



**MnIPS, 2003 November 18**

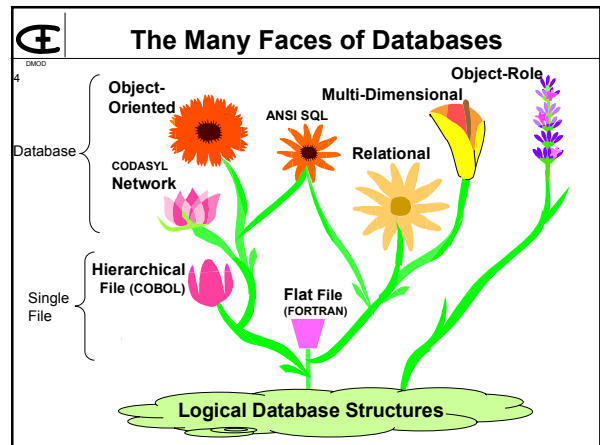
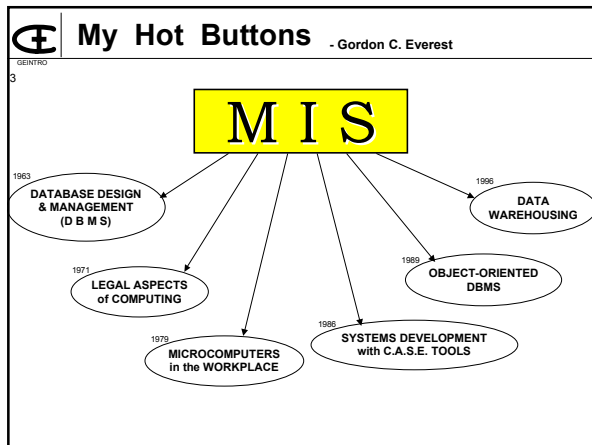
**The Many Faces of Databases**

