# *Data Collection and DBMS*

# Agenda

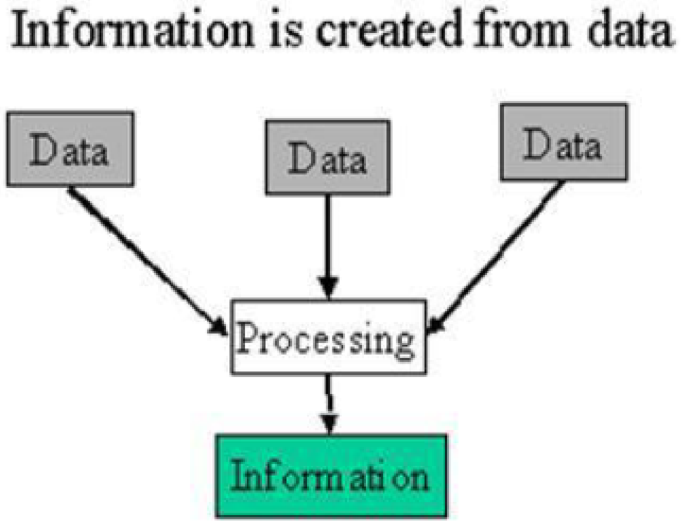
- Database Concepts (File System and DBMS)
    - What is file system, its need?
    - What is DBMS, its need
    - Codd’s 12 rules for RDBMS

