Database Systems

Session 2 – Main Theme
Relational Data Model
& Relational Database Constraints
Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Presentation material partially based on textbook slides
by Ramez Elmasri and Shamkant Navathe
Slides copyright © 2011

Agenda

1 Session Overview
2 Relational Data Model & Database Constraints
3 Summary and Conclusion
Session Agenda

- Session Overview
- Relational Data Model & Database Constraints
- Summary & Conclusion

What is the class about?

- Course description and syllabus:
  - [http://www.nyu.edu/classes/jcf/CSCI-GA.2433-001](http://www.nyu.edu/classes/jcf/CSCI-GA.2433-001)

- Textbooks:
    - Ramez Elmasri and Shamkant Navathe
    - Addison Wesley
Agenda

- The need for database management systems
- Brief overview of the relational model
- Querying relational database directly and through views
- Need for good logical design
- Need for good physical design
- Recovery
- Concurrency
- Layers of database management systems
- Independence between/among layers
- Various roles of designers, users, and maintainers

Two Main Functions Of Databases

- A very large fraction of computer use is devoted to business processing of data using databases
  - Think about what Amazon has to do to manage its operations
- Two main uses of databases
  - **OLTP** (Online Transaction Processing)
    - The database is used for entering, modifying, and querying data
    - Correctness, at least for entering and modifying data must be assured
    - Example: Amazon charges customer's credit card for the price of the book that a customer ordered
  - **OLAP** (Online Analytical Processing)
    - The database is used for business intelligence, including data mining
    - The results do not have to be "completely correct," as this may be too inefficient to guarantee, but complex queries have to be answered (relatively) fast
    - Example: Amazon wants to know how many books that cost less than $10 each were sold in New Jersey during December 2009
Managing The Data Of An Enterprise

- We may consider some enterprise (organization) and the totality of the information it maintains.
- We think about managing this information, focusing on OLTP.
- Ideally, the information should be stored in a (logically) single (possibly physically distributed) database system.
- We start with a very simple example to introduce some concepts and issues to address.
- We look at only a very small part of information of the type that an enterprise may need to keep.
- We need some way of describing sample data.
- We will think, in this unit, of the database as *a set of tables*, each stored as a *file on a disk*.

---

Agenda

- The need for database management systems
- Brief overview of the relational model
  - Querying relational database directly and through views
  - Need for good logical design
  - Need for good physical design
  - Recovery
  - Concurrency
- Layers of database management systems
  - Independence between/among layers
  - Various roles of designers, users, and maintainers
The Relational Data Model

- Relational model
  - First commercial implementations available in early 1980s
  - Has been implemented in a large number of commercial system
- Hierarchical and network models
  - Preceded the relational model

Relational Model Concepts

- Represents data as a collection of relations
- Table of values
  - Row
    - Represents a collection of related data values
    - Fact that typically corresponds to a real-world entity or relationship
    - Tuple
  - Table name and column names
    - Interpret the meaning of the values in each row attribute
Domains, Attributes, Tuples, and Relations

- Domain D
  - Set of atomic values
- Atomic
  - Each value indivisible
- Specifying a domain
  - Data type specified for each domain
Relation schema $R$
- Denoted by $R(A_1, A_2, ..., A_n)$
- Made up of a relation name $R$ and a list of attributes, $A_1, A_2, ..., A_n$

Attribute $A_i$
- Name of a role played by some domain $D$ in the relation schema $R$

Degree (or arity) of a relation
- Number of attributes $n$ of its relation schema

Relation (or relation state)
- Set of $n$-tuples $r = \{t_1, t_2, ..., t_m\}$
- Each $n$-tuple $t$
  - Ordered list of $n$ values $t = \langle v_1, v_2, ..., v_n \rangle$
  - Each value $v_i, 1 \leq i \leq n$, is an element of $\text{dom}(A_i)$ or is a special NULL value
Relation (or relation state) $r(R)$
- Mathematical relation of degree $n$ on the domains $\text{dom}(A_1), \text{dom}(A_2), \ldots, \text{dom}(A_n)$
- Subset of the Cartesian product of the domains that define $R$:
  - $r(R) = (\text{dom}(A_1) \times \text{dom}(A_2) \times \ldots \times \text{dom}(A_n))$

