CISS430 Lecture 3: Data models

Yihsiang Liow

February 19, 2020

Table of contents I

Data models I

- **Data model**: Model/notation for managing data. Made up of 3 parts:
  - Structure
  - Operations
  - Constraints
- **Structure**:
  - Very high level description
  - Higher level than data structures
  - Also called **conceptual model**
- **Operations**:
  - Operations available on the structure
  - More limited than operations in standard programming langs

Data models II

- **Constraints**:
  - Limits on what data can be stored
- Two important data models:
  - Relational (includes object-relational model)
  - Semistructured (includes XML)

Relational data model (brief) I

- Structure:

Movies

title	year	length	genre
Gone with the wind	1939	231	drama
Star Wars	1977	124	scifi
Wayne's World	1992	95	comedy

- Operations:
  - **Relational algebra**
  - Table-oriented operations: input is tables, output is table

Relational data model (brief) II

- Constraints:
  - Example: Uniqueness of a value in a column
  - Example: Type of values that can do into a column

Semistructured data model (brief) I

- XML – hierarchical, nested tagged elements
- Structure:

```
<Movies>
    <Movie title="Gone with the wind">
        <Year>1939</Year>
        <Length>231</Length>
        <Genre>drama</Genre>
    </Movie>
    <Movie title="Star Wars">
        <Year>1977</Year>
        <Length>124</Length>
        <Genre>scifi</Genre>
    </Movie>
    <Movie title="Wayne's World">
        <Year>1992</Year>
        <Length>95</Length>
```

Semistructured data model (brief) II

```
        <Genre>comedy</Genre>
    </Movie>
</Movies>
```

- Operations: Tree/forest operations. Follow path in tree.
- Constraints: Can limit value for a tag element, etc.

Other data models I

- Object-relational data model: Add object-oriented idea to relational data model. Values can be structure instead of just basic types like integers, strings, etc. Relations can have methods.
- Object-oriented data model: Data model is made up of objects.
- Hierarchical data model
- Network data model

Comparison I

- Semistructure appears to be more flexible than relations - graphs can be described using semistructured data models.
- But relational still preferred in DBMS
- Why?
- DB are large:
  - Efficiency of access and modification is important
  - Ease of use very important - productivity of programmers.
- Relational data model:
  - Very simple, very limited structure. Reasonably versatile.
  - Limited operations (you'll see that there are very few operations) but sufficiently useful.

Comparison II

- Limitations become features.
- Allows efficienct implementation of SQL.
- Programs in SQL very short and compact.
- So SQL programs can be optimized and can be faster than code written by hand in other non-SQL languages.

Relational data model: structure I

- Recall example:

Movies

title	year	length	genre
Gone with the wind	1939	231	drama
Star Wars	1977	124	scifi
Wayne's World	1992	95	comedy

- Structure of relational data model is made up of two parts:
  - Attributes – set of names with data types
  - Data – set of tuples

Relational data model: structure II

- For each tuple, there is one data for each attribute

- Note: The attributes is treated as a *set*, not as a tuple. In implementation, attributes are usually ordered.
- Tuple: one value for each attribute of the relation. The value of a tuple is also called a component of the tuple. NOTE: The collection of tuples in a relation is considered a *set*.
- **Schema**: name of relation and attributes. Example:

```
Movies(title, year, length, genre)
```

- Relational model = one or more relations

Relational data model: structure III

- Database schema = set of schemas for all the relations in a database.
- **Domain**: The set of possible values for the values of an attribute.
  - Must be atomic such as integer or string type.
  - NOT record/struct/array/set/list.
  - Also called data type.
  - Example: Schema with data types

```
Movies(title:string, year:integer,
       length:integer, genre:string)
```

- (Relation can change with time. An **instance** of a relation is the relation at one point in time. Conventional DB usually keeps only one instance of a relation.)

