

Relational Model

Data Model



- **Data Model** is a collection of concepts that can be used to describe the structure of a database
- **Structure of a database** we mean the data types, relationships, and constraints that apply to the data
- Data models also include a set of basic operations for specifying retrievals and updates on the database

Categories of Data Models



1. **High-level or conceptual data models** provide concepts that are close to the way many users perceive data (Also called **entity-based** or **object-based** or **semantic** data models)
2. **Low-level or physical or internal data models** provide concepts that describe the details of how data is stored on the computer storage media, typically magnetic disks
 - Not for end users but for specialists
3. **Implementation (representational) data models:** Provide concepts that fall between the above two, balancing user views with some computer storage details.

Entity, Attribute, Relationships



- An **entity** represents a real-world object or concept, such as an employee or a project from the miniworld that is described in the database
- An **attribute** represents some property of interest that further describes an entity, such as the employee's name or salary.
- A **relationship** among two or more entities represents an association among the entities, for example, a works-on relationship between an employee and a project.

Schemas, Instances and Database State



- **Database Schema:** Description of a database, includes descriptions of the database structure; the constraints that should hold on the database and is not expected to change frequently.
- **Schema Diagram:** A diagrammatic display of (some aspects of) a database schema
- A schema diagram displays only some aspects of a schema, such as the names of record types and data items, and some types of constraints

Schemas, Instances and Database State



STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

Schemas, Instances and Database State



- **Schema Construct:** A component of the schema or an object within the schema, e.g., STUDENT, COURSE
- **Database Instance:** The actual data stored in a database at a *particular moment in time*, also called **database state (or occurrence)**.
- **Initial Database State:** Refers to the DB when it is loaded or populated with first the initial data.
- **Valid State:** A state that satisfies the structure and constraints of the database specified in the schema.
- Schema is also called **intension**, whereas state is called **extension**

Three-Schema Architecture

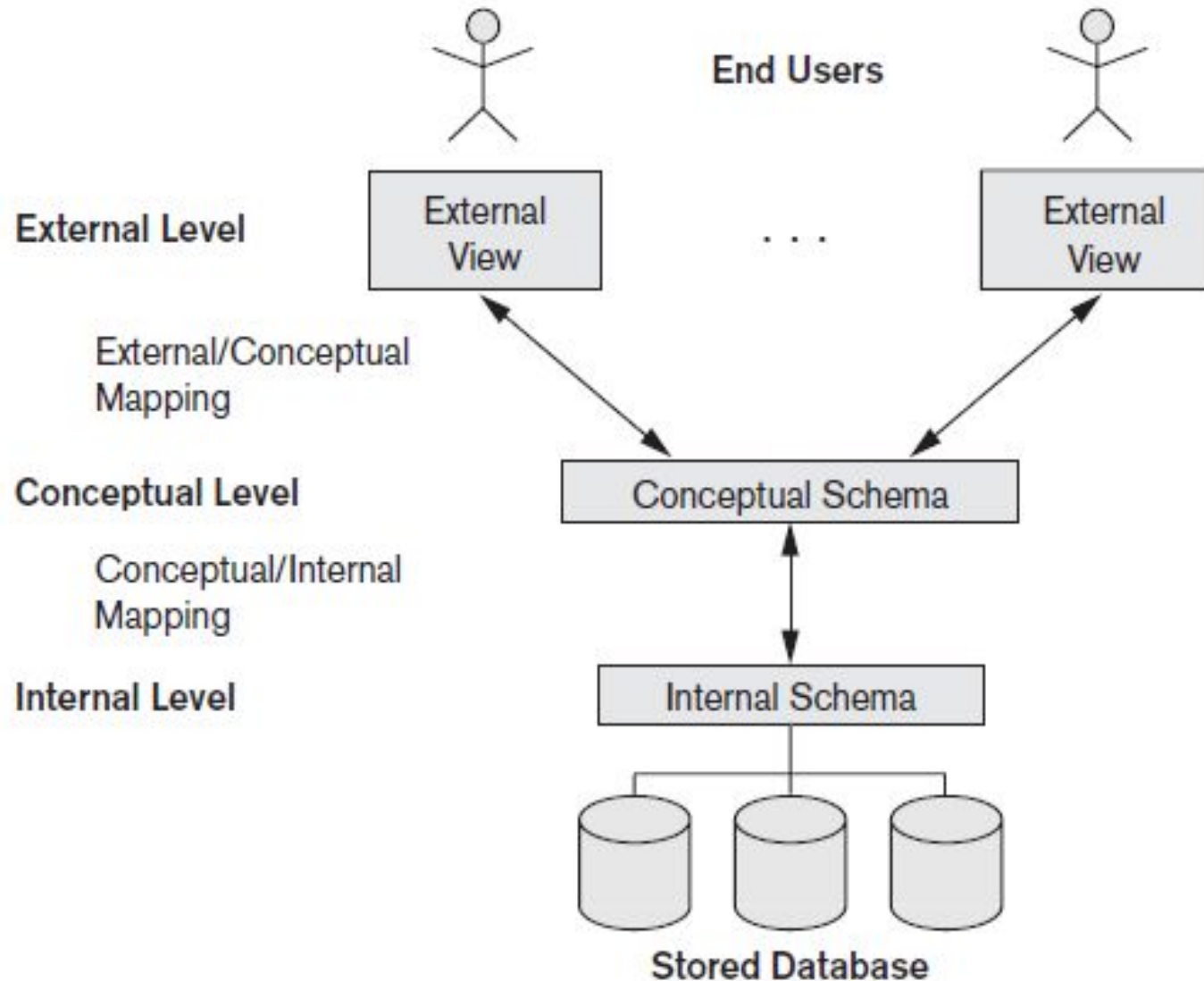
- Goal of the three-schema architecture is to separate the user applications from the physical database.
- Proposed to support DBMS characteristics of:
 - Program-data independence.
 - Support of multiple views of the data
- Three levels:
 1. **Internal level** has an **internal schema** that describes the physical storage structure of the database.
 - Uses a physical data model and describes the complete details of data storage and access paths for the database.

