DATABASE DESIGN

Definition

- organised collection of data
- represents repetitive information
- supports retrieval
- power and flexibility depend on data model

Relational Database

- Tables with columns and rows
- Logical vs. physical representation
- Multiple indexes
- Supports inter-table relationships

Normalisation

- efficiency, redundancy, dependency, flexibility (multiple 'views')
- E.F. Codd, 1970: five normal forms

1NF - values in each column of a table are atomic
    - separate table for each group of related data

Relationships

- one-to-one - patient < > bed

- one-to-many - patient < > > disease

- many-to-many - patient < < > > doctors

The relationship is one-to-one if one and only one item in Table A applies to one and only one item in Table B (e.g., each U.S. citizen has only one Social Security number, and each Social Security number applies to only one U.S. citizen; no citizen can have two Social Security numbers, and no Social Security number can refer to two citizens).

A relationship is one-to-many if one item in Table A can apply to multiple items in Table B. The terms female and male will apply to many people, but each person can be only one or the other. A one-to-many relationship is the most common one between tables in databases.

Finally, a relationship is many-to-many if multiple items in Table A can apply to multiple items in Table B. For example, a record album can contain songs by multiple artists and artists can make multiple albums. You should try to avoid many-to-many relationships in your design because they lead to data redundancy and integrity problems.

Relationships and keys work together in that a key in one table will normally relate to a field in another.

What kinds of things might Amazon store?
username (unique)
password
email address (perhaps unique)

billing address (probably just one)

shipping address (multiple)

credit card number (multiple)

NB where we want to store multiple records of the same type for one user, we'll need to use another table

user
id (unique), un (unique), pw, ea,
1,j,xx,jules@j.com
2,f,ww,fred@bloggs.com

-- getting the user's id

SELECT id
FROM user
WHERE un = ? (e.g. whatever was entered)
AND pw = ?

-- id is then stored in a session variable

shipping
id, u_id, street_number, street, city, postcode
1,1,12,Nicolson Sq, Edinburgh, EH8 9DF
2,2,8, eight street, Edinburgh, EH8 8HE
3,2,9, nine street, edinburgh, EH9 9HE

-- retrieve all shipping addresses for a given user
SELECT *
FROM shipping
WHERE u_id = ? (the id that we just retrieved from user table)

credit
id,u_id,CCnumber
1,3,1234 2345 xxxx xxxx

SQL for the product example

You can create the tables and data shown below in your own database by opening and running the SQL file in SQLPro : jr_product_example-30-01-2013.sql

```
-- Retrieving product details to make it world readable
-- NB pulls all fields for the JOIN
SELECT *
FROM jr_product
JOIN jr_product_cat
ON jr_product.prod_cat = jr_product_cat.id

-- Retrieving product details to make it world readable
-- NB pulls specified fields for the JOIN
SELECT jr_product.id,jr_product_cat.cat
FROM jr_product
JOIN jr_product_cat
ON jr_product.prod_cat = jr_product_cat.id


-- JOIN  on  3 tables from product table
SELECT
jr_product.id,jr_product_cat.cat,jr_product_col.col,jr_product
_sizes.size
FROM jr_product
JOIN jr_product_cat
ON jr_product.prod_cat = jr_product_cat.id
JOIN jr_product_col
ON jr_product.prod_col = jr_product_col.id
JOIN jr_product_sizes
ON jr_product.prod_size = jr_product_sizes.id
```