NOSQL DATA MODELS

2

2

NoSQL taxonomy

- Key-Value stores
- Column stores
- Document stores

3

3

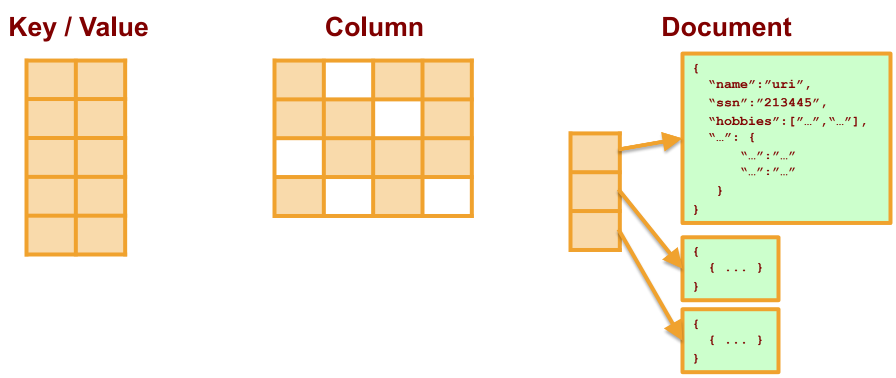
NoSQL taxonomy

- Key-Value stores — Amazon Dynamo
- Column stores — Google Bigtable, Cassandra
- Document stores — CouchDB, MongoDB, SimpleDB

4

4

A Universe of Data Models



5

5

Document

- Think JSON (or BSON, or XML)

6

6

Key/Value

- Have the key? Get the value
  - Map/Reduce (sometimes)
  - Good for
    - cache aside (e.g. Hibernate 2nd level cache)
    - Simple, id based interactions (e.g. user profiles)
- In most cases, values are opaque

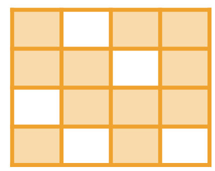
K1	V1
K2	V2
K3	V3
K4	V1

7

7

Column Based

- Mostly derived from Google's BigTable papers
- One giant table of rows and columns
  - Column == pair (name and a value, sometimes timestamp)
  - Table is sparse:
    `(#rows) × (#columns) ≥ (#values)`

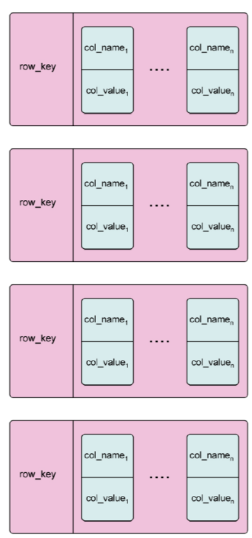


8

8

Column Based

- Query on row key
  - Or column value (aka secondary index)
- Good for a constantly changing, (albeit flat) domain model



9

9

OBJECT ORIENTED LANGUAGES AND RELATIONAL DATABASES

10

10

Objects

- Object

```
class Product {
   string Title;
   string Author;
   int Year;
   int Pages;
   IEnumerable<string> Keywords;
   IEnumerable<string> Ratings;
}
```

11

11

Object Initializer

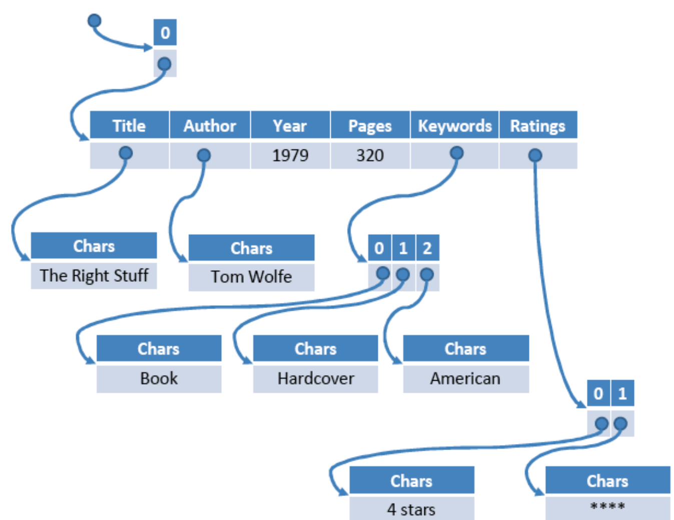
- Object Initializer

```
var _1579124585 =
   new Product {
      Title = “The Right Stuff”,
      Author = “Tom Wolfe”,
      Year = 1979,
      Pages = 320,
      Keywords = new[]{ “Book”,
               “Hardcover”, “American” },
      Ratings = new[]{ “****”, “4 stars” },
   }
var Products = new[]{ _1579124585 };
```

