TED CODD: WINNER OF THE 1981 TURING AWARD

BY

PROF. SHAMKANT B. NAVATHE

COLLEGE OF COMPUTING

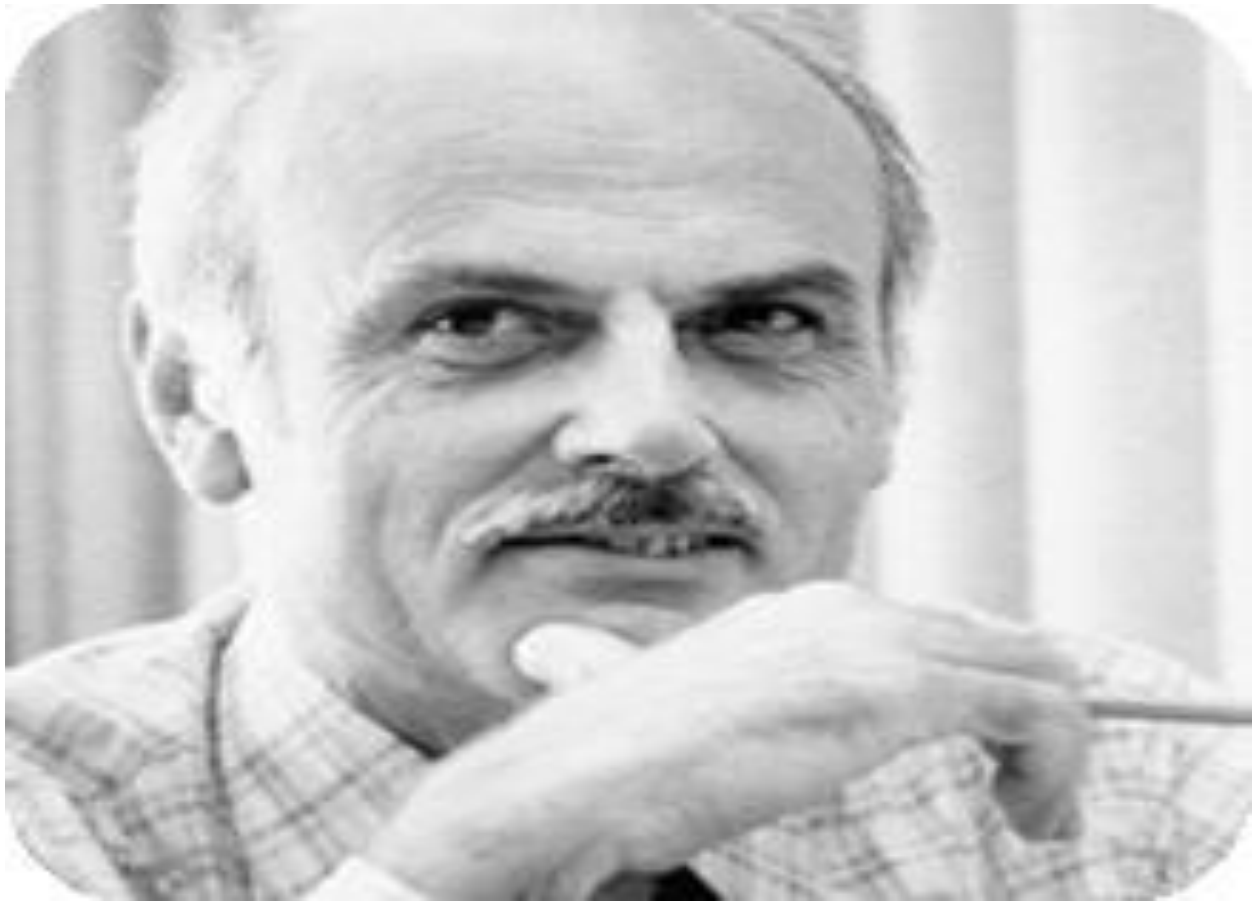
GEORGIA INSTITUTE OF TECHNOLOGY

SHAM@CC.GATECH.EDU

ANNA UNIVERSITY, CHENNAI

29TH DECEMBER 2017

Edgar Frank (E.F.) Codd 1923-2003



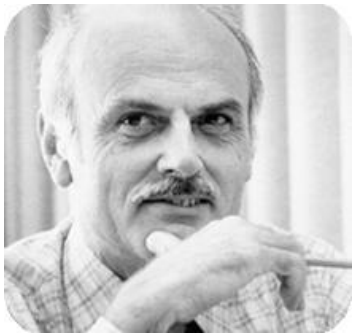
Presentation Outline

- Ted Codd and his personal profile
- The context – historical – before the Relational Model
- The aftermath – historical – after the Relational Model
- Codd's Seminal Work – the Relational Model
- Relational Model – a quick tutorial
- Extensions to the relational model
- His comments at the Turing Award ceremony
- Some concluding thoughts

TED CODD – A PERSONAL PROFILE

- He was presented with the Turing Award on Nov. 9, 1981

  “ *for his fundamental and continuing contribution to the theory and practice of database management systems*”



The Turing Award

- Turing Award is the highest recognition the organization called ACM (Association of Computing Machinery) bestows on anyone.
- The Turing Award, ACM's most prestigious technical award, sometimes called the "Nobel Prize" of computing is given for contributions of lasting importance to the field of computing. It was established in 1966.
- Established to honor the British Mathematician A.M. (Alan Mathison) Turing , 1912- 1954

The Turing Award

- A.M Turing was a British Mathematician, logician and analyst
- A movie was produced about his life (with a romantic slant) in 2014, called “The Thoery of Everything”.
- So far 52 recipients have been awarded this honor in various fields within Computer Science ranging from theory of computation to A.I. to security to programming languages and many other applied fields like databases, robotics and software engineering.

