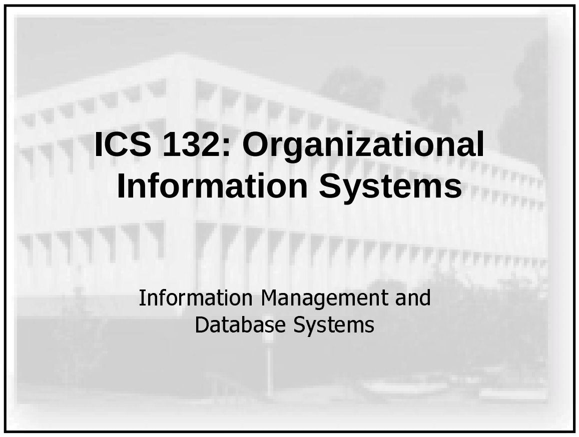


ICS 132: Organizational Information Systems

Information Management and Database Systems

administrivia

- second TA
  - dhawal shah, shahd@uci.edu
- homework
- discussion sections

information management

- organisations depend on information
  - about their own processes
  - about what's going on around them
  - the basis of *monitoring* and *planning*
- the dependence is fundamental
  - modern organisational forms and practices are built around the idea that information is available
    - remember the case of the filing cabinet

keys to information mgmt

- scale
  - dealing with information volume
- flexibility
  - need to deal with information in different ways
    - different questions you want to ask
    - different views from different people
- consistency
  - maintaining information quality and integrity

organisational factors

- centralisation and distribution
  - balancing control and autonomy
  - balancing individual and collective control
  - making information more visible
    - and making patterns of access… e.g. Delphion
- standardisation and classification
  - need to come to agreement about what info *means*
  - examples from the ICD

data, database, DBMS

- data
  - a big pile of bits
- a database
  - structured collection of data
  - organised according to predefined relations
    - paper documents?
    - contact list on my Pilot?
    - world wide web?
- why bother with a database?
  - need to maintain consistency
  - don't want to have to repeat information

data, database, DBMS

- DBMS: Data Base Management System
  - set of programs to define, update, control databases
    - this is what we often mean when we say “database”
    - Sybase, Oracle, DB2, MySQL, Postgres…
  - DBMS responsibilities
    - layout out information on the disk, building indexes, getting from one piece of data to another
  - your responsibilities
    - modeling the information
    - describing the relations
    - creating queries

data modeling

- first step is to model the data
  - looking for generic structure
  - later, encode this as a database format
- modeling
  - modeling languages suit particular forms of encoding
  - ER modelling
    - ER = entity-relationship
    - particularly suited to relational databases (more later…)

ER modeling

- identifying entities and their relationships
  - not unlike OO modeling, but entirely static
- three (not two) elements
  - entities
    - basic objects of the domain
  - attributes
    - relevant features of those objects
  - relationships
    - (limited) ways in which objects related to each other

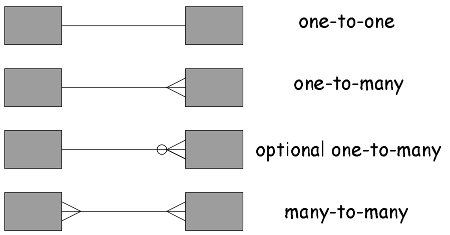
ER modeling

- entities
  - broadly, entities in ER are like classes in Java
    - the describe a class of data
      - *concrete*: person, book, computer
      - *abstract*: account, concept, holiday
  - defining entities defines what you can know
    - e.g. different ways of describing books
      - for a library, a publisher, or a bookstore
- attributes
  - features of entities
    - people have names, books have titles, cars have license numbers, etc.

ER modeling

- relationships between elements
  - the relevant feature is cardinality
    - “how many”
  - relationships describe links between data
  - relationship types describe cardinal properties

ER modeling



one-to-one

one-to-many

optional one-to-many

many-to-many

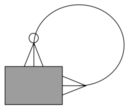
ER modeling: example

ER modeling

- identifying instances
  - – database needs to be able to tell instances apart
  - – all it has to go on is what's in the ER model
- the primary key
  - – one or more attributes that uniquely identify items
    - what identifies people?
    - what identifies books?
    - what identifies houses?
    - what identifies cars?
    - what identifies bank accounts?

ER modeling

- things to remember
  - – the simplicity of ER is useful
    - ER is a communication tool – esp. with the participants
  - – you're dealing with *types*, not *objects*
    - not really entities, but *kinds of entities*



ER modeling

- things to remember
  - – entities in the domain are not entities in the world
    - domain is essentially *things we might want to know about*
  - – sometimes, entities will be virtual objects
    - e.g. representing a *transaction* as an entity
      - – purchasing a book
      - – making a cash withdrawl

