11.520: A Workshop on Geographic Information Systems
11.188: Urban Planning and Social Science Laboratory

Intro to SQL and One-to-Many Relationships

- **Readings, Lab 4, Homework #1, and Census data**
  - Worboys, pp. 45-67 and Hutchinson and Daniel, Chapter 6 and page 145+ on ArcView's database query tools
  - Homework Set #1
  - Homework #1 due
  - Lab #4 due. Homework #1 due

- **Database Management: Motivation and Fundamentals (from previous lecture notes)**
  - The Web as an information repository
  - Often need more highly structured data repositories (and query tools)
  - Planner's perspective and GIS implications
  - Data types, parsing, & mix-n-match issues

- **The Relational Model**
  - All data are represented as tables
  - Gets interesting when "rows" in each table have different meaning
  - The basic SELECT statement to query one or more tables:
  - Qualities of a Good Database Design
  - Why use a more elaborate database management system (DBMS)?

- **One-to-Many Examples and ArcView's Field/Summarize tools**
  - Multiple sales of same house (sales89 data)
  - Multiple geometric features for a spatial objects (town split by river, harbor islands...)
  - Handling one-to-many relationships

- **Database Management: Motivation and Fundamentals (Repeat of outline from end of previous lecture)**
  - The Web as an information repository
    - A rich information source but a loosely structured collection of relatively
unstructured data
- Hard to find what you want without search engines and portals to index and structure the information and standardize the query process
- Hard to utilize and extend knowledge on the Web without controlling/copying it (broken links, complex parsing/extraction, limited quality control, etc.)
  o Often need more highly structured data repositories (and query tools)
    - Desktop tools such as Excel, MS-Access, Filemaker, etc. handle personal database management needs (mailing lists, survey results, etc.)
    - Complex software often needed to manage multi-user access to 'persistent data'
    - Types of databases: single-user, corporate, engineering, science, image/video, geographic, ...
    - Issues: performance, metadata, user interface, data structure, concurrency, distributed, ...
    - Other 'big-system' issues: Security/reliability/integrity requirements (parcel ownership records, census data, major roads)
    - Other complications: transaction processing, data warehousing, online analytic processing, data mining, ...
  - Our focus: data structure issues and query capabilities

Other 'big-system' issues:
- Security
- Reliability
- Integrity

GIS 'demos' are easy but spatial analysis is hard

GIS 'demos' are easy but spatial analysis is hard
- No sweat if the data you want are already cleaned, parsed, and precisely suited to your question
- Useful spatial analyses involves judicious mixing and matching data from official and local sources
- Tapping into distributed, non-static databases can get complex for non automatable tasks
  - Data types, parsing, & mix-n-match issues
    - Alphanumeric: Character strings; integers, floating point numbers, dates, binary codes, ...
    - Multi-dimensional: Images, maps, spatial objects, 3D models, video, math models, ...
    - Encoding/parsing addresses, zips, census tracts (77 Mass Ave, Cambridge, MA 02139)
    - Storage space, column headers (metadata), null/missing values

- The Relational Model
  - All data are represented as tables
    - Each table can be stored or viewed as one 'flat file'
    - Tables are comprised of rows and columns
    - Simple queries select particular rows and columns from a table
    - Each table has a **primary key**, a unique identifier constructed from one or more columns
    - A table is linked (joined*) to another by including the other table's primary key. Such an included column is called a **foreign key**
    - More complex queries relate (join) multiple tables using primary/foreign keys
    - The results of any given query are just another table! (so complex queries can involve sub-queries)
    - One-to-many (and many-to-many) relations can be handled through the use of aggregation functions (sum, count, average, minimum, etc.)
  - Gets interesting when "rows" in each table have different meaning and joining tables involves one-to-many or many-to-many matches. Consider:
    - house sales in a 'sales' table
    - persons in the owner table
    - tax payments in a 'tax' table
    - counts and other statistics in a census tract table
  - Handling one-to-many and many-to-many relations can be useful but tricky:
    - Owners may have multiple properties; properties may sell more than once; etc.
    - How can you join the tables in order to determine all owners that have been in arrears on their taxes within two years of buying a property
    - Are new owners more likely to be in arrears on their taxes if the property is in a low (high) income census tract?