**Gordon C. Everest**  
Information and Decision Sciences  
Carlson School of Management  
University of Minnesota

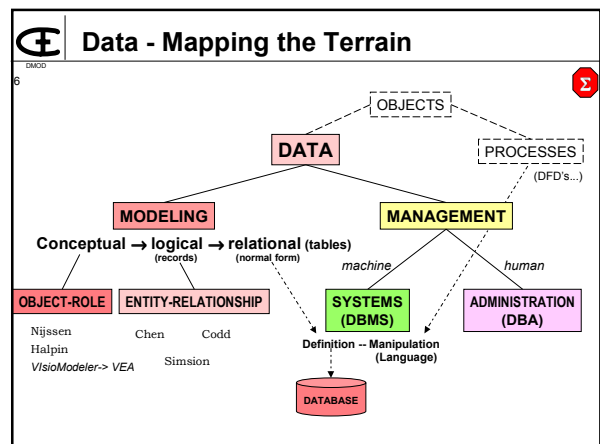
**OUTLINE**

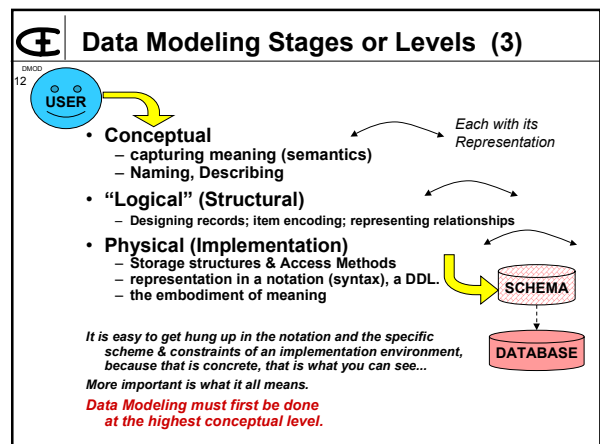
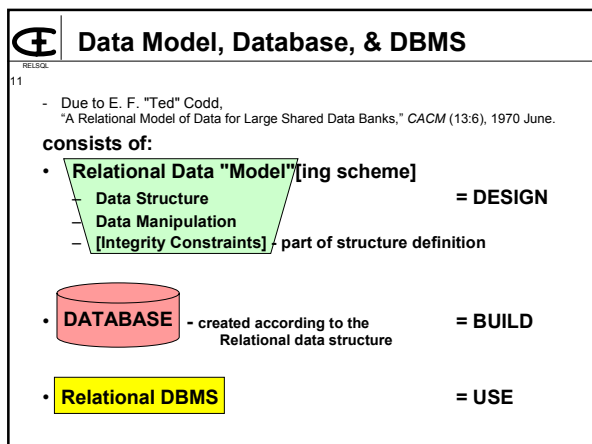
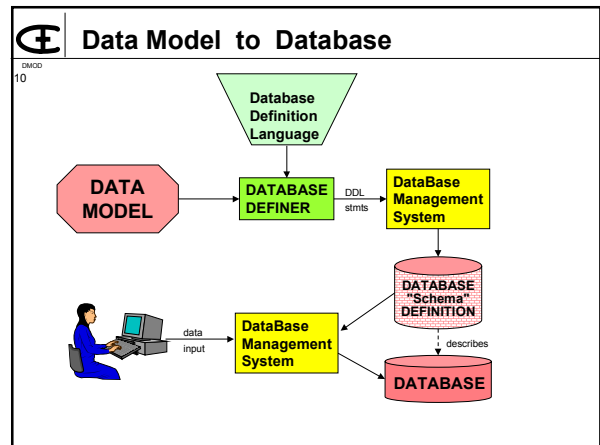
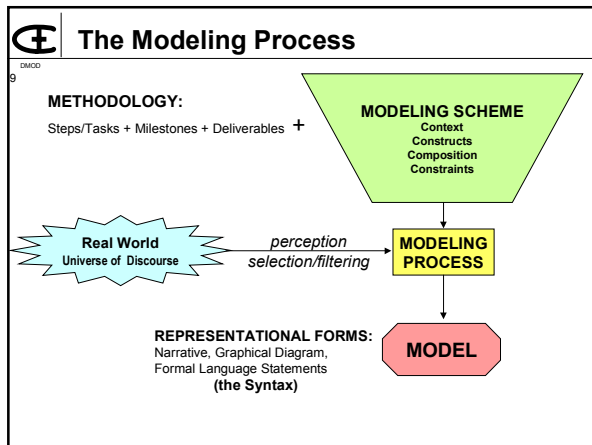
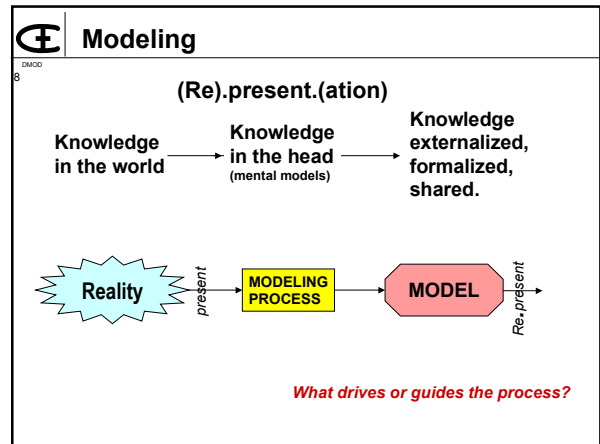
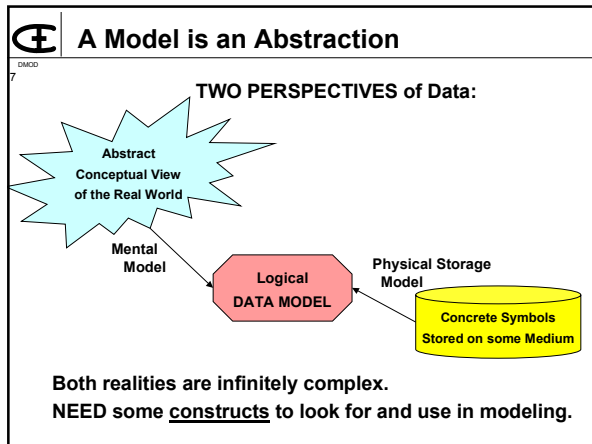
- Models, Databases, and DBMSs
- Logical Data Modeling – main constructs
- Taxonomy of Data Modeling Schemes
  - Hierarchic, Network, & Relational
- ANSI SQL, SQL:1999
  - (Various ER Data modeling schemes)
- Normalization
- Object Databases
- Data Warehousing – multidimensional modeling
  - Cubes, Stars, & Snowflakes
- Object Role Modeling
- Denormalization



**Testing Your Knowledge:**

- Hierarchic / Network / Relational
  - A taxonomy of what?
  - Is it a good taxonomy?
  - What differentiates each?
  - What is left out?
- What is Relational?
  - Referring to what? A data structure? DBMS?...
  - Who originated?
  - SQL? SQL2? SQL3? SQL/99?
- What is a Fourth Generation Language (4GL)?
- What is the dominant data modeling scheme today?
  - What is Wrong with it?
- What is Normalization?
  - Why do we do it?
  - How did we get into trouble in the first place?
  - Can we avoid (the need for) normalization? How?





### ☒ Data Modeling Constructs

DMADO 13

**What to look for:**  
Relative emphasis differentiates Data Modeling approaches  
e.g. ER modeling focuses on Entities and Relationships, de-emphasizing or hiding Attributes.