Three-Schema Architecture



2. **Conceptual level** has a **conceptual schema**, which describes the structure of the whole database for a community of users.
 - Hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints
3. **External or view level** includes a number of external schemas or user views.
 - Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

Three – Schema Architecture



Database Languages



- **Data Definition Language(DDL)**, is used by DBA and database designers to define both conceptual and external schemas
- **Storage definition language (SDL)**, is used to specify the internal schema
- **View definition language (VDL)**, to specify user views and their mappings to the conceptual schema
- **Data manipulation language (DML)**, allows the users to manipulate the database by providing the set of operations or languages

Structure of Relational Databases



- A relational database consists of a collection of tables, each of which is assigned a unique name
- A **tuple** is simply a sequence (or list) of values
- **Relation instance** is a specific instance of a relation, i.e., containing a specific set of rows
- For each attribute of a relation, there is a set of permitted values, called the **domain** of that attribute.
- A domain is **atomic** if elements of that domain are considered to be invisible units.
- The **null value** is a special value that signifies that the value is unknown or does not exist.

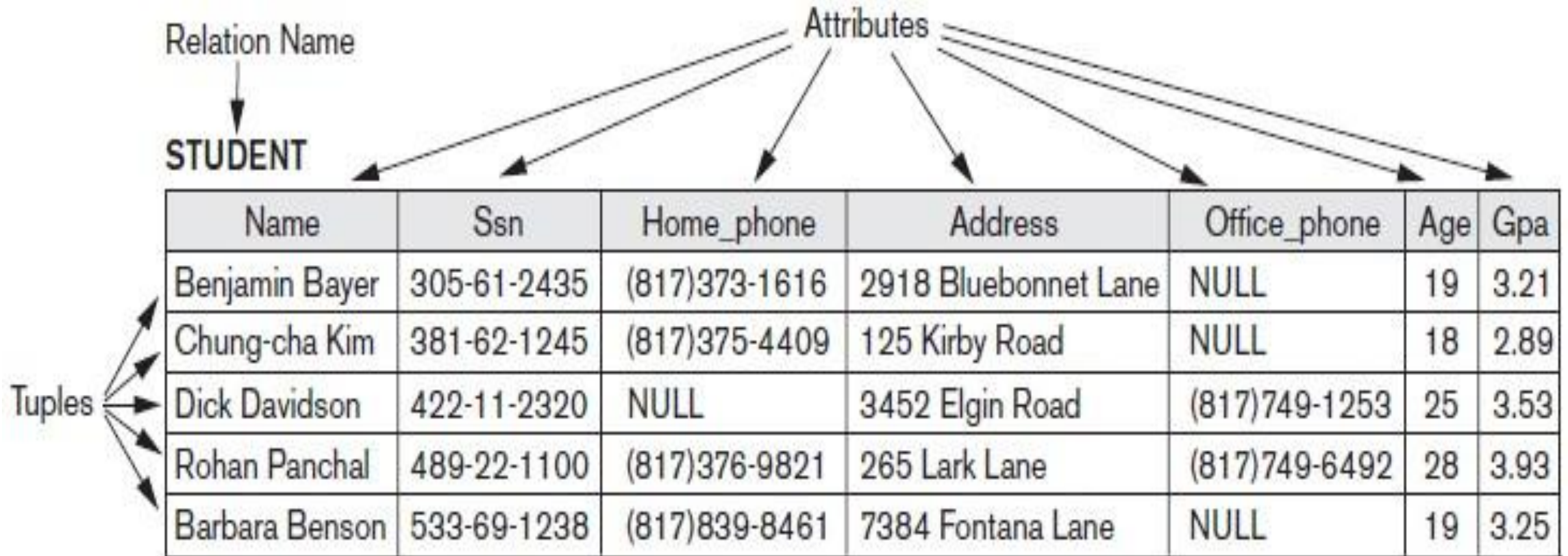
Structure of Relational Databases



- Therefore, a relation (or relation state) r of the relation schema $R(A_1, A_2, \dots, A_n)$, also denoted by $r(R)$, is a set of n -tuples $\mathbf{r} = \{t_1, t_2, \dots, t_m\}$.
- Each n -tuple t is an ordered list of n values $t = \langle v_1, v_2, \dots, v_n \rangle$, where each value v_i , $1 \leq i \leq n$, is an element of $\text{dom}(A_i)$ or is a special NULL value.

STUDENT(Name, Ssn, Home phone, Address, Office phone, Age, Gpa)

Structure of Relational Databases



- There must be a way to specify how tuples within a given relation are distinguished, expressed in terms of the attributes.
- The attribute values of a tuple must be such that they can uniquely identify the tuple i.e., no two tuples in a relation are allowed to have exactly the same value for all attributes.
- A **superkey** is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation.
- If K is a superkey, then so is any superset of K .
- A **key** K of a relation schema R is a superkey of R with the additional property that removing any attribute A from K leaves a set of attributes K' that is not a superkey of R anymore

Keys



- A key satisfies two properties:
 - Two distinct tuples in any state of the relation cannot have identical values for (all) the attributes in the key.
 - It is a **minimal superkey/candidate key** - that is, a superkey from which we cannot remove any attributes and still have the uniqueness constraint in condition 1.
- **STUDENT(Name, Ssn, Home phone, Address, Office phone, Age, Gpa)**
- For the given relation STUDENT, {Ssn} is the key; {Ssn, Name, Age} - is a superkey but it is not a minimal superkey!
- Any superkey formed from a single attribute is also a key whereas a key with multiple attributes must require all its attributes together to have the uniqueness property.

Keys

- There is a possibility of having more than one candidate key for a relation
- Term **primary key** is used to denote the chosen candidate key
- Primary key attributes are also underlined
- Other candidate keys are designated as **unique keys**