* Refer back to the Lecture Notes.
in a low (high) income census tract?

- Limited attention to database management technologies in 11.520 (because ArcView capabilities are limited)
  - Focus on simple relational 'joins' and handling one-to-many relationships.
  - In the next few census-related lectures we’ll also use ESRI's spatial database engine (SDE) but without complex queries
  - More spatial database management (using the structured query language, SQL) in the Spring classes (11.521, 11.523)

The basic SELECT statement to query one or more tables:

```
SELECT [DISTINCT] column_name1[, column_name2, ...] 
  FROM table_name1[, table_name2, ...] 
WHERE search_condition1 
  [AND search_condition2 ...]  
  [OR search_condition3...] 
 [GROUP BY column_names] 
 [ORDER BY column_names];
```

**Example A**: Select address, date, realprice columns from `sales89` table for houses that sold after July 1, 1989 for more than $250,000

```sql
SELECT address, date, realprice 
FROM sales89 
WHERE realprice > 250000 and date > "07/01/1989"
```

**Example B**: Count the number of 1989 sales associated with each address that is listed in the `sales89` table and order the results by sale_count, then, address, and date:

```sql
SELECT address, count(distinct date) sale_count 
FROM sales89 
WHERE realprice > 250000 and date > "07/01/1989" 
GROUP BY address, date 
ORDER BY count(distinct date), address, date
```

**Example C**: For every `sales89` sale in Cambridge, list the address, saledate, and sales price along with the percent of adults in the surrounding census block group who had less than a high school education. The `sales89` and cambbgrp tables could be joined by a common column (if the `sales89` table had a column listing the census block group) or by a spatial join (that used the geographic data to compute which block group contained each sale):
SELECT s.address, s.date, s.realprice, 
100*(c.EDU1 + c.EDU2 / c.EDUTOTAL) low_ed_percent 
FROM cambbgrp c, sales89 s 
WHERE c.stcntrbg = s.stcntrbg

if the sales89 table included as a 'foreign key' the stcntrbg 'primary key' from the cambbgrp table, or:

SELECT s.address, s.date, s.realprice, 
100*(c.EDU1 + c.EDU2 / c.EDUTOTAL) low_ed_percent 
FROM cambbgrp c, sales89 s 
WHERE s.SpatialObject IS CONTAINED WITHIN 
c.SpatialObject

○ Qualities of a Good Database Design

- Tables reflect real-world structure of the problem
- Can represent all expected data over time
- Avoids redundant storage of data items
- Provides efficient access to data
- Supports the maintenance and integrity of data over time
- Clean, consistent, and easy to understand
- Note: These objectives are sometimes contradictory!

○ Why use a more elaborate database management system (DBMS)?
  - Handling multi-table complexity (one-to-many, ...)
  - Ease of documenting/replicating queries/results
  - Performance
  - Security
  - Safe for multiple users
  - Sharing data among applications
  - Built-in data dictionary

• One-to-Many (and many to many) Issues & 'Group by' strategies

  ○ A parcel may have more than one owner; an owner may own more than one parcel; a parcel may sell more than once; ...

  How can we tell if any sales89 houses sold more than once that year? How many times did they sell? What was the average price?

  Determine which homes in sales89 sold more than once using the
A town may have several parts/islands (e.g. the many polygons that make up Boston) but the data table may have only one row for each town.

*How can we compute a meaningful population density? (e.g., town-pop / total-town-area)*

May want to ask more complicated questions that span levels of aggregation.

*How can we join Eastern Mass census tract table to the Cambridge block group table?*

...tract numbers don't match; need to add/edit a new column and manipulate strings.

How do we identify the Cambridge block groups whose median household income is greater than the median for the corresponding tract? What about the block groups in Cambridge that had exceptional numbers of newborns (children under 1 year old) for their tracts, defined by having more newborns than 1 standard deviation above the mean for the tract?

Note: We don't have precalculated statistics for the average or standard deviation of newborns by tract, so we must calculate them using ArcView's "Field/Summarize ..." feature in ArcView

**One-to-Many Examples and ArcView's Field/Summarize tools**

- Density measures
- Handling one-to-many relationships:
  - Aggregation Functions in ArcView
  - Aggregate Functions and 'group by' in SQL queries
- Calculating Population Density for Massachusetts