Relational Data Model

A Brief History of Data Models
- 1950s file systems, punched cards
- 1960s hierarchical
  - IMS
- 1970s network
  - CODASYL, IDMS
- 1980s relational
  - INGRES, ORACLE, DB2, Sybase
  - Paradox, dBase
- 1990s object oriented and object relational
  - O2, GemStone, Ontos

Relational Model
- Sets
  - collections of items of the same type
  - no order
  - no duplicates
- Mappings

Relational Model Notes
- no duplicate tuples in a relation
- a relation is a set of tuples
- no ordering of tuples in a relation
- a relation is a set
- attributes of a relation have an implied ordering
  - but used as functions and referenced by name, not position
- every tuple must have attribute values drawn from all of the domains of the relation or the special value NULL
- all a domain’s values and hence attribute’s values are atomic.

COURSE

<table>
<thead>
<tr>
<th>Courseno</th>
<th>Subject</th>
<th>Lecturer</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS250</td>
<td>Programming</td>
<td>Lindsey</td>
<td>Sun</td>
</tr>
<tr>
<td>CS260</td>
<td>Graphics</td>
<td>Hubbold</td>
<td>Sun</td>
</tr>
<tr>
<td>CS270</td>
<td>Micros</td>
<td>Woods</td>
<td>PC</td>
</tr>
<tr>
<td>CS290</td>
<td>Verification</td>
<td>Barringer</td>
<td>Sun</td>
</tr>
</tbody>
</table>

Comparative Terms

<table>
<thead>
<tr>
<th>Formal</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation schema</td>
<td>Table description</td>
</tr>
<tr>
<td>Relation</td>
<td>Table</td>
</tr>
<tr>
<td>Tuple</td>
<td>Row</td>
</tr>
<tr>
<td>Attribute</td>
<td>Column</td>
</tr>
<tr>
<td>Domain</td>
<td>Value set</td>
</tr>
</tbody>
</table>

Notation
- Course (courseno, subject, equipment)
- Student (studno, name, hons)
- Enrol (studno, courseno, labmark)
Keys

- **SuperKey**
  - a set of attributes whose values together uniquely identify a tuple in a relation
- **Candidate Key**
  - a superkey for which no proper subset is a superkey...a key that is minimal.
  - Can be more than one for a relation
- **Primary Key**
  - a candidate key chosen to be the main key for the relation.
  - One for each relation
- Keys can be composite

Foreign Key

- a set of attributes in a relation that exactly matches a (primary) key in another relation
  - the names of the attributes don't have to be the same but must be of the same domain
  - a foreign key in a relation A matching a primary key in a relation B represents a
  - many:one relationship between A and B

Referential Integrity

- Student(studno, name, tutor, year)
- Staff(lecturer, roomno, appraiser)
  - **CASCADE**
    - delete all matching foreign key tuples
    - eg. STUDENT
  - **RESTRICT**
    - can't delete primary key tuple STAFF whilst a foreign key tuple STUDENT matches
  - **NULLIFY**
    - foreign key STUDENT.tutor set to null if the foreign key ids allowed to take on null

Entity Integrity and Nulls

No part of a key can be null

- Attribute values
  - Atomic
  - Known domain
  - Sometimes can be null

- THREE categories of null values
  1. Not applicable
  2. Not known
  3. Absent (not recorded)

Relational Model

- General
- Simple
- Flexible
- Easy to query declaratively without programming
  - But.....
    - Good design essential
    - Integrity essential
    - Poor semantics
    - Relationships based on 'value-matching'
Relational Design