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

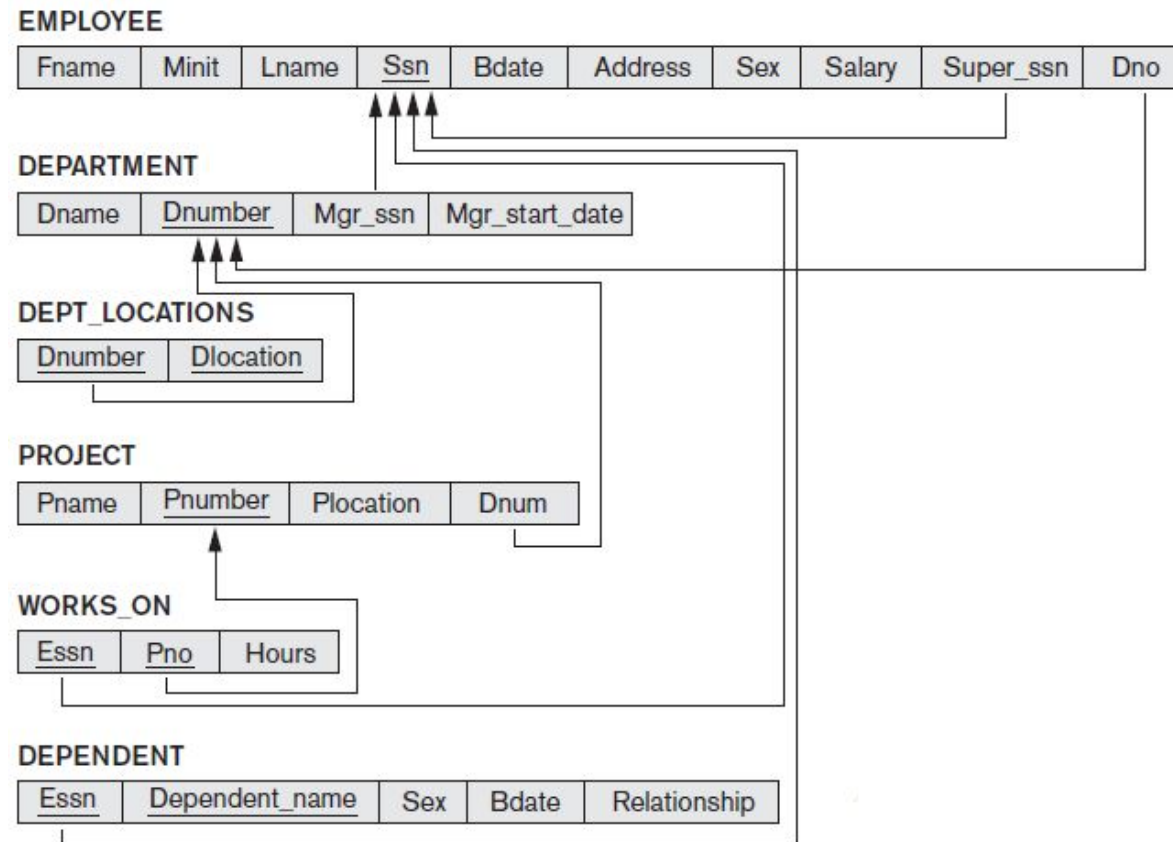
WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

- A relation, say r_1 , may include among its attributes the primary key of another relation, say r_2 , and this attribute is called **foreign key** from r_1 , referencing r_2 .
- Relation r_1 is also called the **referencing relation** of the foreign key dependency, and r_2 is called the **referenced relation** of the foreign key
- **Referential integrity constraint** requires that the values appearing in specified attributes of any tuple in the referencing relation also appear in specified attributes of at least one tuple in the referenced relation or it is NULL

Schema Diagram

- A database schema, along with primary key and foreign key dependencies, can be depicted by schema diagrams

