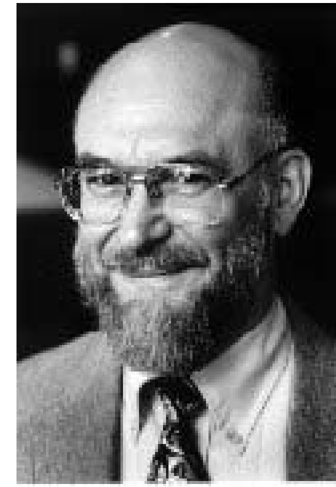
35START



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*:**

Σ

- **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

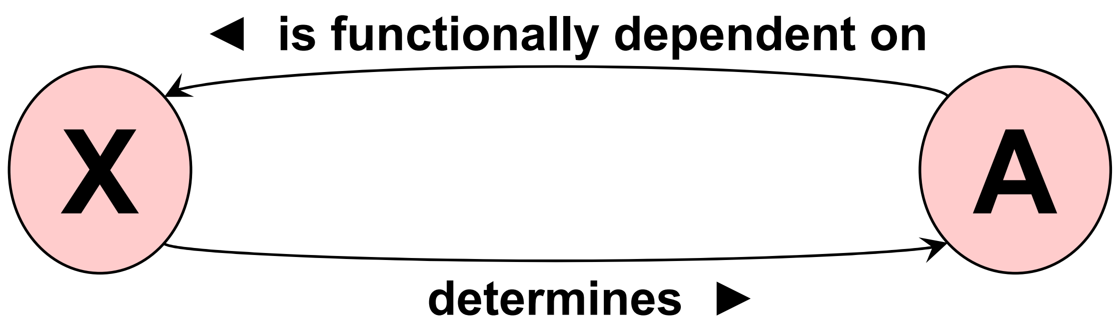
- **1969 IBM Research Report** **(unpublished, limited distribution)**
  - **"the relational view of data permits development of a universal retrieval sub-language based on the second-order predicate calculus."**
  - **a relation defined on domains having relations as elements i.e. 'nested relations'**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

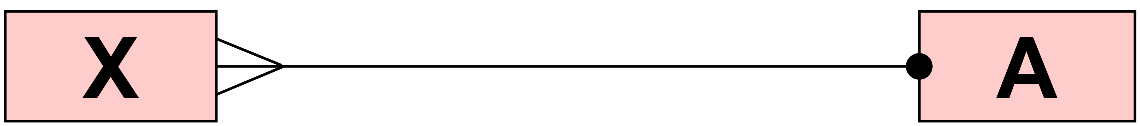
Functional Dependency in Relationships

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

or $X \rightarrow A$

Σ





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 Xs for a given A .

There can be *different* As for the Xs .

Database Normalization

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:

<u>Employee#</u>	EmpName	Skills	Proficiency	...	BossName	DeptID	DeptName

- **DEPTNAME and BOSSNAME stored <u>redundantly</u>**
- **if EMPLOYEE moves to another DEPT#, DEPTNAME and BOSSNAME would also change, needing <u>update</u>.**
- **If a DEPTNAME (or BOSSNAME) for a DEPT changes, must <u>update</u> *all* occurrences, else inconsistency.**
- **To <u>delete</u> a DEPT you must also delete all its EMPLOYEEs (unless null foreign keys allowed*!*)**
- **If you <u>delete</u> the last EMPLOYEE in a DEPT, you also delete that DEPT (unless null keys allowed*!…multiple?*)**
- **No place to <u>insert</u> a DEPT# and its DEPTNAME, if there are no EMPLOYEEs there.**

Database Normalization - Example

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

Database Normalization - Example

CLUSTER DATA ITEMS INTO ENTITIES (to become TABLES):

<u>Employee ID</u>
Employee Name
Department
Dept Address

<u>Item#</u>
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

❶ **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.**

From:

<u>**Item#**</u> Description Price

WarehouseID Address ItemLocation Quantity onHand

WarehouseID Address ItemLocation Quantity onHand

WarehouseID Address ItemLocation Quantity onHand

WarehouseID Address ItemLocation Quantity onHand

:

To:

<u>**Item#**</u> Description Price

<u>**Item# Warehouse ID**</u> Address ItemLocation Quantity onHand

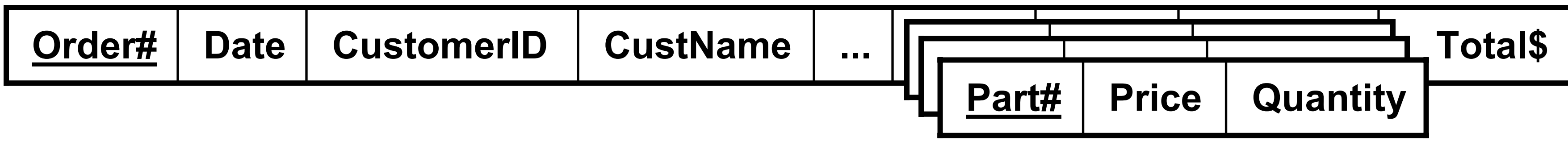
*NOTE: **Item#** propagates down and becomes part of the identifier. **Why?***

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 <u>plural attribute name</u>**
  e.g., Employee -- Skills, Recording -- Artists,
  Course -- Instructors, Student -- addresses

Decomposing for 1NF

Normalize this structure:



Database Normalization - 2NF

❷ **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:

Item#	Description	Price

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

To:

Item#	Description	Price

Item#	**Warehouse ID**	ItemLocation	Quantity onHand

Warehouse ID	Address

Database Normalization - 3NF

❸ **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:

Employee ID	Employee Name	Department	Dept Address

To:

Employee ID	Employee Name	Department

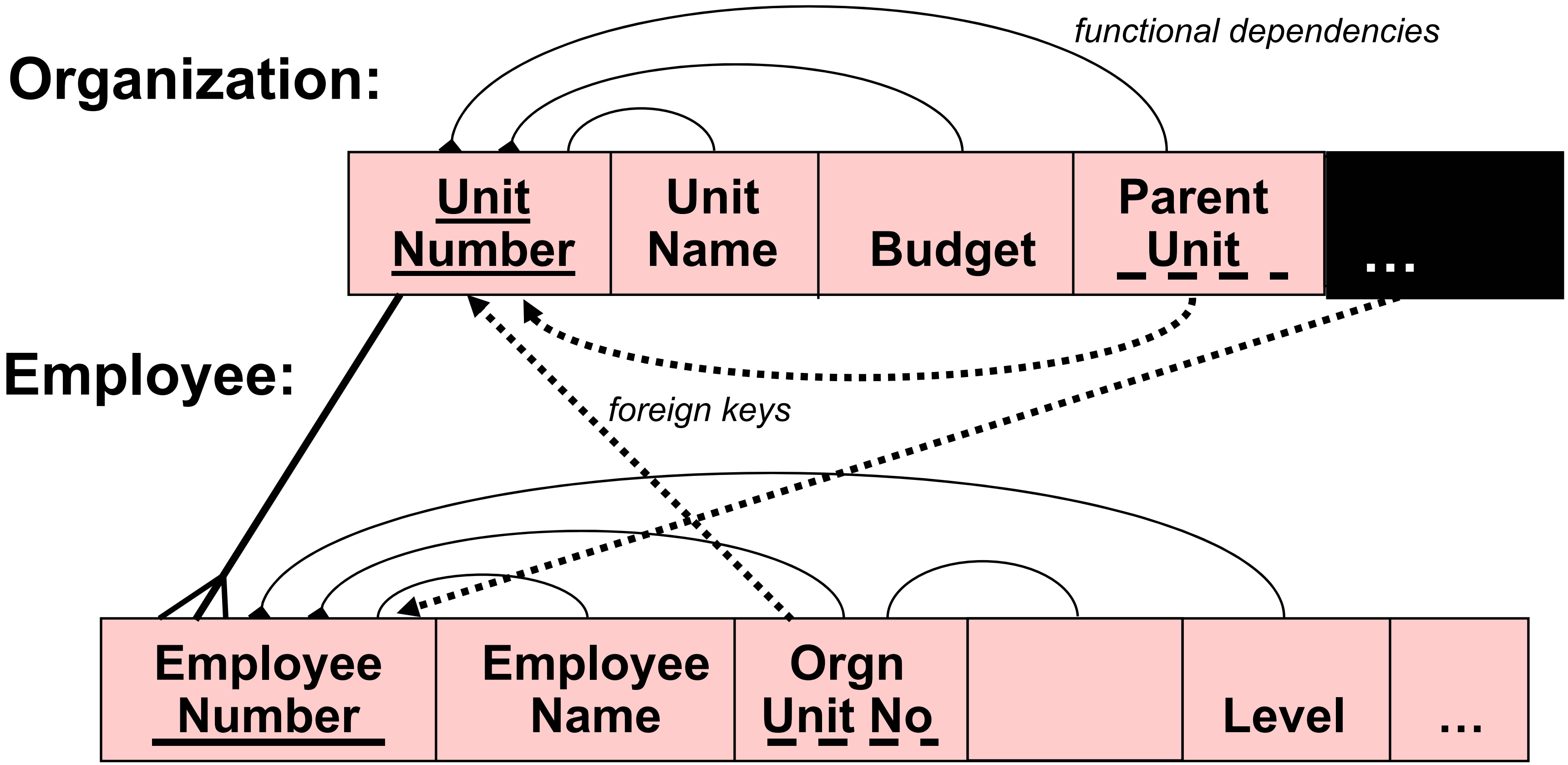
Department ID	Dept Address

What is the "Department" field called in the Employee record?

Why does it remain in the Employee record?

Contrast with WATSON4-6p135-142.

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

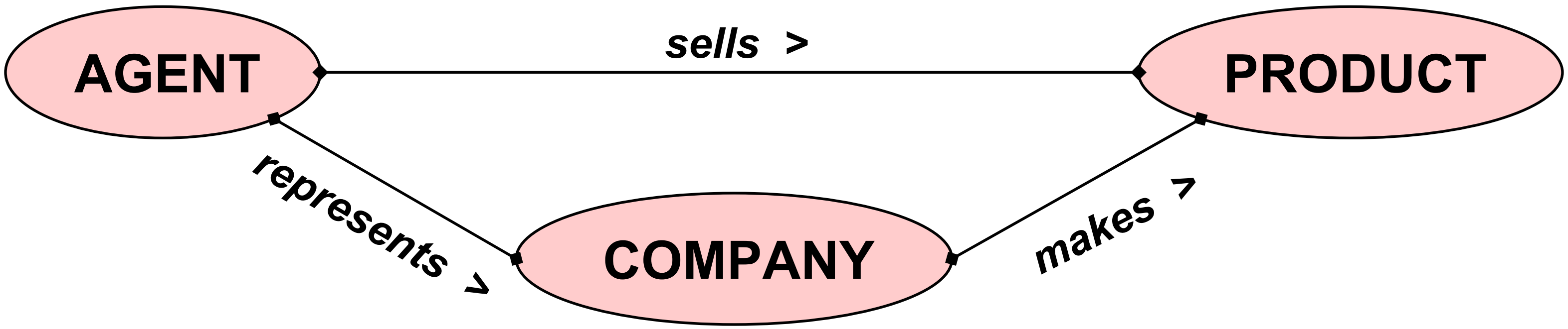
EMPLOYEE SKILL LANGUAGE

DOG TRICK COSTUME

Fifth Normal Form

Wm. Kent, “Simple Guide to Five Normal Forms,” *CACM*, 1983 Feb.

- **EXAMPLE:**



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!

