1. Data models

This is the Data models course theme. In this section we introduce concepts for modelling data and language for communicating those models.

1.1. Introduction

Databases are pervasive in modern society. So many of our actions and attributes are logged and stored in organised information repositories, or Databases.

1.1.01. Databases

- Where do we come into contact with databases?
- Supermarket inventories/EPOS
  - Supplies, shopping habits, store locations, accounts
- Films (iMDB)
  - Cast lists, shooting schedules, histories, budgets
- Department
  - Students, courses, staff, payroll

These are all examples of relatively simple databases. All of the information is textual or referential.

1.1.02. New technologies

- Not just traditional, numeric/textual
- Research 70s biased
- Digital media
  - Video servers (atom/bbc/youtube)
- Multimedia databases
  - Web site, collection of diverse data types
  - Google, AltaVista
- Stock Exchange
  - Futures, Currency markets, trends
  - Databases comprising not only data, but modelling algortihms
- Microsoft's WinFS

Databases don't have to store just text. Increasingly Database servers are storing, indexing and delivering rich-media content, explicitly images, audio and video.

Microsoft's next generation File storage system (WinFS) is a relational database. From a user perspective, searching (the process of indexing content by keyword) is already mainstream. Users are moving away from rigid directory structures (files and folders) and towards keyword-tagged content.

1.1.03. Variations

- Not only in role
- Size
  - Video server example
- Complexity
  - Ford Puma™ example
  - Ford staff organised by production line and car
  - e.g. Each staff member answers to a 'part' manager (engines, bodyshell, chassis) and a 'car' manager (Puma, Mondeo, Ka)
- Expense
- User profiles and demands

We've seen that databases are used in a variety of contexts. Those roles imply properties of each of the systems. An interlinked text-only database (such as Unix/Linux's MAN pages) will require much less storage than a video archive.

Some databases are perceptually more complex. Ford's staff management model would be represented as a matrix (in this case 2 dimensional). Computers are very good at organising multi-dimensional space.

1.1.04. Definition

- Embrace diversity
- Data and semantic
- Database: related data, implicit meaning
- Sample/subset of real world
  - real world with bounds
- Miniworld/Universe of Discourse

A single definition of a database is hard to come by. Dictionary.com defines a database as: a comprehensive collection of related data organized for convenient access, generally in a computer. The Wikipedia definition runs for several pages.

1.1.05. Abstraction

- Previous lab exercises
  - Problem: reading data from a file
- Abstraction theme
- Basis of good OOP and further good P
  - from encapsulation to software component analysis
- Layering, splitting data from design
  - Solution: grammar (language guide) and data

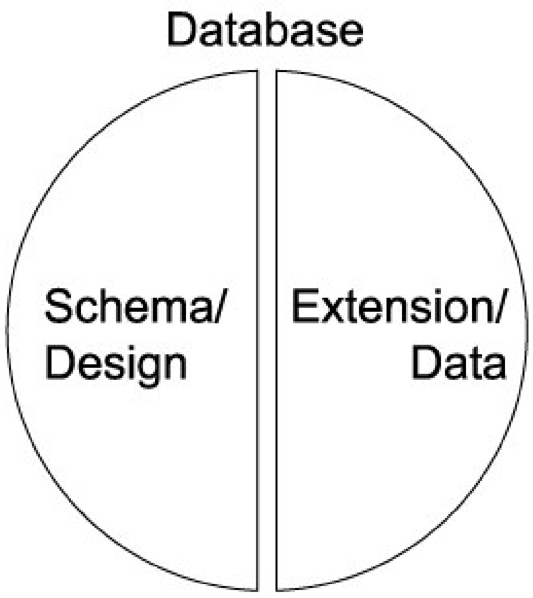
In some of your previous lab assignments, or practical experience, you may have been faced with the problem of caching information persistently in a file, later to be reloaded.

When writing the data into the file, we are storing more than just that information. We are storing implicitly a design/grammar for that data. That implicit design is evident when accessing the file with a naive interface. If you try to read the data out in a different order, it fails.

A better solution is to split the way the information is stored from the actually information stored.

1.1.06. Data-design divide

- Left-hand/right-hand divide
- LHS: Catalog
  - or Meta-data
  - or Intension
  - or Schema
  - i.e. the Design of database
  - Types of data, organisation, constraints
- RHS: Extension
  - or Snapshot
  - The data itself
  - Information stored in the database
  - Tuples



1.1.07. DBMS

- DataBase
- Management
- System
- Collection of programs that enable users to:
  - Define - patterns, boundaries, design
  - Construct - populate to go live
  - Manipulate - runtime changes
- data in a structured, organised store.

1.1.08. Database Management Systems

- Properties
- Data models and independence
    - Requirements
    - Categorisation of Database users
- DBMS components
    - Architecture

When considering the database systems as a whole, we need to look at all the components, including elements that interact with the DBMS (users, whom we categorise for simplicity).

This course will contain a discussion of the components that make up the system and the way they interact (system architecture).

1.1.09. Models

- Data models
- Relational data model (Oracle)
- Object data model (ObjectStore)
- Legacy systems
    - Hierarchical data model
    - Network data model

A data model is an invention. It is a construct that allows us to share an understanding of how the system works. As with all good constructs, it's an abstraction; a simplification; a story.

In this course we're going to look at the Relational model, where the database is organised into tables (relationals) and each row (tuple) within that relation is coded (keyed) to allow referencing between the relationals.

The Relational model, inspite of being innovated in the 1970s is still the most popular, underpining mainstream modern databases such as Oracle 10i and MySQL 5.0

As programming languages are becoming increasingly Object orientated, programmers require a means of persistently storing their Objects. Object Orientated Databases (OODBs) exist to fulfil this purpose. OODBs may ultimately replace relational databases, but it's not clear at this stage when.

1.1.10. Other properties

- Beside Data model
- Number of users
- Number of sites
- Cost
- Types of access paths
- Generality, or inversely specificity

MySQL is a highly general database system, in that it supports many different designs. My mobile phone address book is a highly specific database system and as such is not easily extensible.

1.1.11. Independence

- Based on File processing
- Data definition implicit in
    - Data
    - Application program
- Example of one specific database
- Structure embedded into access program
    - Coursework code re-use example

Earlier I made mention of this problem. Databases tie into the wider Software Engineering field. Within Software Engineering, post-development issues of code re-use, maintenance, future evolution etc. necessitate a logical flexible approach to program design. Databases are such an approach. In order to store information in a database you invest a small amount of time in explicitly structuring it, however you then get things like flexibility (data independence etc.) for free.

1.1.12. Program-data independence

- General databases
    - Separate Data definition and data
    - Catalog/meta-data & tuples
- Data format/structure stored separately
- Program-data independence (e.g. Y2K)
    - Changes in data format
        - Alter data (tuples)
        - Alter grammar (catalog/meta-data)
    - We actually split program-data independence into:
        - Logical
        - Physical
    - ...more in a second

1.1.13. Program-operation independence

- In object-orientated databases
    - Objects consist of attributes and operations
    - Operation defined by
        - Header/Interface/Prototype or Signature
        - Implementation
    - Program-operation independence
        - Implementation change hidden from user
    - Collectively data abstraction

1.1.14. Logical and physical program-data independence

- Three tier data-model diagram
- Mappings, Data independence
    - Logical
        - between conceptual and external
        - changes to conceptual without changing
            - external schemas or application programs
    - Physical
        - between internal and conceptual
        - changes to internal without changing
            - conceptual or external schemas

1.1.15. DBMS Requirements

- DataBase Management System
    - Abstraction (i.e. program-data independence)
    - Conceptual representation (data models)
    - Multiple views and User Interfaces
    - Data sharing and transaction processing
    - Access restriction
    - Redundancy removal/optimisation
    - Persistent storage (Program objects) & Integrity
    - Relationship management & Inference
    - Backup and recovery

Here's a summary of what we need from a DBMS

1.1.16. Database Users

- Database as 1y resource, DBMS as 2y
- Database administrators (DBA)
- Database designers
- End users
    - Naïve - canned transactions e.g. bank/airline
    - Sophisticated - engineers, scientists, query editors
    - Stand-alone (personal databases/MS Access)

1.1.17. Results of use

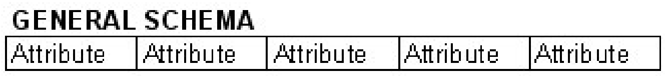
- Knock-on effects of database approach
- Enforcing standards
- Reduced Development time

- Adaptive to change (design changes)
- Up-to-date information (live database)
- Economies of scale
  - centralising commonly required resources

...and this is what you get for free. These are the consequences, largely positive, of adopting a database approach to an information storage problem.

