Normalization

The Achilles Heel of Data Modeling

Gordon C. Everest
Professor Emeritus, Carlson School of Management
University of Minnesota
Community Faculty, MetroState
Forming a Relational Data Structure

Mapping from ER Diagram - some rules:

• Define a TABLE or “Relation” for each Entity type

• SINGLE-VALUED ITEMS (“flat” tables) => 1NF
  – If multivalued or nested repeating group of items, put into a separate table
  – must resolve all M:N relationships into two 1:M's by introducing an association/intersection table

• IDENTIFIER for every table (entity “integrity”)

• Add FOREIGN IDENTIFIERS ("Foreign Keys") to represent all relationships (+ "Referential Integrity")

• NORMALIZE to second, third, ... normal forms
  – done by the data modeler - important for good design
  – but not enforced by RDBMS... WHY?
Codd on Normalization

• 1969 IBM Research Report (unpublished, limited distribution)
  – “the relational view of data permits development of a universal retrieval sub-language based on the second-order predicate calculus.”
  – a relation defined on domains having relations as elements i.e. ‘nested relations’

• 1970/6, CACM, “Relational Model of Data …”
  – a relation should be defined only on domains whose elements are atomic (nondecomposable) values. Codd called this a ‘normalized’ relation. His motive:
    - “can be represented in storage by a two-dimensional column homogeneous array… devoid of embedded pointers, and avoiding dependence on hashing schemes, indexes, and [stored] ordering.”
    - first order predicate calculus suffices

• 1970/10, 1971 introduced “Further Normalization”
  – based on the concept of functional dependence which is fundamental to database design.
Functional Dependency in Relationships

Basis for Database **Normalization**. A $\leftarrow f(X)$ or $X \rightarrow A$

A is dependent on $X$, and the Relationship is exclusive on $A$, multiple on $X$.

**Clustered** into a Record/table for entity of $X$:

| $X$ | $A$ | ... |

There can only be one $A$ for each $X$.
There can be multiple $X$s for a given $A$.
There can be different $A$s for the $X$s.
Start with ENTITIES, their IDENTIFIERS (unique keys) and their ATTRIBUTE FIELDS (facts about each entity). i.e., start with data items clustered into records/tables.

PROBLEM: we may do it wrong; cluster too much; some items in the wrong place, which can lead to redundancy & update anomalies.

Any Flat File is a Relation, but… not all Relations are “well-formed.”

• NORMALIZATION is the test
  – a set of rules to perform internal validation of a data model

• Record DECOMPOSITION is the remedy.
  – Removing attributes from the entity record, and placing them in a different, often a new entity record

(1) First Normal Form: no multivalued items or rgroups of items.
(2) Second Normal Form: no partial dependencies.
(3) Third Normal Form: no transitive dependencies.

“Every non-key data item must be single-valued, and dependent upon the key, the whole key, and nothing but the key… so help me Codd.”
Anomalies

Resulting from (clues to) poor database design:

- DEPTNAME and BOSSNAME stored redundantly
  - if EMPLOYEE moves to another DEPT#, DEPTNAME and BOSSNAME would also change, needing update.
  - If a DEPTNAME (or BOSSNAME) for a DEPT changes, must update all occurrences, else inconsistency.
- To delete a DEPT you must also delete all its EMPLOYEES (unless null foreign keys allowed!)
- If you delete the last EMPLOYEE in a DEPT, you also delete that DEPT (unless null keys allowed!...multiple?)
- No place to insert a DEPT# and its DEPTNAME, if there are no EMPLOYEES there.
STARTING WITH A SET OF DATA ITEMS:

- Employee Name
- Employee ID
- Department
- Dept Address
- Item#
- Item Description
- Item Price
- Warehouse ID
- Warehouse Address
- Item Location in each Warehouse
- Quantity on Hand in each Warehouse
CLUSTER DATA ITEMS INTO ENTITIES (to become TABLES):

**Employee ID**
- Employee Name
- Department
- Dept Address

**Item#**
- Item Description
- Item Price
- Warehouse ID
- Warehouse Address
- Item Location in each Warehouse
- Quantity on Hand in each Warehouse

Can you find any problems with this, violations of the normal forms? 