Relational data model: structure IV

- Given two relations R and S, you have two sets of attributes, one from R and one from S. The two sets of attributes are the same if their names are the same and their data types are the same.
- Given two relations R and S, you can only compare tuples $r \in R$ and $s \in S$ (by equality) only if R and S have the same attributes. $r = s$ if for each attribute, the corresponding values from r and s are the same.
- Given two relations R and S, we say that $R \subseteq S$ if R and S have the same attributes and each tuple of R is a tuple of S, i.e., for each tuple $r \in R$, there is some tuple $s \in S$ such that $r = s$.

Relational data model: structure V

- Given two relations R and S, $R = S$ if $R \subseteq S$ and $S \subseteq R$.
  - $R = S$ simply means R and S have the same attributes and they have the same set of tuples.

Relational data model: constraints I

- Many possible constraints.
- Key constraint.
  - **Key**: Set of attributes of a relation such that tuples in the relation are uniquely identified by the values of the attributes of a key.
- Example: If in the `Movies` relation, a movie is uniquely identified by $\{\texttt{title}, \texttt{year}\}$, then this is a key. This means that there are no two movies with the same title and same year. If $\{\texttt{title}, \texttt{year}\}$ is a key of the `Movies` schema, then write

$$\texttt{Movies}(\underline{\texttt{title}},\ \underline{\texttt{year}},\ \texttt{length},\ \texttt{genre})$$

Relational data model: constraints II

- Example: If you have a schema that looks like this

  `Employee(fname, lname, ...)`

  you might want this:

  Employee(<u>fname</u>, <u>lname</u>, ...)

  But this means that you cannot have two employees with the same first and last name. Better to issue an employee id to each employee:

  Employee(<u>eid</u>, fname, lname)

Relational data model: constraints III

- Example. C# is a corporate issued id.

```
Movies(title:string, year:integer, length,integer,
       genre:string, studioName:string, producerC#)
MovieStar(name:string, address:string, gender:char,
          birthdate:date)
StarsIn(movieTitle:string, movieYear:integer,
        starName:string)
MovieExec(name:string, address:string, C#:integer,
          netWorth:integer)
Studio(name:string, address:stirng, presC#:integer)
```

Relational data model: constraints IV

- In SQL there are two sublanguages:
  - **DDL** – data definition language. For managing schemas.
  - **DML** – data manipulation language. For managing data (tuples).
- SQL supports 3 relations
  - tables – Stored
  - views – Relation defined by computation. Not stored.
  - temporary tables – Create by SQL language processor. Not created by programs. Not stored.

Relational data model: relational algebra I

- **Relational algebra**: operations of relation data model.
  - Union, intersection, difference, cross/cartesian product
  - Selection, projection, rename
- $\bigcup$ (union), $\bigcap$ (intersection), $-$ (difference)
  - If R and S are relations, then $R \cup S$, $R \cap S$, $R - S$ is defined only when R and S have the same set of attributes.
  - If so, then the attributes of $R \cup S$, $R \cap S$, $R - S$ is the same as the attributes of R (or S).
- $R \cup S$ – Tuples are from R or S.
- $R \cap S$ – Tuples in both R and S.
- $R - S$ – Tuples in R but not in S.

Relational data model: relational algebra II

- $R \times S$ – Tuples of the form $(r_1, r_2, ..., r_m, s_1, s_2, ..., s_n)$ where $(r_1, r_2, ..., r_m)$ is a tuple in R and $(s_1, s_2, ..., s_n)$ is a tuple in S. If R has attribute a and S also has an attribute a, then distibguish them in $R \times S$ by calling them $R.a$ and $S.a$.

Relational data model: relational algebra III

- Example.