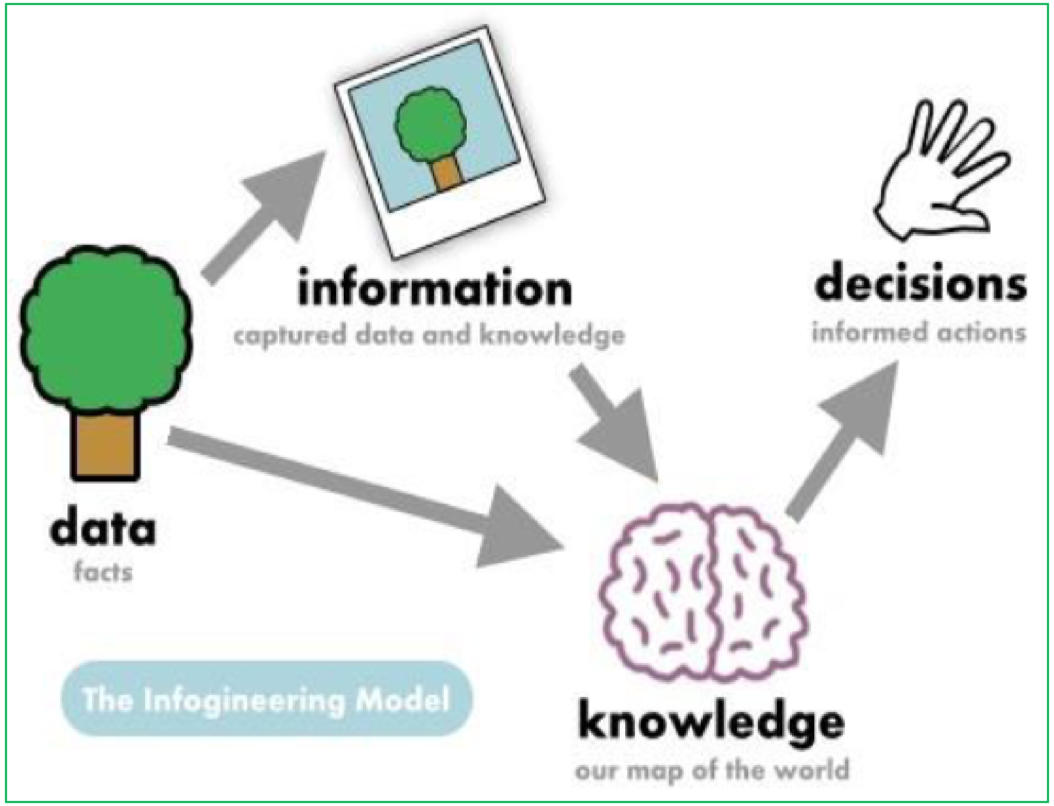
# The Role of Data

***Data**:* Known fact that can be recorded and that has implicit meaning.
Ex. *Name*, *Tel_no*, *city* etc

When **Data** Is Gathered And Analyzed It Yields **Information**

**Information** helps us **foresee** and **plan events**





# TYPES OF DATA STORAGE

Non Computer oriented

As well as Computer oriented File Systems

Database Oriented

Distributed Databases

# Three Pillars of DBMS

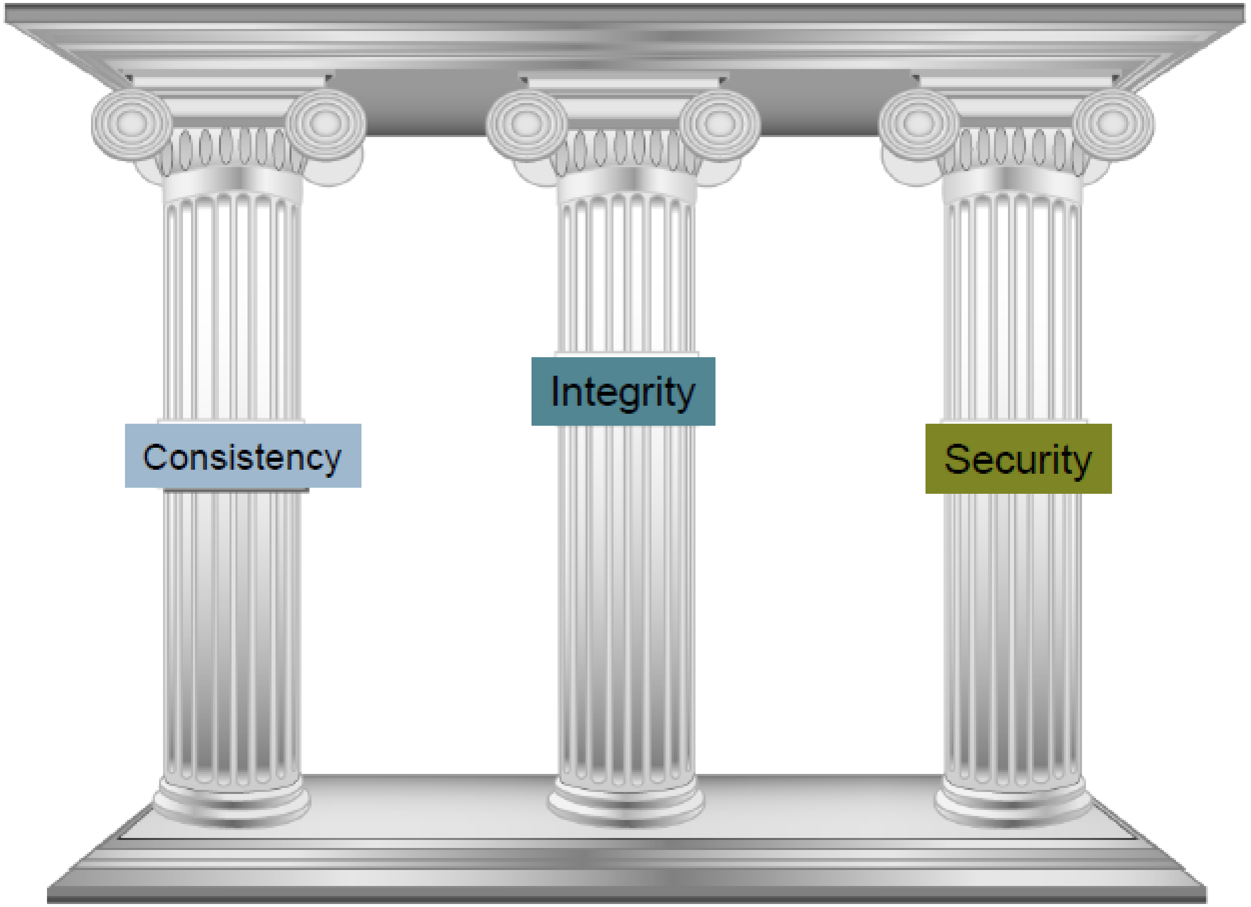
Problem 1

Business needs are always changing, so if we hard code data into programs then the entire programs need to be changed every time

Problem 2

When data is stored in several locations/files, it is difficult to make the changes in all the locations/files at the same time

# Three Pillars of DBMS



# REAL WORLD ENTITIES AS TABLES

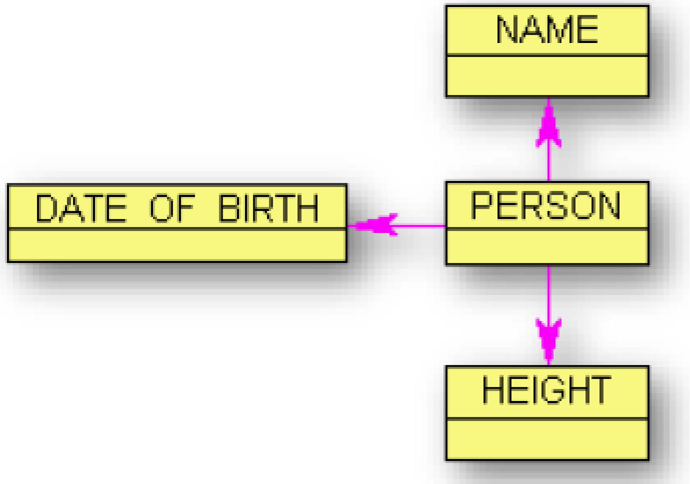
PERSON

AS

| PERSON |
|---|
| NAME |
| HEIGHT |
| BIRTH |

By organizing information into tables, we have already imposed many rules (constraints),

- For example, we structure a PERSON table with exactly ***one column*** for DATE_OF_BIRTH to implement the *obvious business rule* that one human was born only once.

