Yalantis

Лекция 12: Models

Курс лекций по основами web-разработки на языке программирования Ruby

Yalantis

Что такое данные?

Поддающееся многократной интерпретации представление информации в формализованном виде, пригодном для передачи, связи или обработки

Yalantis

База данных

База данных (БД) — это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов.

type	available	location
‘car’	‘t’	0
‘bike’	‘f’	1

Yalantis

Реляционные БД

1. Необходимость соответствия базы данных требованиям ACID (Atomicity, Consistency, Isolation, Durability — атомарность, непротиворечивость, изолированность, долговечность). Это позволяет уменьшить вероятность неожиданного поведения системы и обеспечить целостность базы данных.
2. Данные, с которыми вы работаете, структурированы, при этом структура не подвержена частым изменением.
3. Имеются логические требования к данным, которые могут быть определены заранее.

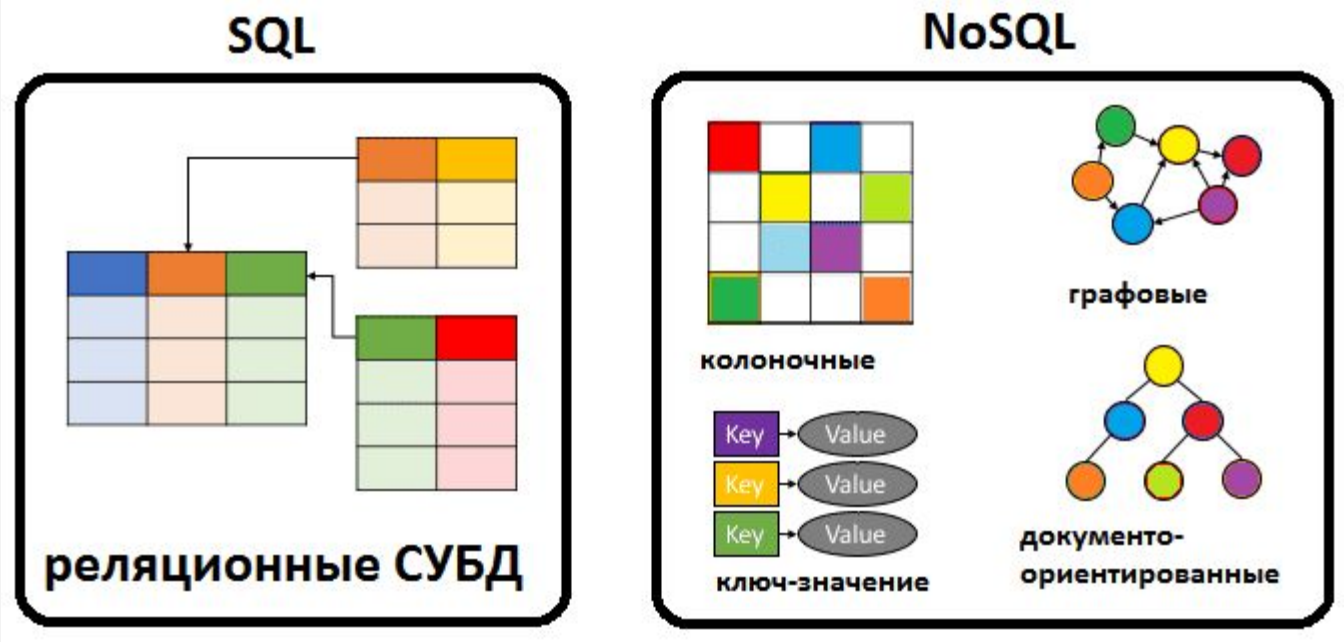
Yalantis

Нереляционные БД

1. Хранение больших объёмов неструктурированной информации. База данных NoSQL не накладывает ограничений на типы хранимых данных. Более того, при необходимости в процессе работы можно добавлять новые типы данных.
2. Быстрая разработка. Если вы разрабатываете систему, используя agile-методы, применение реляционной БД способно замедлить работу. NoSQL базы данных не нуждаются в том же объёме подготовительных действий, которые обычно нужны для реляционных баз.
3. Требования к данным нечёткие, неопределённые, или развивающиеся с развитием проекта.
4. Одни из основных требований к базе данных — скорость обработки данных и масштабируемость.

```
var cars = [
  { Model: "BMW", Color: "Red", Manufactured: 2016 },
  { Model: "Mercedes", Type: "Coupe", Color: "Black", Manufactured: "1-1-2017" }
];
```

Yalantis


SQL
реляционные СУБД
NoSQL
колоночные
графовые
Key
Value
Key
Value
Key
Value
ключ-значение
документо-
ориентированные

Yalantis

Система управления базой данных

Это комплекс программных средств, необходимых для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации

Yalantis

Реляционные БД

users

id (PK)	email	ecnrypted_password
134	john.doe@test.test	asdknfjn3@4fsdajnds

posts

id (PK)	user_id (FK)	title
23	134	My first post

Yalantis

Нереляционные БД

```
127.0.0.1:6379> SET lol kek