Relational data model: relational algebra IV

JohnFriends

id	name	bday
20	Tom	10/10/98
21	Mary	10/8/98
22	Harry	10/7/98

JaneFriends

id	name	bday
20	Tom	10/10/98
21	Sue	5/8/97
24	Harry	10/7/98

JohnFriends $\cup$ JaneFriends

id	name	bday
20	Tom	10/10/98
21	Mary	10/8/98
22	Harry	10/7/98
21	Sue	5/8/97
24	Harry	10/7/98

JohnFriends $\cap$ JaneFriends

id	name	bday
20	Tom	10/10/98

JohnFriends $-$ JaneFriends

id	name	bday
21	Mary	10/8/98
22	Harry	10/7/98

Relational data model: relational algebra V

JohnFriends $\times$ JaneFriends

JohnFriends.id	JohnFriends.name	JohnFriends.bday	JaneFriends.id	JaneFriends.name	JaneFriends.bday
20	Tom	10/10/98	20	Tom	10/10/98
20	Tom	10/10/98	21	Sue	5/8/97
20	Tom	10/10/98	24	Harry	10/7/98
21	Mary	10/8/98	20	Tom	10/10/98
21	Mary	10/8/98	21	Sue	5/8/97
21	Mary	10/8/98	24	Harry	10/7/98
22	Harry	10/7/98	20	Tom	10/10/98
22	Harry	10/7/98	21	Sue	5/8/97
22	Harry	10/7/98	24	Harry	10/7/98

Relational data model: relational algebra VI

- **Selection**:

$$\sigma_c(R) = \text{same as } R \text{ but only keep tuples satisfying } c$$

c is a propositional formula on the attributes of R.

- Assume boolean operators $=$ and $\neq$ available for any data type. For numeric data types, assume $\leq, <, \geq, >$ available.
- Example. $\sigma_{\texttt{numfingers}>5}(\texttt{Persons})$ gives relation of persons with more than 5 fingers.

Relational data model: relational algebra VII

- Example. If

Reserves

sid	bid	day
22	101	10/10/98
22	103	10/8/98
22	104	10/7/98

then $\sigma_{\texttt{bid}>\texttt{102}}(\texttt{Reserves})$ is

sid	bid	day
22	103	10/8/98
22	104	10/7/98

Relational data model: relational algebra VIII

- **Projection**:

  $\pi_{A_1,\ldots,A_n}(R) =$ same as R but only keep attributes $A_1, \ldots, A_n$

  $A_1, \ldots, A_n$ must be attributes of R.

  - Example. If

Reserves

sid	bid	day
22	101	10/10/98
22	103	10/8/98
22	104	10/7/98

  then $\pi_{\texttt{sid,bid}}(\texttt{Reserves})$ is

Relational data model: relational algebra IX

sid	bid
22	101
22	103
22	104

Relational data model: relational algebra X

- **Rename**:
  - $\rho(T, R)$: Make a copy of R and call it T.
  - $\rho(T(A_1 \to A_1', A_2 \to A_2', \ldots), R)$: Same as above and also rename attribute A_1 to A_1', etc.
  - $\rho(T(1 \to A_1', 2 \to A_2', \ldots), R)$: Same as above except first attribute is renamed A_1', second attribute is renamed as A_2', etc.

Relational data model: relational algebra XI

- Example. If

Reserves

sid	bid	day
22	103	10/8/98
22	104	10/7/98

then $\rho(\texttt{X}(\texttt{sid} \to \texttt{s}, \texttt{day} \to \texttt{x}), \texttt{Reserves})$ creates

X

s	bid	x
22	103	10/8/98
22	104	10/7/98

- Note that $R \cap S = R - (R - S)$. So $\cap$ is not really necessary. But it's important and common enough to be a separate operator for clarity and optimization.

Relational data model: relational algebra XII

- **Join**:
  - **Inner join**: Shorthand for "selection of a cross product":

$$R \bowtie_c S = \sigma_c(R \times S)$$

    Also called **conditional join**.
  - **Equi–join**: Join where c is the 'logical and' ($\wedge$) of one or more equalities. Example:

$$\texttt{Humans} \bowtie_{(\texttt{Humans.telescopes=Planet.moons})} \texttt{Planets}$$

    i.e., humans match with planets where the number of telescopes owned by a human is exactly the number of moons (satellites) as the planet.

