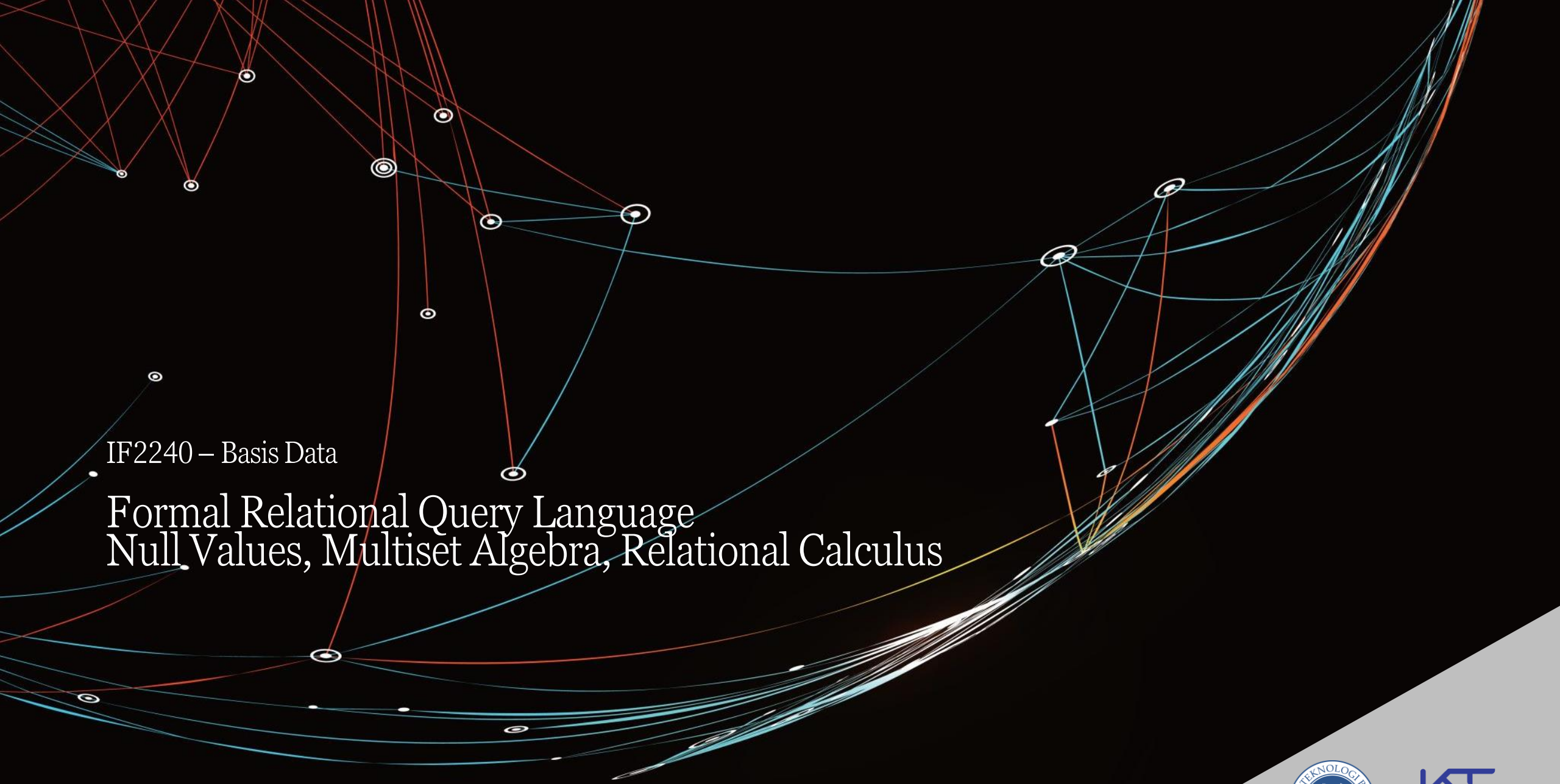
