Inleiding Informatiekunde

Databases

- History, Definitions
- When should DB's be used?
- Consistency, Flexibility of View
- Relational Model
- New Horizons, Logic DB's, OO DB's
- Revival of Hierarchical DB's (LDAP)

Relevant reading: Brookshear, *CS: An Overview*, ch.8-9

Databases: Importance

- IBG: student loans, grants
  - university record
  - high school record
  - used for admission, assessment of effectiveness
- banks: PIN cards, automatic teller machines (ATM)
- libraries: holdings, availability
- hospitals
  - record of diagnosis, status, treatment
  - billing
  - used for epidemiology, assessment of effectiveness
- municipalities: residents, taxes
- tax authorities
- ...??

Databases: History

Background

- Key idea in CS: separate data from algorithms
  - abstraction to improve understanding
  - data separate to avoid redundancy, error
- Very (too) general: DB's are sources of data for several applications
- Early history: long-term storage was tape
  - special (sequential) algorithms
  - COBOL programming — payroll DB's on tape
- Picture: a shoebox of index cards, or a 'rolodex'

Databases: Desiderata

Consider: A tape-processing program that processing payroll algorithms reads records which include names, positions, departments, supervisors, tax numbers, salary level, tax class, health insurance deductions (200 bytes).

Changes possible:

- position names change due to standardization
- department specification changes to department plus division

These are irrelevant to salary, so we would like to guarantee that the salary programs work—even after the data representation changes. This is referred to as DATA INDEPENDENCE.

If tape codes the names in the first twenty bytes, the positiion in the second, etc. the program will need to be adjusted.

“Shoeboxes”

Consider multipurpose deployment of information on items sold in a bookstore

- ordering (“what’s missing?”)
- marketing (“what’s selling?”)
  - promotion (“who might buy it?” “when?”)
  - sales (get rid of *Ladenhüter*)
  - strategy (“which publishers, authors, topics, ... sell?”)
- scholarly (“what’s a bookstore carry in 19xx?”)
- security (“how much is lost due to theft?”)

Complication: access rights.
For example, who should have access to security information?

Moral: we need a flexible way of examining data.

RuG

Simple example

Purpose	Information					
inventory	ISBN nr.	current	re-order level	price		
invoicing	cust. name	addr.	invoice-nr.	item	quantity	total
credit	cust. name	addr.	acc't balance	last-paid	credit lmt	

- not exaggerated, a small business would also need to keep track of
  - personnel
  - expenses, incl. rent, taxes, service fees (accountant, cleaning), advertisement

Dangers

Purpose	Information					
inventory	stock nr.	current	re-order level	price		
invoicing	cust.-name	addr.	invoice-nr.	item	quantity	total
credit	customer	addr.	acc't balance	last-paid	credit lmt	

Redundancy?

Redundancy

Why is redundancy bad?

Ambiguity

- “stock nr.” vs. “item”, “cust.-name” vs. “customer”
- How is information to be interpreted?

Inconsistency

- Consistency depends on data entry
- Changes not local (“update propagation”)

Inefficiency

- Specifying (coding) something twice or more instead of once

Relational Db's

Relational DB's organize all data into redundancy-free TABLES

Purpose	Information					
inventory	stock nr.	current	re-order level	price		
invoicing	cust.-name	addr.	invoice-nr.	item	quantity	total
credit	customer	addr.	acc't balance	last-paid	credit lmt	

inventory	stock nr.	current	re-order level	price	
customer	name	addr.	credit lmt	acc't balance	last-paid
order (item)	name	stock nr.	quantity	total	

Relational Operations

Inl. IK

customer	name	addr.	credit lmt	acc't balance	last-paid
	S.Mahoney	7 Denfield Dr.	$ 500	$ 410.	2003-08-16
	M.Horan	26 Thompson Rd.	$ 500	$ 0.	2002-05-02
	J.Burckhalter	3114 Illinois Ave.	$ 500	$ 10.	1998-12-04
	⋮	⋮	⋮	⋮	⋮

SELECT reduces tables to rows fitting descriptions

`SELECT from CUSTOMER where ACCT-BALANCE` $\neq 0$ `AND LAST-PAID` $<$ `2002-00-00`

customer	J.Burckhalter	3114 Illinois Ave.	$ 500	$ 10.	1998-12-04

This operation provides a novel VIEW of the data, and thereby flexibility.

