



CS425 – Fall 2016
Boris Glavic
Chapter 3: Formal Relational Query Languages

Modified from:
Database System Concepts, 6th Ed.
©Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use



Chapter 3: Formal Relational Query Languages

- Relational Algebra
- Tuple Relational Calculus
- Domain Relational Calculus



Textbook: Chapter 6

CS425 – Boris Glavic

3.2

©Silberschatz, Korth and Sudarshan



Relational Query Languages

- Procedural vs non-procedural (**declarative**)
- “Pure” languages:
 - Relational algebra
 - Tuple relational calculus
 - Domain relational calculus
- Expressive power of a query language
 - What queries can be expressed in this language?
- Relational algebra:
 - Algebra of relations -> set of operators that take relations as input and produce relations as output
 - -> **closed**: the output of evaluating an expression in relational algebra can be used as input to another relational algebra expression
- Now: First introduction to operators of the relational algebra

CS425 – Boris Glavic

3.3

©Silberschatz, Korth and Sudarshan



Relational Algebra

- 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.
 - **composable**

CS425 – Boris Glavic

3.4

©Silberschatz, Korth and Sudarshan



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

CS425 – Boris Glavic

3.5

©Silberschatz, Korth and Sudarshan



Select Operation

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

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

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

Each **term** is one of:

<attribute> op <attribute> or <constant>

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

- Example of selection:

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



CS425 – Boris Glavic

3.6

©Silberschatz, Korth and Sudarshan

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

CS425 – Boris Glavic 3.7 ©Silberschatz, Korth and Sudarshan

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
- Let A be a subset of the attributes of relation r then:

$$\pi_A(r) = \{t.A \mid t \in r\}$$
- Example: To eliminate the *dept_name* attribute of *instructor*

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

! theory Warning

CS425 – Boris Glavic 3.8 ©Silberschatz, Korth and Sudarshan

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

CS425 – Boris Glavic 3.9 ©Silberschatz, Korth and Sudarshan

Union Operation

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

$$r \cup s = \{t \mid t \in r \vee t \in s\}$$
- For $r \cup s$ to be valid.
 - r, s must have the **same arity** (same number of attributes)
 - The attribute domains must be **union 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

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

CS425 – Boris Glavic 3.10 ©Silberschatz, Korth and Sudarshan

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

CS425 – Boris Glavic 3.11 ©Silberschatz, Korth and Sudarshan

Set Difference Operation

- Notation $r - s$
- Defined as:

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

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

CS425 – Boris Glavic 3.12 ©Silberschatz, Korth and Sudarshan

Formal Definition (Syntax)

- A basic expression in the relational algebra consists of either one of the following:
 - A relation in the database
 - A constant relation: e.g., $\{(1),(2)\}$
- 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

Theory Warning

CS425 - Boris Glavic
3.19
©Silberschatz, Korth and Sudarshan

Formal Definition (Semantics)

- Let E be an relational algebra expression. We use $[E](I)$ to denote the evaluation of E over a database instance I
 - For simplicity we will often drop I and $]$
- The result of evaluating a simple relational algebra expression E over a database is defined as
 - Simple relation: $[R](I) = R(I)$
 - Constant relation: $[C](I) = C$

Theory Warning

CS425 - Boris Glavic
3.20
©Silberschatz, Korth and Sudarshan

Formal Definition (Semantics)

- Let E_1 and E_2 be relational-algebra expressions.

$$[E_1 \cup E_2] = \{t \mid t \in [E_1] \vee t \in [E_2]\}$$

$$[E_1 - E_2] = \{t \mid t \in [E_1] \wedge t \notin [E_2]\}$$

$$[E_1 \times E_2] = \{t, t' \mid t \in [E_1] \wedge t' \in [E_2]\}$$

$$[\sigma_P(E_1)] = \{t \mid t \in [E_1] \wedge p(t)\}$$

$$[\pi_A(E_1)] = \{t.A \mid t \in [E_1]\}$$

$$[\rho_X(E_1)] = \{t(X) \mid t \in [E_1]\}$$

Theory Warning

CS425 - Boris Glavic
3.21
©Silberschatz, Korth and Sudarshan

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.
- **Examples:**
 - We register a new employee Peter, but the salary for this employee has not yet been determined
 - ▶ Unknown value
 - A government agency collects data about residents including their SSN. Some residents are not allowed to work and, thus, do not have an SSN
 - ▶ The attribute SSN is not applicable for such residents

CS425 - Boris Glavic
3.22
©Silberschatz, Korth and Sudarshan

Conditions with 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 $A \geq 5$
- 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: $(not \text{ unknown}) = unknown$
 - In SQL "*P* is **unknown**" evaluates to true if predicate P evaluates to *unknown*
- Result of selection predicate is treated as *false* if it evaluates to *unknown*

CS425 - Boris Glavic
3.23
©Silberschatz, Korth and Sudarshan

Arithmetics with Null Values

- 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)

CS425 - Boris Glavic
3.24
©Silberschatz, Korth and Sudarshan

Additional Operations

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

- Set intersection
- Natural join
- Assignment
- Outer join

CS425 – Boris Glavic 3.25 ©Silberschatz, Korth and Sudarshan

Set-Intersection Operation

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

$$r \cap s = \{t \mid t \in r \wedge t \in s\}$$
- Assume:
 - r, s have the same arity
 - attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$
 - That is adding intersection to the language does not make it more expressive

CS425 – Boris Glavic 3.26 ©Silberschatz, Korth and Sudarshan

Set-Intersection Operation – Example

- Relation r, s :

A	B
α	1
α	2
β	1

A	B
α	2
β	3

r s

- $r \cap s$

A	B
α	2

CS425 – Boris Glavic 3.27 ©Silberschatz, Korth and Sudarshan

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:

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

CS425 – Boris Glavic 3.28 ©Silberschatz, Korth and Sudarshan

Natural-Join Operation (cont.)