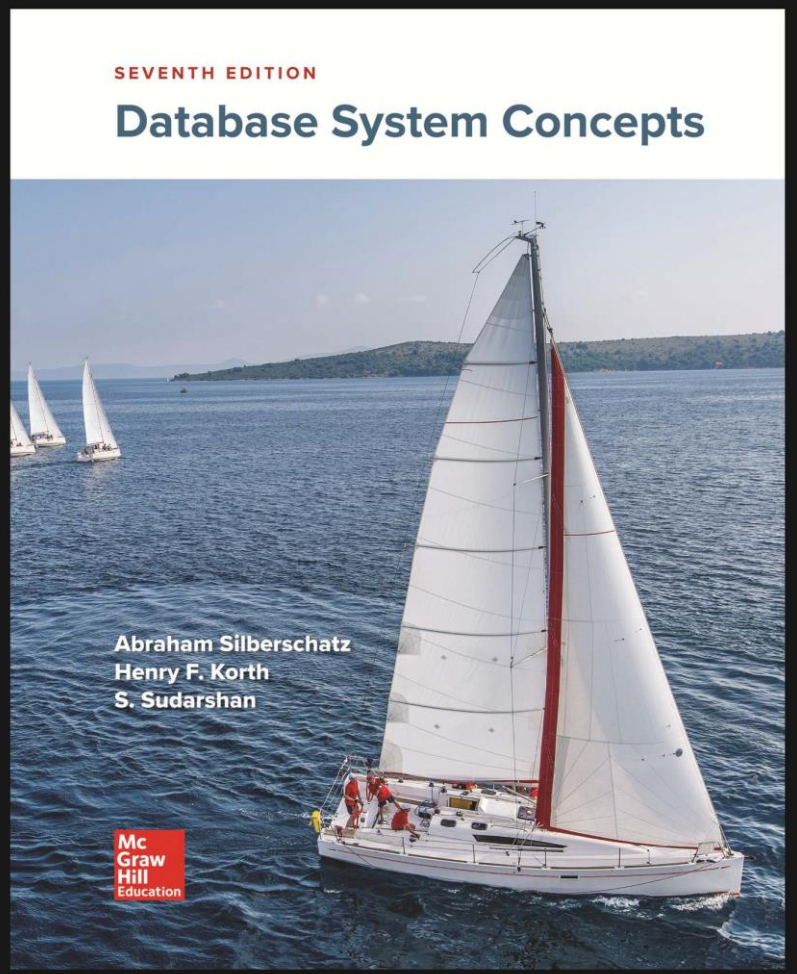


