SWEN-220
Mathematical Models of Software

Relational Models
Introduction
E.F. Codd – The Relational Model

Topics

• Relational Model
  – History & Overview

• Features of the original relational model
  – Structural
  – Integrity
  – Manipulation
Why the relational model?

• When you think of managing data, one of the first things that comes to mind is a relational database.

• The relational model is not product-specific (as in commercial databases), it is concerned with principles.
Why the relational model?

• If you know Oracle for example, your knowledge may not transfer to DB2 or SQL server.

• But if you know the principles, then your knowledge will be transferable, i.e., knowledge that will apply in every environment
The Relational Model
Brief History of the Relational Model & Databases

• **1960s:** Computerized database started in the 1960s, when the use of computers became a more cost-effective option for private organizations. There were two popular data models in this decade: a *network model* called CODASYL and a *hierarchical model* called IMS. One database system that proved to be a commercial success was the SABRE system that was used by IBM to help American Airlines manage its reservations data.

• **1970 to 1972:** A researcher at IBM, E.F. Codd published an *important paper* to propose the use of a relational database model, and his ideas changed the way people thought about databases. In his model, the database’s schema, or *logical organization, is disconnected from physical information storage, and this became the standard principle for database systems.* Codd realized that mathematics could be used to inject some solid principles and rigor into the field of database management

*Codd: "A Relational Model of Data for Large Shared Data Banks"*
Brief History of the Relational Model & Databases

• **1970s**: Two major relational database system prototypes were created between the years 1974 and 1977, and they were the Ingres, which was developed at UBC, and System R, created at IBM San Jose. *It was also in this decade that Relational Database Management System, or RDBMS, became a recognized term.*

• **1976**: A new database model called *Entity-Relationship, or ER*, was proposed by *Peter Chen* this year. This model made it possible for designers to focus on data application, instead of logical table structure.

• **1980s**: Structured Query Language, or *SQL*, became the standard query language.

• Relational database systems became a commercial success as the rapid increase in computer sales boosted the database market, and this caused a major decline in the popularity of network and hierarchical database models. DB2 became the flagship database product for IBM, and the introduction of the IBM PC resulted in the establishments of many new database.

*Chen: "The Entity-Relationship Model - Toward a Unified View of Data".*
The original relational model

• The original model had three major components
  – Structure
  – Integrity
  – Manipulation
Structure: Relations
Formally, a *relation* is a subset of the Cartesian product of zero or more sets:

Example:

\[
S = \{1, 2, 3, 4\}
\]

\[
S \times S = \{(1,1), (1,2), (1,3), (1,4), (2,1), (2,2), (2,3), (2,4),
(3,1), (3,2), (3,3), (3,4), (4,1), (4,2), (4,3), (4,4)\}
\]

The elements of \(S \times S\) are called *tuples* (or, in this case, *pairs*).

\(S \times S\) is itself a relation (subset of itself) but not very interesting.

\[
R = \{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}
\]

is slightly more interesting.
Relations - 2

\[ P = \{ \text{Trump, Obama, Bush} \} \]
Relations - 2

\( P = \{ \text{Trump, Obama, Bush} \} \)

These are both simple unary relations.
Relations - 2

P = \{ Trump, Obama, Bush \}

What is P \times Y?

How many elements are in the product set?
Relations - 2

P = \{ Trump, Obama, Bush \}

How might we interpret the following subset of P x Y?

E = \{ (Bush, 2000), (Bush, 2004),
       (Obama, 2008), (Obama, 2012),
       (Trump, 2016) \}
Codd's Insight

1. Relations can be represented as named tables.
2. Tuples are rows in a table.
3. Access to tuple contents by name (an attribute) rather than position.
4. Assume each attribute (a column in the table) has a type.
5. Every row must be unique (no duplicates)

Presidents

<table>
<thead>
<tr>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bush</td>
</tr>
<tr>
<td>Obama</td>
</tr>
<tr>
<td>Trump</td>
</tr>
</tbody>
</table>
Codd's Insight

1. Relations can be represented as named tables.
2. Tuples are rows in a table.
3. Access to tuple contents by name (an attribute) rather than position.
4. Assume each attribute (a column in the table) has a type.
5. Every row must be unique (no duplicates)

