index (SqlServer, MySql)

- Indexes that specify a different order from the sequential order of the file are non-clustering indexes or secondary indexes.

- Index entry:

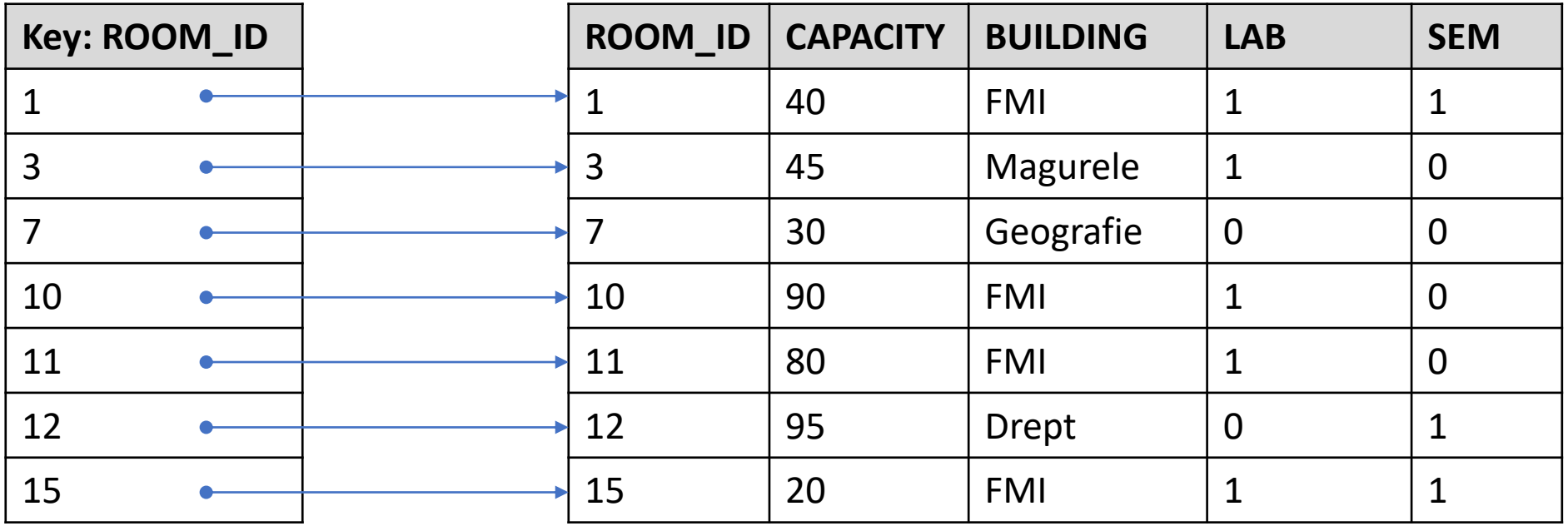
  search key value + pointers to records containing search key value.

  Sparse indices: contains only some key values. Works only for clustering indices. If we want to find all records with search key value *v,* we located the first row with v, then we check the records in order, until key search is different than *v.*

  Dense indices: contains a pair (key_value, first_record_with_value) for each possible search key value (clustering indices).

Clustered index (SqlServer, MySql)