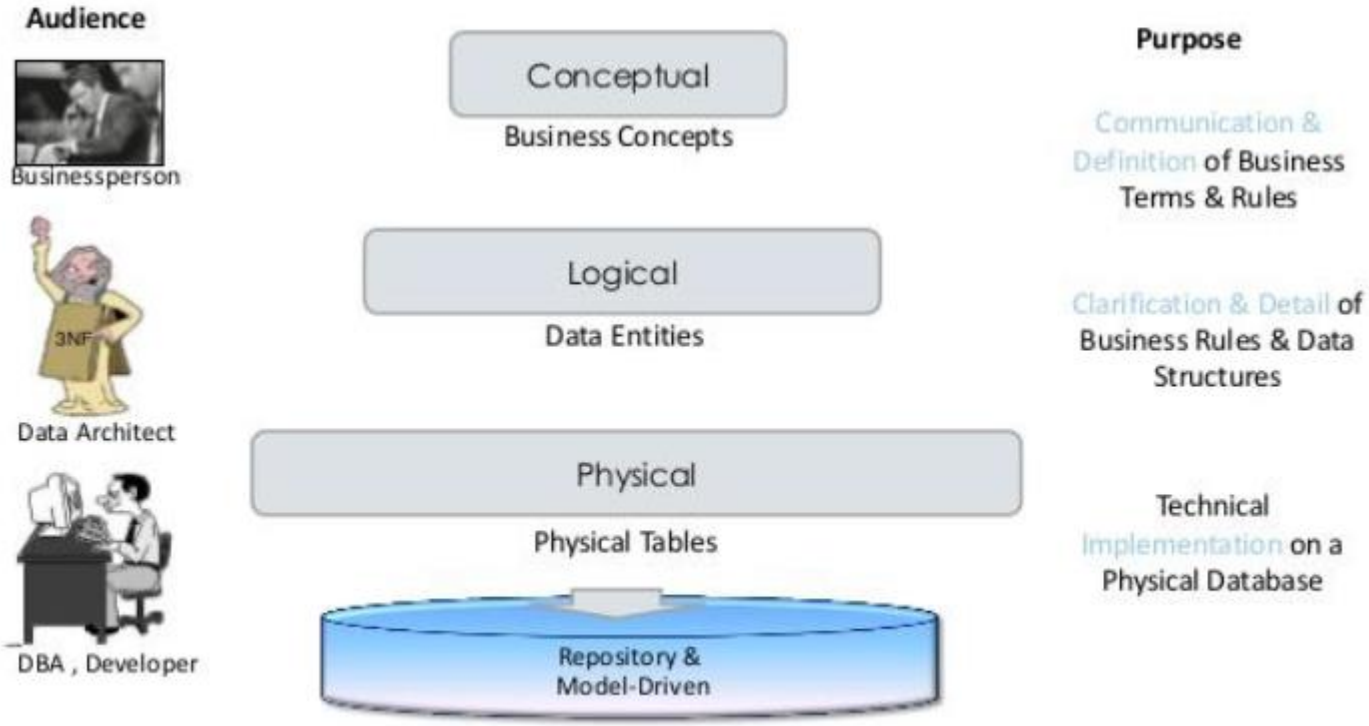


# Embedding Business Rules

## (Adding A Simple Constraint)

```
create table PERSON ( NAME char(30) )
create table PERSON ( NAME char(30) not null)
```

# Multiple Models – Multiple Purposes – Multiple Audiences



Audience

Businessperson

Data Architect

DBA , Developer

Conceptual

Business Concepts

Logical

Data Entities

Physical

Physical Tables

Repository &
Model-Driven

Purpose

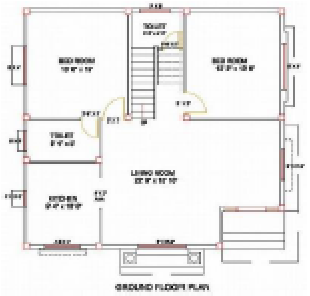
Communication & Definition of Business Terms & Rules