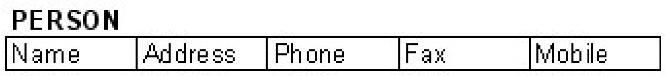
1.1.18. Design side

- Meta-data
  - Database schema, intension
  - Data Model
  - Left hand side of database divide
- Schema diagram
- Entity-relationship (ER diagrams)
- UML diagrams



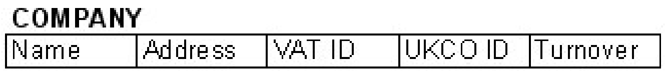
GENERAL SCHEMA

Attribute	Attribute	Attribute	Attribute	Attribute



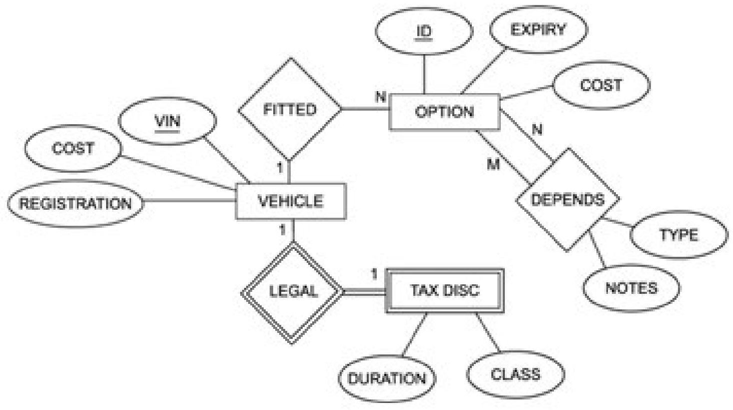
PERSON

Name	Address	Phone	Fax	Mobile



COMPANY

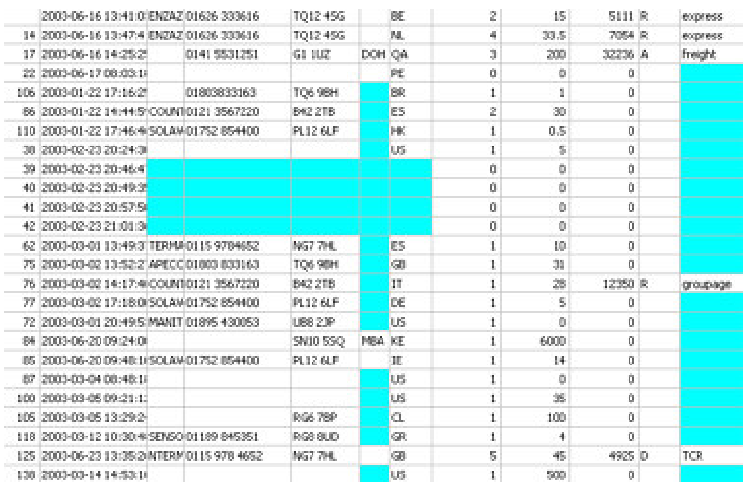
Name	Address	VAT ID	UKCO ID	Turnover



1.1.19. Data side

- Data, under the column heading
- Less easy to look at (volume issue)
- Fundamentally less interesting (more specific)
- Variety of tools for looking at it:

- HeidiSQL, PhpMyAdmin, Sword
- Here's what a snapshot looks like:



	2003-06-16 13:41:0	ENZAZ	01626 333616	TQ12 4SG		BE	2	15	5111	R	express
14	2003-06-16 13:47:4	ENZAZ	01626 333616	TQ12 4SG		NL	4	33.5	7054	R	express
17	2003-06-16 14:25:2		0141 5531251	G1 1UZ	DOH	QA	3	200	32236	A	freight
22	2003-06-17 08:03:1					PE	0	0	0		
106	2003-01-22 17:16:2		01803833163	TQ6 9BH		BR	1	1	0		
86	2003-01-22 14:44:5	COUNT	0121 3567220	B42 2TB		ES	2	30	0		
110	2003-01-22 17:46:4	SOLAV	01752 854400	PL12 6LF		HK	1	0.5	0		
38	2003-02-23 20:24:3					US	1	5	0		
39	2003-02-23 20:46:4						0	0	0		
40	2003-02-23 20:49:3						0	0	0		
41	2003-02-23 20:57:5						0	0	0		
42	2003-02-23 21:01:3						0	0	0		
62	2003-03-01 13:49:3	TERMA	0115 9784652	NG7 7HL		ES	1	10	0		
75	2003-03-02 13:52:2	APECC	01803 833163	TQ6 9BH		GB	1	31	0		
76	2003-03-02 14:17:4	COUNT	0121 3567220	B42 2TB		IT	1	28	12350	R	groupage
77	2003-03-02 17:18:0	SOLAV	01752 854400	PL12 6LF		DE	1	5	0		
72	2003-03-01 20:49:5	MANIT	01895 430053	UB8 2JP		US	1	0	0		
84	2003-06-20 09:24:0			SN10 5SQ	MBA	KE	1	6000	0		
85	2003-06-20 09:48:1	SOLAV	01752 854400	PL12 6LF		IE	1	14	0		
87	2003-03-04 08:48:1					US	1	0	0		
100	2003-03-05 09:21:1					US	1	35	0		
105	2003-03-05 13:29:2			RG6 7BP		CL	1	100	0		
118	2003-03-12 10:30:4	SENSO	01189 845351	RG8 8UD		GR	1	4	0		
125	2003-06-23 13:35:2	INTERM	0115 978 4652	NG7 7HL		GB	5	45	4925	D	TCR
138	2003-03-14 14:53:1					US	1	500	0		

1.1.20. Data model

- Data model
  - Structure of the database
  - Collection of basic operations
  - Collection of behaviours/user defined operations
- Dependent on level of abstraction
- Tier diagram
  - External (user views)
  - Conceptual*
  - Internal

1.1.20. General data model terms

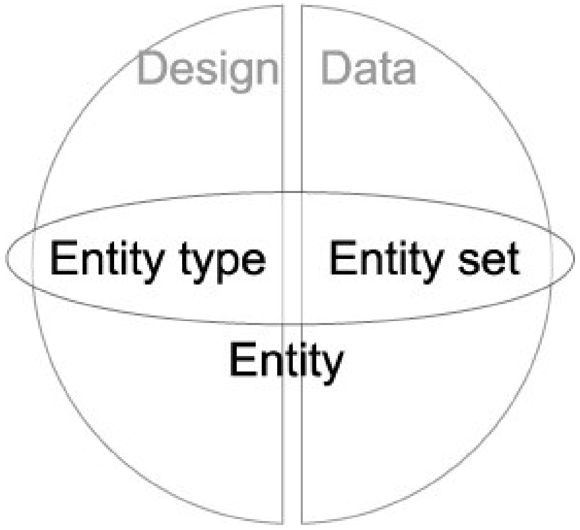
- Entity
- Relationship
- Attributes
- Keys

What follows here is an introduction to the terms which make up the language that we use to describe data models.