Dense indices

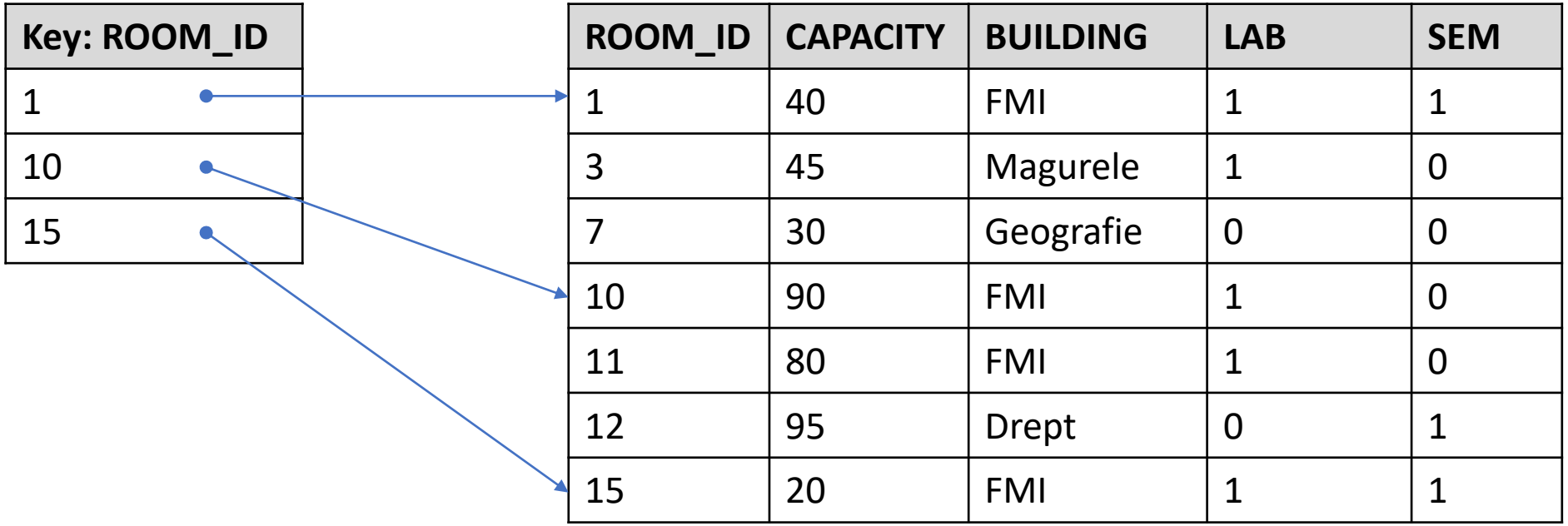


Key: ROOM_ID
1
3
7
10
11
12
15

ROOM_ID	CAPACITY	BUILDING	LAB	SEM
1	40	FMI	1	1
3	45	Magurele	1	0
7	30	Geografie	0	0
10	90	FMI	1	0
11	80	FMI	1	0
12	95	Drept	0	1
15	20	FMI	1	1

Clustered index (SqlServer, MySql)

Sparse indices. How does find(12) works?



Key: ROOM_ID
1
10
15

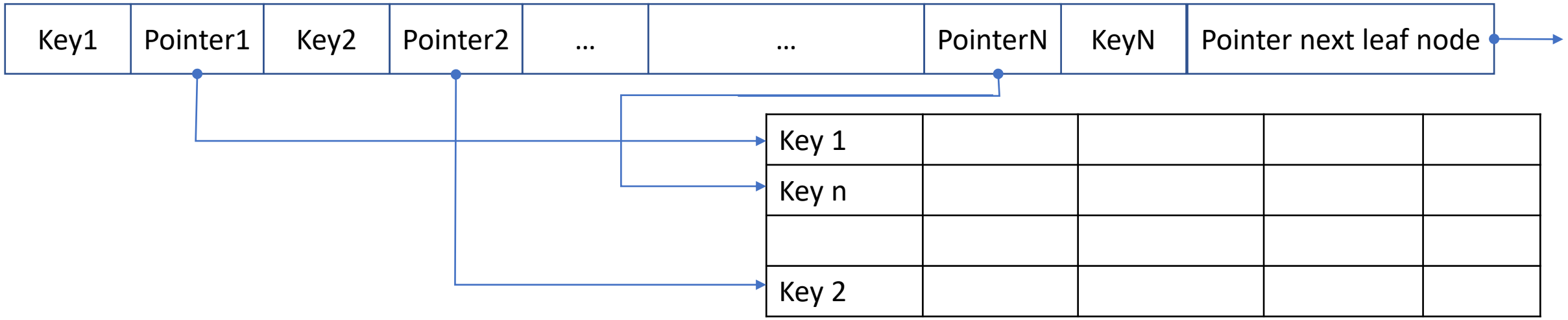
ROOM_ID	CAPACITY	BUILDING	LAB	SEM
1	40	FMI	1	1
3	45	Magurele	1	0
7	30	Geografie	0	0
10	90	FMI	1	0
11	80	FMI	1	0
12	95	Drept	0	1
15	20	FMI	1	1

B – Tree

- B -- Balanced tree.
- Default index type in Oracle.
- Two types of nodes: branch blocks and leaf blocks.
- Branch blocks pointers to lower levels.
- Leaf blocks contain rowids/physical address.
- The number of blocks traversed in order to reach a leaf block is the same for each leaf block.

