Course Contents

- Overview of databases and reasons for using them
- Relational Analysis
- The SQL language for creating and querying databases
- Logical design of relational databases
- Physical design of databases
- Concurrency control and recovery
- Selected advanced issues
What is a Database?

Proposed Answer:
A Database is a Queryable Store of Information

What about:

- phone book
- expert system
Database Management System

- Nicknames: DBMS, Database System
- Software system to manage the database, and its interactions with users and administrator
Conceptual Functions of a Database System

- Data Definition (Schema)
- Data Manipulation (insert, modify, delete)
- Query Processing
Concrete Benefits of a Database System

- Semantics of a Data Model
- Large Amounts of Persistent Storage
- Organized for Fast Access
- Fault Tolerance
- Concurrency Control
- Support for Data-Based Applications
THREE LANGUAGES IN A DATABASE SYSTEM

- Schema Language (metadata)
- Data Language (knowledge)
- Query Language (questions)

Respective illustrations:

- `create table work(name string, hours integer)`
- `insert into work values("George", 20)`
- `select * from work where hours > 10`
Critical Questions for Database Analysis

- What kind of structures in reality do we wish to represent?
- What constructs does the data language need to reflect these reality-structures?
Elementary Ontology: Entities and Relationships

(The “ER Model”)

- Entities, e.g. students, courses
- Attributes, e.g. name, location
- Relationships, e.g. marriages, enrollments

Analyze the domain to be represented in the DB, into its ER constituents.
Relational Model

- Very simple data language, yet can flexibly represent diverse applications. Makes for simple yet rich, query language—SQL
- Predominant model in practice today
- Declarative, rooted in classical logic
A Simple Relational Database – The Schema

Student(SID, Name, Age)
Course(CID, Title, Location)
Enrolls(SID, CID)
Expressing an ER Diagram in the Relational Model

- Can represent both entities and relationships in the relational model
- Consider all the facts about:
  1. Entities and their attribute values
  2. Relationships
- Record each fact as a record/tuple/row of table

Student fact: \{ID=111, Name="George", Age=27\}
Relationship fact: \{SID=111, CID=222\}
Facts, Relations, Tables and Databases

Student fact: \{ID=111, Name=”George”, Age=27\}

As a logic formula: student(111,”George”,27)

A relation is a set of facts with the same attributes:
\{\{ID=111,Name=”George”\}, \{ID=222,Name=”Sally”\}\}

As a logic program:
student(111,”George”).
student(222,”Sally”).

A table is an ordered presentation of a relation.

A relational database is a collection of tables. Its meaning is the conjunction of all of the facts in the tables.
Schema of the Database

- The Logical Structure of the Data, apart from its particular contents
- \{\text{Student}(\text{SID}, \text{Name}, \text{Age}), \text{Course}(\text{CID}, \text{Title}, \text{Loc}), \text{Enroll}(\text{SID}, \text{CID})\}
- A fuller schema also gives the types of the attributes: \{\text{Student}(\text{SID}: \text{Int}, \text{Name}: \text{String}, \text{Age}: \text{Int}), \ldots\}\
**Examples of Queries**

- Send notices to all employees who had the flu to come for a checkup
- Give the names of all students who are older than 20
- Give the names of all students in the History class

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. SQL (to be studied later) is such a language.
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. To illustrate problems with design, consider the table:

<table>
<thead>
<tr>
<th>Student</th>
<th>Course</th>
<th>Building</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>History</td>
<td>Endicott</td>
</tr>
<tr>
<td>Sally</td>
<td>Chem</td>
<td>North Lab</td>
</tr>
<tr>
<td>Sally</td>
<td>History</td>
<td>Endicott</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Problems with this Design

- An item of data should usually appear once only. Here we have the name of the building appearing multiple times. There is a danger of inconsistency, due to partial updates.

- The constraint that Course=Chem implies Building=NorthLab is written many times into the database. This is another type of redundancy, which is less obvious at first.