Clarification & Detail of Business Rules & Data Structures

Technical Implementation on a Physical Database

# Database Design Levels

Finished Product



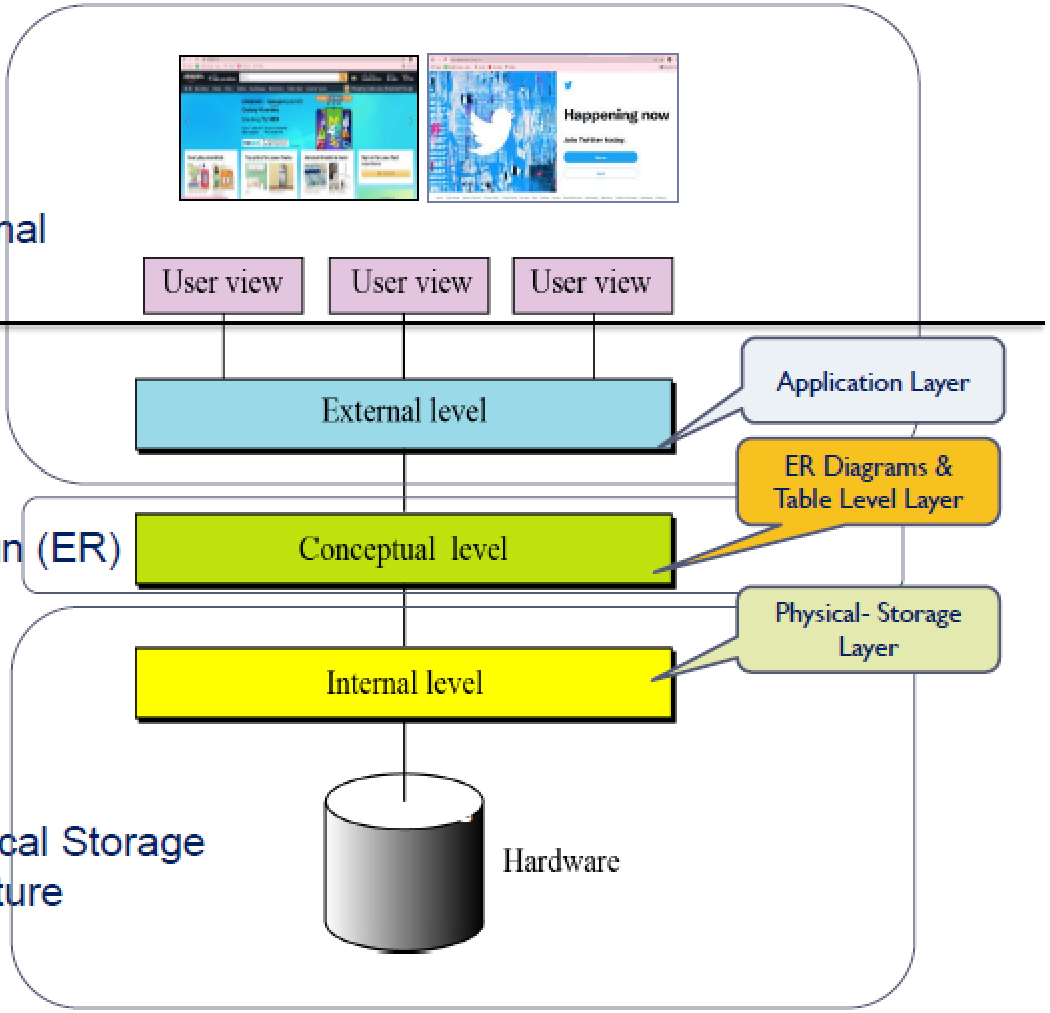
Conceptual/Design Plan

Physical/Structure

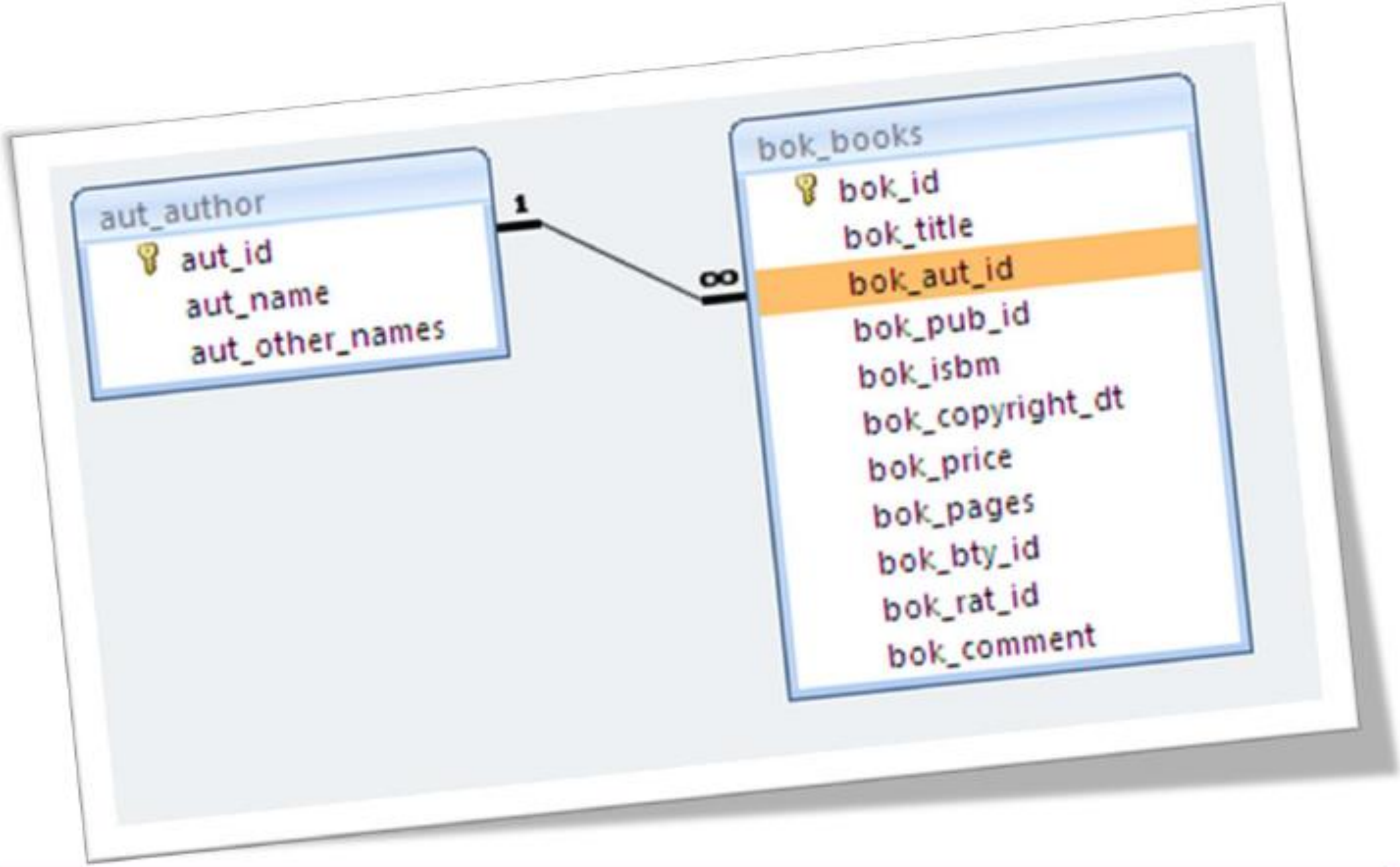
External

Design (ER)

