BASIC DATABASES

Relational Model of Data

NGUYEN Hoang Ha

Email: nguyen-hoang.ha@usth.edu.vn

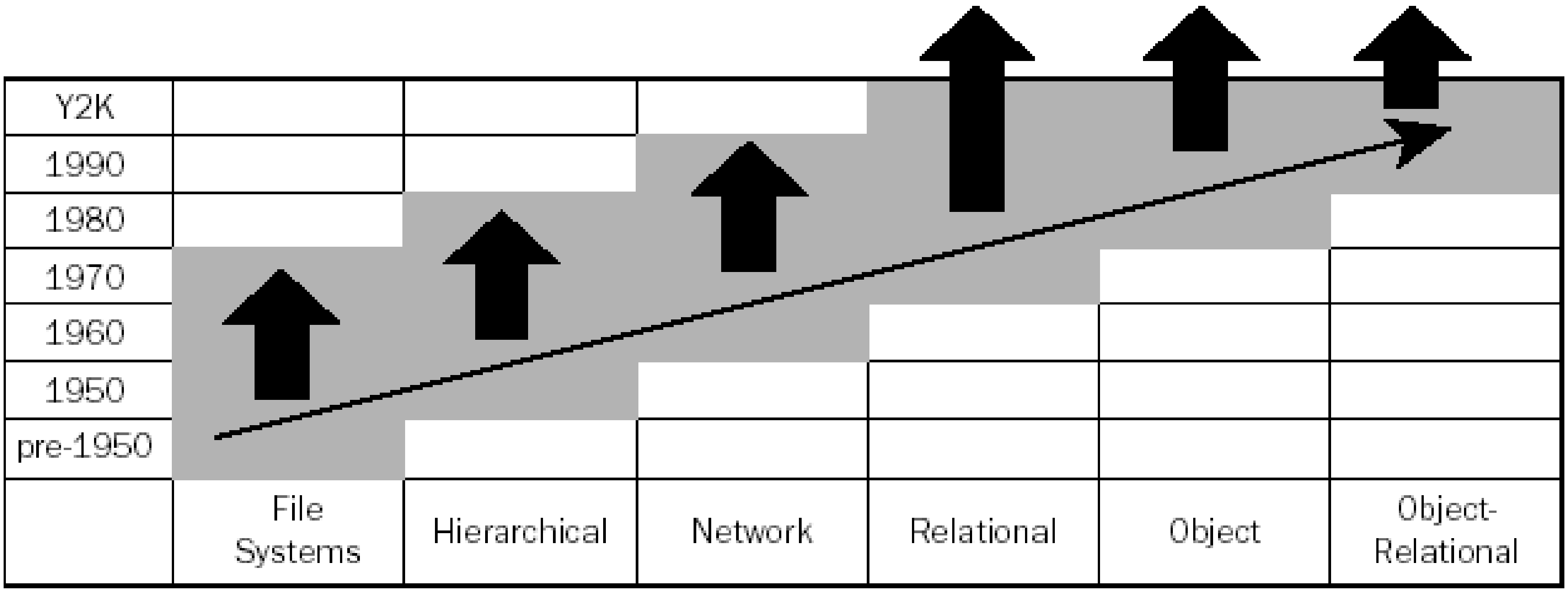
Objectives

- Understand concepts of
  - Data models
  - Relational data model
    - Structure
    - Operations
    - Constraints

What is a Data Model?

- Mathematical representation of data.
  - Relational model = tables;
  - Semi-structured model = trees/graphs.
- A model consists of:
  - Structure of data
  - Operations on data.
  - Constraints.

Data Models in history



The evolution of database modeling techniques.

Why Relation?

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

Reference: E. F. Codd, "A relational model for large shared data banks," Comm. ACM 13:6, pp. 377-387, 1970.

STRUCTURE OF RELATIONAL MODEL

Relational Model in brief

Attribute

Tuple

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

- The relational model represents data as a 2-dimensional table (called a relation)

Attributes

Attribute

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

- Each column represent a property of film and also called a "**attribute**": title, year, length, genre

Domains

- The relational model requires that each component of each tuple must be atomic, that is, it must be of some elementary type such as INTEGER or STRING
- It is **not** permitted for a value to be a record structure, set, list, array or any type that can have its values broken into smaller components
- A Domain is a particular elementary type of an attribute
  - More general: a set of values for an attribute

  → What are domains of title, year, length, genre?