Relational Operations


```
SELECT from ORDER where ORDER.TOTAL > $ 100.
```

order	name	stock-nr.	quantity	total
	S.Mahoney	47A-1	500	$ 150.
	M.Horan	60AAA	5	$ 175.
	M.Horan	26ABC	1	$ 108.
	⋮	⋮	⋮	⋮

Results of relational operations are relations, subject to further operations:

```
100E-ORDERS ← SELECT from ORDER where ORDER.TOTAL > $ 100.
```

Relational Operations

Inl. IK

customer	name	addr.	credit-lmt	acc't-balance	last-paid
	S.Mahoney	7 Denfield Dr.	$ 500	$ 110.	2003-08-16
	M.Horan	26 Thompson Rd.	$ 500	$ 0.	2002-05-02
	J.Burckhalter	3114 Illinois Ave.	$ 500	$ 10.	1998-12-04
	⋮	⋮	⋮	⋮	⋮

PROJECT reduces tables to *columns* fitting descriptions

```
PROJECT NAME, ACCT'T-BALANCE from CUSTOMER
```

customer	S.Mahoney	$ 110.
	M.Horan	$ 0.
	J.Burckhalter	$ 10.

RuG

PROJECT result of SELECT

Results of relational operations are subject to further operations:

`100E-ORDERS ← SELECT from ORDER where ORDER.TOTAL > $ 100.`

order	name	stock-nr.	quantity	total
	S.Mahoney	47A-1	500	$ 150.
	M.Horan	60AAA	5	$ 175.
	M.Horan	26ABC	1	$ 108.
	⋮	⋮	⋮	⋮

`N-T-100E-ORDERS ← PROJECT NAME, TOTAL from 100E-ORDERS`

n-t-100E-order	name	total
	S.Mahoney	$ 150.
	M.Horan	$ 175.
	M.Horan	$ 108.
	⋮	⋮

Relational Operations

Combining tables with JOIN requires that some elements are identified uniquely throughout the database. These are KEYS.

inventory	stock nr.	current	re-order level	price	
customer	name	addr.	credit lmt	acc't balance	last-paid
order (item)	name	stock nr.	quantity	total	

JOIN combines tables based on keys.

Relational JOIN

Inl. IK

```
CUST-N-AB ⟵ PROJECT NAME, ACCT'T-BALANCE from CUSTOMER
```

cust-n-ab		
	S.Mahoney	$ 110.
	M.Horan	$ 0.
	J.Burckhalter	$ 10.

n-t-100E-order	
name	total
S.Mahoney	$ 150.
M.Horan	$ 175.
M.Horan	$ 108.
⋮	⋮

```
JOIN CUST-N-AB, N-T-100E-ORDERS where CUST-N-AB.NAME = N-T-100E-
ORDERS.NAME
```

Relational JOIN

Inl. IK

cust-n-ab

S.Mahoney	$ 110.
M.Horan	$ 0.
J.Burckhalter	$ 10.

n-t-100E-order

name	total
S.Mahoney	$ 150.
M.Horan	$ 175.
M.Horan	$ 108.
⋮	⋮

```
JOIN CUST-N-AB, N-T-100E-ORDERS where CUST-N-AB.NAME = N-T-100E-
ORDERS.NAME
```

join-cust-n-AB-t-100E

cust-n-AB.name (= n-t-100E-order.name)	n-t-100E-order.total	CUST-N-AB.ACC'T-BALANCE
S.Mahoney	$ 150.	$ 100.
M.Horan	$ 175.	$ 0.
M.Horan	$ 108.	$ 0.
⋮	⋮	⋮

RuG

Structured Query Language

IBM developed STRUCTURED QUERY LANGUAGE (SQL) to interact with databases.

- DECLARATIVE—order of execution not specified
- complete—covers all possible interactions
- standard defined by ANSI (American National Standards Institute)

Relational DB's

Relational model is intuitive to most users.

If relational DB is well-constructed (in redundancy-free way), then it guarantees consistency.

Table operations (SELECT, PROJECT, JOIN) sufficient for expression of *all* DB information.

Table operations also provide flexible views of data.

Well-designed, declarative language designed for interactive with databases — allowing access within programs, e.g., javascript.

Limitations of Relational Model

Some kinds of data are not modeled well in fixed-size relations

- recursive structure
  report everyone in N's management chain

$$\forall x(\text{manager}(x, N) \rightarrow \text{m-chain-N}(x))$$
$$\forall x, y(\text{m-chain-N}(x) \wedge \text{manager}(y, x) \rightarrow \text{m-chain-N}(y))$$

  TRANSITIVE CLOSURE

- texts (unlimited size)
  - string operations in most DB systems
- geography
  maps

Other DB models

Hierarchical Databases

- reflect hierarchies in organisation
- redundancy unavoidable
- small revival in web LDAP (light directory applications)

Network

- developed to solve problems in hierarchical systems
- complexity grows quickly

Logic (*addition* to relational core)

allows solution to transitive closure problem, but has never become popular

Object-Oriented (*addition* to relational core)

DB experts enthusiastic; allows tighter integration of procedures and data

JN: conflict with data independence?

