Databases-1 Lecture-01

Introduction, Relational Algebra

Information, 2018 Spring

- About me: Hajas Csilla, Mathematician, PhD, Senior lecturer, Dept. of Information Systems, Eötvös Loránd University of Budapest
- Databases-1 Lecture: Friday 10:15-11:45

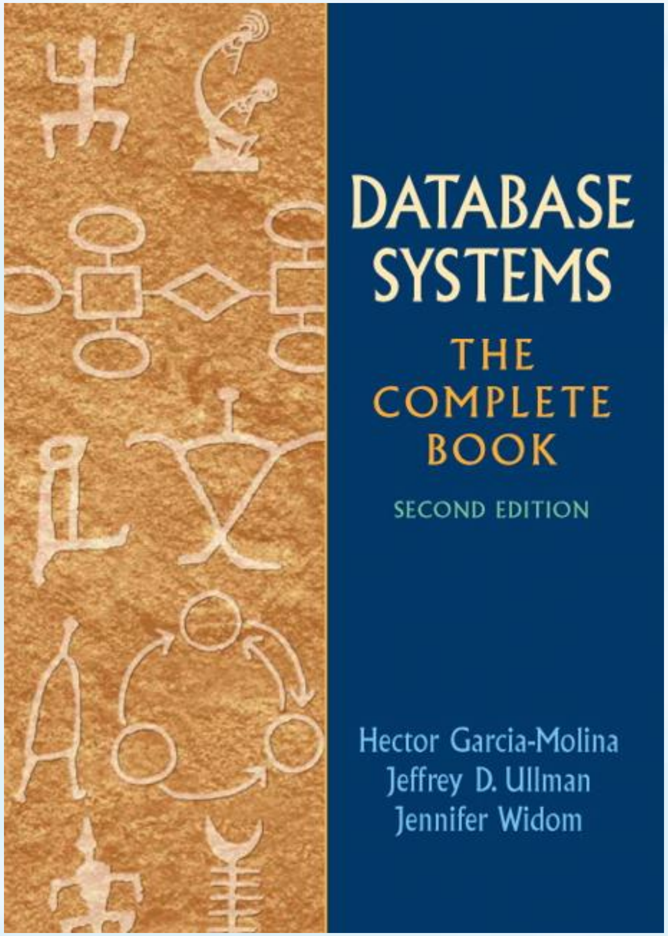
  ELTE South Building, 0-220 Karteszi Room
- Website of the course: http://sila.hajas.elte.hu/edu18feb/DB1L.html

Textbook

- A First Course in Database Systems (3rd ed.)
  by Jeff Ullman and Jennifer Widom

same material and sections as

- Database Systems: The Complete Book (2nd ed)
  by Garcia-Molina, Jeff Ullman and Jennifer Widom



Topics of the semester

- Relational Data Model
- Core and Extended Relational Algebra
- SQL Query and Modification
- Constraints, Triggers and Views
- PSM, Oracle PL/SQL
- Datalog, Recursion
- Entity-Relationship Model
- Design of Relational Databases

What is a Data Model?

- 1. Mathematical representation of data
- 2. Operation on data
- 3. Constraints

Relational Data Model

- A relation is a table

Attributes (column headers)

Beer	Manufacturer
Winterbrew	Pete's
Bud Lite	Anheuser-Busch

Tuples (rows)

Types and schemas

- *Relation schema* = relation name + attributes, in order (+ types of attributes).
  - Example: Beers(name, manf) or Beers(name: string, manf: string)
- *Database* = collection of relations.
- *Database schema* = set of all relation schemas in the database.

Why relations?

- Very simple model.
- *Often* matches how we think about data.
- Abstract model that underlies SQL, the most important database language today.

Relational model

- Logical level:
  - The relations are considered as tables.
  - The tables has unique names
  - The colums address the attributes
  - The rows represent the records
  - Rows can be interchanged, the order of rows is irrelevant
- Physical level:
  - The relations are stored in a file structure

Examples

Example 1

A	B	C
a	b	c
d	a	a
c	b	d

Example 2

B	C	A
b	c	a
a	a	d
d	d	c

Example 3

A	B	C
c	b	d
d	a	a
a	b	c

Example 4

A	B	C
c	b	d
c	b	d
a	b	c

In ex. 1 and ex. 2 the columns are interchanged but the same relation

In ex. 1 and ex. 3 the same tuples are represented in different orders but these are the same relations too.

