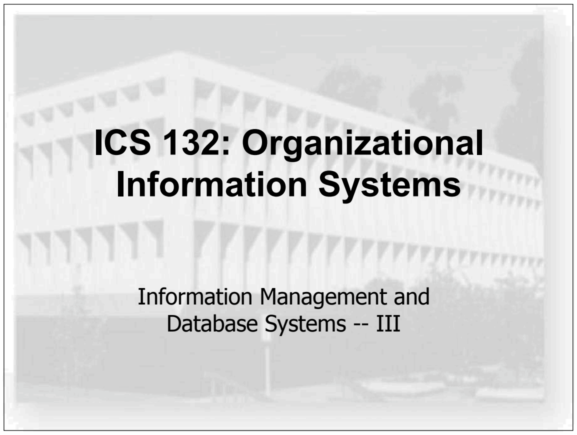
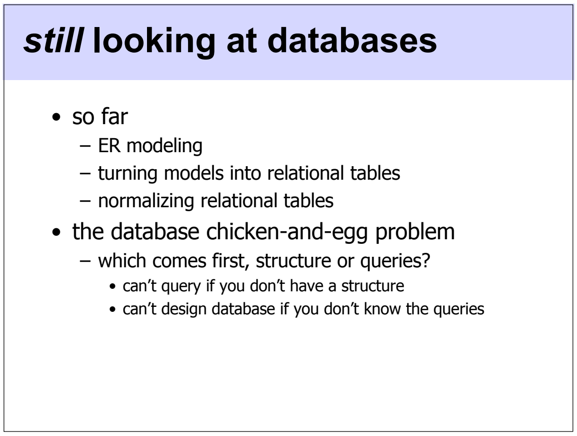


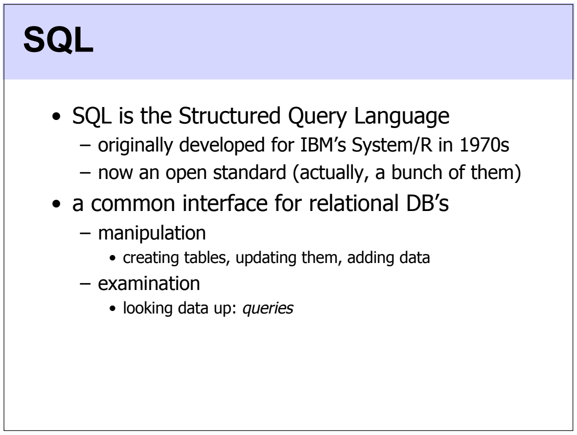
ICS 132: Organizational Information Systems

Information Management and
Database Systems -- III



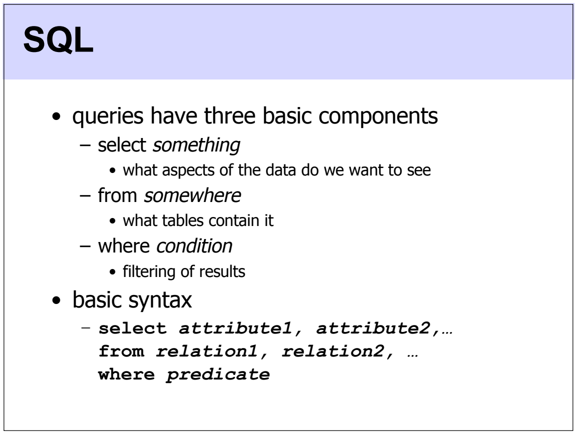
still looking at databases

- so far
  - ER modeling
  - turning models into relational tables
  - normalizing relational tables
- the database chicken-and-egg problem
  - which comes first, structure or queries?
    - can't query if you don't have a structure
    - can't design database if you don't know the queries