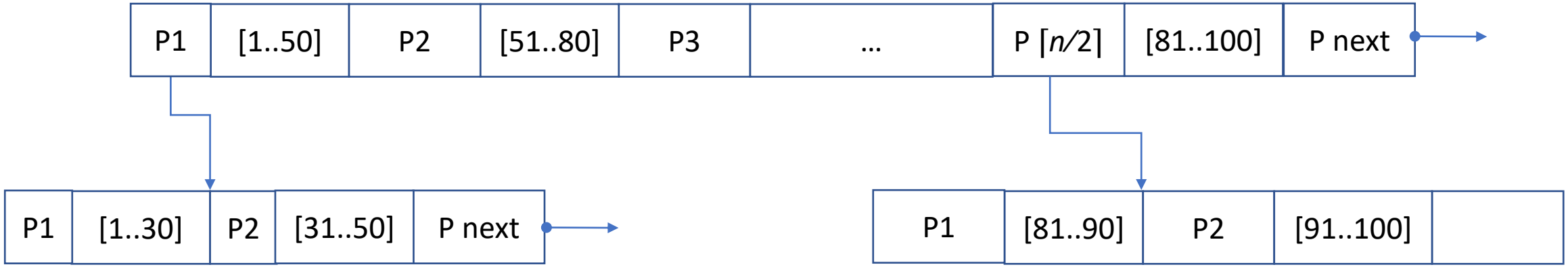
B – Tree

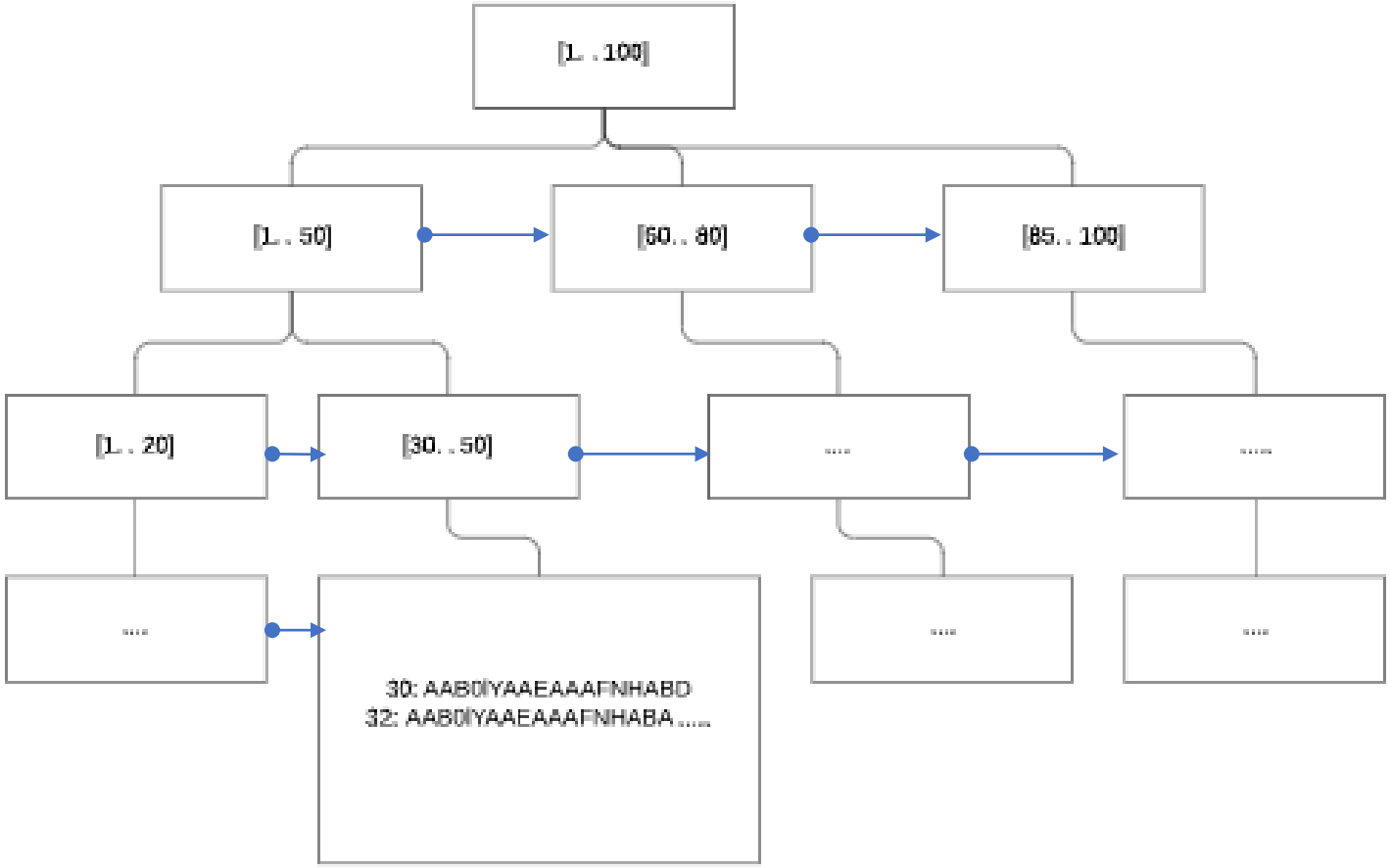
- create index idx_emp_id on employees(employee_id).
  - Divide employee_id values in sorted ranges.
  - Leaves nodes store rowid
- Leaf node:



B – Tree

- Non-leaf node: keys --> intervals; pointers --> pointers to tree nodes.
- A non-leaf node may hold between $\lceil n/2 \rceil$ and n pointers.