**ENTITY (OBJECT)** → **RELATIONSHIP** (diamond)  
**ATTRIBUTE (Data Item)** (oval) ← **IDENTIFIER** (yellow box)  
 [ FOREIGN KEY ] (dashed line) ← **RELATIONSHIP**  
 characteristics (dashed line) ← **RELATIONSHIP**  
 characteristics (dashed line) ← **ATTRIBUTE**

### ☒ Entity-Attribute-Value-Relationship View of Data

DMADO 14

**ENTITY:** Any concrete or abstract object or event in the user's world  
**ATTRIBUTE:** A characteristic of interest about an entity  
**VALUE:** A symbol or character string assigned to an entity instance to describe it  
**RELATIONSHIP:** Some connection or association between entities

An organization collects information about **Entities** - any person, object, or event, however concrete or abstract. **Attributes** are the characteristics of interest about entities. **Values** of attributes represent the actual data pertaining to specific entities in the organization or its environment. **Relationships** may exist between entities and are usually represented by additional attributes. For example, recording the number of the organizational unit for which an employee works establishes a relationship between these two entities.

**ORGANIZATIONAL UNIT (Entity):**  
 UNIT NUMBER: 2100  
 NAME: DEVELOPMENT DEPT.  
 BUDGET: \$391,000  
 PARENT UNIT: 2000  
 HEAD: P.J. Carr

**PERSON (Entity):**  
 EMPLOYEE NUMBER: 15324  
 NAME: R.F. CALLAGAN  
 SEX: F  
 BIRTHDATE: 550606  
 PRIMARY SKILL: 5210  
 SECONDARY SKILLS: 5520, 5220

**Relationship:** connects ORGANIZATIONAL UNIT and PERSON.  
 Attributes: LEVEL: EMPL, POSITION TITLE: SECRETARY, JOB CODE: 5210, ORG UNIT: 2100, SALARY: \$31,400

### ☒ Basic Constructs for Database Design

DMADO 15

- ENTITY** - any concrete or abstract object or event about which data is to be stored; in multiple instances.  
- e.g. Employee, Purchase order line item, Customer call, Invoice, Product, ...
- ATTRIBUTE** - information about an entity which is of interest (a characteristic or property).  
- e.g. Name, EmployeeID, Age, Quantity ordered, Problem description, Price, ...
- IDENTIFIER (or "key")** - one (or more) attributes of an entity which uniquely identifies instances of that entity type. (Used to retrieve, update, and sort entity records)  
- e.g. EmployeeID, P.O. Number, Customer No., Item#, ...

**COMPOSITE KEY** - a combination of attributes required to uniquely identify an entity instance  
 (often represents a many-to-many relationship between two entities)  
 - e.g. [ Order# + ProductID ] for an Order Line Item, City + State, ...

- RELATIONSHIP** - an association between entities. May be dependent on either entity; may be at most one or many on either entity. Represented with a FOREIGN KEY in a relational database.  
- e.g. Customer# in the Order record/table. *What in the Order Line Item table?*

### ☒ Taxonomy of Data Structures

DMADO 16

Based on what the system knows! Everest-DM-4p.121.

**(1) SINGLE FILE** - data items grouped into one entry type

- FLAT FILE "TABLE" \* - single valued items
- HIERARCHICAL - nested repeating groups of items - multiple levels of nesting
  - SINGLE PATH
  - MULTIPATH ("BRANCHING") -- e.g. COBOL

**(M) MULTIFILE (E[A]R)** - data items grouped into multiple [related] entry/record types

- HIERARCHICAL RELATIONSHIPS among entry types
- "CODASYL" NETWORK - hierarchical records
- RELATIONAL (E-A-[R]) - flat records
- GENERAL ("M:N", TERNARY+) RELATIONSHIPS \*

**(O) "NO" FILE \* (OR)** - no clustering of data items into records

- BINARY RELATIONSHIPS - e.g. NIAM/"Binary" Modeling
- GENERAL RELATIONSHIPS - e.g. ORM (Object-Role Modeling - Halpin)

\* 3 missing from the "traditional" taxonomy.

**RECORD-BASED (Clustered Data Items)**

### ☒ Data Modeling Schemes - Clustered

DMADO 17

**HIERARCHIC** - single file  
 - nested repeating groups  
 - implicit hierarchical relationships  
 (special case)

**NETWORK** - multifile, hierarchical record  
 - defined relationships  
 => semantic/OBJECT models

**ER** - Focus on E & R, hidden record structure  
 - Usually flat records [optionally with attributes]  
 - Defined relationships (general M:N)  
 - Usually restricted to binary relationships

**RELATIONAL** - Multifile; flat records only  
 - Relationships as foreign keys  
 so no M:N relationships

### ☒ Taxonomy of "Clustered" Data Structures

DMADO 18

		Single File		Multiple Files	
Intra-Record Structure	Clustered	<b>SINGLE FLAT FILE ("TABLE")</b> ↓ <b>HIERARCHICAL FILE</b>	<b>RELATIONAL ("TABLES")</b> ↓ <b>(CODASYL) NETWORK</b>		
	Flat				
		Flat	Nested		

These data modeling schemes are distinguished at the schema level.