Ex. 4 is not a relation

Defining a Database Schema

- A database schema comprises declarations for the relations (“tables”) of the database.
- Many other kinds of elements may also appear in the database schema, including views, constraints, triggers, indexes, etc.

Declaring a Relation

- Simplest form is:
  - CREATE TABLE <name> (<list of elements>);

```
CREATE TABLE Sells (
    bar       CHAR(20),
    beer      VARCHAR(20),
    price     REAL
);
```

Elements of Table Declarations

- The principal element is a pair consisting of an attribute and a type.
- The most common types are:
  - INT or INTEGER (synonyms).
  - REAL or FLOAT (synonyms).
  - CHAR(*n*) = fixed-length string of *n* characters.
  - VARCHAR(*n*) = variable-length string of up to *n* characters.
  - DATE is a type, and the form of a date value is:

    Example: ‘yyyy-mm-dd’ DATE ‘2002-09-30’

Example: Create Table

```
CREATE TABLE Sells (
      bar       CHAR(20),
      beer      VARCHAR(20),
      price     REAL
);
```

Other Declarations for Attributes

- Declaration for an attributes is a pair consisting of an attribute and a type.
- Other declarations we can make for an attribute are:
  1. NOT NULL means that the value for this attribute may never be NULL.
  2. DEFAULT <value> says that if there is no specific value known for this attribute's component in some tuple, use the stated <value>.

Example: Default Values

```
CREATE TABLE Drinkers (
    name CHAR(30)NOT NULL,
    addr CHAR(50)DEFAULT '3 Sesame St.',
    phone CHAR(16)
);
```

Effect of Defaults -- 1

- Suppose we insert the fact that Sally is a drinker, but we know neither her address nor her phone.
- An INSERT with a partial list of attributes makes the insertion possible:

```
INSERT INTO Drinkers(name)
VALUES('Sally');
```

Effect of Defaults -- 2

- But what tuple appears in Drinkers?

name	addr	phone
‘Sally’	‘123 Sesame St’	NULL

- If we had declared phone NOT NULL, this insertion would have been rejected.

Remove a relation from schema

- Remove a relation from the database schema by:
  - DROP TABLE <name>;

- Example:

```
DROP TABLE Sells;
```

Query Languages: Relational Algebra

- What is an “Algebra”?
- Mathematical system consisting of:
  - *Operands* --- variables or values from which new values can be constructed.
  - *Operators* --- symbols denoting procedures that construct new values from given values.

Core Relational Algebra

- Union, intersection, and difference.
  - Usual set operations, but require both operands have the same relation schema.
- Selection: picking certain rows.
- Projection: picking certain columns.
- Products and joins: compositions of relations.
- Renaming of relations and attributes.

Union, intersection, difference

- To apply these operators the relations must have the same attributes.
- Union ($R1 \cup R2$): all tuples from R1 <u>or</u> R2
- Intersection ($R1 \cap R2$): common tuples from R1 and R2
- Difference ($R1 \setminus R2$): tuples occuring in R1 but not in R2

Example

Relation Sells1:

Bar	Beer	Price
Joe’s	Bud	2.50
Joe’s	Miller	2.75
Sue’s	Bud	2.50

Relation Sells2:

Bar	Beer	Price
Joe’s	Bud	2.50
Jack’s	Bud	2.75

Sells1 $\cup$ Sells2:

Bar	Beer	Price
Joe’s	Bud	2.50
Joe’s	Miller	2.75
Sue’s	Bud	2.50
Jack’s	Bud	2.75

Sells1 $\cap$ Sells2:

Bar	Beer	Price
Joe’s	Bud	2.50

Sells2 $\setminus$ Sells1:

Bar	Beer	Price
Jack’s	Bud	2.75

Selection

- $R1 := \sigma_C (R2)$
  - *C* is a condition (as in "if" statements) that refers to attributes of R2.
  - R1 is all those tuples of R2 that satisfy *C*.

Example

Relation Sells:

Bar	Beer	Price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

JoeMenu := $\sigma_{bar=\text{"Joe's"}}$(Sells):

Bar	Beer	Price
Joe's	Bud	2.50
Joe's	Miller	2.75

Projection

- $R1 := \pi_L (R2)$
  - *L* is a list of attributes from the schema of R2.
  - R1 is constructed by looking at each tuple of R2, extracting the attributes on list *L*, in the order specified, and creating from those components a tuple for R1.
  - Eliminate duplicate tuples, if any.