Tuples

Tuple

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

- A row of a relation is called a tuple (or record)
  - Eg: Each row represents a film
- When we want to write a tuple in isolation, not as part of a relation, we normally use commas to separate components
  - Eg: (Star Wars, 1977, 124, sciFi)

Equivalent representation of a relation

- Relation is a **bag** (extended set) of tuples, not a list of tuples
  - Order of tuples is not important

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

=

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

  - There could be identical tuples

- <u>Question</u>:
  - What is/are the difference(s) between a set and a list?
  - What is/are the difference(s) between a set and a bag?

Relation instances

- A relation about Movies is not static but changing over time:
  - We want to insert tuples for new Movies as these appear
  - We want to edit existing tuples if we get corrected information about a Movies
  - We want to delete a tuple from the relation
- Sometime, we also want to add or delete attributes, this lead to the changing of the schema. So, a set of tuples for a given relation is called an instance of that relation

Relation instances example

R

A	B
1	2
3	4

R

A	B
1	3
2	5

R

A	B
3	5
4	6

Keys of relations

- A set of attributes forms a key for a relation if we don't allow 2 tuples in a relation instance to have the same values in all the attributes of the key
- Eg: The key of relation People(<u>ID</u>, name, address) is ID

Schemas

- Relation schema:
  - Name
  - Set of attributes.
    - Order of attributes is arbitrary. In practice we assume the order given in the relation schema
  - Other structure infor: keys…
  - Eg: Movies (<u>Id</u>, title, year, length, genre)
- Database schema: set of relation schema
  - Movies (<u>Id</u>, title, year, length, genre)
  - Producers (name, address, country)

DB Schema about Movies

```
Movies(
    title:string,
    year:integer,
    length:integer,
    genre:string,
    studioName:string,
    producerC#:integer
)
MovieStar(
    name:string,
    address:string,
    gender:char,
    birthdate:date
)
```

```
StarsIn(
    movieTitle:string,
    movieYear:integer,
    starName:string
)
MovieExec(
    name:string,
    address:string,
    cert#:integer,
    netWorth:integer
)
Studio(
    name:string,
    address:string,
    presC#:integer
)
```

OPERATIONS OF RELATIONAL MODEL

Relational Algebra (RA)

- Algebra is 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.
- Relational Algebra: an offshoot of algebra of **bags**
  - Operands:
    - Variables that stand for relations
    - Constants, which are finite relations
  - Operators work with relation(s) to create a new relation

18

Notations

Operation	My HTML	Symbol
Projection	PROJECT	π
Selection	SELECT	σ
Renaming	RENAME	ρ
Union	UNION	$\cup$
Intersection	INTERSECTION	$\cap$
Assignment	<-	$\leftarrow$

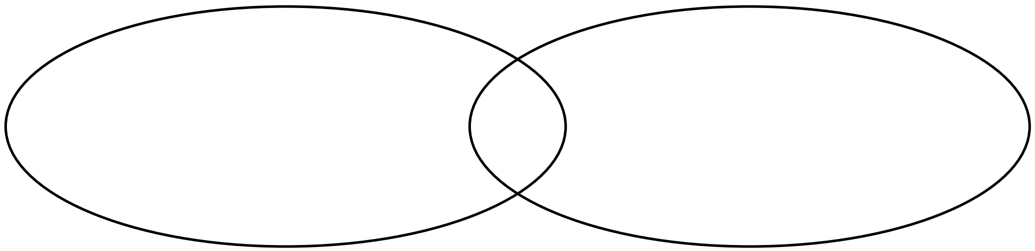
Operation	My HTML	Symbol
Cartesian product	X	$\times$
Join	JOIN	$\bowtie$
Left outer join	LEFT OUTER JOIN	⟕
Right outer join	RIGHT OUTER JOIN	⟖
Full outer join	FULL OUTER JOIN	⟗
Semijoin	SEMIJOIN	$\ltimes$