The Turing Award

In the field of databases, 4 recipients so far:

Charles W Bachman, 1973 : for introducing the concept of data structures and the network database model **("for his outstanding contributions to the database technology")**

Ted Codd, 1981: for the relational model

Jim Gray, 1998: for his work on database transactions**("for seminal contributions to database and transaction processing research and technical leadership in system implementation").**

Michael Stonebraker 2014: for fundamental contributions to the concepts and practices underlying modern database systems

TED (Edgar F.) CODD- personal history

- Born 1923 in southeast England– parents: father was a leather manufacturer and mother was a school teacher.
- Went on full scholarship to Exeter college, Oxford, initially studying Chemistry
- Voluntarily joined the Royal Air Force and served in WW II– after the war earned B.A. and M.A. in Mathematics from Oxford.
- Emigrated to the U.S. in 1948, worked at Macy's as a salesman and taught at U of Tennessee for 6 months

TED CODD – personal history-contd.

- Joined IBM in 1949- programming the “Selective Sequence Electronic Calculator”
- Left US in 1953 – went to Canada and managed a computer center in Canada
- 1957: rejoined IBM in U.S.
- Did computer design: IBM 701 and “Stretch” (IBM 7030, which led to IBM 7090 and the mainframe technology)
- Led the development of the world’s first multiprogramming system

TED CODD –personal history-contd.

- Joined University of Michigan and obtained M.Sc. And Ph.D. in computer and communication sciences in 1965.
- His Overall Academic work:
  - Ph.D.: Theory of self-reproducing automata
    - later published as a book “Cellular Automata” – by Academic Press
  - High level techniques for software specification, multiprogramming systems
  - Creation and extension of the relational approach to database management
  - Developed “Rendezvous”, an English based question answering system for casual users of relational databases

My Interactions with Ted Codd

- First saw him in the debate on “which model is going to be the “standard” for database management in SIGMOD (SIGFIDET) 1974. The debate featured speakers on hierarchical model (included Dennis Tsichritzis- who wrote the first textbook on database management, from U of Toronto) and on the network model (included Charles Bachman)
- Met and interacted with him during the 1978 summer I spent at IBM San Jose research center where he showed me the demo of his “Rendezvous” system. Also got to interact with most of the “system R” developers including Turing Award winner Jim Gray.

My Interactions with Ted Codd – contd.

- 1978-79: workshops at A.T. and T. in Holmdel, NJ where I used to attend as NYU faculty to discuss various DB related issues
- Met him at several conferences in the 1980's
- Shared stage as a speaker with him in a special invited seminar in Amsterdam by information society of Holland, 1989.

CODD's non-IBM career

- Not happy with the "incomplete implementation of the relational model," he resigned from IBM in 1984.

SQL based data is a "convenient" and "informal" representation of the relational model.

- 1985- formed with Sharon Weinberg and Chris Date two companies: Relational Institute and the Codd and Date consulting group
- Saw the relational DB industry grow into billions of dollars

Today the revenue of products and services related to relational databases amounts to over 50 billion dollars!

- Died April 18th 2003 survived by wife Sharon and daughter Katherine and sons Ronald , Frank and David.

PROGRESSION OF WORK: BEFORE AND AFTER THE RELATIONAL MODEL

The Context for Codd's Relational Model Work

- 1963-64: IBM developed the IMS Hierarchical Model-based DBMS
- 1964-65: Honeywell developed IDS – the Network Model-based database system

Bachman, C.W., "Software for Random Access Processing" Datamation, April 1965.

- 1968: First proposal for set oriented data management by Dave Childs, also from Ann Arbor , Michigan.

Childs, David L." Feasibility of a set-theoretical data structure – a general structure based on a reconstituted definition of a relation" Proc. IFIP Congress 1968, pp. 162-172.

Codd's Significant Publications -1

- The work that led to his classic tech report:

E.F. Codd, "The derivability, redundancy and consistency of relations stored in large data bases," IBM research report RJ 599, Aug. 19, 1969.

- The classic paper that was a major turning point for data management:

E.F. Codd, "A Relational Model of Data for Large Shared Data Banks," Communcations of the ACM, 13:6, June 1970, pp.377-387.

- The paper he wrote to extend the relational model, now known as the RM/T model:

E.F. Codd, "Extending the Relational Model to capture More Meaning," ACM Trans. on Database Systems, 4:4, December 1979, pp.397-434.

- His proposal for "OLAP" – a term that he coined

Codd's Significant Publications -2

- The book he published to define relational database systems:

E.F. Codd, "The Relational Model for Database Management – Version 2,"Addison Wesley, 1990.

- The twelve rules for relational database systems he proposed:

Published in Computerword, a popular weekly (1985).

Aftermath at IBM of Codd's Relational Model Proposal in 1970

- IBM initially reluctant to commercialize the idea
- INGRES: the first fully relational system arrived in 1976 from University of California, Berkeley. Later commercialized by Relational Technology Inc. (renamed as Ingres Inc.)
- Relational Software Inc. (now Oracle corporation) brought a product to market
- IBM heavily invested in "System R" that developed the relational DBMS ideas fully
- IBM commercial products: In 1981- SQL/DS for VSE and in 1983 DB2 for MVS

Relevant Research **AFTER** Codd's Relational Model Landmark Paper (before the Turing Award) -1

NON- RELATIONAL DEVELOPMENTS:

- 1971: The DBTG (Database Task Group) report from ANSI (American Natiional Standards Institute) – SPARC – a committee in charge of data management- this report endorsed the network data model.