[1. . 100]
[1. . 50]
[50. . 80]
[85. . 100]
[1. . 20]
[30. . 50]
....
.....
....
30: AABDIYAAEAAAFNHABD
32: AABDIYAAEAAAFNHABA.....
....
....

Reverse index

- B – tree where keys are in reverse order. Key 4573 is stored 3754.
- Optimized insert operations.
- Key 4573 will be stored in the same block with key 9573
  while 4574 will be stored in a different block.

Bitmap index

- Used for columns with limited number of domain values, and with limited number of records.
- Example: language proficiency levels (en or fr)

emp_id	en_level	fr_level
0	A1	B1
1	A2	B2
2	C1	A1
3	A1	B1
4	A1	C2

row_id	A1	A2	B1	B2	C1	C2
AAB0lYAAEAAAFNHABD	1	0	0	0	0	0
AAB0lYAAEAAAFNHABV	0	1	0	0	0	0
AAB0lYAAEAAAFNHABX	0	0	0	0	1	0
AAB0lYAAEAAAFNHAAv	1	0	0	0	0	0
AAB0lYAAEAAAFNHAAV	1	0	0	0	0	0

Bitmap index

- A bitmap index on an attribute *A* of relation *R* consists of one bitmap for each value in the domain of A. For attribute en_level we have 6 possible values: A1, A2, B1, B2, C1, C2, hence 6 bitmaps.

emp_id	en_level	fr_level
0	A1	B1
1	A2	B2
2	C1	A1
3	A1	B1
4	A1	C2

row_id	A1	A2	B1	B2	C1	C2
AAB0lYAAEAAAFNHABD	1	0	0	0	0	0
AAB0lYAAEAAAFNHABV	0	1	0	0	0	0
AAB0lYAAEAAAFNHABX	0	0	0	0	1	0
AAB0lYAAEAAAFNHAAv	1	0	0	0	0	0
AAB0lYAAEAAAFNHAAV	1	0	0	0	0	0

Bitmap index

- For bitmap indices to be used, records in a relation must be numbered sequentially.
- Bitmap en_level_A1: 10011 fr_level_B1: 10010

emp_id	en_level	fr_level
1	A1	B1
2	A2	B2
3	C1	A1
4	A1	B1
5	A1	C2

row_id	A1	A2	B1	B2	C1	C2
AAB0lYAAEAAAFNHABD	1	0	0	0	0	0
AAB0lYAAEAAAFNHABV	0	1	0	0	0	0
AAB0lYAAEAAAFNHABX	0	0	0	0	1	0
AAB0lYAAEAAAFNHAAv	1	0	0	0	0	0
AAB0lYAAEAAAFNHAAV	1	0	0	0	0	0

Bitmap index

- Used in queries involving or/and conditions
  - select emp_id from emp where en_level = ‘A1’ and fr_level = ‘B1’
- Bitmap en_level_A1: 10011 fr_level_B1: 10010 10011 & 10010

emp_id	en_level	fr_level
1	A1	B1
2	A2	B2
3	C1	A1
4	A1	B1
5	A1	C2

row_id	A1	A2	B1	B2	C1	C2
AAB0lYAAEAAAFNHABD	1	0	0	0	0	0
AAB0lYAAEAAAFNHABV	0	1	0	0	0	0
AAB0lYAAEAAAFNHABX	0	0	0	0	1	0
AAB0lYAAEAAAFNHAAv	1	0	0	0	0	0
AAB0lYAAEAAAFNHAAV	1	0	0	0	0	0

Bitmap index

- Used in queries that count the number of records satisfying A = val
  - select count(emp_id) from emp where en_level = ‘A1’
- Bitmap en_level_A1: 10011 10011 -- count 3

emp_id	en_level	fr_level
1	A1	B1
2	A2	B2
3	C1	A1
4	A1	B1
5	A1	C2

row_id	A1	A2	B1	B2	C1	C2
AAB0lYAAEAAAFNHABD	1	0	0	0	0	0
AAB0lYAAEAAAFNHABV	0	1	0	0	0	0
AAB0lYAAEAAAFNHABX	0	0	0	0	1	0
AAB0lYAAEAAAFNHAAv	1	0	0	0	0	0
AAB0lYAAEAAAFNHAAV	1	0	0	0	0	0