The Relational Model theoretical foundation

• Relational Model Concepts
• Characteristics of Relations
• Relational Integrity Constraints
  – Key Constraints
  – Entity Integrity Constraints
  – Referential Integrity Constraints
• Operations
Relational Model

• A particular way of structuring data (relations)
• Simple
• Mathematically based
  – Expressions (queries) can be analyzed by DBMS
  – Transformed to equivalent expressions automatically (query optimization)
    • Optimizers have limits (⇒ programmer needs to know how queries are evaluated and optimized)

Basis of Relational Model

• A RELATION is a mathematical concept based on the ideas of sets.
• Relational model of data is based on the concept of a Relation.
• was first proposed by Dr. E.F. Codd of IBM in 1970 in the following paper - "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970.
Informal Definition

- **RELATION:** A table of values
- A relation may be thought of as a *set of rows.*
- A relation may alternately be thought of as a *set of columns.*
- Each row of the relation may be given an identifier.
- Each column typically is called by its column name or column header or attribute name.

### Routes

<table>
<thead>
<tr>
<th>RID</th>
<th>RName</th>
<th>Grade</th>
<th>Rating</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Last Tango</td>
<td>II</td>
<td>12</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Garden Path</td>
<td>I</td>
<td>2</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>The Sluice</td>
<td>I</td>
<td>8</td>
<td>60</td>
</tr>
<tr>
<td>4</td>
<td>Picnic</td>
<td>III</td>
<td>3</td>
<td>400</td>
</tr>
</tbody>
</table>

### Climbers

<table>
<thead>
<tr>
<th>CId</th>
<th>Cname</th>
<th>Skill</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Edmund</td>
<td>EXP</td>
<td>80</td>
</tr>
<tr>
<td>214</td>
<td>Arnold</td>
<td>BEG</td>
<td>25</td>
</tr>
<tr>
<td>313</td>
<td>Bridget</td>
<td>EXP</td>
<td>33</td>
</tr>
<tr>
<td>212</td>
<td>James</td>
<td>MED</td>
<td>27</td>
</tr>
</tbody>
</table>

### Climbs

<table>
<thead>
<tr>
<th>CId</th>
<th>RID</th>
<th>Date</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>1</td>
<td>10/10/88</td>
<td>5</td>
</tr>
<tr>
<td>123</td>
<td>3</td>
<td>11/08/87</td>
<td>1</td>
</tr>
<tr>
<td>313</td>
<td>1</td>
<td>12/08/89</td>
<td>5</td>
</tr>
<tr>
<td>214</td>
<td>2</td>
<td>08/07/92</td>
<td>2</td>
</tr>
<tr>
<td>215</td>
<td>3</td>
<td>06/07/94</td>
<td>3</td>
</tr>
</tbody>
</table>
• Each **route** has an id, a name, a grade (an estimate of the time needed), a rating (how difficult it is), and a height.

• Each **climber** has an id, a name, a skill level and an age.

• A **climb** records who climbed what route on what date and how long it took (duration).

• Observe that the data values in these tables are all “simple”. None of them are complex structures -- like other relations.

---

**Relational Model Terminology**

• Table = *relation*.

• Column headers = *attributes*

• Row = *tuple*

• The possible value of each attribute = *domain*
  – E.g., the domain of **CName** is *string* and that for **Rating** is *real*.

• *Relation schema* = name(attributes) + other structure info.,
  – e.g., keys, other constraints.

• *Relation instance* is current set of rows for a relation schema.

• *Database schema* = collection of relation schemas.
FORMAL DEFINITIONS

• A **Relation** may be defined in multiple ways
• The **Schema** of a Relation:
  R (A1, A2, .....An)
  Relation R is defined over **attributes** A1, A2, .....An

CUSTOMER (Cust-id, Cust-name, Address, Phone#)

• CUSTOMER is a *relation* defined over the four **attributes** Cust-id, Cust-name, Address, Phone#
• Each attribute has a *domain* or a set of valid values.
  – E.g., the *domain* of Cust-id is 6 digit numbers.
• A **tuple** is an ordered set of values
  – <632895, "John Smith", "101 Main St. Atlanta, GA  30332", "(404) 894-2000"> is a triple belonging to the CUSTOMER relation.
• A relation may be regarded as a set of tuples (rows).
Formal Definition (Cont.)