HINT: Identify all the functional dependencies, what item(s) determine each data item?
Database Normalization - 1NF

1. PULL OUT MULTI-VALUED ITEMS or REPEATING GROUPS:
   into separate “entities”, copy in the “parent” entity identifier, which *may* become part of the identifier of the new subentity.

<table>
<thead>
<tr>
<th>Item#</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>WarehouseID</td>
<td>Address</td>
<td>ItemLocation</td>
</tr>
<tr>
<td>WarehouseID</td>
<td>Address</td>
<td>ItemLocation</td>
</tr>
<tr>
<td>WarehouseID</td>
<td>Address</td>
<td>ItemLocation</td>
</tr>
<tr>
<td>WarehouseID</td>
<td>Address</td>
<td>ItemLocation</td>
</tr>
</tbody>
</table>

**From:**

**To:**

<table>
<thead>
<tr>
<th>Item#</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item#</td>
<td>Warehouse ID</td>
<td>Address</td>
</tr>
</tbody>
</table>

*NOTE: Item# propagates down and becomes part of the identifier. Why?*
First Normal Form (1NF)

• is enforced by construction in a Relational DBMS
  – cannot define a multi-valued item or a repeating group of items

• If you do have a repeating item/group in your conceptual/logical data model, you must take it out (decompose the record) and migrate 'down' the key of the 'parent'.
  – the parent key becomes a foreign key
  – and *may* become part of the primary key

• a clue to a repeating item is the use of a plural attribute name
  e.g., Employee -- Skills, Recording -- Artists, Course -- Instructors, Student -- addresses
Normalize this structure:

<table>
<thead>
<tr>
<th>Order#</th>
<th>Date</th>
<th>CustomerID</th>
<th>CustName</th>
<th>...</th>
<th>Part#</th>
<th>Price</th>
<th>Quantity</th>
<th>Total$</th>
</tr>
</thead>
</table>

Decomposing for 1NF
2. **PULL OUT FACTS ABOUT A PORTION OF THE KEY**

*(partial dependency):*

Find data items which are facts about (determined by) only a portion of a composite key. Split out into separate entities with the portion of the composite key as its identifier.

**From:**

<table>
<thead>
<tr>
<th>Item#</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
</table>

| Item# | Warehouse ID | Address | ItemLocation | Quantity onHand |

**To:**

<table>
<thead>
<tr>
<th>Item#</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
</table>

| Item# | Warehouse ID | ItemLocation | Quantity onHand |

| Warehouse ID | Address |
### Database Normalization - 3NF

**3. Pull out facts about a non-key data item (transitive dependency):**

Find data items which are facts about (determined by) some other non-key items, not the whole identifier (key). Split out into separate entities with the non-key field as the identifier.

**From:**

<table>
<thead>
<tr>
<th>Employee ID</th>
<th>Employee Name</th>
<th>Department</th>
<th>Dept Address</th>
</tr>
</thead>
</table>

**To:**

<table>
<thead>
<tr>
<th>Employee ID</th>
<th>Employee Name</th>
<th>Department</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Department ID</th>
<th>Dept Address</th>
</tr>
</thead>
</table>

*What is the “Department” field called in the Employee record?*

*Why does it remain in the Employee record?*
Transitive Dependency: “Who’s the Boss”

What is wrong here?
How would you fix it?
Fourth Normal Form

• Separate multiple, independent multi-valued facts in the same table.
  – NOTE: each multi-valued fact (M:N relationship) requires a composite key.
  – Storing them together creates a spurious relationship.

• A genuine ternary relationship will be in 4NF, otherwise, break it up into multiple binary relationships

Examples:

PERSON LANGUAGE SPOKEN CUISINE LIKED

EMPLOYEE SKILL LANGUAGE

DOG TRICK COSTUME
• **EXAMPLE:**

![Diagram showing relationships between AGENT, COMPANY, and PRODUCT]

Could store as a ternary relationship, but

IF you have a rule that says:
   "if an Agent sells trucks and represents General Motors,
    then that Agent must sell GM Trucks”
   i.e. all possible combinations are valid

THEN, break it up into three binary tables.
   Can reconstruct the combined table with no loss of information.