- We are unable to store the building for a course that has not yet been enrolled in.
An Advantage of this Design

From an efficiency point of view, it might be useful to replicate information, to speed up the access.
Example: a query that prints the students who need access to the North Lab.
Solution

Make the constraints explicit by factoring the schema Bigtable(Student,Course,Building) into two relations:

Enroll(Student,Course)
Location(Course,Building)

The design of relational databases deals with such issues.
### Enroll

<table>
<thead>
<tr>
<th>Student</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>History</td>
</tr>
<tr>
<td>Sally</td>
<td>Chem</td>
</tr>
<tr>
<td>Sally</td>
<td>History</td>
</tr>
</tbody>
</table>

### Location

<table>
<thead>
<tr>
<th>Course</th>
<th>Building</th>
</tr>
</thead>
<tbody>
<tr>
<td>History</td>
<td>Endicott</td>
</tr>
<tr>
<td>Chem</td>
<td>North Lab</td>
</tr>
</tbody>
</table>
Different users need different data

We may want a design that best reflects the inherent properties of the data. Various user groups may need to see different portions and presentations of the same data.

- Health department should see diseases but not salaries
- Payroll department should see salaries but not diseases
- Health department may prefer to see AGE instead of BIRTH
Views

One solution: give each user (class of users) privileges to look at a \textit{view}, which is a small derived database. E.g., health department sees only the derived table: \texttt{health(Name,Address,Age,Disease)}.

Such a view is computed from the existing files, without letting the user see the other information.

We need to leave flexibility for unanticipated queries. Some people may need to ask the query: How are salaries and diseases correlated?
Problems with Updating Data through Views

Unfortunately, this is sometimes impossible or difficult.

• If the user wants to change the age of an employee, how should this be reflected in the date of birth?

• How to change the sum of salaries, if some view contains this information?
Physical Design

The database system should be organized to process queries efficiently.

- If we frequently want to know what the grade for various SID is, maybe the file should be hashed on SID, allowing direct access

- If we want to print the salaries of all the employees born in a given date range, maybe the file should be sorted by BIRTH

What to do in general? Physical design of databases deals with such issues, which are also closely related to the optimization of query processing.
Recovery

The database should remain “consistent” even when hardware fails. Consider a table describing the balances of employees’ accounts in a credit union:

\text{balance}(\text{EID}, \text{Savings}, \text{Checking}).

Suppose each employee is to receive a bonus of 10 in the savings account. A single “transaction”—execution of a user program—will update the value of Savings in every record.

The recovery subsystem guarantees that this change is all-or-nothing. Once the transaction “commits,” the update is secure.
CONCURRENCY

There may also be problems because of the concurrent execution of several transactions in a time-sharing system. Consider the following table of employee balances:

<table>
<thead>
<tr>
<th>EID</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>
Concurrency (Cont’d)

• Suppose that we are running a transaction T1 computes and print the total of employee balances

• Concurrently, employee 121 wants to move 40 from account #121 to #101 using transaction T2

• Concurrency control must guarantee that the outcome is as if T1 happened before T2 or T2 happened before T1

• AVOID: debit from account #121, T1, then credit to #101
The Layers/Levels of a Database

External: Totality of user views

Conceptual: Essential structure of the data

Internal: Files, indexes and access methods

Database O.S.: Recovery and concurrency control
WHO does WHAT?

• The manufacturer sends:
  – The database operating system
  – Tools to create and manipulate the three top levels: external, conceptual and internal

• The database administrators and users together plan the conceptual structure of the database. ER model is frequently used as a design language.

• Programmers write application programs against the conceptual or view levels of the data
**Important Goal: Independence among the Levels**

Changes at one level should disturb as little as possible the other levels. For example:

- Manufacturer upgrades the recovery mechanism, in a way that is transparent to all application software
- Database administrator rearranges the indexes, without requiring any changes to applications executing at the conceptual level
Topics covered in this unit

- Why databases?
- Introduction to the rest of the course, with previews of the issues to be studied