Cardinality
- Total number of values in domain

Current relation state
- Relation state at a given time
- Reflects only the valid tuples that represent a particular state of the real world

Attribute names
- Indicate different roles, or interpretations, for the domain
### Characteristics of Relations

- Ordering of tuples in a relation
  - Relation defined as a set of tuples
  - Elements have no order among them
- Ordering of values within a tuple and an alternative definition of a relation
  - Order of attributes and values is not that important
  - As long as correspondence between attributes and values maintained

### Characteristics of Relations (cont'd.)

- Alternative definition of a relation
  - Tuple considered as a set of \((\text{attribute}, \text{value})\) pairs
  - Each pair gives the value of the mapping from an attribute \(A_i\) to a value \(v_i\) from \(\text{dom}(A_i)\)
- Use the first definition of relation
  - Attributes and the values within tuples are ordered
  - Simpler notation
Values and NULLs in tuples
- Each value in a tuple is atomic
- Flat relational model
  - Composite and multivalued attributes not allowed
  - First normal form assumption
- Multivalued attributes
  - Must be represented by separate relations
- Composite attributes
  - Represented only by simple component attributes in basic relational model
NULL values
- Represent the values of attributes that may be unknown or may not apply to a tuple
- Meanings for NULL values
  - Value unknown
  - Value exists but is not available
  - Attribute does not apply to this tuple (also known as value undefined)

Interpretation (meaning) of a relation
- Assertion
  - Each tuple in the relation is a fact or a particular instance of the assertion
- Predicate
  - Values in each tuple interpreted as values that satisfy predicate
Relational Model Notation

- Relation schema $R$ of degree $n$
  - Denoted by $R(A_1, A_2, ..., A_n)$
- Uppercase letters $Q$, $R$, $S$
  - Denote relation names
- Lowercase letters $q$, $r$, $s$
  - Denote relation states
- Letters $t$, $u$, $v$
  - Denote tuples

Name of a relation schema: STUDENT
- Indicates the current set of tuples in that relation
- Notation: STUDENT(Name, Ssn, ...)
  - Refers only to relation schema
- Attribute $A$ can be qualified with the relation name $R$ to which it belongs
  - Using the dot notation $R.A$
Relational Model Notation

- **n-tuple** $t$ in a relation $r(R)$
  - Denoted by $t = <v_1, v_2, ..., v_n>$
  - $v_i$ is the value corresponding to attribute $A_i$
- Component values of tuples:
  - $t[A_i]$ and $t.A_i$ refer to the value $v_i$ in $t$ for attribute $A_i$
  - $t[A_{u}, A_{w}, ..., A_{z}]$ and $t.(A_{u}, A_{w}, ..., A_{z})$ refer to the subtuple of values $<v_u, v_w, ..., v_z>$ from $t$ corresponding to the attributes specified in the list

A Sample Relational Database

<table>
<thead>
<tr>
<th>SSN</th>
<th>City</th>
<th>DOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Boston</td>
<td>3498</td>
</tr>
<tr>
<td>106</td>
<td>London</td>
<td>2987</td>
</tr>
<tr>
<td>121</td>
<td>Bosto n</td>
<td>2367</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>Book</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>132</td>
<td>Plants</td>
<td>8976</td>
</tr>
<tr>
<td>121</td>
<td>Animals</td>
<td>9003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>DOB</th>
<th>Grade</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>121</td>
<td>2367</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>A</td>
<td>132</td>
<td>3678</td>
<td>3</td>
<td>70</td>
</tr>
<tr>
<td>B</td>
<td>101</td>
<td>3498</td>
<td>4</td>
<td>70</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>Illness</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Cold</td>
<td>3498</td>
</tr>
<tr>
<td>121</td>
<td>Flu</td>
<td>2987</td>
</tr>
</tbody>
</table>

- Of course, the values do not pretend to be real, they were chosen to be short, so can be easily fitted on the slide
- The database talks about employees, books they have checked out from the library (and when), and various illnesses they have had (and when)
Agenda

- The need for database management systems
- Brief overview of the relational model
- Querying relational database directly and through views
  - Need for good logical design
  - Need for good physical design
  - Recovery
  - Concurrency
- Layers of database management systems
- Independence between/among layers
- Various roles of designers, users, and maintainers

