COMP163
Database Management Systems
September 4, 2008

Lecture 4 – Chapter 5
The Relational Data Model
Ted Codd proposed the *relational data model* in 1970.

He received the ACM Turing Award in 1981.
Relational Data Model

- Core of majority of modern databases
- Virtually all business relies on some form of relational database
- Solid theoretical/mathematical foundation
- Simple but robust implementation
Models, Schemas and States

- A *data model* defines the constructs available for defining a schema
  - defines possible schemas

- A *schema* defines the constructs available for storing the data
  - defines database structure
  - limits the possible database states

- A *database state* (or *instance*) is all the data at some point in time
  - the database content
Models, Schemas and States

- **data model**
  - fixed by the DBMS

- **schema**
  - defined by the DB designer
  - generally fixed once defined *

- **database state**
  - changes over time due to user updates

* schema modifications are possible once the database is populated, but this generally causes difficulties
The Relational Data Model

- All data is stored in relations
  - relations are sets, but generally viewed as 2D tables

- DB schema = a set of relation specifications
  - the specification of a particular relation is called a relation schema

- DB state = the data stored in the relations
  - the data in a particular relation is called a relation state
    (or relation instance or simply relation)

*Principle of Uniform Representation:*
The entire content of a relational database is represented in one and only one way: namely, as attribute values within tuples within relations.
RDM Schemas

- External View
- Conceptual Schema
- Internal Schema

relation specifications

mapping from relations to storage layout (files)
Relational Data Definition

Database designer enters the definition of relation schemas

SQL DDL = relation definition language (CREATE TABLE)
Relation Schemas and Relation Instances
A relation is defined by a name and a set of attributes.

Each attribute has a name and a domain:
- A domain is a set of possible values.
- Types are domain names.
- All domains are sets of atomic values – RDM does not allow complex data types.
- Domains may contain a special null value.
Example Relation Schema

relation name → StockItem

Attribute: 
- ItemID: string(4)
- Description: string(50)
- Price: currency/dollars
- Taxable: boolean

Domain: 
- ItemID: string(4)
- Description: string(50)
- Price: currency/dollars
- Taxable: boolean

set of attributes

attribute names

attribute domains
Definition: Relation Schema

- Relation Schema
  \[ R(A_1, A_2, \ldots, A_n) \]
  - \( R \) is the relation name
  - \( A_1 \ldots A_n \) are the attribute names

- Domains are denoted by
  \[ \text{dom}(A_i) \]

- degree = the number of attributes
Example Relation Schema

STOCKITEM(ItemId, Description, Price, Taxable)

\[\text{dom(ItemId)} = \text{string(4)}\]
\[\text{dom(Description)} = \text{string(50)}\]
\[\text{dom(Price)} = \text{currency/dollars}\]
\[\text{dom(Taxable)} = \text{boolean}\]

\text{degree of STOCKITEM} = 4
Definition: Relation

- A relation is denoted by \( r(R) \)
  - \( R \) is the name of the relation schema for the relation

- A relation is a set of tuples

  \[ r(R) = (t_1, t_2, \ldots, t_m) \]
Definition: Relation

- Each tuple is an ordered list of \( n \) values
  \[ t = < v_1, v_2, \ldots, v_n > \]
  - \( n \) is the degree of \( R \)

- Each value in the tuple must be in the domain of the corresponding attribute
  \[ v_i \in \text{dom}(A_i) \]

- Alternate notations:
  - \( i^{th} \) value of tuple \( t \) is also referred to as
    \[ v_i = t[A_i] \quad \text{or} \quad v_i = t.A_i \]
Example Relation

\[ r(\text{STOCKITEM}) = \{ < \text{I119}, \text{"Monopoly"}, \$29.95, \text{true} >, \]
\[ < \text{I007}, \text{"Risk"}, \$25.45, \text{true} >, \]
\[ < \text{I801}, \text{"Bazooka Gum"}, \$0.25, \text{false} > \} \]

\[ t_2 = < \text{I007}, \text{"Risk"}, \$25.45, \text{true} > \]

\[ t_2[\text{Price}] = t_2.\text{Price} = \$25.45 \]

\[ t_2[\text{Price}] \in \text{dom(Price)} = \text{currency/dollars} \]
Characteristics of Relations

