



Lecture 2 of 42

Relational Databases Discussion: Problem Set 1

Friday, 26 January 2008

William H. Hsu
Department of Computing and Information Sciences, KSU

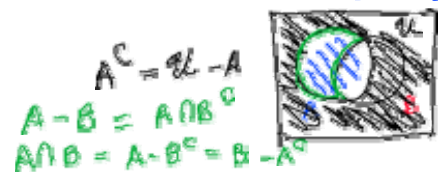
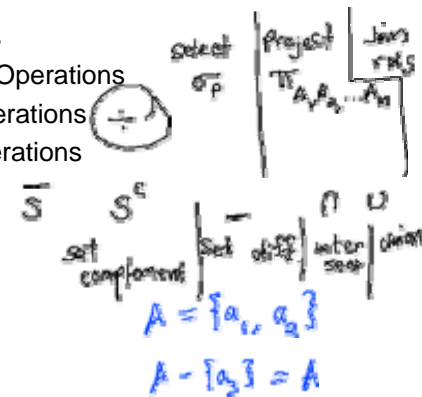
KSOL course page: <http://snipurl.com/1pq4c>
Course web site: <http://www.kddresearch.org/Courses/Spring-2008/CIS560>
Instructor home page: <http://www.cis.ksu.edu/~bhsu>

Reading for Next Class:
Chapter 2, Silberschatz *et al.*, 5th edition – next week
Problem Set 1



Chapter 2: Relational Model

- Structure of Relational Databases
- Fundamental Relational-Algebra-Operations
- Additional Relational-Algebra-Operations
- Extended Relational-Algebra-Operations
- Null Values
- Modification of the Database





Example of a Relation

<i>account_number</i>	<i>branch_name</i>	<i>balance</i>
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350



Basic Structure

- 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 $a_i \in D_i$

- Example: If

customer_name = {Jones, Smith, Curry, Lindsay}

customer_street = {Main, North, Park}

customer_city = {Harrison, Rye, Pittsfield}

Then $r = \{$ (Jones, Main, Harrison),
(Smith, North, Rye),
(Curry, North, Rye),
(Lindsay, Park, Pittsfield) $\}$

is a relation over

customer_name \times *customer_street* \times *customer_city*



Attribute Types

- Each attribute of a relation has a name
- 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
 - * Note: multivalued attribute values are not atomic
 - * Note: composite attribute values are not atomic
- The special value *null* is a member of every domain
- The null value causes complications in the definition of many operations
 - * We shall ignore the effect of null values in our main presentation and consider their effect later



Relation Schema

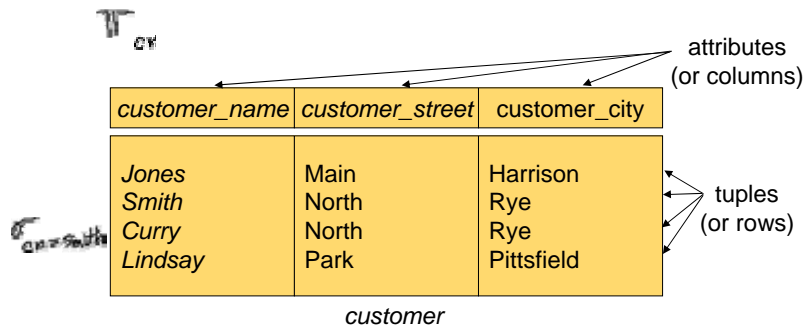
- A_1, A_2, \dots, A_n are *attributes*
- $R = (A_1, A_2, \dots, A_n)$ is a *relation schema*
Example:
 $Customer_schema = (customer_name, customer_street, customer_city)$
- $r(R)$ is a *relation* on the *relation schema* R
Example:
 $customer (Customer_schema)$





Relation Instance

- 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: *account* relation with unordered tuples

<i>account_number</i>	<i>branch_name</i>	<i>balance</i>
A-101	Downtown	500
A-215	Mianus	700
A-102	Perryridge	400
A-305	Round Hill	350
A-201	Brighton	900
A-222	Redwood	700
A-217	Brighton	750



Database



- A database consists of multiple relations
- Information about an enterprise is broken up into parts, with each relation storing one part of the information

account : stores information about accounts

depositor : stores information about which customer owns which account

customer : stores information about customers

- Storing all information as a single relation such as *bank(account_number, balance, customer_name, ..)* results in
 - * repetition of information (e.g., two customers own an account)
 - * the need for null values (e.g., represent a customer without an account)
- Normalization theory (Chapter 7) deals with how to design relational schemas



The *customer* Relation

<i>customer_name</i>	<i>customer_street</i>	<i>customer_city</i>
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton



The *depositor* Relation

<i>customer_name</i>	<i>account_number</i>
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305



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 $r(R)$
 - * by "possible r " we mean a relation r that could exist in the enterprise we are modeling.
 - * Example: $\{customer_name, customer_street\}$ and $\{customer_name\}$ are both superkeys of *Customer*, if no two customers can possibly have the same name.
- K is a **candidate key** if K is minimal
Example: $\{customer_name\}$ is a candidate key for *Customer*, since it is a superkey (assuming no two customers can possibly have the same name), and no subset of it is a superkey.
- **Primary Key**



Query Languages

- Language in which user requests information from the database.
- Categories of languages
 - * Procedural
 - * Non-procedural, or declarative
- "Pure" languages:
 - * Relational algebra
 - * Tuple relational calculus
 - * Domain relational calculus
- Pure languages form underlying basis of query languages that people use.



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.

$$\sigma_p(r)$$

$$\Pi_{i=1}^n$$



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

$$\sigma_{p_1 \wedge p_2}(r) = \sigma_{p_1}(r) \cap \sigma_{p_2}(r)$$



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:

$$\sigma_{\text{branch_name}=\text{"Perryridge"}}(\text{account})$$

$$\sigma_p(r)$$

```
SELECT *
FROM r
WHERE p
```



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

SELECT DISTINCT AC
FROM r



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 *branch_name* attribute of *account*

$$\Pi_{\text{account_number}, \text{balance}}(\text{account})$$



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.
 - 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 customers with either an account or a loan

$$\Pi_{customer_name}(depositor) \cup \Pi_{customer_name}(borrower)$$



Set Difference Operation – Example

- 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

Extreme case

$$|r - s| \geq |r| - |s| \rightarrow s \in r$$

$$|r \cup s| \leq |r| + |s| \rightarrow r \cap s = \emptyset$$



Set Difference Operation

- Notation $r - s$
- Defined as:

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

$$r \cap s^c$$

- 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



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:

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

- 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.



Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$
- $r \times s$

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

- $\sigma_{A=C}(r \times s)$

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



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 .