SQL

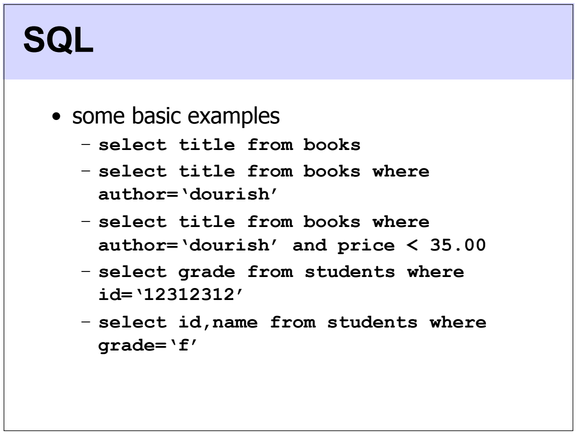
- SQL is the Structured Query Language
  - originally developed for IBM's System/R in 1970s
  - now an open standard (actually, a bunch of them)
- a common interface for relational DB's
  - manipulation
    - creating tables, updating them, adding data
  - examination
    - looking data up: *queries*



SQL

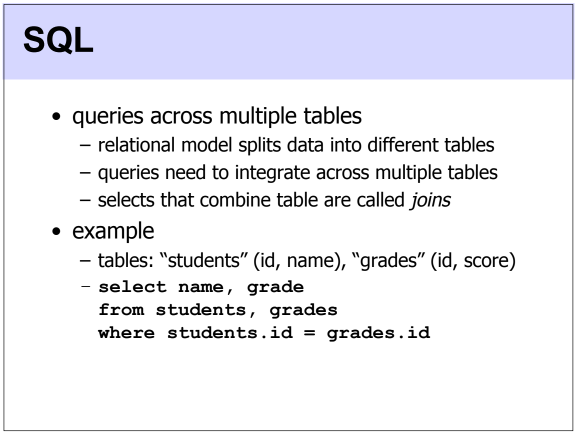
- queries have three basic components
  - select *something*
    - what aspects of the data do we want to see
  - from *somewhere*
    - what tables contain it
  - where *condition*
    - filtering of results
- basic syntax

  -
    ```
    select attribute1, attribute2,…
    from relation1, relation2, …
    where predicate
    ```



SQL

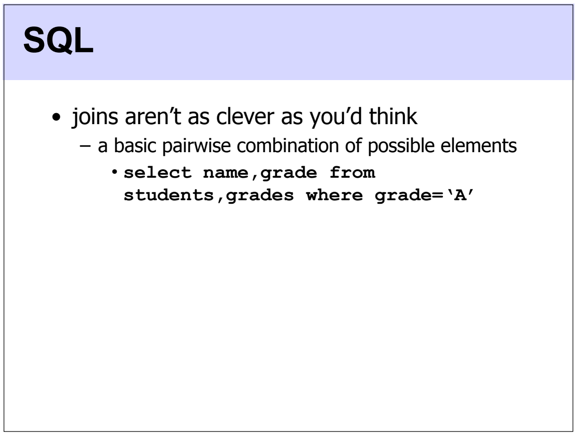
- some basic examples
  - `select title from books`
  - `select title from books where author='dourish'`
  - `select title from books where author='dourish' and price < 35.00`
  - `select grade from students where id='12312312'`
  - `select id,name from students where grade='f'`



SQL

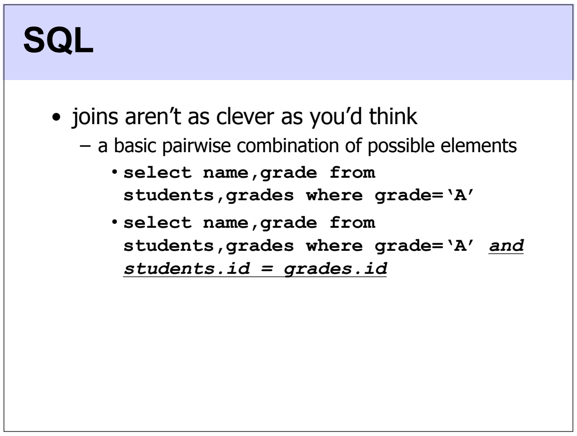
- queries across multiple tables
  - relational model splits data into different tables
  - queries need to integrate across multiple tables
  - selects that combine table are called *joins*
- example
  - tables: "students" (id, name), "grades" (id, score)