1.1.21. Entity

- Selected from real world
- Populate Miniworld/UoD
- Entity is an approximation
- Two elements: Entity types and sets
- Collection of attributes
  - Object similarity, classes as entity types

- Entities inter-relate



1.1.22. Attributes

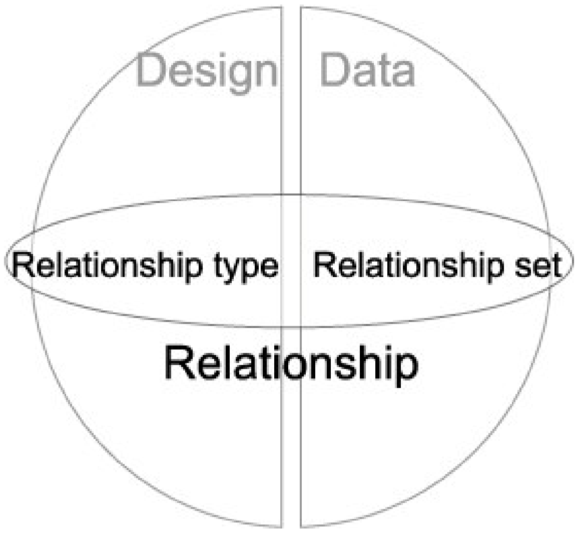
- Data type, domain
- Simple or composite
- Single or multivalued
- Stored or derived
- Null

1.1.23. Keys

- Mechanism for unique identification
- Uniqueness constraint
- Strong and weak entities
- Key attribute
- Composite keys
- Multiple keys

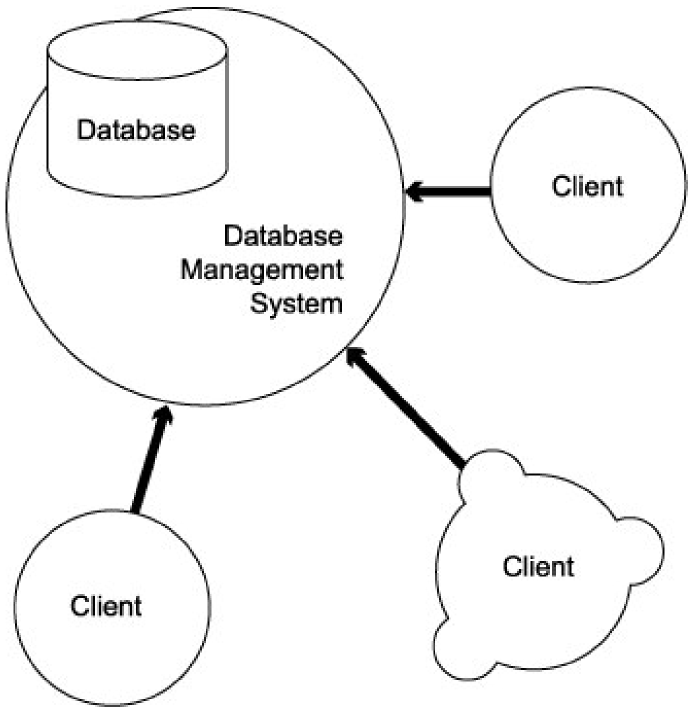
1.1.27. Relationships

- Types and Sets
- Participation by Entities
- Degree - e.g. binary
- Cardinality ratios - e.g. 1:1, 1:M
    - Determine occurrence
- Tuples example
    - Foreign keys



1.1.28. Database architecture

- Client-Server
- Distributed databases
  - Fragmentation by attribute/tuple/relation
- Language and description
  - Storage Definition Language (SDL) DESIGN
  - Data Definition Language (DDL) DESIGN
  - View Definition Language (VDL) DESIGN
  - Data Manipulation Language (DML) DATA
    - High level, can be embedded but precompiled
    - Procedural, record-at-a-time, requires high level support



1.1.29. Structured Query

- Structured Query Language (SQL or SEQUEL)
- Success of relational databases
    - Developed for SystemR at IBM
- ANSI standardised
    - SQL1 or SQL-86, ongoing extension
- SQL2 or SQL-92, current version
- SQL3 (1999), SQL2003
- DDL, DML(low-level) and VDL

1.2. Data models

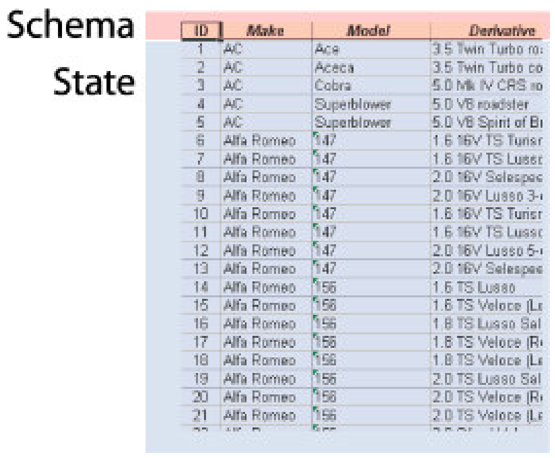
A data model is a model that describes in an abstract way how data are represented in a business organization, an information system or a database management system - Wikipedia.

1.2.01. Introduction

- Relational model
- Abstract operations on relations
    - Set theoretic operations
    - Relational-specific operations
- Basic algebra operations
    - Union, Intersection, Difference
    - Cross product

1.2.02. Relational model

- Table as relation
- Row as tuple
    - real world entity or relationship
    - fact
- Column as attribute
    - Domain

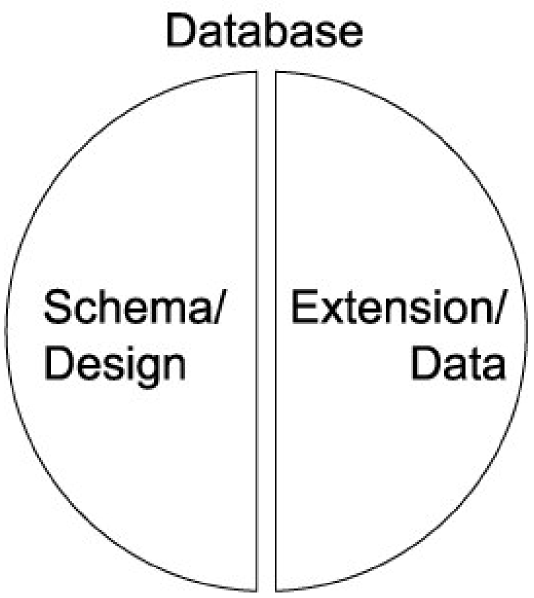


ID	Make	Model	Derivative
1	AC	Ace	3.5 Twin Turbo ro
2	AC	Aceca	3.5 Twin Turbo co
3	AC	Cobra	5.0 Mk IV CRS ro
4	AC	Superblower	5.0 V8 roadster
5	AC	Superblower	5.0 V8 Spirit of Br
6	Alfa Romeo	147	1.6 16V TS Turisr
7	Alfa Romeo	147	1.6 16V TS Lussc
8	Alfa Romeo	147	2.0 16V Selespec
9	Alfa Romeo	147	2.0 16V Lusso 3-
10	Alfa Romeo	147	1.6 16V TS Turisr
11	Alfa Romeo	147	1.6 16V TS Lussc
12	Alfa Romeo	147	2.0 16V Lusso 5-
13	Alfa Romeo	147	2.0 16V Selespec
14	Alfa Romeo	156	1.6 TS Lusso
15	Alfa Romeo	156	1.6 TS Veloce (Lı
16	Alfa Romeo	156	1.8 TS Lusso Sal
17	Alfa Romeo	156	1.8 TS Veloce (R
18	Alfa Romeo	156	1.8 TS Veloce (Lı
19	Alfa Romeo	156	2.0 TS Lusso Sal
20	Alfa Romeo	156	2.0 TS Veloce (R
21	Alfa Romeo	156	2.0 TS Veloce (Lı

The concept of a relation is abstract, therefore we have a number of different ways of visualising it.