Some Typical Queries

- Some typical queries
  - Give Name of every employee born before 3500
  - Give Name and City for every employee who took out a Book after 9000
  - Prepare a recall notice to for every employee who had a flu to come for a checkup

  *Note that some queries involve a single table, and some involve several tables*

- We would like to have a convenient language, as close as possible to a natural language, to express these queries, and similar ones, thinking of tables, not of lower-level structures (files)

- Some languages
  - SQL (used to be called Structured Query Language): every relational database supports some “close to standard” version
  - QBE (Query By Example); underlying, e.g., Microsoft Access’s GUI
Two Queries in SQL

- Imagine that the tables are have names (as they of course do in SQL)
  - Table1: with columns SSN, City, DOB
  - Table2: with columns Name, SSN, DOB, Grade, Salary
  - Table3: with columns SSN, Book, Date
  - Table4: with columns SSN, Illness, date
- Give Name of every employee born before 3500
  
  ```sql
  SELECT Name
  FROM Table2
  WHERE DOB < 3500;
  ```
- Give Name and City for every employee who took out a Book after 9000
  
  ```sql
  SELECT Name, City
  FROM Table2, Table1
  WHERE Table2.SSN = Table1.SSN;
  ```

Agenda

- The need for database management systems
- Brief overview of the relational model
- Querying relational database directly and through views
- Need for good logical design
- Need for good physical design
- Recovery
- Concurrency
- Layers of database management systems
- Independence between/among layers
- Various roles of designers, users, and maintainers
The Need For Good Design

- It is important also to think carefully about the correct (or just good!) choice of which tables to use and what should be their structure.
- This we should do in order to have good logical design, not worrying (yet) about efficient storage in files.
- Our initial design suffers (for pedagogical reasons) from various problems, which we will see next.

Redundancy

- A data item appears more than once **unnecessarily**
  - Assuming that each SSN has only one DOB, DOB appears twice unnecessarily (in two different tables).
  - There is a danger that this will be inconsistent.
  - Even more dangerous would have been multiple storage of employee’s City.
  - If the employee moves, the City must be changed everywhere it appears.
- Note, however, that from an efficiency point of view, it might be useful to replicate information, to speed up access.
  - In our example, if frequently we want to correlate DOB with Grade and also DOB with City, it may be good to have it in both tables, and not insist on a “clean” design.
- Note that **it was necessary** for SSN to appear in two different tables, as otherwise we could not “assemble” information about employees.
Relational Model Constraints

- Constraints
  - Restrictions on the actual values in a database state
  - Derived from the rules in the miniworld that the database represents
- Inherent model-based constraints or implicit constraints
  - Inherent in the data model

Relational Model Constraints (cont’d.)

- Schema-based constraints or explicit constraints
  - Can be directly expressed in schemas of the data model
- Application-based or semantic constraints or business rules
  - Cannot be directly expressed in schemas
  - Expressed and enforced by application program
Domain Constraints

- Typically include:
  - Numeric data types for integers and real numbers
  - Characters
  - Booleans
  - Fixed-length strings
  - Variable-length strings
  - Date, time, timestamp
  - Money
  - Other special data types

Key Constraints and Constraints on NULL Values

- No two tuples can have the same combination of values for all their attributes.
- Superkey
  - No two distinct tuples in any state $r$ of $R$ can have the same value for SK
- Key
  - Superkey of $R$
  - Removing any attribute $A$ from $K$ leaves a set of attributes $K$ that is not a superkey of $R$ any more
Key Constraints and Constraints on NULL Values (cont'd.)

- Key satisfies two properties:
  - Two distinct tuples in any state of relation cannot have identical values for (all) attributes in key
  - Minimal superkey
    - Cannot remove any attributes and still have uniqueness constraint in above condition hold

Key Constraints and Constraints on NULL Values (cont’d.)

