



Chapter 2: Intro to Relational Model & Chapter 6.1: Relational Algebra

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use



Example of a Relation

Relation
(or table)

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

attributes
(or columns)

tuples
(or rows)



Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain
- The null value causes complications in the definition of many operations



Relation Schema and Instance

- A_1, A_2, \dots, A_n are *attributes*
- $R = (A_1, A_2, \dots, A_n)$ is a *relation schema*

Example:

instructor = (*ID*, *name*, *dept_name*, *salary*)

- Formally, given sets D_1, D_2, \dots, D_n a **relation** r is a subset of
 $D_1 \times D_2 \times \dots \times D_n$

Thus, a relation is a set of n -tuples (a_1, a_2, \dots, a_n) where each $a_i \in D_i$

- The current values (**relation instance**) of a relation are specified by a table
- An element t of r is a *tuple*, represented by a *row* in a table



Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

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



Database

- A database consists of multiple relations
- Information about an enterprise is broken up into parts

instructor
student
advisor

- Bad design:

univ (instructor -ID, name, dept_name, salary, student_Id, ..)

results in

- repetition of information (e.g., two students have the same instructor)
 - the need for null values (e.g., represent an student with no advisor)
- Normalization theory (Chapter 7) deals with how to design “good” relational schemas

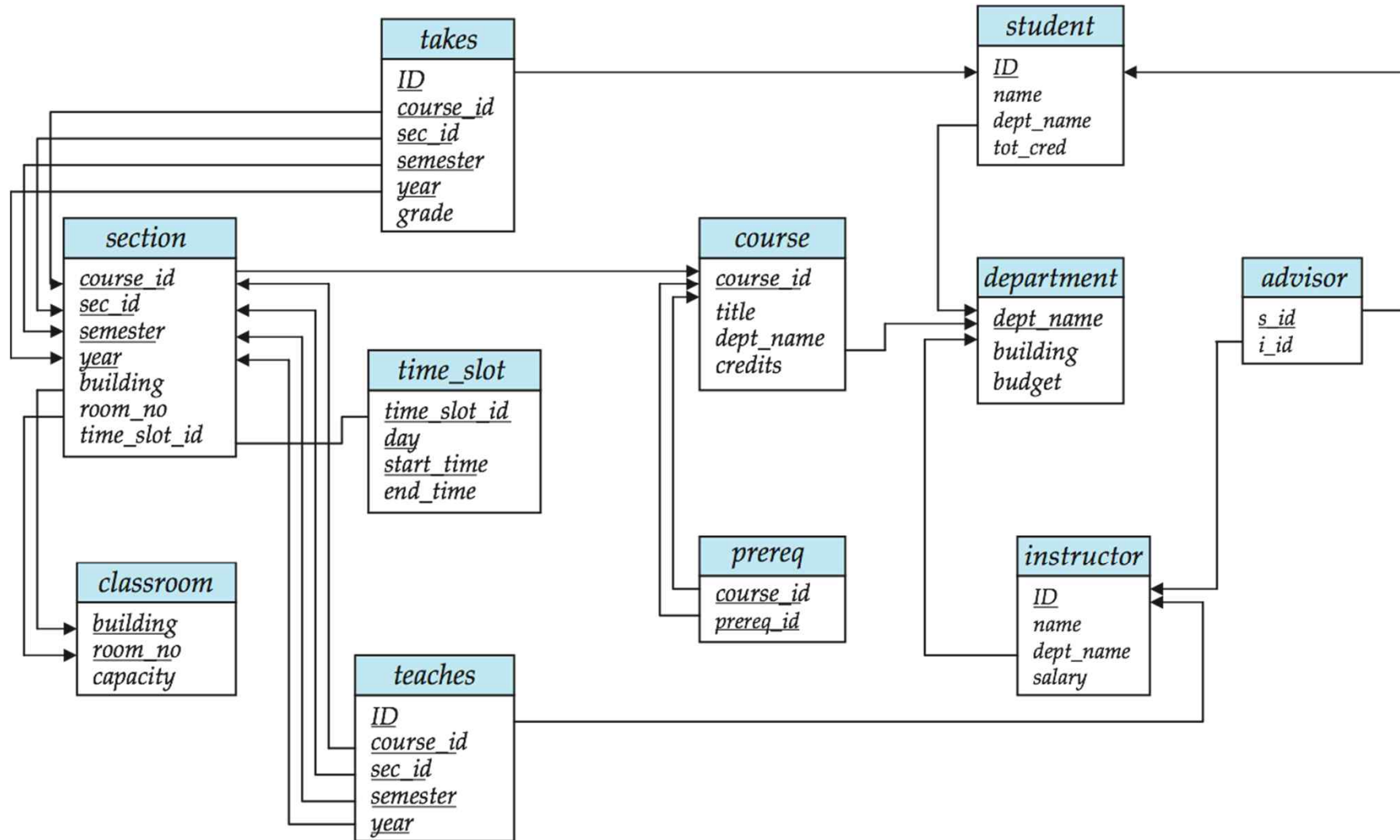


Keys

- Let $K \subseteq R$
- K is a **superkey** of R if values for K are sufficient to identify a unique tuple of each possible relation $r(R)$
 - Example: $\{ID\}$ and $\{ID, name\}$ are both superkeys of *instructor*.
- Superkey K is a **candidate key** if K is minimal
 - Example: $\{ID\}$ is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
 - which one?
- **Foreign key** constraint: Value in one relation must appear in another
 - **Referencing** relation
 - **Referenced** relation



Schema Diagram for University Database





Relational Query Languages

- Procedural vs. non-procedural (declarative)
- “Pure” languages: fundamental, lacking the “syntactic sugar”
 - **Relational algebra** (procedural)
 - Tuple relational calculus
 - Domain relational calculus } (non-procedural)



Relational Algebra

- Algebra: operators and operands
 - Relational algebra
 - ▶ Operands: relations
 - ▶ Operators: basic operators (+ additional operations)
- 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 – Example

- Relation r

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

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

A	B	C	D
α	α	1	7
β	β	23	10



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:

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

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

- Example of selection:

instructor (*ID*, *name*, *dept_name*, *salary*)

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



Project Operation – Example

- Relation r

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

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

A	C
α	1
α	1
β	1
β	2

 $=$

A	C
α	1
β	1
β	2



Project Operation

- 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*
instructor (*ID*, *name*, *dept_name*, *salary*)

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



Composition of Operations