• A relation is a set
  • tuples are unordered
  • no duplicate tuples

• Attribute values within tuples are ordered
  • values are matched to attributes by position

• alternate definition defines a tuple as a set of (name, value) pairs, which makes ordering of tuple unnecessary (we won’t use this definition)
Values in tuples are *atomic*

- atomic = non-structured 
  (similar to primitive types in C++)
- implication:
  no nested relations or other complex data structures

If domain includes *null* values,
*null* may have many interpretations
- "does not exist"
- "not applicable"
- "unknown"
Theory vs. Reality

- The theoretical data model is mathematical:
  - a relation is a set of tuples
  - this is Codd's definition

- In the real-world, the model is practical:
  - efficiency concerns
  - excepted standard: SQL
  - a relation is a table, not a set
  - a relation may have order and duplicates
SQL: Relation States

- A relation is viewed as a table
- The attributes define the columns of the table
- Each row in the table holds related values for each attribute
  - a row often represents a conceptual entity (object)
- Values in each column must come from the domain of the attribute
  - the values are instances of the attribute type
### Relation: Table Representation

Each row collects related attribute values

<table>
<thead>
<tr>
<th>StockItem</th>
<th>ItemId</th>
<th>Description</th>
<th>Price</th>
<th>Taxable</th>
</tr>
</thead>
<tbody>
<tr>
<td>I119</td>
<td>Monopoly</td>
<td>$29.95</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>I007</td>
<td>Risk</td>
<td>$25.45</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>I801</td>
<td>Bazooka Gum</td>
<td>$0.25</td>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>

Column values all come from the same domain
## Example Relation

**Relation Name:** STUDENT

**Attributes:**
- Name
- Ssn
- Home_phone
- Address
- Office_phone
- Age
- Gpa

<table>
<thead>
<tr>
<th>Name</th>
<th>Ssn</th>
<th>Home_phone</th>
<th>Address</th>
<th>Office_phone</th>
<th>Age</th>
<th>Gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benjamin Bayer</td>
<td>305-61-2435</td>
<td>373-1616</td>
<td>2918 Bluebonnet Lane</td>
<td>NULL</td>
<td>10</td>
<td>3.21</td>
</tr>
<tr>
<td>Chung-cha Kim</td>
<td>381-62-1245</td>
<td>375-4409</td>
<td>125 Kirby Road</td>
<td>NULL</td>
<td>18</td>
<td>2.89</td>
</tr>
<tr>
<td>Dick Davidson</td>
<td>422-11-2320</td>
<td>NULL</td>
<td>3452 Elgin Road</td>
<td>749-1253</td>
<td>25</td>
<td>3.53</td>
</tr>
<tr>
<td>Rohan Panchal</td>
<td>489-22-1100</td>
<td>376-9821</td>
<td>265 Lark Lane</td>
<td>749-6492</td>
<td>28</td>
<td>3.93</td>
</tr>
<tr>
<td>Barbara Benson</td>
<td>533-69-1238</td>
<td>839-8461</td>
<td>7384 Fontana Lane</td>
<td>NULL</td>
<td>19</td>
<td>3.25</td>
</tr>
</tbody>
</table>

**Tuples:**
- Benjamin Bayer
- Chung-cha Kim
- Dick Davidson
- Rohan Panchal
- Barbara Benson
## Example Schema

### EMPLOYEE
<table>
<thead>
<tr>
<th>Fname</th>
<th>Minit</th>
<th>Lname</th>
<th>Ssn</th>
<th>Bdate</th>
<th>Address</th>
<th>Sex</th>
<th>Salary</th>
<th>Super_ssn</th>
<th>Dno</th>
</tr>
</thead>
</table>

### DEPARTMENT
<table>
<thead>
<tr>
<th>Dname</th>
<th>Dnumber</th>
<th>Mgr_ssn</th>
<th>Mgr_start_date</th>
</tr>
</thead>
</table>