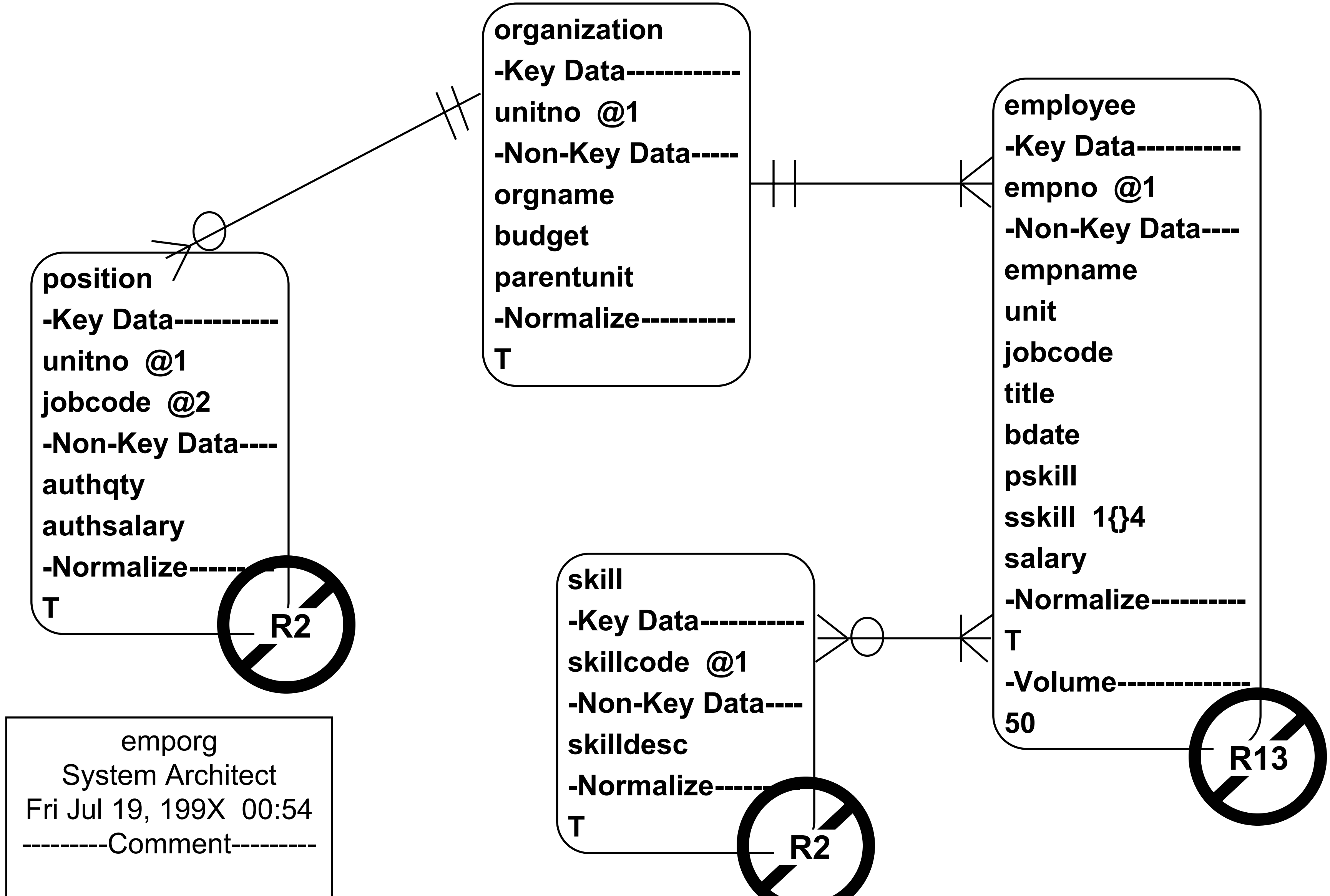
- Codd's original 1970 paper spoke of normalized structure
- Finkelstein's "Business Normal Forms" [1989, A-W]
- *System Architect* from Popkin had a test for normalization
- One popular DBMS vendor offered an add-on software module called "The Normalizer" for $50,000

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



Normalization – Testing your Understanding

Assuming that A is single valued with respect to X (i.e. 1NF).

GIVEN:

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

$\underline{X}$	A

2NF? 3NF? 4NF?

$\underline{X}$	A	B

2NF? 3NF? 4NF?

$\underline{X}$	$\underline{A}$	B

2NF? 3NF? 4NF?

Normalization Exercise

see also 'Skill Builder' WATSON4-8p210.
and answer on the web.

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

Normalization Exercise

<u>CD-ID</u>	<u>RecordingID</u>	CD-title	CD-label	Track#	TrackLength	Artists	LabelAddr	QtyOnHand

Same recording can appear on different CDs and it could be a different track# on each.

Normalization Steps and Rules

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.

CustomerID (KEY)	**Account# (KEY)**
Customer Name	**CustomerID (FKEY)**
Address	**Customer Name**
Phone	**Account Type**
Branch Name	**Interest Rate**
Branch Address	**Minimum Balance**
Pers.Banker ID	**Account Balance**
Pers.Banker Name	
Pers.Banker Phone	

- **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!

Database Normalization - Why?

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

- **KENT, William, "A Simple Guide to Five Normal Forms in Relational Database Theory,"** ***Communications of the ACM*** **(26:2), 1983 Feb., p.120.**
- **DATE, C. J., "Thirty Years of Relational: The First Three Normal Forms," in 2 parts,** ***Intelligent Enterprise*** **(2:5 & 6), 1999 March & April.**
- **HALPIN, Terry,** ***Information Modeling and Relational Databases,*** **Morgan Kaufmann, 2001, §12.6, p.627-642.**
- **SIMSION, Graeme C. and Graham C. WITT,** ***Data Modeling Essentials*****, 3e, Morgan Kaufmann, 2005, Ch. 2 & 13.**
- **BECKER, Scot, "Normalization and ORM," 1998 August, and "Data Schema Normalization," 1999 June,** ***Journal of Conceptual Modeling*****. (www.inconcept.com/JCM)**
- **FINKELSTEIN, Clive, "Business Normalization," ch. 4 in** ***Information Engineering: Strategic Systems Development*****, Addison-Wesley, 1992. (first 3 are the same, 4BNF and 5BNF are different)**

Sales Data in a Spread Sheet (the "Cube")

Annual product sales by region ($,000)

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

Is this a Relational Table?