Example

Relation Sells:

Bar	Beer	Price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

Prices := $\pi_{beer,price}$(Sells):

Beer	Price
Bud	2.50
Miller	2.75
Miller	3.00

Product

- R3 := R1 x R2
  - Pair each tuple t1 of R1 with each tuple t2 of R2.
  - Concatenation t1t2 is a tuple of R3.
  - Schema of R3 is the attributes of R1 and R2, in order.
  - But beware attribute *A* of the same name in R1 and R2: use R1.*A* and R2.*A*.

Example: R3=R1 x R2

- R1

A	B
1	2
3	4

- R2

B	C
5	6
7	8
9	10

R3=R1 x R2

A	R1.B	R2.B	C
1	2	5	6
1	2	7	8
1	2	9	10
3	4	5	6
3	4	7	8
3	4	9	10

Theta-Join

- R3 := R1 $\bowtie_C$ R2
  - Take the product R1 * R2.
  - Then apply σ_C to the result.
- As for σ, *C* can be any boolean-valued condition.
  - Historic versions of this operator allowed only A theta B, where theta was =, <, etc.; hence the name "theta-join."

Example

Sells:

Bar	Beer	Price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

Bars:

Name	Address
Joe's	Maple st.
Sue's	River rd.

Barinfo= Sells $\bowtie_{\text{Sells.bar = Bars.name}}$ Bars

Bar	Beer	Price	Name	Address
Joe's	Bud	2.50	Joe's	Maple st.
Joe's	Miller	2.75	Joe's	Maple st.
Sue's	Bud	2.50	Sue's	River rd.
Sue's	Miller	3.00	Sue's	River rd.

Natural Join

- A frequent type of join connects two relations by:
  - Equating attributes of the same name, and
  - Projecting out one copy of each pair of equated attributes.
- Called *natural* join.
- Denoted R3 := R1 ⋈ R2.

Example

Sells:

Bar	Beer	Price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

Bars:

Bar	Address
Joe's	Maple st.
Sue's	River rd.

Barinfo= Sells $\bowtie$ Bars

Bar	Beer	Price	Address
Joe's	Bud	2.50	Maple st.
Joe's	Miller	2.75	Maple st.
Sue's	Bud	2.50	River rd.
Sue's	Miller	3.00	River rd.

Renaming

- The RENAME operator gives a new schema to a relation.
- $R1 := \rho_{1(A1,\ldots,An)}(R2)$ makes R1 be a relation with attributes $A1,\ldots,An$ and the same tuples as R2.
- Simplified notation: $R1(A1,\ldots,An) := R2$.

Example

Bars:

Name	Address
Joe's	Maple st.
Sue's	River rd.

R(Bar, Address) := Bars

Bar	Address
Joe's	Maple st.
Sue's	River rd.

Building Complex Expressions

- Algebras allow us to express sequences of operations in a natural way
  - Example: in arithmetic --- (x + 4)*(y - 3).
- Relational algebra allows the same.
- Three notations, just as in arithmetic:
  1. Sequences of assignment statements.
  2. Expressions with several operators.
  3. Expression trees.

Sequences of Assignments

- Create temporary relation names.
- Renaming can be implied by giving relations a list of attributes.
- Example: $R3 := R1 \bowtie_C R2$ can be written:

  $R4 := R1 \times R2$

  $R3 := \sigma_C (R4)$

Expressions in a Single Assignment

- Example: the theta-join R3 := R1 $\bowtie_C$ R2 can be written: R3 := σ_C (R1 x R2)
- Precedence of relational operators:
  1. Unary operators --- select, project, rename --- have highest precedence, bind first.
  2. Then come products and joins.
  3. Then intersection.
  4. Finally, union and set difference bind last.
- But you can always insert parentheses to force the order you desire.