OK
127.0.0.1:6379> GET lol
"kek"
127.0.0.1:6379> KEYS *
1) "lol"
127.0.0.1:6379> DEL lol
(integer) 1
127.0.0.1:6379> GET lol
(nil)
127.0.0.1:6379> KEYS *
(empty list or set)
127.0.0.1:6379>
```

Yalantis

Structured Query Language

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Yalantis

1. Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

- CREATE
- ALTER
- DROP
- TRUNCATE

```
[local]:5432 bavykin@bavykin=# CREATE DATABASE test;
CREATE DATABASE
Time: 111.192 ms
[local]:5432 bavykin@bavykin=# \c test
You are now connected to database "test" as user "bavykin".
[local]:5432 bavykin@test=# CREATE TABLE users (id int PRIMARY KEY, name varchar(255));
CREATE TABLE
Time: 4.838 ms
[local]:5432 bavykin@test=#
```

Yalantis

2. Data Manipulation Language

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

- INSERT
- UPDATE
- DELETE

```
[local]:5432 bavykin@test=# INSERT INTO users VALUES (1, 'John');
INSERT 0 1
Time: 2.074 ms
[local]:5432 bavykin@test=# DELETE FROM users where id = 1;
DELETE 1
Time: 1.017 ms
[local]:5432 bavykin@test=#
```

Yalantis

3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

- Grant
- Revoke

```
[local]:5432 bavykin@test=# GRANT SELECT ON users TO bavykin;
GRANT
Time: 0.558 ms
[local]:5432 bavykin@test=# \dp users;
                                Access privileges
 Schema | Name  | Type  |     Access privileges     | Column privileges | Policies
--------+-------+-------+---------------------------+-------------------+----------
 public | users | table | bavykin=arwdDxt/bavykin   |                   |
(1 row)

[local]:5432 bavykin@test=#
```

Yalantis

4. Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

- COMMIT
- ROLLBACK
- SAVEPOINT

```
[local]:5432 bavykin@test=# BEGIN;
BEGIN
Time: 0.181 ms
[local]:5432 bavykin@test=# SELECT * FROM users;
 id |   name
----+----------
  2 | Iron Man
(1 row)

Time: 0.208 ms
[local]:5432 bavykin@test=# DELETE FROM users;
DELETE 1
Time: 0.235 ms
[local]:5432 bavykin@test=# SELECT * FROM users;
 id | name
----+------
(0 rows)

Time: 0.231 ms
[local]:5432 bavykin@test=# ROLLBACK;
ROLLBACK
Time: 0.153 ms
[local]:5432 bavykin@test=# SELECT * FROM users;
 id |   name
----+----------
  2 | Iron Man
(1 row)

Time: 0.271 ms
[local]:5432 bavykin@test=#
```