RELATIONAL DEVELOPMENTS:

- 1975: The QBE (Query by Example) Language

Zloof, Moshe" Query By Example" Proc. National Computer Conference, 44, may 1975, pp.431-438.

- 1976: The proposal for SEQUEL language (later named as SQL for copyright reasons):

Chamberlin David, Boyce, Ralph et al." SEQUEL2: A Unified Approach to Data Definition, Manipulation and Control" IBM J. of Res. and Development , 20:6, Nov. 1976.

- 1976: The relational database system designed by PhD students at UC/Berkeley:

Stonebraker Mike et al.,"The design and implementation of INGRES,"ACM TODS, 1,3, Sept. 1976, pp. 189-222.

Relevant Research **AFTER** Codd's Relational Model Landmark Paper (before the Turing Award) - 2

- 1976: System R : first relational prototype system from IBM research

Astrahan M.M. et al. "System R: Relational Approach to database management," ACM TODS, 1,2, June 1976, pp.97-137.

- 1979: First proposal for cost-based query optimization:

Selinger, Patricia et al.," Access Path Selection in a Relational Database System" ACM SIGMOD 1979.

- 1981: The database transactions concept

Gray, Jim, "The transaction concept: Virtue and Limitations," VLDB 1981, pp.144-154

- 1981: Announcement of the first commercial relational database product by IBM:

SQL /DS (SQL data system for VSE): A relational data system for application development, IBM corp. Data Processing Division, G320-6590, Feb. 1981.

CODD'S SEMINAL WORK: THE RELATIONAL DATA MODEL

Codd's Motivation in proposing the Relational Model

- There were two main motivations behind why Codd came up with the relational model:

1)The dependence of a programmer or application developer on the way the data is modeled, stored, and navigated:

Ordering dependence

Indexing dependence

Access path dependence

2) Loss of Programmer productivity due to the effort spent on "manual optimization"

A simple illustrative example

- From Codd (1970),

Typical example from hierarchical systems:

Navigational problems, ordering of instances ?
access paths?, indexes?

Relational Solution

- THREE RELATIONS:
- STUDENT

Stud#	Sname	Major

COURSE

Course#	Cname	Text

ENROLL

Stud#	Course#	Grade

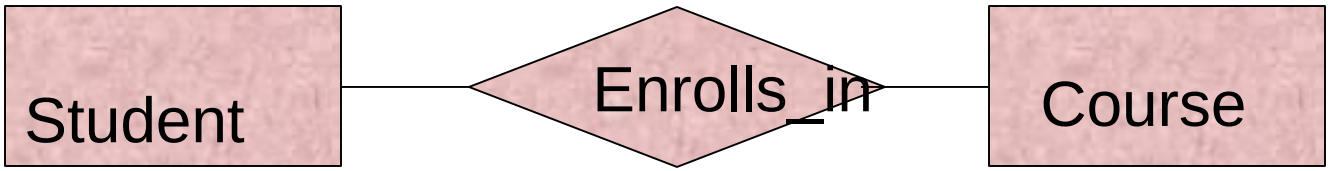
Fundamental Concepts from Codd's 1969 and 1970 papers

- The idea that a relation can represent an Entity as well as an Inter-entity relationship

E.g., STUDENT (Stud#, Sname, Major) is a relation that stands for real-world entities called students, whereas,

ENROLL (Stud#, Course#, Grade) is also a relation – but it stands for the relationship among two entity types STUDENT and COURSE.

Later formalized by Peter Chen in the ER model (1976).



Fundamental Concepts - 2

- The notion of "Symmetric exploitation of relations"- by this he meant that out of the number of attributes in the relation, you could supply any subset and retrieve the remaining set.

E.g., Given a postal code, find all employees (their name, age, salary, year_of_joining) or given a salary value, find all employees who earn more than that salary.

Today's query languages and search engines support this.

- He predicts that while first order predicates will suffice to deal with most relations, there is a need for second order predicates in case the domain (or attribute) is itself a relation.

The languages like SQL have stayed within the first order predicate calculus for simplicity. Second order predicates were employed in Non-normalized experimental systems (e.g., IBM Hedelberg work, Ph.D. work of Anand Deshpande, Roth etc.) only and may appear in future in non_SQL embedded data environments.

Fundamental Concepts - 3

- Codd defined the initial operations of Relational Algebra in his 1969-70 papers. They included:

Permutation

Projection

Join and

Composition

- He defined a naming scheme for attributes that looked like-

R(g).r.d

Where,

R is a relation name

g is the generation of a relation (not followed in today's systems)

r is a role name (roles are more common in security context)

d is the domain name

Fundamental Concepts - 4

- He defined the notions of

Expressible relations: these are the **"views"** of data

Named relations: this is what gets defined in the **schema** with what he calls "public names"

Stored relations: this is what is stored in the **base tables** as instances of actual data

- He defined the notion of derivability:

The result of a query is a derivable relation which is derived by applying a sequence of operations to a set S of relations.

The result of a query (in most cases) is a derived relation and hence we can query that relation further with another query.

Fundamental Concepts - 5

- Strongly Redundant:

A set of relations is strongly redundant if it contains at least one relation that is derivable from the rest of the members.