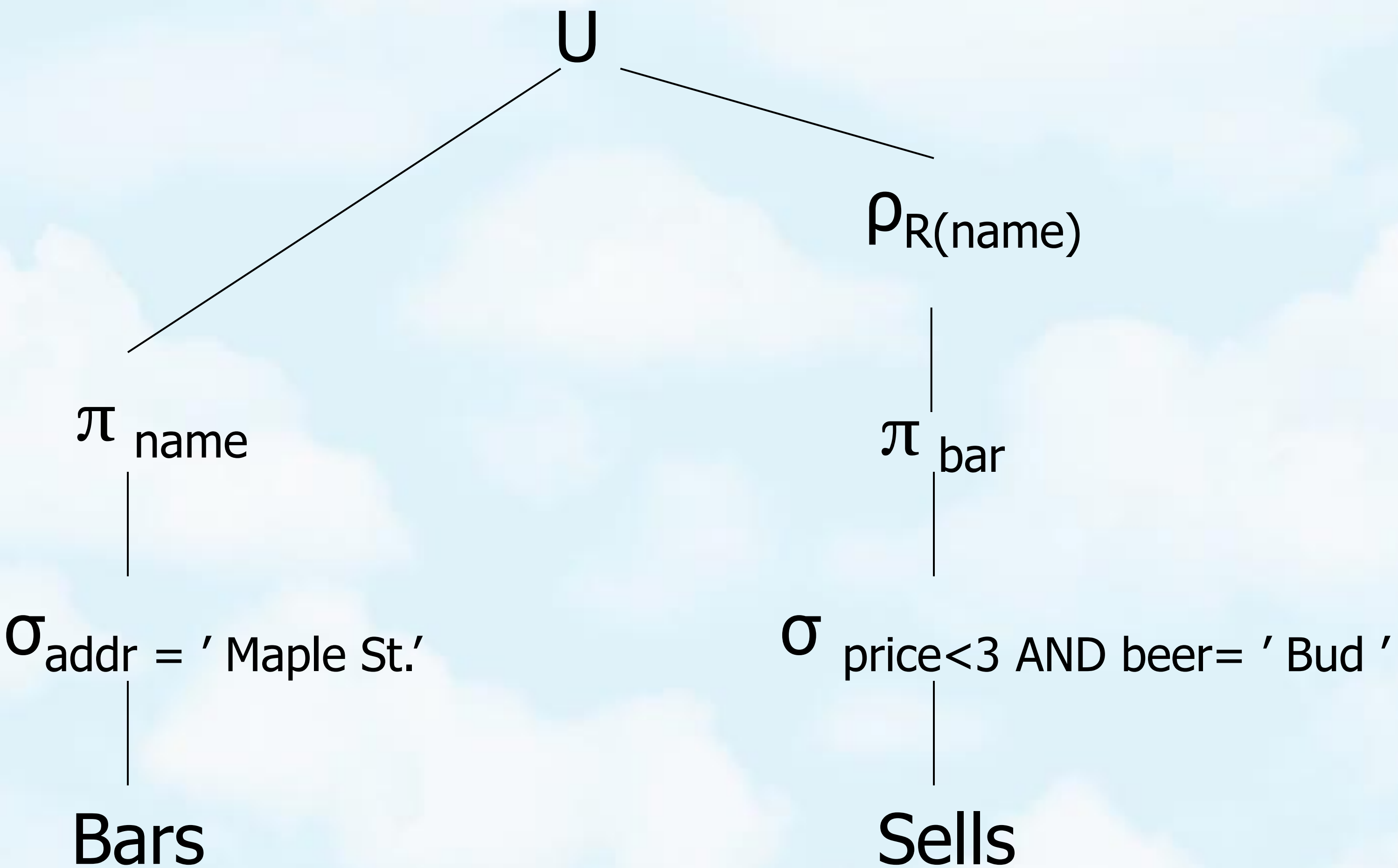
Expression Trees

- Leaves are operands --- either variables standing for relations or particular, constant relations.
- Interior nodes are operators, applied to their child or children.

Example

- Using the relations Bars(name, address) and Sells(bar, beer, price), find the names of all the bars that are either on Maple St. or sell Bud for less than $3.

As a Tree:


$\cup$
$\rho_{R(name)}$
π_{name}
π_{bar}
$\sigma_{addr\ =\ 'Maple\ St.'}$
$\sigma_{price<3\ AND\ beer=\ 'Bud'}$
Bars
Sells

Schema-Defining Rules

- For union, intersection, and difference, the schemas of the two operands must be the same, so use that schema for the result.
- Selection: schema of the result is the same as the schema of the operand.
- Projection: list of attributes tells us the schema.
- Product, Theta-join: the schema is the attributes of both relations.
  - Use R.*A*, etc., to distinguish two attributes named A.
- Natural join: use attributes of both relations.
  - Shared attribute names are merged.
- Renaming: the operator tells the schema.

Relational algebra: Monotonity

- Monotone non-decreasing expression:
  - applied on more tuples, the result contains more tuples
  - Formally if Ri $\subseteq$ Si for every i=1,…,n, then E(R1,…,Rn) $\subseteq$ E(S1,…,Sn).
- Difference is the only core expression which is not monotone:

A	B
1	0
2	1

\-

A	B
1	0

$\not\subseteq$

A	B
1	0
2	1

\-

A	B
1	0
2	1