What is a Database

- An abstraction for storing and retrieving related pieces of data
- Many different kinds of databases have been proposed
  - hierarchical, network, etc.
  - each kind supports a different abstract model for organizing data
  - in this class, we will only explain relational databases
    - sets of tables of related data

Example DB: Fortune 500 Companies

- company

<table>
<thead>
<tr>
<th>company</th>
<th>sales</th>
<th>assets</th>
<th>netincome</th>
<th>empls</th>
<th>indcode</th>
<th>yr</th>
</tr>
</thead>
<tbody>
<tr>
<td>allied</td>
<td>911500</td>
<td>13271000</td>
<td>-279000</td>
<td>143800</td>
<td>37</td>
<td>85</td>
</tr>
<tr>
<td>boeing</td>
<td>903500</td>
<td>7593000</td>
<td>292000</td>
<td>95700</td>
<td>37</td>
<td>82</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- industry codes

<table>
<thead>
<tr>
<th>indcode</th>
<th>indname</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>pharmaceuticals</td>
</tr>
<tr>
<td>44</td>
<td>computers</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The Relational Abstraction

- Information is in tables
  - Also called (base) relations
- Columns define attributes
  - Also called fields or domains
- Rows define records
  - Also called tuples
- Cells contain values
  - All cells in column have information of same type
    - e.g., integer, floating point, text, date

Operating on Databases: SQL

- Every abstraction needs an interface through which users invoke abstract operations
  - graphical interface
  - language
- Structured Query Language
- Has all those operations
- We'll focus only on queries
  - Query = question
  - Extract some data from one or more tables to answer a particular question
The Select Statement

- Every select statement yields a table of values as output
  - Sometimes there’s only one row in the table!

```sql
select columns and/or expressions
from tables
where conditions on the rows
group by group rows together
having conditions on the groups
order by order the rows
into temp save results of query in a temporary table
```

Display Company Data

```sql
SELECT *
FROM company;
```
Choose Columns

- Choosing a subset of columns is sometimes called "project" operation
- Display company name and income for each year
- 

\[
\text{SELECT compname, netincome, yr} \\
\text{FROM company;}
\]

<table>
<thead>
<tr>
<th>compname</th>
<th>netincome</th>
<th>yr</th>
</tr>
</thead>
<tbody>
<tr>
<td>allied</td>
<td>-279000</td>
<td>85</td>
</tr>
<tr>
<td>boeing</td>
<td>292000</td>
<td>82</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Choose Rows

- Find performance data for 1984 for boeing

\[
\text{SELECT compname, netincome, yr} \\
\text{FROM company} \\
\text{WHERE yr = 84 AND compname = "boeing";} \\
\]

- Which companies lost money in 1984?
Compute Columns

- Find return on assets for each year
  ```sql
  SELECT compname, yr, 
  (netincome/assets) AS roa 
  FROM company;
  ```

- Nice names for output columns
  - Name following computed column (e.g., roa) will be used to name output column

- Find company-years with roa of more than 15%

Sorting

- Can sort output by contents of a column
  - sort in ascending or descending order
  - sort by more than one column (second one breaks ties)

- Sort companies by 1984 profits
  ```sql
  SELECT compname, netincome 
  FROM company 
  WHERE yr = 84 
  ORDER BY netincome DESC;
  ```

- Sort companies by 1984 return on assets
Aggregates

- Can make calculations on entire columns
  - \( \text{sum, avg, max, min, count} \)
- How many apparel companies are in database and what are their total sales for 1984?
  
  ```sql
  SELECT Count(*) AS number, 
          Sum(sales) AS totalsales
  FROM company
  WHERE indcode = 40 and yr = 84;
  ```
  - returns a table with just one row!
- What is average percent roa for apparel companies in 1984?

Grouping and Aggregates

- Each different value for the group by fields defines a new group
- One row of output is produced for each group
- Several rows may belong to same group
  - Aggregate those using aggregation operator
- Compute total sales by all companies for each year
  
  ```sql
  SELECT yr,
          Sum(sales) AS totalsales
  FROM company
  GROUP BY yr;
  ```
More examples

- Compute total sales by all companies for each year

  ```sql
  SELECT yr, Sum(sales) AS totalsales
  FROM company
  GROUP BY yr;
  ```

- Compute total sales for each company

- What are the leading industries in total sales for 1984?