1.2.03. Relation

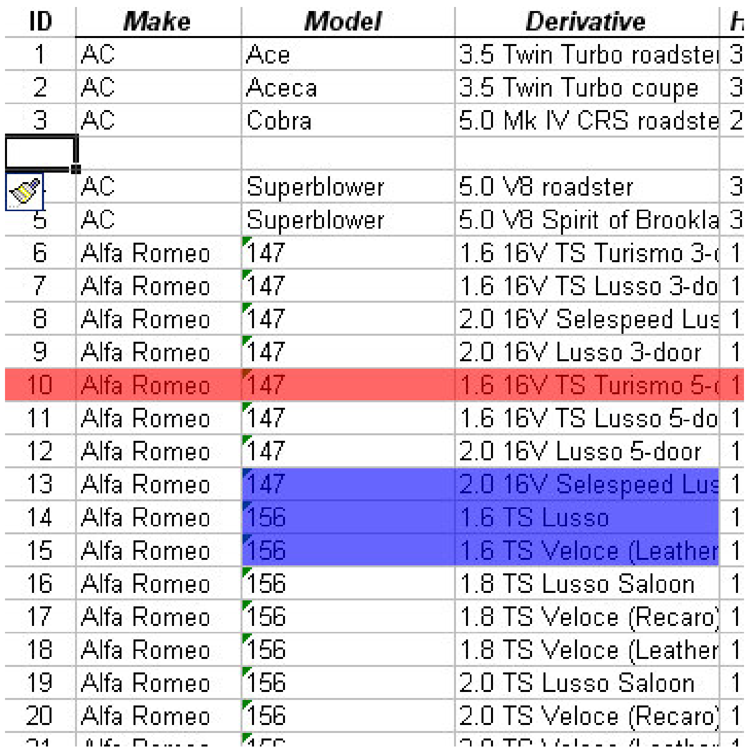
- Relation schema R(A_1, A_2, A_3.. A_n)
  - Design side
  - Assertion/declaration
- Relation state
  - Data side
  - set of n-tuples
    - each one an ordered list of values
      - 1NF: each value is atomic, no composite/multivalue



1.2.04. Abstract operations

- Database lifecycle
  - design, populate, evolve

- Insert
  - tuple (a_1,a_2,a_3…a_n)
- Delete
  - tuple (a_1,a_2,a_3…a_n)
- Update (or modify)
  - tuple (a_1,a_2,a_3…a_n)
  - attribute to change, new value



ID	Make	Model	Derivative	H
1	AC	Ace	3.5 Twin Turbo roadster	3
2	AC	Aceca	3.5 Twin Turbo coupe	3
3	AC	Cobra	5.0 Mk IV CRS roadste	2
	AC	Superblower	5.0 V8 roadster	3
5	AC	Superblower	5.0 V8 Spirit of Brookla	3
6	Alfa Romeo	147	1.6 16V TS Turismo 3-(	1
7	Alfa Romeo	147	1.6 16V TS Lusso 3-do	1
8	Alfa Romeo	147	2.0 16V Selespeed Lus	1
9	Alfa Romeo	147	2.0 16V Lusso 3-door	1
10	Alfa Romeo	147	1.6 16V TS Turismo 5-(	1
11	Alfa Romeo	147	1.6 16V TS Lusso 5-do	1
12	Alfa Romeo	147	2.0 16V Lusso 5-door	1
13	Alfa Romeo	147	2.0 16V Selespeed Lus	1
14	Alfa Romeo	156	1.6 TS Lusso	1
15	Alfa Romeo	156	1.6 TS Veloce (Leather	1
16	Alfa Romeo	156	1.8 TS Lusso Saloon	1
17	Alfa Romeo	156	1.8 TS Veloce (Recaro)	1
18	Alfa Romeo	156	1.8 TS Veloce (Leather	1
19	Alfa Romeo	156	2.0 TS Lusso Saloon	1
20	Alfa Romeo	156	2.0 TS Veloce (Recaro)	1

All the operations described in the next few sections are abstract. We're going to see how valuable they can be in processing real world data later.

1.2.05. Basic algebra

- Two categories
  - Set theoretic operations
    - Union, Intersection etc.
  - Relational specific
    - Select, project and join

At this stage we're talking about set theoretical operators on the Relational model, not SQL instructions which confusingly have identical names and only similar behaviour.

1.2.06. Select operation

- SELECT a subset of tuples from a relation
  - Uses selection condition
    - Evaluate each tuple to true of false

        - False tuples discarded
    - Sigma (s)
    - output = $s_{(cond)}$(input_relation)
    - Relation schema: R(output) = R(input_relation)
    - Commutative

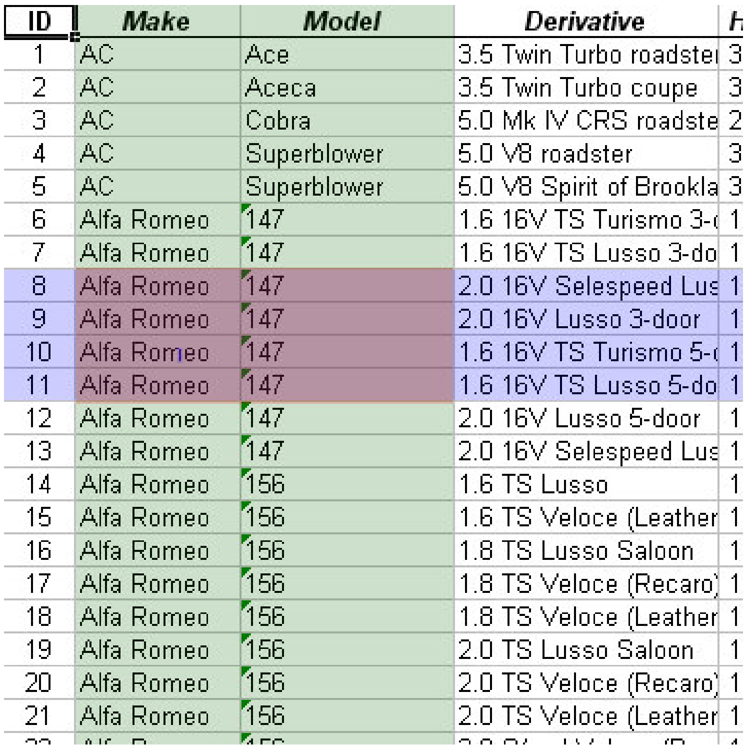
1.2.07. Project operation

- PROJECT a subset of attributes for all tuples from a relation
    - Pi (p)
    - $p_{<attribute\ list>}$(R)
- If sublist is only non-key attributes
    - might get duplicates
- Removes duplicates
- Attribute list:sublist example

The result set of the operation is itself a relational. That output relation will contain the same number of rows as the input, however it may contain a different number of columns; fewer if a subset of attributes is projected; more if derived or aggregated attributes are included.