If no such symmetric rule or constraint,
   4NF will be in 5NF and you must have one
ternary table to represent only the valid combinations.
Summary of all Normal Forms

GIVEN:
- a set of attributes, clustered into tables/records with identifiers
- all functional dependencies on the attributes

- No multi-valued, non-key attributes (1NF)
- No partial dependencies on non-key attributes (2NF)
- No transitive dependencies in non-key attributes (3NF)
  - No partial or transitive dependencies within any key (BCNF), i.e., consider all candidate keys.
  - No multiple, independent multi-valued attributes in the same table (4NF)
  - No join dependencies, i.e., a relation can be reconstructed without loss of information by joining some of its projections (5NF).
- No more than one table with the same key (“minimal”).
- No transitive dependencies across tables (“optimal”).

NOTE: number order is arbitrary, i.e., there is no necessary sequence to the normal forms.
Others on "Normalization"

Not everyone means the same thing!

• Codd's original 1970 paper spoke of normalized structure
• Finkelstein's "Business Normal Forms" [1989, A-W]
• System Architect from Popkin had a test for normalization
• One popular DBMS vendor offered an add-on software module called "The Normalizer" for $50,000

Ask the vendor of your DBMS or Data Modeling tool if/how their system helps produce a normalized structure.

If they say 'yes',
ask how you define the functional dependencies.

If they say 'you can't' or 'don't know',
then their answer to the first question was wrong!
Sample Data Model in System Architect from Popkin

organization
  -Key Data------------
  unitno @1
  -Non-Key Data-----
  orgname
  budget
  parentunit
  -Normalize--------
  T

employee
  -Key Data------------
  empno @1
  -Non-Key Data-----
  empname
  unit
  jobcode
  title
  bdate
  pskill
  sskill 1\4
  salary
  -Normalize--------
  T

position
  -Key Data------------
  unitno @1
  jobcode @2
  -Non-Key Data-----
  authqty
  authsalary
  -Normalize--------
  T

skill
  -Key Data------------
  skillcode @1
  -Non-Key Data-----
  skilldesc
  -Normalize--------
  T

emporg
  System Architect
  Fri Jul 19, 199X  00:54
  ---------Comment---------
20. Assuming that A is single valued with respect to X (i.e. 1NF).

**GIVEN:**

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
<th>2NF?</th>
<th>3NF?</th>
<th>4NF?</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Could you have a violation of: (if not, why not?)

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
<th>2NF?</th>
<th>3NF?</th>
<th>4NF?</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
<th>2NF?</th>
<th>3NF?</th>
<th>4NF?</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>A</th>
<th>2NF?</th>
<th>3NF?</th>
<th>4NF?</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Normalization Exercise

To find and remedy the violations of Normal form:
1. Show all the Identifiers
2. Show all the Functional Dependencies
3. Remove all the offending non-key attributes
4. Create additional tables to contain those attributes

How many tables do you get?

| EmpID | ProjID | EmpName | EmpDept | DeptBoss | EmpSkills | ProjTitle | ProjBoss | HoursWorked |
Same recording can appear on different CDs and it could be a different track# on each.
Normalization Steps and Rules

23

NEEDED: a method to produce a normalized structure
MUST KNOW: all Functional Dependencies
MUST UNDERSTAND: the notion of Determinant
GIVEN: a relational table (flat file)

• Designate the **Identifier** (Primary Key)

• For each non-key **Data Item** (field):
  • Find its **Determinant**
  • Store it in a record with its Determinant, where its Determinant is the Primary Key
  • Store it only once
    (key fields can be stored more than once, either because they serve as a foreign key, or are in more than one Determinant)
  • If it is plural (multi-valued), store with its 'Determinant' as a composite Primary Key

• Should not have multiple tables with the same Determinant (Primary Key); if so, combine
Normalization: Branch Banking

• Given the following data structure (of two tables):
  Customers have accounts at bank branches. A personal banker may
  be assigned to a customer at each branch where they have an account.
  A customer may have multiple accounts. Each different account type
  has its own rules and interest rate.

