3. Database Design

This is the Database Design course theme.
[Complete set of notes PDF 295Kb].

3.1. Functional Dependency

In this lecture we look at...
[Section notes PDF 64Kb]

3.1.01. Introduction

- What is relational design?
  - Notion of attribute distribution
  - Conceptual-level optimisation
- How do we asses the quality of a design?

3.1.02. Value in design

- Allocated arbitrarily by DBD under ER/EER
- Goodness at
  - Internal/storage level (base relations only)
    - Reducing nulls - obvious storage benefits /frequent
    - Reducing redundancy - for efficient storage/anomalies
  - Conceptual level
    - Semantics of the attributes /single entity:relation
    - No spurious tuple generation /no match Attr,-PK/FK

3.1.03. Initial state

- Database design
- Universal relation
  - $R = \{A_1, A_2, \ldots, A_n\}$
  - Set of functional dependencies F
- Decompose R using F to
  - $D = \{R_1, R_2, \ldots, R_n\}$
  - D is a decomposition of R under F

3.1.05. Aims

- Attribute preservation
  - Union of all decomposed relations = original
- Lossless/non-additive join
  - For every extension, total join of $r(R_i)$ yeilds r(R)
  - no spurious/erroneous tuples

3.1.06. Aims (preservation)

- Dependency preservation
  - Constraints on the database
    - X -> Y in F of R, appears directly in R_i
  - Attributes X and Y all contained in R_i
  - Each relation R_i in 3NF
- But what's a dependency?

3.1.07. Functional dependency

- Constraint between two sets of attributes
  - Formal method for grouping attributes
- DB as one single universal relation/-literal
  - $R = \{A_1, A_2, \ldots, A_n\}$
  - Two sets of attributes, X subset R,Y subset R
- Functional dependency (FD or f.d.) X -> Y
- If $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$
  - Values of the Y attribute depend on value of X
  - X functionally determines Y, not reverse necessarily

3.1.08. Dependency derivation

- Rules of inference
- reflexive: if X implies Y then X -> Y
- augment: {X -> Y} then XZ -> YZ
- transitive: {X -> Y,Y -> Z} then X -> Z
- Armstrong demonstrated complete for closures

3.1.09. Functional dependency

- If X is a key (primary and/or candidate)
  - All tuples t_i have a unique value for X
  - No two tuples (t_1,t_2) share a value of X
- Therefore X -> Y
  - For any subset of attributes Y
- Examples
  - SSN -> {Fname, Minit, Lname}

- {Country of issue, Driving license no} -> SSN
- Mobile area code -> Mobile network (not anymore)

3.1.10. Process

- Typically start with set of f.d., F
  - determined from semantics of attributes
- Then use IR1,2,3 to infer additional f.d.s
- Determine left hand sides (Xs)
  - Then determine all attributes dependent on X
- For each set of attributes X,
  - determine X+ :the set of attributes f.d'ed by X on F

3.1.11. Algorithm

- Compute the closure of X under F: X+
  - xplus = x;
  - do
    - oldxplus = xplus;
    - for (each f.d. Y -> Z in F)
      - if (xplus implies Y) then
      - xplus = xplus union Z;
  - while (xplus != oldxplus);

3.1.12. Function dependency

- Consider a relation schema R(A,B,C,D) and a set F of functional dependencies
  - Aim to find all keys (minimal superkeys),
  - by determining closures of all possible X subsets of R's attributes, e.g.
    - A+, B+, C+, D+,
    - AB+, AC+, AD+, BC+, BD+, CD+
    - ABC+, ABD+, BCD+
    - ABCD+

3.1.13. Worked example

- Let R be a relational schema R(A, B, C, D)
- Simple set of f.d.s
- AB -> C, C -> D, D -> A
- Calculate singletons
  - A+, B+, C+, D+,
- Pairs
  - AB+, AC+,…

- Triples
  - and so on

3.1.14. Worked example

- Compute sets of closures
  - AB -> C, C -> D, D -> A
- 1.Singletons
  - A+ -> A
  - B+ -> B
  - C+ -> CDA
  - D+ -> AD
- Question: are any singletons superkeys?

3.1.15. F.d. closure example

- 2.Pairs (note commutative)
  - AB+ -> ABCD
  - AC+ -> ACD
  - AD+ -> AD
  - BC+ -> ABCD
  - BD+ -> ABCD
  - CD+ -> ACD
- Superkeys?