12

12

Object Graph



13

13

LINQ Query

- Products with Four-Star Ratings

```
var q =
   from product in Products
   where
       product.Ratings.Any
                   (rating ⇒ rating == “****”)
   select new{ product.Title,
               product.Keywords };
```

14

14

LINQ Query

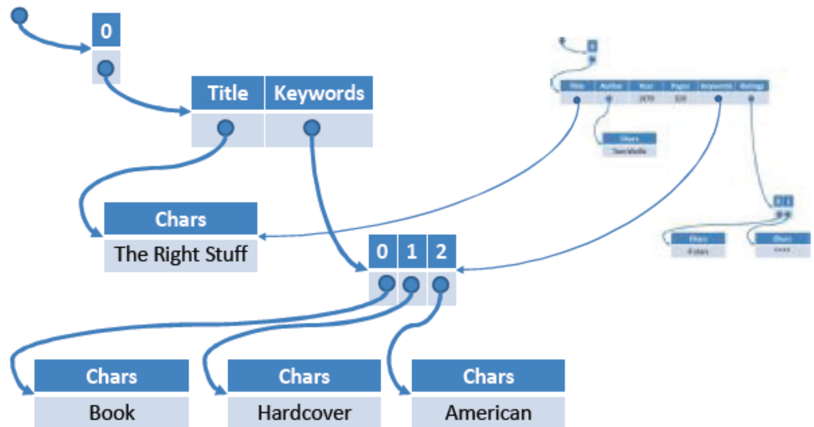
- Products with Four-Star Ratings

```
var q =
   Products
   .Where(product ⇒
       product.Ratings.Any
                   (rating ⇒ rating == “****”))
   .Select(product ⇒
       new{ product.Title,
            product.Keywords });
```

15

15

Object Graph



16

16

Tables

- The relational model is a particularly suitable structure for the truly casual user (i.e., a nontechnical person who merely wishes to interrogate the database, for example a housewife who wants to make enquiries about this week's best buys at the supermarket). In the not too distant future the majority of computer users will probably be at this level.
- *C.J. Date & E.F. Codd*

17

17

```
table Products
{
   int ID;
   string Title;
   string Author;
   int Year;
   int Pages;
}

table Keywords
{
   int ID;
   string Keyword;
   int ProductID;
}

table Ratings
{
   int ID;
   string Rating;
   int ProductID;
}
```

```
Products.Insert
          ( 1579124585
          , “Tom Wolfe”
          , 1979
          , 304
          );

Keywords.Insert
          ( 4711
          , “Book”
          , 1579124585
          );
Keywords.Insert
          ( 1843
          , “Hardcover”
          , 1579124585
          );
Keywords.Insert
          ( 2012
          , “American”
          , 1579124585
          );
```

```
Ratings.Insert
          ( 787
          , “*****”
          , 1579124585
          );
Ratings.Insert
          ( 747
          , “4 stars”
          , 1579124585
          );
```

In SQL rows
are not expressible

18

18

Ratings

ID	Rating	ProductID
787	****	1579124585
747	4 stars	1579124585

Products

ID	Title	Author	Year	Pages
1579124585	The Right Stuff	Tom Wolfe	1979	304

Keywords

ID	Keyword	ProductID
4711	Book	1579124585
1843	Hardcover	1579124585
2012	American	1579124585

19

19

Referential Integrity

ID	Rating	ProductID
787	****	1579124585
747	4 stars	1579124585

Foreign key must have corresponding primary key

ID	Title	Author	Year	Pages
1579124585	The Right Stuff	Tom Wolfe	1979	304

Primary key must be unique

20

20

LINQ to SQL

- Naïve Query:

```
var q=
    from product in Products
    from rating in Ratings
    from keyword in Keywords
    where product.ID == rating.ProductId
       && product.ID == keyword.ProductID
       && rating == “****”
    select new{ product.Title,
                keyword.Keyword };
```

