ICS 132: Organizational Information Systems

Information Management and Database Systems

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 Management

- 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
- Note the role of the machine metaphor
  - Standardization, repeatability, consistency...
  - Not concerned with the data but with its form

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
  - Controlling the form is a very powerful position
  - Examples from the ICD

Data, Database, DBMS

- 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

Database Styles

- DBMS store generic information
  - Distinguishing characteristic is the basic data type
database styles

- DBMS store generic information
  - distinguishing characteristic is the basic data type
  - network

- object-oriented

- relational

database styles

- DBMS store generic information
  - distinguishing characteristic is the basic data type
  - network

- object-oriented

- relational

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 modeling
      - ER = entity-relationship
      - particularly suited to relational databases
        - based on the relational calculus
        - a systematic procedure for turning models into tables

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
    - (constrained) ways in which objects related to each other

ER modeling

- entities & entity sets
  - entities occur in sets
  - broadly, entity sets in ER are like classes in Java
    - the describe a class of data
      - concrete: person, book, computer
      - abstract: account, concept, holiday
  - entities are like instances
    - the important thing about entities is that they can be distinguished from one other
  - defining entities defines what you can know
    - definitions suited to different purposes
      - e.g. different ways of describing books
        - for a library, a publisher, or a bookstore
ER modeling

- attributes
  - attributes are properties of an entity
  - attributes have values
    - normally, single-valued ("atomic")
      - e.g. a person has just one SSN
    - sometimes, multi-valued
      - e.g. a person may have more than one phone number

- relationships define relations between entities
  - relationship sets link entity sets
    - essentially, a typology of relations, e.g.
      - from employee to office
      - from course to instructor
      - from course to student

- relationships can have attributes
  - attributes not of one entity or other, but the relationship between them
    - e.g. last-accessed
      - for bank accounts and account holders

- relationships have cardinality (number)

  - one-to-one
    - (e.g. person to office)
  - one-to-many
    - (e.g. department to person)
  - optional one-to-many
    - (e.g. classes to TAs)
  - many-to-many
    - (e.g. classes to students)

ER modeling: example

- the primary key
  - 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 individual entities
      - what identifies people?
      - what identifies books?
      - what identifies houses?
      - what identifies cars?
      - what identifies bank accounts?
the primary key

- relationships also have primary keys
  - primary key of relationship is set of primary keys of the entity sets involved
  - might add descriptive attributes of relationship

ER modeling

- the simplicity of ER is useful
  - ER is a communication tool – esp. with the participants in a process/setting
- you’re dealing with types, not objects
  - not really entities, but entity sets
- relationship vs attribute?
  - depends on what you want to know
  - structure of data depends on the questions you’ll want to ask of it

ER modeling exercise

- draw an ER model for a car rental database
  - identify cardinality
  - identify primary keys

turning models into tables

- step 1
  - for each entity in the ER model
    - create a relation that includes all the atomic attributes
    - choose one or more attributes as the primary key

turning models into tables

- step 2
  - for each one-to-one relationship in the schema
    - identify the two entity sets S and T
    - choose one (say, S)
    - include the primary key of T as an attribute of S
    - include the atomic attributes of the relationship as attributes of S

turning models into tables

- step 3
  - for each 1:N relationship
    - identify the relation S at the "N" side of the relationship
    - include the primary key of T as an attribute of S
    - include the atomic attributes of the relationship as attributes of S
turning models into tables

• step 4
  – for each two-way N1:N2 relationship
    • create a new relation S to represent this relationship
    • include primary keys of both relations in S
    • include relationship’s atomic attributes in S

turning models into tables

• step 5
  – for each multi-valued attribute
    • create a table to represent this attribute
    • one column for a single value of the attribute
    • add the primary key of the entity (or relationship) of which it is an attribute

turning models into tables

• step 6
  – finally, for each multi-way relationship
    • create new relation S
    • include all the primary keys as attributes of S
    • include atomic attributes of relation as attributes of S

next time

• more databases
  – relational database normalization
  – SQL queries
• read the Bowker paper