- Let r and s be relations on schemas R and S respectively. Then, $r \bowtie s$ is defined as:

$$X = R \cap S$$

$$S' = S - R$$

$$r \bowtie s = \pi_{R, S'}(\sigma_{r.X = s.X}(r \times s))$$

CS425 – Boris Glavic 3.29 ©Silberschatz, Korth and Sudarshan

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

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

r 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

CS425 – Boris Glavic 3.30 ©Silberschatz, Korth and Sudarshan

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 (we ignore attribute order)
 - $instructor \bowtie teaches$ is equivalent to $teaches \bowtie instructor$
- The **theta join** operation $r \bowtie_{\theta} s$ is defined as

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

CS425 – Boris Glavic 3.31 ©Silberschatz, Korth and Sudarshan

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.

$$E_1 \leftarrow \sigma_{salary > 40000}(instructor)$$

$$E_2 \leftarrow \sigma_{salary < 10000}(instructor)$$

$$E_3 \leftarrow E_1 \cup E_2$$

CS425 – Boris Glavic 3.32 ©Silberschatz, Korth and Sudarshan

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

CS425 – Boris Glavic 3.33 ©Silberschatz, Korth and Sudarshan

Outer Join – Example

- Relation *instructor1*

ID	name	dept_name
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music
- Relation *teaches1*

ID	course_id
10101	CS-101
12121	FIN-201
76766	BIO-101

CS425 – Boris Glavic 3.34 ©Silberschatz, Korth and Sudarshan

Outer Join – Example

- Join
 $instructor \bowtie teaches$

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
- Left Outer Join
 $instructor \bowtie_{\leftarrow} teaches$

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	<i>null</i>

CS425 – Boris Glavic 3.35 ©Silberschatz, Korth and Sudarshan

Outer Join – Example

- Right Outer Join
 $instructor \bowtie_{\rightarrow} teaches$

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
76766	<i>null</i>	<i>null</i>	BIO-101
- Full Outer Join
 $instructor \bowtie_{\leftrightarrow} teaches$

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	<i>null</i>
76766	<i>null</i>	<i>null</i>	BIO-101

CS425 – Boris Glavic 3.36 ©Silberschatz, Korth and Sudarshan

Defining Outer Join using Join

- Outer join can be expressed using basic operations

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

$$r \ltimes s = (r \bowtie s) \cup (\{\text{null}, \dots, \text{null}\} \times (s - \Pi_S(r \bowtie s)))$$

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

CS425 - Boris Glavic 3.37 ©Silberschatz, Korth and Sudarshan

Division Operator

- Given relations $r(R)$ and $s(S)$, such that $S \subset R$, $r \div s$ is the largest relation $t(R-S)$ such that $t \times s \subseteq r$
 - Alternatively, all tuples from r (R-S) such that all their extensions on $R \cap S$ with tuples from s exist in R
- E.g. let $r(ID, course_id) = \Pi_{ID, course_id}(takes)$ and $s(course_id) = \Pi_{course_id}(\sigma_{dept_name='Biology'}(course))$ then $r \div s$ gives us students who have taken all courses in the Biology department
- Can write $r \div s$ as

$$E_1 \leftarrow \Pi_{R-S}(r)$$

$$E_2 \leftarrow \Pi_{R-S}((E_1 \times s) - \Pi_{R-S,S}(r \bowtie s))$$

$$r \div s = E_1 - E_2$$

CS425 - Boris Glavic 3.38 ©Silberschatz, Korth and Sudarshan

Division Operator Example

- Return the name of all persons that read all newspapers

reads	
name	newspaper
Peter	Times
Bob	Wall Street
Alice	Times
Alice	Wall Street

newspaper
Times
Wall Street

$$E_1 \leftarrow \Pi_{name}(reads)$$

$$E_2 \leftarrow \Pi_{name}((E_1 \times newspaper) - \Pi_{name,newspaper}(reads \bowtie newspaper))$$

$$reads \div newspaper = E_1 - E_2$$

$$[reads \div newspaper] = \{Alice\}$$

CS425 - Boris Glavic 3.39 ©Silberschatz, Korth and Sudarshan

Extended Relational-Algebra-Operations

- Generalized Projection
- Aggregate Functions

CS425 - Boris Glavic 3.40 ©Silberschatz, Korth and Sudarshan

Generalized Projection

- Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\pi_{F_1, \dots, F_n}(E)$$

- E is any relational-algebra expression
- Each of F_1, F_2, \dots, F_n are arithmetic expressions and function calls involving constants and attributes in the schema of E .
- Given relation $instructor(ID, name, dept_name, salary)$ where salary is annual salary, get the same information but with monthly salary
 - $\Pi_{ID, name, dept_name, salary/12}(instructor)$
- Adding functions increases expressive power!
 - In standard relational algebra there is no way to change attribute values