  -
    ```
    select name, grade
    from students, grades
    where students.id = grades.id
    ```



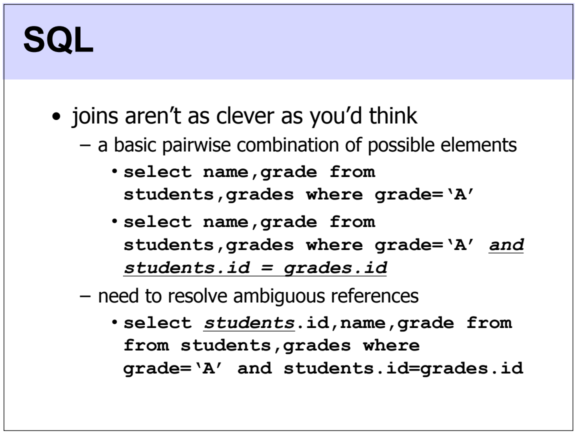
SQL

- joins aren’t as clever as you’d think
  - a basic pairwise combination of possible elements
    - `select name,grade from students,grades where grade=‘A’`



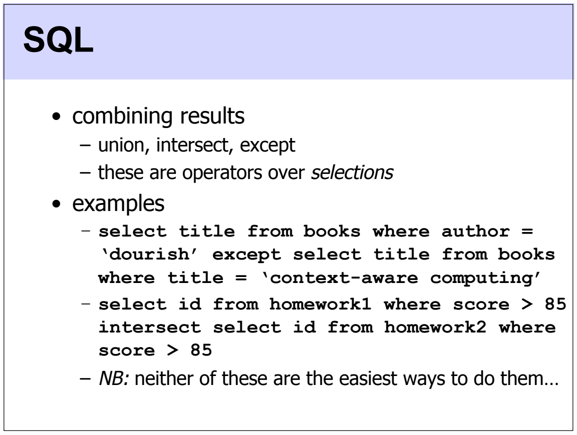
SQL

- joins aren’t as clever as you’d think
  - a basic pairwise combination of possible elements
    - `select name,grade from students,grades where grade=‘A’`
    - `select name,grade from students,grades where grade=‘A’` *and students.id = grades.id*



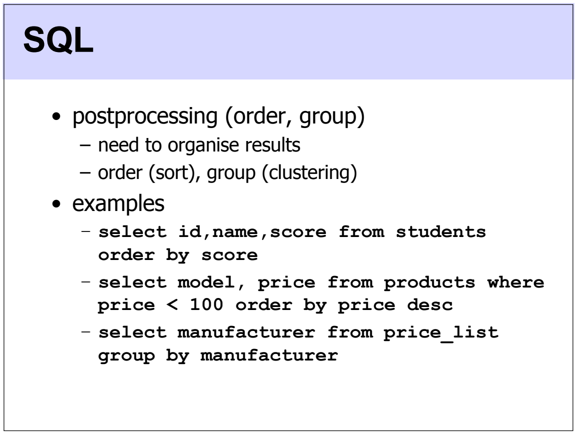
SQL

- joins aren’t as clever as you’d think
  - a basic pairwise combination of possible elements
    - `select name,grade from students,grades where grade=‘A’`
    - `select name,grade from students,grades where grade=‘A’` *and students.id = grades.id*
  - need to resolve ambiguous references
    - `select` *students*`.id,name,grade from from students,grades where grade=‘A’ and students.id=grades.id`