<table>
<thead>
<tr>
<th>CustomerID (KEY)</th>
<th>Account# (KEY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Name</td>
<td>CustomerID (FKEY)</td>
</tr>
<tr>
<td>Address</td>
<td>Customer Name</td>
</tr>
<tr>
<td>Phone</td>
<td>Account Type</td>
</tr>
<tr>
<td>Branch Name</td>
<td>Interest Rate</td>
</tr>
<tr>
<td>Branch Address</td>
<td>Minimum Balance</td>
</tr>
<tr>
<td>Pers.Banker ID</td>
<td>Account Balance</td>
</tr>
<tr>
<td>Pers.Banker Name</td>
<td></td>
</tr>
<tr>
<td>Pers.Banker Phone</td>
<td></td>
</tr>
</tbody>
</table>

• Produce a normalized data model.
  • How many tables?
  • any missing relationships?
Normalization - the Bottom Line

• In a record-based (ER) data model, there are both inter-record (explicit) and intra-record (implicit) relationships.
• So, start with individual domains for all entities and attributes, and explicitly represent all relationships.
• Build up your model from elementary facts
• Thus, defining all functional dependencies
• Enabling the system to produce a normalized data structure

==> If you don't a priori cluster attributes into records, you won't ever need to normalize!
Why does anyone need to do this?

• Results in good database design with attributes in the right place.
• Avoids inconsistency due to redundancy.
• Avoids update processing inefficiency, complexity, anomalies.
• Avoids wasted space due to redundancy.

Why should you know how to do this?

• If you develop your own personal or departmental database.
• If you work with central IS to develop a corporate database.
• If you are auditing/evaluating the goodness of a database design.
Implications for Non-IS People

• You give the systems analysts a few tables of data
• Systems analysts develop an ERD and Normalize the database (“flat” Relational records)
• They return their results to you in the form of an ERD of the (hopefully) Normalized database
• You need to be able to read and understand ERDs to review their work:
  – Does it contain all the information you need?
  – Are the Entities and Relationships right?
  – Are all the Attributes (data fields) included, in the right place?
  – Can you do the queries and get the results you want?
• You need to feel confident to ask the systems analysts to show you the ERD in the form of tables with links, and to allow you to do queries (preferably via a database, but at least via query statements) to ensure the Normalization works for you
• Will needed Referential Integrity be enforced?
Denormalization

- Normalization results in Record Decomposition, which impacts performance
  - Retrieval (-); Update (+) once, maintain consistency

- Denormalization means *recombining* “attributes” to form fewer, larger records.

- The sole objective is performance:
  - handling larger chunks on disk I/O
  - effectively Prejoining files results in fewer joins

- Denormalization is done at implementation time, NOT at conceptual / logical database design time.

- Denormalization should be a conscious decision (to violate the rules of normalization), NOT the result of unnormalized database designs (because the designer did not recognize violations of the normal forms)
Normalization: References

29. KENT, William, “A Simple Guide to Five Normal Forms in Relational Database Theory,” *Communications of the ACM* (26:2), 1983 Feb., p.120.


29. FINKE STEIN, Clive, “Business Normalization,” ch. 4 in *Information Engineering: Strategic Systems Development*, Addison-Wesley, 1992. (first 3 are the same, 4BNF and 5BNF are different)
Annual product sales by region ($,000)

<table>
<thead>
<tr>
<th>PRODUCT:</th>
<th>SOUTHERN</th>
<th>WESTERN</th>
<th>NORTHERN</th>
<th>EASTERN</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stibes</td>
<td>$7,140</td>
<td>$14,790</td>
<td>$13,260</td>
<td>$15,810</td>
<td>$51,000</td>
</tr>
<tr>
<td>Farkles</td>
<td>5,460</td>
<td>11,310</td>
<td>10,140</td>
<td>12,090</td>
<td>39,000</td>
</tr>
<tr>
<td>Teglers</td>
<td>3,150</td>
<td>6,525</td>
<td>5,850</td>
<td>6,975</td>
<td>22,500</td>
</tr>
<tr>
<td>Qwerts</td>
<td>5,250</td>
<td>11,875</td>
<td>10,750</td>
<td>12,625</td>
<td>40,500</td>
</tr>
<tr>
<td><strong>TOTALS</strong>:</td>
<td><strong>$21,000</strong></td>
<td><strong>$44,500</strong></td>
<td><strong>$40,000</strong></td>
<td><strong>$47,500</strong></td>
<td><strong>$153,000</strong></td>
</tr>
</tbody>
</table>