3.1.16. F.d. closure example

- 3.Triples
  - ABC+ -> ABCD
  - ABD+ -> ABCD
  - BCD+ -> ABCD
- Superkeys? Minimal superkeys (keys)?
- 4.Quadruples
  - ABCD+ -> ABCD

3.1.17. F.d. closure summary

- Superkeys:
  - AB, BC, BD, ABC, ABD, BCD, ABCD
- Minimal superkeys (keys)
  - AB, BC, BD

3.2. Normal Forms

In this lecture we look at...
[Section notes PDF 121Kb]

3.2.01. Orthogonal design

- Information Principle:
  - The entire information content of the database is represented in one and only one way, namely as explicit values in column positions in tables
- Implies that two relations cannot have the same meaning
  - unless they explicitly have the same design/attributes (including name)

3.2.02. Normalization

- Reduced redundancy
- Organised data efficiently
- Improves data consistency
  - Reduces chance of update anomalies
  - Data duplicated, then updated in only one location
- Only duplicate primary key
  - All non-key data stored only once
- Data spread across multiple tables, instead of one Universal relation R

3.2.03. Good or bad?

- Depends on Application
- OLTP (Transaction processing)
  - Lots of small transactions
  - Need to execute updates quickly
- OLAP (Analytical processing/DSS)
  - Largely Read-only
  - Redundant data copies facilitate Business Intellegence applications, e.g. star schema (later)
- 3NF considered 'normalised'
  - save special cases

3.2.04. Normal forms (1NF)

- First Normal form (1NF)
  - Disallows multivalued attributes
  - Part of the basic relational model
- Domain must include only atomic values
  - simple, indivisible
- Value of attribute-tuple in extension of schema
- $t[A_i] \in (A_i)$

3.2.05. Normalisation (1NF)

- Remove fields containing comma separated lists
- Multi-valued attribute (A_{MV}) of R_i
- Create new relation (R_{NEW})
  - with FK to R_i[PK]
  - R_{NEW}(UID, A_{MV}, FK_I)

3.2.06. Normalisation (1NF)

- On weak entity
- On strong entity

PersonDietWeak

Person	Dietary requirements
Bob	No eggs
Fred	No meat, diary or gluten
Jamal	No fish

->

PersonDietWeak1NF

Person	Dietary requirements
Bob	No eggs
Fred	No meat
Fred	No dairy
Fred	No gluten
Jamal	No fish

PersonDietStrong

ID	Person	Dietary requirements
1	Bob	No eggs
2	Fred	No meat, diary or gluten
3	Jamal	No fish

->

PersonDiet1NFK

ID	Person	DietFK
1	Bob	1
2	Fred	2
3	Jamal	3

FK(DietFK) to Diet1NFK(ID)

Diet1NFK

ID	Requirement
1	No eggs
2	No meat
2	No dairy
2	No gluten
3	No fish

3.2.07. Normal forms (2NF)

- A relation R_i is in 2NF if:
  - Every nonprime attribute A in R_i is
  - fully functionally dependent on 1y key of R
- If all keys are singletons, guaranteed
- If R_i has composite key are
  - all non-key attributes fully functionally dependent
  - on all attributes of composite key?

3.2.08. Normal forms (2NF)

- Second normal form (2NF)
  - Full functional dependency $X \rightarrow Y$
    - $A \in X$, $(X - \{A\}) \neg\rightarrow Y$

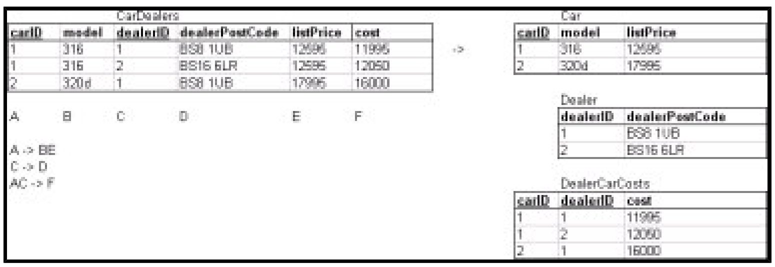
- If any attribute A is removed from X
- Then X → Y no longer holds
  - Partial functional dependency
  - A ∈ X, (X - {A}) → Y

3.2.09. Normal forms (2NF)

- In context
  - Not 2NF: AB → C, A → C
    - AB → C is not in 2NF, because B can be removed
  - Not 2NF: AB → CDE, B → DE
    - because attributes D&E are dependent on part of the composite key (B of AB), not all of it