- Weakly Redundant:

A set of relations is weakly redundant if it contains at least one relation that is not derivable from other members of the set but is at all times a projection of some join of other members of the set

In general, we use the above ideas during design to make sure that we avoid strong redundancy at any cost. But weak redundancy is tolerated and allowed in light of practical business reporting and similar requirements.

Fundamental Concepts - 6

- Consistency and Inconsistency

Given a set S of relations and an associated set C of constraints, the set S is consistent or inconsistent according to whether or not it complies with those constraints (which are nothing but some form of stated redundancies.)

e.g. If a student is enrolled in Course# 6400 he must have previously been enrolled in course #3110.

These notions were never formalized in DBMS's before. Today we call constraints as "business logic or rules " and make sure that data values remain consistent with the business rules or constraints.

Fundamental Concepts - 7

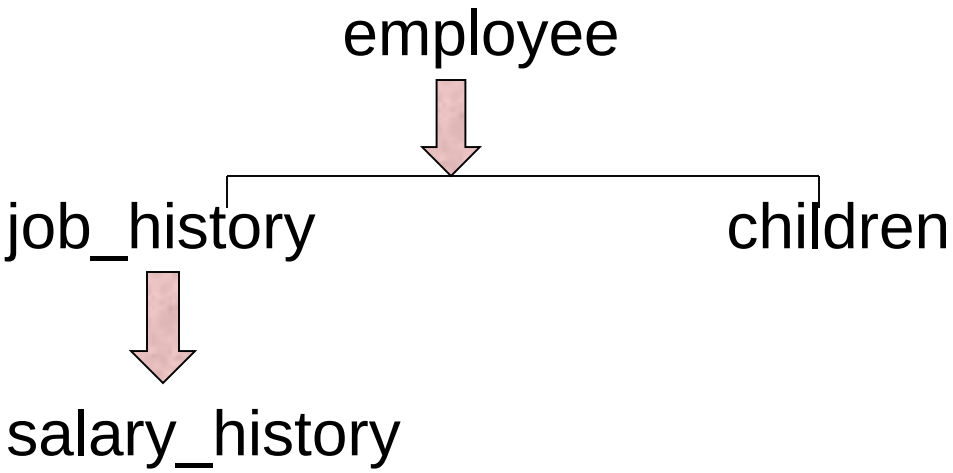
- Besides the above concepts, Codd also defined the ideas of:

Primary Key

Foreign Key

Normal form (based on proper propagation of keys in a hierarchy):

Example from (Codd, 1970)



Fundamental Concepts - 8

- Unnormalized relations:

Employee(Emp_id, name, DoB, jobhistory, children)

Job_history(jobdate, job_title, salary_history)

Salary_history (salarydate, salary)

Children(childname, birthyear)

- Normalized relations:

Employee(Emp_id, name, DoB, jobhistory, children)

Job_history(Emp_id, jobdate, job_title)

Salary_history (Emp_id, jobdate, salarydate, salary)

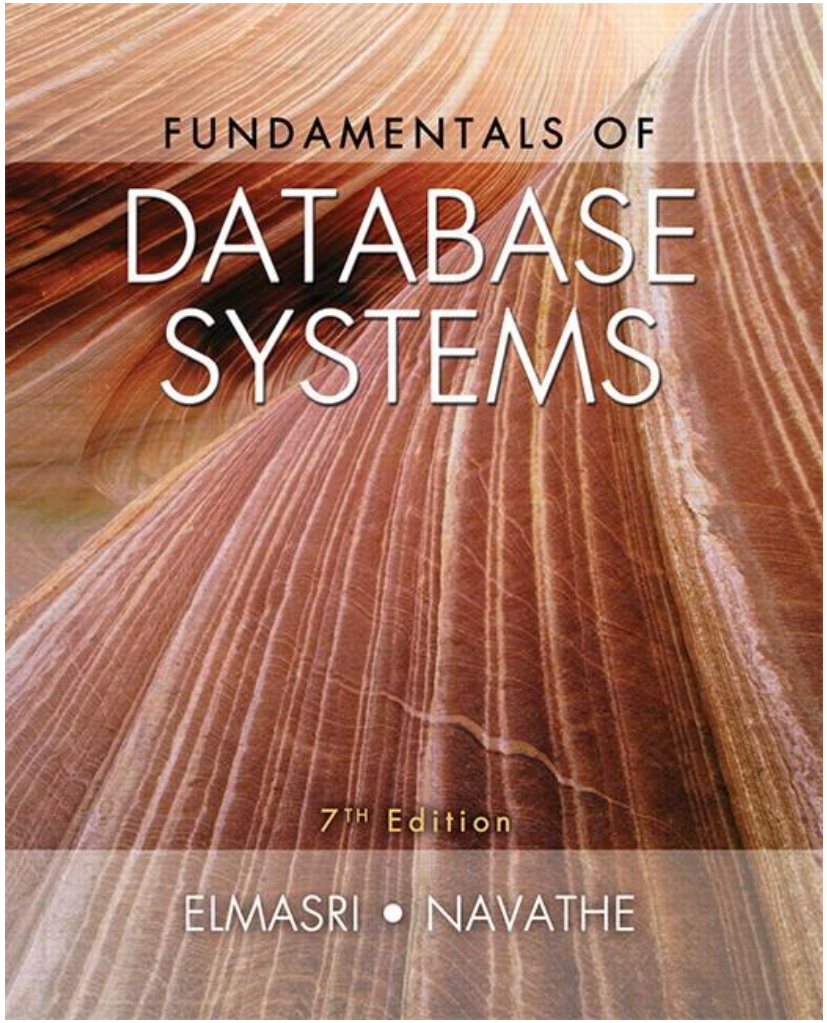
Children(Emp_id, childname, birthyear)

We use this technique to represent any embedded data in relational systems today.