Relational data model: relational algebra XIII

- **Natural join**: Inner join where c involves the keys of R and S with the same name. Example: Assume that in the Cars and Bikes table, (owner, color) is a key.

$$\texttt{Cars} \bowtie \texttt{Bikes}$$

$$= \texttt{Cars} \times_{\texttt{Cars.owner=Bikes.owner} \,\wedge\, \texttt{Cars.color=Bikes.color}} \texttt{Bikes}$$

- **Left outer join**: Like inner join except that unmatches tuples of left-hand side relation is included and the right-hand side tuples set to NULL.
- **Right outer join**: Like inner join except that unmatches tuples of right-hand side relation is included and the left-hand side tuples set to NULL.
- **Outer join**: Union of left and right outer joins.

Examples I

- Will use the following relations.
- Schemas:

```
Sailors(sid: integer, sname: string, rating: integer,
        age: real)
Boats(bid:integer, bname: string, color: string)
Reserves(sid:integer, bid: integer, date: date)
```

Examples II

- Data:

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

sid	bid	date
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Examples III

- Query 1. Names of all sailors who reserved boat with id 103.

  Solution.

$$\pi_{\texttt{sname}}(\sigma_{\texttt{bid}=103}(\texttt{Reserves}) \bowtie \texttt{Sailors})$$

  Also:

$$\rho(\texttt{Temp}_1, \sigma_{\texttt{bid}=103}(\texttt{Reserves}))$$
$$\rho(\texttt{Temp}_2, \texttt{Temp}_1 \bowtie \texttt{Sailors})$$
$$\pi_{\textsf{sname}}(\texttt{Temp}_2)$$

Examples IV

Note that

$$\pi_{\mathsf{sname}}(\sigma_{\mathtt{bid}=103}(\mathtt{Reserves}) \bowtie \mathtt{Sailors}) \tag{1}$$

and

$$\pi_{\mathsf{sname}}(\sigma_{\mathtt{bid}=103}(\mathtt{Reserves} \bowtie \mathtt{Sailors})) \tag{2}$$

gives the same result.
(2) is probably more efficient: Selection occurs earlier so that the join operation has a smaller left hand side parameter.
Query optimizer will choose the expression that optimizes the computation of the resulting relation.

Examples V

- Query 2. Names of sailors who reserved a red boat.

  Solutions:

  $$\pi_{\texttt{sname}}\left(\sigma_{\texttt{color}=\text{`}\texttt{red}\text{'}}(\texttt{Boats}) \bowtie \texttt{Reserves} \bowtie \texttt{Sailors}\right)$$

  or

  $$\pi_{\texttt{sname}}\left(\pi_{\texttt{sid}}\left(\sigma_{\texttt{color}=\text{`}\texttt{red}\text{'}}(\texttt{Boats}) \bowtie \texttt{Reserves}\right) \bowtie \texttt{Sailors}\right)$$

  Which is better?

Examples VI

- Query 2. Names of sailors who reserved a red boat.

  Solutions:

  $$\pi_{\texttt{sname}}\left(\sigma_{\texttt{color}=\text{'}\texttt{red}\text{'}}(\texttt{Boats}) \bowtie \texttt{Reserves} \bowtie \texttt{Sailors}\right)$$

  or

  $$\pi_{\texttt{sname}}\left(\pi_{\texttt{sid}}\left(\sigma_{\texttt{color}=\text{'}\texttt{red}\text{'}}(\texttt{Boats}) \bowtie \texttt{Reserves}\right) \bowtie \texttt{Sailors}\right)$$

  Which is better?

Examples VII

- Query 3. Colors of boats reserved by Lubber.

  Solutions:

$$\pi_{\texttt{color}} \left(\sigma_{\texttt{sname}=\texttt{'Lubber'}} \left(\texttt{Sailors} \right) \bowtie \texttt{Reserves} \bowtie \texttt{Boats} \right)$$

Examples VIII

- Query 4. Names of sailors who reserved at least one boat.