Elections

<table>
<thead>
<tr>
<th>name</th>
<th>elected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bush</td>
<td>2000</td>
</tr>
<tr>
<td>Bush</td>
<td>2004</td>
</tr>
<tr>
<td>Obama</td>
<td>2008</td>
</tr>
<tr>
<td>Obama</td>
<td>2012</td>
</tr>
<tr>
<td>Trump</td>
<td>2016</td>
</tr>
</tbody>
</table>
Codd's Insight

1. Relations can be represented as *named tables*.
2. *Tuples* are *rows* in a *table*.
3. Access to tuple contents by *name* (an *attribute*) rather than *position*.
4. Assume each attribute (a *column in the table*) has a *type*.
5. Every row must be *unique* (no duplicates)

**Elections**

<table>
<thead>
<tr>
<th>name</th>
<th>elected</th>
<th>party</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bush</td>
<td>2000</td>
<td>Republican</td>
</tr>
<tr>
<td>Bush</td>
<td>2004</td>
<td>Republican</td>
</tr>
<tr>
<td>Obama</td>
<td>2008</td>
<td>Democratic</td>
</tr>
<tr>
<td>Obama</td>
<td>2012</td>
<td>Democratic</td>
</tr>
<tr>
<td>Trump</td>
<td>2016</td>
<td>Republican</td>
</tr>
</tbody>
</table>
Codd's Insight

1. Relations can be represented as *named tables*.
2. *Tuples* are *rows* in a *table*.
3. Access to tuple contents by *name* (an *attribute*) rather than *position*.
4. Assume each attribute (a *column in the table*) has a *type*.
5. Every row must be *unique* (no duplicates)

### Elections

<table>
<thead>
<tr>
<th>name</th>
<th>elected</th>
<th>party</th>
<th>electoralVotes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bush</td>
<td>2000</td>
<td>Republican</td>
<td>271</td>
</tr>
<tr>
<td>Bush</td>
<td>2004</td>
<td>Republican</td>
<td>286</td>
</tr>
<tr>
<td>Obama</td>
<td>2008</td>
<td>Democratic</td>
<td>365</td>
</tr>
<tr>
<td>Obama</td>
<td>2012</td>
<td>Democratic</td>
<td>332</td>
</tr>
<tr>
<td>Trump</td>
<td>2016</td>
<td>Republican</td>
<td>304</td>
</tr>
</tbody>
</table>
Structural features

• The principal structural feature is the *relation* itself

• Relations are defined over types
  – Types are a conceptual pool of values which actual attributes in actual relations take their actual values from.
## Structural features - example

<table>
<thead>
<tr>
<th>DNO</th>
<th>DNAME</th>
<th>BUDGET</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Marketing</td>
<td>10M</td>
</tr>
<tr>
<td>D2</td>
<td>Development</td>
<td>12M</td>
</tr>
<tr>
<td>D3</td>
<td>Research</td>
<td>5M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENO</th>
<th>ENAME</th>
<th>DNO</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Lopez</td>
<td>D1</td>
<td>40K</td>
</tr>
<tr>
<td>E2</td>
<td>Cheng</td>
<td>D1</td>
<td>42K</td>
</tr>
<tr>
<td>E3</td>
<td>Finzi</td>
<td>D2</td>
<td>30K</td>
</tr>
<tr>
<td>E4</td>
<td>Saito</td>
<td>D2</td>
<td>35K</td>
</tr>
</tbody>
</table>
Relational Keys
Structural features - Keys

- The relational model also supports various *keys*
  - Super key
  - Candidate key
  - Primary key
  - Foreign key
Example Relation