CS425 - Boris Glavic 3.41 ©Silberschatz, Korth and Sudarshan

Aggregate Functions and Operations

- Aggregation function takes a set of values and returns a single value as a result.
 - avg: average value
 - min: minimum value
 - max: maximum value
 - sum: sum of values
 - count: number of values
- Aggregate operation in relational algebra

$$G_1, G_2, \dots, G_m \mathcal{G}_{F_1(A_1), F_2(A_2), \dots, F_n(A_n)}(E)$$

E is any relational-algebra expression

- G_1, G_2, \dots, G_m is a list of attributes on which to group (can be empty)
- Each F_i is an aggregate function
- Each A_i is an attribute name

- Note: Some books/articles use \mathcal{G} (Calligraphic G)

CS425 - Boris Glavic 3.42 ©Silberschatz, Korth and Sudarshan

Aggregate Operation – Example

- Relation r .

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

- $\mathcal{G}_{\text{sum}(c)}(r)$

$\text{sum}(c)$
27

CS425 – Boris Glavic 3.43 ©Silberschatz, Korth and Sudarshan

Aggregate Operation – Example

- Find the average salary in each department

$\text{dept_name } \mathcal{G} \text{ avg}(\text{salary}) (\text{instructor})$

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

dept_name	avg_salary
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

CS425 – Boris Glavic 3.44 ©Silberschatz, Korth and Sudarshan

Aggregate Functions (Cont.)

- What are the names for attributes in aggregation results?
 - Need some convention!
 - E.g., use the expression as a name $\text{avg}(\text{salary})$
 - For convenience, we permit renaming as part of aggregate operation

$\text{dept_name } \mathcal{G} \text{ avg}(\text{salary}) \text{ as } \text{avg_sal} (\text{instructor})$

CS425 – Boris Glavic 3.45 ©Silberschatz, Korth and Sudarshan

Modification of the Database

- The content of the database may be modified using the following operations:
 - Deletion
 - Insertion
 - Updating
- All these operations can be expressed using the assignment operator
- Example: Delete instructors with salary over \$1,000,000

$$R \leftarrow R - (\sigma_{\text{salary} > 1000000}(R))$$

CS425 – Boris Glavic 3.46 ©Silberschatz, Korth and Sudarshan

Restrictions for Modification

- Consider a modification where $R=(A,B)$ and $S=(C)$

$$R \leftarrow \sigma_{C > 5}(S)$$

- This would change the schema of R!
 - Should not be allowed
- Requirements for modifications
 - The name R on the left-hand side of the assignment operator refers to an existing relation in the database schema
 - The expression on the right-hand side of the assignment operator should be union-compatible with R

CS425 – Boris Glavic 3.47 ©Silberschatz, Korth and Sudarshan

Tuple Relational Calculus

CS425 – Boris Glavic 3.48 ©Silberschatz, Korth and Sudarshan



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

CS425 - Boris Glavic

3.49

©Silberschatz, Korth and Sudarshan



Predicate Calculus Formula

1. Set of attributes and constants
2. Set of comparison operators: (e.g., $<$, \leq , $=$, \neq , $>$, \geq)
3. Set of logical 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 in 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

CS425 - Boris Glavic

3.50

©Silberschatz, Korth and Sudarshan



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

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

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

- 1) t is not bound to any relation by the predicate
- 2) we implicitly state that t has an *ID* attribute ($t[ID] = s[ID]$)

CS425 - Boris Glavic

3.51

©Silberschatz, Korth and Sudarshan



Example Queries

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

$$\{t \mid \exists s \in instructor (t[name] = s[name] \wedge \exists u \in department (u[dept_name] = s[dept_name] \wedge u[building] = "Watson"))\}$$

- Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

$$\{t \mid \exists s \in section (t[course_id] = s[course_id] \wedge s[semester] = "Fall" \wedge s[year] = 2009) \vee \exists u \in section (t[course_id] = u[course_id] \wedge u[semester] = "Spring" \wedge u[year] = 2010)\}$$

CS425 - Boris Glavic

3.52

©Silberschatz, Korth and Sudarshan



Example Queries

- Find the set of all courses taught in the Fall 2009 semester, and in the Spring 2010 semester

$$\{t \mid \exists s \in section (t[course_id] = s[course_id] \wedge s[semester] = "Fall" \wedge s[year] = 2009) \wedge \exists u \in section (t[course_id] = u[course_id] \wedge u[semester] = "Spring" \wedge u[year] = 2010)\}$$

- Find the set of all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\{t \mid \exists s \in section (t[course_id] = s[course_id] \wedge s[semester] = "Fall" \wedge s[year] = 2009) \wedge \neg \exists u \in section (t[course_id] = u[course_id] \wedge u[semester] = "Spring" \wedge u[year] = 2010)\}$$

CS425 - Boris Glavic

3.53

©Silberschatz, Korth and Sudarshan



Safety of Expressions

- It is possible to write tuple calculus expressions that generate infinite relations.
- For example, $\{t \mid \neg t \in r\}$ 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. $\{t \mid t[A] = 5 \vee \text{true}\}$ is not safe --- it defines an infinite set with attribute values that do not appear in any relation or tuples or constants in P .

