Nested Schemes

- Flat schemes often have replicated data values.
- Nested schemes allow us to collapse some of these replicated data values.

<table>
<thead>
<tr>
<th>NrBeds</th>
<th>RoomNr</th>
<th>NrBeds</th>
<th>RoomNr</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

Redundancy in Nested Schemes

- The redundancy definition is the same as for flat relations.
- If a value change causes a constraint violation, the value is redundant.

- Table for redundancy in nested schemes.
Algorithm 10.3

Input: a canonical, acyclic, binary ORM hypergraph.

Output: a set of nested schemes with no potential redundancy.

Repeat
  Mark an unmarked node in as the first attribute in a new nested scheme.
  While an unmarked edge is incident on a marked node A:
    Mark the edge.
    If A \rightarrow B: Add B with A; Mark B.
    If A \rightarrow B: Add B with A; Mark B if all B’s incident edges are marked.
    If A \rightarrow B: Nest B under A; Mark B.
    Else (A \rightarrow B): Nest B under A; Mark B if all B’s incident edges are marked.
  Until all nodes have been marked

Nested Scheme Generation Example

1. NrBeds (RoomNr, RoomName, Cost (View)* (GuestNr, GuestName)* )*
2. RoomNr, RoomName, Cost, NrBeds (View)* (GuestNr, GuestName)*
3. GuestNr GuestName RoomNr
   RoomNr, RoomName, Cost, NrBeds (View)*
Redundancy Prevention

![Diagram](image)

This replication ... ... causes this redundancy.

Generalization of Algorithm 10.3 for N-ary Relationship Sets

- “Composite nodes” can be treated as a node (in Algorithm 10.3).
  - B C (A)* (D)*
  - D (B C)*; A B C
- NNF (see Exercise 10.35), basically:
  - Schemes should be constructed along hypergraph paths.
  - Schemes should not violate the natural 1-many hierarchical structure.
Guidelines for Selecting Nested Schemes

- Select “important nodes” as the initial nodes for nested-scheme generation – e.g., Scheme 3 or 2 in earlier Bed-&-Breakfast example.
- Maximize the size of schemes.  
  - Select nodes included in the largest number of FD closures (i.e., when Algorithm 10.3 requires a new node to be arbitrarily selected, compute the set of unmarked nodes in the FD closure of every unmarked node and choose a node included in at least as many sets as any other node) – e.g., Scheme 1 in earlier example.
  - When possible, adjust these generated maximal schemes by placing the most important node first – e.g., Scheme 2 in earlier example.

Cost Analysis for Nested Schemes

- Nested schemes impose variable-length records.
- Recall variable-length record implementation strategies:
  - Reserve enough space for maximum.
  - Chain each nested record.
  - Reserve space for the expected number and chain the rest.
- Insertion, deletion, modification, retrieval tradeoffs.