ER modeling exercise

- draw an ER model for a car rental database

database styles

- DBMS are generic
  - – represent many different forms of information
  - – search for common structure
- early DBMS styles
  - – *hierarchical* data models
    - hierarchical storage
    - greater or lesser constraint on branch structure
  - – *network* data models
    - objects and arbitrary relationships

database styles

- relational database style
  - data is stored in tables
  - each row represents a relationship amongst values
    - in fact, tables are often known as relations
  - link to mathematical notion of relation
    - mapping between domains
      - domain of keys
      - domain of values

relational databases

- tables and relations
  - a relationship database involves multiple tables
  - why split them up?
    - avoid repetition
      - e.g. don't store delivery address separately for each order
      - inefficient
      - can lead to inconsistency
  - putting them together again
    - need to correlate information
      - draw from many places
      - integrate across tables
  - we'll talk more on Wednesday about constraints

relational databases

- the key field
  - data uniquely identifying a particular row
  - "uniquely identifying" is not a technical condition
    - it's up to you to figure this out
      - John, Paul, George and Ringo may all be different
      - but given name is a poor key field in most applications
    - sometimes it can be a single field
      - e.g. SSN
    - sometimes it may be a combination of fields
      - e.g. name & address

relational databases

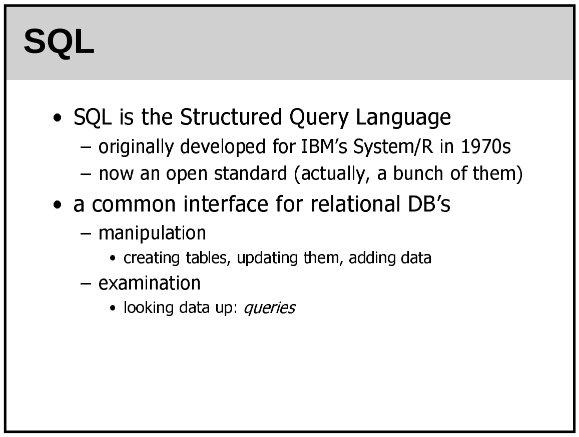
- first, turn entities into tables
  - with attributes as columns
- then, examine relations as tables
  - many-to-many relationships almost always tables
  - one-to-one relationships?
  - one-to-many relationships?
- why am I prevaricating?
  - ER model is informal
  - rules for formalising data definitions (next time)

relational databases

- schemas
  - relational databases based on formal data definitions
  - again, like specifying classes
  - schema describes table structure and storage req'ts
  - table "book":
    - author CHAR(50)
    - title CHAR(100)
    - isbn CHAR(30)
    - price DECIMAL(3,2)

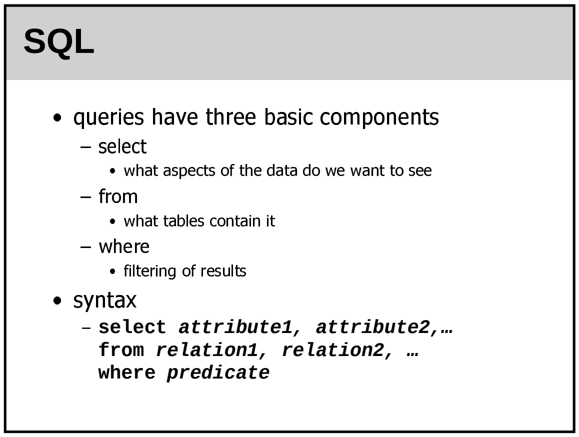
exploiting structure

- all DBMS exploit common structure
  - common structure across instances
    - all books have these properties
  - common structure across databases
    - all data can be modeled in this way
      - e.g. relational data model
  - what's the point of this common structure?



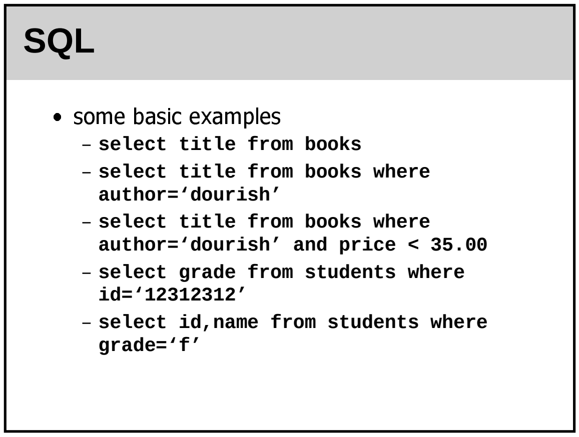
SQL

- SQL is the Structured Query Language
  - – originally developed for IBM's System/R in 1970s
  - – now an open standard (actually, a bunch of them)
- a common interface for relational DB's
  - – manipulation
    - creating tables, updating them, adding data
  - – examination
    - looking data up: *queries*



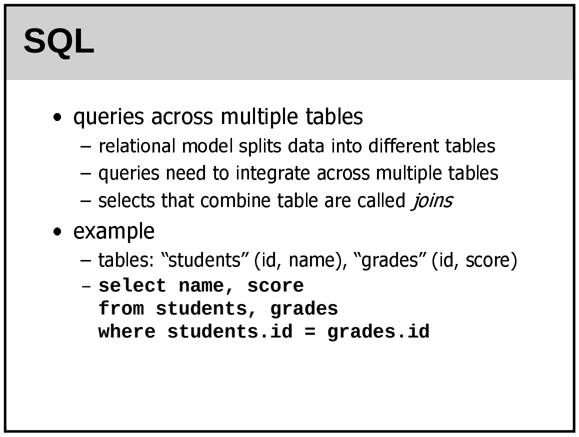
SQL

- queries have three basic components
  - – select
    - what aspects of the data do we want to see
  - – from
    - what tables contain it
  - – where
    - filtering of results
- syntax
  - – **select *attribute1, attribute2,…***
    from *relation1, relation2, …*
    where *predicate*