### DEPT_LOCATIONS
<table>
<thead>
<tr>
<th>Dnumber</th>
<th>Dlocation</th>
</tr>
</thead>
</table>

### PROJECT
<table>
<thead>
<tr>
<th>Pname</th>
<th>Pnumber</th>
<th>Plocation</th>
<th>Dnum</th>
</tr>
</thead>
</table>

### WORKS_ON
<table>
<thead>
<tr>
<th>Essn</th>
<th>Pno</th>
<th>Hours</th>
</tr>
</thead>
</table>

### DEPENDENT
<table>
<thead>
<tr>
<th>Essn</th>
<th>Dependent_name</th>
<th>Sex</th>
<th>Bdate</th>
<th>Relationship</th>
</tr>
</thead>
</table>
### EMPLOYEE

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

### DEPARTMENT

<table>
<thead>
<tr>
<th>Dname</th>
<th>Dnumber</th>
<th>Mgr_ssn</th>
<th>Mgr_start_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
<td>1988-05-22</td>
</tr>
<tr>
<td>Administration</td>
<td>4</td>
<td>987654321</td>
<td>1995-01-01</td>
</tr>
<tr>
<td>Headquarters</td>
<td>1</td>
<td>888665555</td>
<td>1981-06-19</td>
</tr>
</tbody>
</table>

### DEPT_LOCATIONS

<table>
<thead>
<tr>
<th>Dnumber</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Houston</td>
</tr>
<tr>
<td>4</td>
<td>Stafford</td>
</tr>
<tr>
<td>5</td>
<td>Bellaire</td>
</tr>
<tr>
<td>5</td>
<td>Sugarland</td>
</tr>
<tr>
<td>5</td>
<td>Houston</td>
</tr>
</tbody>
</table>

### WORKS_ON

<table>
<thead>
<tr>
<th>Essn</th>
<th>Pno</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>1</td>
<td>32.5</td>
</tr>
<tr>
<td>123456789</td>
<td>2</td>
<td>7.5</td>
</tr>
<tr>
<td>666884444</td>
<td>3</td>
<td>40.0</td>
</tr>
<tr>
<td>453453453</td>
<td>1</td>
<td>20.0</td>
</tr>
<tr>
<td>453453453</td>
<td>2</td>
<td>20.0</td>
</tr>
<tr>
<td>333445555</td>
<td>2</td>
<td>10.0</td>
</tr>
<tr>
<td>333445555</td>
<td>2</td>
<td>10.0</td>
</tr>
<tr>
<td>333445555</td>
<td>3</td>
<td>10.0</td>
</tr>
<tr>
<td>333445555</td>
<td>10</td>
<td>10.0</td>
</tr>
<tr>
<td>999887777</td>
<td>10</td>
<td>30.0</td>
</tr>
<tr>
<td>999887777</td>
<td>10</td>
<td>10.0</td>
</tr>
<tr>
<td>987987987</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>987987987</td>
<td>30</td>
<td>5.0</td>
</tr>
<tr>
<td>987654321</td>
<td>30</td>
<td>20.0</td>
</tr>
<tr>
<td>987654321</td>
<td>20</td>
<td>15.0</td>
</tr>
<tr>
<td>888665555</td>
<td>20</td>
<td>NULL</td>
</tr>
</tbody>
</table>

### PROJECT

<table>
<thead>
<tr>
<th>Pname</th>
<th>Pnumber</th>
<th>Location</th>
<th>Dnum</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductX</td>
<td>1</td>
<td>Bellaire</td>
<td>5</td>
</tr>
<tr>
<td>ProductY</td>
<td>2</td>
<td>Sugarland</td>
<td>5</td>
</tr>
<tr>
<td>ProductZ</td>
<td>3</td>
<td>Houston</td>
<td>5</td>
</tr>
<tr>
<td>Computerization</td>
<td>10</td>
<td>Stafford</td>
<td>4</td>
</tr>
<tr>
<td>Reorganization</td>
<td>20</td>
<td>Houston</td>
<td>1</td>
</tr>
<tr>
<td>Newbenefits</td>
<td>30</td>
<td>Stafford</td>
<td>4</td>
</tr>
</tbody>
</table>