Physical Storage Structure



# LOGICAL MODEL- TABLES & KEYS

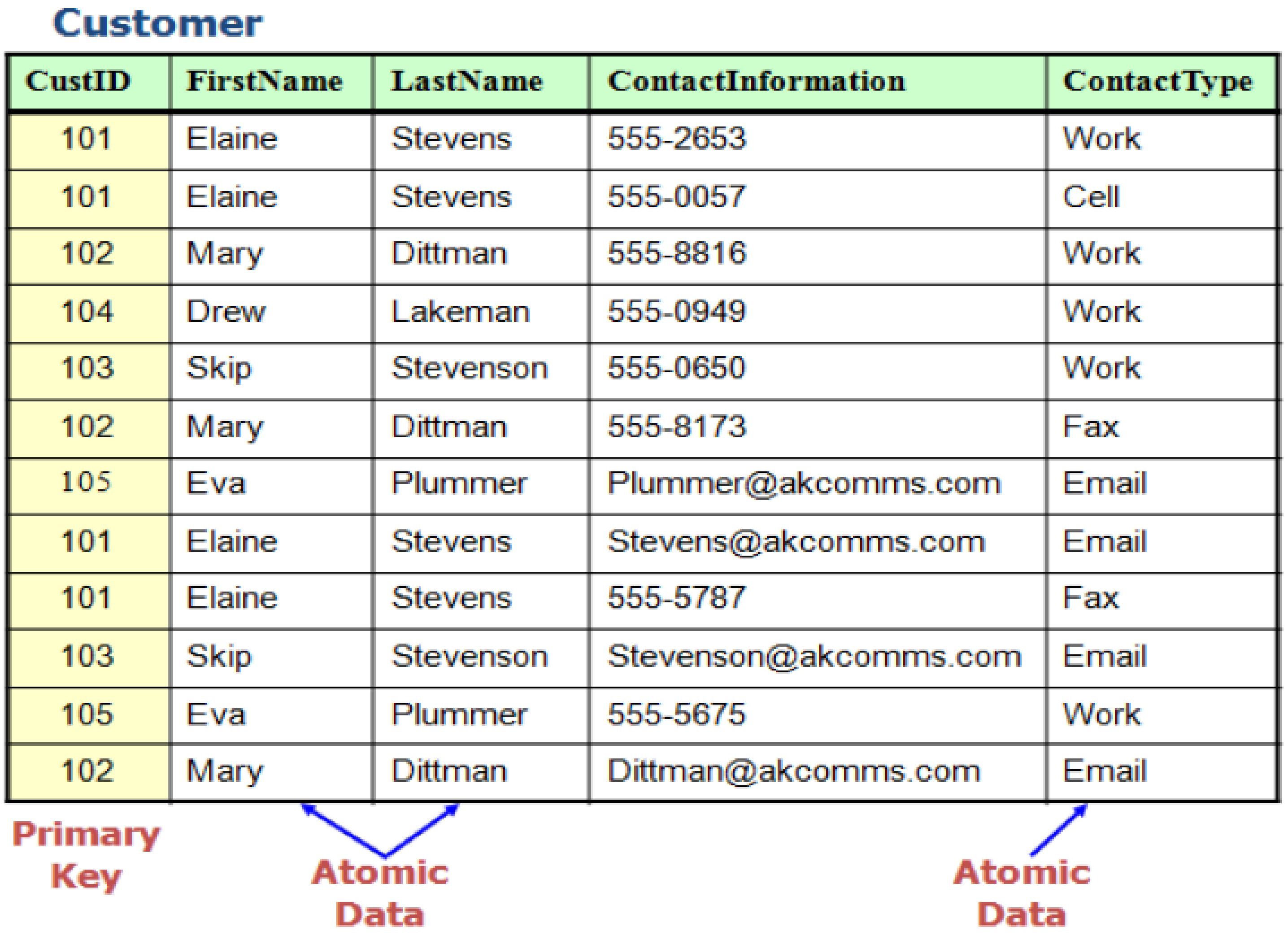


aut_author
- aut_id
- aut_name
- aut_other_names

bok_books
- bok_id
- bok_title
- bok_aut_id
- bok_pub_id
- bok_isbm
- bok_copyright_dt
- bok_price
- bok_pages
- bok_bty_id
- bok_rat_id
- bok_comment

*Means available to us to navigate amongst tables* of data is by the reference of a *foreign key* to some key of another table.