3.2.10. Normalisation (2NF)

- Split attributes not depended on all of the primary key into separate relations



CarDealers

carID	model	dealerID	dealerPostCode	listPrice	cost
1	316	1	BS8 1UB	12595	11995
1	316	2	BS16 6LR	12595	12050
2	320d	1	BS8 1UB	17995	16000
A	B	C	D	E	F

A -> BE
C -> D
AC -> F

->

Car

carID	model	listPrice
1	316	12595
2	320d	17995

Dealer

dealerID	dealerPostCode
1	BS8 1UB
2	BS16 6LR

DealerCarCosts

carID	dealerID	cost
1	1	11995
1	2	12050
2	1	16000

3.2.11. Normal forms (BCNF)

- Boyce-Codd Normal form (BCNF)
  - Simpler, stricter 3NF
    - BCNF → 3NF
    - 3NF does not imply BCNF
  - nontrivial functional dependency X → Y
  - Then X must be a superkey

3.2.12. Normal forms (3NF)

- Third Normal form (3NF)
- Derived/based on transitive dependency
- For all nontrivial functional dependencies
  X → A
- Either X must be a superkey
- Or A is a prime attribute
  (member of a key)

3.2.13. Normal forms in context

- AB → C, C → D, D → A
- In context
  - 3NF? Yes
    - Because AB is a superkey and
    - D and A are prime attributes
  - BCNF? No
    - Because C and D are not superkeys
    - (even though AB is)

3.2.14. Normalisation (3NF)

- CarMakes not in 3NF because:
  - singleton key A
  - non-trivial fd B → C
    - B not superkey, C not prime attribute

CarMakes

carID	make	makeHeadOffice
1	Audi	NW1 8TQ
2	BMW	SW4E 9GG
3	Ford	LE17 9EE
A	B	C

A -> BC
B -> C

->

Car

carID	make
1	Audi
2	BMW
3	Ford

FK(make) to Make(make)

Make

make	makeHeadOffice
Audi	NW1 8TQ
BMW	SW4E 9GG
Ford	LE17 9EE

3.3. OODB

In this lecture we look at...
[Section notes PDF 34Kb]

3.3.01. Introduction

- Database architectures, beyond
- Why OODBMS?
- ObjectStore
- CORBA object distribution standard

3.3.02. Large DBMS

- Complex entity fragmentation

    - across many relations
- Breaks the miniworld-realworld dichotomy
- Requires conceptual abstracting layer
- Difficult to retrieve all information for x
- Compounded by version control

3.3.03. Object orientation

- Object components (icv triples)
    - Object Identity (OID), I
        - replaces primary key
    - Type constructor, c
        - how the object state is constructed from sub-comp
        - e.g. atom, tuple (struct), set, list, bag, array
    - Object state, v
    - Object behaviour/action

3.3.04. Desirable features

- Encapsulation
    - Abstract data types
    - Information hiding
- Object classes and behavior
    - Defined by operations (methods)
- Inheritance and hierarchies
- Strong typing (no illegal casting)
    - don’t think about inheritance just yet

3.3.05. Desirable features

- Persistence
    - Objects exist after termination
    - Naming and reachability mechanism
    - Late binding in Java
- Performance
    - user-def functions executed on server, not client
- Extension into relational model
    - Domains of objects, not just values
    - Domain hierarchies etc.

3.3.06. Desirable features

- Polymorphism
    - aka. operator overloading

  - same method name/symbol
  - multiple implementations
- Easy link to Programming languages
  - popular OO language like Java/C++
  - Better than PL/SQL integration
  - Much better than PL-JDBC integration!
- Highly suitable for multimedia data

3.3.07. Undesirable features

- Object Ids
  - Artificial Real-Mini, double edged sword
  - Exposes inner workings (suspended abstraction)
- Lack of integrity constraints
- No concept of normalisation/forms
- Extreme encapsulation
  - e.g. creation of many accessor/mutator methods
- Lack of (better) standardisation

3.3.08. More undesirables

- Originally no mechanism for specifying
- relationships between objects
- In RDBMS – relationships are tuples
- In OODBMS – relationships should be properties of objects

3.3.09. ObjectStore

- Packages supporting Java or C++
- C++ package uses:
  - C++ class definition for DDL
  - insert(e), remove(e) and create collections, for DML
- Bidirectional relationship facility
- Persistency transparency
  - Identical pointers to persistent and transient objects