  Solutions:

  $$\pi_{\texttt{sname}}\left(\texttt{Sailors} \bowtie \texttt{Reserves}\right)$$

  Note: There are two `Sailors` tuples with `sname` $=$ ‘`Horatio`’ that have reserved a boat, but result is only one ‘`Horatio`’ since result is a relation, and relation is a set of tuples – no duplicates.
- Notice that joins are very common.

Examples IX

- Query 5. Names of sailors who reserved a red or green boat.

  Solution:

  $$\rho\left(T, \sigma_{\texttt{color}=\text{`}\texttt{red}\text{'}}(\texttt{Boats}) \cup \sigma_{\texttt{color}=\text{`}\texttt{green}\text{'}}(\texttt{Boats})\right)$$
  $$\pi_{\texttt{sname}}\left(T \bowtie \texttt{Reserves} \bowtie \texttt{Sailors}\right)$$

  or

  $$\rho\left(T, \sigma_{\texttt{color}=\text{`}\texttt{red}\text{'} \vee \texttt{color}=\text{`}\texttt{green}\text{'}}(\texttt{Boats})\right)$$
  $$\pi_{\texttt{sname}}\left(T \bowtie \texttt{Reserves} \bowtie \texttt{Sailors}\right)$$

Examples X

- Query 6. Names of sailors who reserved a red and a green boat.

  Solution:
  Tempting ...

  $$\rho(T, \sigma_{\texttt{color}=\texttt{'red'}}(\text{Boats}) \cap \sigma_{\texttt{color}=\texttt{'green'}}(\text{Boats})$$
  $$\pi_{\texttt{sname}}(T \bowtie \texttt{Reserves} \bowtie \texttt{Sailors})$$

  Wrong! T is relation of boats which are both red and green. This gives sailors s such that s reserved a red boat and s reserved a green boat – and there are no such boats. It does

Examples XI

not ask for s such that s reserved a boat that is both red and green (there are no such boats).
Therefore need to

- Find sailors who reserved a red boat
- Find sailors who reserved a green boat
- Intersect two relations above

Examples XII

But you have to be careful ...

$$\rho(T_1, \pi_{\texttt{sname}}(\sigma_{\texttt{color}=\texttt{`red'}}(\texttt{Boats}) \bowtie \texttt{Reserves} \bowtie \texttt{Sailors}))$$
$$\rho(T_2, \pi_{\texttt{sname}}(\sigma_{\texttt{color}=\texttt{`red'}}(\texttt{Boats}) \bowtie \texttt{Reserves} \bowtie \texttt{Sailors}))$$
$$T_1 \cap T_2$$

Wrong! What if there are two sailors, both names John, one who reserved a red boat and the other reserved a green boat.

Examples XIII

So need sid.

$$\rho(T_1, \pi_{\texttt{sid,sname}}(\sigma_{\texttt{color}=\text{`}\texttt{red}\text{'}}(\texttt{Boats}) \bowtie \texttt{Reserves} \bowtie \texttt{Sailors}))$$
$$\rho(T_2, \pi_{\texttt{sid,sname}}(\sigma_{\texttt{color}=\text{`}\texttt{green}\text{'}}(\texttt{Boats}) \bowtie \texttt{Reserves} \bowtie \texttt{Sailors}))$$
$$\pi_{\texttt{sname}}(T_1 \cap T_2)$$

Or

$$\rho(T_1, \pi_{\texttt{sid}}(\sigma_{\texttt{color}=\text{`}\texttt{red}\text{'}}(\texttt{Boats}) \bowtie \texttt{Reserves}))$$
$$\rho(T_2, \pi_{\texttt{sid}}(\sigma_{\texttt{color}=\text{`}\texttt{green}\text{'}}(\texttt{Boats}) \bowtie \texttt{Reserves}))$$
$$\pi_{\texttt{sname}}((T_1 \cap T_2) \bowtie \texttt{Sailors})$$