<table>
<thead>
<tr>
<th>student</th>
<th>name</th>
<th>tutor</th>
<th>roomno</th>
<th>course</th>
<th>lab</th>
<th>mark</th>
<th>subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>jones</td>
<td>bush</td>
<td>2.26</td>
<td>cs250</td>
<td>66</td>
<td></td>
<td>programming</td>
</tr>
<tr>
<td>s1</td>
<td>jones</td>
<td>bush</td>
<td>2.26</td>
<td>cs260</td>
<td>80</td>
<td></td>
<td>graphics</td>
</tr>
<tr>
<td>s1</td>
<td>jones</td>
<td>bush</td>
<td>2.26</td>
<td>cs270</td>
<td>47</td>
<td></td>
<td>electronics</td>
</tr>
<tr>
<td>s2</td>
<td>brown</td>
<td>kahn</td>
<td>IT206</td>
<td>cs250</td>
<td>67</td>
<td></td>
<td>programming</td>
</tr>
<tr>
<td>s2</td>
<td>brown</td>
<td>kahn</td>
<td>IT206</td>
<td>cs270</td>
<td>65</td>
<td></td>
<td>electronics</td>
</tr>
<tr>
<td>s3</td>
<td>smith</td>
<td>goble</td>
<td>2.82</td>
<td>cs270</td>
<td>49</td>
<td></td>
<td>electronics</td>
</tr>
<tr>
<td>s4</td>
<td>bloggs</td>
<td>goble</td>
<td>2.82</td>
<td>cs280</td>
<td>50</td>
<td></td>
<td>design</td>
</tr>
<tr>
<td>s5</td>
<td>jones</td>
<td>goble</td>
<td>2.34</td>
<td>cs250</td>
<td>60</td>
<td></td>
<td>programming</td>
</tr>
<tr>
<td>s5</td>
<td>jones</td>
<td>goble</td>
<td>2.34</td>
<td>cs250</td>
<td>60</td>
<td></td>
<td>programming</td>
</tr>
<tr>
<td>s6</td>
<td>brown</td>
<td>kahn</td>
<td>IT206</td>
<td>cs250</td>
<td>2</td>
<td></td>
<td>programming</td>
</tr>
<tr>
<td>s6</td>
<td>brown</td>
<td>kahn</td>
<td>IT206</td>
<td>cs270</td>
<td>65</td>
<td></td>
<td>electronics</td>
</tr>
<tr>
<td>s7</td>
<td>null</td>
<td>null</td>
<td>2.82</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>

Informal guidelines

- Semantics of the attributes
  - easy to explain relation
  - doesn’t mix concepts
- Reducing the redundant values in tuples
- Choosing attribute domains that are atomic
- Reducing the null values in tuples
- Disallowing spurious tuples

Definitions

- **Cartesian Product** The cartesian product \( \times \) between n sets is the set of all possible combinations of the elements of those sets.
- **Domain** set of all possible values for an attribute; for attribute \( A \), the domain is represented as \( \text{dom}(A) \). A domain has a format and a base data type.
- **Relation Schema** denoted by \( R(A_1, A_2, \ldots, A_n) \), is made up of relation name \( R \) and list of attributes \( A_1, A_2, \ldots, A_n \).
- **Relation** a subset of the cartesian product of its domains. Given a relation schema \( R \), a relation on that schema \( r \), a set of attributes \( A_1, A_2, \ldots, A_n \) for that relation then \( r(R) \subseteq (\text{dom}(A_1) \times \text{dom}(A_2) \times \ldots \times \text{dom}(A_n)) \).
- **Attribute** a function on a domain for each instance of the mapping or tuple
- **Attribute Value** the result of the attribute function. Each instance of the mapping is represented by one attribute value drawn from each domain or a special NULL value. Given a tuple \( t \) and an attribute \( A \) for a relation \( r \), \( t[A] \rightarrow a \), where \( a \) is the attribute’s value for that tuple.

- **(N)-tuple** a set of (n) attribute-value pairs representing a single instance of a relation’s mapping between its domains.
- **Degree** the number of attributes a relation has.
- **Cardinality** a number of tuples a relation has.
- **Roles** several attributes can have the same domain; the attributes indicate different roles in the relation.
- **Key**/**SuperKey** a set of attributes whose values together uniquely identify every tuple in a relation. Let \( t1 \) and \( t2 \) be two tuples on relation \( r \) of relation schema \( R \), and \( sk \) be a set of attributes whose values are the key for the relation schema \( R \), then \( t1[sk] \neq t2[sk] \).
- **Candidate Key** a (super)key that is minimal, i.e., has no proper subsets that still uniquely identify every tuple in a relation. There can be several for one relation.
- **Primary Key** a candidate key chosen to be the main key for the relation. There is only one for each relation.
- **Foreign Key** a candidate key of relation \( A \) situated in relation \( B \).
- **Database** a set of relations.