- Candidate key
  - Relation schema may have more than one key
- Primary key of the relation
  - Designated among candidate keys
  - Underline attribute
- Other candidate keys are designated as unique keys
Relational Databases and Relational Database Schemas

- Relational database schema $S$
  - Set of relation schemas $S = \{R_1, R_2, \ldots, R_m\}$
  - Set of integrity constraints $IC$
- Relational database state
  - Set of relation states $DB = \{r_1, r_2, \ldots, r_m\}$
  - Each $r_i$ is a state of $R_i$ and such that the $r_i$ relation states satisfy integrity constraints specified in $IC$
Invalid state
- Does not obey all the integrity constraints

Valid state
- Satisfies all the constraints in the defined set of integrity constraints IC

Entity integrity constraint
- No primary key value can be NULL

Referential integrity constraint
- Specified between two relations
- Maintains consistency among tuples in two relations
Integrity, Referential Integrity, and Foreign Keys (cont’d.)

- Foreign key rules:
  - The attributes in FK have the same domain(s) as the primary key attributes PK
  - Value of FK in a tuple $t_1$ of the current state $r_1(R_1)$ either occurs as a value of PK for some tuple $t_2$ in the current state $r_2(R_2)$ or is NULL

Integrity, Referential Integrity, and Foreign Keys (cont’d.)

- Diagrammatically display referential integrity constraints
  - Directed arc from each foreign key to the relation it references
  - All integrity constraints should be specified on relational database schema
Other Types of Constraints

- Semantic integrity constraints
  - May have to be specified and enforced on a relational database
  - Use triggers and assertions
  - More common to check for these types of constraints within the application programs

Other Types of Constraints (cont’d.)

- Functional dependency constraint
  - Establishes a functional relationship among two sets of attributes $X$ and $Y$
  - Value of $X$ determines a unique value of $Y$

- State constraints
  - Define the constraints that a valid state of the database must satisfy

- Transition constraints
  - Define to deal with state changes in the database
Storage Of Constraints (aka., “Business Rules”)

- Assume that it is the policy of our enterprise that the value of Salary is determined only by the value of Grade; this is an example of a business rule
  - Thus the fact that the Grade = 2 implies Salary = 80 is written twice in the database
  - This is another type of redundancy, which is less obvious at first
- There are additional problems with this design.
  - We are unable to store the salary structure for a Grade that does not currently exist for any employee.
  - For example, we cannot store that Grade = 1 implies Salary = 90
  - For example, if employee with SSN = 132 leaves, we forget which Salary should be paid to employee with Grade = 3
  - We could perhaps invent a fake employee with such a Grade and such a Salary, but this brings up additional problems, e.g., What is the SSN of such a fake employee?
- Note that our constraints specify a pay scale, which is independent of a particular employee

Handling Storage Of Constraints

- The problem can be solved by replacing

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>DOB</th>
<th>Grade</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>121</td>
<td>2367</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>A</td>
<td>132</td>
<td>3678</td>
<td>3</td>
<td>70</td>
</tr>
<tr>
<td>B</td>
<td>101</td>
<td>3498</td>
<td>4</td>
<td>70</td>
</tr>
<tr>
<td>C</td>
<td>106</td>
<td>2987</td>
<td>2</td>
<td>80</td>
</tr>
</tbody>
</table>

with two tables

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>DOB</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>121</td>
<td>2367</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>132</td>
<td>3678</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>101</td>
<td>3498</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>106</td>
<td>2987</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grad</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>3</td>
<td>70</td>
</tr>
<tr>
<td>4</td>
<td>70</td>
</tr>
</tbody>
</table>
Handling Storage Of Constraints

- And now we can store information more naturally
  - We can specify that Grade 3 implies Salary 70, even after the only employee with this Grade, i.e., employee with SSN 132 left the enterprise
  - We can specify that Grade 1 (a new Grade just established) implies Salary 90, even before any employee with this grade is higher

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>DOB</th>
<th>Grade</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>121</td>
<td>2367</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>B</td>
<td>101</td>
<td>3498</td>
<td>4</td>
<td>70</td>
</tr>
<tr>
<td>C</td>
<td>106</td>
<td>2987</td>
<td>2</td>
<td>80</td>
</tr>
</tbody>
</table>

Clean Design Versus Efficiency

- However, if the correlation between an employee and salary is needed frequently, e.g., for producing payroll, it may be inefficient to recompute this correlation repeatedly.
- So, returning to our original instance of the database, perhaps we should have (despite some redundancy) both the original table and the table associating salaries with grades

