# Building complex expressions

- Our relational-model queries may become quite complex
- Readability is improved by building expression up in a manner similar to our notation used in and around programming
- Combine operators with parentheses and precedence rules
- Three approaches possible:
  1. **Sequences of assignment statements**
  2. **Expressions with several operators**
  3. **Expressions trees**

# 1. Sequences of Assignments

- We've seen this used informally already
- Principle: Create temporary relation names
- Renaming can be implied by giving relations a list of attributes
- Example:

$$R3 = R1 \bowtie_C R2$$

can be written as:

$$R4 = R1 \times R2$$
$$R3 = \sigma_c(R4)$$

## 2. Expressions in a single assignment

- Instead of one operator per assignment, we combine all operators in a single expression
- To evaluate such an expression, we must be clear about operator precedence
- Example:

  $$R3 = R1 \bowtie_C R2$$

  can be written as:

  $$R3 = \sigma_c(R1 \times R2)$$

- Precedence: highest to lowest
  - $[\sigma, \pi, \rho]$
  - $[\times, \bowtie]$
  - $\cap$
  - $[\cup, -]$

# 3. Expression Trees

- **Leaves = operands**
  - Variables standing in for relations
  - Constant-value relations
- **Interior nodes = operators/operations**
  - Operators are applied to children of node
  - In effect, the node will evaluate to a relation
- Expression trees are similar to those used to visualize operator precedence in Java or C
  - Order of an operation is clearly indicated by its position in the tree
  - Tree eliminates any ambiguity

# 3. Expression Trees: Example 1

- Using the two relations **Pub(name, addr, URL)** and **Sells(pub, beer, price)**:
  - Find the names of all the pubs that are either on Fort on sell "Blue Buck" for less than $3
- Even before drawing an expression tree:
  - What is our strategy (denoted using either of the two previous schemes)?