3.3.10. CORBA

- Common Object Request Broker Architecture
- Object communication (unifying paradigm)
  - Distributed
  - Heterogeneous
  - Network, OS and language transparency
- Java implementation org.omg.CORBA

- Also C, C++, ADA, SMALLTALK

3.4. Type Inheritance and EER diagrams

In this lecture we look at...
[Section notes PDF 117Kb]

3.4.01. Introduction

- Design/schema side (Entity types)
- Object-orientated concepts
  - Java, C++ or UML
  - Sub/superclasses and inheritance
- EER diagrams
- EER to Relational mapping

3.4.02. OO

- Inheritance concept
  - Attributes (and methods)
- Subtypes and supertypes
- Specialisation and Generalisation
- ER diagrams
  - show entities/entity sets
- EER diagrams
  - show type inheritance
  - additional 8^{th} step to ER→Relational mapping

3.4.03. Objects

- Basic guide to Java
- Object, classes as blueprints
- Object, collection of methods and attributes
- Miniworld model of real world things
- Object, entity in database terms

3.4.04. Abstract

- Similar objects
- Car Park example
- Student example
- Shared properties/attributes
- Generalisation

- Reverse, specialisation

3.4.05. Relationships

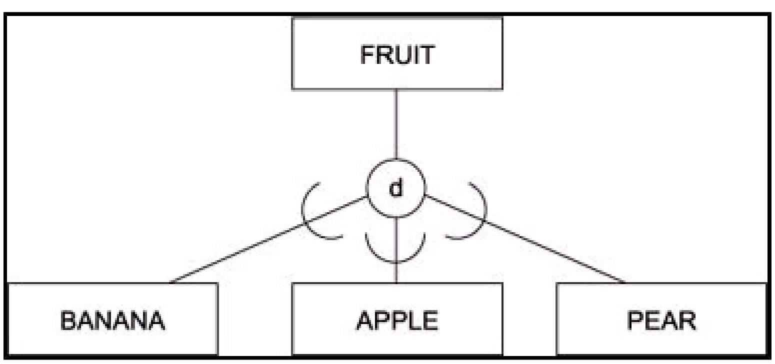
- Using English as model
- ‘Is a’ (inheritance)
- ‘Has a’ (containment)
- Nouns as objects
- Verbs as methods
- Adjectives as variables (sort of)

3.4.06. Classes

- Superclasses (Student)
- Subclasses (Engineer, Geographer, Medic)
- Inheritance
- Subclass inherits superclass attributes
  - Union of specific/local and general attributes
- Inheritance chains
  - Person → Student → Engineer → Computer Scientist

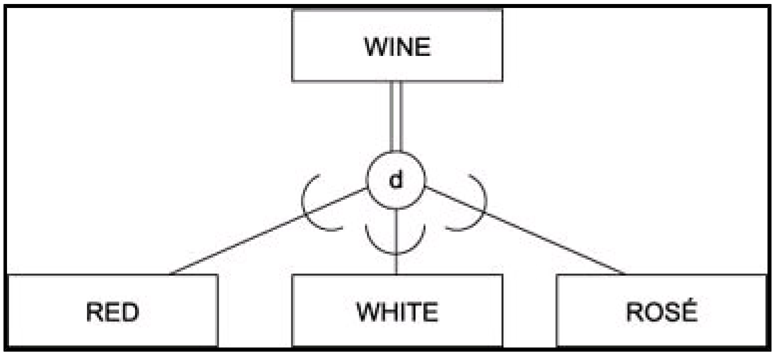
3.4.07. EER Fruit example

- Partial participation
- Disjoint subclasses
- A fruit may be either a pear or an apple or a banana, or none of them. A fruit may not be a pear and a banana, an apple and a banana, an apple and a pear …



3.4.08. EER Wine example

- Total, disjoint
- Equivalent to Java Abstract classes
- A Wine has to be either Red, White or Rosé cannot be both more



3.4.09. More extended (EER)

- Specialisation lattices
  - and Hierarchies
- Multiple inheritance
- Union of two superclasses (u in circle)
- In addition to basic ER notation

3.4.10. EER diagramatic notation

- Subset symbol to illustrate
- sub/superclass relationship
- direction of relationship
- Circle to link super to subclasses
  - Disjoint
  - Overlapping
  - Union

3.4.11. Disjointness constraint

- Disjointness (d in circle) – single honours
- Overlapping (o in circle) – joint honours/sports
- Membership condition on same attribute
  - attribute-defined specialisation
  - defining attribute
  - implies disjointness
- versus user-defined
  - each entity type specifically defined by user

