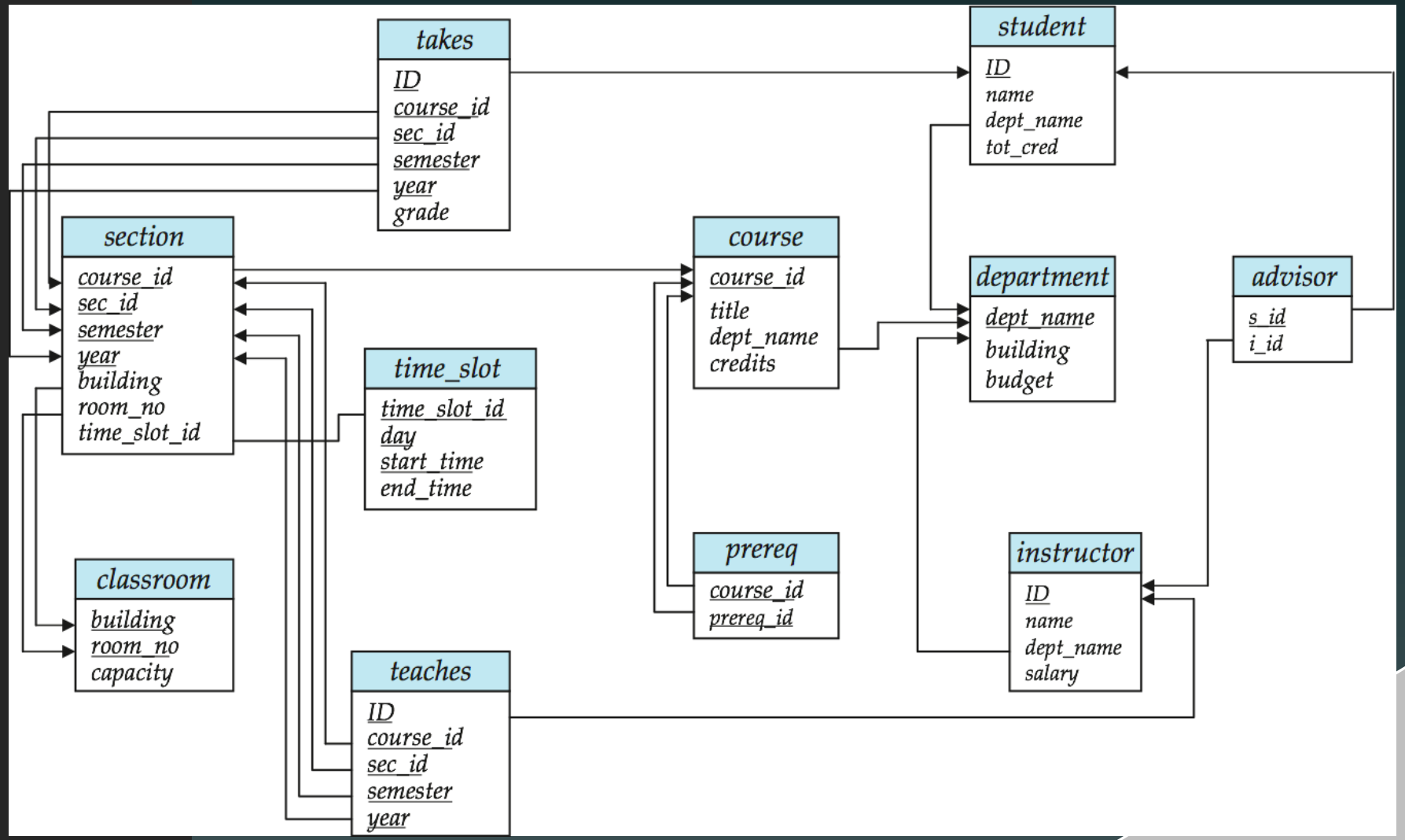
IF2240 – Basis Data

Formal Relational Query Language (2)

KNOWLEDGE & SOFTWARE ENGINEERING

Schema Diagram for University Database



Relational Algebra

Procedural language

Six basic operators

- select: σ
- project: Π
- union: $\cup$
- set difference: $-$
- cartessian product: $\times$
- rename: ρ

The operators take one or two relations as inputs and produce a new relation as a result.

- Additional Operators
  - Intersection: $\cap$
  - natural join: $\bowtie$
  - assignment: $\leftarrow$
  - outer join: $⟗$
  - division: $\div$
- Extended Operators
  - generalized projection
  - aggregation

Additional Operators

Additional Operations

We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

- Set intersection
- Natural join
- Assignment
- Outer join
- Division

Set-Intersection Operation

Notation

$$r \cap s$$

Defined as

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

Requirement

Set intersection must be taken between *compatible* relations.

Set-Intersection Operation – Example 1

RELATIONS

R

A	B
α	1
α	2
β	1

S

A	B
α	2
β	3

R ∩ S:

A	B
α	2

Set-Intersection Operation

Define set-intersection by using basic operators

$$r \cap s = r - (r - s)$$

Natural-Join Operation

Notation

$$r \bowtie s$$

Defined as

- Let r and s be relations on schemas R and S respectively.
  Then, $r \bowtie s$ is a relation on schema $R \cup S$ obtained as follows:
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result (concatenation of t_r and t_s)