IF2240 – Basis Data

Formal Relational Query Language
Null Values, Multiset Algebra, Relational Calculus

KNOWLEDGE & SOFTWARE ENGINEERING



Sumber

Silberschatz, Korth, Sudarshan: "Database System Concepts", 7th Edition

- Chapter 2: Relational Model
  - Section 2.6. The Relational Algebra
- Chapter 27: Formal-Relational Query Languages (*online chapter*)
  - Section 27.1. The Tuple Relational Calculus
  - Section 27.2. The Domain Relational Calculus

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*

Multiset Relational Algebra

Pure relational algebra removes all duplicates

- e.g. after projection

Multiset relational algebra retains duplicates, to match SQL semantics

- SQL duplicate retention was initially for efficiency, but is now a feature

Multiset relational algebra defined as follows

- selection: has as many duplicates of a tuple as in the input, if the tuple satisfies the selection
- projection: one tuple per input tuple, even if it is a duplicate
- cross product: If there are *m* copies of *t1* in *r*, and *n* copies of *t2* in *s*, there are *m* x *n* copies of *t1.t2* in *r* x *s*
- Other operators similarly defined
  - E.g. union: *m* + *n copies,* intersection: min(*m*, *n*) copies
    difference: min(0, *m* – *n*) copies

Relational Calculus

Tuple Relational Calculus

A nonprocedural query language, where each query is of the form

$$\{t \mid P(t)\}$$

It is the set of all tuples t such that predicate P is true for t

t is a *tuple variable*, $t[A]$ denotes the value of tuple t on attribute A

$t \in r$ denotes that tuple t is in relation r

P is a *formula* similar to that of the predicate calculus

Predicate Calculus Formula

1. Set of attributes and constants
2. Set of comparison operators: (e.g., $<$, $\leq$, $=$, $\neq$, $>$, $\geq$)
3. Set of connectives: and ($\wedge$), or ($\vee$), not ($\neg$)
4. Implication ($\Rightarrow$): x $\Rightarrow$ y, if x is true, then y is true

$$x \Rightarrow y \equiv \neg x \vee y$$

5. Set of quantifiers:
   - $\exists\, t \in r\, (Q\,(t\,)) \equiv$ "there exists" a tuple t in relation r such that predicate $Q\,(t\,)$ is true
   - $\forall t \in r\, (Q\,(t\,)) \equiv Q$ is true "for all" tuples t in relation r

Example Queries

Find the *ID, name, dept_name, salary* for instructors whose salary is greater than $80,000

$\{t \mid t \in instructor \wedge t[salary] > 80000\}$

As in the previous query, but output only the *ID* attribute value

$\{t \mid \exists\ s \in \text{instructor}\ (t[ID] = s[ID] \wedge s[salary] > 80000)\}$

Notice that a relation on schema (*ID*) is implicitly defined by the query

schema

Example Queries

Find the names of all instructors whose department is in the Watson building

$$\{t \mid \exists s \in instructor\ (t[name] = s[name] \\ \wedge \exists u \in department\ (u[dept\_name] = s[dept\_name] \\ \wedge\ u[building] = \text{“Watson”}))\}$$

Find the set of all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or both

$$\{t \mid \exists s \in section\ (t[course\_id] = s[course\_id] \wedge \\ s[semester] = \text{“Fall”} \wedge s[\text{year}] = 2017) \\ \vee \exists u \in section\ (t[course\_id] = u[course\_id] \wedge \\ u[semester] = \text{“Spring”} \wedge u[\text{year}] = 2018)\}$$

schema

Example Queries

Find the set of all courses taught in the Fall 2017 semester, and in the Spring 2018 semester

$$\{t \mid \exists s \in section\ (t[course\_id] = s[course\_id] \wedge s[semester] = \text{“Fall”} \wedge s[year] = 2017) \wedge \exists u \in section\ (t[course\_id] = u[course\_id] \wedge u[semester] = \text{“Spring”} \wedge u[year] = 2018)\}$$

Find the set of all courses taught in the Fall 2017 semester, but not in the Spring 2018 semester

$$\{t \mid \exists s \in section\ (t[course\_id] = s[course\_id] \wedge s[semester] = \text{“Fall”} \wedge s[year] = 2017) \wedge \neg \exists u \in section\ (t[course\_id] = u[course\_id] \wedge u[semester] = \text{“Spring”} \wedge u[year] = 2018)\}$$

schema

Safety of Expressions

It is possible to write tuple calculus expressions that generate infinite relations.

For example, { t | $\neg$ $t \in r$ } results in an infinite relation if the domain of any attribute of relation r is infinite

To guard against the problem, we restrict the set of allowable expressions to safe expressions.

An expression $\{t \mid P(t)\}$ in the tuple relational calculus is *safe* if every component of t appears in one of the relations, tuples, or constants that appear in P

- NOTE: this is more than just a syntax condition.
  - E.g. { $t \mid t[A] = 5 \vee$ **true** } is not safe --- it defines an infinite set with attribute values that do not appear in any relation or tuples or constants in P.