RELATIONAL MODEL – A QUICK TUTORIAL

Chapter 5, “Fundamentals of Database Systems,” by Elmasri and Navathe, Addison Wesley, Ed.7, 2016

The Relational Data Model and Relational Database Constraints



Informal Definitions

- Informally, a **relation** looks like a **table** of values.
- A relation typically contains a **set of rows**.
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
  - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
  - In the formal model, the column header is called an **attribute name** (or just **attribute**)

Informal Definitions

- Key of a Relation:
  - Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
    - Called the *key*
  - In the STUDENT table, SSN is the key
  - Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table
    - Called *artificial key* or *surrogate key*

Formal Definitions - Schema

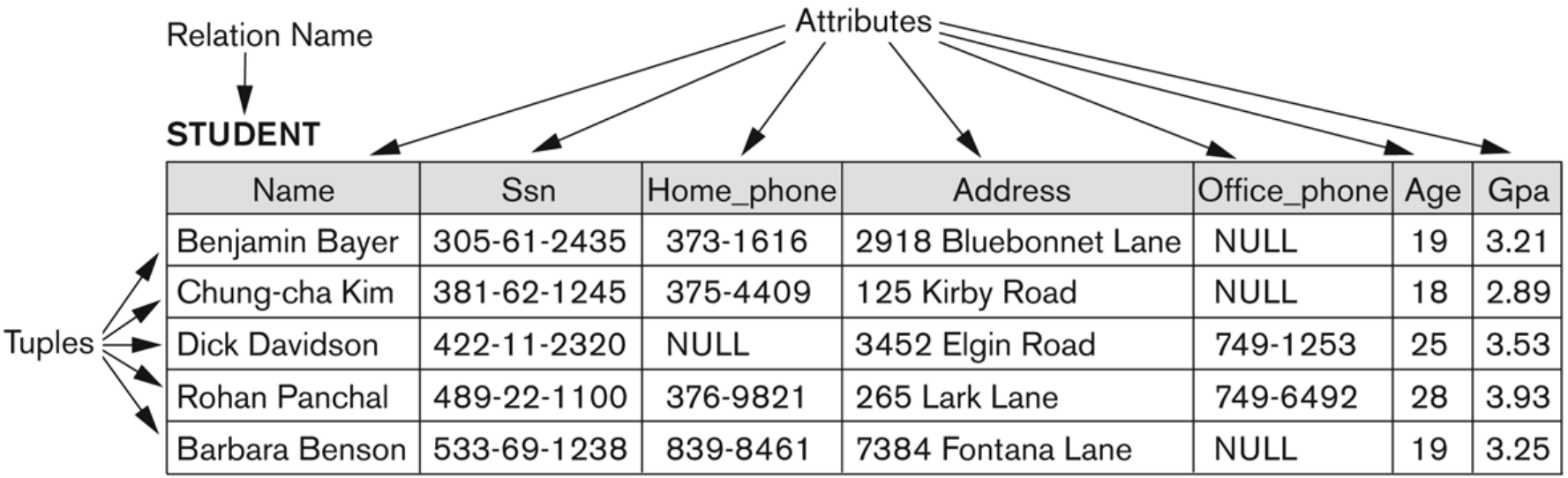
- The **Schema** (or description) of a Relation:
  - Denoted by R(A1, A2, ...An)
  - R is the **name** of the relation
  - The **attributes** of the relation are A1, A2, ..., An
- Example:

  CUSTOMER (Cust-id, Cust-name, Address, Phone#)
  - CUSTOMER is the relation name
  - Defined over the four attributes: Cust-id, Cust-name, Address, Phone#
- Each attribute has a **domain** or a set of valid values.
  - For example, the domain of Cust-id is 6 digit numbers.

Definition Summary

Informal Terms		Formal Terms
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Example – A relation STUDENT



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Characteristics Of Relations

- Ordering of tuples in a relation r(R):
  - The tuples are *not considered to be ordered*, even though they appear to be in the tabular form.
- Ordering of attributes in a relation schema R (and of values within each tuple):
  - We will consider the attributes in R(A1, A2, ..., An) and the values in t=<v1, v2, ..., vn> to be ordered .
    - (However, a more general alternative definition of relation does not require this ordering).

Characteristics Of Relations

- Values in a tuple:
  - All values are considered atomic (indivisible).
  - Each value in a tuple must be from the domain of the attribute for that column
    - If tuple t = <v1, v2, …, vn> is a tuple (row) in the relation state r of R(A1, A2, …, An)
    - Then each *vi* must be a value from *dom(Ai)*
  - A special **null** value is used to represent values that are unknown or inapplicable to certain tuples.