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>DOB</th>
<th>Grade</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>121</td>
<td>2367</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>A</td>
<td>132</td>
<td>3678</td>
<td>3</td>
<td>70</td>
</tr>
<tr>
<td>B</td>
<td>101</td>
<td>3498</td>
<td>4</td>
<td>70</td>
</tr>
<tr>
<td>C</td>
<td>106</td>
<td>2987</td>
<td>2</td>
<td>80</td>
</tr>
</tbody>
</table>
One More Problem

- What if it becomes illegal to use social security numbers for anything other than payroll related matters?
- We will have an incredible mess and enormous amount of work to restructure the database, unless we have designed the application appropriately to begin with.
- Of course we did not know that it would become illegal to use social security numbers and it was convenient to do so, so that’s what we used.
- So how to be able to anticipate potential problems?
- NYU had to spend considerable effort to switch from social security numbers to University ID's.
- We will discuss how to “anticipate” such problems, so such switching is painless.

Update Operations, Transactions, and Dealing with Constraint Violations

- Operations of the relational model can be categorized into retrievals and updates.
- Basic operations that change the states of relations in the database:
  - Insert
  - Delete
  - Update (or Modify)
Sample Database State

Figure 3.6
One possible database state for the COMPANY relational database schema.

### EMPLOYEE

<table>
<thead>
<tr>
<th>Name</th>
<th>Prefix</th>
<th>Lname</th>
<th>Sex</th>
<th>Bdate</th>
<th>Address</th>
<th>Salary</th>
<th>Super sire</th>
<th>Dno</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>B</td>
<td>Smith</td>
<td>M</td>
<td>1965-01-09</td>
<td>731 Fendren, Houston, TX</td>
<td>30000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Franklin</td>
<td>T</td>
<td>Wing</td>
<td>M</td>
<td>1955-12-08</td>
<td>638 Vaux, Houston, TX</td>
<td>40000</td>
<td>888666555</td>
<td>5</td>
</tr>
<tr>
<td>Akira</td>
<td>J</td>
<td>Zolnya</td>
<td>M</td>
<td>1968-01-19</td>
<td>9212 Castle, Spring, TX</td>
<td>25000</td>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>Jennifer</td>
<td>S</td>
<td>Wallace</td>
<td>F</td>
<td>1961-06-20</td>
<td>291 Barry, Bellaire, TX</td>
<td>40000</td>
<td>888666555</td>
<td>4</td>
</tr>
<tr>
<td>Ramesh</td>
<td>K</td>
<td>Nairyan</td>
<td>M</td>
<td>1962-09-15</td>
<td>275 Fire Oak, Humble, TX</td>
<td>80000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Joyce</td>
<td>A</td>
<td>English</td>
<td>F</td>
<td>1972-07-01</td>
<td>5631 Rice, Houston, TX</td>
<td>50000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Ahmad</td>
<td>V</td>
<td>Jabbar</td>
<td>M</td>
<td>1966-03-29</td>
<td>930 Dallas, Houston, TX</td>
<td>25000</td>
<td>888666555</td>
<td>1</td>
</tr>
<tr>
<td>James</td>
<td>E</td>
<td>Borg</td>
<td>M</td>
<td>1981-06-19</td>
<td>450 Stone, Houston, TX</td>
<td>NULL</td>
<td>NULL</td>
<td>1</td>
</tr>
</tbody>
</table>

### DEPARTMENT

<table>
<thead>
<tr>
<th>Name</th>
<th>Dnumber</th>
<th>Mgr sire</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>888666555</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>

Sample Database State (cont'd)

Figure 3.6
One possible database state for the COMPANY relational database schema.