Primitive operations

- In any algebra, some operators are primitive and the others, being definable in terms of the primitive ones, are derived
- The six primitive operators of relational algebra are:
  - the SELECTION,
  - the PROJECTION,
  - the CARTESIAN PRODUCT (also called the *cross product* or *cross join*),
  - the SET UNION,
  - the SET DIFFERENCE, and
  - the RENAME

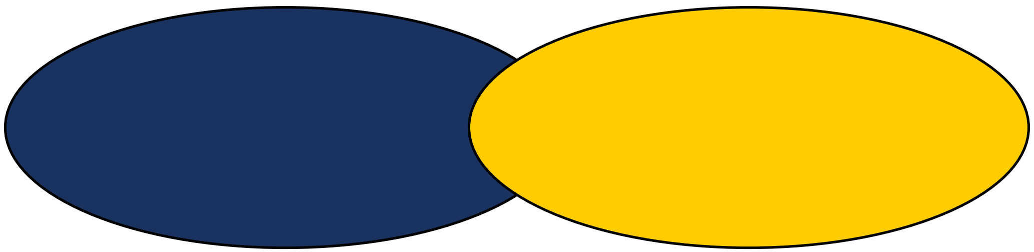
Operations on Relations - Bag Union

- Example: {1,2,1} U {1,1,2,3,1} = {1,1,1,1,1,2,2,3}



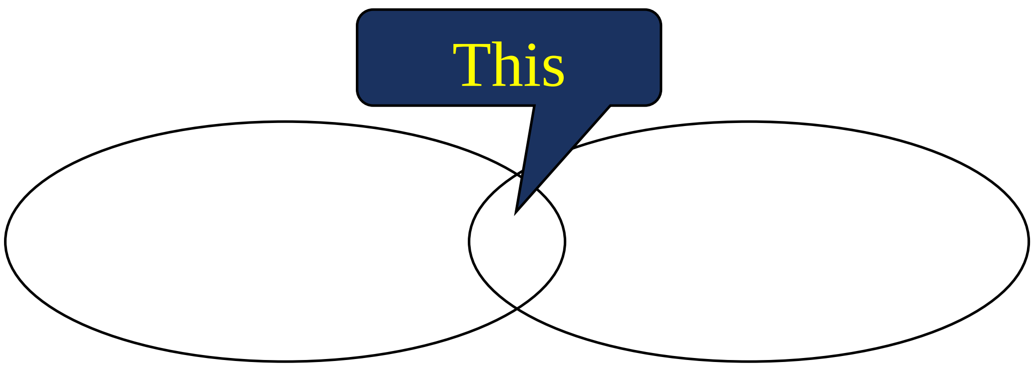
Operations on Relations - Bag Difference

- Example: {1,2,1,1} – {1,2,3} = {1,1}.



Set Operations on Relations - Bag Intersection

- Example: {1,2,1,1} $\cap$ {1,2,1,3} = {1,1,2}.



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.

Projection 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

Extended Projection

- Using the same π_L operator, we allow the list L to contain arbitrary expressions involving attributes:
  - Arithmetic on attributes, e.g., A+B → C.
  - Duplicate occurrences of the same attribute.

R =

A	B
1	2
3	4

$\pi_{A+B\text{->}C,\ A,\ A}$ (R) =

C	A1	A2
3	1	1
7	3	3

Exercises - 1

- Suppose relation R(A,B,C) has the tuples:

A	B	C
1	2	3
4	2	3
4	5	6
2	5	3
1	2	6

- Compute the projection $\pi_{C,B}(R)$

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.

Selection 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

Renaming

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

Renaming Example

Bars(

name,	addr
Joe's	Maple St.
Sue's	River Rd.

)

R(bar, addr) := Bars

R(

bar,	addr
Joe's	Maple St.
Sue's	River Rd.

)

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 then R2, in order.
  - But beware attribute *A* of the same name in R1 and R2: use R1.*A* and R2.*A*.