# Data is stored in the form of Tables



**Customer**

| CustID | FirstName | LastName | ContactInformation | ContactType |
|---|---|---|---|---|
| 101 | Elaine | Stevens | 555-2653 | Work |
| 101 | Elaine | Stevens | 555-0057 | Cell |
| 102 | Mary | Dittman | 555-8816 | Work |
| 104 | Drew | Lakeman | 555-0949 | Work |
| 103 | Skip | Stevenson | 555-0650 | Work |
| 102 | Mary | Dittman | 555-8173 | Fax |
| 105 | Eva | Plummer | Plummer@akcomms.com | Email |
| 101 | Elaine | Stevens | Stevens@akcomms.com | Email |
| 101 | Elaine | Stevens | 555-5787 | Fax |
| 103 | Skip | Stevenson | Stevenson@akcomms.com | Email |
| 105 | Eva | Plummer | 555-5675 | Work |
| 102 | Mary | Dittman | Dittman@akcomms.com | Email |

Being understood how data can be represented in the form of table…….

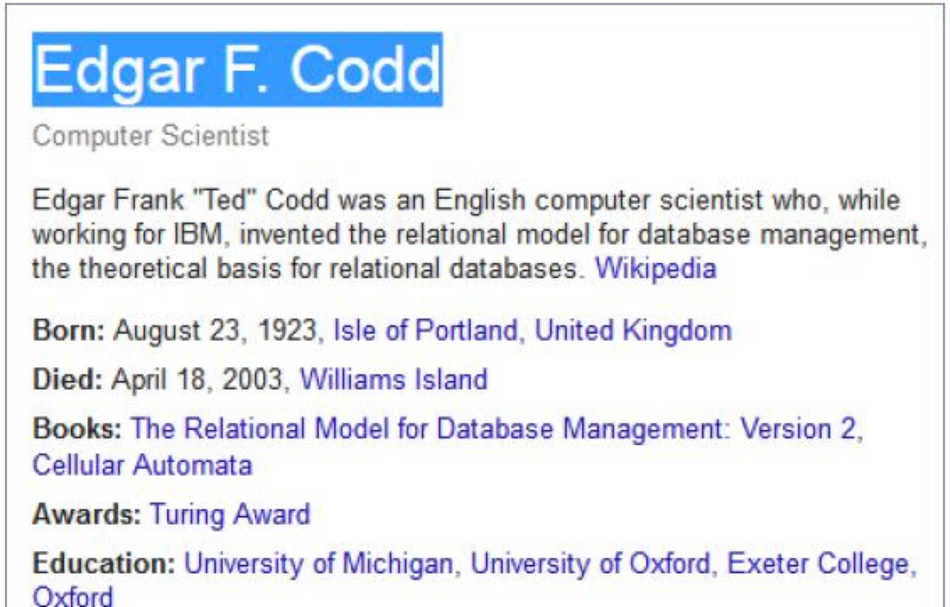
………Let us understand how data is stored in the Database

# Codd's 12 Rules for a Relational Database

# Codd's Rules

**Codd's 12 rules** are a set of thirteen rules (numbered zero to twelve) proposed by Edgar F. Codd, a pioneer of the relational model for databases,

designed to define what is required from a database management system in order for it to be considered *relational*, i.e., a relational database management system RDBMS



## Edgar F. Codd

Computer Scientist

Edgar Frank "Ted" Codd was an English computer scientist who, while working for IBM, invented the relational model for database management, the theoretical basis for relational databases. Wikipedia

**Born:** August 23, 1923, Isle of Portland, United Kingdom

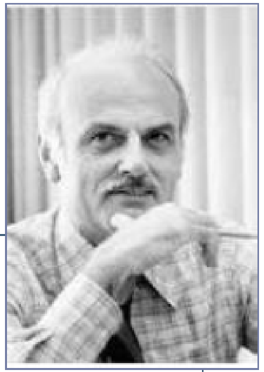
**Died:** April 18, 2003, Williams Island

**Books:** The Relational Model for Database Management: Version 2, Cellular Automata

**Awards:** Turing Award

**Education:** University of Michigan, University of Oxford, Exeter College, Oxford

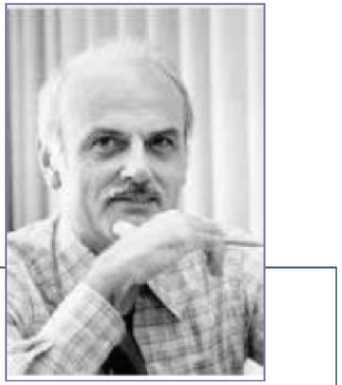
# Codd's rules…



- **Rule 0:** The system must qualify as *Relational*, as a *Database*, and as a *Management System*.

  For a system to qualify as a Relational Database Management System (RDBMS), that system must use its *relational* facilities (exclusively) to *manage* the *Database*.

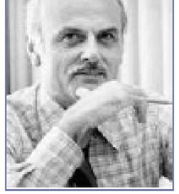
# Codd's rules…



- **Rule 1**: THE *INFORMATION RULE*

  All information in the database is to be represented in one and only one way, namely by values in column positions within rows of tables.

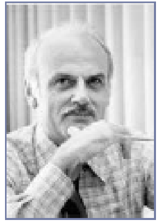
# Codd's rules…



- **Rule 2:** THE *GUARANTEED ACCESS RULE:* All data must be accessible.
- It says that every individual scalar value in the database must be logically addressable by specifying
  - the name of the containing **table,**
    - the name of the **containing column** and
    - the **primary key value of the containing row**.