- A relation may be regarded as a set of tuples
- A relation is formed over the cartesian product of the sets
  - each set has values from a domain
  - domain is used in a specific role which is conveyed by the attribute name.
  - E.g., attribute Cust-name: the domain of strings of 25 characters.
- Formally,
  Given R(A1, A2, ........, An)
  \( r(R) \) subset-of dom (A1) X dom (A2) X ....X dom(An)
  R: schema of the relation (intension)
  \( r \) of R: a specific "value" or population of R. (extension)

Definition Summary

<table>
<thead>
<tr>
<th>Informal Terms</th>
<th>Formal Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Relation</td>
</tr>
<tr>
<td>Column</td>
<td>Attribute/Domain</td>
</tr>
<tr>
<td>Values in a column</td>
<td>Domain</td>
</tr>
<tr>
<td>Row</td>
<td>Tuple/Instance</td>
</tr>
<tr>
<td>Table Definition</td>
<td>Schema of Relation</td>
</tr>
<tr>
<td>Populated Table</td>
<td>Extension</td>
</tr>
</tbody>
</table>
Characteristics of Relations

- **Ordering of tuples in a relation** \( r(R) \):
  - Not ordered (like a set)

- **Ordering of attributes in a relation** schema \( R \) (and of values within each tuple):
  - the attributes in \( R(A_1, A_2, ..., A_n) \) and the values in \( t=\langle v_1, v_2, ..., v_n \rangle \) to be *ordered*
  - Not important
• **Values in a tuple**: *atomic* (indivisible)
  – **Null** value: unknown or inapplicable values to certain tuples

• **Notation**
  – \( t[Ai] = vi \) (the value of attribute \( Ai \) for tuple \( t \)).
  – \( t[Au, Av, ..., Aw] \) refers to the tuple of \( t \) containing the values of attributes \( Au, Av, ..., Aw \), respectively.

---

**Figure 7.2** The relation **STUDENT** from Figure 7.1, with a different order of tuples.

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>Name</th>
<th>SSN</th>
<th>HomePhone</th>
<th>Address</th>
<th>OfficePhone</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dick Taylor</td>
<td>421-01-2030</td>
<td>null</td>
<td>365 Ege Road</td>
<td>748-1930</td>
<td>25</td>
<td>3.00</td>
<td></td>
</tr>
<tr>
<td>Barbara Brown</td>
<td>562-89-4230</td>
<td>265/891</td>
<td>1796 Forest Lane</td>
<td>null</td>
<td>19</td>
<td>2.26</td>
<td></td>
</tr>
<tr>
<td>Charlie Cooper</td>
<td>498-20-1100</td>
<td>395-652</td>
<td>295 Lab Lane</td>
<td>746-6420</td>
<td>19</td>
<td>3.95</td>
<td></td>
</tr>
<tr>
<td>Katharine Ashley</td>
<td>831-02-1425</td>
<td>375-439</td>
<td>526 Happy Road</td>
<td>null</td>
<td>18</td>
<td>2.95</td>
<td></td>
</tr>
</tbody>
</table>
Relational Data Model

Relation as table
- Rows = tuples
- Columns = components
- Names of columns = attributes
- Set of attribute names = schema
  REL (A1, A2, ..., An)

Set theoretic
- Domain — set of values
  like a data type
- Cartesian product (or product)
  \[ D_1 \times D_2 \times \ldots \times D_n \]
  n-tuples (V1, V2, ..., Vn)
  s.t., V1 \in D_1, V2 \in D_2, ..., Vn \in D_n
- Relation-subset of cartesian product
  of one or more domains
  FINITE only; empty set allowed
- Tuples = members of a relation inst.
- Arity (degree) = number of domains
- Components = values in a tuple
- Domains — corresp. with attributes
- Cardinality = number of tuples

Relation Instance

• Relation is a set of tuples
  - Tuple ordering immaterial
  - No duplicates
  - Cardinality of relation = number of tuples
• All tuples in a relation have the same structure; constructed from the same set of attributes
  - Attributes named (ordering immaterial)
  - Value of an attribute drawn from the attribute’s domain
  - Arity (Degree) = number of attributes
Relation Instance (Example)

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Address</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111111</td>
<td>John</td>
<td>123 main</td>
<td>freshman</td>
</tr>
<tr>
<td>2345678</td>
<td>Mary</td>
<td>456 cedar</td>
<td>sophomore</td>
</tr>
<tr>
<td>4433322</td>
<td>Art</td>
<td>77 so. 3rd</td>
<td>senior</td>
</tr>
<tr>
<td>7654321</td>
<td>Pat</td>
<td>88 no. 4th</td>
<td>sophomore</td>
</tr>
</tbody>
</table>

Student

Relation Schema

- Relation name
- Attribute names and domains
- Integrity constraints - e.g.,:
  - The values of a particular attribute in all tuples are unique
  - The values of a particular attribute in all tuples are greater than 0
- Default values
Relational Database

- **Finite** set of relations
- Each relation consists of a schema and an instance
- Database schema = set of relation schemas (and other things)
- Database instance = set of (corresponding) relation instances

Integrity Constraints

- Part of schema
- Restriction on state (or sequence of states) of database
- Enforced by DBMS
- Intra-relational - involve only one relation
  - Part of relation schema
  - e.g., all Ids are unique
- Inter-relational - involve several relations
  - Part of relation schema or database schema
Integrity Constraints

• Constraints: *conditions* that must hold on *all* valid relation instances.