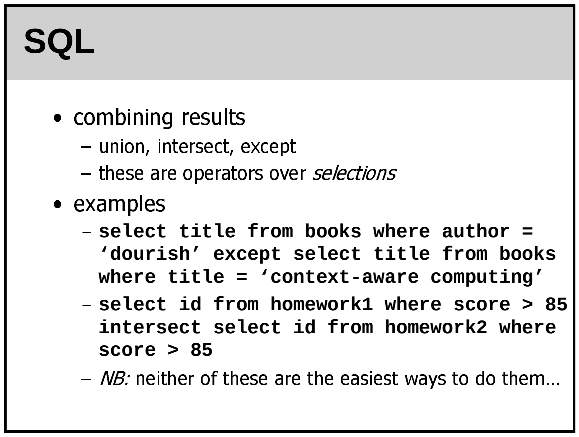
SQL

- some basic examples
  - – **`select title from books`**
  - – **`select title from books where author='dourish'`**
  - – **`select title from books where author='dourish' and price < 35.00`**
  - – **`select grade from students where id='12312312'`**
  - – **`select id,name from students where grade='f'`**



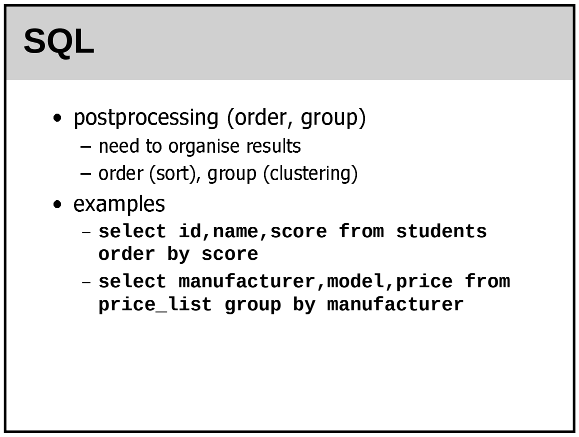
SQL

- queries across multiple tables
  - – relational model splits data into different tables
  - – queries need to integrate across multiple tables
  - – selects that combine table are called *joins*
- example
  - – tables: "students" (id, name), "grades" (id, score)
  - –
    ```
    select name, score
    from students, grades
    where students.id = grades.id
    ```



SQL

- combining results
  - – union, intersect, except
  - – these are operators over *selections*
- examples
  - – **`select title from books where author = 'dourish' except select title from books where title = 'context-aware computing'`**
  - – **`select id from homework1 where score > 85 intersect select id from homework2 where score > 85`**
  - – *NB:* neither of these are the easiest ways to do them…



SQL

- postprocessing (order, group)
  - – need to organise results
  - – order (sort), group (clustering)
- examples
  - – **`select id,name,score from students order by score`**
  - – **`select manufacturer,model,price from price_list group by manufacturer`**

SQL

- some processing over results
  - e.g. avg(), sum(), count()
- examples
  - `select count(*) from students`
  - `select avg(score) from grades`
  - `select author, avg(price)`
    `from books group by author`

SQL

- summary
  - selecting, combining, processing
- there’s more, of course…
  - subqueries
  - update and modification as well as querying

using SQL

- what SQL is not
  - not a full programming language
  - not a development environment
- sql queries normally embedded in programs
  - e.g. from java, using JDBC
  - languages differ in their degrees of integration

using SQL

```
Class.forName(JDBC_CLASS);
Connection conn = DriverManager.getConnection(DB_URL, "ics132", “password");
Statement statement = conn.createStatement();
ResultSet rs = statement.executeQuery(“select title,author from books”);
ResultSetMetaData md = rs.getMetaData();

out.println("<TABLE BORDER=2>");
out.println("<TR>");
for (int i = 1; i < md.getColumnCount() + 1; i++) {
  out.println("<TD><B>" + md.getColumnName(i).trim() + "</B></TD>");
}
out.println("<TR>");
while (rs.next()) {
  out.println("<TR>");
  for (int i = 1; i < md.getColumnCount() + 1; i++) {
    out.println("<TD>" + rs.getString(i) + "</TD>");
  }
  out.println("</TR>");
}
out.println("</TABLE>");
```

summary

- key points:
  - information processing is about making the world tractable
    - amenable to summarisation, modeling & prediction
  - DBMS provides a framework for data management
    - regularised for efficiency, consistency & maintenance
  - relational databases
    - organise information according to relations & tables
    - sql provides uniform access

what’s coming up

- Wednesday
  - more on database design and normalisation
  - homework on databases
- Friday
  - discussion section
- Monday
  - performance and competition
  - read Alter chapter 6
- next Wednesday is the mid-term