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- There are three *main types* of constraints in the relational model:
  - **Key** constraints
  - **Entity integrity** constraints
  - **Referential integrity** constraints
- Another implicit constraint is the **domain** constraint
  - Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

Key Constraints

- **Superkey** of R:
  - Is a set of attributes SK of R with the following condition:
    - No two tuples in any valid relation state r(R) will have the same value for SK
    - That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK]
    - This condition must hold in *any valid state* r(R)
- **Key** of R:
  - A "minimal" superkey
  - That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)

Key Constraints (continued)

- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - CAR has two keys:
    - Key1 = {State, Reg#}
    - Key2 = {SerialNo}
  - Both are also superkeys of CAR
  - {SerialNo, Make} is a superkey but *not* a key.
- In general:
  - Any *key* is a *superkey* (but not vice versa)
  - Any set of attributes that *includes a key* is a *superkey*
  - A *minimal* superkey is also a key

Key Constraints (continued)

- If a relation has several **candidate keys**, one is chosen arbitrarily to be the **primary key**.
  - The primary key attributes are underlined.
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - We chose SerialNo as the primary key
- The primary key value is used to *uniquely identify* each tuple in a relation
  - Provides the tuple identity
- Also used to *reference* the tuple from another tuple
  - General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
  - Not always applicable – choice is sometimes subjective

COMPANY Database Schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

Entity Integrity

- **Entity Integrity:**
  - The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of r(R).
    - This is because primary key values are used to *identify* the individual tuples.
    - t[PK] $\neq$ null for any tuple t in r(R)
    - If PK has several attributes, null is not allowed in any of these attributes
  - Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity

- A constraint involving **two** relations
  - The previous constraints involve a single relation.
- Used to specify a **relationship** among tuples in two relations:
  - The **referencing relation** and the **referenced relation**.

Referential Integrity

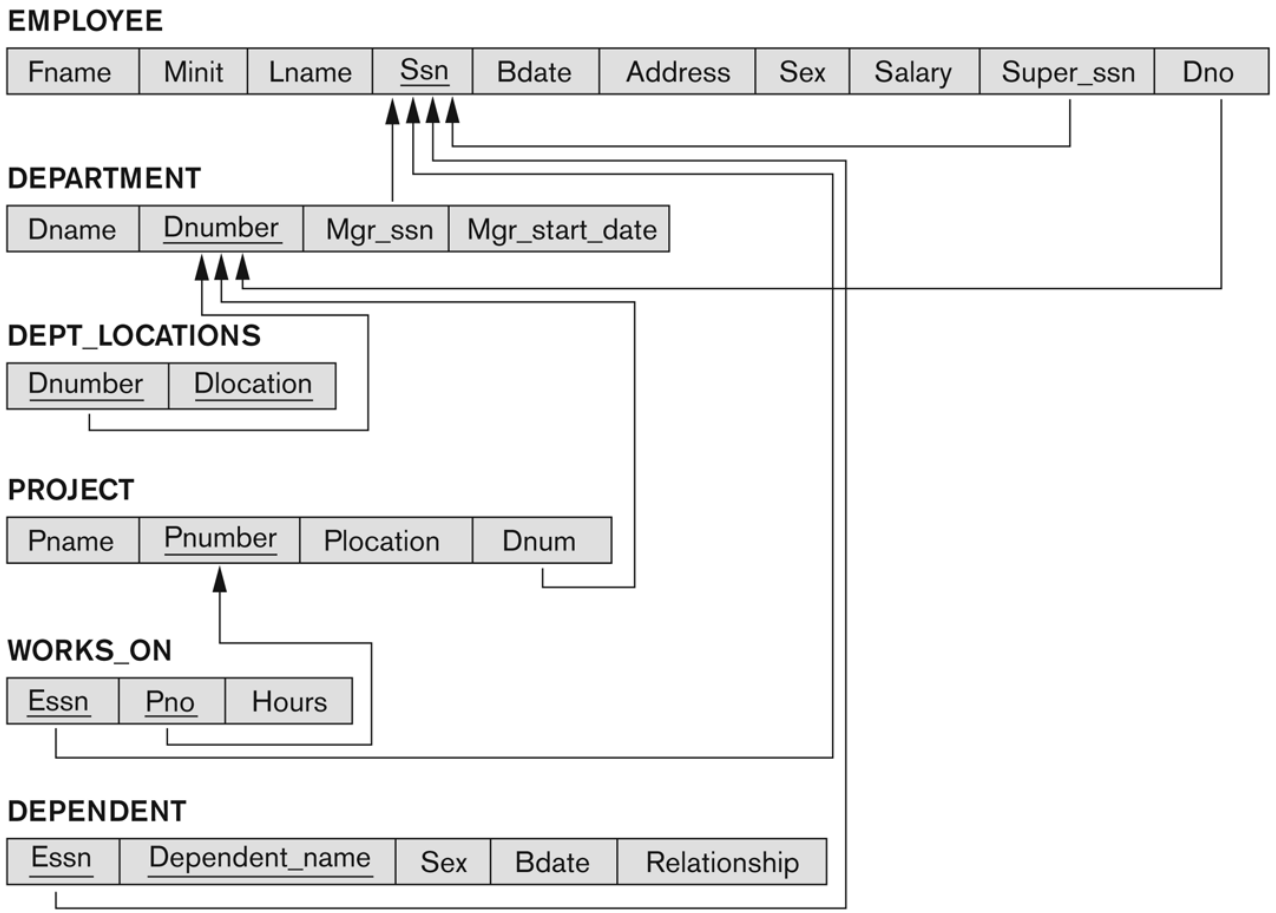
- Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R2.
  - A tuple t1 in R1 is said to **reference** a tuple t2 in R2 if t1[FK] = t2[PK].
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

Referential Integrity (or foreign key) Constraint

- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the the **referencing relation** R1 can be **either**:
    - (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, or
    - (2) a **null**.
- In case (2), the FK in R1 should **not** be a part of its own primary key.