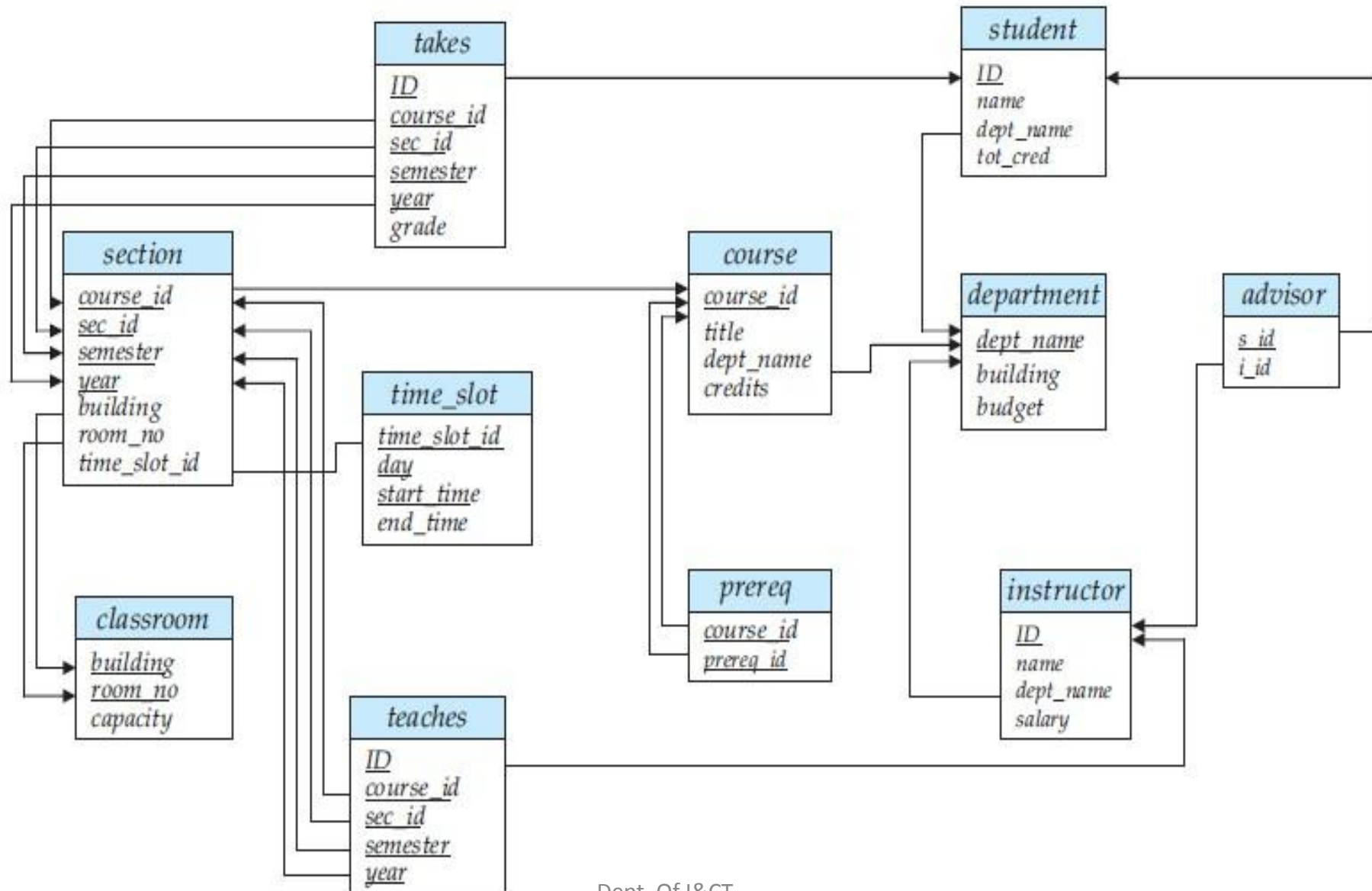


Schema Diagram



classroom(building, room_number, capacity)
department(dept_name, building, budget)
course(course_id, title, dept_name, credits)
instructor(ID, name, dept_name, salary)
section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
teaches(ID, course_id, sec_id, semester, year)
student(ID, name, dept_name, tot_cred)
takes(ID, course_id, sec_id, semester, year, grade)
advisor(s_ID, i_ID)
time_slot(time_slot_id, day, start_time, end_time)
prereq(course_id, prereq_id)

Schema Diagram



Relational Query Languages



- A **query language** is a language in which a **user requests information from the database**.
- Categorized as procedural or non procedural
- **Procedural:** The user instructs the system to perform a **sequence of operations** on the database to compute the desired result.
- **Nonprocedural:** user describes the desired information without giving a specific procedure for obtaining that information.

- In a procedural query language, like Relational Algebra, you write a query as an expression consisting of relations and Algebra Operators, like join, cross product, projection, restriction, etc.
- On the contrary, query SQL are called “non procedural” since they express the expected result only through its properties, and not the order of the operators to be performed to produce it



Relational Algebra

- Created by Edgar F Codd at IBM in 1970
- Procedural language
- Six basic operators
 - select: σ
 - project: Π
 - union: \cup
 - set difference: $-$
 - Cartesian product: \times
 - rename: ρ
- The operators take one or two relations as inputs and produce a new relation as a result