• Tree main constrains:
  – **Key** constraints (single relation)
  – **entity integrity** constraints (single relation)
  – **referential integrity** constraints (two relations)

Key Constrains

• **Superkey**: A set of attributes SK of R such that no two tuples *in any valid relation instance r*(*R*) will have the same value for SK
  – for any distinct tuples t1 and t2 in r(*R*),
    \[ t1[SK] \neq t2[SK] \]

• **Key**: A "minimal" superkey.
  – a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.
Example:

CAR(State, Reg#, SerialNo, Make, Model, Year)

- Two candidate keys:
  - Key1 = {State, Reg#}
  - Key2 = {SerialNo}
  - Are Key1 and Key2 superkeys?
- {SerialNo, Make}, key or superkey?
- Primary key: If a relation has several candidate keys, one is chosen arbitrarily to be the primary key.
  - The primary key attributes are underlined.

Entity Integrity

- Relational Database Schema: A set S of relation schemas that belong to the same database.
  - S is the name of the database.
    S = {R1, R2, ..., Rn}
- Entity Integrity: The primary key attributes PK of each relation schema R in S cannot have null values in any tuple of r(R).
  - Why?
Entity Integrity

- **Relational Database Schema**: A set $S$ of relation schemas that belong to the same database.
  - $S$ is the *name* of the database.
  - $S = \{R1, R2, ..., Rn\}$
- **Entity Integrity**: The *primary key attributes* $PK$ of each relation schema $R$ in $S$ *cannot* have null values in any tuple of $r(R)$.
  - primary key values are used to *identify* the individual tuples.
  - $t[PK] <> \text{null}$ for any tuple $t$ in $r(R)$

Entity Integrity

- Other attributes (Non PK) of $R$ may be similarly constrained to disallow null values
- E.g., Name, Address
### Referential Integrity
(Foreign Key Constraints)

- A constraint involving *two* relations
- specify a *relationship* among tuples in two relations:
  - the *referencing relation* (R1) and the *referenced relation* (R2)
  - FK(*foreign key* attributes) of R1 reference PK of R2

- displayed in a relational database schema as a directed arc from R1.FK to R2.PK
Figure 7.6 One possible relational database state corresponding to the COMPANY schema.

Figure 7.7 Referential integrity constraints displayed on the COMPANY relational database schema diagram.
Database Schema (Example)

- **Student** (Id: INT, Name: STRING, Address: STRING, Status: STRING)
- **Professor** (Id: INT, Name: STRING, DeptId: DEPTS)
- **Course** (DeptId: DEPTS, CrsName: STRING, CrsCode: COURSES)
- **Transcript** (CrsCode: COURSES, StudId: INT, Grade: GRADES, Semester: SEMESTERS)
- **Department** (DeptId: DEPTS, Name: STRING)

Foreign Key Constraint (Cont.)

- **Referential integrity** - attribute named in one relation must correspond to tuple(s) in another that describes the item
  - Transcript (CrsCode) references Course(CrsCode)
  - Professor(DeptId) references Department(DeptId)
- **K1 is a foreign key** of R1 referring to K2 in R2
  - if v is a value of K1, there is a unique tuple of R2 in which K2 has value v
  - This is a special case of referential integrity: K2 must be a candidate key of R2 (CrsCode is a key of Course)
  - If no row exists in R2 -- violation of referential integrity
  - Not all rows of R2 need to be referenced. Relationship is not symmetric (some course might not be taught)
  - Value of a foreign key might not be specified (DeptId column of some professor might be null)
Foreign Key Constraint
(Example)

- Names of K1 and K2 need not be the same.
  - With tables:
    Teaching(CrsCode: COURSES, Sem: SEMESTERS, ProfId: INT)
    Professor(Id: INT, Name: STRING, DeptId: DEPTS)
    ProfId attribute of Teaching references Id attribute of Professor
- R1 and R2 need not be distinct.
  - Employee(Id:INT, MgrId:INT, ....)
    • Employee(MgrId) references Employee(Id)
  - Every manager is also an employee and hence has a unique row in Employee
Foreign Key (con’t)

• Foreign key might consist of several columns
  – (CrsCode, Semester) of Transcript references
    (CrsCode, Sem) of Teaching
• R1(A1, …An) references R2(B1, …Bn)
  – There exists a 1:1 relationship between A1,….An and B1,…Bn
  – Ai and Bi have same domains (although not necessarily the same names)
  – B1,…Bn is a candidate key of R2