      (Note: This rule is essentially a restatement of the fundamental requirement for primary keys. )

# Codd's rules…



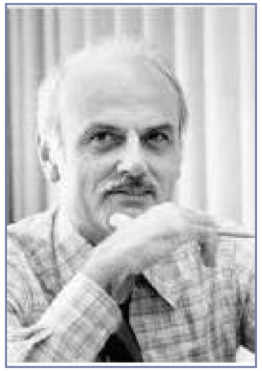
- **Rule 3:** *SYSTEMATIC TREATMENT OF NULL VALUES*

  The DBMS must allow each field to remain null (or empty).

  Specifically, it must support a representation of "missing information and inapplicable information" that is systematic, distinct from all regular values

  - for example, "distinct from zero or any other number", in the case of numeric values,
  - Independent of data type.
  - It is also implied that such representations must be manipulated by the DBMS in a systematic way.

# Codd's rules…

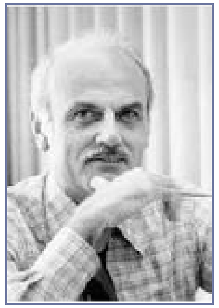


- **Rule 4**: *ACTIVE ONLINE CATALOG BASED ON THE RELATIONAL MODEL:*

  The system must support an online, inline, relational catalog (database's structure) that is accessible to authorized users by means of their regular query language.

  That is, users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data.

# Codd's rules…

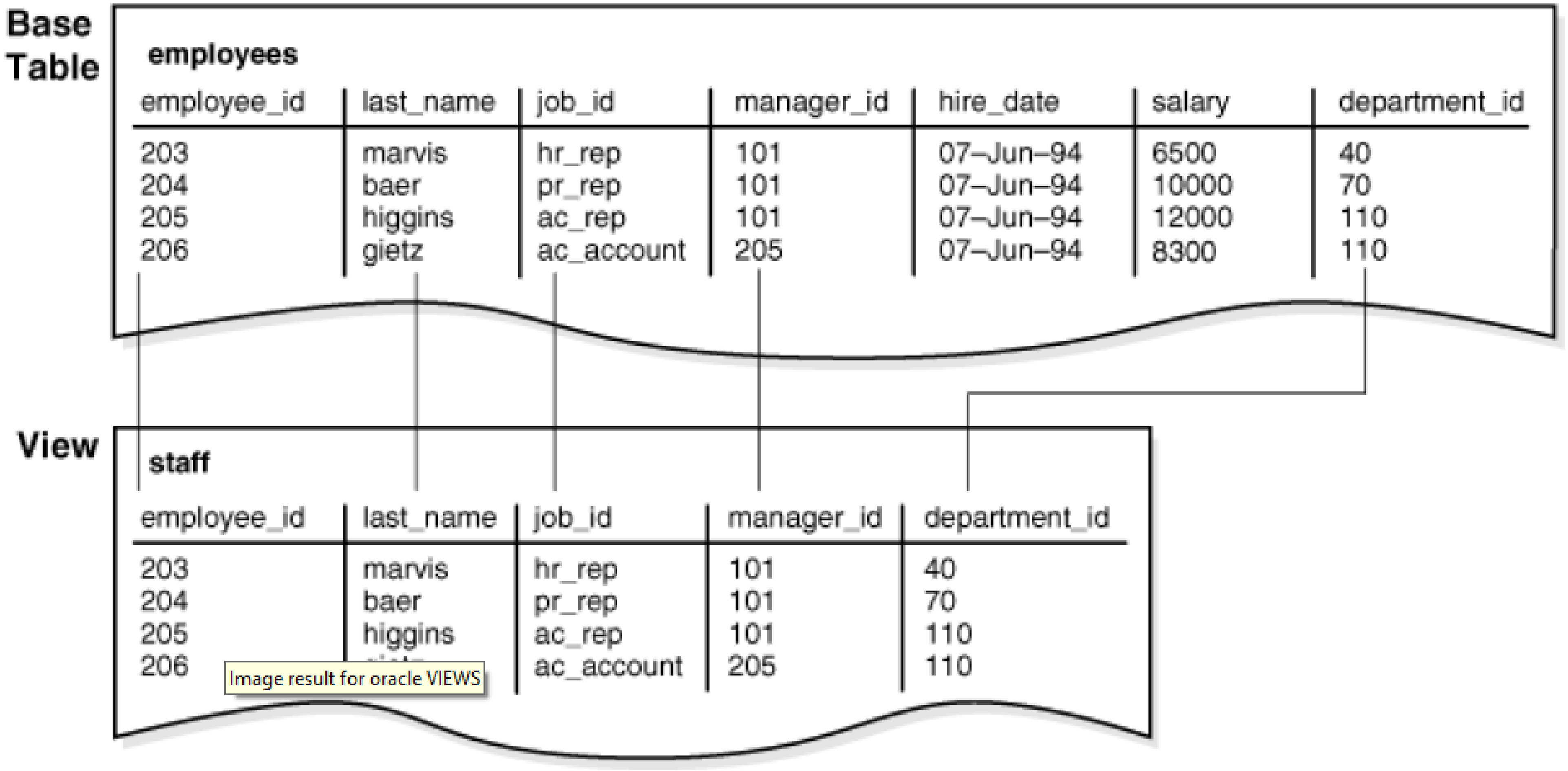


- **Rule 5:** THE *COMPREHENSIVE DATA SUBLANGUAGE RULE*

  The system must support at least one relational language that

  - Has a linear syntax
  - Can be used both interactively and within application programs,
  - Supports data definition operations (including view definitions), data manipulation operations (update as well as retrieval), security and integrity constraints, and transaction management operations (begin, commit, and rollback).