*Is this a Relational Table?*  
*What is the Entity?*  
*What is the Identifier?*  
*What are the Attributes?*  
*How many Fact types?*  
*How many Dimensions?*  
*How to make it a Relational Table?*
Sales Data in a Relational Table:

**How many Facts?**

**What is the Identifier?**

**How many Dimensions?**

**Where are the Dimension Tables?**

**How many rollup levels?**

**What is the business process?**

**What is the Grain?**

**How far can you Drill Down?**

<table>
<thead>
<tr>
<th>REGION:</th>
<th>PRODUCT:</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>LEVEL</td>
<td></td>
</tr>
<tr>
<td>Southern</td>
<td>Stibes</td>
<td>$7,140</td>
</tr>
<tr>
<td>Southern</td>
<td>Farkles</td>
<td>5,460</td>
</tr>
<tr>
<td>Southern</td>
<td>Teglers</td>
<td>3,150</td>
</tr>
<tr>
<td>Southern</td>
<td>Qwerts</td>
<td>5,250</td>
</tr>
<tr>
<td>Western</td>
<td>Stibes</td>
<td>14,790</td>
</tr>
<tr>
<td>Western</td>
<td>Farkles</td>
<td>11,310</td>
</tr>
<tr>
<td>Western</td>
<td>Teglers</td>
<td>6,525</td>
</tr>
<tr>
<td>Western</td>
<td>Qwerts</td>
<td>11,875</td>
</tr>
<tr>
<td>Northern</td>
<td>Stibes</td>
<td>13,260</td>
</tr>
<tr>
<td>Northern</td>
<td>Farkles</td>
<td>10,140</td>
</tr>
<tr>
<td>Northern</td>
<td>Teglers</td>
<td>5,850</td>
</tr>
<tr>
<td>Northern</td>
<td>Qwerts</td>
<td>10,750</td>
</tr>
<tr>
<td>Eastern</td>
<td>Stibes</td>
<td>15,810</td>
</tr>
<tr>
<td>Eastern</td>
<td>Farkles</td>
<td>12,090</td>
</tr>
<tr>
<td>Eastern</td>
<td>Teglers</td>
<td>6,975</td>
</tr>
<tr>
<td>Eastern</td>
<td>Qwerts</td>
<td>12,625</td>
</tr>
<tr>
<td>(all)</td>
<td>Stibes</td>
<td>51,000</td>
</tr>
<tr>
<td>(all)</td>
<td>Farkles</td>
<td>39,000</td>
</tr>
<tr>
<td>(all)</td>
<td>Teglers</td>
<td>22,500</td>
</tr>
<tr>
<td>(all)</td>
<td>Qwerts</td>
<td>40,500</td>
</tr>
<tr>
<td>Southern</td>
<td>(all)</td>
<td>21,000</td>
</tr>
<tr>
<td>Western</td>
<td>(all)</td>
<td>44,500</td>
</tr>
<tr>
<td>Northern</td>
<td>(all)</td>
<td>40,000</td>
</tr>
<tr>
<td>Eastern</td>
<td>(all)</td>
<td>47,500</td>
</tr>
<tr>
<td>(all)</td>
<td>(all)</td>
<td>153,000</td>
</tr>
</tbody>
</table>

Aggregations (Rollups):

<table>
<thead>
<tr>
<th>REGION:</th>
<th>PRODUCT:</th>
<th>LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>LEVEL</td>
<td></td>
</tr>
<tr>
<td>Southern</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Western</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Northern</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Eastern</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(all)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Stibes</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Farkles</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Teglers</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Qwerts</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(all)</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Star (Join) Schema

FACT TABLE
Date_key
Product_key
Region_key
Sales_dollars

DATE
Date_key
Year

PRODUCT
Product_key
Description
Brand
Category

REGION
Region_key
Region_name
Location
Size
...
A typical customer billing fact table: in which the extended net price can be derived from the other quantities, but nevertheless we want to store it in the table.