Joins

- Combine rows from one table with rows from another

- Usually join on some common column
  - Don't combine rows unless their value in the common column is the same
  - Where clause says the common column must be same in each table

- Find the industry name for each company

  ```sql
  SELECT company.compname AS compname,
        codes.indname AS industry
  FROM company, codes
  WHERE company.indcode = codes.indcode;
  ```

Lecture notes taken from 15.561 by Chrysanthos Dellarocas
Example DB: Fortune 500 Companies

- **company**

<table>
<thead>
<tr>
<th>company</th>
<th>sales</th>
<th>assets</th>
<th>netincome</th>
<th>empls</th>
<th>indcode</th>
<th>yr</th>
</tr>
</thead>
<tbody>
<tr>
<td>allied</td>
<td>9115000</td>
<td>13271000</td>
<td>-279000</td>
<td>143800</td>
<td>37</td>
<td>85</td>
</tr>
<tr>
<td>boeing</td>
<td>9035000</td>
<td>7593000</td>
<td>292000</td>
<td>95700</td>
<td>37</td>
<td>82</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **industry codes**

<table>
<thead>
<tr>
<th>indcode</th>
<th>indname</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>pharmaceuticals</td>
</tr>
<tr>
<td>44</td>
<td>computers</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

SQL Summary

- **select** columns and/or expressions
- **from** tables
- **where** conditions on the rows
- **group by** group rows together
- **having** conditions on the groups
- **order by** order the rows
- **into temp** save results of query in a temporary table
Database Design Checklist

- Meaningful tables
- Each cell holds only 1 piece of data
- Each table has a key
- Tables related with foreign keys
- Avoid redundant storage of data
- Minimize empty cells

Meaningful Tables

- Each row should represent one instance of an entity or relationship
  - One employee
  - One project-employee relationship
- One table should not contain data about several entities
  - E.g., employee id and department location in separate tables
    - Even though employee is currently assigned to a department, which has a location
    - Easier to update if employee switches departments
- Litmus test: succinct answer to:
  - “What's in this table?”
Each cell holds only 1 piece of data

- PHONE_NUM field should contain only 1 phone number
- If more than one phone number
  - Add another column if exactly two
  - Separate phone numbers table if number of phones not predetermined

<table>
<thead>
<tr>
<th>Employee id</th>
<th>Phone1</th>
<th>Phone2</th>
</tr>
</thead>
</table>

Each table has a key

- Key: a set of columns that picks out a unique row from the table
  - Last name not a key
  - First name not a key
  - First + middle + last may be a key
    - Social security number may be a more reliable key
- A table can have several keys
  - Choose one as the primary key
- Each table must have at least one key
  - Just means no duplicate rows
  - Key could be the entire set of columns
- Key cannot be null (blank)
Tables related with foreign keys

- Tables can be related via column(s) in common
- Design goal
  - A row in one table that refers to another table must refer to an existing row in that table
  - Example: Employee table and Department table
    - Don't assign employee to department 10 if that department doesn't exist in other table
  - Foreign key design rule ensures that
- A set of columns in table 1 is a foreign key for table 2 if:
  - The foreign key takes on values from the same domain as the primary key of table 2
  - When the value of the foreign key in table 1 is not null, there is a row in table 2 that has that value

Avoid redundant storage of data

- Redundant storage is wasteful
- Example
  - Suppose employee table keeps track of department and its address for each employee
  - Address repeated for every employee in department
  - What can go wrong?
    - insert new employee
    - modify department address
    - delete last employee for department

<table>
<thead>
<tr>
<th>Employee_id</th>
<th>Dept_id</th>
<th>Dept_address</th>
</tr>
</thead>
</table>

Lecture notes taken from 15.561 by Chrysanthos Dellarocas

© Copyright 2003, Chrysanthos Dellarocas – Brian Subirana
The Design Process

- Analyze the needs
  - Queries that will be made on database
  - Data entities (potential tables)
  - Relationships between entities
  - Constraints on data

- Fill out the design
  - What columns needed for each entity?

- Adjust design based on checklist above
  - May need to remove some columns into separate tables
  - Many-to-many relationships become their own tables
    - Employees table
    - Projects table
    - Employee assignments table