Yalantis

5. Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

- SELECT

```
[local]:5432 bavykin@test=# SELECT name FROM users WHERE users.id > 1;
 name
 Iron Man
(1 row)

Time: 0.351 ms
```

Yalantis

Distinct

The SELECT DISTINCT statement is used to return only distinct (different) values.

```
[local]:5432 bavykin@bossdesk_development=# SELECT DISTINCT time_zone FROM users;
```

time_zone
Eastern Time (US & Canada)
Tallinn
Pacific Time (US & Canada)
Ekaterinburg

```
(4 rows)

Time: 0.615 ms
[local]:5432 bavykin@bossdesk_development=# SELECT COUNT(*) FROM users WHERE time_zone = 'Eastern Time (US & Canada)' ;
```

count
292

```
(1 row)

Time: 0.516 ms
[local]:5432 bavykin@bossdesk_development=#
```

Yalantis

SQL AND, OR and NOT Operators

The WHERE clause can be combined with AND, OR, and NOT operators.

```
[local]:5432 bavykin@bossdesk_development=# SELECT email FROM users WHERE id > 1 AND email LIKE '%test%' AND NOT email LIKE '%.com';
        email
---------------------
 someone@holtca.comtest
 testuser@teset.test
 requester@test.test
(3 rows)

Time: 0.547 ms
```

Yalantis

SQL ORDER BY Keyword

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

```
[local]:5432 bavykin@bossdesk_development=# SELECT id FROM accounts ORDER BY id LIMIT 2 OFFSET 1;
 id
----
  2
  3
(2 rows)

Time: 0.353 ms
```

```
[local]:5432 bavykin@bossdesk_development=# SELECT id FROM accounts ORDER BY 1 LIMIT 2 OFFSET 1;
 id
----
  2
  3
(2 rows)

Time: 0.302 ms
```

Yalantis

SQL NULL Values

```
[local]:5432 bavykin@bossdesk_development=# SELECT id FROM users WHERE email IS NULL LIMIT 1;
```

id
17132

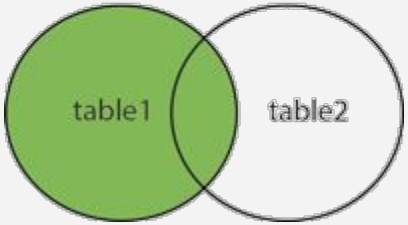
```
(1 row)

Time: 0.382 ms
```

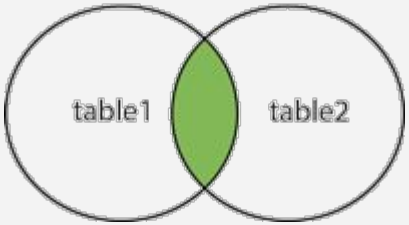
Yalantis

JOINS

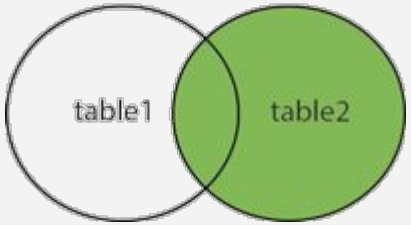
LEFT JOIN



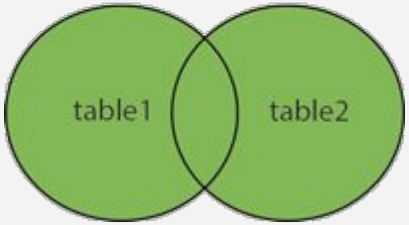
INNER JOIN



RIGHT JOIN



FULL OUTER JOIN



Yalantis

INNER JOIN

```
[local]:5432 bavykin@bossdesk_development=# SELECT users.account_id, accounts.id FROM users INNER JOIN accounts ON users.account_id = accounts.id LIMIT 1;
```