- Can build expressions using multiple operations
- Example: $\Pi_{B,C} (\sigma_{A=\alpha} (r))$

- Relation r

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

- $\sigma_{A=\alpha} (r)$

A	B	C	D
α	α	1	7
α	β	5	7

- $\Pi_{B,C} (\sigma_{A=\alpha} (r))$

B	C
α	1
β	5



Exercise

employee (person_name, street, city, salary)

- Find the names of all employees who live in city “Seoul”

- Find the names of all employees whose salary is greater than 100,000

- Find the names of all employees who live in “Seoul” and whose salary is greater than 100,000



Union Operation – Example

- Relations r , s :

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cup s$:

A	B
α	1
α	2
β	1
β	3



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.
 1. r, s must have the same **arity** (same number of attributes)
 2. 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
section (course_id, sec_id, semester, year, building, room_number, time_slot_id)

$$\Pi_{\text{course_id}} (\sigma_{\text{semester}=\text{"Fall"} \wedge \text{year}=2009} (\text{section})) \cup \\ \Pi_{\text{course_id}} (\sigma_{\text{semester}=\text{"Spring"} \wedge \text{year}=2010} (\text{section}))$$



Set difference of two relations

- Relations r , s :

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r - s$:

A	B
α	1
β	1



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

section (course_id, sec_id, semester, year, building, room_number, time_slot_id)

$$\Pi_{course_id} (\sigma_{semester="Fall" \wedge year=2009} (section)) - \Pi_{course_id} (\sigma_{semester="Spring" \wedge year=2010} (section))$$



Cartesian-Product Operation – Example

■ Relations r , s :

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

■ $r \times s$:

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



Cartesian-Product Operation

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

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

- 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.



Exercise

branch (branch-name, branch-city, assets)

customer (customer-name, customer-street, customer-city)

account (account-number, branch-name, balance)

loan (loan-number, branch-name, amount)

depositor (customer-name, account-number)

borrower (customer-name, loan-number)

- Find the names of all customers who have a loan, an account, or both, from the bank.

- Find the names of all customers who have a loan at the “Gwanak” branch.

- Find the names of all customers who have a loan at the “Gwanak” branch but do not have an account at any branch of the bank.



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 .



Example Query

- Find the largest salary in the university

instructor (*ID*, *name*, *dept_name*, *salary*)

- Step 1: find instructor salaries that are less than some other instructor salary (i.e. not maximum)
 - ▶ using a copy of *instructor* under a new name *d*

$\Pi_{instructor.salary} (\sigma_{instructor.salary < d.salary} (instructor \times \rho_d (instructor)))$

- Step 2: Find the largest salary

$\Pi_{salary} (instructor) -$

$\Pi_{instructor.salary} (\sigma_{instructor.salary < d.salary} (instructor \times \rho_d (instructor)))$



Example Queries

- Find the names of all instructors in the Physics department, along with the *course_id* of all courses they have taught

- Query 1

$$\Pi_{instructor.ID, course_id} (\sigma_{dept_name="Physics"} (\sigma_{instructor.ID=teaches.ID} (instructor \times teaches)))$$