Title	Keyword
The Right Stuff	Book
The Right Stuff	Hardcover
The Right Stuff	American

21

21

LINQ to SC

Title	Keyword
The Right Stuff	Book
The Right Stuff	Hardcover
The Right Stuff	American

- Efficient Query:

```
var q =
 from product in Products
      join rating in Ratings
      on product.ID equals rating.ProductId
 where rating == "****"
 select product into FourStarProducts
 from fourstarproduct in FourStarProducts
      join keyword in Keywords
      on product.ID equals keyword.ProductID
 select new { product.Title, keyword.Keyword };
```

22

22

Compositionality

- In mathematics, semantics, and philosophy of language, the **Principle of Compositionality** is the principle that the meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them.
- *Gottlob Frege 1848-1925*

23

23

Objects vs Tables

- Objects
  - Fully compositional

```
value ::= scalar
value ::= new { …, name = value, … }
```

- Tables
  - Non compositional

```
value ::= new { …, name = scalar, … }
```

24

24

Tables

- Non compositional
  - Query results denormalized
  - Query can only return single table
  - No recursion (but have CTEs)

- NULL semantics a mess

```
Sum(1,NULL) = 1
1+NULL = NULL
```

25

25

Impedance Mismatch

- The problem with having two languages is "impedance mismatch " One mismatch is conceptual – the data language and the programming languages might support widely different programming paradigms. [...] The other mismatch is structural – the languages don't support the same data types, [...]
- *George Copeland & David Maier* 1984

26

26

LINQ to SQL MSDN Documentation

- LINQ to SQL provides a runtime infrastructure for managing relational data as objects without losing the ability to query. Your application is free to manipulate the objects while LINQ to SQL stays in the background tracking your changes automatically.

27

27

Entity Framework
MSDN Documentation

- When one takes a look at the amount of code that the average application developer must write to address the impedance mismatch across various data representations (for example objects and relational stores) it is clear that there is an opportunity for improvement.

28

28

ORM (Entity Framework)

```
[Table(name="Products")]
class Product {
  [Column(PrimaryKey=true)]int ID;
  [Column]string Title;
  [Column]string Author;
  [Column]int Year;
  [Column]int Pages;
  private EntitySet<Rating>
    _Ratings;
  [Association(Storage=" _ Ratings",
  ThisKey="ID",OtherKey="ProductID",
  DeleteRule="ONDELETECASCADE")]
  ICollection<Rating> Ratings{ ... }
  private EntitySet<Keyword>
    _Keywords;
  [Association(Storage="_ Keywords",
  ThisKey="ID",OtherKey="ProductID",
  DeleteRule="ONDELETECASCADE")]
  ICollection<Keyword>
    Keywords{ ... }
}
```

```
[Table(name="Keywords")]
class Keyword {
  [Column(PrimaryKey=true)]int ID;
  [Column]string Keyword;
  [Column(IsForeignKey=true)]
  int ProductID;
}

[Table(name="Ratings")]
class Rating {
  [Column(PrimaryKey=true)]int ID;
  [Column]string Rating;
  [Column(IsForeignKey=true)]
  int ProductID;
}
```

29

29

Query

- O/R mapper constructs nested result structures:

```
var q = from product in Products
        where product.Ratings.Any
            (rating ⇒ rating.Rating == "****")
        select new { product.Title,
                     product.Keywords };
```

ID	Title
1579124585	The Right Stuff

ID	Keyword	ProductID
4711	Book	1579124585
1843	Hardcover	1579124585
2012	American	1579124585

30

30

Indexes Recover Nesting

ID	Title	Author	Year	Pages
1579124585	The Right Stuff	Tom Wolfe	1979	304

ID	from rating in Ratings where ID = rating.ID select rating.ID		from keyword in Keywords where ID = keyword.ID select keyword.ID		
1579124585	787	747	4711	1843	2012

ID	Keyword	ProductID
4711	Book	1579124585
1843	Hardcover	1579124585
2012	American	1579124585