CS425 - Boris Glavic

3.54

©Silberschatz, Korth and Sudarshan



Universal Quantification

- Find all students who have taken all courses offered in the Biology department
 - $\{t \mid \exists r \in \text{student} (t[ID] = r[ID]) \wedge (\forall u \in \text{course} (u[\text{dept_name}] = \text{"Biology"} \Rightarrow \exists s \in \text{takes} (t[ID] = s[ID] \wedge s[\text{course_id}] = u[\text{course_id}])))\}$
 - Note that without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.

CS425 - Boris Glavic

3.55

©Silberschatz, Korth and Sudarshan



Domain Relational Calculus

CS425 - Boris Glavic

3.56

©Silberschatz, Korth and Sudarshan



Domain Relational Calculus

- A nonprocedural query language equivalent in power to the tuple relational calculus
- Each query is an expression of the form:

$$\{ \langle x_1, x_2, \dots, x_n \rangle \mid P(x_1, x_2, \dots, x_n) \}$$

- x_1, x_2, \dots, x_n represent domain variables
 - Variables that range of attribute values
- P represents a formula similar to that of the predicate calculus
- Tuples can be formed using $\langle \rangle$
 - E.g., $\langle \text{"Einstein"}, \text{"Physics"} \rangle$

CS425 - Boris Glavic

3.57

©Silberschatz, Korth and Sudarshan



Example Queries

- Find the ID , $name$, $dept_name$, $salary$ for instructors whose salary is greater than \$80,000
 - $\{ \langle i, n, d, s \rangle \mid \langle i, n, d, s \rangle \in \text{instructor} \wedge s > 80000 \}$
- As in the previous query, but output only the ID attribute value
 - $\{ \langle i \rangle \mid \langle i, n, d, s \rangle \in \text{instructor} \wedge s > 80000 \}$
- Find the names of all instructors whose department is in the Watson building
 - $\{ \langle n \rangle \mid \exists i, d, s (\langle i, n, d, s \rangle \in \text{instructor} \wedge \exists b, a (\langle d, b, a \rangle \in \text{department} \wedge b = \text{"Watson"})) \}$

CS425 - Boris Glavic

3.58

©Silberschatz, Korth and Sudarshan



Example Queries

- Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both
 - $\{ \langle c \rangle \mid \exists a, s, y, b, r, t (\langle c, a, s, y, b, t \rangle \in \text{section} \wedge s = \text{"Fall"} \wedge y = 2009) \vee \exists a, s, y, b, r, t (\langle c, a, s, y, b, t \rangle \in \text{section} \wedge s = \text{"Spring"} \wedge y = 2010) \}$
 - This case can also be written as
 - $\{ \langle c \rangle \mid \exists a, s, y, b, r, t (\langle c, a, s, y, b, t \rangle \in \text{section} \wedge ((s = \text{"Fall"} \wedge y = 2009) \vee (s = \text{"Spring"} \wedge y = 2010))) \}$
- Find the set of all courses taught in the Fall 2009 semester, and in the Spring 2010 semester
 - $\{ \langle c \rangle \mid \exists a, s, y, b, r, t (\langle c, a, s, y, b, t \rangle \in \text{section} \wedge s = \text{"Fall"} \wedge y = 2009) \wedge \exists a, s, y, b, r, t (\langle c, a, s, y, b, t \rangle \in \text{section} \wedge s = \text{"Spring"} \wedge y = 2010) \}$

CS425 - Boris Glavic

3.59

©Silberschatz, Korth and Sudarshan



Safety of Expressions

The expression:

$$\{ \langle x_1, x_2, \dots, x_n \rangle \mid P(x_1, x_2, \dots, x_n) \}$$

is safe if all of the following hold:

- All values that appear in tuples of the expression are values from $\text{dom}(P)$ (that is, the values appear either as constants in P or in a tuple of a relation mentioned in P).
- 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 $\text{dom}(P_1)$ such that $P_1(x)$ is true.
- 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 $\text{dom}(P_1)$.

CS425 - Boris Glavic

3.60

©Silberschatz, Korth and Sudarshan



Universal Quantification

- Find all students who have taken all courses offered in the Biology department
 - $\{ \langle i \rangle \mid \exists n, d, tc (\langle i, n, d, tc \rangle \in student \wedge (\forall ci, ti, dn, cr (\langle ci, ti, dn, cr \rangle \in course \wedge dn = \text{"Biology"} \Rightarrow \exists si, se, y, g (\langle i, ci, si, se, y, g \rangle \in takes))) \}$
 - Note that without the existential quantification on student, the above query would be unsafe if the Biology department has not offered any courses.

* Above query fixes bug in page 246, last query



Relationship between Relational Algebra and Tuple (Domain) Calculus

- Codd's theorem**
 - Relational algebra and tuple calculus are equivalent in terms of expressiveness
- That means that every query expressible in relational algebra can also be expressed in tuple calculus and vice versa
- Since domain calculus is as expressive as tuple calculus the same holds for the domain calculus
- Note: Here relational algebra refers to the standard version (no aggregation and projection with functions)



End of Chapter 3

Modified from:
 Database System Concepts, 6th Ed.
 ©Silberschatz, Korth and Sudarshan
 See www.db-book.com for conditions on re-use



Outline

- Introduction
- Relational Data Model
- Formal Relational Languages (relational algebra)
- SQL - Introduction**
- Database Design
- Transaction Processing, Recovery, and Concurrency Control
- Storage and File Structures
- Indexing and Hashing
- Query Processing and Optimization