account_id	id
1	1

```
(1 row)

Time: 1.099 ms
```

LEFT JOIN

```
[local]:5432 bavykin@bossdesk_development=# SELECT articles.id, article_feedbacks.article_id FROM articles LEFT JOIN article_feedbacks ON articles.id = article_feedbacks.article_id ORDER BY articles.id DESC LIMIT 2;
```

id	article_id
13	(NULL)
12	12

```
(2 rows)

Time: 0.487 ms
```

Yalantis

FULL JOIN

FirstName	LastName	CustomerCountry	SupplierCountry	CompanyName
NULL	NULL	NULL	Australia	Pavlova, Ltd.
NULL	NULL	NULL	Australia	G'day, Mate
NULL	NULL	NULL	Japan	Tokyo Traders
NULL	NULL	NULL	Japan	Mayumi's
NULL	NULL	NULL	Netherlands	Zaanse Snoepfabriek
NULL	NULL	NULL	Singapore	Leka Trading
Patricio	Simpson	Argentina	NULL	NULL
Yvonne	Moncada	Argentina	NULL	NULL
Sergio	Gutiérrez	Argentina	NULL	NULL
Georg	Pipps	Austria	NULL	NULL
Roland	Mendel	Austria	NULL	NULL
Pascale	Cartrain	Belgium	NULL	NULL
Catherine	Dewey	Belgium	NULL	NULL
Bernardo	Batista	Brazil	Brazil	Refrescos Americanas LTDA
Lúcia	Carvalho	Brazil	Brazil	Refrescos Americanas LTDA
Janete	Limeira	Brazil	Brazil	Refrescos Americanas LTDA

Yalantis

SQL GROUP BY Statement

The GROUP BY statement groups rows that have the same values into summary rows

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

```
[local]:5432 bavykin@bossdesk_development=# SELECT accounts.id, COUNT(users.id) FROM accounts LEFT JOIN users ON accounts.id = users.account_id GROUP BY accounts.id;
```

id	count
7	1
1	290
5	1
4	1
2	2
6	1
3	1

```
(7 rows)

Time: 0.720 ms
```

Yalantis

The SQL HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

```
[local]:5432 bavykin@bossdesk_development=# SELECT accounts.id, COUNT(users.id) FROM accounts LEFT JOIN users ON accounts.id = users.account_id GROUP BY accounts.id HAVING COUNT(users.id) > 1;
```

id	count
1	290
2	2

(2 rows)

Time: 0.654 ms

```
[local]:5432 bavykin@bossdesk_development=# SELECT accounts.id, COUNT(users.id) FROM accounts LEFT JOIN users ON accounts.id = users.account_id WHERE accounts.id = 1 GROUP BY accounts.id HAVING COUNT(users.id) > 1;
```

id	count
1	290

(1 row)

Time: 0.623 ms

Yalantis

CREATE SEQUENCE

CREATE SEQUENCE создаёт генератор последовательности. Эта операция включает создание и инициализацию специальной таблицы имя, содержащей одну строку. Владельцем генератора будет пользователь, выполняющий эту команду.

```
[local]:5432 bavykin@bossdesk_development=# CREATE SEQUENCE test START 5 INCREMENT BY 3 NO MAXVALUE NO MINVALUE;
CREATE SEQUENCE
Time: 1.868 ms
[local]:5432 bavykin@bossdesk_development=# select nextval('test');
 nextval
---------
       5
(1 row)

Time: 0.598 ms
[local]:5432 bavykin@bossdesk_development=# select nextval('test');
 nextval
---------
       8
(1 row)

Time: 0.243 ms
[local]:5432 bavykin@bossdesk_development=#
```

Yalantis

SQL Constraints

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Uniquely identifies a row/record in another table
- CHECK - Ensures that all values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column when no value is specified
- INDEX - Used to create and retrieve data from the database very quickly