1.2.07. Sequences of operations

- Select followed by projection
- Area clipping: rows then columns
- $p_{<attr\ list>}$
  ($s_{(select\_cond)}$(R))
- Rename operation (r)
    - Renames attributes list2 from list1
    - $r_{(new\_attr\_names)}$(R)



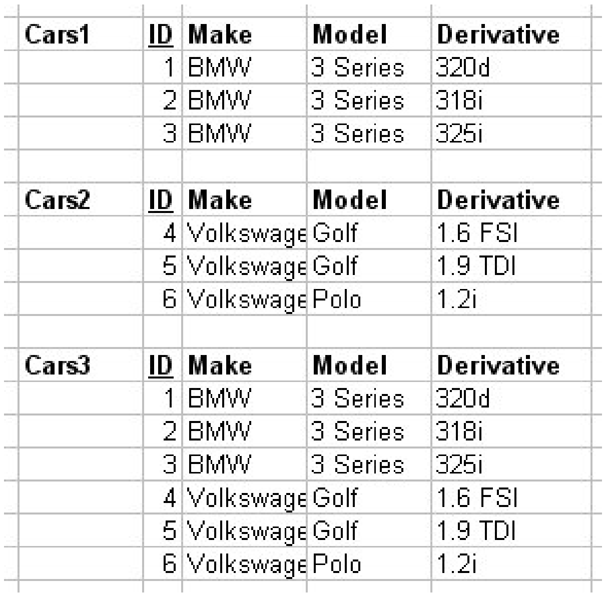
ID	Make	Model	Derivative	H
1	AC	Ace	3.5 Twin Turbo roadster	3
2	AC	Aceca	3.5 Twin Turbo coupe	3
3	AC	Cobra	5.0 Mk IV CRS roadste	2
4	AC	Superblower	5.0 V8 roadster	3
5	AC	Superblower	5.0 V8 Spirit of Brookla	3
6	Alfa Romeo	147	1.6 16V TS Turismo 3-(	1
7	Alfa Romeo	147	1.6 16V TS Lusso 3-do	1
8	Alfa Romeo	147	2.0 16V Selespeed Lus	1
9	Alfa Romeo	147	2.0 16V Lusso 3-door	1
10	Alfa Romeo	147	1.6 16V TS Turismo 5-(	1
11	Alfa Romeo	147	1.6 16V TS Lusso 5-do	1
12	Alfa Romeo	147	2.0 16V Lusso 5-door	1
13	Alfa Romeo	147	2.0 16V Selespeed Lus	1
14	Alfa Romeo	156	1.6 TS Lusso	1
15	Alfa Romeo	156	1.6 TS Veloce (Leather	1
16	Alfa Romeo	156	1.8 TS Lusso Saloon	1
17	Alfa Romeo	156	1.8 TS Veloce (Recaro)	1
18	Alfa Romeo	156	1.8 TS Veloce (Leather	1
19	Alfa Romeo	156	2.0 TS Lusso Saloon	1
20	Alfa Romeo	156	2.0 TS Veloce (Recaro)	1
21	Alfa Romeo	156	2.0 TS Veloce (Leather	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

1.2.08. Rename operation

- Attribute renaming only
  - Cannot alter domain, or add/remove attr
- Rename operation (r)
  - Renames attributes list2 from list1
  - $r_{(new\_attr\_names)}(R)$
- Implicit renaming
  - Order dictated by relational schema

1.2.08. Set Theoretic

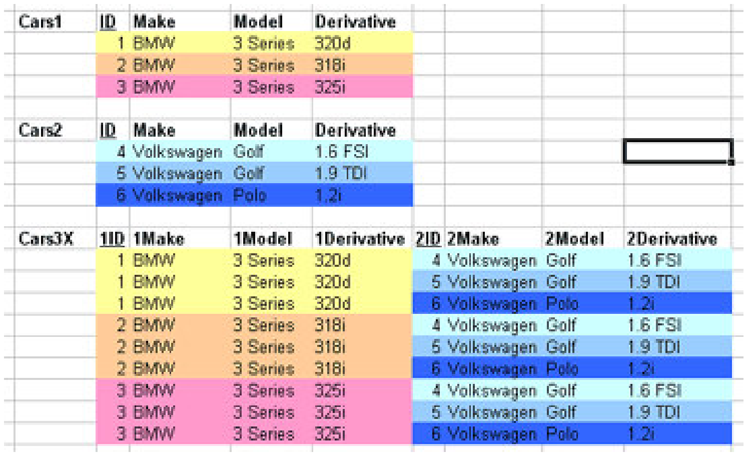
- Binary operation: two relations
  - Sets of tuples
  - Union compatibility (same attributes)
- Union (R u S)
- Intersection (R n S)
  - Commutative (R u (S u T) = (R u S) u T)
- Set difference
  - Non-commutative (R-S != S-R)



Cars1	ID	Make	Model	Derivative
	1	BMW	3 Series	320d
	2	BMW	3 Series	318i
	3	BMW	3 Series	325i
Cars2	ID	Make	Model	Derivative
	4	Volkswage	Golf	1.6 FSI
	5	Volkswage	Golf	1.9 TDI
	6	Volkswage	Polo	1.2i
Cars3	ID	Make	Model	Derivative
	1	BMW	3 Series	320d
	2	BMW	3 Series	318i
	3	BMW	3 Series	325i
	4	Volkswage	Golf	1.6 FSI
	5	Volkswage	Golf	1.9 TDI
	6	Volkswage	Polo	1.2i

1.2.09. Cross product

- Cartesian product of two relations
- R x S
- Also known as
  - Cross product
  - Cross join
    - Cross product diagram
  - Introduction to complexity
    - Computationally explosive



Cars1	ID	Make	Model	Derivative				
	1	BMW	3 Series	320d				
	2	BMW	3 Series	318i				
	3	BMW	3 Series	325i				
Cars2	ID	Make	Model	Derivative				
	4	Volkswagen	Golf	1.6 FSI				
	5	Volkswagen	Golf	1.9 TDI				
	6	Volkswagen	Polo	1.2i				
Cars3X	1ID	1Make	1Model	1Derivative	2ID	2Make	2Model	2Derivative
	1	BMW	3 Series	320d	4	Volkswagen	Golf	1.6 FSI
	1	BMW	3 Series	320d	5	Volkswagen	Golf	1.9 TDI
	1	BMW	3 Series	320d	6	Volkswagen	Polo	1.2i
	2	BMW	3 Series	318i	4	Volkswagen	Golf	1.6 FSI
	2	BMW	3 Series	318i	5	Volkswagen	Golf	1.9 TDI
	2	BMW	3 Series	318i	6	Volkswagen	Polo	1.2i
	3	BMW	3 Series	325i	4	Volkswagen	Golf	1.6 FSI
	3	BMW	3 Series	325i	5	Volkswagen	Golf	1.9 TDI
	3	BMW	3 Series	325i	6	Volkswagen	Polo	1.2i

1.2.10. Relational algebra/model notation

- Relational schema $R(A_1, A_2,\ldots,A_n)$
- Relation state r or r(R)
  - Set of unordered tuples
  - $r = \{t_1, t_2,\ldots,t_n\}$
- Each n-tuple is an ordered list of values

    - $t = <v_1, v_2,\ldots,v_n>$
- i^{th} value in t = v_i called $t[A_i]$
- r(R) subset of ($dom(A_1)$ x $dom(A_2)$... x $dom(A_n)$)

1.2.11. Constraints

- Domain constraint
    - For all v in t of r(R)
      v_i is an element of $dom(A_i)$
- Entity constraint
    - $K = SK_{min}$
    - t[K] != null
- Key constraint
    - Superkey SK as identifying subset of attributes
    - $t_1[SK]$!= $t_2[SK]$

1.2.12. Referential integrity

- Given two relations R_1 and R_2
    - R_1 contains a foreign key (FK) that references
    - A primary key (PK) in R_2
        - R_1 referencing relation, R_2 referenced relation
    - Shared domains: dom(FK) = dom(PK)
    - Foreign exists: t_1 in $r(R_1)$, t_2 in $r(R_2)$
        - $t_1[FK] = t_2[PK]$ || NULL

1.3. Joins

In this lecture we look at...