### WORKS_ON

<table>
<thead>
<tr>
<th>Loan</th>
<th>Proj</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>668884444</td>
<td>3</td>
<td>40.0</td>
</tr>
<tr>
<td>453453453</td>
<td>4</td>
<td>20.0</td>
</tr>
<tr>
<td>333445555</td>
<td>5</td>
<td>10.0</td>
</tr>
<tr>
<td>333445555</td>
<td>6</td>
<td>10.0</td>
</tr>
<tr>
<td>998877777</td>
<td>7</td>
<td>30.0</td>
</tr>
<tr>
<td>998877777</td>
<td>8</td>
<td>10.0</td>
</tr>
<tr>
<td>987907987</td>
<td>9</td>
<td>30.5</td>
</tr>
<tr>
<td>987907987</td>
<td>10</td>
<td>20.0</td>
</tr>
<tr>
<td>987907987</td>
<td>11</td>
<td>15.0</td>
</tr>
<tr>
<td>888666555</td>
<td>12</td>
<td>NULL</td>
</tr>
</tbody>
</table>

### PROJECT

<table>
<thead>
<tr>
<th>Loan</th>
<th>Proj</th>
<th>Location</th>
<th>Dnumber</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>Loan</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>1988-04-05</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>1942-02-12</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>
The Insert Operation

- Provides a list of attribute values for a new tuple $t$ that is to be inserted into a relation $R$
- Can violate any of the four types of constraints
- If an insertion violates one or more constraints
  - Default option is to reject the insertion
The Delete Operation

- Can violate only referential integrity
  - If tuple being deleted is referenced by foreign keys from other tuples
- Restrict
  - Reject the deletion
- Cascade
  - Propagate the deletion by deleting tuples that reference the tuple that is being deleted
- Set null or set default
  - Modify the referencing attribute values that cause the violation

The Update Operation

- Necessary to specify a condition on attributes of relation
  - Select the tuple (or tuples) to be modified
- If attribute not part of a primary key nor of a foreign key
  - Usually causes no problems
- Updating a primary/foreign key
  - Similar issues as with Insert/Delete
Different Users Need Different Data

- It may be our goal to create a design that best reflects the inherent properties of the data.
  - But, various user groups may need to look at the data assuming different structure (organization) of the data.
- For privacy/security reasons we may want to give different users different access privileges to the database.
  - The payroll department can see salaries but cannot see diseases.
  - The health department can see diseases but cannot see salaries.
- Users may prefer to look at different aspects of the information.
  - The payroll department may prefer to see the salary in a different currency.
  - The health department may prefer to see Age instead of, or in addition to DOB.

Views

- A possible solution: give each user (class of users) privileges to look at a view, that is, a small derived database.
- The health department may think that there is a table:

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>City</th>
<th>DOB</th>
<th>Age</th>
<th>Illness</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>121</td>
<td>Portland</td>
<td>2367</td>
<td>47</td>
<td>Flu</td>
<td>2987</td>
</tr>
<tr>
<td>B</td>
<td>101</td>
<td>Boston</td>
<td>3498</td>
<td>25</td>
<td>Cold</td>
<td>3498</td>
</tr>
</tbody>
</table>

- The database should provide such a view, which is computed from the existing tables (and the current date), without the user knowing other (prohibited for this user) information.
- We need to leave flexibility for unanticipated queries.
  - Some people may later be given the right and want to ask the query: “How are salaries and diseases correlated?”
Manipulating Data Through Views

- The ideal goal is for the users to both query and modify the database through views
- Unfortunately, sometimes it impossible or difficult to do so
  - If the user wants to change the age of an employee, how should the change be reflected in the date of birth?
    - There is no unique way of doing it
  - How to change the sum of salaries, if some view contains this information?
    - We want to give a total raise of 5% (increase sum of salaries by 5%), so how to reflect this in individual salaries?
    - Some employees may get more than 5% and some may get less than 5%

Agenda

- The need for database management systems
- Brief overview of the relational model
- Querying relational database directly and through views
- Need for good logical design
- Need for good physical design
- Recovery
- Concurrency
- Layers of database management systems
- Independence between/among layers
- Various roles of designers, users, and maintainers
Physical Design

- The database system must be organized so that it is able to process queries efficiently
- To do this:
  - Files must be organized appropriately
  - Indices may be employed
- For example, if we frequently want to find the grade for various SSN, perhaps the file should be hashed on this value, allowing direct access
- But, if we want to print the salaries of all the employees born in 2783, maybe the file should be sorted by DOB
- Physical design of databases deals with such issues (including how to distribute information among various sites), which are also closely related to the optimization of query processing

