

Основни сведения за NoSQL бази от данни. Шаблони за денормализация. Прости CRUD примери с MongoDB

Калин Георгиев

20 април 2021 г.

Relational vs. NoSQL DB



Релационни БД vs. Нерелационни БД

Books

id	title	genre
0	Програмиране на C++	Учебна
1	Сборник задачи по C++	Учебна

Authors

id	name	position
0	Магдалина Тодорова	Просефор
1	Калин Георгиев	Асистент

BooksAuthors

idBook	idAuthor
0	0
1	0
1	1

Заявка

```
SELECT Authors.name FROM (Books
  INNER JOIN BooksAuthors ON
    Books.id = BooksAuthors.idBook
) INNER JOIN Authors ON
  BooksAuthors.idAuthor = Author
.id WHERE Books.id = 1;
```

```
Books[0] = {id:0,
  title:"Програмиране на C++",
  authors:
    [{id:0, name: "Магдалина
      Тодорова"}]
}
Books[1] = {id:1,
  title:"Сборник задачи C++",
  authors:
    [{id:0, name: "Магдалина
      Тодорова"},
     {id:1, name: "Калин Георгиев"}]}
```

Заявка

```
db.books
  .findOne({id:1})
  .authors
  .map(a=>a.name);
```

Релационни БД vs. Нерелационни БД

Books

id	title	genre
0	Програмиране на C++	Учебна
1	Сборник задачи по C++	Учебна

Authors

id	name	position
0	Магдалина Тодорова	Просефор
1	Калин Георгиев	Асистент

BooksAuthors

idBook	idAuthor
0	0
1	0
1	1

Заявка

```
SELECT Authors.name FROM (Books
  INNER JOIN BooksAuthors ON
    Books.id = BooksAuthors.idBook
) INNER JOIN Authors ON
  BooksAuthors.idAuthor = Author
.id WHERE Books.id = 1;
```

```
Books[0] = {id:0,
  title:"Програмиране на C++",
  authors:
    [{id:0, name: "Магдалина
      Тодорова"}]}
```

```
Books[1] = {id:1,
  title:"Сборник задачи C++",
  authors:
    [{id:0, name: "Магдалина
      Тодорова"},
     {id:1, name: "Калин Георгиев"}]}
```

Заявка

```
db.books
  .findOne({id:1})
  .authors
  .map(a=>a.name);
```

Релационни БД vs. Нерелационни БД

Books

id	title	genre
0	Програмиране на C++	Учебна
1	Сборник задачи по C++	Учебна

Authors

id	name	position
0	Магдалина Тодорова	Просефор
1	Калин Георгиев	Асистент

BooksAuthors

idBook	idAuthor
0	0
1	0
1	1

Заявка

```
SELECT Authors.name FROM (Books
  INNER JOIN BooksAuthors ON
    Books.id = BooksAuthors.idBook
) INNER JOIN Authors ON
  BooksAuthors.idAuthor = Author
.id WHERE Books.id = 1;
```

```
Books[0] = {id:0,
  title:"Програмиране на C++",
  authors:
    [{id:0, name: "Магдалина
      Тодорова"}]
}

Books[1] = {id:1,
  title:"Сборник задачи C++",
  authors:
    [{id:0, name: "Магдалина
      Тодорова"},
     {id:1, name: "Калин Георгиев"}]}
```

Заявка

```
db.books
  .findOne({id:1})
  .authors
  .map(a=>a.name);
```

Релационни БД vs. Нерелационни БД

Books

id	title	genre
0	Програмиране на C++	Учебна
1	Сборник задачи по C++	Учебна

Authors

id	name	position
0	Магдалина Тодорова	Просефор
1	Калин Георгиев	Асистент

BooksAuthors

idBook	idAuthor
0	0
1	0
1	1

Заявка

```
SELECT Authors.name FROM (Books
  INNER JOIN BooksAuthors ON
    Books.id = BooksAuthors.idBook
) INNER JOIN Authors ON
  BooksAuthors.idAuthor = Author
.id WHERE Books.id = 1;
```

```
Books[0] = {id:0,
  title:"Програмиране на C++",
  authors:
    [{id:0, name: "Магдалина
      Тодорова"}]
}

Books[1] = {id:1,
  title:"Сборник задачи C++",
  authors:
    [{id:0, name: "Магдалина
      Тодорова"},
     {id:1, name: "Калин Георгиев"}]}
```

