1. The Relational Model

Stéphane Bressan

January 22, 2015
This lecture is based on material by Professor Ling Tok Wang.
Content

1. Introduction
   - Introduction

2. Codd’s Motivation
   - Readings
   - Codd’s Motivation

3. The Relational Model
   - Definitions

4. The Universal Relation
   - Motivation
   - Readings
   - Definition
   - The Universal Relation as a User Interface

5. Design Anomalies
   - Motivating Example
“Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). ”

_A Relational Model of Data for Large Shared Data banks_ [CACM 1970], by Edgar F. Codd
Example

The database of a manufacturing company contains information about parts and projects. For each part, the part number, part name, part description, quantity-on-hand, and quantity-on-order are recorded. For each project, the project number, project name and project description are recorded. Whenever a project makes use of a certain part, the quantity of that part committed to the given project is also recorded.
Many different ways of representing in the hierarchical model.
The Relational Design Question

How many tables? What tables? How many columns in each table? What columns?

But Also

What Integrity Constraints?

Integrity Constraints in SQL

- PRIMARY KEY
- UNIQUE
- NOT NULL
- FOREIGN KEY
- CHECK
### Part

<table>
<thead>
<tr>
<th>Part#</th>
<th>PartName</th>
<th>Desc</th>
<th>QuantityHand</th>
<th>QuantityOrder</th>
</tr>
</thead>
<tbody>
<tr>
<td>35212</td>
<td>nut</td>
<td>FISH HEAD BOLT</td>
<td>10000</td>
<td>560</td>
</tr>
<tr>
<td>23212</td>
<td>bolt</td>
<td>CAGE NUT</td>
<td>24366</td>
<td>123</td>
</tr>
<tr>
<td>6653</td>
<td>screw</td>
<td>Pan Head Screw</td>
<td>123</td>
<td>5000</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Project

<table>
<thead>
<tr>
<th>Project#</th>
<th>ProjectName</th>
<th>Desc</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Bicycle</td>
<td>Build a bicycle with side car</td>
</tr>
<tr>
<td>203</td>
<td>Electric Car</td>
<td>Build an electric car that runs on solar power</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Commit

<table>
<thead>
<tr>
<th>Project#</th>
<th>Part#</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>35212</td>
<td>203</td>
<td>500</td>
</tr>
<tr>
<td>23212</td>
<td>101</td>
<td>232</td>
</tr>
<tr>
<td>6653</td>
<td>101</td>
<td>65</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Readings

Motivation

Codd motivates the relational model by the inadequacy of hierarchical and network models with respect to:

- the lack of data independence;
- and the poor management of data inconsistencies.

Data Independence

- Ordering dependence;
- Indexing dependence;
- Access Path Dependence.

Data Consistency

- Structural constraint;
- Logical constraint.
### Codd’s Motivation

<table>
<thead>
<tr>
<th>Part#</th>
<th>PartName</th>
<th>Desc</th>
<th>QuantityHand</th>
<th>QuantityOrder</th>
</tr>
</thead>
<tbody>
<tr>
<td>35212</td>
<td>nut</td>
<td>FISH HEAD BOLT</td>
<td>10000</td>
<td>560</td>
</tr>
<tr>
<td>23212</td>
<td>bolt</td>
<td>CAGE NUT</td>
<td>24366</td>
<td>123</td>
</tr>
<tr>
<td>6653</td>
<td>screw</td>
<td>Pan Head Screw</td>
<td>123</td>
<td>5000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project#</th>
<th>ProjectName</th>
<th>Desc</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Bicyle</td>
<td>Build a bicycle with side car</td>
</tr>
<tr>
<td>203</td>
<td>Electric Car</td>
<td>Build an electric car that runs on solar power</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project#</th>
<th>Part#</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>35212</td>
<td>203</td>
<td>500</td>
</tr>
<tr>
<td>23212</td>
<td>101</td>
<td>232</td>
</tr>
<tr>
<td>6653</td>
<td>101</td>
<td>65</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Student#</td>
<td>Course#</td>
<td>S-name</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>95001</td>
<td>CS1101</td>
<td>Tan CK</td>
</tr>
<tr>
<td>95023</td>
<td>CS1101</td>
<td>Lee SL</td>
</tr>
<tr>
<td>95023</td>
<td>CS2103</td>
<td>Tan CK</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Let us consider