### DEPENDENT

<table>
<thead>
<tr>
<th>Essn</th>
<th>Dependent_name</th>
<th>Sex</th>
<th>Bdate</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>333445555</td>
<td>Alice</td>
<td>F</td>
<td>1986-04-05</td>
<td>Daughter</td>
</tr>
<tr>
<td>333445555</td>
<td>Theodore</td>
<td>M</td>
<td>1983-10-25</td>
<td>Son</td>
</tr>
<tr>
<td>333445555</td>
<td>Joy</td>
<td>F</td>
<td>1958-05-03</td>
<td>Spouse</td>
</tr>
<tr>
<td>987654321</td>
<td>Abner</td>
<td>M</td>
<td>1942-02-28</td>
<td>Spouse</td>
</tr>
<tr>
<td>123456789</td>
<td>Michael</td>
<td>M</td>
<td>1988-01-04</td>
<td>Son</td>
</tr>
<tr>
<td>123456789</td>
<td>Alice</td>
<td>F</td>
<td>1988-12-30</td>
<td>Daughter</td>
</tr>
<tr>
<td>123456789</td>
<td>Elizabeth</td>
<td>F</td>
<td>1967-05-05</td>
<td>Spouse</td>
</tr>
</tbody>
</table>
Constraints
Constraints

- Constraints are restrictions on legal relation states
  - they add further semantics to the schema

- Domain constraints $v_i \in \text{dom}(A_i)$
  - values for an attribute must be from the domain associated with the attribute

- Non-null constraints
  - the domain of some attributes may not include null, implying that a value for that attribute is required for all tuples
Key Constraints

- By definition, all tuples in a relation are unique
- Often, we want to restrict tuples further such that some subset of the attributes is unique for all tuples
- Example: in the StockItem relation, no ItemID should appear in more than one tuple
  - ItemID is called a key attribute
Any subset of attributes that must be unique is called a superkey.
If no subset of the attributes of a superkey must also be unique, then that superkey is called a key.

Example:

VEHICLE(LicenseNumber, SerialNumber, Model, Year)
Candidate and Primary Keys

- If a relation has more than one key, each key is called a *candidate key*

- One candidate key must be chosen to be the *primary key*

- The primary key is the one that will be used to *identify* tuples

- If there is only one key, it is the primary key
Primary keys are indicated by underlining the attributes that make up that key.

Candidate key

**VEHICLE**(*LicenseNumber*, *VIN*, *Model*, *Year*)

Primary key
Example Keys

STOCKITEM( ItemId, Description, Price, Taxable )

superkeys:
   (ItemId), (Description), (ItemId, Description)

keys:
   (ItemId), (Description)

candidate keys:
   (ItemId), (Description)

primary key:
   (ItemId)

(assumes that Description is unique for all items)
Integrity Constraints

- **Entity integrity constraint**
  - no primary key value can be null
  - the primary key is the tuple identifier

- **Referential integrity constraint**
  - references between relations must be valid
  - the *foreign key* of a referencing relation must exist as a primary key in the referenced relation
Example: Referential Integrity

STOCKITEM( ItemId, Description, Price, Taxable )
STORERESTOCK( StoreId, Item, Quantity )

STORERESTOCK[Item] refers to STOCKITEM[ItemID]

STORERESTOCK[Item] is a foreign key referencing the primary key STOCKITEM[ItemID]

Any value appearing in STORERESTOCK[Item] must appear in STOCKITEM[ItemID]

It must be true that
\[ \text{dom(STORERESTOCK[Item])} = \text{dom(STOCKITEM[ItemID])} \]
Referential Integrity

- PK = primary key in $R_2$
- FK = foreign key in $R_1$
- $\text{dom}(R_1[\text{FK}]) = \text{dom}(R_2[\text{PK}])$
- constraint:
  
  \[
  \text{if } v \in R_1[\text{FK}] \text{ then } v \in R_2[\text{PK}]
  \]

- note: FK is not necessarily a key of $R_1$
Example: Referential Integrity

| STOCKITEM( ItemId, Description, Price, Taxable ) |
| STORESTOCK( StoreId, Item, Quantity ) |
| STORE( StoreID, Manager, Address, Phone ) |

