Functional Programming for Databases

Stefan Stanczyk
stef@brookes.ac.uk

OBJECTIVES

Presentation of the functional programming paradigm, its attributes and structures to achieve deeper insight into the database architecture and thus better design

REQUIRED KNOWLEDGE

- **Familiarity with database concepts and reasonable programming skills**
- **Certain mathematical ability (logic, discrete mathematics) beneficial**
- **No prior knowledge of functional programming or its theoretical basis**

CONTENT

- **Database foundations - concepts, structures, operations, behaviour.**
- **Data models for databases.**
- **Model of computation based on λ–calculus.**
- **Imperative versus declarative programming. Functional programming paradigm.**
- **Programming elements and components in a strongly typed FP language:**
  functions, primitive and defined types, overloading, guards, currying, recursion, list comprehension pattern matching, lambda expressions, higher order functions, type classes, algebraic types, infinite lists, sets, relations.
- **Data structures and abstract data types.**
- **Examples of classical algorithms expressed in a functional style.**
- **Examples of common database processes coded in Haskell.**
- **Functional database programming systems.**
- **Relational DB structures and operations expressed in a functional style.**

REFERENCES

S Thompson, *Haskell - The Craft of Functional Programming*, 2nd ed., Addison-Wesley, 1999

www.cs.ukc.ac.uk/people/staff/sjt/craft2e & www.haskell.org

F Rabhi, G Lapalme, *Algorithms: a Functional Programming Approach*, 2nd ed., Addison-Wesley, 1999

J R Abrial, *Data Semantics*, Proceedings of IFIP Working Conference on DB Management, North Holland, 1974

D Shipman, *The Functional Data Model and the Data Language DAPLEX*, ACM TODS, Vol.6, No 1, March 1981, pp. 140-173

S Stanczyk, et al, *Theory & Practice of Relational Databases*, Taylor & Francis, 2001

database recap

database philosophy

data design separated from process design

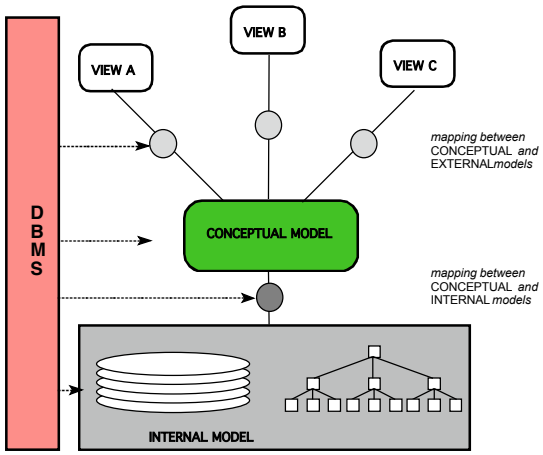
↓

the effect of process structure omitted for data design

↓

relationships between data and processes are of the *first-order* type thus the final *technical* design is achieved by superposition

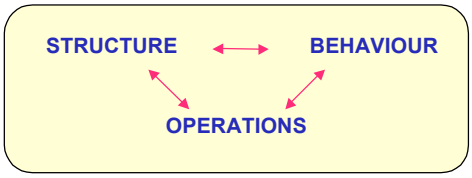
database architecture



benefits

- Minimised data redundancy
- Data shared amongst applications
- Data maintained centrally
- Common processes between applications
- Application software transparent

relational model



structure

a relation - a constrained subset of a product of simple domains

$R = (r_1, r_2, \ldots r_{i-1}, r_i, r_{i+1}, \ldots r_n)$

r_1	r_2	r_3	r_4	.....	r_n

- none of r_i is a structure itself — 1NF

behaviour

- $\exists\ k = (r_i, r_j, \ldots)$: $[\Pi R(k)] \equiv [R]$ — superkey → identifier (PK)

r_1	r_2	r_3	r_4	.....	r_n

behaviour

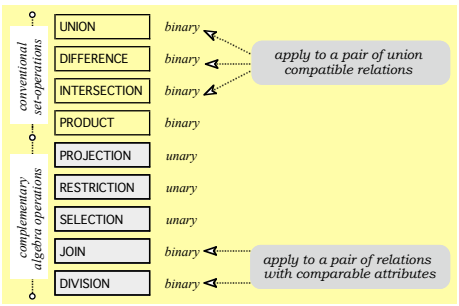
R(X, Y, Z) and S (T, U, V) :

S.V **ref** R.X **iff**:

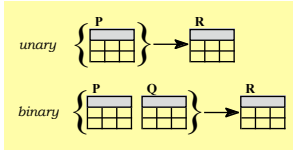
- X is identifier in R
- ∀ v ∈ V, ∃ x ∈ X : v = x
- πS (V) = πR (X)

operations

relational algebra



- $\{\mathbf{R}_i\}$ **is closed under Π, σ, η, ...**



optimisation

relations $\{R_i\}$ with some undesirable properties
↓
relations $\{S_k\}$ with better update properties