- the countably infinite set \( \mathcal{R} \) is a set of relations (relation names),
- the countably infinite set \( \mathcal{A} \) is a set of attributes (attribute names) such that \( \mathcal{R} \cap \mathcal{A} = \emptyset \), and,
- the set \( \mathcal{D} \) is the domain (set of atomic values).
- If attributes need different domains the function \( \text{Dom} \) on \( \mathcal{A} \) (\( \text{Dom} : \mathcal{A} \rightarrow 2^\mathcal{D} \)) defines the domain of an attribute \( A \in \mathcal{A} \): \( \text{Dom}(A) \subset \text{dom} \).

\[
\text{TAKE} \in \mathcal{R} \\
\{ \text{Student}\#, \text{Course}\#, \text{S-name}, \text{C-desc}, \text{Mark} \} \subset \mathcal{A}
\]
**Definition**

The structure of a table is given by the relation name and a finite set of attributes. We assume that there exists a function from the set of relation names to the set of finite subsets of attribute names.

\[
\text{sort} : \mathcal{R} \mapsto 2_{\text{finite}}^A
\]

\(\text{sort}(R)\) is the schema of the relation. We write \(R = \text{sort}(R)\).

\[
\text{sort}(\text{TAKE}) = \{\text{Student}\#, \text{Course}\#, \text{S-name}, \text{C-desc}, \text{Mark}\}
\]

\(\text{TAKE} = \{\text{Student}\#, \text{Course}\#, \text{S-name}, \text{C-desc}, \text{Mark}\}\)
Definition

The arity (or degree) of a relation name, $R$, is the number of its attributes.

$$\text{arity}(R) = |\text{sort}(R)|$$

TAKE is quinary.

$$\text{arity}(\text{TAKE}) = 5$$

degree 0 = nullary, degree 1 = unary, degree 2 = binary, degree 3 = ternary ...

Do we have degree 0 relations in SQL?
Original Definition by Codd

The tabular representation of relations is a convenient (visualization), practical (implementation) but not essential part (design and query) of the relational model.

1. Each row represents an n-tuple of $R$.
2. The ordering of rows is immaterial.
3. All rows are distinct.
4. The ordering of columns is significant—it corresponds to the ordering $S_1, S_2, \ldots, S_n$ of the domains on which $R$ is defined.
5. The significance of each column is partially conveyed by labeling it with the name of the corresponding domain.
Original Definition by Codd

Given the non-necessarily distinct sets of atomic (i.e. non-decomposable) elements $S_1, S_2, \cdots, S_n$, a first normal form (1NF) relation $R$ on these sets if it is a subset of the Cartesian (cross) product of these sets.

$$R \subset S_1 \times S_2 \times \cdots \times S_n$$

Domain

We refer to $S_i$ as the $n^{th}$ domain.
T-uples

$R$ is a set of (ordered) $n$-tuples (t-uples)

\[
< e_1, e_2, \ldots, e_n > \in S_1 \times S_2 \times \cdots \times S_n
\]

T-uples Constructor

$R$ is a set of (ordered) $n$-tuples (t-uples) \(< . >\) is the t-uple constructor symbol.
Duplicates

A set does not contain duplicate elements:

$$\{ a, a, b \} = \{ a, b \}$$

In the definition we gave the relation instances differ from the tables as they do not contain duplicate elements.
Order

A set is not ordered:

\[ \{a, b\} = \{b, a\} \]

The order of t-uples in relations and rows in tables is irrelevant.
Order

Components of a t-uple in the unnamed view are ordered.

< Andrew, Jackson >

Andrew is the first name and Jackson is the family name.
Codd’s Definition: Unnamed View

Under the unnamed view a tuple is an element of the Cartesian product of the domain(s).

\[ t_1 \in dom \times dom \times dom \times dom \times dom \]

\[ t_1 = \langle 95001, CS1101, TanCK, Programming, 75 \rangle \]

Named View

Under the named view a tuple is a function mapping an attribute to a value in the domain of the attribute.

\[ t_1 : R \mapsto dom \]

\[ t_1(Student\#) = 95001 \]
**Conventional View**

Under the conventional view a relation instance of a relation schema $R[U]$ (over the attributes $U$) is a finite set $I(R)$ of tuples.