# 3. Expression Trees: Example 1

**Recall....**

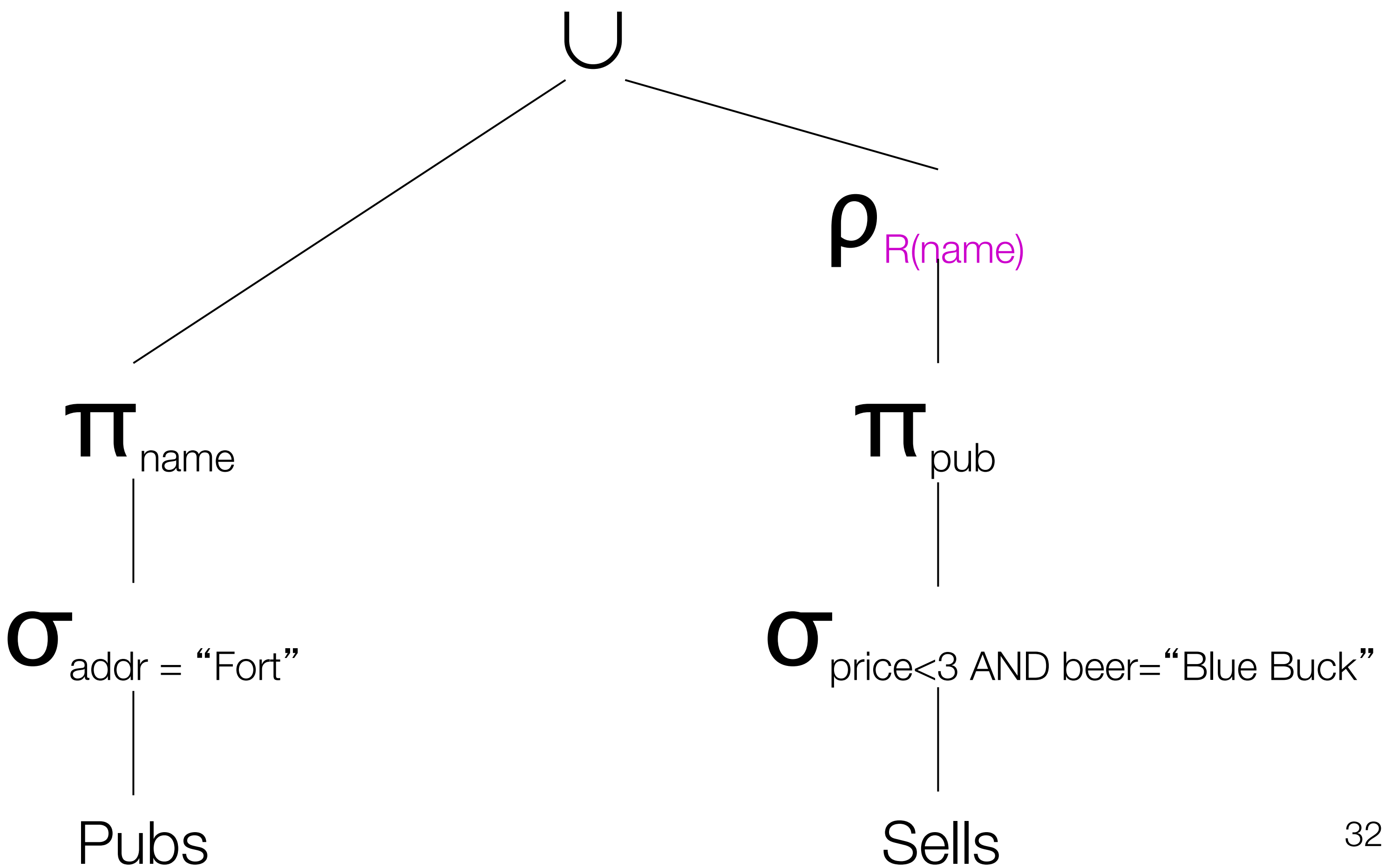
Sells

| pub | beer | price |
|---|---|---|
| Rob's | Amnesiac | 7.50 |
| Rob's | Blue Buck | 3.25 |
| Pat's | Amnesiac | 7.50 |
| Pat's | Blue Buck | 2.95 |

Pubs

| name | addr | URL |
|---|---|---|
| Rob's | Fort | http://robsplace.com |
| Pat's | Broughton | http://patspub.ca |