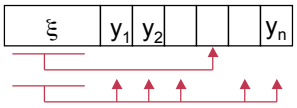
- Update transactions must not
  - cause loss of information
  - violate entity integrity
  - carry any risk of inconsistent updating
- Data redundancy minimised

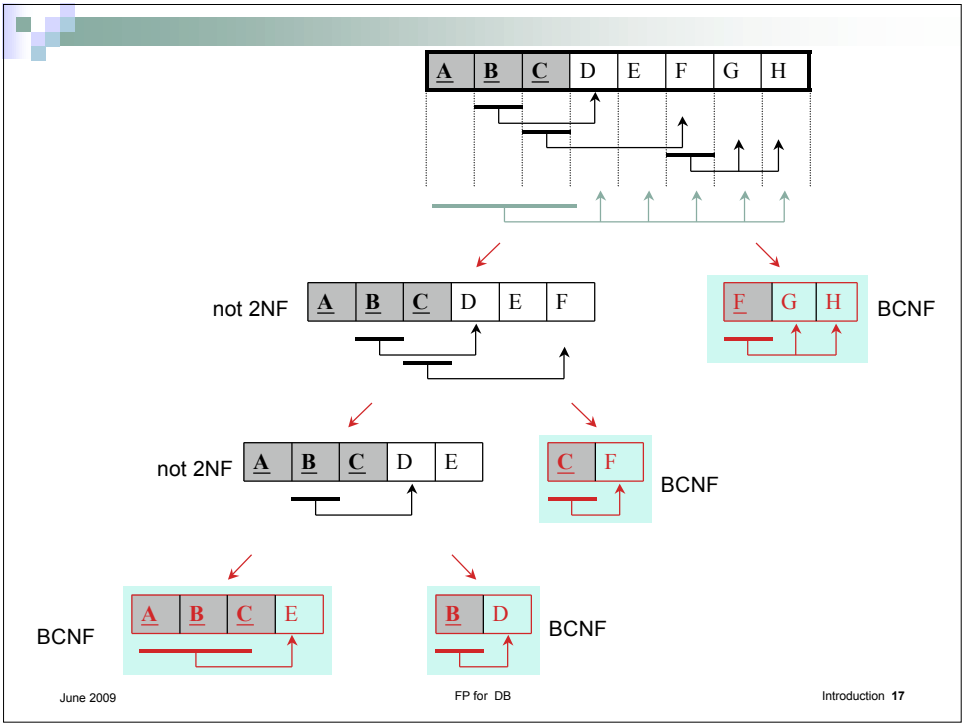
optimisation

FD: $X \rightarrow Y$ holds for $\mathbf{R} = (..., X, Y, ...)$
iff $\forall\, x \in X,\ [\Pi\sigma\, \mathbf{R}(X{=}x)Y] \leq 1$

functional dependency

- BCNF
  $\xi \rightarrow y_i \Rightarrow \xi \rightarrow y_k\ \forall\ k = 1 .. n$





observation

The process of normalisation is

- **discrete**
- **deterministic**
- **guaranteed to terminate**

➡ **when FD are found, the decomposition mechanism is detached from semantics**

Though decomposition is mechanistic, the results are semantically Interpretable - when the decomposition is completed, the objects get the meaning re-assigned (akin to e.g. algebraic transformations).

conclusion

All characteristics, properties, processes of RDB

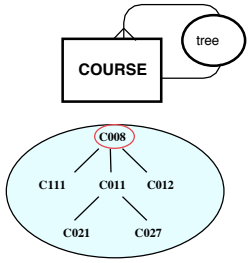
- The relation (whether 1NF or not)
- Primary Key & Entity integrity
- Foreign Keys & Referential Integrity
- FD, MVD, JD, PD
- Normal Forms
- Decomposition

are expressible as functions

Why relational model has been so attractive ? ☺

- **separation of physical & logical aspects**
- **data - process independence**
- **high level of data abstraction**
- **universal & uniform data structure**
- **global behavioural rules**
- **set of higher-level operations**
- **structure optimisation algorithm**

representing intra-structures



CODE	TITLE	PREQ
C008	.....	none
C111	.....	C008
C011	.....	C008
C012	.....	C008
C021	.....	C011
C027	.....	C011

p ::= remove **C008**

any q(PC(CODE, DETAILS))
has to wait until p terminates

non-FD decomposition

CODE	TITLE

CODE	PREQ

conclusion

RDB insufficient in terms of provision for recursive structures and processes

- Convoluted structures embedded in relations (tree, graph) are not directly supported in RDB
- Recursive processing not supported either

A consequence of algebraic foundations addressed by e.g. recursive union in SQL

conclusion

any kind of ordering (set inclusion, tree, graph, convolution) imposed on a structure contradicts relational foundations

→ **evolution of RDB imminent**

principles for futher development

- **structural simplicity → structural regularity**
- **separation of logical and physical aspects of database processing**
- **set-oriented processing → algebra-oriented processing**

R&D progress

{ADT, procedures}

↓

{relations, r-operations} E. Codd

↓

{nested relations, xr-operations} H. Korth *at al.*

↓

{algebra (components, operations), transformations} ?