ID	Rating	ProductID
787	****	1579124585
747	4 stars	1579124585

31

31

Indexes Recover Nesting

ID	Title	Author	Year	Pages
1579124585	The Right Stuff	Tom Wolfe	1979	304

Keywords		
4711	1843	2012

Ratings	
787	747

ID	Keyword	ProductID
4711	Book	1579124585
1843	Hardcover	1579124585
2012	American	1579124585

ID	Rating	ProductID
787	****	1579124585
747	4 stars	1579124585

32

32

Ad-Hoc Queries

- Example:
- Ad-hoc queries

```
from p1 in Products
from p2 in Products
where p1.Title.Length == p2.Author.Length
select new{ p1, p2 };
```

- Issues:
  - Complexity ($O(N^2)$)
  - No referential integrity (closed world assumption)

33

33

Summary

- Designer
  – Remove original hierarchical structure into normalized data
- App Developer
  – Recover original hierarchical structure from normalized data
- Database Implementer
  – Recover original hierarchical structure from normalized data

34

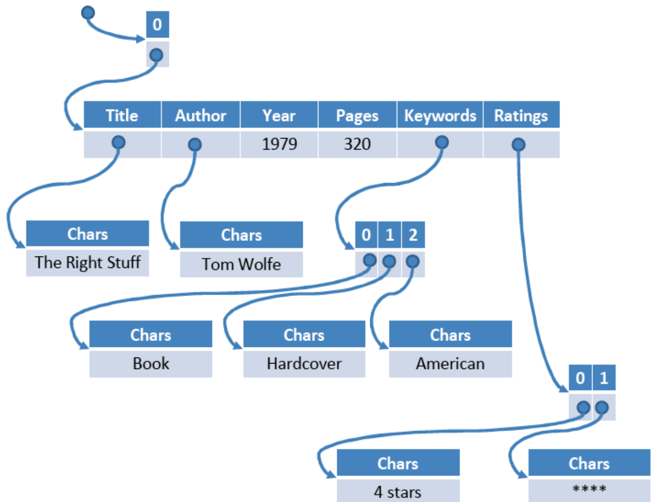
34

FROM ALGEBRAS TO COALGEBRAS

35

35

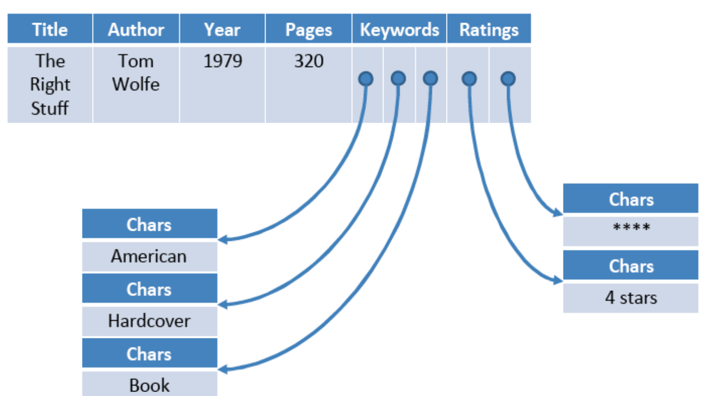
Object Graph



36

36

Ignore Identity of Collections



37

37

Tables

ID	Rating	ProductID
787	****	1579124585
747	4 stars	1579124585

ID	Title	Author	Year	Pages
1579124585	The Right Stuff	Tom Wolfe	1979	304

ID	Keyword	ProductID
4711	Book	1579124585
1843	Hardcover	1579124585
2012	American	1579124585