**Logic Programming View**

Under the logic programming view a relation instance of a relation schema $R[U]$ (over the attributes $U$) is a finite set of facts over $R$. 
A database.

\[
\begin{array}{|c|c|}
\hline
R & \hline
A & B \\
\hline
a & b \\
\hline
c & d \\
\hline
a & a \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
S & \hline
C & d \\
\hline
\end{array}
\]
Query

Find the A-value in R such that the corresponding B-value in R is a C-value in S.
Unnamed and Conventional (Codd’s View)

\[ I(R) = \{ < a, b >, < c, d >, < a, a > \} \]
\[ I(S) = \{ < d > \} \]
Introduction

Codd’s Motivation

The Relational Model

The Universal Relation

Design Anomalies

Definitions

Domain Relational Calculus

\[ \{ < X > \mid \exists Y ( < X, Y > \in R \land < Y > \in S) \} \]
Named and Conventional

\[ I(R) = \{ f_1, f_2, f_3 \} \]
\[ f_1(A) = a, f_2(A) = c, f_3(A) = a, f_1(B) = b, f_2(B) = d, f_3(B) = a \]
\[ I(S) = \{ g_1 \} \]
\[ g_1(C) = d \]
SQL

```
SELECT R.A FROM R, S WHERE R.B = S.C
```
Domain Relational Calculus

$$\pi_{R.A}(\sigma_{R.B=S.C}(R \times S))$$

$$\pi_{R.A}(R \Join_{R.B=S.C} S)$$

SQL

SELECT R.A FROM R, S WHERE R.B = S.C

SELECT R.A FROM R INNER JOIN S ON R.B = S.C
Named and Logic Programming

\[ I = \{ I(t_1 \in R) = true, I(t_2 \in R) = true, I(t_3 \in R) = true, \\
I(t_1.A = a) = true, I(t_1.B = b) = true, \\
I(t_2.A = c) = true, I(t_2.B = d) = true, \\
I(t_3.A = a) = true, I(t_3.B = a) = true, \\
I(t_4 \in S) = true, I(t_4.C = d) = true, \ldots \} \]

\[ ^a \text{The rest is false.} \]
Domain T-uple Calculus

\[
\{ T \mid \exists T_1 \exists T_2 \ (T_1 \in R \land T_2 \in S \land T_1.B = T_2.B \land T.A = T_1.A) \}
\]

\[
\{ < T_1.A > \mid \exists T_1 \exists T_2 \ (T_1 \in R \land T_2 \in S \land T_1.B = T_2.B) \}
\]

\[
\{ < T_1.A > \mid \exists T_1 \in R \exists T_2 \in S \ (T_1.B = T_2.B) \}
\]

SQL

\[
SELECT T_1.A FROM R T_1, S T_2 WHERE T_1.B = T_2.C
\]
Unnamed and Logic Programming

\[ I = \{ I(R(a, b)) = true, I(R(c, d)) = true, \\
i(R(a, a)) = true, I(S(d)) = true, \ldots \} \]

\(^a\) The rest is false.
Domain Relational Calculus

\{ <X> \mid \exists Y \ (R(X, Y) \land S(Y)) \}\}

Datalog

\[ Q(X) \leftarrow R(X, Y), S(Y). \]
\[ \leftarrow Q(X). \]
Definition

A database schema is a non-empty finite set $\mathcal{R}$ of relation and constraints on these relations.

The Design Question

How many tables? What tables? How many columns in each table? What columns? What Integrity Constraints?
The Universal Relation

Do we need more than one relation?
The Universal Relation

All the data is kept in a single relation whose scheme consists of all attributes. If necessary, null values are used to pad out \( t \)-uples.

<table>
<thead>
<tr>
<th>C</th>
<th>T</th>
<th>H</th>
<th>R</th>
<th>S</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>M9</td>
<td>222</td>
<td>Weenie</td>
<td>B+</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>W9</td>
<td>333</td>
<td>Weenie</td>
<td>B+</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>F9</td>
<td>222</td>
<td>Weenie</td>
<td>B+</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>M9</td>
<td>222</td>
<td>Grind</td>
<td>C</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>W9</td>
<td>333</td>
<td>Grind</td>
<td>C</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>F9</td>
<td>222</td>
<td>Grind</td>
<td>C</td>
</tr>
</tbody>
</table>
### Readings

Find the rooms in which Prof Deawood is teaching.

In the language of Stanford System/U:

```
RETRIEVE R WHERE T = 'Deadwood'
```


<table>
<thead>
<tr>
<th>C</th>
<th>T</th>
<th>H</th>
<th>R</th>
<th>S</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>M9</td>
<td>222</td>
<td>Weenie</td>
<td>B+</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>W9</td>
<td>333</td>
<td>Weenie</td>
<td>B+</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>F9</td>
<td>222</td>
<td>Weenie</td>
<td>B+</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>M9</td>
<td>222</td>
<td>Grind</td>
<td>C</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>W9</td>
<td>333</td>
<td>Grind</td>
<td>C</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>F9</td>
<td>222</td>
<td>Grind</td>
<td>C</td>
</tr>
</tbody>
</table>

RETRIEVE t1.R WHERE t1.R=t2.R AND t2.C='CS101'

Find the modules using a room used by CS101.
The following database has the following three relations.

\[ \text{Supplier}(\text{code}, \text{sname}) \]

\[ \text{Part}(\text{code}, \text{pname}, \text{color}) \]

\[ \text{Supply}(\text{supplier}, \text{part}, \text{price}) \]
**Universal Relation Assumption: Same Name**

Two attributes with the same name correspond to the same attribute in the universal relation, i.e. they are from the same attribute and of the same semantics (same meaning).