Заявка

```
db.books
  .findOne({id:1})
  .authors
  .map(a=>a.name);
```

Релационни БД vs. Нерелационни БД

Java

```
package com.helloworld;

public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
    }
}
```

Node.js

```
// Call the console.log function.
console.log("Hello World");
```

НРСУБД: Характеристика и видове

Характеристика

- Класическият релационен модел е неприложим
- Не поддържат информацията под формата на таблици и не предоставят поддръжка на стандартния език за структурирани заявки SQL
- Често не предоставят функционалност отвъд съхранението на записите
- Нарушават ACID

Видове

- Key-value
- Графови бази данни
- Документни бази данни

НРСУБД: Характеристика и видове

Характеристика

- Класическият релационен модел е неприложим
- Не поддържат информацията под формата на таблици и не предоставят поддръжка на стандартния език за структурирани заявки SQL
- Често не предоставят функционалност отвъд съхранението на записите
- Нарушават ACID

Видове

- Key-value
- Графови бази данни
- Документни бази данни

НРСУБД: Проблем с референтната цялост

- СУБД не поддържат ограничения за цялостност (relational constraints)
- В частност външни ключове
- Други ACID предизвикателства

Някои представители и приложения

- Key-value: Apache Cassandra, Dynamo, memcached, Redis, BigTable,
- Document: Apache CouchDB, MongoDB, SimpleDB

Шаблони за денормализация

“Denormalization” patterns

- Collapsing relations
- Partitioning relation
- Redundant attributes
- Derived attributes

Seung Kyoon Shin, G. Lawrence Sanders Denormalization strategies for data retrieval from data warehouses, Decision Support Systems 42 (2006) 267–282

Collapsing relations

EMPLOYEE (Employee_ID, Last_Name, First_Name, Office_Location)
WORKSTATIONS (WS_ID, WS_Type, Operating_System, Employee_ID)

EMPLOYEE(Employee_ID, Last_Name, First_Name, Office_Location,
 WS_ID, WS_Type, Operating_System)

```
{ Employee_ID:"12",  
  Last_Name: "Georgiev",  
  First_Name: "Kalin",  
  Office_Location: "Sofia",  
  Workstations:  
    [{WS_ID:70, WS_Type: "Desktop", Operating_System: "Ubuntu Linux"},  
     {WS_ID:71, WS_Type: "Laptop", Operating_System: "MacOS"}]}
```

Partitioning relations (vertical)

PART (Part_ID, Width, Length, Height, Weight, ...,
Price, Stock, Supplier_ID, Warehouse,...)

PART_SPEC_INFO (Part_ID, Width, Length,
Height, Weight, Strength,...)

PART_INVENTORY_INFO (Part_ID, Price,
Stock, Supplier_ID, Warehouse,...)

Partitioning relations (horizontal)

```
SALE_HISTORY (Sale_ID, Timestamp, Store_ID,  
              Customer_ID, Amount, ...)
```

```
SALE_HISTORY_Period_1 (Sale_ID, Timestamp, Store_ID,  
                      Customer_ID, Amount, ...)
```

```
SALE_HISTORY_Period_2 (Sale_ID, Timestamp, Store_ID,  
                      Customer_ID, Amount, ...)
```