### A Typical "Flat" File

DMDC

19 (1) FILE NAME (Entity) (2) ATTRIBUTE NAME

Schema Definition (5)

Emp No	Employee Name	Org	Job	Position	Birth	Prim	Secondary	Skills	Actual
45584	PETERSON, N.M.	2000	0110	HEAD DIVISION MANAGER	M	280607	0110	6130 6625 6040	56000
32579	LYNN, K.R.	2000	5210	EMP, SECRETARY	F	530121	5210	5520	12000
57060	CARR, P.J.	2100	1110	HEAD MANAGER DEVELOPMENT DEPT	M	303720	1110	1130 1135 0130 1355	48000
15324	CALLAGAN, R.F.	2100	5210	EMP, SECRETARY	F	550606	5210	5520 5220	10800
10261	GUTTMAN, G.J.	2110	1110	HEAD MANAGER SYSTEMS GROUP	M	301110	1110	1130 1135 0150	35000
72558	HARRIS, D.L.	2110	5210	EMP, SECRETARY	F	550517	5210	5520	8400
24188	WALTERS, R.J.	2111	1110	HEAD CHIEF PROPOSAL SECTION	M	260202	1110	1120	28000
21675	SCARBOROUGH, J.B.	2111	1120	EMP, MECH ENGR	M	240914	1120		21000
18130	HENDERSON, R.C.	2111	1130	EMP, ELEC ENGR	M	240121	1130		23000
81152	GARBER, R.E.	2111	1130	EMP, ELEC ENGR	M	440707	1330	1130	16400
30793	COMPTON, D.R.	2111	1350	EMP, COST ESTIMATOR	M	260328	1350	1351 1355 1130	16200
81950	FRIEDMAN, J.M.	2112	1110	HEAD CHIEF DESIGN SECTION	M	260317	1110	1130	20000
21777	FRANCIS, G.C.	2112	1110	EMP, SYSTEMS ENGR	M	321111	1110	1130	24000
24749	FALLONER, W.M.	2112	1120	EMP, MECH ENGR	M	400621	1120	1130 1330	24000
13581	FITZGER, G.J.	2112	1130	EMP, ELEC ENGR	M	431218	1130	1355	20000
82802	APGAR, A.J.	2112	1130	EMP, ELEC ENGR	M	500715	1130	1330	21000
63653	BLANK, L.F.	2112	1330	EMP, DRAFTSMAN	F	491010	1330		18000
22959	BIRDGS, G.R.	2115	1110	HEAD CHIEF PROD SPEC SECTION	M	400598	1110	1120	24000
28414	ARTHUR, P.J.	2115	1120	EMP, MECH ENGR	M	300109	1120		22000
37113	ARNETTE, L.J.	2115	1130	EMP, ELEC ENGR	M	450729	1130		22000

ENTRIES describing ENTITIES in the real world

(4) VALUE (3) DOMAIN

Where is the "schema" definition for the File?  
Data Modeling is always done at the Schema level

MANAGER CHIEF SECRETARY  
MECH ENGR ELEC ENGR SYSTEMS ENGR  
DRAFTSMAN COST ESTIMATOR ...

### Hierarchical Data Structure in COBOL

DMDC

20 Central characteristic is replication of data within a record, formally defined and known to the system. (Also called nested records or nested relations.)

In COBOL:  
FD EMPLOYEE-FILE  
01 EMPLOYEE-RECORD  
02 Employee-Number PIC 9(5).  
02 Employee-Name PIC X(20).  
02 #Skills PIC 9.  
02 Salary PIC \$\$\$,ZZ,9.  
...  
02 Skill OCCURS 5 TIMES [ DEPENDING ON #Skills ]

### A Hierarchical Structure (? Tree)

DMDC

21 IN SCHEMA DIAGRAMS - single path, multipath:

What if an EMPLOYEE is in an unauthorized POSITION?  
What is the real relationship between EMPLOYEE and SKILL?

- Now what is the relationship between EMPLOYEE and POSITION?
- What is the real relationship between EMPLOYEE and SKILL?

### Student-Course Data Model - Entities

DMDC

22

### CODASYL Network Data Structure

RECORD

23 The "hierarchy" of data constructs:

- Added the "SET" construct to relate COBOL files.
- A COBOL record has a hierarchical structure.
- Became the ANSI NDL standard (1986)

- A named collection of RECORDs and SET relationships among them.
- A named collection of RECORDs forming a two-level hierarchy; an OWNER record type and one or more MEMBER record types.
- A named collection of DATA ITEMS and/or DATA AGGREGATES
- A named collection of DATA ITEMS or DATA AGGREGATES within a record. May be a (non-repeating) GROUP, MULTI-VALUED DATA ITEM ("vector") or, a REPEATING GROUP of data items ... OCCURS clause defines multiplicity.
- The smallest (atomic) named unit of data.