**TIME dimension**
- Time_key

**PRODUCT dimension**
- Customer_key
- Product_key
- Promotion_key
- Salesperson_key
- Status_key

**CUSTOMER dimension**

**PROMOTION dimension**

**SALESPERSON dimension**

**STATUS dimension**

\[
\text{THIS} \rightarrow \text{MINUS} \rightarrow \text{MINUS} \rightarrow \text{EQUALS} \rightarrow = \text{REVENUE}
\]

**Quantity Sold**
- Extended List Price
- Total Allowances
- Total Discounts
- Extended Net Price

**SHOULD WE STORE IT? . . . YES!**

*Kimball, DBMS, 1997 January. Figure 3.*

*Is this fact table normalized?*
Example Dimension Table

LOCATION
Location Key
Location Desc.
Region ID
Region Desc.*
Store ID
Store Desc.*
Level

DATE
Date Key
Date Desc.
Date Week ID
Month ID
Year ID
Level

PRODUCT
Product Key
Product Desc.
Department ID
Department Desc.*
Product Group ID
Product Group Desc.*
Product Item ID
Product Item Desc.*
Level

SALES FACTS
Date Key
Location Key
Product Key
Sale Amount
Sale Units

• Descriptions added (*)
• Levels added

<table>
<thead>
<tr>
<th>LOC KEY</th>
<th>LOC DESC</th>
<th>R.ID</th>
<th>R.DESC*</th>
<th>S.ID</th>
<th>S.DESC*</th>
<th>LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Northeast</td>
<td>1</td>
<td></td>
<td>202</td>
<td>Larpenteur</td>
<td>2</td>
</tr>
<tr>
<td>105</td>
<td>Midwest</td>
<td>2</td>
<td></td>
<td>234</td>
<td>Lexington</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>Southeast</td>
<td>3</td>
<td></td>
<td>254</td>
<td>Snelling</td>
<td>1</td>
</tr>
<tr>
<td>115</td>
<td>Boston</td>
<td>1</td>
<td>Northeast</td>
<td>221</td>
<td>Hamline</td>
<td>1</td>
</tr>
<tr>
<td>120</td>
<td>Chicago</td>
<td>2</td>
<td>Midwest</td>
<td>232</td>
<td>Dale</td>
<td>1</td>
</tr>
<tr>
<td>125</td>
<td>New York</td>
<td>1</td>
<td>Northeast</td>
<td>202</td>
<td>Larpenteur</td>
<td>2</td>
</tr>
<tr>
<td>130</td>
<td>Atlanta</td>
<td>3</td>
<td>Southeast</td>
<td>232</td>
<td>Dale</td>
<td>1</td>
</tr>
<tr>
<td>135</td>
<td>Chicago 2</td>
<td>2</td>
<td>Midwest</td>
<td>232</td>
<td>Dale</td>
<td>1</td>
</tr>
<tr>
<td>140</td>
<td>All</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>
Snowflaking a Dimension Table

- Removing low cardinality attributes and putting them into a separate table

NOTE the use of new keys to join the tables

Snowflaking is NOT RECOMMENDED but is useful for understanding.
• A dimension may have one or more hierarchies
• Showing the hierarchy in the structure produces a “snowflaked” Dimension

Can you find another hierarchy in the STORE Dimension Table?

Also commercial customer hierarchies
- KIMBALL, Toolkit, 2e, 6p.161.

Snowflaking is NOT RECOMMENDED for stored dimension tables but is useful for human understanding.
Flattening the Hierarchy in a Dimension

- Flattening all hierarchies in a Dimension produces a single “denormalized” table
- Flattening all hierarchies in all Dimensions produces a “Star” Schema

FACT TABLE
-------------------
Date_key
Product_key
Customer_key
Store_key
-----------
Sales_dollars
Sales_units
Cost_dollars

STORE
-----------------
Store_key
Store_name
Address
City
Community
Metro_Area
Territory
Region
County
Zip_Code
State
Telephone
Floor_plan

FLATTENED DIMENSION TABLE

- These would be the Descriptions/Names
- Only store the IDs if used by the Users