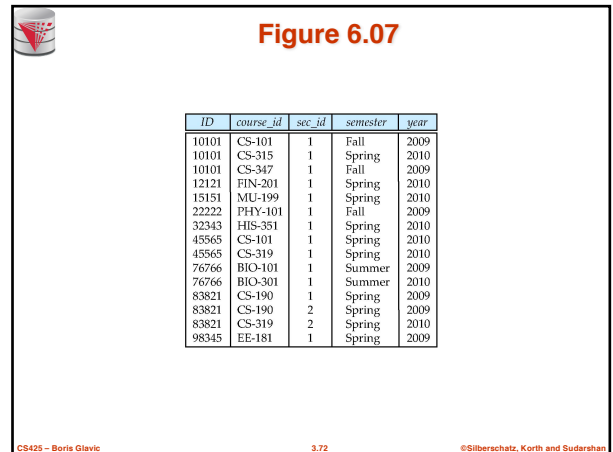
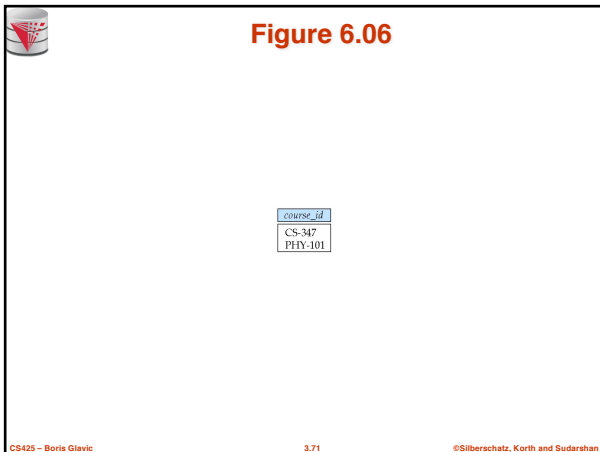
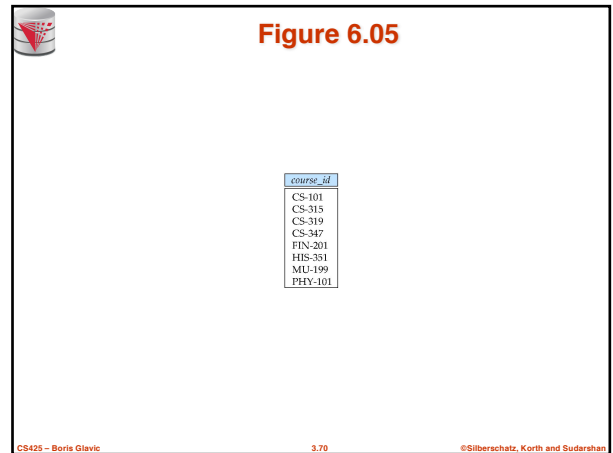
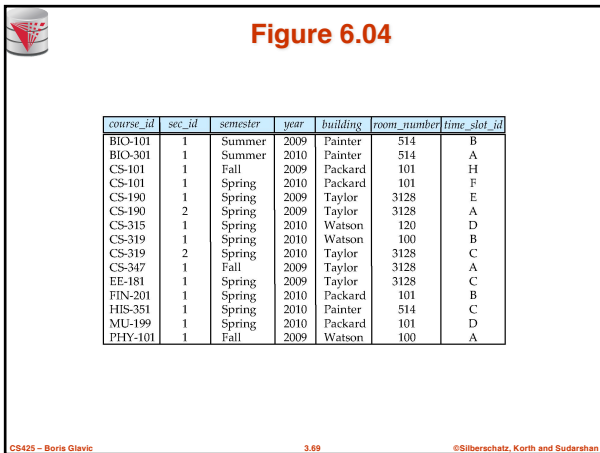
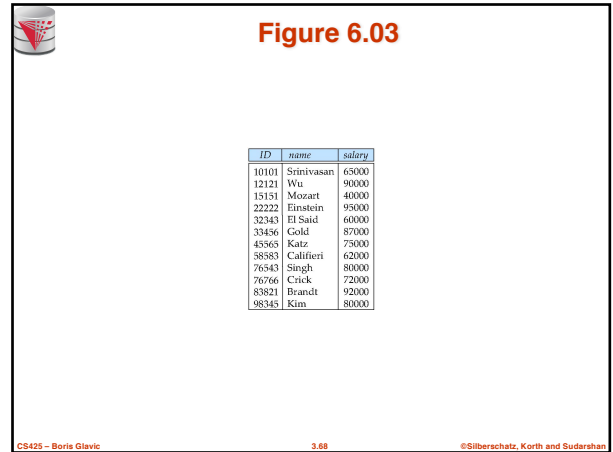
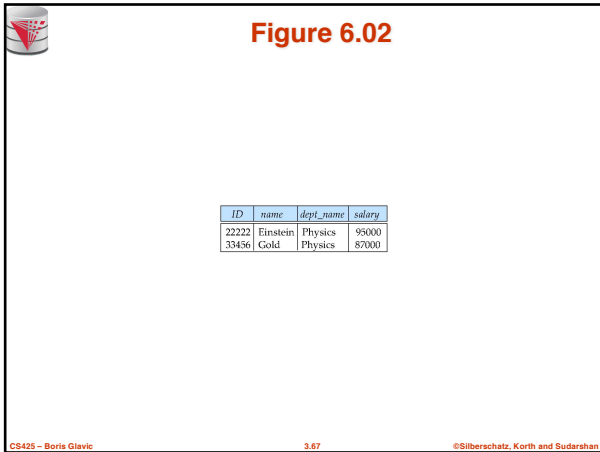
Recap