Referential Integrity Constraints for COMPANY database

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



Other Types of Constraints

- Semantic Integrity Constraints:
  - based on application semantics and cannot be expressed by the model per se
  - Example: “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
- A **constraint specification** language may have to be used to express these
- SQL-99 allows triggers and **ASSERTIONS** to express for some of these

Populated database state

- Each *relation* will have many tuples in its current relation state
- The *relational database state* is a union of all the individual relation states
- Whenever the database is changed, a new state arises
- Basic operations for changing the database:
  - INSERT a new tuple in a relation
  - DELETE an existing tuple from a relation
  - MODIFY an attribute of an existing tuple
- Next slide shows an example state for the COMPANY database

Populated database state for COMPANY

Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Update Operations on Relations

- INSERT a tuple.
- DELETE a tuple.
- MODIFY a tuple.
- Integrity constraints should not be violated by the update operations.
- Several update operations may have to be grouped together.
- Updates may **propagate** to cause other updates automatically. This may be necessary to maintain integrity constraints.

Update Operations on Relations

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (RESTRICT or REJECT option)
  - Perform the operation but inform the user of the violation
  - Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
  - Execute a user-specified error-correction routine

Relational Algebra Operations

Table 6.1
Operations of Relational Algebra

Operation	Purpose	Notation
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{<\text{selection condition}>}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{<\text{attribute list}>}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{<\text{join condition}>} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{<\text{join condition}>} R_2$, OR $R_1 \bowtie_{(<\text{join attributes 1}>), (<\text{join attributes 2}>)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{<\text{join condition}>} R_2$, OR $R_1 *_{(<\text{join attributes 1}>), (<\text{join attributes 2}>)} R_2$ OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Relational Calculus

- Defines a language based on first order predicate calculus
- Relates relational data to facts or predicates
- Two branches:

Tuple calculus (defined by Codd and others):

- Later led to SQL

Domain calculus (defined by others):

- Was based on QBE

EXTENSIONS TO THE RELATIONAL MODEL

The Extended RM/T Model (1979)

- T stands for Tasmania where Codd first presented this.
- This was Codd's response to the semantic models developed after the relational model (e.g., work of Smith and Smith, 1977, Hammer and Mcleod 1978 etc.)
- Develops a theoretical framework that addresses metadata and a detailed construction of a catalog called the RM/T catalog.

RM/T – contd.

- Defines various types of relations and graphs:
- E-relation
- P-relation
- Characteristic relation
- Characteristic graph relation
- Property graph relation
- Entity Association
- Non-entity Association
- Generalization , etc.

Codd's Relational Model – The 12 rules (1985)

Published in a COMPUTERWORD (a popular business weekly) article in October 1985.

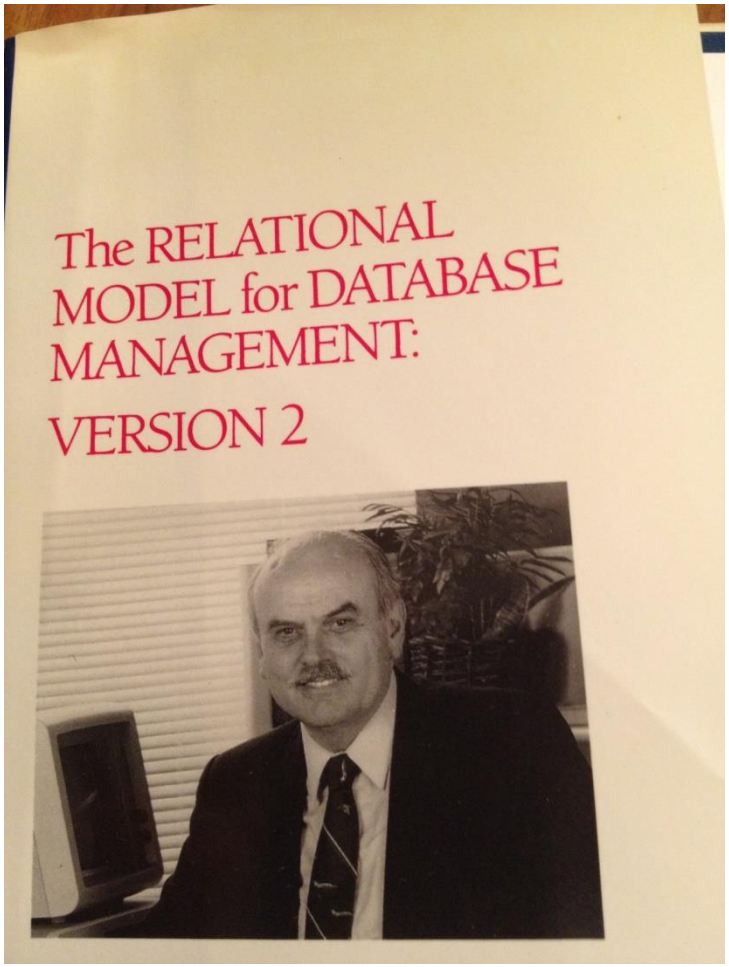
Rule 0: The *foundation rule*:
For any system that is advertised as, or claimed to be, a relational data base management system that system must be able
to manage data bases entirely through its relational capabilities.