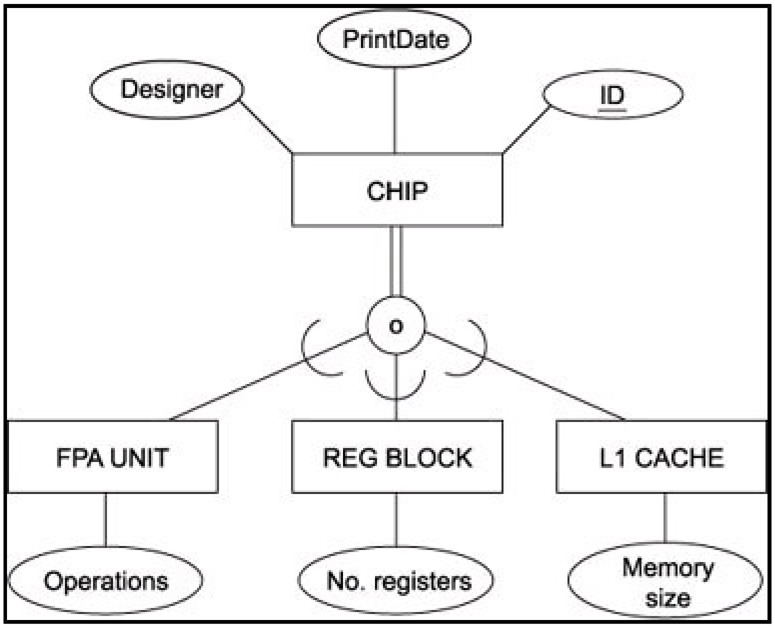
3.4.12. Completeness constraint

- Total specialisation
  - Every entity in the superclass must be a member of atleast 1 subclass
  - Double line (as ER)

- Partial specialisation
  - Some entites may belong to atleast 1 subclass, or none at all
  - Single line
- Yields 4 possibilities
  - (Total-Dis, Total-Over, Partial-Dis, Partial-Over)

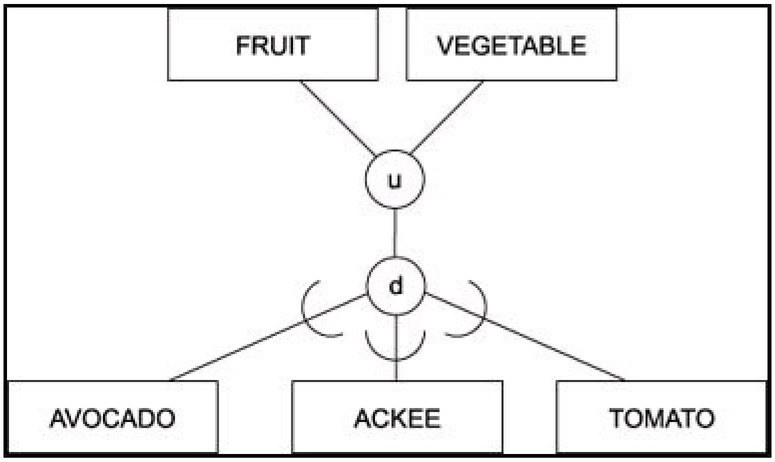
3.4.13. EER Chip example

- Total, overlapping
- A Chip may has to be at least one of FPA Unit, Reg Block, L1 Cache, and may be more than one type



3.4.14. EER Multiple inheritance

- Type hierarchies
- Specialisation lattices
- *Well, sir, the Supreme Court of the United States has determined that the tomato is for legal and commercial purposes both a fruit and a vegetable. So we can legally refer to tomato juice as 'vegetable' juice.*
- Candice, General Foods



3.4.15. EER to Relational Mapping

- Initially following 7 ER stages
- Stage 8
- 4 different options
  - Optimal solution based on problem
- Let C be superclass, $S_{1..m}$ subclasses

3.4.16. Stage 8

- Create relation for C, and relations for $S_{1..m}$ each with a foreign key to C (primary key)
- Create relations for $S_{1..m}$ each including all attributes of C and its primary key

3.4.17. Stage 8

- Create a single relation including all attributes of $C \cup S_{1..m}$ and a type/discriminating attribute
  - only for disjoint subclasses
- Create a single relation as above, but include a boolean type flag for each subclass
  - works for overlapping, and also disjoint