- Query language**
 - Declarative**
 - Retrieve, combine, and analyze data from a database instance
- Relational algebra**
 - Standard relational algebra:
 - Selection, projection, renaming, cross product, union, set difference
 - Null values
 - Semantic sugar operators:
 - Intersection, joins, division,
 - Extensions:
 - Aggregation, extended projection
- Tuple Calculus**
 - safety
- Domain Calculus**



Figure 6.01

ID	name	dept_name	salary
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	Califleri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000



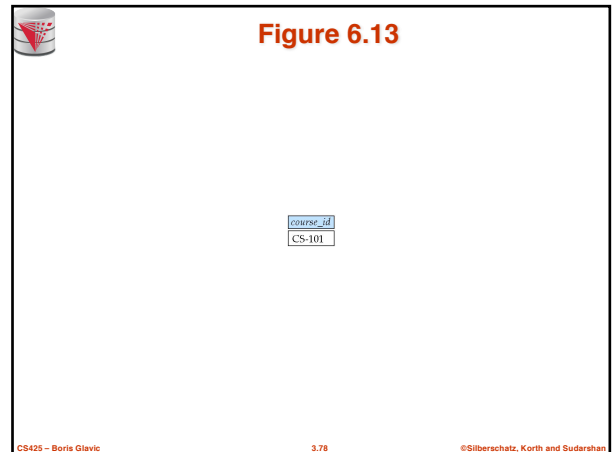
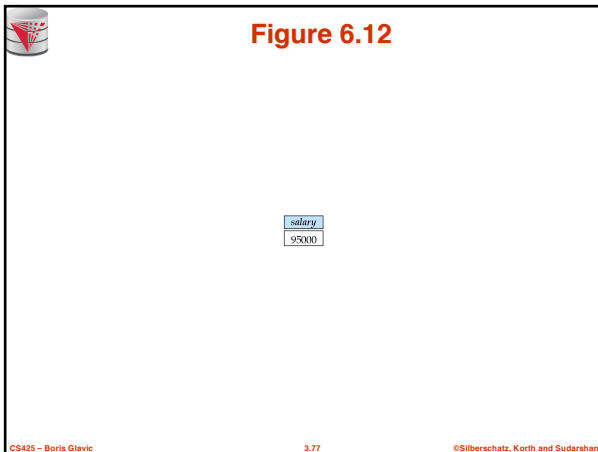
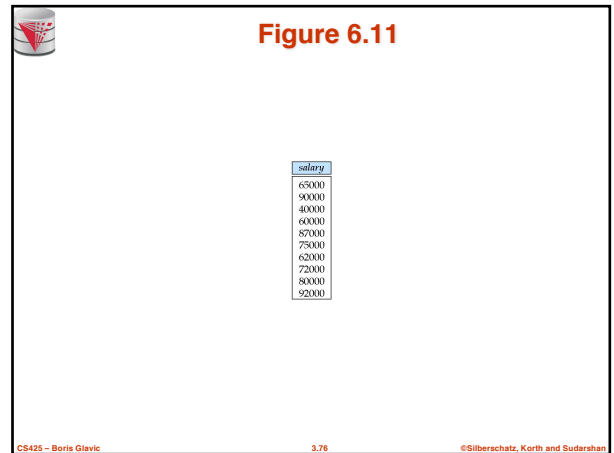
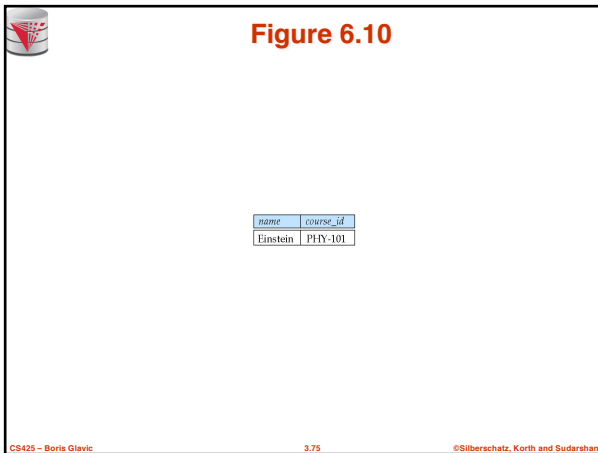
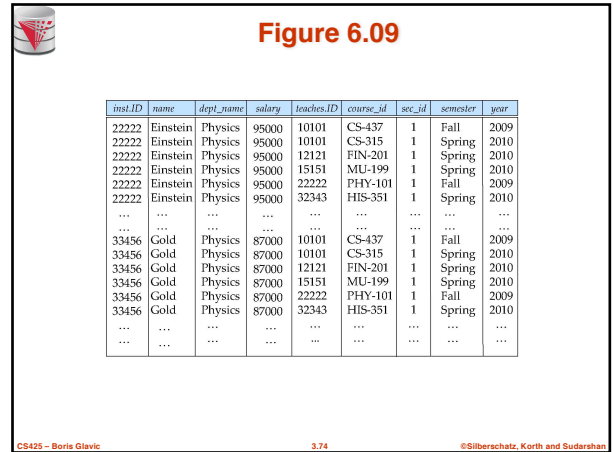
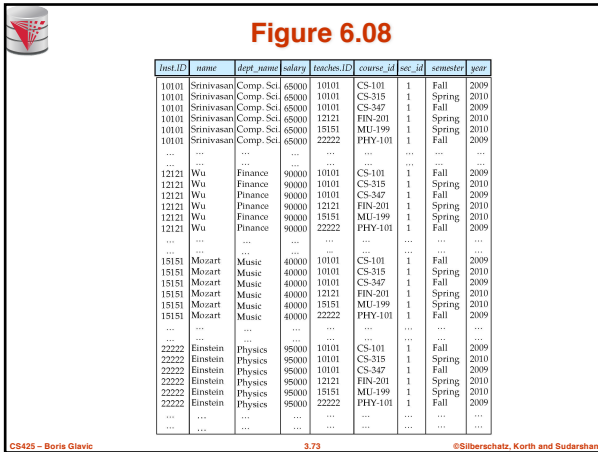