# VIEWS



Base Table

employees

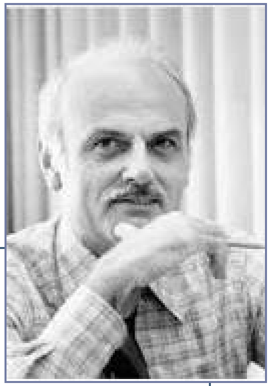
| employee_id | last_name | job_id | manager_id | hire_date | salary | department_id |
|---|---|---|---|---|---|---|
| 203 | marvis | hr_rep | 101 | 07–Jun–94 | 6500 | 40 |
| 204 | baer | pr_rep | 101 | 07–Jun–94 | 10000 | 70 |
| 205 | higgins | ac_rep | 101 | 07–Jun–94 | 12000 | 110 |
| 206 | gietz | ac_account | 205 | 07–Jun–94 | 8300 | 110 |

View

staff

| employee_id | last_name | job_id | manager_id | department_id |
|---|---|---|---|---|
| 203 | marvis | hr_rep | 101 | 40 |
| 204 | baer | pr_rep | 101 | 70 |
| 205 | higgins | ac_rep | 101 | 110 |
| 206 | [illegible] | ac_account | 205 | 110 |

# Codd's rules…

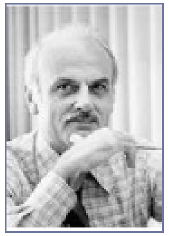


- **Rule 6**: THE *VIEW UPDATING RULE*

All views that are theoretically updatable must be updatable by the system.

If, for example, you could join three tables as the basis for a view, but not be able to update that view, then this rule would be violated.

# Codd's rules…



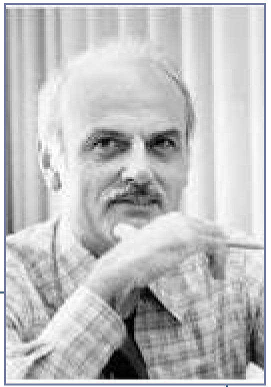
- **Rule 7:** *HIGH-LEVEL INSERT, UPDATE, AND DELETE*

The system must support set-at-a-time *insert*, *update*, and *delete* operators.

This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

This means that data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables.

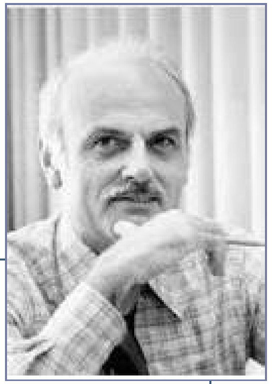
# Codd's rules…



- **Rule 8**: *PHYSICAL DATA INDEPENDENCE*

  Changes to the physical level (how the data is stored, whether in arrays or linked lists etc.) must not require a change to an application based on the structure.

# Codd's rules…

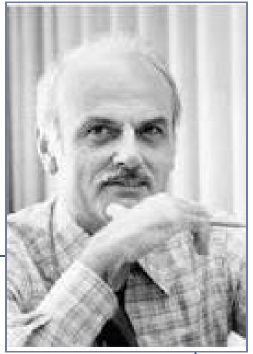


- **Rule 9**: *LOGICAL DATA INDEPENDENCE*

  Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure.

  "Logical data independence is more difficult to achieve than physical data independence."

# Codd's rules…



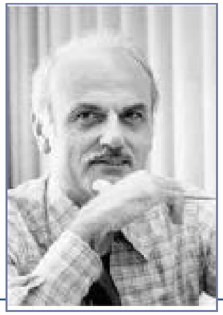
- **Rule 10:** *INTEGRITY INDEPENDENCE*

Integrity constraints must be specified separately from application programs and stored in the catalog.

It must be possible to change such constraints as and when appropriate without unnecessarily affecting existing applications.

Primary key constraints, foreign key constraints, check constraints, triggers, and so forth should all be stored in the data dictionary.

# Codd's rules…

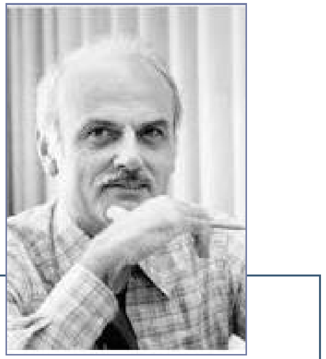


- Rule 11: *DISTRIBUTION INDEPENDENCE*

  The distribution of portions of the database to various locations should be invisible to users of the database. Existing applications should continue to operate successfully:

- when a distributed version of the DBMS is first introduced;
- when existing distributed data are redistributed around the system.

# Codd's rules…



- **Rule 12: THE *NON SUBVERSION RULE***

  If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system, for example, bypassing a relational security or integrity constraint.

  Example: A third party IDE (MyOra) or backup or load utility, for example, should not be able to bypass authentication, constraints, and locks.