3.5. System Design

In this lecture we look at...
[Section notes PDF 64Kb]

3.5.01. Databases in Application

- Where's the data?
- Programmer driven future

- OODBMS limitations
- RDBMS longevity
- System design by
  - Data store, delivery, interface
- Case study

3.5.02. Where’s the data?

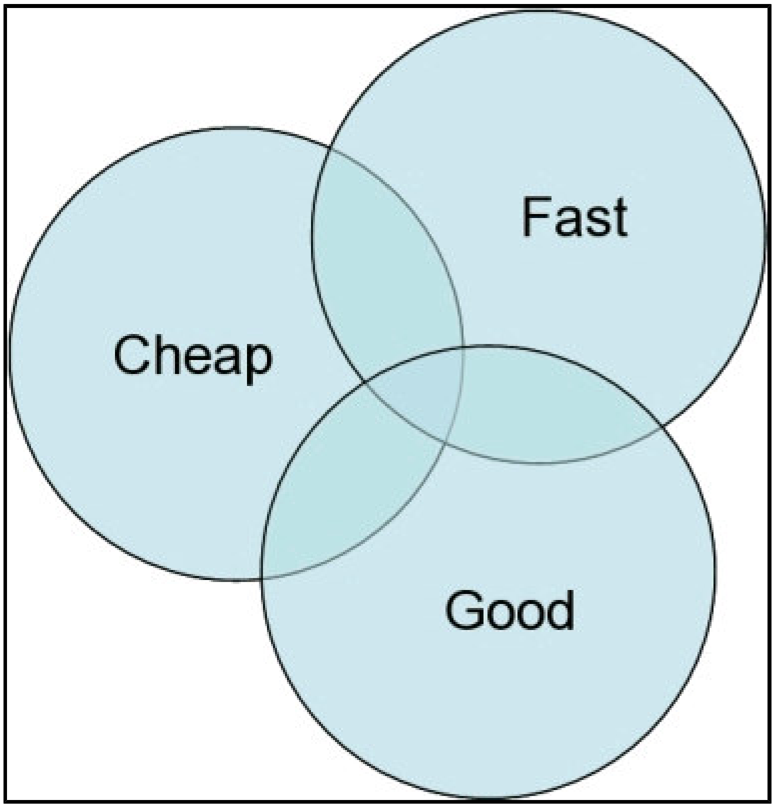
- Previously covered distance from User to Data (and reason for it)
- Client-Server data model creates DBMS
  - P2P alternative
- Accountability
- Distribution (BitTorrent, eDonkey)
- Caching

3.5.03. Where’s the data?

- Answer: everywhere
- But where is it meaningful?
- Answer: for whom?

3.5.04. Quality paradigm

- Large projects require large teams
- Team overhead (ref 2nd year)
- Code responsibilities
- Data/data model resp.
- Object responsibilities



3.5.05. Web application data support

- Web application programming
- Goal, dynamically produced XHTML
- Client side designer-programmer split
  - CSS, XHTML
- Server side programmer-programmer split
  - Old school: query design, integrator
  - New school: MVC (Model-View-Controller)
    - Controller – user input
    - Model – modelling of external world
    - View – visual feedback

3.5.06. CMS

- Content Management System
- part of other courses
- CMS is a DBMS
- Zope/Plone and ZODB
- e107, Drupal and Seagull
- Zend MVC Framework (pre-beta)

3.5.07. OODBMS limitations

- Future unknown
- RDBMS supports
  - Application data sharing
  - Physical/logical data independence/views
  - Concurrency control
  - Constraints
- at inception these requirements not known
- RDBMS mathematical basis → extensible
- Crude Type Inheritance (see EER mapping)
- OODBMS as construction kit

3.5.08. Weaknesses in RDBMS

- Data type support
- Unwieldy, created 3VL (nulls)
- Type Inheritance and Relationships
- Tuple:Entity fragmentation
  - not to be confused with ‘fragmentation’
- Entity approximation requires joins

3.5.09. System design

- Client specifications
- Variance amongst Mobile devices
- Rich-media Content delivery
- Where’s the data? (M – media database)
- Where’s it going? (C – mobile browser)
- How’s it going to get there? (query design)
- What’s it going to look like? (V – XHTML)

3.5.10. Muddy boots

- The real world of databases
- Massive Excel spreadsheets
- Access Migration
- Normalisation
- Update implications
- Visual language of the Internet limitations
- Future of browser components