R3 = R1 X R2

R1(A, B)

A,	B
1	2
3	4

R2(B, C)

B,	C
5	6
7	8
9	10

R3(A, R1.B, R2.B, C)

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

R3 = R1 X R2

The table **E** (for **EMPLOYEE**)

enr	ename	dept
1	Bill	A
2	Sarah	C
3	John	A

The table **D** (for **DEPARTMENT**)

dnr	dname
A	Marketing
B	Sales
C	Legal

<table>
<tr><th>SQL</th><th>Result</th><th>Relational algebra</th></tr>
<tr><td><code>select * from E, D</code></td><td>
<table>
<tr><th>enr</th><th>ename</th><th>dept</th><th>dnr</th><th>dname</th></tr>
<tr><td>1</td><td>Bill</td><td>A</td><td>A</td><td>Marketing</td></tr>
<tr><td>1</td><td>Bill</td><td>A</td><td>B</td><td>Sales</td></tr>
<tr><td>1</td><td>Bill</td><td>A</td><td>C</td><td>Legal</td></tr>
<tr><td>2</td><td>Sarah</td><td>C</td><td>A</td><td>Marketing</td></tr>
<tr><td>2</td><td>Sarah</td><td>C</td><td>B</td><td>Sales</td></tr>
<tr><td>2</td><td>Sarah</td><td>C</td><td>C</td><td>Legal</td></tr>
<tr><td>3</td><td>John</td><td>A</td><td>A</td><td>Marketing</td></tr>
<tr><td>3</td><td>John</td><td>A</td><td>B</td><td>Sales</td></tr>
<tr><td>3</td><td>John</td><td>A</td><td>C</td><td>Legal</td></tr>
</table>
</td><td>E X D</td></tr>
</table>

Join

- Types
  - Theta join
  - Natural join
  - Outer join:
    - Left outer join
    - Right outer join
    - Full outer join

Theta-Join

- R3 := R1 $\bowtie_C$ R2
  - Take the product R1 X 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 θ B, where θ is =, <, etc.; hence the name "theta-join."

36

Theta-Join Example

Sells(bar, beer, price)

bar	beer	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

Bars(name, addr)

name	addr
Joe's	Maple St.
Sue's	River Rd.

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

BarInfo(bar, beer, price, name, addr)

bar	beer	price	name	addr
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	Coors	3.00	Sue's	River Rd.

Natural Join

- A useful join variant (*natural* join) connects two relations by:
  - Equating attributes of the same name, and
  - Projecting out one copy of each pair of equated attributes.
- Denoted $R3 := R1 \bowtie R2$.

Eg: Natural Join

Sells(

bar,	beer,	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

)

Bars(

bar,	addr
Joe's	Maple St.
Sue's	River Rd.

)

BarInfo := Sells $\bowtie$ Bars

Note: Bars.name has become Bars.bar to make the natural join "work."

BarInfo(

bar,	beer,	price,	addr
Joe's	Bud	2.50	Maple St.
Joe's	Milller	2.75	Maple St.
Sue's	Bud	2.50	River Rd.
Sue's	Coors	3.00	River Rd.

)

Inner Join

The table **E** (for **EMPLOYEE**)

enr	ename	dept
1	Bill	A
2	Sarah	C
3	John	A

The table **D** (for **DEPARTMENT**)

dnr	dname
A	Marketing
B	Sales
C	Legal

<table>
<tr><th>SQL</th><th>Result</th><th>Relational algebra</th></tr>
<tr>
<td><code>select *
from E, D
where dept = dnr</code></td>
<td>
<table>
<tr><th>enr</th><th>ename</th><th>dept</th><th>dnr</th><th>dname</th></tr>
<tr><td>1</td><td>Bill</td><td>A</td><td>A</td><td>Marketing</td></tr>
<tr><td>2</td><td>Sarah</td><td>C</td><td>C</td><td>Legal</td></tr>
<tr><td>3</td><td>John</td><td>A</td><td>A</td><td>Marketing</td></tr>
</table>
</td>
<td>$\textbf{SELECT}_{\text{dept = dnr}}\ (\text{E X D})$
or, using the equivalent join operation
$\text{E }\textbf{JOIN}_{\text{dept = dnr}}\ \text{D}$</td>
</tr>
</table>