Natural Join Operation – Example 1

RELATIONS

R

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

S

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

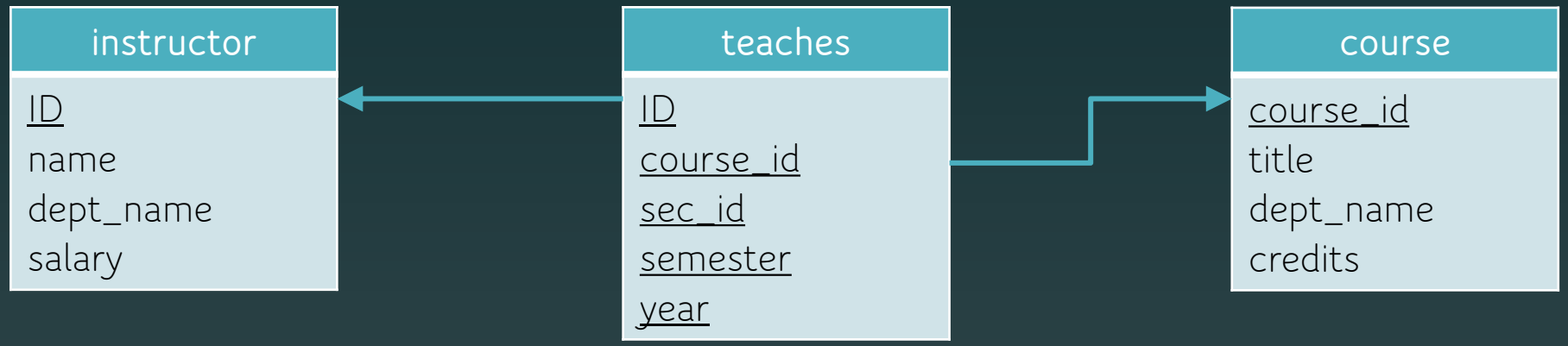
R ⋈ S

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

Note: natural join can be expressed using basic operations.
e.g. $r \bowtie s$ can be written as

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

Natural Join Operation – Example 2



Find the names of all instructors in the Comp. Sci. department together with the course titles of all the courses that the instructors teach

$$\Pi_{name,\ title}\ (\sigma_{dept\_name=\text{“Comp. Sci.”}}\ (instructor \bowtie teaches \bowtie course))$$

$$\Pi_{name,\ title}\ (\sigma_{dept\_name=\text{“Comp. Sci.”}}\ (instructor \bowtie teaches \bowtie \Pi_{course\_id,\ title}\ (course)))$$

Natural Join and Theta Join

Natural join is associative

$(instructor \bowtie teaches) \bowtie course$ is equivalent to
$instructor \bowtie (teaches \bowtie course)$

Natural join is commutative

$instructor \bowtie teaches$ is equivalent to
$teaches \bowtie instructor$

The **theta join** operation

$r \bowtie_\theta s = \sigma_\theta (r \times s)$

Assignment Operation

The assignment operation (←) provides a convenient way to express complex queries.

- Write query as a sequential program consisting of
  - a series of assignments
  - followed by an expression whose value is displayed as a result of the query.
- Assignment must always be made to a temporary relation variable.

Example: Find all instructor in the "Physics" and "Music" department.

$Physics \leftarrow \sigma_{dept\_name="Physics"}(instructor)$

$Music \leftarrow \sigma_{dept\_name="Music"}(instructor)$

$Physics \cup Music$

Outer Join

Notation:

- Left outer join: ⟕
- Right outer join: ⟖
- Full outer join: ⟗

Defined as

- Computes the join and then adds tuples from one relation that does not match tuples in the other relation to the result of the join.

Uses *null* values

- *null* signifies that the value is unknown or does not exist
- All comparisons involving *null* are (roughly speaking) **false** by definition.

Outer Join Operation

Express outer join (e.g. r ⟖ s) using basic [and join – to simplify] operations

$$(r \bowtie s) \cup ((r - \prod_R (r \bowtie s)) \times \{(null, \ldots, null)\})$$

Null Values

It is possible for tuples to have a null value, denoted by *null*, for some of their attributes

null signifies an unknown value or that a value does not exist.

The result of any arithmetic expression involving *null* is *null*.

Aggregate functions simply ignore null values (as in SQL)

For duplicate elimination and grouping, null is treated like any other value, and two nulls are assumed to be the same (as in SQL)

Null Values

Comparisons with null values return the special truth value: *unknown*

- If *false* was used instead of *unknown*, then *not (A < 5)*
  would not be equivalent to *A >= 5*

Three-valued logic using the truth value *unknown*:

- OR: (*unknown* **or** *true*) = *true*,
  (*unknown* **or** *false*) = *unknown*
  (*unknown* **or** *unknown*) = *unknown*
- AND: (*true* **and** *unknown*) = *unknown*,
  (*false* **and** *unknown*) = *false*,
  (*unknown* **and** *unknown*) = *unknown*
- NOT: (**not** *unknown*) = *unknown*
- In SQL "*P* **is unknown**" evaluates to true if predicate *P* evaluates to *unknown*

Result of select predicate is treated as *false* if it evaluates to *unknown*

Division Operation

Notation

$$r \div s$$

Defined as

Given relations r(R) and s(S), such that $S \subset R$,
r ÷ s is the largest relation t(R-S) such that $\mathbf{t \times s \subseteq r}$

E.g.

let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and
$s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name="Biology"}(course))$
then **r ÷ s** = students who have taken all courses in the Biology Dept.

Division Operation

r

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ε	6
ε	1
β	2

s

B
1
2

$r \div s$

A
α
β

Express r $\div s$ using basic operations

$temp1 \leftarrow \prod_{R-S} (r)$

$temp2 \leftarrow \prod_{R-S} ((temp1 \times s) - \prod_{R-S,S} (r))$

$temp1 - temp2$