### CODASYL Network Data Structure Definition (DDL)

RECORD

24

- Records have a hierarchical structure (contrast with Relational)
- Cannot directly represent M:N relationships (same as Relational)
- A SET may have zero or multiple MEMBER record types
- INSERTION may be AUTOMATIC, STRUCTURAL, or MANUAL
- RETENTION may be FIXED, MANDATORY, or OPTIONAL
- STRUCTURAL clause is optional

RECORD Part IDENTIFIER PART# PART#, Name, Cost, ...  
RECORD Supplier IDENTIFIER SUPP# SUPP#, Name, Location, ...  
RECORD Order IDENTIFIER O# O#, P#, S#, Date, Quantity

SET Part-Order OWNER Part MEMBER Order INSERTION AUTOMATIC, RETENTION MANDATORY STRUCTURAL P# OF ORDER = PART# OF PART SELECTION STRUCTURAL  
SET Supp-Order OWNER Supplier MEMBER Order INSERTION AUTOMATIC, RETENTION MANDATORY STRUCTURAL S# OF Order = SUPP# OF Supplier

**RELSQL** **Relational Data Modeling Scheme**

25

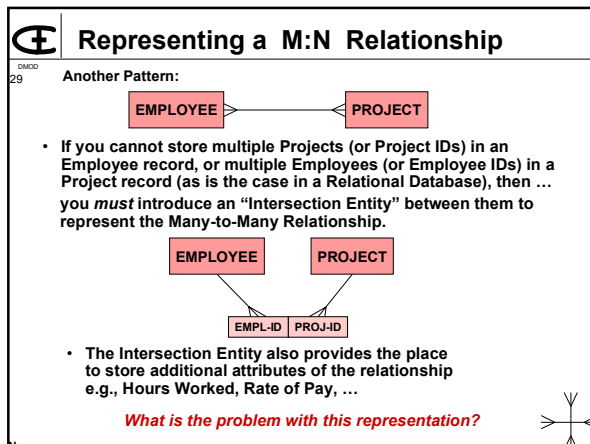
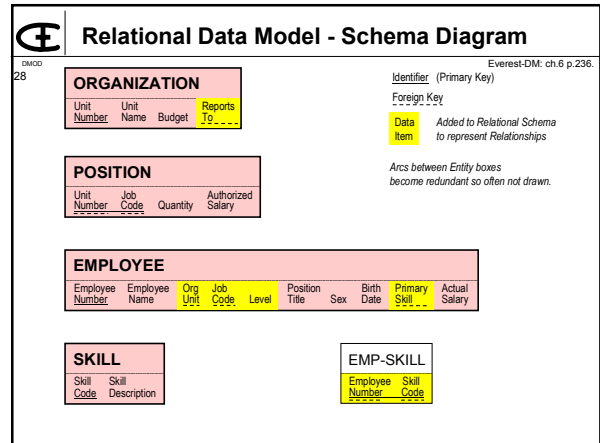
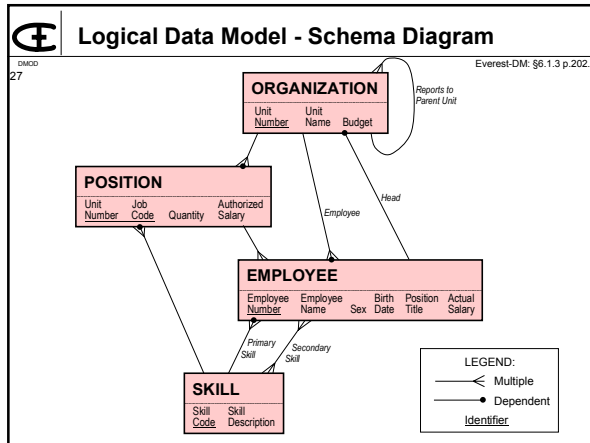
- E. F. "Ted" Codd, 1970
- Introduced new terminology: Relation, Table, Tuple, Row, Column ("Domain"), Foreign Key, Join
  - Relation = a set of (unordered) n-tuples taking one value from each Domain
- Main contribution was the high (set)-level operations manipulating multiple records at a time
  - (in contrast to CODASYL one-record-at-a-time processing)
  - Laid the groundwork for 4GL, and SQL
- Added some rigor – based on set theory (ordered pair) and first order predicate calculus
- All Information represented at the *logical level*, in user-defined, atomic (single-valued) fields, thus
  - Mandatory primary key (entity rule – guaranteed access)
  - Foreign key to represent relationships between entities (in contrast to CODASYL Network info-bearing physical pointers)