Outer join

The table **E** (for **EMPLOYEE**)

enr	ename	dept
1	Bill	A
2	Sarah	C
3	John	A

The table **D** (for **DEPARTMENT**)

dnr	dname
A	Marketing
B	Sales
C	Legal

<table>
<tr><th>SQL</th><th>Result</th><th>Relational algebra</th></tr>
<tr><td><pre>select *
from (E right outer join D on edept = dnr)</pre></td><td>
<table>
<tr><th>enr</th><th>ename</th><th>dept</th><th>dnr</th><th>dname</th></tr>
<tr><td>1</td><td>Bill</td><td>A</td><td>A</td><td>Marketing</td></tr>
<tr><td>2</td><td>Sarah</td><td>C</td><td>C</td><td>Legal</td></tr>
<tr><td>3</td><td>John</td><td>A</td><td>A</td><td>Marketing</td></tr>
<tr><td>null</td><td>null</td><td>null</td><td>B</td><td>Sales</td></tr>
</table>
</td><td>E RIGHT OUTER JOIN$_{\text{edept = dnr}}$ D</td></tr>
</table>

Building Complex Expressions

- Combine operators with parentheses and precedence rules.
- Three notations, just as in arithmetic:
  - Sequences of assignment statements.
  - Expressions with several operators.
  - Expression trees.

42

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 X R2

  R3 := σ_C (R4)

43

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. [σ, π, ρ] (highest).
  2. [x, $\bowtie$].
  3. $\cap$.
  4. [$\cup$, —]

44

Expression Trees

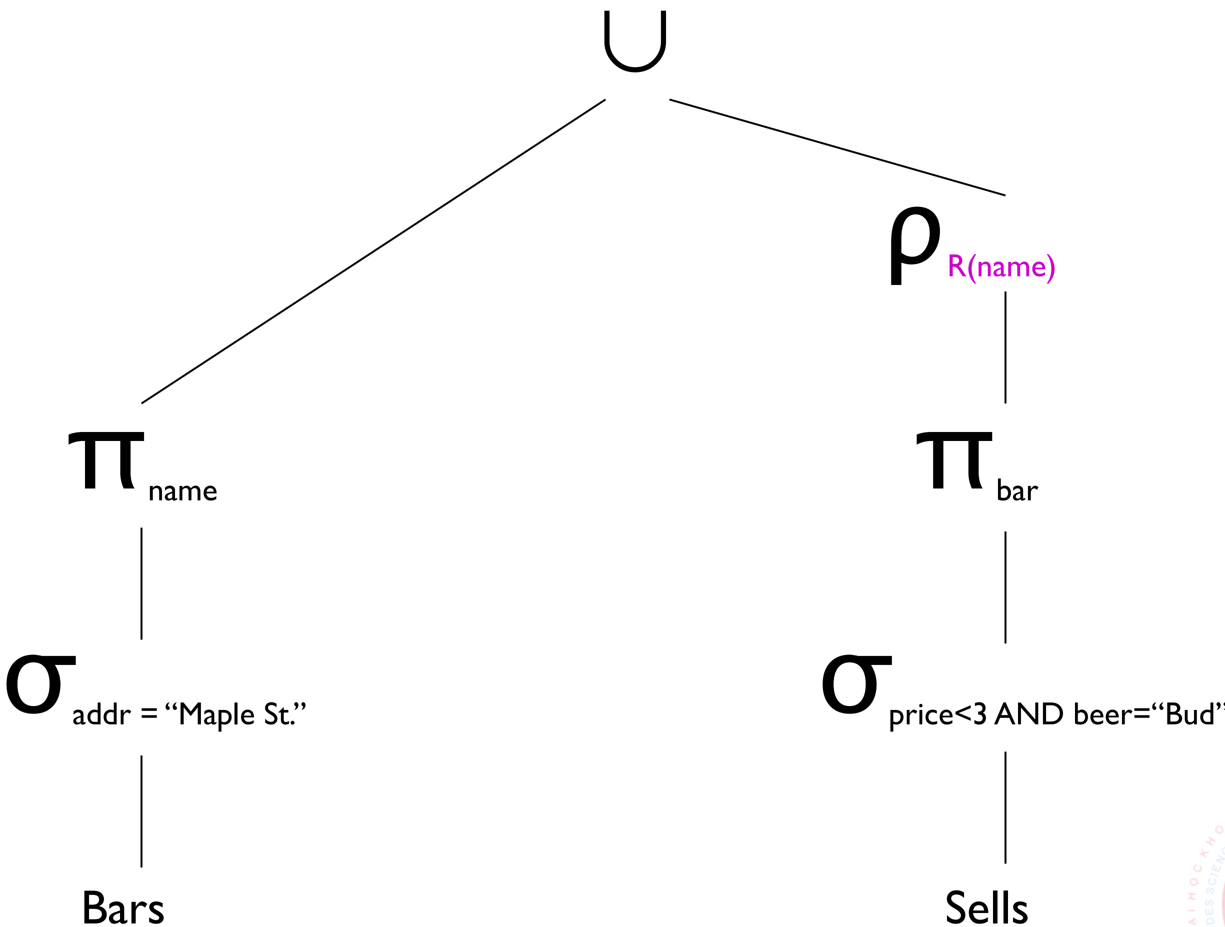
- Leaves are operands --- either variables standing for relations or particular, constant relations.
- Interior nodes are operators, applied to their child or children.