### Vehicles - licenses unique to state

<table>
<thead>
<tr>
<th>id</th>
<th>vin</th>
<th>license</th>
<th>state</th>
<th>make</th>
<th>year</th>
<th>ownerId</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9875</td>
<td>RY3256</td>
<td>OH</td>
<td>Subaru</td>
<td>2003</td>
<td>AG9865</td>
</tr>
<tr>
<td>2</td>
<td>8365</td>
<td>ZZ876X</td>
<td>OH</td>
<td>Ford</td>
<td>2010</td>
<td>AG9865</td>
</tr>
<tr>
<td>3</td>
<td>1313</td>
<td>IBH987</td>
<td>PA</td>
<td>Ford</td>
<td>2010</td>
<td>FN9187</td>
</tr>
<tr>
<td>4</td>
<td>6512</td>
<td>RY3256</td>
<td>NY</td>
<td>Chevy</td>
<td>2011</td>
<td>KY1674</td>
</tr>
<tr>
<td>5</td>
<td>7635</td>
<td>IBH987</td>
<td>NY</td>
<td>Subaru</td>
<td>2003</td>
<td>KY1674</td>
</tr>
<tr>
<td>6</td>
<td>4433</td>
<td>UNIX99</td>
<td>CA</td>
<td>Chevy</td>
<td>1999</td>
<td>KY1674</td>
</tr>
<tr>
<td>7</td>
<td>2987</td>
<td>UNIX99</td>
<td>OH</td>
<td>Cadillac</td>
<td>2009</td>
<td>FN9187</td>
</tr>
<tr>
<td>8</td>
<td>5566</td>
<td>NZK219</td>
<td>OH</td>
<td>Cadillac</td>
<td>2009</td>
<td>FN9187</td>
</tr>
</tbody>
</table>

### Owners - ownerId unique to state

<table>
<thead>
<tr>
<th>state</th>
<th>ownerId</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OH</td>
<td>AG9865</td>
<td>Al Glover</td>
</tr>
<tr>
<td>NY</td>
<td>KY1674</td>
<td>Ken Yang</td>
</tr>
<tr>
<td>CA</td>
<td>KY1674</td>
<td>Karen Young</td>
</tr>
<tr>
<td>CA</td>
<td>KY1953</td>
<td>Karen Young</td>
</tr>
<tr>
<td>PA</td>
<td>FN9187</td>
<td>Fred North</td>
</tr>
<tr>
<td>OH</td>
<td>FN9187</td>
<td>Fred North</td>
</tr>
</tbody>
</table>
Super Keys

- A superkey is a set of attributes such that no two rows may have the same values for these attributes.

- Which are superkeys for the Vehicles relation?
  - \{ id, vin, license, state, make, year, ownerId \}
  - \{ state, make, year, ownerId \}
  - \{ id, vin, state \}
  - \{ license, state \}
  - \{ license, year, make \}
  - \{ id \}
  - \{ vin, ownerId \}
  - \{ vin \}
Super Keys

- A superkey is a set of attributes such that no two rows may have the same values for these attributes.

- Which are superkeys for the Vehicles relation?
  
  \[
  \{ \text{id, vin, license, state, make, year, ownerId} \} \checkmark \\
  \{ \text{state, make, year, ownerId} \} \times \\
  \{ \text{id, vin, state} \} \checkmark \\
  \{ \text{license, state} \} \checkmark \\
  \{ \text{license, year, make} \} \times \\
  \{ \text{id} \} \checkmark \\
  \{ \text{vin, ownerId} \} \checkmark \\
  \{ \text{vin} \} \checkmark
  \]
Candidate Keys

- A *candidate key* is a superkey with no unnecessary (extraneous) attributes.
- Minimal set of attributes needed for unique row identification.
- Which are the candidate keys for the Vehicles relation?
  
  - \{ license, state, ownerId \}
  - \{ id, vin, make \}
  - \{ license, state \}
  - \{ id, make, year \}
  - \{ id \}
  - \{ vin, ownerId \}
  - \{ vin \}
  - \{ id, vin, year \}
Candidate Keys

- A **candidate key** is a superkey with no unnecessary (extraneous) attributes.
- Minimal set of attributes needed for unique row identification.
- Which are the candidate keys for the Vehicles relation?
  
  - `{ license, state, ownerld } ×
  - `{ id, vin, make } ×
  - `{ license, state } ✓
  - `{ id, make, year } ×
  - `{ id } ✓
  - `{ vin, ownerld } ×
  - `{ vin } ✓
  - `{ id, vin, year } ×
Primary keys

• A candidate key that gets special treatment (syntactic advantage)

• If a relation has one candidate key, then it can be the primary key. If more than 1, you choose arbitrarily

• Main thing to remember is that candidate keys, not primary keys, have significance from a relational point of view
Key Classification

Primary vs. Alternate

- The **primary key** is the candidate key typically used to find a row and for cross-referencing with other relations.
- All other candidate keys are **alternate keys**.