Examples XIV

- Query 7. Names of sailors who reserved at least two boats

  Solution: We more or less want a relation containing $\texttt{sid}, \texttt{bid1}, \texttt{bid2}$ such that sailor with $\texttt{sid}$ reserved boat with $\texttt{bid1}$ and boat with $\texttt{bid2}$. The following

  $$\rho(R_1(\texttt{sid} \to \texttt{sid}_1, \texttt{bid} \to \texttt{bid}_1), \pi_{\texttt{sid},\texttt{bid}}(\texttt{Sailors} \bowtie \texttt{Reserves}))$$
  $$\rho(R_2(\texttt{sid} \to \texttt{sid}_2, \texttt{bid} \to \texttt{bid}_2), \pi_{\texttt{sid},\texttt{bid}}(\texttt{Sailors} \bowtie \texttt{Reserves}))$$
  $$\sigma_{\texttt{sid}_1=\texttt{sid}_2 \wedge \texttt{bid}_1 \neq \texttt{bid}_2}(R_1 \times R_2)$$

  gives us a table with four columns where row has values $(\texttt{sid}_1, \texttt{sid}_2, \texttt{bid}_1, \texttt{bid}_2)$ denoting the fact for sailor with $\texttt{sid}_1$ $(= \texttt{sid}_2)$ reserved boats with $\texttt{bid}_1$ and $\texttt{bid}_2$.

Examples XV

We need the sailor name, so we include that in R_1:

$$\rho(R_1(\texttt{sid} \to \texttt{sid}_1, \texttt{bid} \to \texttt{bid}_1), \pi_{\texttt{sid,bid,sname}}(\texttt{Sailors} \bowtie \texttt{Reserves}))$$

$$\rho(R_2(\texttt{sid} \to \texttt{sid}_2, \texttt{bid} \to \texttt{bid}_2), \pi_{\texttt{sid,bid}}(\texttt{Sailors} \bowtie \texttt{Reserves}))$$

$$\pi_{\texttt{sname}}(\sigma_{\texttt{sid}_1=\texttt{sid}_2 \wedge \texttt{bid1} \neq \texttt{bid2}}(R_1 \times R_2))$$

Examples XVI

- Query 8. Find all sids of sailors with age over 20 who have not reserved a red boat.

  Solution: The `sid`s of sailors of age > 20 is

  $$\pi_{\texttt{sid}}(\sigma_{\texttt{age}>20}(\texttt{Sailors}))$$

  The `sid`s of sailors who reserved a red boat is

  $$\pi_{\texttt{sid}}(\sigma_{\texttt{color}=\text{`}\texttt{red}\text{'}}(\texttt{Boats}) \bowtie \texttt{Reserves})$$

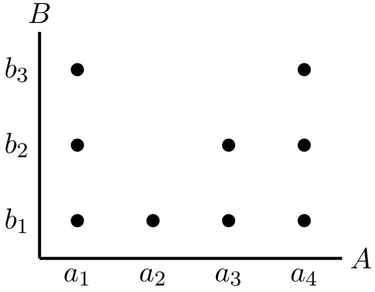
  Therefore the answer is

  $$\pi_{\texttt{sid}}(\sigma_{\texttt{age}>20}(\texttt{Sailors})) - \pi_{\texttt{sid}}(\sigma_{\texttt{color}=\text{`}\texttt{red}\text{'}}(\texttt{Boats}) \bowtie \texttt{Reserves})$$

Examples XVII

- Query 9. Names of sailors who reserved all boats.

  Solution: Look at this diagram ...



Examples XVIII

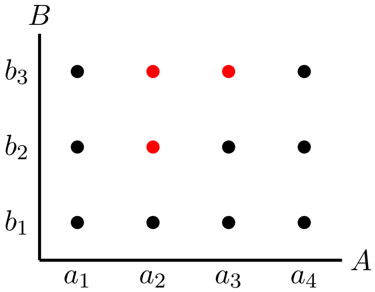
Think of A as the sailors (or their sids) and B as the boats (or their bids).