38

38

Tables

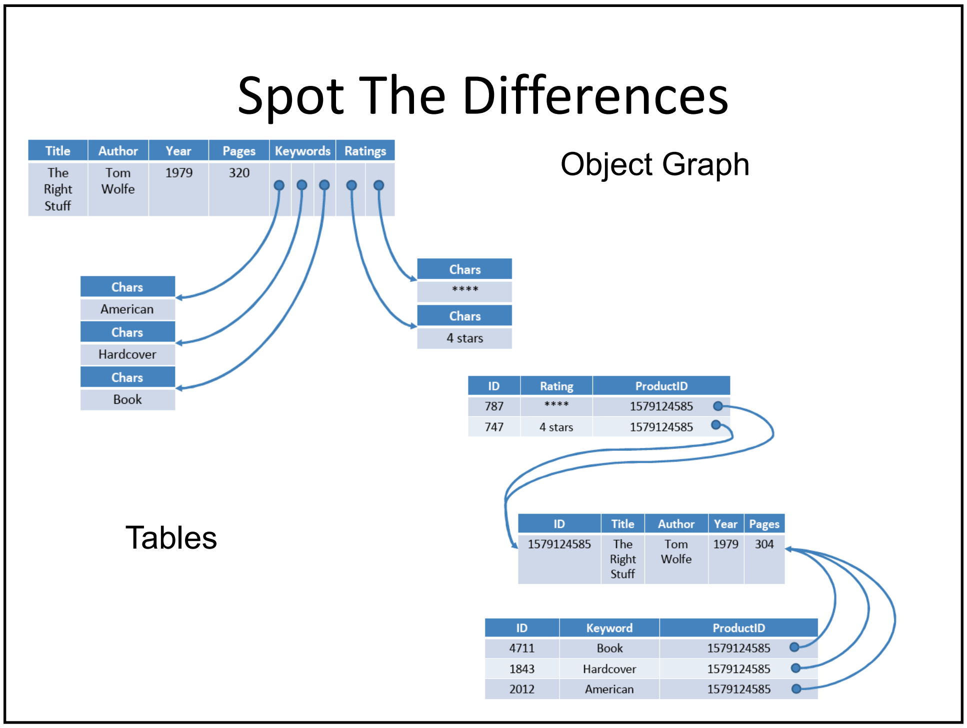
ID	Rating	ProductID
787	****	1579124585
747	4 stars	1579124585

ID	Title	Author	Year	Pages
1579124585	The Right Stuff	Tom Wolfe	1979	304

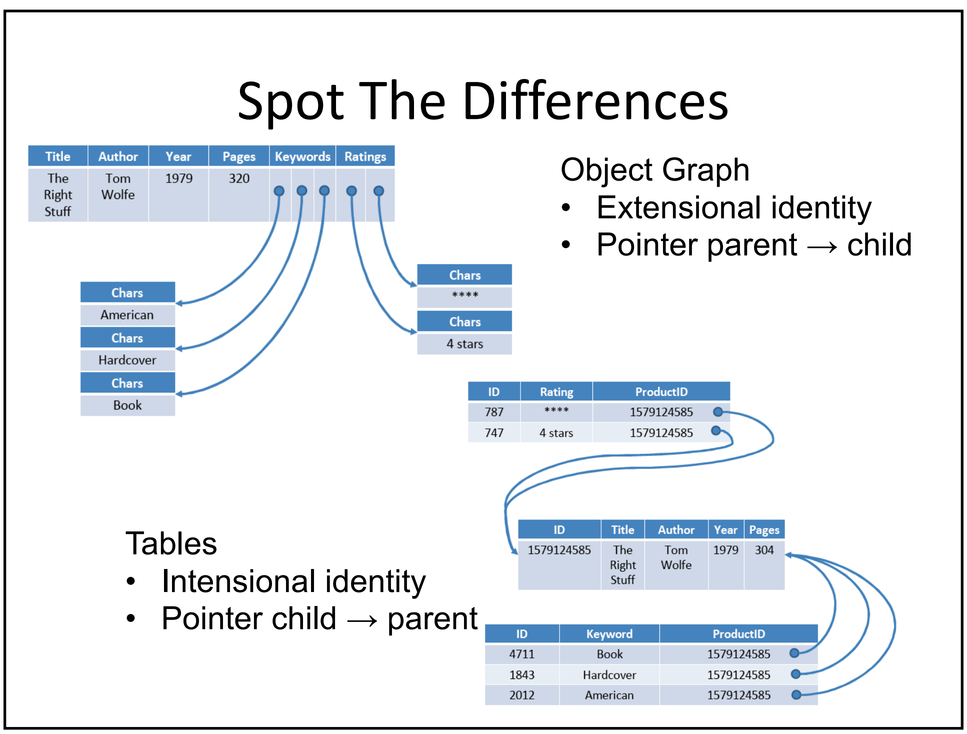
ID	Keyword	ProductID
4711	Book	1579124585
1843	Hardcover	1579124585
2012	American	1579124585