Example: Tree for a Query

- Using the relations Bars(name, addr) 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.

46

As a Tree:


∪
ρ R(name)
π name
π bar
σ addr = "Maple St."
σ price<3 AND beer="Bud"
Bars
Sells

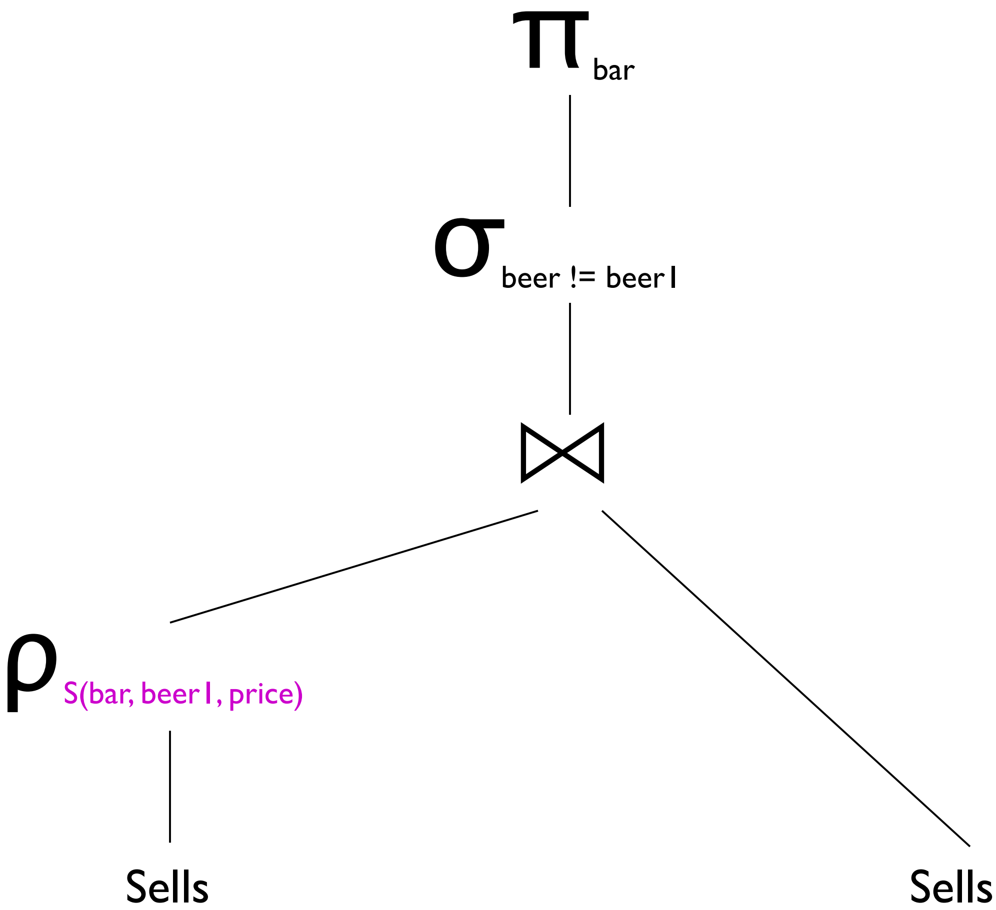
47

Example: Self-Join

- Using Sells(bar, beer, price), find the bars that sell two different beers at the same price.
- Strategy: by renaming, define a copy of Sells, called S(bar, beer1, price). The natural join of Sells and S consists of quadruples (bar, beer, beer1, price) such that the bar sells both beers at this price.