Simple vs. Compound

- A **simple key** is a single attribute candidate key (e.g. vin).
- A **compound key** is a multi-attribute candidate key (e.g., state+license).

Natural vs. Surrogate (or Synthetic)

- A **natural key** is part of the problem context (e.g., vin or state+license).
- A **surrogate** or **synthetic key** is generated by the underlying database system and has no inherent meaning (e.g., id).
Owners Relation

1. What are the superkeys? How many?

2. What are the candidate keys? How many?

3. What would you select as the primary key?

4. Is the primary key simple or compound?

5. Is the primary key natural or synthetic?
Owners Relation

1. What are the superkeys? How many?
   { state, ownerld, name }, { state, ownerld }

2. What are the candidate keys? How many?
   { state, ownerld }

3. What would you select as the primary key?
   { state, ownerld } - the only available choice

4. Is the primary key simple or compound?
   Compound

5. Is the primary key natural or synthetic?
   Natural
Foreign keys

- A set of attributes whose values are required to match the values of some candidate key in some other relation
- In the example, EMP.DNO values need to appear in DEPT.DNO

<table>
<thead>
<tr>
<th>DEPT</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DNO</td>
<td>DNAME</td>
<td>BUDGET</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>Marketing</td>
<td>10M</td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>Development</td>
<td>12M</td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>Research</td>
<td>5M</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMP</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ENO</td>
<td>ENAME</td>
<td>DNO</td>
<td>SALARY</td>
</tr>
<tr>
<td>E1</td>
<td>Lopez</td>
<td>D1</td>
<td>40K</td>
</tr>
<tr>
<td>E2</td>
<td>Cheng</td>
<td>D1</td>
<td>42K</td>
</tr>
<tr>
<td>E3</td>
<td>Finzi</td>
<td>D2</td>
<td>30K</td>
</tr>
<tr>
<td>E4</td>
<td>Saito</td>
<td>D2</td>
<td>35K</td>
</tr>
</tbody>
</table>
Foreign Keys

A foreign key is a set of one or more attributes used to reference a row (tuple) in the same or a different relation. A foreign key refers to the attributes of the relation being referenced. Foreign keys may be simple or compound. Similarly, foreign keys may be natural or synthetic. What foreign key is sufficient to find the Owner of a specific Vehicle?
What foreign key is sufficient to find the Owner of a specific Vehicle?

{ ownerID, state } in Vehicles is a foreign key to Owners

### Vehicles - licenses unique to state

<table>
<thead>
<tr>
<th>id</th>
<th>vin</th>
<th>license</th>
<th>state</th>
<th>make</th>
<th>year</th>
<th>ownerID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9875</td>
<td>RY3256</td>
<td>OH</td>
<td>Subaru</td>
<td>2003</td>
<td>AG9865</td>
</tr>
<tr>
<td>2</td>
<td>8365</td>
<td>ZZ876X</td>
<td>OH</td>
<td>Ford</td>
<td>2010</td>
<td>AG9865</td>
</tr>
<tr>
<td>3</td>
<td>1313</td>
<td>IBH987</td>
<td>PA</td>
<td>Ford</td>
<td>2010</td>
<td>FN9187</td>
</tr>
<tr>
<td>4</td>
<td>6512</td>
<td>RY3256</td>
<td>NY</td>
<td>Chevy</td>
<td>2011</td>
<td>KY1674</td>
</tr>
<tr>
<td>5</td>
<td>7635</td>
<td>IBH987</td>
<td>NY</td>
<td>Subaru</td>
<td>2003</td>
<td>KY1674</td>
</tr>
<tr>
<td>6</td>
<td>4433</td>
<td>UNIX99</td>
<td>CA</td>
<td>Chevy</td>
<td>1999</td>
<td>KY1674</td>
</tr>
<tr>
<td>7</td>
<td>2987</td>
<td>UNIX99</td>
<td>OH</td>
<td>Cadillac</td>
<td>2009</td>
<td>FN9187</td>
</tr>
<tr>
<td>8</td>
<td>5566</td>
<td>NZK219</td>
<td>OH</td>
<td>Cadillac</td>
<td>2009</td>
<td>FN9187</td>
</tr>
</tbody>
</table>