39

39


Spot The Differences
Object Graph
Title Author Year Pages Keywords Ratings
The Right Stuff Tom Wolfe 1979 320
Chars
American
Chars
Hardcover
Chars
Book
Chars
****
Chars
4 stars
ID Rating ProductID
787 **** 1579124585
747 4 stars 1579124585
Tables
ID Title Author Year Pages
1579124585 The Right Stuff Tom Wolfe 1979 304
ID Keyword ProductID
4711 Book 1579124585
1843 Hardcover 1579124585
2012 American 1579124585

40


Spot The Differences
Object Graph
• Extensional identity
• Pointer parent → child
Title Author Year Pages Keywords Ratings
The Right Stuff Tom Wolfe 1979 320
Chars
American
Chars
Hardcover
Chars
Book
Chars
****
Chars
4 stars
ID Rating ProductID
787 **** 1579124585
747 4 stars 1579124585
Tables
• Intensional identity
• Pointer child → parent
ID Title Author Year Pages
1579124585 The Right Stuff Tom Wolfe 1979 304
ID Keyword ProductID
4711 Book 1579124585
1843 Hardcover 1579124585
2012 American 1579124585

41

Categories

Google

mathematics arrows reversed

Web Images Maps Shopping More Search tools

About 11,500,000 results (0.56 seconds)

[Category (**mathematics**) - Wikipedia, the free encyclopedia]

en.wikipedia.org/wiki/Category_(**mathematics**)

In **mathematics**, a category is an algebraic structure that comprises "objects" that are **...** A category has two basic properties: the ability to compose the **arrows** **....** in the original category but the **arrows** are those of the original category **reversed**.

Definition - History - Small and large categories - Examples

42

42

Categories

Categories

[edit]

A *category* C consists of the following three mathematical entities:

- A class ob(C), whose elements are called *objects*;
- A class hom(C), whose elements are called morphisms or maps or *arrows*. Each morphism ***f*** has a *source object **a*** and *target object **b***.
  The expression $f : a \to b$, would be verbally stated as "f is a morphism from a to b".
  The expression **hom(*a*, *b*)** — alternatively expressed as $\mathbf{hom}_C(a, b)$, **mor(*a*, *b*)**, or ***C*(*a*, *b*)** — denotes the *hom-class* of all morphisms from a to b.
- A binary operation $\circ$, called *composition of morphisms*, such that for any three objects a, b, and c, we have hom(b, c) × hom(a, b) → hom(a, c). The composition of $f : a \to b$ and $g : b \to c$ is written as $g \circ f$ or gf,[3] governed by two axioms:
  - Associativity: If $f : a \to b$, $g : b \to c$ and $h : c \to d$ then $h \circ (g \circ f) = (h \circ g) \circ f$, and
  - Identity: For every object x, there exists a morphism $1_x : x \to x$ called the *identity morphism for* x, such that for every morphism $f : a \to b$, we have $1_b \circ f = f = f \circ 1_a$.

    From the axioms, it can be proved that there is exactly one identity morphism for every object. Some authors deviate from the definition just given by identifying each object with its identity morphism.

43

43

Duality

Further concepts and results

[edit]

The definitions of categories and functors provide only the very basics of categorical algebra; additional important topics are listed below. Although there are strong interrelations between all of these topics, the given order can be considered as a guideline for further reading.

- The functor category D^C has as objects the functors from C to D and as morphisms the natural transformations of such functors. The Yoneda lemma is one of the most famous basic results of category theory; it describes representable functors in functor categories.
- Duality: Every statement, theorem, or definition in category theory has a *dual* which is essentially obtained by "reversing all the arrows". If one statement is true in a category C then its dual will be true in the dual category C^{op}. This duality, which is transparent at the level of category theory, is often obscured in applications and can lead to surprising relationships.
- Adjoint functors: A functor can be left (or right) adjoint to another functor that maps in the opposite direction. Such a pair of adjoint functors typically arises from a construction defined by a universal property; this can be seen as a more abstract and powerful view on universal properties.