Agenda

- The need for database management systems
- Brief overview of the relational model
- Querying relational database directly and through views
- Need for good logical design
- Need for good physical design
- Recovery
- Concurrency
- Layers of database management systems
- Independence between/among layers
- Various roles of designers, users, and maintainers
Recovery

- The database must be resilient even though the system is prone to faults.
- Assume one more table, describing employees' accounts in the credit union

<table>
<thead>
<tr>
<th>SSN</th>
<th>Savings</th>
<th>Checking</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>106</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>121</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>132</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

- We want to give each employee a bonus of 10 in the savings account.
  - To do that, a transaction (execution of a user program) will sequentially change the values of Savings

Example Of A Problem

- The file describing the table is stored on a disk, values are read into RAM, modified and written out
- If X is a local variable then we have a trace of the desired execution (in shorthand):
  
  X := Savings[101] \quad \text{read from disk}
  X := X + 10 \quad \text{process in RAM}
  \text{Savings[101] := X} \quad \text{write to disk}

- What if the system crashes in the middle, say power goes out
- We do not know which of the values have been changed, so what to do to recover (get back a correct state)?
- Various techniques exist for managing the execution, so that reliable execution is possible
Agenda

- The need for database management systems
- Brief overview of the relational model
- Querying relational database directly and through views
- Need for good logical design
- Need for good physical design
- Recovery
- Concurrency
- Layers of database management systems
- Independence between/among layers
- Various roles of designers, users, and maintainers

Concurrency

- There may also be problems because of the concurrent execution of several transactions in a time sharing system
- Assume that we are running a transaction, T1 (an “reporting” transaction), that should compute and print for each employee the sum of Savings and Checking:

<table>
<thead>
<tr>
<th>SSN</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>70</td>
</tr>
<tr>
<td>106</td>
<td>60</td>
</tr>
<tr>
<td>121</td>
<td>80</td>
</tr>
<tr>
<td>132</td>
<td>10</td>
</tr>
</tbody>
</table>

- Concurrently SSN = 121 wants to move 40 from Checking to Savings, using transaction T2 (a “moving” transaction)
- In a time-sharing system we could have an incorrect execution.
- We will write “CH” for Checking and “SA” for Savings
The Transaction Concept

- **Transaction**
  - Executing program
  - Includes some database operations
  - Must leave the database in a valid or consistent state

- **Online transaction processing (OLTP) systems**
  - Execute transactions at rates that reach several hundred per second

---

**Execution Trace Of Two Transactions**

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1 := SA[121] (X1 = 0)</td>
<td>Y1 := CH[121] (Y1 = 80)</td>
</tr>
<tr>
<td></td>
<td>Y1 := Y1 - 40 (Y1 = 40)</td>
</tr>
<tr>
<td></td>
<td>CH[121] := Y1</td>
</tr>
<tr>
<td>X2 := CH[121] (X2 = 40)</td>
<td></td>
</tr>
<tr>
<td>X1 := X1 + X2 (X1 = 40)</td>
<td></td>
</tr>
<tr>
<td>PRINT X1 (X1 = 40)</td>
<td>Y2 := SA[121] (Y2 = 0)</td>
</tr>
<tr>
<td></td>
<td>Y2 := Y2 + 40 (Y2 = 40)</td>
</tr>
<tr>
<td></td>
<td>SA[121] := Y2</td>
</tr>
</tbody>
</table>

- We get 40, an incorrect value of Balance for SSN = 121
- Standard operating system constructs do not help here, but concurrency control mechanisms that solve the problem exist in databases (but not in Microsoft Access)
Some Concurrency Aspects

- In the previous examples, we could allow the two transactions to interleave in this way, with the user of the "reporting" transaction being told that correct results are not guaranteed.
- The user may get only approximate result, which perhaps is sufficient if we are producing "statistical" reports.
- But the database will remain consistent (correct) and the "moving" transaction can execute.
- But if instead of the "reporting" transaction which only read the database, we have a "multiplying" transaction that updates all the values in the database by multiplying them by 2, then the database could be corrupted, and the interleaving cannot be permitted.