Yalantis

Indexes

```
CREATE INDEX index_timesheets_on_agent_id ON public.timesheets USING btree (agent_id);
```

```
[local]:5432 bavykin@bossdesk_development=# EXPLAIN (COSTS OFF) SELECT * FROM users  WHERE id = 1000;
                QUERY PLAN
-------------------------------------------
 Index Scan using users_pkey on users
   Index Cond: (id = 1000)
(2 rows)

Time: 1.718 ms
```

- Indexes should not be used on small tables.
- Tables that have frequent, large batch update or insert operations.
- Indexes should not be used on columns that contain a high number of NULL values.
- Columns that are frequently manipulated should not be indexed.

Yalantis

Indexes with ORDER BY

```
[local]:5432 bavykin@bossdesk_development=# EXPLAIN (COSTS OFF) SELECT * FROM users  WHERE id > 1000 ORDER BY id LIMIT 10;
                QUERY PLAN
-------------------------------------------
 Limit
   ->  Index Scan using users_pkey on users
         Index Cond: (id > 1000)
(3 rows)

Time: 0.833 ms
[local]:5432 bavykin@bossdesk_development=# EXPLAIN (COSTS OFF) SELECT * FROM users  WHERE id > 1000 ORDER BY id;
        QUERY PLAN
---------------------------
 Sort
   Sort Key: id
   ->  Seq Scan on users
         Filter: (id > 1000)
(4 rows)

Time: 0.653 ms
```

Yalantis

1ая нормальная форма

Все поля являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

universe	characters
Marvel	Iron Man, Thor, Hulk
DC	Harley Quinn

universe	characters
Marvel	Iron Man
Marvel	Thor
Marvel	Hulk
DC	Harley Quinn

Yalantis

2ая нормальная форма

1НФ + каждый неключевой атрибут зависит от Первичного Ключа.
Отсутствует частичная зависимость от составного ключа

brand	model	discount
Apple	iPhone 11	10
Apple	iPhone X	10
Xiaomi	Mi Note 8	5
Apple	iPhone SE	10

brand	model
Apple	iPhone 11
Apple	iPhone X
Xiaomi	Mi Note 8
Apple	iPhone SE

brand	discount
Apple	10
Xiaomi	5

Yalantis

3ья нормальная форма

2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

store	model	support
Citrus	iPhone 11	+380631231231
Allo	iPhone X	+380671234567
Allo	Mi Note 8	+380631231231

store	support
Citrus	+380671234567
Allo	+380631231231

store	model
Citrus	iPhone 11
Allo	iPhone X
Allo	Mi Note 8

Yalantis

Денормализация

Это осознанное нарушение нормализация для уменьшения времени обработки запроса. Порой дешевле по ресурсам сделать вспомогательную колонку с дубляжём информации, чем реально выполнять весь запрос

```
[local]:5432 bavykin@bossdesk_development=# SELECT articles.id, article_feedbacks.article_id FROM articles LEFT JOIN article_feedbacks ON articles.id = article_feedbacks.article_id ORDER BY articles.id DESC LIMIT 2;
```

id	article_id
13	(NULL)
12	12

```
(2 rows)

Time: 0.487 ms
```

Вот пример несложного запроса. При увеличении размера БД может быть эффективнее сделать колонку feedbacks_count и хранить в ней количество отзывов

Yalantis

Типы данных PostgreSQL

```
# add column Array of integers
add_column :users, :sibling_ids, :integer, array: true, default: []
```

```
# How to use it
User.where('sibling_ids @> ARRAY[?]::integer[]', ids)
```

Yalantis

https://habr.com/ru/company/ruvds/blog/324936/

https://habr.com/ru/company/postgrespro/blog/326096/

https://tproger.ru/translations/types-of-nosql-db/

https://www.postgresql.org/docs/9.5/datatype.html

https://www.w3schools.com/sql/

https://habr.com/ru/post/64524/