Select Operation

- Notation: $\sigma_p(r)$
- p is called the **selection predicate**
- Defined as:

$$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$$

where p is a formula in propositional calculus consisting of **terms** connected by : \wedge (and), \vee (or), \neg (not)

Each **term** is one of:

$\langle \text{attribute} \rangle \quad op \quad \langle \text{attribute} \rangle$ or $\langle \text{constant} \rangle$

where op is one of: $=, \neq, >, \geq, <, \leq$

- Example of selection:

$$\sigma_{dept_name="Physics"}(instructor)$$

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

A	B	C	D
α	α	1	7
β	β	23	10

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



Project Operation

PPD

- Notation:

$$\Pi_{A_1, A_2, \dots, A_k}(r)$$

where A_1, A_2 are attribute names and r is a relation name

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the *dept_name* attribute of *instructor*

$$\Pi_{ID, name, salary}(instructor)$$

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

A	C
α	1
α	1
β	1
β	2

$$=$$

A	C
α	1
β	1
β	2

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

Union Operation

- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$
- For $r \cup s$ to be valid.
 - r, s must have the same **arity** (same number of attributes)
 - The attribute domains must be **compatible** (example: 2nd column of r deals with the same type of values as does the 2nd column of s)
- Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

$$\Pi_{\text{course_id}}(\sigma_{\text{semester}=\text{"Fall"} \wedge \text{year}=2009}(\text{section})) \cup$$

$$\Pi_{\text{course_id}}(\sigma_{\text{semester}=\text{"Spring"} \wedge \text{year}=2010}(\text{section}))$$