1.3.01. Introduction

- Recap: pulling data out of individual relations
    - By row, by column
    - Select and project
- Access across multiple relations
- Miniworld approximation
    - Fragmenting entities by cardinality
    - Tuples as entity fragments
    - Relationships within relations
- Joins
- Join types (condition and unmatched)

1.3.02. Access across relations

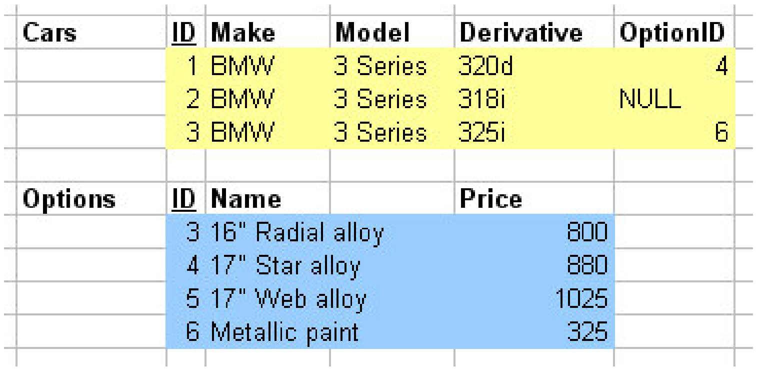
- Relational model allows multiple relations to exist within one database schema
- Relations can be accessed individually or together (joins).
- Referential integrity
  - Relations relating
- Pulling data out of single relations
  - Select and project
- Pulling related data out of
  - Multiple relations using Join

1.3.03. Miniworld approximation

- Universe of Discourse, or Miniworld
- Miniworld is an incomplete model of the real world
- The relational data model as a model for the miniworld
- Approximation
  - Separate and distinct entities
  - Single complex entities
  - Separate related entities
  - Cardinality of relationships
- Each relation made up of attributes
- Values can be used as references

1.3.04. Pointing mechanism

- Relation has a Primary key
- Tuple contains Primary key value
- Foreign keys
  - Tuples can contain a reference to another relation's Primary key
- Just numbers



Cars	ID	Make	Model	Derivative	OptionID
	1	BMW	3 Series	320d	4
	2	BMW	3 Series	318i	NULL
	3	BMW	3 Series	325i	6

Options	ID	Name	Price
	3	16" Radial alloy	800
	4	17" Star alloy	880
	5	17" Web alloy	1025
	6	Metallic paint	325

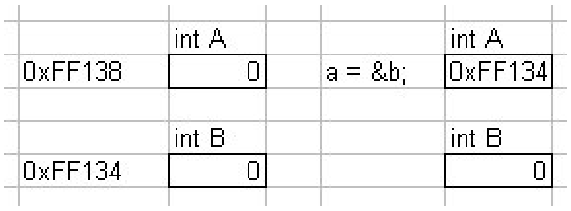
One number identifies a single tuple in one relation (local), one number identifies a single tuple in another relation (foreign).

1.3.04b. Pointing mechanism example in C

- C programming language
- Memory addresses, or pointers

```
int a=0;
int b=0;
a = &b;
```

- a points to b



	int A		int A
0xFF138	0	a = &b;	0xFF134
	int B		int B
0xFF134	0		0

In databases, typically done with unique identifiers (IDs) rather than memory addresses.

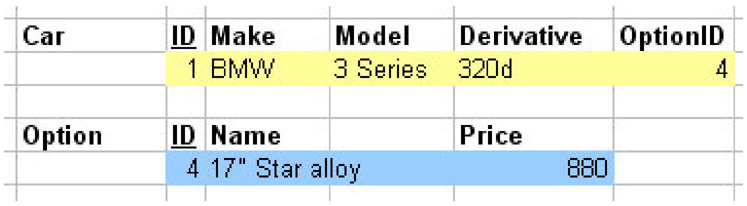
1.3.04c. Pointing mechanism with structures

- Foreign key importing

```
typedef struct car
{
  int ID;
  char[] make;
  char[] model;
  char[] derivative;
  int optionID;
} car;

typedef struct option
{
  int ID;
  char[] name;
  int price;
} option;

car c;
option o;
//...data structure populating
c.optionID = o.ID;
```



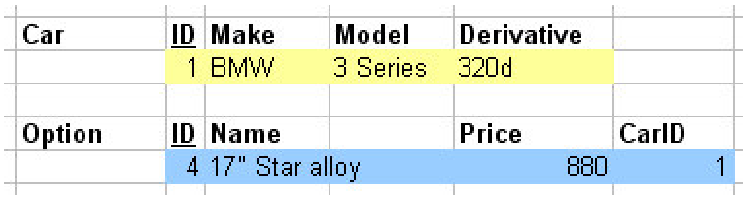
Car	ID	Make	Model	Derivative	OptionID
	1	BMW	3 Series	320d	4
Option	**ID**	**Name**		**Price**	
	4	17" Star alloy		880	

1.3.05. Relational cardinality

- 1:0 relationships
  - Single entity
  - Uniquely indentifiable
  - Candidate keys
  - Primary Key
- 1:1 relationships
  - Two entities, A and B
  - 1 A relates to 1 B and vice versa
- 1:N relationships
- M:N relationships

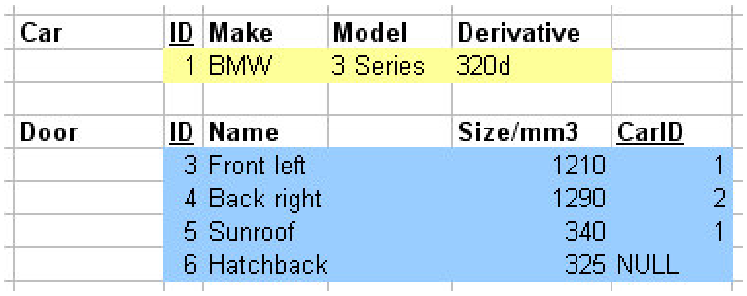
1.3.06. Relationships in the relational model

- Two relations, A and B
- A side, B side, 1 side, N side
- 1:1 relationships
  - Key can go on either side



Car	ID	Make	Model	Derivative	
	1	BMW	3 Series	320d	
Option	ID	Name		Price	CarID
	4	17" Star alloy		880	1

- 1:N relationships
- Key cannot go on 1 side
- Has to go on N side



Car	ID	Make	Model	Derivative	
	1	BMW	3 Series	320d	
Door	ID	Name		Size/mm3	CarID
	3	Front left		1210	1
	4	Back right		1290	2
	5	Sunroof		340	1
	6	Hatchback		325	NULL

- M:N relationships
  - Nowhere obvious for the key to go
  - Create new pairing relation

1.3.07. Joins

- Phase change, different point in lifecycle
- Join operation

- Combines related tuples, conditionally
- From two relations
- Into single tuples

- Allows processing of relationships
- Among multiple relations

1.3.08. Joins, canonical algebraic form

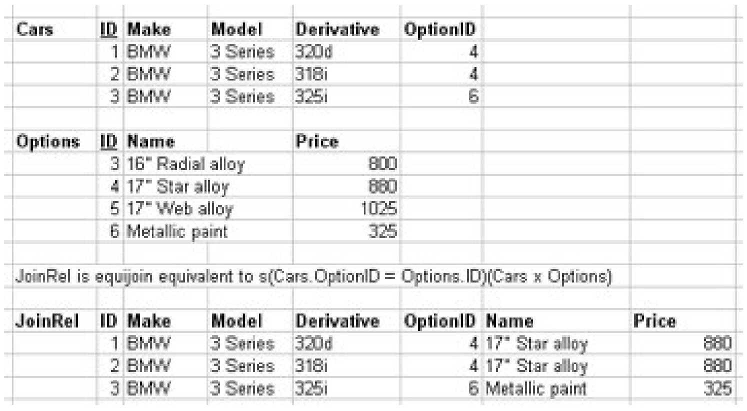
- Conditional (on join condition)
    - Only combines tuples where true
- Cartesian product (conditionless)
    - example of conditionless join
    - all tuples combined
    - $R \bowtie_{true} S$
- $\bowtie$, Binary operator
  e.g. $R \bowtie_{<join\_condition>} S$

1.3.09. Join equivalence

- Equivalent to sequence
    - Cartesian product (X)
    - followed by Selection (s)
- ACTUAL_DEPENDENTS =
  $s_{SSN=ESSN}$(EMPNAMES X DEPENDENT)
- or
- ACTUAL_DEPENDENTS =
  EMPNAMES $\bowtie_{SSN=ESSN}$(DEPENDENT)