What is the Entity?

What is the Identifier?

What are the Attributes?

How to make it a Relational Table?

How many Fact types?

How many Dimensions?

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?

REGION:

NAME	LEVEL
Southern	2
Western	2
Northern	2
Eastern	2
(all)	1

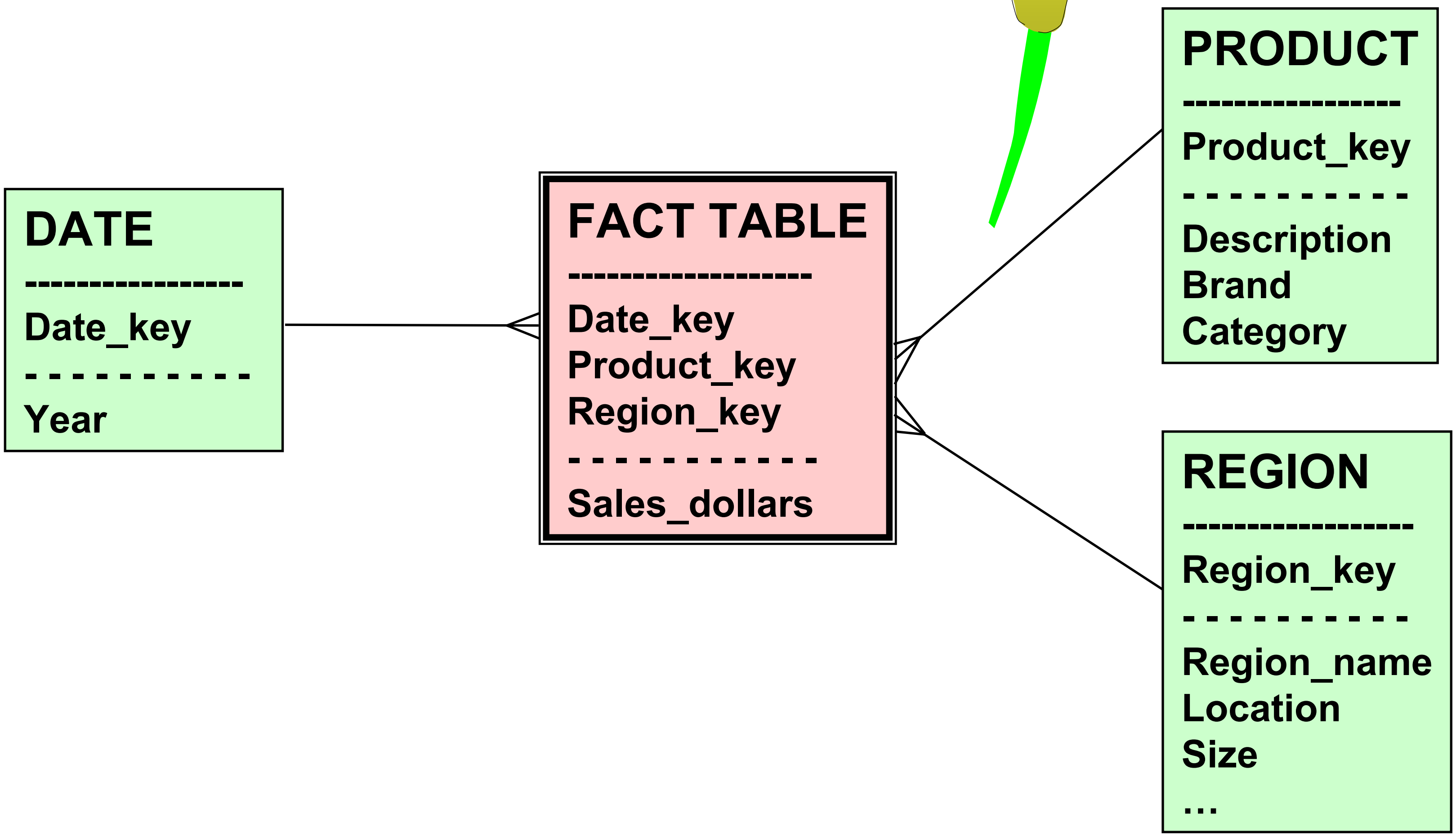
PRODUCT:

NAME	LEVEL
Stibes	2
Farkles	2
Teglers	2
Qwerts	2
(all)	1

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

Aggregations (Rollups)

Star (Join) Schema



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**

Customer_key — • **CUSTOMER dimension**

PRODUCT dimension • — **Product_key**

Promotion_key — • **PROMOTION dimension**

SALESPERSON dimension • — **Salesperson_key**

Status_key — • **STATUS dimension**

Quantity Sold

THIS--------> **Extended List Price**

MINUS--------> **Total Allowances**

MINUS--------> **Total Discounts**

EQUALS--------> **Extended Net Price** = *REVENUE*

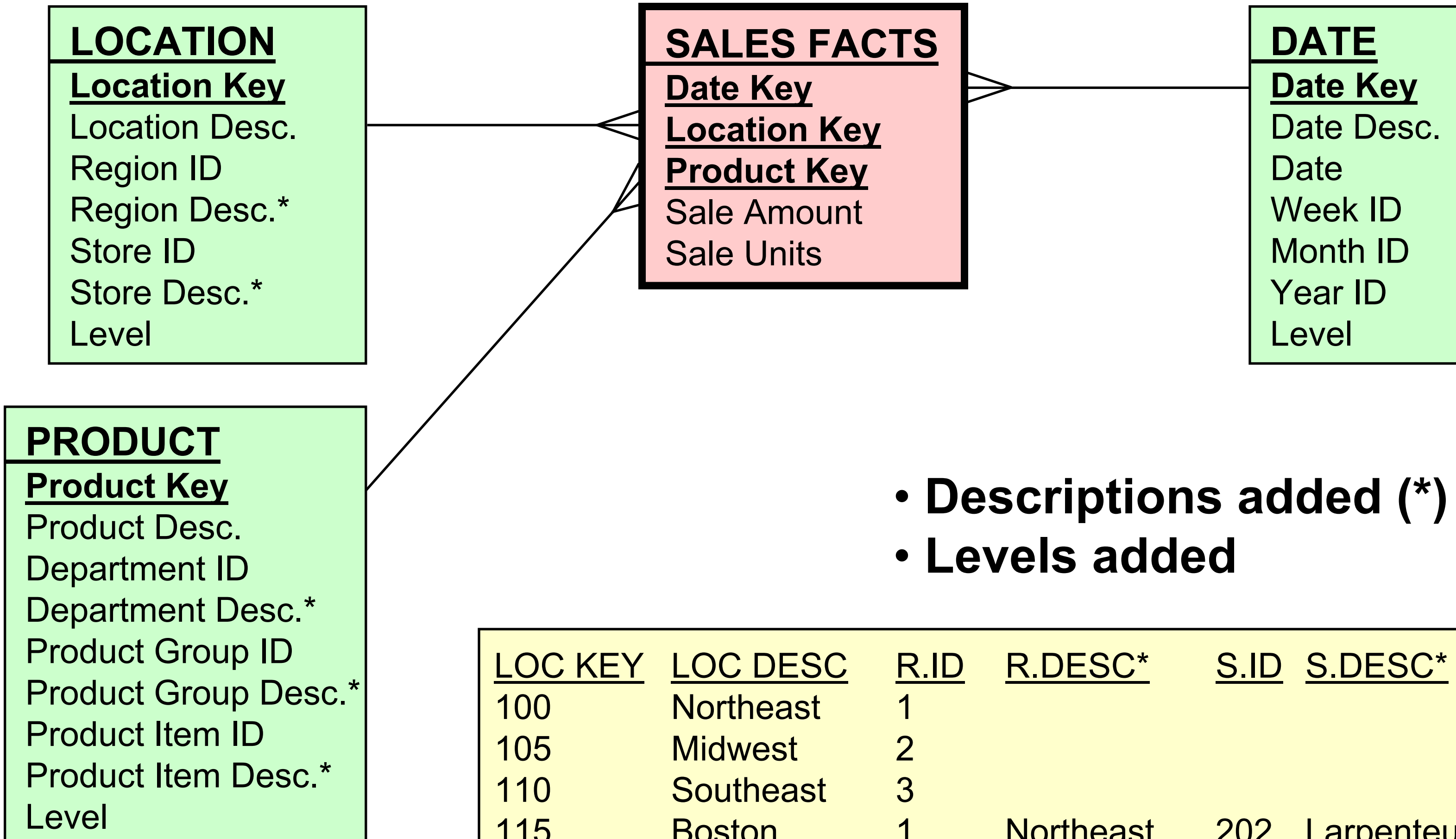
SHOULD WE STORE IT?. . . . **YES**!