- Query 2

$$\Pi_{instructor.ID, course_id} (\sigma_{instructor.ID=teaches.ID} (\sigma_{dept_name="Physics"} (instructor \times teaches)))$$



Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
 - A relation in the database
 - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
 - $E_1 \cup E_2$
 - $E_1 - E_2$
 - $E_1 \times E_2$
 - $\sigma_p(E_1)$, P is a predicate on attributes in E_1
 - $\Pi_S(E_1)$, S is a list consisting of some of the attributes in E_1
 - $\rho_x(E_1)$, x is the new name for the result of E_1



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



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



Set-Intersection Operation – Example

■ Relation r, s :

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

■ $r \cap s$

A	B
α	2



Natural-Join Operation

- Notation: $r \bowtie s$
- 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:
 - Consider each pair of tuples t_r from r and t_s from s .
 - 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, where
 - ▶ t has the same value as t_r on r
 - ▶ t has the same value as t_s on s

- Example:

$R = (A, B, C, D)$

$S = (E, B, D)$

- Result schema = (A, B, C, D, E)
- $r \bowtie s$ is defined 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 Example

■ 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

■ $r \bowtie s$

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



Natural Join and Theta Join

- Find the names of all instructors in the Comp. Sci. department together with the course titles of all the courses that the instructors teach
 - $\Pi_{name, title} (\sigma_{dept_name="Comp. Sci."} (instructor \bowtie teaches \bowtie course))$
- Natural join is associative
 - $(instructor \bowtie teaches) \bowtie course$ is equivalent to $instructor \bowtie (teaches \bowtie course)$
- Natural join is commutative
 - $instruct \bowtie teaches$ is equivalent to $teaches \bowtie instructor$
- The **theta join** operation $r \bowtie_{\theta} s$ is defined as
 - $r \bowtie_{\theta} s = \sigma_{\theta} (r \times s)$



Exercise

branch (branch-name, branch-city, assets)

customer (customer-name, customer-street, customer-city)

account (account-number, branch-name, balance)

depositor (customer-name, account-number)

- Find all customers who have an account from at least the “Gwanak” and “Gangnam” branches.



Assignment Operation

- The assignment operation (\leftarrow) 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.
- Modification of the database can be expressed using the assignment operator



Assignment Example

- Rewrite $r \bowtie s$ with assignment operations

temp1 $\leftarrow r \times s$

temp2 $\leftarrow \sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \dots \wedge r.A_n = s.A_n}(\text{temp1})$

result $\leftarrow \Pi_{R \cap S}(\text{temp2})$



Outer Join

- An extension of the join operation that avoids loss of information
- 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.
 - ▶ We shall study precise meaning of comparisons with nulls later



Natural Join – Example

Relation *course*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

Relation *prereq*

<i>course_id</i>	<i>prereq_id</i>
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

■ Natural Join

course ⋈ *prereq*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101



Left Outer Join – Example

Relation *course*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

Relation *prereq*

<i>course_id</i>	<i>prereq_id</i>
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

■ Left Outer Join

course ⋈ *prereq*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	<i>null</i>



Right Outer Join – Example

Relation *course*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

Relation *prereq*

<i>course_id</i>	<i>prereq_id</i>
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

■ Right Outer Join

course ⋈_⊃ *prereq*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	<i>null</i>	<i>null</i>	<i>null</i>	CS-101



Full Outer Join – Example

Relation *course*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

Relation *prereq*

<i>course_id</i>	<i>prereq_id</i>
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

■ Full Outer Join

course \bowtie *prereq*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	<i>null</i>
CS-347	<i>null</i>	<i>null</i>	<i>null</i>	CS-101



Outer Join using Joins

- Outer join can be expressed using basic operations

- e.g. $r \bowtie s$ can be written as

$$(r \bowtie s) \cup (r - \Pi_R(r \bowtie s) \times \{null, \dots, null\})$$



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* is used instead of *unknown*?
 $(1 < null) = false \Rightarrow \text{not } (1 < null) = true (!)$
- Three-valued logic using the truth value *unknown*:
 - OR: $(unknown \text{ or } true) = true,$
 $(unknown \text{ or } false) = unknown$
 $(unknown \text{ or } unknown) = unknown$
 - AND: $(true \text{ and } unknown) = unknown,$
 $(false \text{ and } unknown) = false,$
 $(unknown \text{ and } unknown) = unknown$
 - NOT: $(\text{not } unknown) = unknown$
 - 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 $m \times n$ copies of $t1.t2$ in $r \times s$
 - Other operators similarly defined
 - ▶ E.g. union: $m + n$ copies, intersection: $\min(m, n)$ copies
difference: $\min(0, m - n)$ copies



End of Chapter 2 & 6.1

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use