Redundant attributes

```
PART (Part_ID, Width, Length, Height,  
      Weight, Price, Stock, Supplier_ID, ...)  
SUPPLIER (Supplier_ID, Supplier_Name, Address,  
          State, ZIP, Phone, Sales_Rep, ...)
```

```
PART (Part_ID, Width, Length, Height, Weight,  
      Price, Stock, Supplier_ID, Supplier_Name, ...)  
SUPPLIER (Supplier_ID, Supplier_Name, Address,  
          State, ZIP, Phone, Sales_Rep, ...)
```

Derived attributes

```
CUSTOMER (Customer_ID, Customer_Name, Address, ZIP)
ORDER (Order_ID, Customer_ID, Order_Date,
       Standard_Price, Quantity)
```

```
CUSTOMER (Customer_ID, Customer_Name,
          Address, ZIP, Sum_Of_Purchases)
ORDER (Order_ID, Customer_ID, Order_Date,
       Standard_Price, Quantity)
```

Прости CRUD заявки с MongoDB

“CRUD” заявки

- Create
- Read
- Update
- Delete

Read

```
var bookOne =  
  db.books.findOne ({id:1});  
  
var books =  
  db.books.find ({id: { $lt: 2}});  
  
var kalinsBooks =  
  db.books.find(  
    {authors:  
      {$elemMatch:  
        {name: "Калин Георгиев" }}}});
```

```
Books[0] =  
  {id:0,  
    title:"Програмиране на C++",  
    authors:  
      [{id:0, name: "Магдалина Тодорова"}] }  
Books[1] =  
  {id:1,  
    title:"Сборник задачи C++",  
    authors:  
      [{id:0, name: "Магдалина Тодорова"},  
        {id:1, name: "Калин Георгиев"}] }
```

Create

```
use kalindb;

KalinsNewBook =
  {id:3,
   title:"Нещо умно",
   authors:
     [{id:1, name: "Калин Георгиев"}]};

db.books.insert (KalinsNewBook);
```

Update

```
db.books.update(  
  { id: 3 },  
  { $set: { title: "Нещо нетолковаумно" } },  
  { multi: false });  
)
```

Delete

```
db.books.remove({ id: 3 });
```


MapReduce

Нова концепция за “заявки”

```
[1,2,3].map ((x)=>x+1);
```

```
[1,2,3].reduce ((sum,current)=>sum+current);
```

Нова концепция за “заявки”

```
[1,2,3].map ((x)=>x+1);
```

```
[1,2,3].reduce ((sum,current)=>sum+current);
```

MapReduce

```
people[0] = {gender:"male", age:34};
people[1] = {gender:"female", age:24};
people[2] = {gender:"male", age:35};
people[3] = {gender:"female", age:54};
```

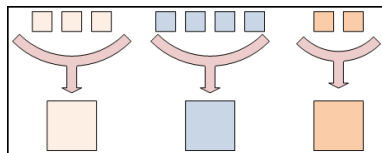
```
var mapFunction = function(){
  emit(this.gender, this.age);
};
```

```
{key:"male", values:[34,35]}
{key:"female", values:[24,54]}
```

```
var reduceFunction = function(keyGender, valuesAges){
  return Array.sum(valuesAges)/valuesAges.length;
};
```

```
reduce ("male", [34,35]) --> 34.5
reduce ("female", [24,54]) --> 39
```

```
db.people.mapReduce ( mapFunction, reduceFunction, { out: "mrexample" }));
```



MapReduce

```
people[0] = {gender:"male", age:34};  
people[1] = {gender:"female", age:24};  
people[2] = {gender:"male", age:35};  
people[3] = {gender:"female", age:54};
```

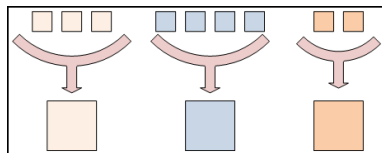
```
var mapFunction = function(){  
  emit(this.gender, this.age);  
};
```

```
{key:"male", values:[34,35]}  
{key:"female", values:[24,54]}
```

```
var reduceFunction = function(keyGender, valuesAges){  
  return Array.sum(valuesAges)/valuesAges.length;  
};
```

```
reduce ("male", [34,35]) --> 34.5  
reduce ("female", [24,54]) --> 39
```

```
db.people.mapReduce ( mapFunction, reduceFunction, { out: "mrexample" }));
```