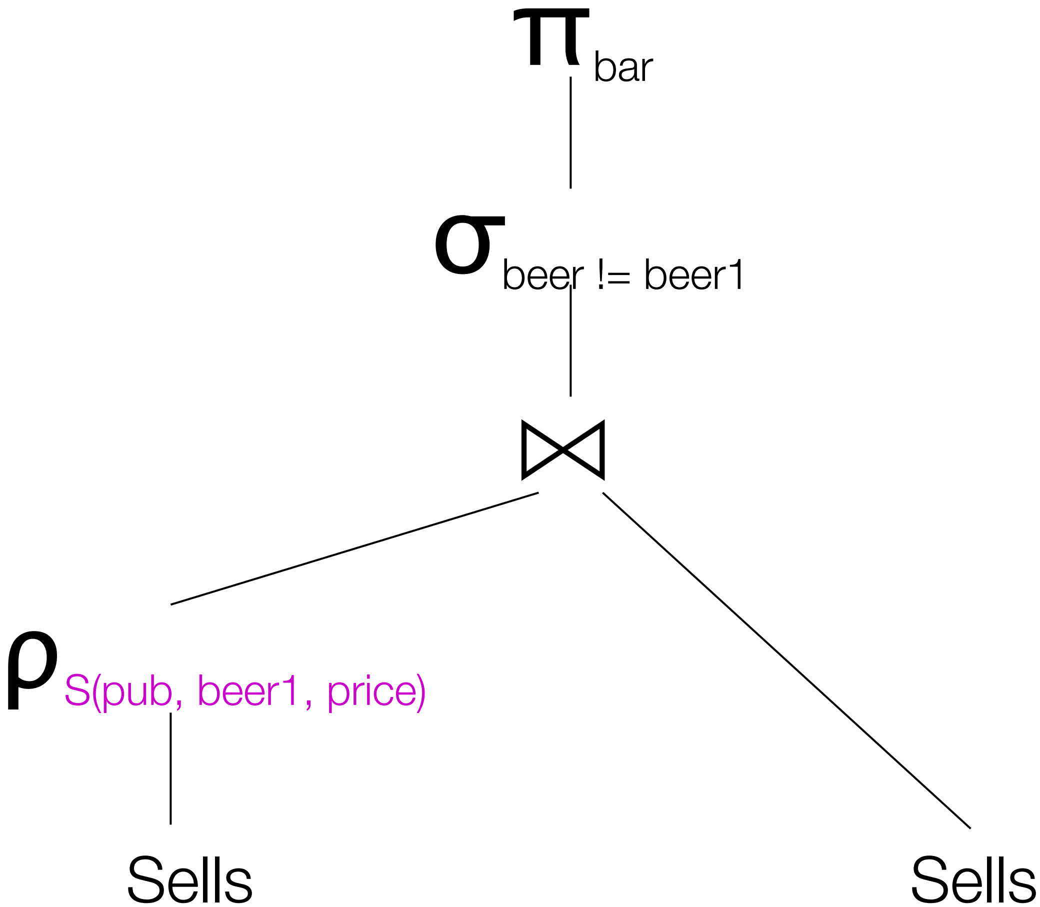
# 3. Expression Trees: Example 1



# 3. Expression Trees: Example 2

- Using the relation **Sells(pub, beer, price)**:
  - Find the pubs that sell two different beers at the same price
- Even before drawing an expression tree:
  - What is our strategy?
  - Hint: Could use something that looks like a "self join"

# 3. Expression Trees: Example 2



# Schemas for results: a recap

- $\cup$, $\cap$, $-$
  - Schemas of two operands must be the same
  - Use that schema for the result
- $\sigma$
  - Schema of the result is same as schema of the operand
- $\pi$
  - List of attributes tells us the schema

# Schemas for results: summary

- $\times$
  - Schema consists of the attributes of both relations
  - We distinguish between two attributes having the same name by prefixing with a source relation (e.g., R.A for A, etc.)
- $\bowtie_C$
  - Same as for product
- $\bowtie$
  - Union of the attributes of the two relations
- $\rho$
  - The operator's list argument gives us the schema

# Relational Algebra on Bags

- Bag
  - Also called a multiset
  - Is like a set, yet an element may appear more than once in a bag
- Example of a bag:
  - {1, 2, 1, 3}
- Example of a bag that is also a set:
  - {1, 2, 3}
- Why???
  - SQL is actually a bag language
  - Some operations (e.g., \pi) are more efficient on bags than on sets

# Operations on bags

- $\sigma$
  - Applies to each tuple
  - Therefore effect on bags is like its effect on sets
- $\pi$
  - Also applies to each tuple
  - However, when used as a bag operator, **duplicates are not eliminated**
- $\times$, $\bowtie$
  - Performed on each pair of tuples
  - Duplicates in bags will have no effect on the way we perform these operations

# Example: bag selection

R

| A | B |
|---|---|
| 1 | 2 |
| 5 | 6 |
| 1 | 2 |

$\sigma_{A+B<5}(R)$

| A | B |
|---|---|
| 1 | 2 |
| 1 | 2 |

# Example: bag projection

R

| A | B |
|---|---|
| 1 | 2 |
| 5 | 6 |
| 1 | 2 |

$\pi_A(R)$

| A |
|---|
| 1 |
| 5 |
| 1 |

# Example: bag theta-join

R

| A | B |
|---|---|
| 1 | 2 |
| 5 | 6 |
| 1 | 2 |

S

| B | C |
|---|---|
| 3 | 4 |
| 7 | 8 |

$T = R \bowtie_{R.B<S.B} S$

| A | R.B | S.B | C |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 1 | 2 | 7 | 8 |
| 5 | 6 | 7 | 8 |
| 1 | 2 | 3 | 4 |
| 1 | 2 | 7 | 8 |

# Operations on bags: union & intersection

- An element appears in the **union** of two bags **the sum of the number of times it appears** in each bag
  - Example:
    $\{1, 2, 1\} \cup \{1, 1, 2, 3, 1\} = \{1, 1, 1, 1, 1, 2, 2, 3\}$
- An element appears in the **intersection** of two bags **the minimum of the number of times it appears** in either bag
  - Example:
    $\{1, 2, 1\} \cap \{1, 2, 1, 3\} = \{1, 1, 2\}$

# Operations on bags: difference

- An element appears in the **difference** of A-B
  - **as many times as it appears** in A...
  - **minus the number of times it appears** in B.
  - However, elements never appear less than zero times.
  - Examples:

    {1, 2, 1} — {1, 1, 2, 3, 1} = {}

    {1, 2, 1, 1} — {1, 2, 3} = {1, 1}

# **Achtung** : Bag Laws ≠ Set Laws

- Set union is **idempotent**
  - Means: $S \cup S = S$
- However, bag union is not idempotent
  - If "1" appears n times in S...
  - ... then it appears 2n times in $S \cup S$
- Thus $S \cup S \neq S$ in general
  - Example: $\{1\} \cup \{1\} = \{1,1\} \neq \{1\}$

# Summary

- Operators in relational algebra:
  - Selection, projection, renaming
  - Set operations
  - Joins
- Some extended operations
- Techniques for building up complicated expressions
- Bag version of operations