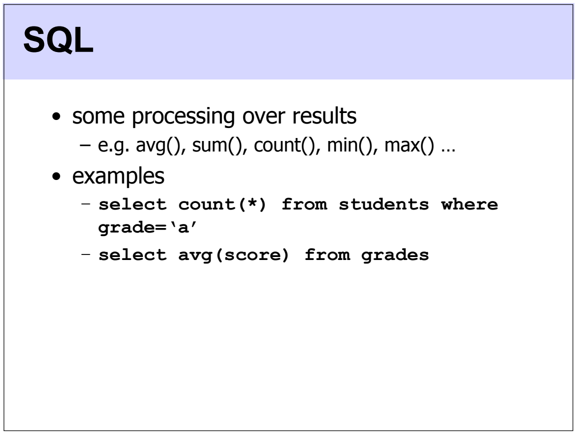
SQL

- 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…



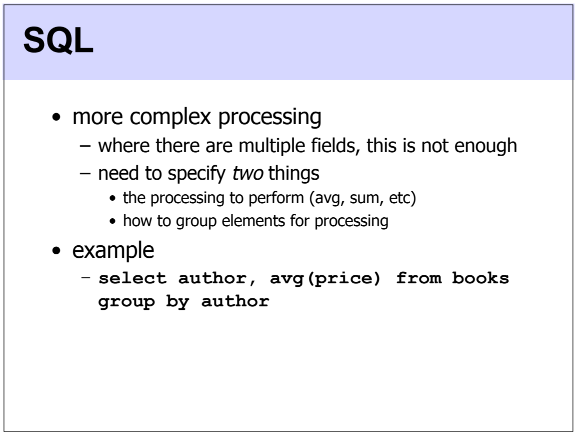
SQL

- postprocessing (order, group)
  - need to organise results
  - order (sort), group (clustering)
- examples
  - `select id,name,score from students order by score`
  - `select model, price from products where price < 100 order by price desc`
  - `select manufacturer from price_list group by manufacturer`



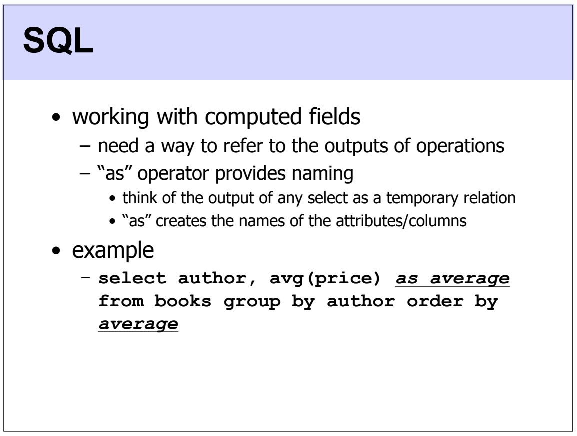
SQL

- some processing over results
  - e.g. avg(), sum(), count(), min(), max() …
- examples
  - `select count(*) from students where grade=‘a’`
  - `select avg(score) from grades`



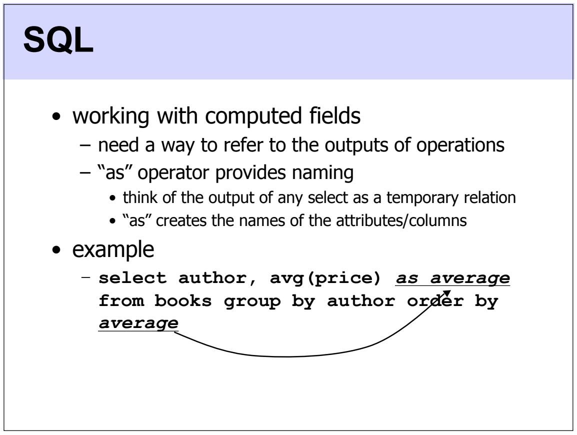
SQL

- more complex processing
  - where there are multiple fields, this is not enough
  - need to specify *two* things
    - the processing to perform (avg, sum, etc)
    - how to group elements for processing
- example
  - **select author, avg(price) from books group by author**