44

44

Duality: Tables vs Objects

$$S \xrightarrow{f} T$$

ForeignKey(f,S) = PrimaryKey(T)

$$S \xleftarrow{f} T$$

Address(S) = Property(f,T)

45

45

Intension vs Extension

- “In logic and mathematics, an *intensional* definition gives the meaning of a term by specifying all the properties required to come to that definition, that is, the necessary and sufficient conditions for belonging to the set being defined.”

- “An *extensional* definition of a concept or term formulates its meaning by specifying its extension, that is, every object that falls under the definition of the concept or term in question.”

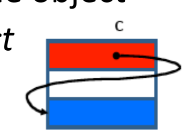
46

46

Intension vs Extension

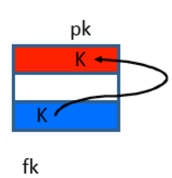
Subject | Verb | Direct Object

- Object
  - A memory location contains an object
  - A pointer is the memory location of some object
  - *Memory location is not part of the object*



Subject | Verb | Direct Object

- Rows
  - A row has a primary key
  - A foreign key is the value of a primary key
  - *Primary key is part of a row*



47

47

F-algebra

Definition [edit]

An ***F*-algebra** for an endofunctor

$$F : \mathcal{C} \longrightarrow \mathcal{C}$$

is an object A of $\mathcal{C}$ together with a $\mathcal{C}$-morphism

$$\alpha : FA \longrightarrow A.$$

In this sense *F*-algebras are dual to *F*-coalgebras.

$$\begin{tikzcd}
F(A) \arrow[r, "\alpha_A"] \arrow[d, "F(f)"'] & A \arrow[d, "f"] \\
F(B) \arrow[r, "\alpha_B"'] & B
\end{tikzcd}$$

- Example: `F(List) = int x List`

α_{List} = `insert : int x List → List`

48

48

F-algebra: Constructors

Selectors of L	Constructors of L	
	empty	**insert(n,L1)**
isEmpty(L)	true	false
head(L)	-	n
tail(L)	-	L1
append(L,L2)	L2	insert(n, append(L1,L2))

49

49

F-coalgebra

- Example: `F(List) = int x List`

$\alpha_{\texttt{List}}$ = `head_tail : List → int x List`

Definition [edit]

An F-**coalgebra** for an endofunctor

$$F : \mathcal{C} \longrightarrow \mathcal{C}$$

is an object A of $\mathcal{C}$ together with a $\mathcal{C}$-morphism

$$\alpha : A \longrightarrow FA.$$

$$\begin{tikzcd} A \arrow[r, "\alpha_A"] \arrow[d, "f"'] & F(A) \arrow[d, "F(f)"] \\ B \arrow[r, "\alpha_B"'] & F(B) \end{tikzcd}$$

50

50

F-coalgebra: Observer

Selectors of L	Constructors of L	
	empty	**insert(n,L1)**
isEmpty	true	false
head(L)	-	n
tail(L)	-	L1
append(L1,L2)	L2	insert(n, append(L1,l2))

51

51

F-algebra vs F-coalgebra

Definition [edit]

An ***F*-algebra** for an endofunctor

$$F : \mathcal{C} \longrightarrow \mathcal{C}$$

is an object A of $\mathcal{C}$ together with a $\mathcal{C}$-morphism

$$\alpha : FA \longrightarrow A.$$

In this sense *F*-algebras are dual to *F*-coalgebras.

Definition [edit]

An F**-coalgebra** for an endofunctor

$$F : \mathcal{C} \longrightarrow \mathcal{C}$$

is an object A of $\mathcal{C}$ together with a $\mathcal{C}$-morphism

$$\alpha : A \longrightarrow FA.$$

$$\begin{tikzcd} F(A) \arrow[r, "\alpha_A"] \arrow[d, "F(f)"'] & A \arrow[d, "f"] \\ F(B) \arrow[r, "\alpha_B"'] & B \end{tikzcd}$$

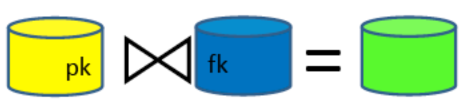
$$\begin{tikzcd} A \arrow[r, "\alpha_A"] \arrow[d, "f"'] & F(A) \arrow[d, "F(f)"] \\ B \arrow[r, "\alpha_B"'] & F(B) \end{tikzcd}$$

