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

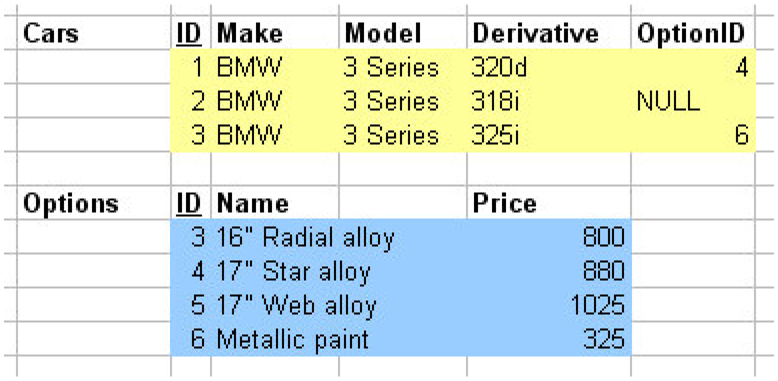
- Relational model allows multiple relations to exist within one database schema
- Relations can be accessed individually or together (joins).
- Referential integrity
    - Relations relating
- Pulling data out of single relations
    - Select and project
- Pulling related data out of
    - Multiple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

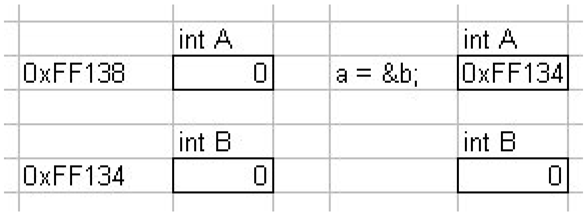
One number identifies a single tuple in one relation (local), one number identifies a single tuple in another relation (foreign).

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.

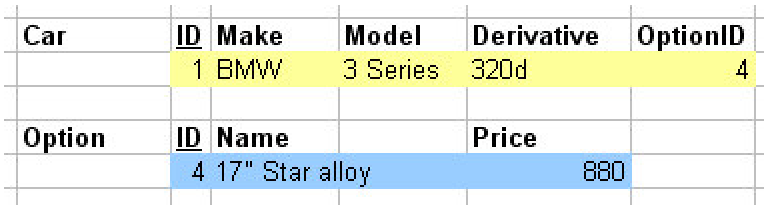
1.3.04c. Pointing mechanism with structures

- 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;
```



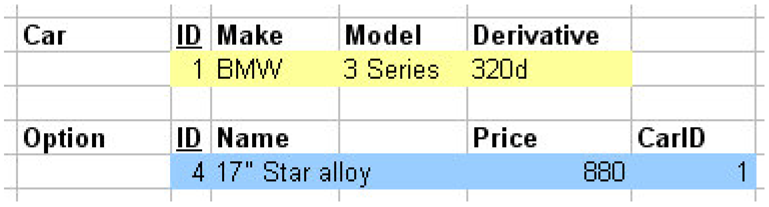
Car	ID	Make	Model	Derivative	OptionID
	1	BMW	3 Series	320d	4
Option	ID	Name		Price	
	4	17" Star alloy		880	

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

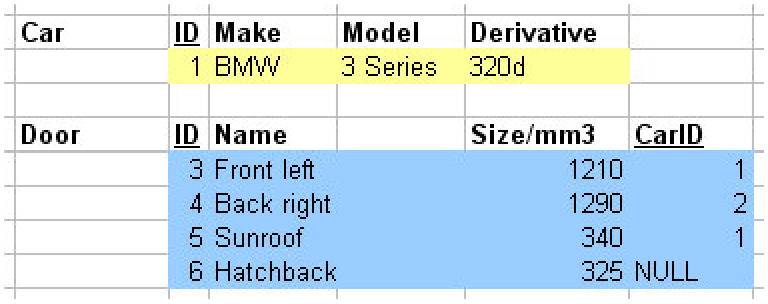
- Two relations, A and B
- A side, B side, 1 side, N side
- 1:1 relationships
    - Key can go on either side



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. $R \bowtie_{<join\_condition>} S$

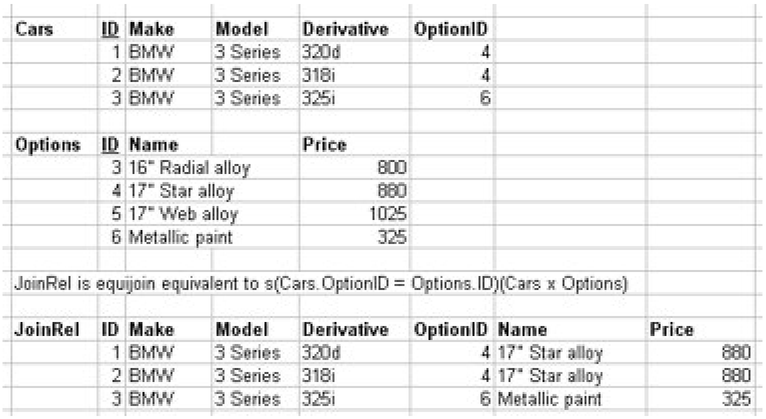
1.3.09. Join equivalence

- 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

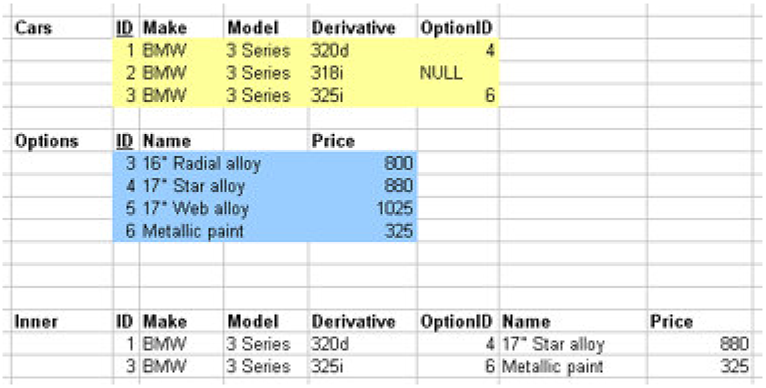
Options	ID	Name	Price
	3	16" Radial alloy	800
	4	17" Star alloy	880
	5	17" Web alloy	1025
	6	Metallic paint	325

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



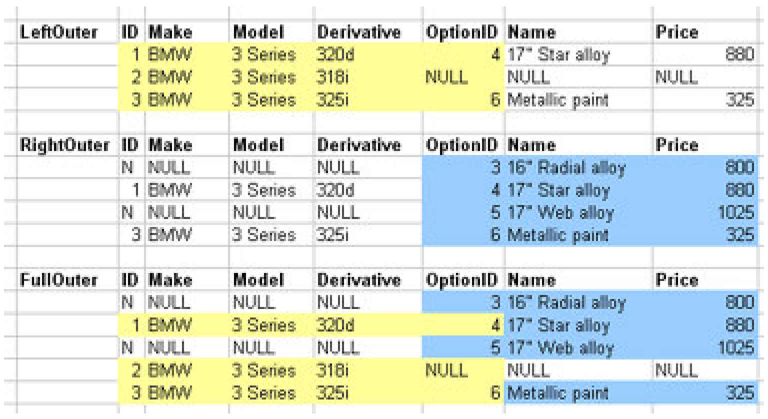
Cars	ID	Make	Model	Derivative	OptionID
	1	BMW	3 Series	320d	4
	2	BMW	3 Series	318i	NULL
	3	BMW	3 Series	325i	6

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