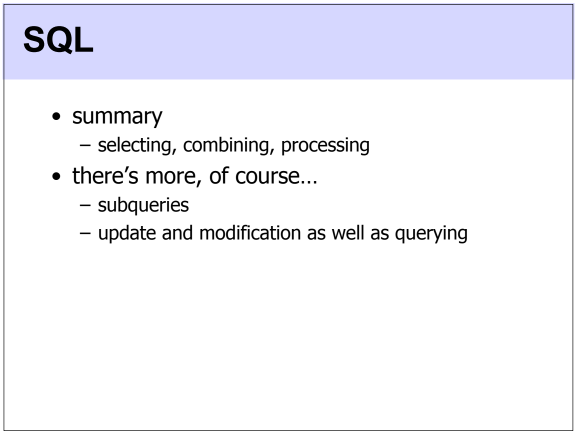
SQL

- working with computed fields
  - need a way to refer to the outputs of operations
  - “as” operator provides naming
    - think of the output of any select as a temporary relation
    - “as” creates the names of the attributes/columns
- example
  - **select author, avg(price) *as average* from books group by author order by *average***



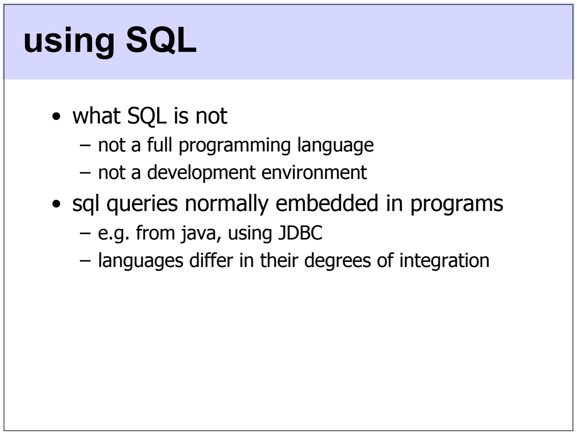
SQL

- working with computed fields
  - need a way to refer to the outputs of operations
  - “as” operator provides naming
    - think of the output of any select as a temporary relation
    - “as” creates the names of the attributes/columns
- example
  - **select author, avg(price) *as average* from books group by author order by *average***



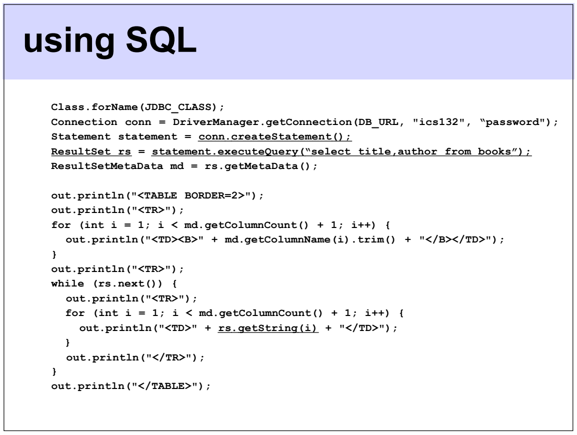
SQL

- 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>");
```

the organizational context

- okay, fine, so databases are important
  - understand technology to understand opportunities
- but, the 132 perspective
  - internal and external variety of organizations
  - co-evolution of technology and organizational practice
- an example
  - unified filing in The Department (a different one!)

summary

- key points:
  - modeling are about *making the world tractable*
    - amenable to encoding, summarisation, & prediction
  - relational databases
    - organise information according to relations & tables
    - sql provides uniform access
  - same two problems process representations
    - the detail of the representation
    - the object of the representation
  - need to see info use in organizational context
    - uses to which it is put
    - practices in which it is enmeshed