Codd's Relational Model – The 12 rules (1985)

Rule 1: Information Rule
Rule 2: The Guaranteed Access Rule
Rule 3: Systematic Treatment of Null Values
Rule 4: Dynamic Online Catalog
Rule 5: Comprehensive Data Sublanguage
Rule 6: The View Updating Rule
Rule 7: High Level Insert , Update, Delete
Rule 8: Physical Data Independence
Rule 9: Logical Data Independence
Rule 10: Integrity Independence
Rule 11: Distribution Independence
Rule 12; The non-subversion rule.

Codd's Relational Model – Version 2 (1990)

- He published 12 rules in Computerworld (14 Oct. 1985) about "what makes a DBMS relational"
- According to these rules none of the products on the market qualified as relational DBMSs.
- In the book he prescribes 333 rules



Codd's Relational Model – Version 2 (1990)-contd.

- Three goals for this book:
- To simplify interaction of users with databases
- To increase productivity of professionals
- For DBA's: to support tools for access and integrity control
- Proposes a need for an "Abstract Machine Standard" for database management
- Book has 30 chapters (538 pages) and includes chapters on user-defined data types, authorization, scalar and aggregate functions, language design principles, distributed database management

CODD'S TURING AWARD LECTURE - 1981

The prevalent system environment (circa 1981)

- Popular hardware included IBM System 370, 3090 etc.- mainframes with CICS
- Popular languages: COBOL, PL/1
- PC was yet to arrive (arrived in 1983/84)
- IBM making major revenues from disk storage systems
- No email, no distributed data, NO WWW!!!

Codd's 1981 Turing Award Lecture

Title: "Relational Database : A Practical Foundation for Productivity"

- Mentions the demands of new applications as overwhelming for the data processing (DP) departments; suggests:

1) placing users directly in touch with the stored information

2) increase the productivity of DP professionals in developing application programs

- Expresses concern that the label "relational" is being used in misleading ways by systems and they should really be called a "relational processing capability".

Codd (1981) -2

- Mentions 3 pitfalls:

1) Navigation dependency in existing databases
2) No set-oriented processing
3) Need for end users to have direct interaction with databases

- Revises his original objectives in the Codd (1970) paper and states 3 objectives:

1) Data independence objective
2) Communicability objective
3) Set-processing objective

Codd (1981) - 3

- Mentions the symbolic addressibilty of data in relational systems:

Just like in a 2-d array we address an element as X(i, j), we can address any element here by

(relation name, primary key value, attribute name)

- Contrasts relational model with “tabular model”:

- Relational model does not need any “linked data structure” to represent a relationship among 2 objects
- Tables are at a lower level of abstraction and have array-like “positional addressing”

Codd (1981) - 4

- "In the development of the Relational model there has been a strong coupling between structural, manipulative and integrity aspects."

Implies that it is a "complete" data model.

- " The set processing objective of the relational model is intended to be met by means of a data sublanguage having at least the power of relational algebra *without making use of iteration and recursion statements*.

Relational tuple calculus was such a language which gave rise to SEQUEL (or SQL). Domain calculus also fits the requirement and gave rise to QBE.

Codd (1981) - 5

- Distinguishes “relational DBMSs” from systems with “relational processing capability”.

By his definition no database system is truly relational.

- A system has the relational processing capability if it provides a data sublanguage L that can specify the transformations specified by SELECT, PROJECT and JOIN operators.
- Two No – No’s of relational systems:

1) No user-visible navigation links between tables
2) No information should be conveyed in the way tuples are ordered.

Codd (1981) - 6

- He termed Relational Systems that support a "double mode capability" (1- interactive from a terminal, 2- embedded in an application program) as "uniformly relational".
- Expresses Skepticism, an interesting point:

1) "can a relational system provide services we have grown to expect from a DBMS"? (mentions a range of services)

2) "if yes, then can such a system perform as well as a non-relational system?"

The latter will be a discussion point in light of the "emerging database technologies".

Codd (1981) - 7

- Performance of relational system: based on-

Performance oriented data structures

Efficient and optimized compilation of user requests into code

Systems continue to develop new index structures and work on query optimization is never ending

“Must generate code comparable to average hand-written code in simple cases and superior in complex cases”.

Codd (1981) – 8- Future Directions

In Future Directions in his Turing Award talk, he includes:

Stronger support for domains and primary keys

Better support for updating join-type views

Support for outer-joins

Catalogs

Design aids at both logical and physical levels

Location and Replication transparency in distributed databases

Need to capture more meaning of data

Improved treatment of missing, null and inapplicable values

Heterogeneous data as well as new data types (e.g., images and text)

In the last 36 years since the above talk was given, substantial work has taken place in both the academic, research labs and industrial development groups.

My Concluding Thoughts

- The relational model has laid a very firm foundation for database management with no comparable (competing) development for last 47 years!
- Relational model was a landmark development that provided a basis for formulating a variety of data management problems related to design, derivability, redundancy, consistency, replication as well as language issues.
- It provides a very well defined data model with the three strong foundational pillars of data modeling
    - structure, operations and constraints

Concluding thoughts – contd.

- His ideas of an abstract machine standard for data management are valid and recent developments in Non-SQL as well as Map-reduce/Hadoop type of processing need to keep in mind that we cannot let data management proceed in an undisciplined ad hoc manner.

On the side of shortcomings:

- His RM/T model (relational model- version 2) provided a detailed analysis of metadata and defined catalog relations. However, that work had little impact on the practice of metadata management.

- Relational model inherently loses the distinction between what is an entity and what is a relationship and therefore the role of conceptual and semantic models for better understanding of application semantics will remain. Relational model still falls short on modeling semantics of data.

THANKS