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

Is this fact table normalized?

Example Dimension Table

REF: B&A, Ch.14



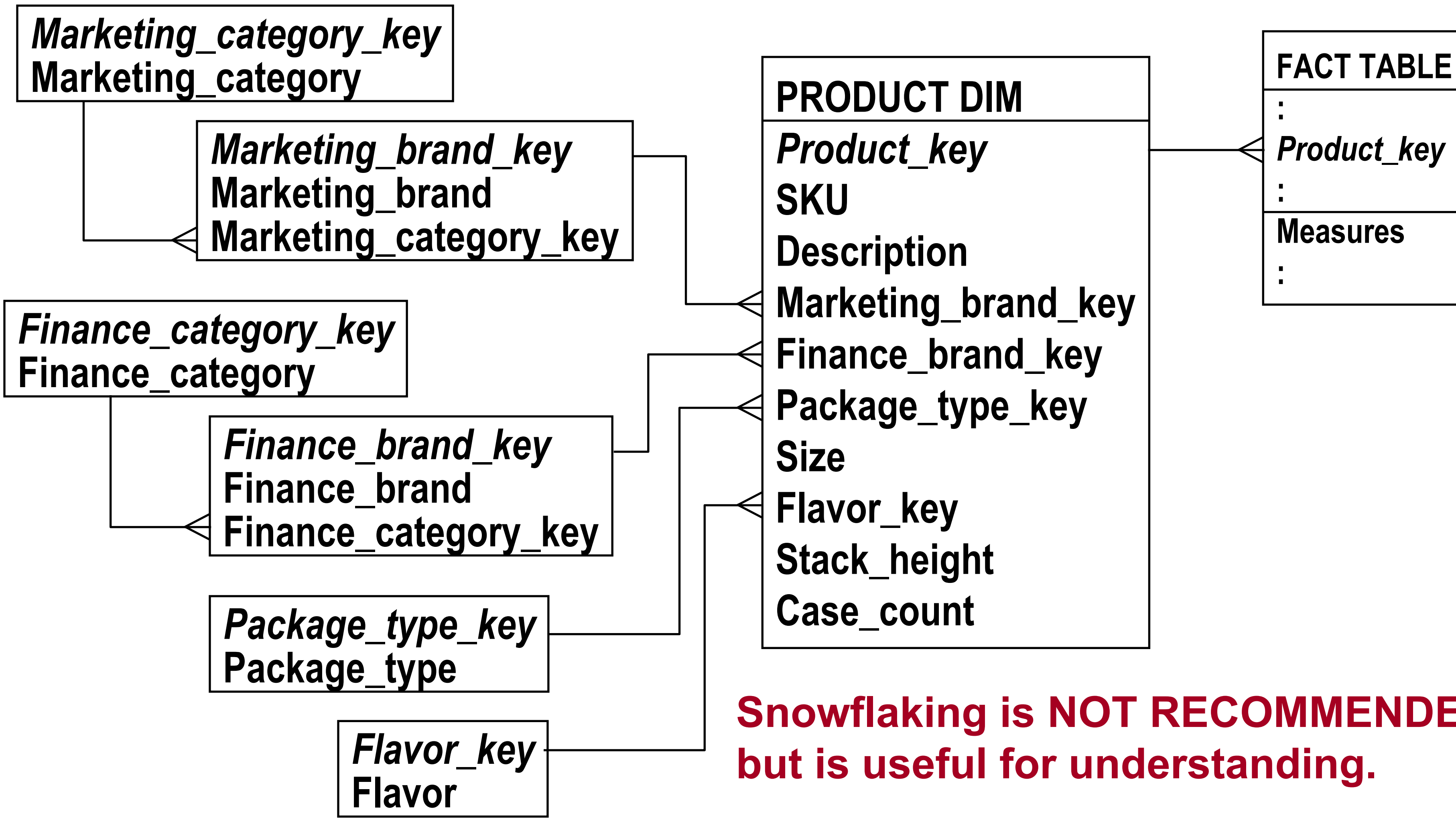
- **Descriptions added (*)**
- **Levels added**

LOC KEY	LOC DESC	R.ID	R.DESC*	S.ID	S.DESC*	LEVEL
100	Northeast	1				2
105	Midwest	2				2
110	Southeast	3				2
115	Boston	1	Northeast	202	Larpenteur	1
120	Chicago	2	Midwest	234	Lexington	1
125	New York	1	Northeast	254	Snelling	1
130	Atlanta	3	Southeast	221	Hamline	1
135	Chicago 2	2	Midwest	232	Dale	1
140	All					3

Snowflaking a Dimension Table

Kimball, *Lifecycle Toolkit*, p. 171.