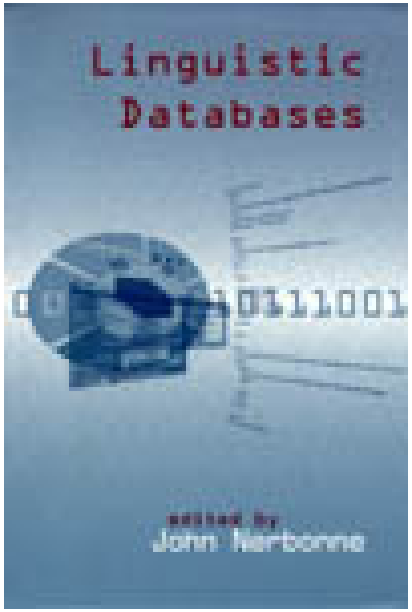
Groningen Work on DB's

Language Processing: Organizing language data.

Background: between 1900 and 1955 linguistics organized data into CORPORA, collections of texts. Chomsky called for an end to that methodology and suggested asking for native speaker judgements instead.

- in 1957 corpus-based techniques were slow, expensive
- intuition-based techniques confronted theory with much more data (Chomsky, 1957)
- the opposite is true today

Language DB's



- new focus on exploiting data
- intuition-based techniques not consistently applied, never systematically organized
- current work focuses on large texts *without* DB organization

Historical DB's

George Welling (Groningen) digitized and organized the import records (*Paalgeld*) of Amsterdam 1771-1817

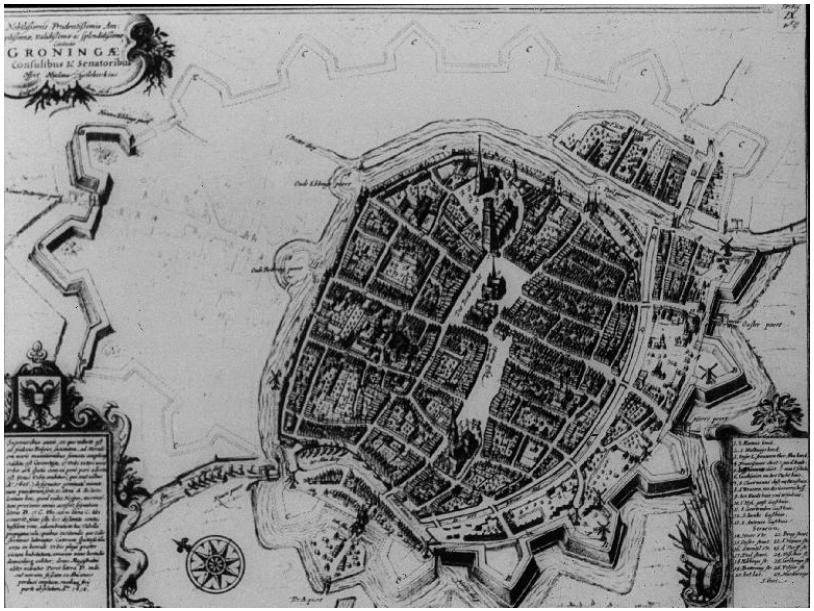
computational methods deployed for organization (database) and verification (consistency) and exploration (nominal record linkage)

historical results

- Older estimations of trade value distorted by counting ships and ignoring departure point
- Baltic trade ("moedernegotie") eclipsed by American trade even in 1771 (*pace* Israel, de Vries)
- American shipping took over Dutch business prevented by British blockade in 4th Anglo-Dutch war (1780-84)
  - American shipping catapulted to world-wide second place

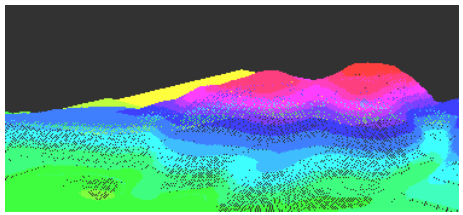
History of Architecture

Elwin Koster (Groningen) has digitized and organized city maps



computational methods used to reconstruct architectural work for which plans (and buildings) were inaccessible

Groningen Development



results in architectural history

- more complete reconstruction of urban development

A digital terrain model of 17th cent. Groningen

DB combined with map = Geographic Information System (GIS)

This uses standard DB technology, but also goes beyond it.

RuG