### Owners - ownerID unique to state

<table>
<thead>
<tr>
<th>state</th>
<th>ownerID</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OH</td>
<td>AG9865</td>
<td>Al Glover</td>
</tr>
<tr>
<td>NY</td>
<td>KY1674</td>
<td>Ken Yang</td>
</tr>
<tr>
<td>CA</td>
<td>KY1674</td>
<td>Karen Young</td>
</tr>
<tr>
<td>CA</td>
<td>KY1953</td>
<td>Karen Young</td>
</tr>
<tr>
<td>PA</td>
<td>FN9187</td>
<td>Fred North</td>
</tr>
<tr>
<td>OH</td>
<td>FN9187</td>
<td>Fred North</td>
</tr>
</tbody>
</table>
Integrity: Well Formed Relations
Integrity Constraints

An *integrity constraint* is a boolean expression that must be true for a relation (or set of relations) to be in a valid state.

- No student can enroll in more than four classes.
- Every employee must have a positive salary.

There are two generic integrity constraints every valid model must adhere to:

- Entity integrity
- Referential integrity
Generic Integrity Constraints

**Entity integrity**
- No tuple / row has a NULL value for a primary key attribute.
- NULL represents nothing / void / unknown / non-existent.
- NULL makes it impossible to find unique tuples

**Referential integrity**
- A valid (non-NULL) foreign key must always reference an existing primary key in the referenced relation.
- No "dangling references" (like dangling pointers in C).
- Must be checked when changes are made to the referenced relation.
Integrity features
- Entity integrity

• Primary key attributes don’t permit nulls
  – A null is a marker for an unknown value
• If the primary key is unknown, then we would not ever know which tuple we are referencing

<table>
<thead>
<tr>
<th>EMP</th>
<th>ENAME</th>
<th>DNO</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Lopez</td>
<td>D1</td>
<td>40K</td>
</tr>
<tr>
<td>E2</td>
<td>Cheng</td>
<td>D1</td>
<td>42K</td>
</tr>
<tr>
<td>E3</td>
<td>Finzi</td>
<td>D2</td>
<td>30K</td>
</tr>
<tr>
<td>null</td>
<td>Saito</td>
<td>D2</td>
<td>35K</td>
</tr>
</tbody>
</table>

vs.

<table>
<thead>
<tr>
<th>EMP</th>
<th>ENAME</th>
<th>DNO</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Lopez</td>
<td>D1</td>
<td>40K</td>
</tr>
<tr>
<td>E2</td>
<td>Cheng</td>
<td>D1</td>
<td>42K</td>
</tr>
<tr>
<td>E3</td>
<td>Finzi</td>
<td>D2</td>
<td>30K</td>
</tr>
<tr>
<td>E4</td>
<td>Saito</td>
<td>D2</td>
<td>null</td>
</tr>
</tbody>
</table>
Integrity features
-Referential integrity

• There must not be any unmatched foreign key values

• “if B references A, then A must exist”

• In the foreign keys example, a database would be in violation if EMP.DNO had a value with no corresponding value in DEPT.DNO
Manipulation: Relational Algebra
Manipulative Features

• The manipulative part of the relation model consists of:

  – A set of relational operators, called relational algebra

  – A relational assignment operator
Manipulative features
- Assignment operator

- Allows the value of some relational expression to be assigned to some other relation

- Fundamentally how updates (INSERT, DELETE, UPDATE) are done
Manipulative features

- Relational algebra

• Consists of a set of operators that allow “new” relations to be derived from “old” relations

• Each operators takes at least one relation as input and produces another relation as output
  – e.g. minus: takes 2 relations as input and subtracts one from the other

• Important: the output is also a relation. This is the closure property of relational algebra
Manipulative features – the original eight

- Restrict
- Project
- Product
- Intersect
- Union
- Difference
- Join
- Divide

*(more on these operations when we get to Alloy)*