**Universal Relation Assumption: Different Names**

Two attributes with different names from two different relations or from one relation correspond to two different attributes in the universal relation, and have different semantics.
The following database has the following three relations.

\[
\text{Supplier}(\text{code}, \text{sname})
\]

\[
\text{Part}(\text{code}, \text{pname}, \text{color})
\]

\[
\text{Supply}(\text{supplier}, \text{part}, \text{price})
\]

The database does not satisfy the universal relation assumptions (Why? Bad design of attribute names).
### Codd's Motivation

The Relational Model

#### The Universal Relation

- **The Universal Relation as a User Interface**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>T</td>
<td>H</td>
<td>R</td>
<td>S</td>
<td>G</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>M9</td>
<td>222</td>
<td>Weenie</td>
<td>B+</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>W9</td>
<td>333</td>
<td>Weenie</td>
<td>B+</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>F9</td>
<td>222</td>
<td>Weenie</td>
<td>B+</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>M9</td>
<td>222</td>
<td>Grind</td>
<td>C</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>W9</td>
<td>333</td>
<td>Grind</td>
<td>C</td>
</tr>
<tr>
<td>CS101</td>
<td>Deawood</td>
<td>F9</td>
<td>222</td>
<td>Grind</td>
<td>C</td>
</tr>
</tbody>
</table>

with the underlying scheme \(\{CT, CHR, CSG\}\),
**Window**

The window $[X]$ is a relation with scheme $X$, where $X$ is the set of attributes mentioned in the query.

**Example**

The window has schema $(R, T)$.

```
RETRIEVE R WHERE T = 'Deadwood'
```
Window Function

The window function defines the window from the actual database.

Example

The window function is, for example, the natural join of the minimal set of relations whose scheme includes all the attributes.

with the underlying scheme \{CT, CHR, CSG\},

\[ [X] = \pi_{R,T}(CT \bowtie CHR) \]
## Design Anomalies

### Motivating Example

<table>
<thead>
<tr>
<th>Student#</th>
<th>Course#</th>
<th>S-name</th>
<th>C-desc</th>
<th>Mark</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>95001</td>
<td>CS1101</td>
<td>Tan CK</td>
<td>Programming</td>
<td>75</td>
<td>The art of Programming</td>
</tr>
<tr>
<td>95023</td>
<td>CS1101</td>
<td>Lee SL</td>
<td>Programming</td>
<td>58</td>
<td>The art of Programming</td>
</tr>
<tr>
<td>95023</td>
<td>CS2103</td>
<td>Tan CK</td>
<td>D.S. and Alg.</td>
<td>64</td>
<td>The art of Programming</td>
</tr>
<tr>
<td>95001</td>
<td>CS1101</td>
<td>Tan CK</td>
<td>Programming</td>
<td>75</td>
<td>Java</td>
</tr>
<tr>
<td>95023</td>
<td>CS1101</td>
<td>Lee SL</td>
<td>Programming</td>
<td>58</td>
<td>Java</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Redundant

Exceeding what is necessary or natural; superfluous.

Lecturers and texts are repeated for each student and course.

Anomaly

An inconsistency.

If a new course is created but no students have taken this course, then we cannot enter the information about this course because the use of null values or undefined values in the primary key could cause problem.
Motivating Example

- Redundant storage
- insertion/deletion anomaly
- update anomaly

What causes the anomalies?
One process which attempts to remove these undesirable updating anomalies from the relation is called normalization.

\[
R1(\underline{STUDENT}\#, \text{S-NAME}) \\
R2(\underline{COURSE}\#, \text{C-DESCRIPTION}) \\
R3(\underline{STUDENT}\#, \underline{COURSE}\#, \text{MARK}) \\
R4(\underline{COURSE}\#, \text{Text})
\]

These relations do not have the discussed anomalies.

Underlined attributes indicate a key of the relation. e.g., attributes \textit{STUDENT\#} and \textit{COURSE\#}, together, form a key of the relation R3.