52

52

Relational Algebra

- Algebraic: Table ⋈ Table → Table



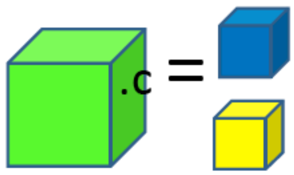
- Join *constructs* new row
  by combining other rows

53

53

Object Coalgebra

- coAlgebraic: Object . Member → List<Object>



- Member access *destructs* existing object into constituent objects

54

54

In other words…

- *…Key-Value Store*

  is dual to

  Primary/Foreign-Key Store!

55

55

Duality

SQL	coSQL
Children point to parents	Parents point to children
Closed World	Open world
Entities have identity (extensional)	Environment determines identity (intensional)
Synchronous (ACID)	Asynchronous (BASE)
Environment coordinates changes (transactions)	Entities responsible to react to changes (eventually consistent)
Not compositional	Compositional
Query optimizer	Developer/pattern

56

56

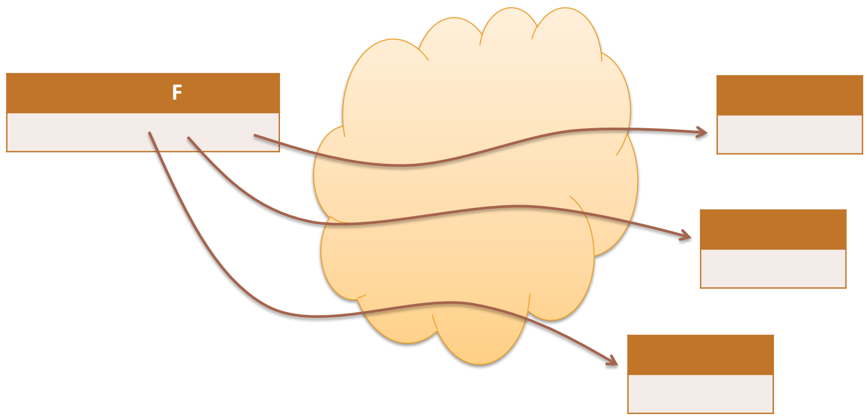
PK
13
42

PK	FK
A	13
B	13
C	42

- Open world
- Cannot join
- Cannot coordinate transactions
- Cannot maintain referential integrity

57

57



- Pre-computed indexes
- Eventually consistent
- Weak pointers (expect 404)

58

58

Life Beyond Distributed Transactions: An Apostate's Opinion

- Entities are collections of named (keyed) data which may be atomically updated within the entity but never atomically updated across entities.

- Pat Helland

59

59

SimpleDB Datamodel

- Domain ::= {Item; Row}*
- Row ::= { …; Attribute = Value+; … }
- Value ::= string | key

Title	Author	Year	Pages	Keywords	Ratings
The Right Stuff	Tom Wolfe	1979	320	Hardcover American Book	**** 4 stars

- Mathematical dual of flat relational tables with scalars in columns (dual of 1-NF)

60

60

SimpleDB Downside

Title	Author	Year	Pages	Keywords	Ratings
The Right Stuff	Tom Wolfe	1979	320	Hardcover American Book	**** 4 stars

- No way to retrieve multi-valued attributes using select query. Needs two round trips (can batch writes).

```
sdb.GetAttributes(new GetAttributesRequest
{
   AttributeName = {"Keyword", "Rating"},
   DomainName = "Books",
   ItemName = "… itemName() from query …",
});
```

61

61

HTML5 Storage

```
interface Storage {
  readonly attribute unsigned long length;
  getter DOMString key(in unsigned long index);
  getter any getItem(in DOMString key);
  setter creator void setItem(in DOMString key,
                              in any data);
  deleter void removeItem(in DOMString key);
  void clear();
}
```

Mathematical dual of relational tables with blobs.

62

62

Conclusion

- Co-relational: an alternative foundation for ~~NoSQL~~ coSQL databases
  - SimpleDB, Dynamo
- Other applications of category theory:
  - Monads: abstract over query domains
  - i.e. LINQ!

63

63