Figure 6.14

ID	name	dept_name	salary	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

CS425 - Boris Glavic

3.79

©Silberschatz, Korth and Sudarshan



Figure 6.15

name	course_id
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

CS425 - Boris Glavic

3.80

©Silberschatz, Korth and Sudarshan



Figure 6.16

name	title
Brandt	Game Design
Brandt	Image Processing
Katz	Image Processing
Katz	Intro. to Computer Science
Srinivasan	Intro. to Computer Science
Srinivasan	Robotics
Srinivasan	Database System Concepts

CS425 - Boris Glavic

3.81

©Silberschatz, Korth and Sudarshan



Figure 6.17

ID	name	dept_name	salary	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
33456	Gold	Physics	87000	null	null	null	null
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
58583	Califrieri	History	62000	null	null	null	null
76543	Singh	Finance	80000	null	null	null	null
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

CS425 - Boris Glavic

3.82

©Silberschatz, Korth and Sudarshan



Figure 6.18

ID	course_id	sec_id	semester	year	name	dept_name	salary
10101	CS-101	1	Fall	2009	Srinivasan	Comp. Sci.	65000
10101	CS-315	1	Spring	2010	Srinivasan	Comp. Sci.	65000
10101	CS-347	1	Fall	2009	Srinivasan	Comp. Sci.	65000
12121	FIN-201	1	Spring	2010	Wu	Finance	90000
15151	MU-199	1	Spring	2010	Mozart	Music	40000
22222	PHY-101	1	Fall	2009	Einstein	Physics	95000
32343	HIS-351	1	Spring	2010	El Said	History	60000
33456	null	null	null	null	Gold	Physics	87000
45565	CS-101	1	Spring	2010	Katz	Comp. Sci.	75000
45565	CS-319	1	Spring	2010	Katz	Comp. Sci.	75000
58583	null	null	null	null	Califrieri	History	62000
76543	null	null	null	null	Singh	Finance	80000
76766	BIO-101	1	Summer	2009	Crick	Biology	72000
76766	BIO-301	1	Summer	2010	Crick	Biology	72000
83821	CS-190	1	Spring	2009	Brandt	Comp. Sci.	92000
83821	CS-190	2	Spring	2009	Brandt	Comp. Sci.	92000
83821	CS-319	2	Spring	2010	Brandt	Comp. Sci.	92000
98345	EE-181	1	Spring	2009	Kim	Elec. Eng.	80000

CS425 - Boris Glavic

3.83

©Silberschatz, Korth and Sudarshan



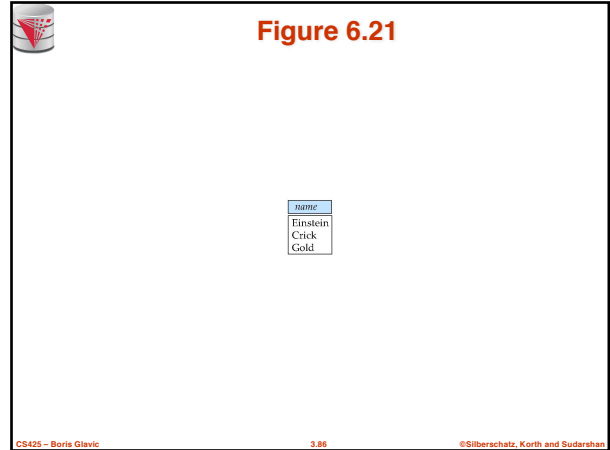
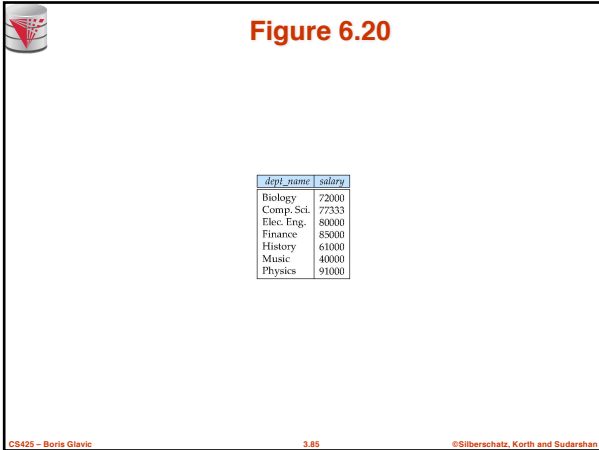
Figure 6.19

ID	name	dept_name	salary
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
32343	El Said	History	60000
58583	Califrieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

CS425 - Boris Glavic

3.84

©Silberschatz, Korth and Sudarshan



Deletion

- A delete request is expressed similarly to a query, except instead of displaying tuples to the user, the selected tuples are removed from the database.
- Can delete only whole tuples; cannot delete values on only particular attributes
- A deletion is expressed in relational algebra by:

$$r \leftarrow r - E$$
 where r is a relation and E is a relational algebra query.