The set of black dots R represent the reservation relation. For instance the pair $(a_1, b_1) \in R$ represents the fact that sailor a_1 reserved b_1.

So a_1 and a_4 have each reserved all boats. We want to compute $\{a_1, a_4\}$.

Note the following: The complement of R, i.e., $A \times B - R$ is denoted by the red dots:

Examples XIX



Examples XX

Furthermore, when you project down to A you get $\{a_2, a_3\}$, which is the A–complement of what we are after. Therefore what we are after is

$$\{a_2, a_3\} = \pi_A(A \times B - R)$$

Back to our schema, the sids of those who reserved all boats is given by

$$\pi_{\texttt{sid}}(\pi_{\texttt{sid,bid}}(\texttt{Sailors} \times \texttt{Boats}) - \pi_{\texttt{sid,bid}}(\texttt{Reserves}))$$

Examples XXI

To make it easy to read:

$$\rho(A, \pi_{\texttt{sid}}(\texttt{Sailors})$$
$$\rho(B, \pi_{\texttt{bid}}(\texttt{Boats}))$$
$$\rho(R, \pi_{\texttt{sid,bid}}(\texttt{Reserves}))$$
$$\pi_{\texttt{sid}}(A \times B - R)$$

Examples XXII

But we need the snames. So we do this:

$$
\begin{aligned}
&\rho(A, \pi_{\texttt{sid}}(\texttt{Sailors})\\
&\rho(B, \pi_{\texttt{bid}}(\texttt{Boats}))\\
&\rho(R, \pi_{\texttt{sid,bid}}(\texttt{Reserves}))\\
&\pi_{\texttt{sname}}(\pi_{\texttt{sid}}(A \times B - R) \bowtie \texttt{Sailors})
\end{aligned}
$$

Note: If A and B are sets and R is a relation on $A \times B$ (i.e., $R \subseteq A \times B$), then the division A/B is defined as

$$A/B = \{a \,|\, (a, b) \in R \text{ for } \underline{\text{all}}\ b \in B\}$$

Examples XXIII

- Query 10. Names of all sailors who have reserved all boats named Interlake.

  Solution: We use the same idea from Query 9:

$$\rho(A, \pi_{\texttt{sid}}(\texttt{Sailors}))$$
$$\rho(B, \pi_{\texttt{bid}}(\sigma_{\texttt{bname}=\text{'Interlake'}}(\texttt{Boats})))$$
$$\rho(R, \pi_{\texttt{sid,bid}}(\texttt{Reserves}))$$
$$\pi_{\texttt{sname}}(\pi_{\texttt{sid}}(A \times B - R) \bowtie \texttt{Sailors})$$

Aggregate functions I

- **Extended relational algebra** – the standard relational algebra with the aggregate operators below. The aggregate operator depends on 5 aggregate functions.
  - SUM – sum
  - MAX – maximum
  - MIN – minimum
  - AVG – average
  - COUNT – number of tuples in group

Aggregate functions II

- Suppose R is a relation containing attribute A which is numeric (integer or real). Then

$$G_{\mathrm{SUM}(A)}(R)$$

is a relation with one attribute $\mathrm{SUM}(A)$ and one tuple which is the sum of all values in column A of R.

Aggregate functions III

- Example.

R

A	B
2	b
5	a
1	c
9	a

$\Longrightarrow$

$G_{\text{SUM}(A)}(R)$

SUM(A)
17

Aggregate functions IV

- You can group up tuples based on the values of attribute B.

R

A	B
2	b
5	a
1	c
9	a

$\Longrightarrow$

${}_B G_{\mathrm{SUM}(A)}(R)$

B	SUM(A)
b	2
a	14
c	1

Aggregate functions V

- You can group up tuples based on the values in attributes B and D and sum up the values of A.