Agenda

- The need for database management systems
- Brief overview of the relational model
- Querying relational database directly and through views
- Need for good logical design
- Need for good physical design
- Recovery
- Concurrency

- Layers of database management systems
- Independence between/among layers
- Various roles of designers, users, and maintainers
The Layers/Levels Of The Ideal Database

- It is customary to think of the database as made of several layers or levels, which are not completely standardized.
- Different levels have different roles.
- We will think of 4 levels:
  - **External** (User) Various user views
  - **Conceptual** (Community) Description of the enterprise
  - **Internal** (Physical) Files, access methods, indices, distribution
  - **Database O.S.** Recovery and concurrency

- The database, does not run on a bare machine.
- The Database O.S. (DBOS) runs on top of the O.S., such as Windows or Linux.

The Conceptual Level

- The conceptual level is most fundamental as it describes the total information and its structure/meaning.
  - to the extent we understand the information and know how to express our understanding.
- It is also generally used for manipulating the database, that is querying and modifying it.
- The tools we have:
  - **Data Definition Language** (DDL), for description
  - **Data Manipulation Language** (DML), for querying and modifying
- Tables in our example (their structure, not the specific values which change in time) were a kind of DDL.
  - They form a schema, a description of the structure.
- Of course, this level changes as the needs of the enterprise change.
The External Level

- The external level is seen by various users.
- Each view (subschema) is like a small conceptual level.
- It can also change in time.
- A particular view may be modified, deleted, or added even if the conceptual level does not change
  » For example, it may become illegal for some user to see some information

The Internal Level

- The internal level deals with file organization/storage management
- It changes in time too
  » New storage devices are brought
  » Files may have indices created because some queries have become more frequent
  » The data may be geographically distributed
The Data Base Operating System Level

- The data base operating system level deals with concurrency and recovery
- The data base operating system can change too
- The vendor of the data base may discover better methods to handle recovery/concurrency

Agenda

- The need for database management systems
- Brief overview of the relational model
- Querying relational database directly and through views
- Need for good logical design
- Need for good physical design
- Recovery
- Concurrency
- Layers of database management systems
- Independence between/among layers
- Various roles of designers, users, and maintainers
Independence Among Levels

- A very important goal is (Data-) independence between/among levels
- We must make sure that changes in one level disturb as little as possible the other levels (propagate as little as possible)

Agenda

- The need for database management systems
- Brief overview of the relational model
- Querying relational database directly and through views
- Need for good logical design
- Need for good physical design
- Recovery
- Concurrency
- Layers of database management systems
- Independence between/among layers
- Various roles of designers, users, and maintainers
Who Does What?

- The vendor sends:
  - The database operating system
  - Tools to create and manipulate the three top levels: external, conceptual, and internal

- The database administrator (DBA) and his/her staff discuss with the users what information the database should contain and its structure
  - A common model (language for describing reality) is needed for them to communicate
    - Entity-relationship model is frequently used

- The DBA and the users design the conceptual and the external levels

Design And Management Of The System

- During the actual design the database administrator uses a specific data model and a specific DDL (a completely precise way of describing the data)
  - Now, it is usually relational
  - Previously hierarchical and network models were popular
  - In the future, perhaps object-oriented (some in use)
- The database administrator and the users write programs in DML to access and manipulate the database
- The database administrator maintains the internal level changing it depending on changing applications and equipment
- The database administrator makes backups, arranges for recovery, etc
- The above description is idealized
Challenges

- We have seen just the tip of the iceberg of what needs to happen for database systems to function as required
- We need
  - Natural semantics
  - Convenient syntax
  - Efficiency
  - 100% reliability
- Enormous effort has been spent since mid 70s to achieve this

Agenda

1. Session Overview
2. Relational Data Model & Database Constraints
3. Summary and Conclusion
Summary

- Characteristics differentiate relations from ordinary tables or files
- Classify database constraints into:
  - Inherent model-based constraints, explicit schema-based constraints, and application-based constraints
- Modification operations on the relational model:
  - Insert, Delete, and Update

Assignments & Readings

- Readings
  - Slides and Handouts posted on the course web site
  - Textbook: Chapter 3

- Assignment #1
  - Textbook exercises: 1.13, 2.14, 3.14, 3.15, 3.18, 3.20

- Project Framework Setup (ongoing)
Next Session: Relational Design via ER/EER to Relational Mapping

Any Questions?