48

The Tree

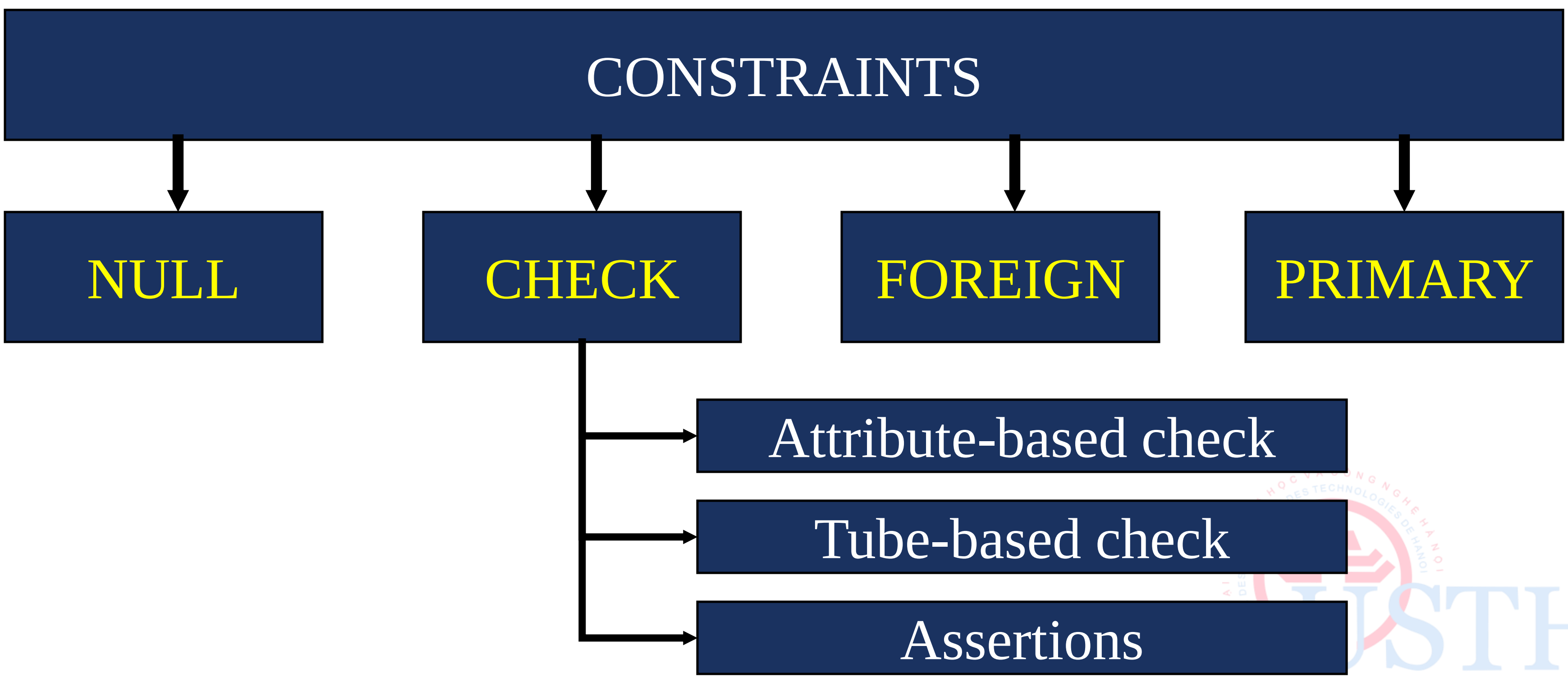


49

CONSTRAINS IN RELATIONAL MODEL

Introduction

- A *constraint* is a relationship among data elements that the DBMS is required to enforce.
- → mechanism that may be used to limit the values entered into a relation.