1.3.10. Join types (condition)

- Theta: A_i q B_j
  (A from R, B from S)
    - q is comparison operator
      =,<,>,!=,>=
    - A_i and B_j share the same domain
- Equi: $A_i = B_j$
    - Theta join where q is =
- Natural: A_i and B_j are the same attribute
    - in two separate relations (name and domain)
    - * denotes natural join
    - e.g. EMPNAMES * DEPENDENTS



Cars	ID	Make	Model	Derivative	OptionID
	1	BMW	3 Series	320d	4
	2	BMW	3 Series	318i	4
	3	BMW	3 Series	325i	6

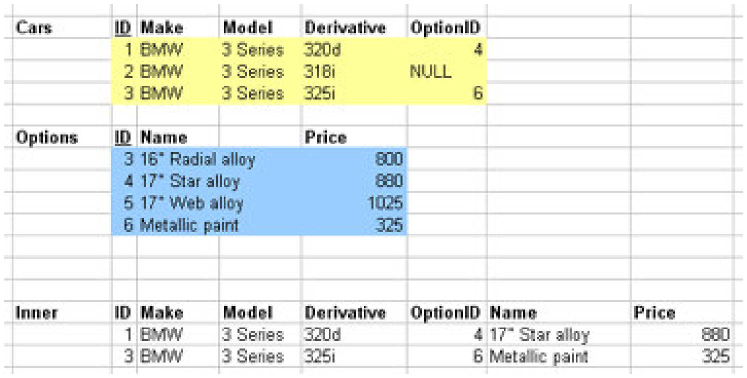
Options	ID	Name	Price
	3	16" Radial alloy	800
	4	17" Star alloy	880
	5	17" Web alloy	1025
	6	Metallic paint	325

JoinRel is equijoin equivalent to s(Cars.OptionID = Options.ID)(Cars x Options)

JoinRel	ID	Make	Model	Derivative	OptionID	Name	Price
	1	BMW	3 Series	320d	4	17" Star alloy	880
	2	BMW	3 Series	318i	4	17" Star alloy	880
	3	BMW	3 Series	325i	6	Metallic paint	325

1.3.11. Join types (inner and outer)

- Inner joins
  - not the only joins
  - eliminate tuples without a matching counterpart
  - i.e. tuples with a null value for the join attribute are discarded



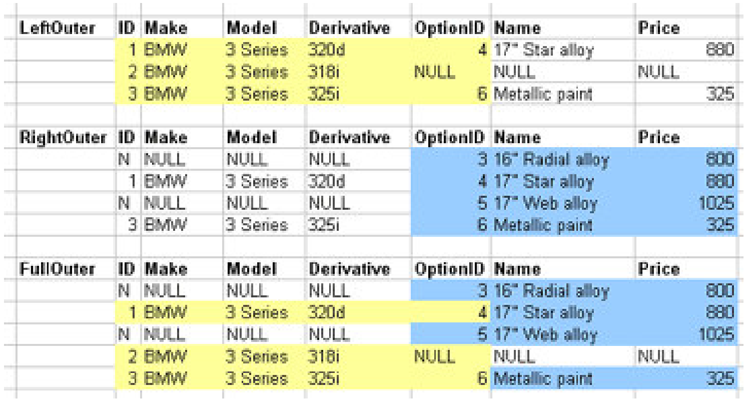
Cars	ID	Make	Model	Derivative	OptionID
	1	BMW	3 Series	320d	4
	2	BMW	3 Series	318i	NULL
	3	BMW	3 Series	325i	6

Options	ID	Name	Price
	3	16" Radial alloy	800
	4	17" Star alloy	880
	5	17" Web alloy	1025
	6	Metallic paint	325

Inner	ID	Make	Model	Derivative	OptionID	Name	Price
	1	BMW	3 Series	320d	4	17" Star alloy	880
	3	BMW	3 Series	325i	6	Metallic paint	325

1.3.12. Outer joins

- Outer joins control what's discarded
  - Keep unmatched tuples in either
    - Left, right, or both relations
    - Left, right of full outer join correspondingly



LeftOuter	ID	Make	Model	Derivative	OptionID	Name	Price
	1	BMW	3 Series	320d	4	17" Star alloy	880
	2	BMW	3 Series	318i	NULL	NULL	NULL
	3	BMW	3 Series	325i	6	Metallic paint	325

RightOuter	ID	Make	Model	Derivative	OptionID	Name	Price
	N	NULL	NULL	NULL	3	16" Radial alloy	800
	1	BMW	3 Series	320d	4	17" Star alloy	880
	N	NULL	NULL	NULL	5	17" Web alloy	1025
	3	BMW	3 Series	325i	6	Metallic paint	325

FullOuter	ID	Make	Model	Derivative	OptionID	Name	Price
	N	NULL	NULL	NULL	3	16" Radial alloy	800
	1	BMW	3 Series	320d	4	17" Star alloy	880
	N	NULL	NULL	NULL	5	17" Web alloy	1025
	2	BMW	3 Series	318i	NULL	NULL	NULL
	3	BMW	3 Series	325i	6	Metallic paint	325

1.4. ER diagrams

In this lecture we look at...

1.4.01. ER Diagrams and Relational mapping

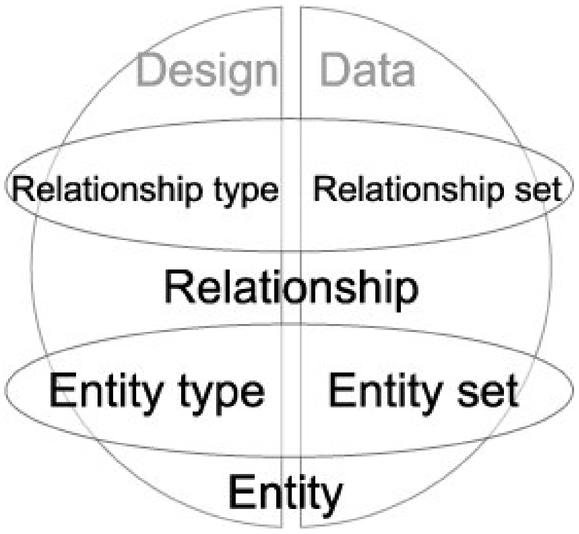
- Design communication techniques
  - ER diagrams
  - ER to relational mapping
- Entities to Objects
- Type Inheritance
  - EER diagrams
  - UML
- Web DB Integration

1.4.03. Design in the modern context

- Team based development
- Documentation
  - Value of design over description
- DB sketching (left hand side)
  - Concept more important than perfection
  - Design iteration
- Mini-world as approximation
  - Categorisation to create entities
  - Verb'ing to create actions/relationships

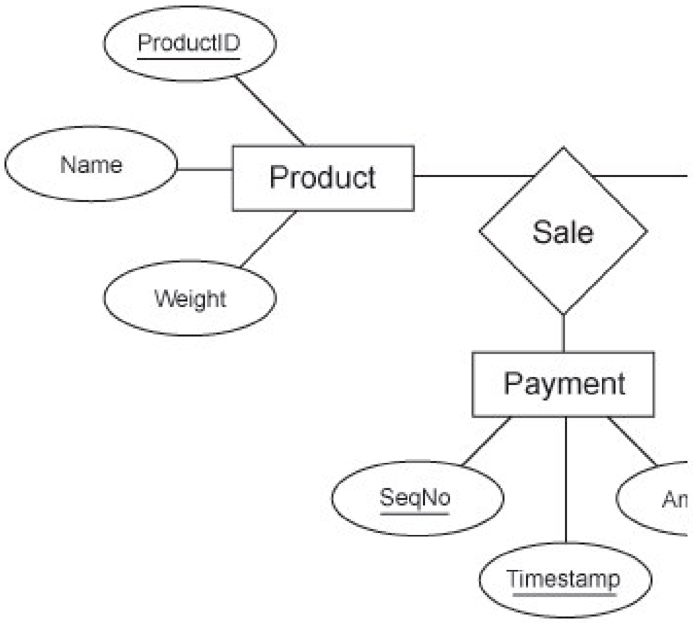
1.4.04. Database Left:right divide

- Design
  - Catalog, Meta-data, Intension, or Database schema
  - Entity type
  - Relationship type
- State
  - Set of occurences/instances, Extension, snapshot
  - Entity set
  - Relationship set