- (StoreId, Item) is the primary key of STORESTOCK
- STORESTOCK[StoreId] is a foreign key referencing STORE
- STORESTOCK[Item] is a foreign key referencing STOCKITEM
Referential Integrity: Diagrammatic Representation

STOCKITEM( ItemId, Description, Price, Taxable )

STORESTOCK( StoreId, Item, Quantity )

STORE( StoreID, Manager, Address, Phone )
Referential Integrity: Textual Representation

STOCKITEM( ItemId, Description, Price, Taxable )
STORESTOCK( StoreId, Item, Quantity )
STORE( StoreID, Manager, Address, Phone )

constraints:
STORESTOCK[StoreId] refers to STORE[StoreID]
STORESTOCK[Item] refers to STOCKITEM[ItemId]
Referential Integrity: Example State

\[ r(\text{STORESTOCK}) = \]
\[ \{ \langle \text{"S002"}, \text{"I065"}, 120 \rangle, \]
\[ \langle \text{"S047"}, \text{"I954"}, 300 \rangle, \]
\[ \langle \text{"S002"}, \text{"I954"}, 198 \rangle \} \]

StoreId is a foreign key but not a key

\[ r(\text{STORE}) = \]
\[ \{ \langle \text{"S002"}, \text{"Tom"}, \text{"112 Main"}, \text{"999-8888"} \rangle, \]
\[ \langle \text{"S047"}, \text{"Sasha"}, \text{"13 Pine"}, \text{"777-6543"} \rangle \} \]

all values in FK exist in PK
Referential Integrity: Constraint Violation

\[ r(\text{STORESTOCK}) = \begin{cases} < "S002", "I065", 120 >, \\ < "S047", "I954", 300 >, \\ < "S333", "I954", 198 > \end{cases} \]

StoreId “S333” does not exist in PK:
this is an illegal database state

\[ r(\text{STORE}) = \begin{cases} < "S002", "Tom", "112 Main", "999-8888" >, \\ < "S047", "Sasha", "13 Pine", "777-6543" > \end{cases} \]

Both relation states are legal, but the database state is illegal.
### Schema with FKs

**EMPLOYEE**

<table>
<thead>
<tr>
<th>Fname</th>
<th>Minit</th>
<th>Lname</th>
<th>Ssn</th>
<th>Bdate</th>
<th>Address</th>
<th>Sex</th>
<th>Salary</th>
<th>Super_ssn</th>
<th>Dno</th>
</tr>
</thead>
</table>

**DEPARTMENT**

<table>
<thead>
<tr>
<th>Dname</th>
<th>Dnumber</th>
<th>Mgr_ssn</th>
<th>Mgr_start_date</th>
</tr>
</thead>
</table>

**DEPT_LOCATIONS**

<table>
<thead>
<tr>
<th>Dnumber</th>
<th>Dlocation</th>
</tr>
</thead>
</table>

**PROJECT**

<table>
<thead>
<tr>
<th>Pname</th>
<th>Pnumber</th>
<th>Plocation</th>
<th>Dnum</th>
</tr>
</thead>
</table>

**WORKS_ON**

<table>
<thead>
<tr>
<th>Essn</th>
<th>Pno</th>
<th>Hours</th>
</tr>
</thead>
</table>

**DEPENDENT**

<table>
<thead>
<tr>
<th>Essn</th>
<th>Dependent_name</th>
<th>Sex</th>
<th>Bdate</th>
<th>Relationship</th>
</tr>
</thead>
</table>

State Change and Constraint Enforcement
Causes of Constraint Violations

- What can cause a referential integrity constraint violation?
  - inserting a tuple in \( R_1 \) with an illegal FK
  - modifying a tuple in \( R_1 \) to have an illegal FK
  - deleting a tuple in \( R_2 \) that had the PK referenced by some FK in \( R_1 \)

- How can a referential integrity constraint be enforced?
  - reject the operation that attempts to violate it (may cause other operations to be rejected ... transactions)
  - repair the violation, by cascading inserts or deletes
Data Manipulation Operations

There are three ways to modify the value of a relation:

- **Insert:** add a new tuple to \( R \)
- **Delete:** remove an existing tuple from \( R \)
- **Update:** change the value of an existing tuple in \( R \)

Delete and Update both require some way to identify an existing tuple (a selection)
Inserting Tuples

\[ r_1(\text{STORESTOCK}) = \{ \langle "S002", "l065", 120 \rangle, \langle "S047", "l954", 300 \rangle, \langle "S333", "l954", 198 \rangle \} \]

\[ \text{insert} \langle "S047", "l099", 267 \rangle \]

\[ r_2(\text{STORESTOCK}) = \{ \langle "S002", "l065", 120 \rangle, \langle "S333", "l954", 198 \rangle, \langle "S047", "l099", 267 \rangle, \langle "S047", "l954", 300 \rangle \} \]

any constraint violations?
Deleting Tuples


delete tuples with Item = "I954"

\[ r_3(\text{STORESTOCK}) = \{ < "S002", "I065", 120 >, < "S047", "I099", 267 > \} \]
Updating Tuples

\[ r_3(\text{STORESTOCK}) = \left\{ \begin{array}{c}
< "S002", "I065", 120 >, \\
< "S047", "I099", 267 >
\end{array} \right\} \]

change the **Quantity** of tuples with **StoreID** = "S002" and **Item** = "I954" to 250

\[ r_3(\text{STORESTOCK}) = \left\{ \begin{array}{c}
< "S002", "I065", 250 >, \\
< "S047", "I099", 267 >
\end{array} \right\} \]
Any update can be viewed as (delete and insert)

update: <"S002", "I065", 120> to <"S002", "I065", 250> is equivalent to
delete: <"S002", "I065", 120>
insert: <"S002", "I065", 250>

Any database state change can be viewed as a set of deletes and inserts on individual relations

This makes the analysis of potential constraint violations a well defined problem
Enforcing Constraints

- *constraint enforcement*: ensuring that no invalid database states can exist

- *invalid state*: a state in which a constraint is violated

- Possible ways to enforce constraints:
  - reject any operation that causes a violation, or
  - allow the violating operation and then attempt to correct the database
To automate constraint enforcement, the operations that can cause violations need to be identified.

<table>
<thead>
<tr>
<th></th>
<th>insert</th>
<th>delete</th>
<th>update</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain, non-null</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>key</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>entity integrity</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>referential integrity</td>
<td>yes/FK</td>
<td>yes/PK</td>
<td>yes/FK/PK</td>
</tr>
</tbody>
</table>
## Correcting Constraint Violations

<table>
<thead>
<tr>
<th>Violation</th>
<th>Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain, non-null</td>
<td>ask user to enter a valid value or use a default value</td>
</tr>
<tr>
<td>key</td>
<td>ask user to enter a unique key or generate a unique key</td>
</tr>
<tr>
<td>entity integrity</td>
<td>ask user to enter a unique key or generate a unique key</td>
</tr>
<tr>
<td>referential integrity FK insertion</td>
<td>force an insert in the PK (may cascade)</td>
</tr>
<tr>
<td>referential integrity PK deletion</td>
<td>propagate delete to FK (may cascade)</td>
</tr>
</tbody>
</table>
Summary: Relational Schemas

- A relational schema consists of a set of relation schemas and a set of constraints.

- Relation schema
  - list of attributes: name and domain constraint
  - superkeys: key constraints
  - primary key: entity integrity constraint

- Foreign keys: referential integrity constraints
  - defined between relation schemas
<table>
<thead>
<tr>
<th>AIRPORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airport_code</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FLIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight_number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FLIGHT_LEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight_number</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LEG_INSTANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight_number</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight_number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AIRPLANE_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airplane_type_name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CAN_LAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airplane_type_name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AIRPLANE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airplane_id</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEAT_RESERVATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight_number</td>
</tr>
</tbody>
</table>

Schema for Airline Database
NEXT UP

- skip ahead to Chapter 7:
  Translating ER Schemas to Relational Schemas

- then back to Chapter 6:
  The Relational Algebra: operations on relations
PREVIEW: ER to Relational