Key of a relation

- A set attributes $\{A_1, A_2, .., A_n\}$ is called a **key** of the relation R if:
  - 1. Those attributes determine all other attributes.
    → It is impossible for 2 tuples of R to agree on all of $\{A_1, A_2, .., A_n\}$
  - 2. No subset of $\{A_1, A_2, .., A_n\}$ determines all other attributes
  - → A Key must be minimal
- Example
  - AccademicResults (<u>StudentID</u>, <u>SubjectID</u>, grade, comment)

Different key types of a relation

- Super key
  - Set of attributes (columns) to uniquely identify rows
- Key or candidate key
  - Minimal super key
- Primary key
  - One selected from candidate keys
- Alternate key
  - Candidate key other than PK
- Foreign key
  - Attribute refers to a PK of another relation

Express Primary Key Constraints by RA

- Suppose we have a schema $R(A_1,A_2,A_3)$
- So, we have A_1 is the primary key of R if:

$$\delta_{R1.A1=R2.A1 \textbf{ AND } R1.A2<>R2.A2 \textbf{ AND } R1.A3 <> R2.A3}(R \times R) = \Phi$$

Example: Express Primary Key Constraints

Order ID	Customer
10248	Vins et alcools Chevalier
10249	Toms Spezialitäten
10250	Hanari Carnes
10251	Victuailles en stock
10252	Suprêmes délices
10253	Hanari Carnes
10254	Chop-suey Chinese
10255	Richter Supermarkt
10256	Wellington Importadora
10257	HILARIÓN-Abastos
10258	Ernst Handel
10259	Centro comercial Moctezuma
10260	Ottilies Käseladen
10261	Que Delícia
10262	Rattlesnake Canyon Grocery
10263	Ernst Handel

- $\delta_{\text{R1.OrderId = R2.OrderId } \textbf{AND} \text{ R1.Customer<>R2.Customer}}(R \times R) = \Phi$

Express Foreign Key Constraints

- Suppose we have 2 relations R and S.
  R contains an attribute A
  S contains an attribute B

- "R references to S via A and B" if: $\prod_A(R) - \prod_B(S) = \Phi$

Example: Express Foreign Key Constraints

Sells(

bar,	beer,	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Coors	3.00

)

Bars(

bar,	addr
Joe's	Maple St.
Sue's	River Rd.

)

- $\Pi_{Bar}(Sells) - \Pi_{Bar}(Bars) = \Phi$

Express Check Constraints

- Give a schema:
  EMPLOYEE(EmpId, Name, Sex)

- If we want to specify that the only legal values for SEX attribute are ‘F’ and ‘M’, we can write:

  $$\text{SELECT}_{\text{sex} <> \text{'F' and sex} <> \text{'M'}}(\text{EMPLOYEE}) = \Phi$$

Exercises 1

- Give a schema:
  PC (Model, Speed, RAM, HDD, Price)

- Use Relational Algebra to express following constraints:

  A PC with a processor speed less than 3.00 must not sell for more than $800

Exercises 2

- Give a schema:
  LAPTOP (Model, speed, RAM, HDD, Screen, Price)

- Use Relational Algebra to express following constraints:

  A laptop with a screen size less than 15.4 inches must have at least a 120GB hard disk or sell for less than $1000

Exercises 3

- PC (Model, Speed, RAM, HDD, Price)
- LAPTOP (Model, speed, RAM, HDD, Screen, Price)

- Use Relational Algebra to express following constraints:

  If a laptop has a larger RAM than a PC, then the laptop must also have a higher price than the PC

Exercises 4

- PC (Maker, Model)
- PRINTER (Maker, Model)

- Use Relational Algebra to express following constraints:

  No manufacturer of PC's may also make printers

Exercises 5

- PRODUCT (Maker, model)
- PC (Model)
- PRINTER (Model)

- Use Relational Algebra to express following constraints:

  No manufacturer of PC's may also make printers

Exercises 6

- PC (Model, Speed, RAM, HDD, Price)
- Use Relational Algebra to express following constraints:
  - Higher model, higher price
  - With the same model, higher Speed and RAM and HDD, higher price