1.4.05. Basic ER diagram

- Typically part of a system
- (Strong) Entities
  - Product
  - Customer
  - Payment
- Relationships
  - Sale



1.4.06. Mapping ER to Relation DB tables

- Intuitive mapping
  - Entities as tables

  - Attributes as columns
- Relationships are more difficult
- Key sharing mechanism
  - Foreign key references primary key
- Where to put the foreign key forms the intuitive guide to the rest of the mapping

1.4.07. ER to Relational mapping

- Step-by-step approach

1. Strong entities
   - Create relation including (simplified) attributes
2. Weak entities
   - Create relation inc. attr, foreign/pri key of owner
3. Binary relationship S:T, 1:1
   - Choose relation, say S (with total participation) and inc. foreign/pri key of T
   - inc. relationship attributes

1.4.08. Cardinality

- Specifies number of relationship instances a single entity can participate in
- S:T (1:1)
- An entity from table S can is related to one, and only one entity from table T
- 1:1, 1:N, N:M
  - DEPARTMENT : EMPLOYEE
  - EMPLOYEE : EMPLOYEE
  - PROJECT : EMPLOYEE

1.4.09. ER to Relational mapping

5. Binary relationship 1:N
   - Choose relation T (N-side) inc. foreign/pri key of S
6. Binary relationship M:N
   - Create relation, inc. foreign/pri keys of S&T
7. Multivalued
   - For each mv_attr, create new relation, inc. foreign/pri key of parent
8. n-ary relationship
   - Create new relation, inc. all foreign/pri keys of participating entities

1.4.10. Participation

- Participation constraints

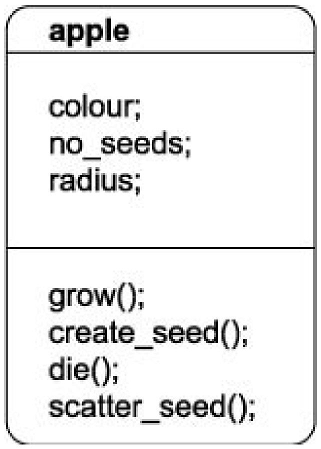
- Existence of an entity dependant upon
  - being related to another entity
  - via relationship type (left hand/design)
- Total (")/Existence dependency (double line)
  - Every student must be in a faculty
  - For every entity in the total set of students
- Partial ($) (single line)
  - Some students are student_representatives
  - There exists some entity(s) within the set of all…

1.5. UML

In this lecture we look at...

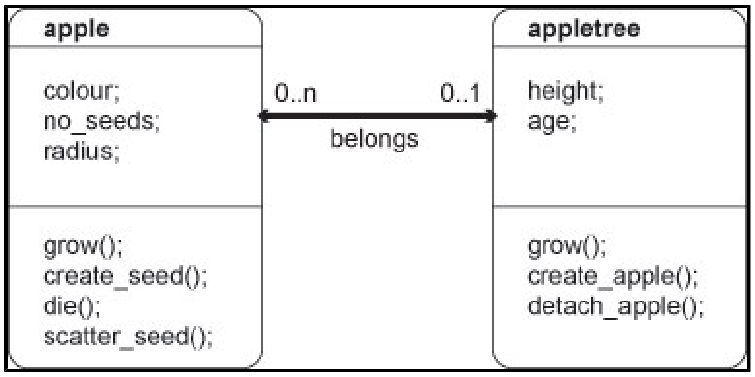
1.5.01. UML diagrams

- Not just one type of diagram
- ER Entities -> UML objects
- Adds scope to include methods



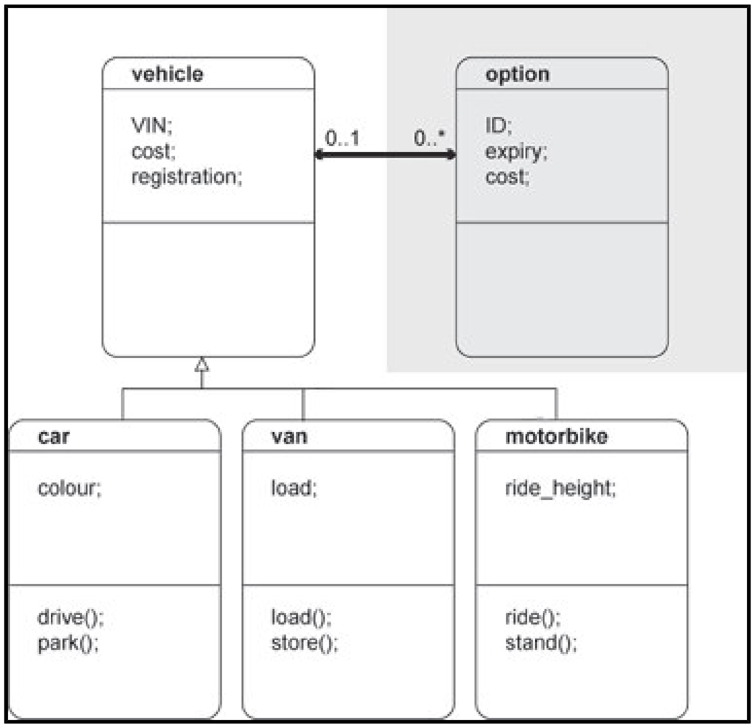
1.5.03. Simple UML relationship diagram

- Classes
  - Attributes
  - Methods
- Relationships
  - Participation
  - Cardinality
  - Roles



1.5.04. UML inheritance

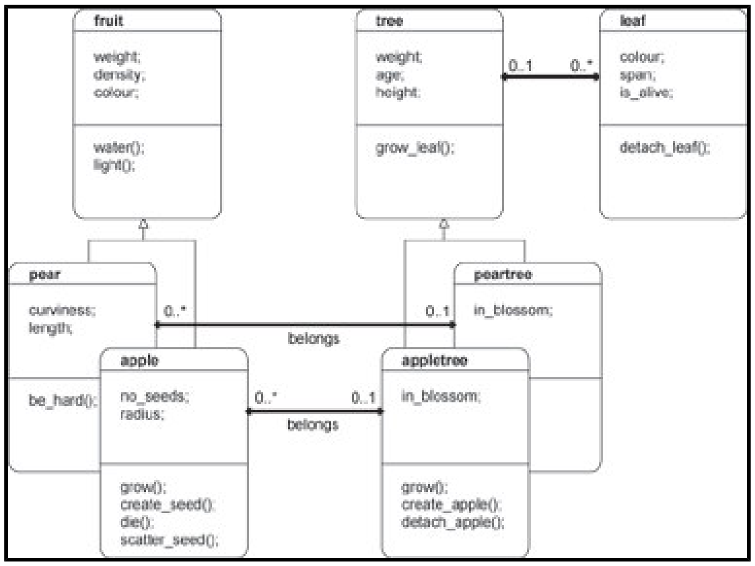
- Notion of inheritance
    - Java parallel
    - 'is a' relationships
    - Database Student
        - is a Computer Science Student
            - is an Engineering Faculty Student
            -
                - is a Student
                    - is a Person
    - Inheritance hierarchies
- UML diagrams can be used to show inheritance



1.5.05. UML diagram

- Classes

- Relationships
- Inheritance


fruit
weight;
density;
colour;
water();
light();
tree
weight;
age;
height;
grow_leaf();
leaf
colour;
span;
is_alive;
detach_leaf();
0..1
0..*
pear
curviness;
length;
be_hard();
peartree
in_blossom;
0..*
0..1
belongs
apple
no_seeds;
radius;
grow();
create_seed();
die();
scatter_seed();
appletree
in_blossom;
grow();
create_apple();
detach_apple();
0..*
0..1
belongs