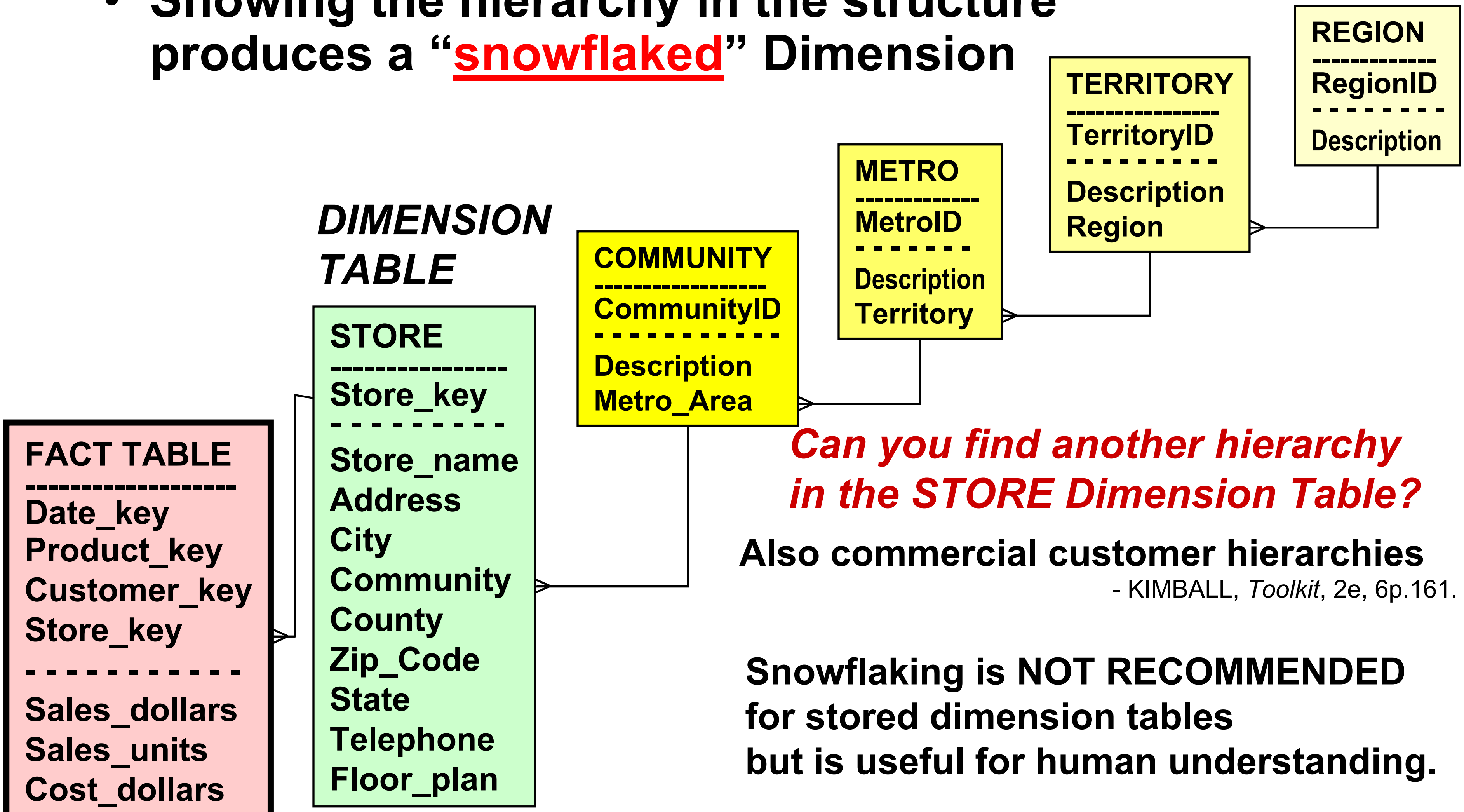
- **Removing low cardinality attributes and putting them into a separate table NOTE the use of new keys to join the tables**



Snowflaking is NOT RECOMMENDED but is useful for understanding.

Hierarchy in a Dimension

- **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**

FLATTENED DIMENSION TABLE

STORE

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

FACT TABLE

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

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