**RELSQL** **Forming a Relational Data Structure**

26

**Some rules:**

- Define a TABLE or "Relation" for each Entity type
  - Types of Entities: base/reference, dependent ("weak"), association/intersection, event/transaction
  - Assumes mutually exclusive (non-overlapping) populations
- SINGLE-VALUED ITEMS ("flat" tables) => 1NF
  - If multivalued or nested repeating group of items, put into a separate table
- IDENTIFIER for every table (entity "integrity")
- FOREIGN IDENTIFIERS to represent all relationships
  - 1:M - stored in the child / dependent entity
  - 1:1 - should probably merge into one table
  - M:N - must introduce an association/intersection table
- NORMALIZE to second and third normal form
  - important for good design
  - but not enforced by RDBMS... *WHY?*



**RELSQL** **ANSI SQL**

30

- Defines both data structure and manipulation
- 1986 the first standard (along with ISO), updated 1992
- "Flat" records – atomic attributes not support complex objects<sup>^</sup>
- User-defined and managed identifiers<sup>^</sup>
- Referential integrity on Foreign Keys
- User defined domains
- Not require record identifiers<sup>\*\*</sup>
- Permits duplicate rows in a relation (table)\*

\* Makes ANSI SQL not truly relational  
<sup>^</sup> Contrary to the needs of object orientation (motivated ANSI SQL:1999)

**ANSI SQL:1999**  
**"Great News: The Relational Model is Dead"**

31 By: Michael Gorman, Whitmarsh Information Systems (WIS)  
 (get a copy of the paper at his web site - address below)  
 Secretary ANSI NCITS H2 - Technical Committee on Database,  
 responsible for the SQL standard.

**SQL:1999 - Record Structures**

32 **GENERIC CLASSIFICATION:**

- Single valued items <----- *Only one in ANSI SQL2*
- Multivalued items
- Groups of items (non-repeating)
- Repeating Groups of items
- Nested Repeating Groups of items

**IN SQL:1999 - no longer adhering to the relational model**

- New built-in data types
- BLOB (Binary) and CLOB (Character)(Large Objects)
- Abstract Data Types (user defined with behavior)
- Array - with outward references to other (external) data
- Row types (nested repeating groups - hierarchical)
- User-Defined Functions (derived items)

**SQL:1999 - Relationships**

33 **Between record types:**

- One-to-one <----- *Only ones in ANSI SQL2*
- One-to-many <-----/
- One-to-many (multiple member record types)
- Many-to-many - modeled directly
- Singular (single and multiple member record types)
- Inferential
- Recursive

**Functional Dependency in Relationships**

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

is functionally dependent on

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**

35 **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.  
 (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**

36 **Resulting from (clues to) poor database design:**

EMPLOYEE#	EMPNAME	SKILL	PROFICIENCY ...	BOSSNAME	DEPT#	DEPTNAME
-----------	---------	-------	-----------------	----------	-------	----------

- 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.

**Normalization – Testing your Understanding**

37 Assuming that **A** is single valued with respect to **X** (i.e. 1NF).  
**GIVEN:** Could you have a violation of: (if not, why not?)

<b>X</b> <b>A</b>	2NF?	3NF?	4NF?
<b>X</b> <b>A</b> <b>B</b>	2NF?	3NF?	4NF?
<b>X</b> <b>A</b> <b>B</b>	2NF?	3NF?	4NF?
<b>X</b> <b>A</b> <b>B</b>	What does this diagram mean? How does this differ from diagram above, if any?		

**Normalization Example**

38

To find and remedy the violations of Normal form:

- Show all the Identifiers
- Show all the Functional Dependencies
- Remove all the offending non-key attributes
- Create additional tables to contain those attributes

EmpID	ProjID	EmpName	EmpDept	DeptBoss	EmpSkills	ProjTitle	ProjBoss	HoursWorked
-------	--------	---------	---------	----------	-----------	-----------	----------	-------------

**Object-Oriented Programming Paradigm**

39

- Object Orientation is essentially a *process view*  
– From ACTION -> OBJECT to OBJECT -> ACTION (right click)
- Encapsulation is the distinguishing characteristic

In pure O-O, everything is an OBJECT.

**Object Databases**

40

**Data Centric OO:**

- Still record-based, thus augmented ER Diagrams
- Object (actor) is an entity that behaves (acts)

Object Model = a data model with attitude, that has learned how to behave; an ER model in disguise – D. Hay

- Class Diagram in UML
- Data Model Diagram in Rational Rose

**OBJECT**

NAME	
Object ID	- Global Object identifiers (OIDs); system managed
Attributes	- Complex structure (violating 1NF)
...	
Methods	- "Behavior" added to the ER Diagram Schema
...	