Universal Quantification

Find all students who have taken all courses offered in the Biology department

$$\{t \mid \exists\, r \in \mathit{student}\ (t\,[\mathit{ID}] = r\,[\mathit{ID}]) \wedge$$
$$(\forall\, u \in \mathit{course}\ (u\,[\mathit{dept\_name}]=\text{“Biology”} \Rightarrow$$
$$\exists\, s \in \mathit{takes}\ (t\,[\mathit{ID}] = s\,[\mathit{ID}\ ] \wedge$$
$$s\,[\mathit{course\_id}] = u\,[\mathit{course\_id}]))\}$$

- Note that without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.

schema

Domain Relational Calculus

A nonprocedural query language equivalent in power to the tuple relational calculus

Each query is an expression of the form:

$$\{ < x_1, x_2, \ldots, x_n > \mid P(x_1, x_2, \ldots, x_n) \}$$

- $x_1, x_2, \ldots, x_n$ represent domain variables
- P represents a formula similar to that of the predicate calculus

Example Queries

Find the *ID, name, dept_name, salary* for instructors whose salary is greater than $80,000

$\{< i, n, d, s> \mid < i, n, d, s> \in instructor \wedge s > 80000\}$

As in the previous query, but output only the *ID* attribute value

$\{< i> \mid \exists\, n, d, s\ (< i, n, d, s> \in instructor \wedge s > 80000)\}$

Find the names of all instructors whose department is in the Watson building

$\{< n > \mid \exists\, i, d, s\ (< i, n, d, s > \in instructor$
$\wedge \exists\, b, a\ (< d, b, a> \in department \wedge b = \text{“Watson”}\ ))\}$

schema

Example Queries

Find the set of all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or both

$$\{\langle c\rangle \mid \exists\, a, s, y, b, r, t\ (\langle c, a, s, y, b, r, t\rangle \in section \wedge s = \text{“Fall”} \wedge y = 2017) \vee \exists\, a, s, y, b, r, t\ (\langle c, a, s, y, b, r, t\rangle \in section \wedge s = \text{“Spring”} \wedge y = 2018)\}$$

This case can also be written as

$$\{\langle c\rangle \mid \exists\, a, s, y, b, r, t\ (\langle c, a, s, y, b, r, t\rangle \in section \wedge ((s = \text{“Fall”} \wedge y = 2017) \vee (s = \text{“Spring”} \wedge y = 2018))\}$$

Find the set of all courses taught in the Fall 2017 semester, and in the Spring 2018 semester

$$\{\langle c\rangle \mid \exists\, a, s, y, b, r, t\ (\langle c, a, s, y, b, r, t\rangle \in section \wedge s = \text{“Fall”} \wedge y = 2017) \wedge \exists\, a, s, y, b, r, t\ (\langle c, a, s, y, b, r, t\rangle \in section \wedge s = \text{“Spring”} \wedge y = 2018)\}$$

schema

Safety of Expressions

The expression:

$$\{ < x_1, x_2, \ldots, x_n > \mid P(x_1, x_2, \ldots, x_n) \}$$

is safe if all of the following hold:

1. All values that appear in tuples of the expression are values from *dom* (P) (that is, the values appear either in P or in a tuple of a relation mentioned in P).
2. For every "there exists" subformula of the form $\exists x (P_1(x))$, the subformula is true if and only if there is a value of x in *dom* (P_1) such that $P_1(x)$ is true.
3. For every "for all" subformula of the form $\forall_x (P_1 (x))$, the subformula is true if and only if $P_1(x)$ is true for all values x from *dom* (P_1).

Universal Quantification

Find all students who have taken all courses offered in the Biology department

$$\{< i > \mid \exists\, n, d, tc\ (< i, n, d, tc > \in student \ \wedge$$
$$(\forall\, ci, ti, dn, cr\ (< ci, ti, dn, cr > \in course \wedge dn = \text{“Biology”}$$
$$\Rightarrow \exists\, si, se, y, g\ (< i, ci, si, se, y, g > \in takes\ ))\}$$

- Note that without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.