R

A	B	C	D
2	b	...	x
3	b	...	y
5	a	...	x
1	c	...	z
7	b	...	x
9	a	...	y
2	a	...	x
3	b	...	x

$\Longrightarrow$

${}_{B,D}G_{\text{SUM}(A)}(R)$

B	D	SUM(A)
b	y	3
b	x	12
a	x	7
c	z	1
a	x	5
a	y	9

Aggregate functions VI

- You can sum up the values in two columns:

R

A	B	C
2	b	13
5	a	12
1	c	19
9	a	15

$\Longrightarrow$

$G_{\text{SUM}(A),\text{SUM}(C)}(R)$

SUM(A)	SUM(C)
17	59

Aggregate functions VII

- You can have multiple aggregate functions and group by multiple attributes

$$_{B,C}G_{\mathrm{SUM}(A),\mathrm{MAX}(D),\mathrm{SUM}(F)}(R)$$

where A, B, C, D, F are attributes of R and A, D are numeric. In this case, the resulting schema has 4 attributes:

  - B
  - C
  - $\mathrm{SUM}(A)$
  - $\mathrm{MAX}(D)$
  - $\mathrm{SUM}(F)$

Aggregate functions VIII

- The following are some other notations for $G_{\mathrm{SUM(A)}}(R)$:

$$F_{\mathrm{SUM(A)}}(R), \quad \mathcal{G}_{\mathrm{SUM(A)}}(R), \quad \gamma_{\mathrm{SUM(A)}}(R)$$

Identities I

- The following is an incomplete list of relational algebra identities.
- Set identities applies for $\cup, \cap, -, \times$. For instance

$$R \cup R' = R' \cup R$$

Identities II

- Selection: Let c, c' be predicates on attributes of R.

$$
\begin{aligned}
\sigma_c\left(\sigma_c(R)\right) &= \sigma_c(R) \\
\sigma_c\left(\sigma_{c'}(R)\right) &= \sigma_{c'}(\sigma_c(R)) = \sigma_{c' \wedge c}(R) \\
\sigma_c(R) \cap \sigma_{c'}(R) &= \sigma_{c \wedge c'}(R) \\
\sigma_c(R) \cup \sigma_{c'}(R) &= \sigma_{c \vee c'}(R) \\
\sigma_c(R) - \sigma_{c'}(R) &= \sigma_{c \wedge \neg c'}(R) \\
\sigma_c(R \cup R') &= \sigma_c(R) \cup \sigma_c(R') \\
\sigma_c(R \cap R') &= \sigma_c(R) \cap \sigma_c(R') \\
\sigma_c(R - R') &= \sigma_c(R) - \sigma_c(R')
\end{aligned}
$$

Identities III

Suppose c is a predicate on attributes in R and c' is a predicate on attributes in R' and c'' is a predicate

$$R \bowtie_{c \wedge c' \wedge c''} R' = \sigma_c(R) \bowtie_{c''} \sigma_{c'}(R')$$

Identities IV

- Projection: Let A, B be sets of attributes of R.

$$\begin{aligned} \pi_A\left(R \cup R'\right) &= \pi_A(R) \cup \pi_A(R') \\ \pi_A\left(\pi_A(R)\right) &= \pi_A(R) \\ \text{if } A \subseteq B, \pi_A\left(\pi_B(R)\right) &= \pi_A(R) \end{aligned}$$

- Selection and projection: If condition c only contains attributes in A, then

$$\pi_A\left(\sigma_c(R)\right) = \sigma_c(\pi_A(R))$$

- $R \cup S = S \cup R$ where R, S are relations with the same attributes.

Identities V

- $(R \cup S) \cup T = R \cup (S \cup T)$ where R, S, T are relations with the same attributes.
- $(R \times S) \times T = R \times (S \times T)$ where R, S, T are relations with the same attributes.
- $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$ if the joins are possible.