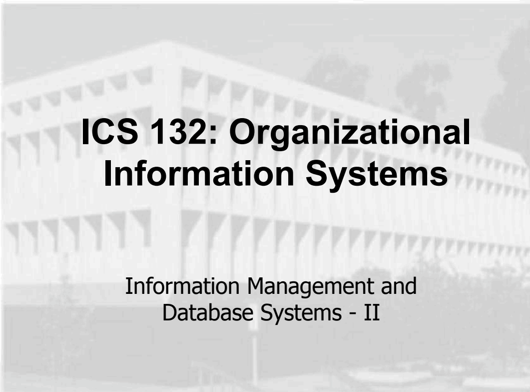


ICS 132: Organizational Information Systems

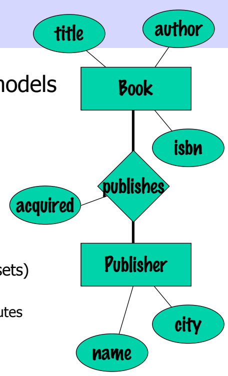
Information Management and Database Systems - II

administrivia

- midterm
  - next Tuesday
  - sample paper on web site
    - remember, the syllabus has changed some

recap

- last time, we covered ER models
  - primary objects
    - entity sets
      - roughly, object types
    - entities
      - individually distinguishable
    - attributes
      - atomic or multi-valued
    - relationships (between entity sets)
      - relationships have cardinality
      - relationships also have attributes



recap

- some observations
  - the variability in models of a domain
    - degree of specificity
    - attributes verses entities
    - relationship attributes
  - generally, we don't model the domain
    - model the *information needs* of some users
    - what you need to know determines what you represent
      - this is inevitable
      - but hopefully, you buy yourself some future-proofing too if you do your job right

database styles

- relational database style
  - origins at IBM
    - algebraic model developed by Edgar (Ted) Codd at IBM
    - first large-scale implementation in System R (1970s)
      - also the origin of SQL, Structured Query Language
  - data is stored in tables
  - each row represents a relationship amongst values
    - in fact, tables are called "relations" in the relational model
  - link to mathematical notion of relation
    - mapping between domains
      - domain of keys
      - domain of values

relational databases

- tables and relations
  - a relational 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

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 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

turning models into tables

- representing entities
  - tables that represent the attributes of each entity
  - a primary key to uniquely identify each row
- representing relationships
  - an association of primary keys
    - inside one of the entity relations
    - as a separate relation

normalization

- again, relationship between defn and queries
  - the structure of your database is intimately tied to the queries you will perform against it
  - query languages have different constraints
    - so, need to ensure that database design matches the needs of the query language
  - we'll be using SQL
    - based on the relational calculus
    - designed alongside relational model
  - database *normalization*
    - ensure database meets a set of structural criteria
    - enshrined as a set of "normal forms"

normalization

- there's a whole set of normal forms…
- we'll just look at three
  - first normal form
    - rule: no repeating groups
  - second normal form
    - rule: no non-key attribute depends on *part* of the key
  - third normal form
    - rule: no non-key attribute depends on another non-key attribute

first normal form

- no repeating groups
  - essentially, normalise the record length

Title	**Price**	**Author1**	**Author2**	**Author3**
Where the Action Is	$30.00	Dourish		
Analyzing Social Settings	$31.95	Lofland	Lofland	
Compilers	$72.00	Aho	Sethi	Ullman

first normal form

- no repeating groups
  - essentially, normalise the record length

Title	**Price**	**Author**
Where the Action Is	$30.00	Dourish
Analyzing Social Settings	$31.95	Lofland
Compilers	$72.00	Aho
Compilers	$72.00	Sethi
Compilers	$72.00	Ullman

second normal form

- no non-key attributes depend on *part* of the key
  - essentially, make key as small as it can be

Author	**Title**	**Price**	**Email**
Dourish	Where the Action Is	$30.00	jpd@ics.uci.edu
Baldi	Bioinformatics	$49.95	baldi@ics.uci.edu

second normal form

- no non-key attributes depend on *part* of the key
  - essentially, make key as small as it can be

Author	Email
Dourish	jpd@ics.uci.edu
Baldi	baldi@ics.uci.edu

Author	Title	Price
Dourish	Where the Action Is	$30.00
Baldi	Informatics	$49.95

third normal form

- no attributes depend on other *non*-key attributes
  - essentially, a relation should be about just one thing

Author	Title	Price	Purchaser	Seller	Employed
Dourish	Where the Action Is	$30.00	Maria	Hans	1/1/03
Dourish	Where the Action Is	$30.00	Joey	Amy	1/1/02
Baldi	Bioinformatics	$49.95	Lisa	Jaime	7/1/01

third normal form

- no attributes depend on other *non*-key attributes
  - essentially, a relation should be about just one thing

Title	Purchaser	Seller
Where the Action Is	Maria	Hans
Where the Action Is	Joey	Amy
Bioinformatics	Lisa	Jaime

Seller	Employed
Hans	1/1/03
Amy	1/1/02
Jaime	7/1/01

Author	Title	Price
Dourish	Where the Action Is	$30.00
Baldi	Informatics	$49.95

normalization

- normalization transforms database structure
  - eliminates repetition
  - disentangles dependencies
  - clarifies relationships
- two benefits of these transformations
  - semantic
    - cleaner definitions
    - clarifies "meaning"
  - practical
    - optimizes for SQL-based queries

next time

- next time, SQL syntax and queries