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
α	2
β	1
β	3

 $r \cup s$

Set Difference Operation

- Notation $r - s$

- Defined as:

$$r - s = \{t \mid t \in r \text{ and } t \notin s\}$$

- Set differences must be taken between **compatible** relations
 - r and s must have the **same** arity
 - attribute domains of r and s must be compatible
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\Pi_{\text{course_id}}(\sigma_{\text{semester}=\text{"Fall"} \wedge \text{year}=2009}(\text{section})) -$$

$$\Pi_{\text{course_id}}(\sigma_{\text{semester}=\text{"Spring"} \wedge \text{year}=2010}(\text{section}))$$

A	B
α	1
α	2
β	1

 r

A	B
α	2
β	3

 s

A	B
α	1
β	1

 $r - s$

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:

$$r \cap s = \{t \mid t \in r \text{ and } t \in s\}$$
- Assume:
 - r, s have the *same arity*
 - attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	2

$r \cap s$

Cartesian-Product Operation

- Notation $r \times s$
- Defined as:

$$r \times s = \{t \mid t \in r \text{ and } t \in s\}$$

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- Assume that attributes of $r(R)$ and $s(S)$ are disjoint. (That is, $R \cap S = \emptyset$)
- If attributes of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

$r \times s$

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- Example:

$$\rho_X(E)$$

returns the expression E under the name X

- If a relational-algebra expression E has arity n , then

$$\rho_X(A_1, A_2, \dots, A_n)(E)$$

returns the result of expression E under the name X , and with the attributes renamed to A_1, A_2, \dots, A_n .

Natural Join- Example

- The natural join operation on two relations matches tuples whose values are the same on **all attribute names that are common to both relations**.

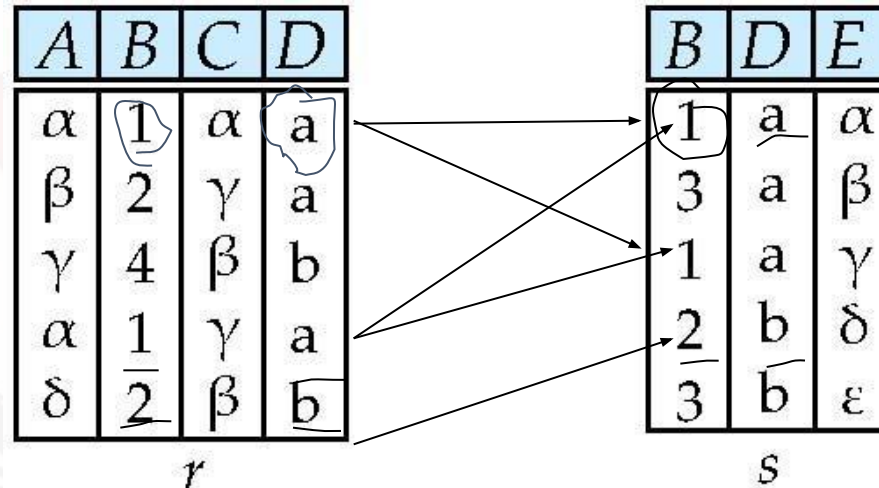
- Relations r , s :

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

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

s



- Natural Join

- $r \bowtie s$

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

- JOIN operation
- Joins two relation instructor and department on a COMMON attribute

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
12121	Wu	90000	Finance	Painter	120000
15151	Mozart	40000	Music	Packard	80000
22222	Einstein	95000	Physics	Watson	70000
32343	El Said	60000	History	Painter	50000
33456	Gold	87000	Physics	Watson	70000
45565	Katz	75000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
76543	Singh	80000	Finance	Painter	120000
76766	Crick	72000	Biology	Watson	90000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000

Exercise for students

Player relation

Player Id	Team Id	Country	Age	Runs	Wickets
1001	101	India	25	10000	300
1004	101	India	28	20000	200
1006	101	India	22	15000	150
1005	101	India	21	12000	400
1008	101	India	22	15000	150
1009	103	England	24	6000	90
1010	104	Australia	35	1300	0

1. Retrieve play all the tuples for which runs are greater than or equal to 15000.
2. Retrieve all the player Id whose runs are greater than or equal to 6000 and age is less than 25
3. List all the countries in Player relation.