MapReduce

```
people[0] = {gender:"male", age:34};
people[1] = {gender:"female", age:24};
people[2] = {gender:"male", age:35};
people[3] = {gender:"female", age:54};
```

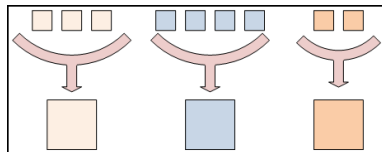
```
var mapFunction = function(){
  emit(this.gender, this.age);
};
```

```
{key:"male", values:[34,35]}
{key:"female", values:[24,54]}
```

```
var reduceFunction = function(keyGender, valuesAges){
  return Array.sum(valuesAges)/valuesAges.length;
};
```

```
reduce ("male", [34,35]) --> 34.5
reduce ("female", [24,54]) --> 39
```

```
db.people.mapReduce ( mapFunction, reduceFunction, { out: "mrexample" }));
```



MapReduce

```
people[0] = {gender:"male", age:34};  
people[1] = {gender:"female", age:24};  
people[2] = {gender:"male", age:35};  
people[3] = {gender:"female", age:54};
```

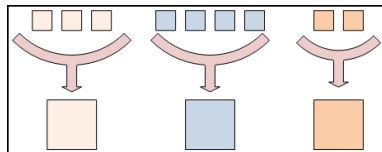
```
var mapFunction = function(){  
  emit(this.gender, this.age);  
};
```

```
{key:"male", values:[34,35]}  
{key:"female", values:[24,54]}
```

```
var reduceFunction = function(keyGender, valuesAges){  
  return Array.sum(valuesAges)/valuesAges.length;  
};
```

```
reduce ("male", [34,35]) --> 34.5  
reduce ("female", [24,54]) --> 39
```

```
db.people.mapReduce ( mapFunction, reduceFunction, { out: "mrexample" }));
```



MapReduce

```
people[0] = {gender:"male", age:34};  
people[1] = {gender:"female", age:24};  
people[2] = {gender:"male", age:35};  
people[3] = {gender:"female", age:54};
```

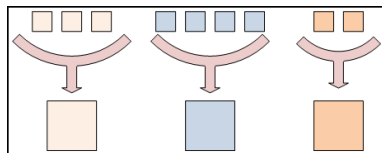
```
var mapFunction = function(){  
  emit(this.gender, this.age);  
};
```

```
{key:"male", values:[34,35]}  
{key:"female", values:[24,54]}
```

```
var reduceFunction = function(keyGender, valuesAges){  
  return Array.sum(valuesAges)/valuesAges.length;  
};
```

```
reduce ("male", [34,35]) --> 34.5  
reduce ("female", [24,54]) --> 39
```

```
db.people.mapReduce ( mapFunction, reduceFunction, { out: "mrexample" }));
```



MapReduce

```
people[0] = {gender:"male", age:34};  
people[1] = {gender:"female", age:24};  
people[2] = {gender:"male", age:35};  
people[3] = {gender:"female", age:54};
```

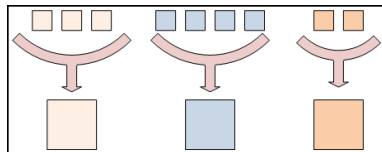
```
var mapFunction = function(){  
  emit(this.gender, this.age);  
};
```

```
{key:"male", values:[34,35]}  
{key:"female", values:[24,54]}
```

```
var reduceFunction = function(keyGender, valuesAges){  
  return Array.sum(valuesAges)/valuesAges.length;  
};
```

```
reduce ("male", [34,35]) --> 34.5  
reduce ("female", [24,54]) --> 39
```

```
db.people.mapReduce ( mapFunction, reduceFunction, { out: "mrexample" }));
```



Благодаря за вниманието!