CS425 – Boris Glavic 3.87 ©Silberschatz, Korth and Sudarshan

Deletion Examples

- Delete all account records in the Perryridge branch.

$$account \leftarrow account - \sigma_{branch_name = \text{Perryridge}}(account)$$
- Delete all loan records with amount in the range of 0 to 50

$$loan \leftarrow loan - \sigma_{amount \geq 0 \text{ and } amount \leq 50}(loan)$$
- Delete all accounts at branches located in Needham.

$$r_1 \leftarrow \sigma_{branch_city = \text{Needham}}(account \bowtie branch)$$

$$r_2 \leftarrow \pi_{account_number, branch_name, balance}(r_1)$$

$$r_3 \leftarrow \pi_{customer_name, account_number}(r_2 \bowtie depositor)$$

$$account \leftarrow account - r_2$$

$$depositor \leftarrow depositor - r_3$$

CS425 – Boris Glavic 3.88 ©Silberschatz, Korth and Sudarshan

Insertion

- To insert data into a relation, we either:
 - specify a tuple to be inserted
 - write a query whose result is a set of tuples to be inserted
- in relational algebra, an insertion is expressed by:

$$r \leftarrow r \cup E$$
 where r is a relation and E is a relational algebra expression.
- The insertion of a single tuple is expressed by letting E be a constant relation containing one tuple.

CS425 – Boris Glavic 3.89 ©Silberschatz, Korth and Sudarshan

Insertion Examples

- Insert information in the database specifying that Smith has \$1200 in account A-973 at the Perryridge branch.

$$account \leftarrow account \cup \{(\text{A-973}, \text{Perryridge}, 1200)\}$$

$$depositor \leftarrow depositor \cup \{(\text{Smith}, \text{A-973})\}$$
- Provide as a gift for all loan customers in the Perryridge branch, a \$200 savings account. Let the loan number serve as the account number for the new savings account.

$$r_1 \leftarrow (\sigma_{branch_name = \text{Perryridge}}(borrower \bowtie loan))$$

$$account \leftarrow account \cup \pi_{loan_number, branch_name, 200}(r_1)$$

$$depositor \leftarrow depositor \cup \pi_{customer_name, loan_number}(r_1)$$

CS425 – Boris Glavic 3.90 ©Silberschatz, Korth and Sudarshan



Updating

- A mechanism to change a value in a tuple without changing *all* values in the tuple
- Use the generalized projection operator to do this task

$$r \leftarrow \prod_{F_1, F_2, \dots, F_i} (r)$$

- Each F_i is either
 - the i^{th} attribute of r , if the i^{th} attribute is not updated, or,
 - if the attribute is to be updated F_i is an expression, involving only constants and the attributes of r , which gives the new value for the attribute

CS425 – Boris Glavic

3.31

©Silberschatz, Korth and Sudarshan



Update Examples

- Make interest payments by increasing all balances by 5 percent.

$$account \leftarrow \prod_{account_number, branch_name, balance * 1.05} (account)$$

- Pay all accounts with balances over \$10,000 6 percent interest and pay all others 5 percent

$$account \leftarrow \prod_{account_number, branch_name, balance * 1.06} (\sigma_{BAL > 10000} (account)) \cup \prod_{account_number, branch_name, balance * 1.05} (\sigma_{BAL \leq 10000} (account))$$

CS425 – Boris Glavic

3.32

©Silberschatz, Korth and Sudarshan



Example Queries

- Find the names of all customers who have a loan and an account at bank.

$$\prod_{customer_name} (borrower) \cap \prod_{customer_name} (depositor)$$

- Find the name of all customers who have a loan at the bank and the loan amount

$$\prod_{customer_name, loan_number, amount} (borrower \bowtie loan)$$

CS425 – Boris Glavic

3.33

©Silberschatz, Korth and Sudarshan



Example Queries

- Find all customers who have an account from at least the "Downtown" and the Uptown" branches.

- Query 1

$$\prod_{customer_name} (\sigma_{branch_name = "Downtown"} (depositor \bowtie account)) \cap$$

$$\prod_{customer_name} (\sigma_{branch_name = "Uptown"} (depositor \bowtie account))$$

- Query 2

$$\prod_{customer_name, branch_name} (depositor \bowtie account) \div$$

$$\div P_{temp}(branch_name) (\{ ("Downtown"), ("Uptown") \})$$

Note that Query 2 uses a constant relation.

CS425 – Boris Glavic

3.34

©Silberschatz, Korth and Sudarshan



Bank Example Queries

- Find all customers who have an account at all branches located in Brooklyn city.

$$\prod_{customer_name, branch_name} (depositor \bowtie account)$$

$$\div \prod_{branch_name} (\sigma_{branch_city = "Brooklyn"} (branch))$$

CS425 – Boris Glavic

3.35

©Silberschatz, Korth and Sudarshan