**Data Warehouse Modeling**

41

- **OBJECTIVE:** a schema optimized for decision support  
– getting useful information **OUT**, reliably and consistently  
– efficient and accurate analysis  
– accuracy & consistency demand a data warehouse protected from change and whose integrity is rigorously maintained
- **The USER is a manager or business analyst**  
– knows the business  
– wants to drill down into the facts  
– to find patterns, trends, anomalies which drive decision making
- **NEEDED:** *simplicity* in the user view of data  
– Designed for the user  
– Designed around a subject category  
– NOT to model the real world directly, exclusively, or completely

**How can we achieve simplicity (and performance)?**

REF: Livingston & Rumbsky, Planning & Designing the DW, B&E.

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

42

Annual product sales by region (\$,000)

REGION:					
PRODUCT:	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
<b>TOTALS:</b>	<b>\$21,000</b>	<b>\$44,500</b>	<b>\$40,000</b>	<b>\$47,500</b>	<b>\$153,000</b>

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

### 43 Sales Data

in a Relational Table:

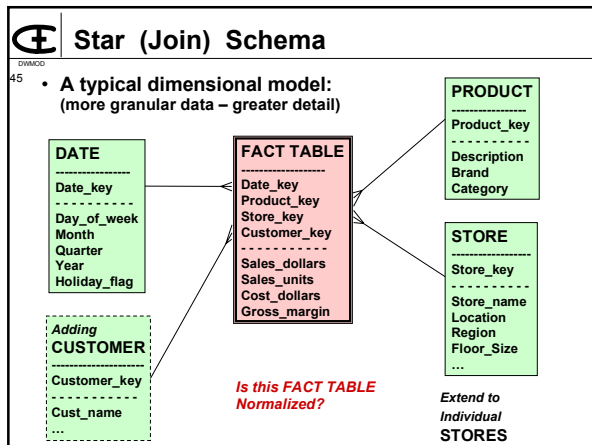
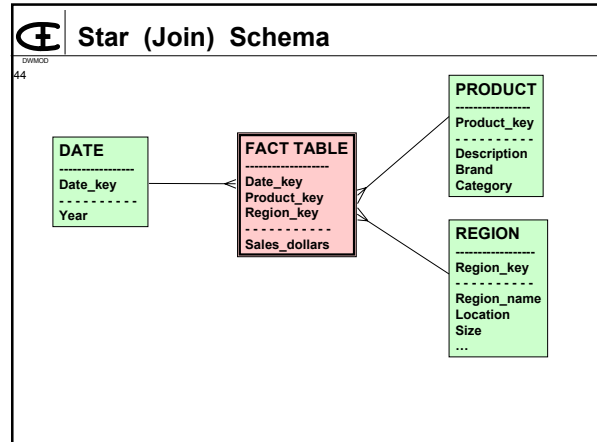
*How many Facts?*  
*What is the Identifier?*  
*How many Dimensions?*  
*Where are the Dimension Tables?*  
*How many rollup levels?*  
*What is the business process?*  
*What is the Grain?*  
*How far can you Drill Down?*

REGION	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

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

Aggregations (Rollups)



### 46 The "Original" Relational Database

S	SN	SNAME	STATUS	CITY
S1	Smith	20	London	
S2	Jones	10	Paris	
S3	Blake	30	Paris	
S4	Clark	20	London	
S5	Adams	30	Athens	

SP	SN	PN	QTY
S1	P1	300	
S1	P2	200	
S1	P3	400	
S1	P4	200	
S1	P5	100	
S1	P6	100	
S2	P1	300	
S2	P2	400	
S3	P2	200	
S4	P2	200	
S4	P4	300	
S4	P5	400	

P	PN	COLO R	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

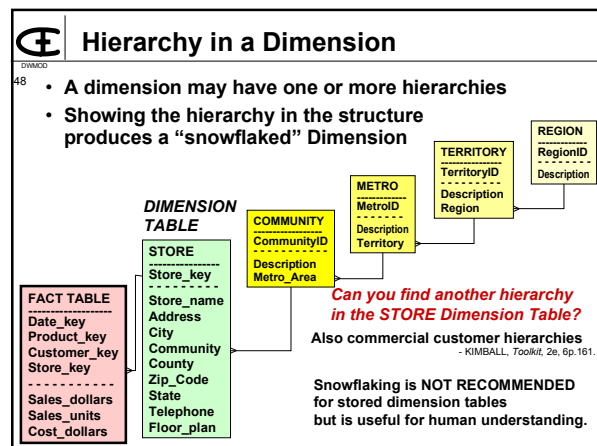
*Draw the Schema ER Diagram.*  
*What do you observe about the diagram?*

