Why Decomposition “Works”?  
• What does it mean to “work”? Why can’t we just tear sets of attributes apart as we like?  
• Answer: the decomposed relations need to represent the same information as the original.  
  – We must be able to reconstruct the original from the decomposed relations.  
• Projection and join connect the original and decomposed relations.
**Example**

*Drinkers3* \( \bowtie \) *Drinkers4 =*

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>beersLiked</th>
<th>manf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Bud</td>
<td>A.B.</td>
<td></td>
</tr>
<tr>
<td>Mike</td>
<td>Blonde Ale</td>
<td>G.I.</td>
<td></td>
</tr>
<tr>
<td>Anna</td>
<td>BudLite</td>
<td>A.B.</td>
<td></td>
</tr>
</tbody>
</table>

*• Join of above with Drinkers1 = original R.*

**Theorem**

*• Suppose we decompose a relation with schema XYZ into XY and XZ and project the relation for XYZ onto XY and XZ. Then XY \( \rightarrow \) XZ is **guaranteed** to reconstruct XYZ if and only if X \( \rightarrow \rightarrow \) Y (or equivalently, X \( \rightarrow \rightarrow \) Z).

*• Usually, the MVD is really a FD, X \( \rightarrow \) Y or X \( \rightarrow \) Z.*

**Implications**

*• BCNF: When we decompose XYZ into XY and XZ, it is because there is a FD X \( \rightarrow \) Y or X \( \rightarrow \) Z that violates BCNF.

  – Thus, we can always reconstruct XYZ from its projections onto XY and XZ.

*• 4NF: when we decompose XYZ into XY and XZ, it is because there is an MVD X \( \rightarrow \rightarrow \) Y or X \( \rightarrow \rightarrow \) Z that violates 4NF.

  – Again, we can reconstruct XYZ from its projections onto XY and XZ.*

**Bag Semantics**

*• A relation (in SQL, at least) is really a bag.

  • It may contain the same tuple more than once, although there is no specified order (unlike a list).

  • Example: \{1,2,1,3\} is a bag and not a set.

  • Select, project, and join work for bags as well as sets.

    – Just work on a tuple-by-tuple basis, and don’t eliminate duplicates.*

**Bag Operations**

*• Union: sum the times an element appears in the two bags.

  • Example: \{1,2,1\} \( \cup \) \{1,2,3,3\} = \{1,1,1,2,2,3,3\}.

  • Intersection: take the minimum of the number of occurrences in each bag.

  • Example: \{1,2,1\} \( \cap \) \{1,2,3,3\} = \{1,2\}.

  • Difference: subtract the number of occurrences in the two bags.

  • Example: \{1,2,1\} – \{1,2,3,3\} = \{1\}.*

**Different Laws for Bags**

*• Some familiar laws continue to hold for bags.

  – Examples: union and intersection are still commutative and associative.

  • But other laws that hold for sets do **not** hold for bags!*


Example

• $R \cap (S \cup T) = (R \cap S) \cup (R \cap T)$ holds for sets but not for bags!
• Let $R$, $S$, and $T$ each be the bag $\{1\}$.
• Left side: $S \cup T = \{1,1\}$; $R \cap (S \cup T) = \{1\}$.
• Right side: $R \cap S = R \cap T = \{1\}$; $(R \cap S) \cup (R \cap T) = \{1\} \cup \{1\} = \{1,1\} \neq \{1\}$.

Extended Relational Algebra

• Adds features needed for SQL, bags.
• Duplicate-elimination operator $\delta$.
• Extended projection.
• Sorting operator $\tau$.

Duplicate Elimination

• $\delta(R)$ = relation with one copy of each tuple that appears one or more times in $R$.

<table>
<thead>
<tr>
<th>Beers</th>
<th>$\delta$(Beers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>beer</td>
<td>price</td>
</tr>
<tr>
<td>Guinness</td>
<td>6</td>
</tr>
<tr>
<td>Guinness</td>
<td>7</td>
</tr>
<tr>
<td>Guinness</td>
<td>7</td>
</tr>
<tr>
<td>Bud</td>
<td>5</td>
</tr>
</tbody>
</table>

Sorting

• $\tau_l(R)$ = list of tuples of $R$, ordered according to attributes on list $L$.
• Note that result type is outside the normal types (set or bag) for relational algebra.
  – Consequence: $\tau$ cannot be followed by other relational operators.

Extended Projection

• Allow the columns in the projection to be functions of one or more columns in the argument relation.

<table>
<thead>
<tr>
<th>Beers</th>
<th>$\pi_{\text{price, price, price-cost}}$(Beers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>beer</td>
<td>price</td>
</tr>
<tr>
<td>Guinness</td>
<td>4</td>
</tr>
<tr>
<td>Guinness</td>
<td>4</td>
</tr>
<tr>
<td>Guinness</td>
<td>4</td>
</tr>
<tr>
<td>Guinness</td>
<td>4</td>
</tr>
<tr>
<td>Bud</td>
<td>1</td>
</tr>
</tbody>
</table>

Sad Drinkers Example

• Find all drinkers that only frequent bars that do not sell their favorite beer.

$Sells(\text{bar}, \text{beer}, \text{price})$
$Bars(\text{name}, \text{addr})$
$Frequents(\text{drinker}, \text{bar})$
$Drinker(\text{name}, \text{addr}, \text{favBeer})$