### 47 Example Dimension Table

REF: B&A, Ch.14

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
*	Atlanta	3	Southeast	221	Hamline	1
135	Chicago 2	2	Midwest	232	Dale	1
140	All					3

• Descriptions added (\*)  
 • Levels added



### Flattening the Hierarchy in a Dimension

49

- 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

**FLATTENED DIMENSION TABLE**

STORE  
Store\_key  
Store\_name  
Address  
City  
Community  
Metro\_Area  
Territory  
Region  
County  
Zip\_Code  
State  
Telephone  
Floor\_plan

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

### Stages of Data Modeling

50

**Start at the highest Conceptual Level!**

**ORM**

- Objects
- Obj. ID's
- Roles/Relships
- (Fnl. Dep)
- NO clustering => NO "attributes"

**CLUSTERED ER**

- Attribs in Records
- MultiValued, Nested
- Ternaries
- M:N
- Normalized (2,3,4)
- Relationships w/attributes
- Sub/SupTypes

**"LOGICAL" RELATIONAL**

- Flat (1NF)
- Binary only
- 1:Many only
- Primary Keys
- Foreign Keys

**PHYSICAL**

- Implementation in/for a DBMS
- Denormalize (for performance) + triggers, stored procedures

DATABASE SCHEMA

### ER / Record-based Modeling

51

CLUSTERING of ATTRIBUTES into RECORDS/RELATIONS

- NOT a necessary or desirable first step
- gets us into trouble: if too much, must decompose to normalize

### Object-Role (ORM) Data Modeling

52

**THE ESSENTIAL DIFFERENCE:**

- Three main constructs ..rolled into.. Two main constructs

Record-based modeling: ENTITY, ATTRIBUTE, RELATIONSHIP

NIAM/ORM modeling: ? ? ? ?

What to call it? OBJECT, ENTITY, ENTRIBUTE!

Role in RELATIONSHIP

### Data Modeling Terminology

53

O-R ("conceptual")	E-R ("logical")	COBOL/DBTG ("physical" implementation)	RELATIONAL
OBJECT	ENTITY (TYPE)	RECORD TYPE	RELATION TABLE
FACT SENTENCE	ATTRIBUTE	DATA ITEM (ELEMENT)	COLUMN FIELD
	INSTANCE	RECORD	ROW TUPLE
	IDENTIFIER		KEY
PREDICATE	RELATIONSHIP	"SET"	FOREIGN KEY
CONSTRAINT	CHARACTERISTICS		CONSTRAINT

### Record-based Design

54

**WHAT DOES THIS "RECORD" REPRESENT?**

X A B C

X A  
Design minimal records with at most one non-key domain.  
Now what do these "records" represent?

X B  
Perhaps Codd was right in naming it a \_\_\_\_\_!

X C  
Avoids spurious associations, e.g., A - B ...  
Could there be any violations of normal forms?  
What about the representation of the entity X?  
What if A is related to other "entities"?

**Transform Record-based (ER) Design**

55 TO REALLY REPRESENT THE ENTITY DOMAINS

Object Role Model:

**At the Root,**

# What's wrong with ER Modeling?

HINT: What is the Achilles' heel of data modeling?  
 How do we know when we have a good database design?  
 Can a DBMS or Data Modeling CASE tool help?

**Denormalization**

57

- 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)

**Data Modeling - Representation Stages**

A SECOND CUT:

- Conceptual** (ORM<sub>HALPIN/NUSSSEN</sub> SUMM<sub>FULTON</sub> UDM<sub>COMTG</sub>)
  - only what the user knows or needs to know
  - functional dependencies fully represented
  - Elementary Facts - no clustering of "attributes" into "records"
- Clustered** (ER<sub>CHEN</sub> EER<sub>TEOREY</sub> SDM<sub>AL-EOD</sub> SOM<sub>KROENIG</sub> SQL:99<sub>ANSI</sub> UML)
  - identifiers (attributes or dependent relationships)
  - keep: M:N, ternary relationships, super/subtypes, attributed relationships, multi-valued items/rgroups
- "Logical"** (RELATIONAL<sub>COOD</sub> SQL<sub>ANSI</sub> IE<sub>Finkelstein</sub> IDEF1X<sub>US Govt</sub>)
  - flat files/tables; - stored identifiers; - 3NF (decompose)
  - resolve: M:N, ternary, super/subtype relationships
  - foreign keys to represent relationships
- Denormalize (Recluster) - for performance
- Physical** (IMPLEMENTATION in a DBMS)
  - triggers, stored procedures, user code to represent and enforce semantics beyond the DBMS.

**Next Generation(s) DBMS**

59

Design / Modeling

Implementation

Entity-Relationship ER Diagrams

Relational RDBMS

Object-Oriented DBMS

OBJECT-ROLE MODELING

DATA WAREHOUSING for decision support

HETEROGENEOUS MULTIMEDIA information