Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

онлайн университет

# Программная инженерия. Разработка ПО (Python для продвинутых специалистов. Машинное обучение)

Модуль: Введение в Python для машинного обучения

Лекция 9: SQL. Коротко о самом важном при работе с данными

Дата: 21.04.2025

ПЕРЕДОВАЯ
ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

# Содержание лекции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

- Как теряются данные?
- Работа с NULL
- Ловушка NOT IN
- Так ли хорошо вы понимаете соединения?
- Встроенные аналитические функции
- Кто живет в базе данных?
- Ловушка с представлениями
- Практическая часть

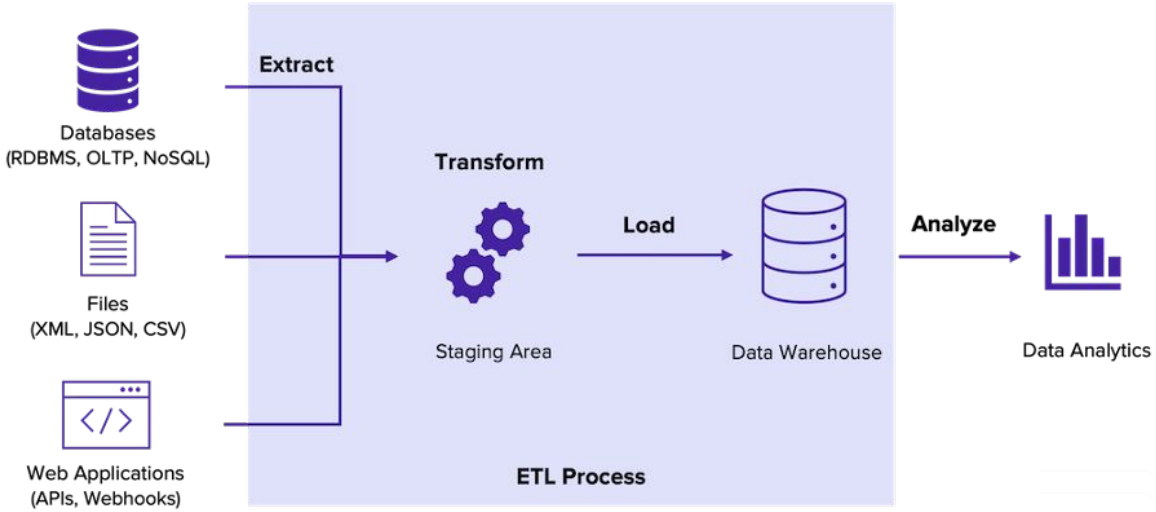
# Как теряются данные?

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Говоря о потере данных мы имеем ввиду ситуации, когда важная для анализа информация была потеряна в ходе ее движения от системы источника до целевого DataSet-а

Вот возможные причины потери данных:

- несинхронность источников
- большие объемы данных
- сетевые или аппаратные проблемы
- ошибки в обработке данных на разных этапах



# Что такое NULL в базе данных?

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Возможно, вы слышали про двоичную (бинарную) логику? Это про True и False.
Базы данных работают по троичной логике: к известным булевым величинам добавляется NULL

| value | NOT value |
|---|---|
| False | True |
| True | False |
| NULL | NULL |

| Value 1 | Value 2 | V1 OR V2 | V1 AND V2 |
|---|---|---|---|
| False | False | False | False |
| False | True | True | False |
| True | False | True | False |
| True | True | True | True |
| False | NULL | NULL | False |
| True | NULL | True | NULL |

**SELECT поля**
**FROM таблица**
**WHERE булево выражение или их *комбинация***

# Ловушка NOT IN

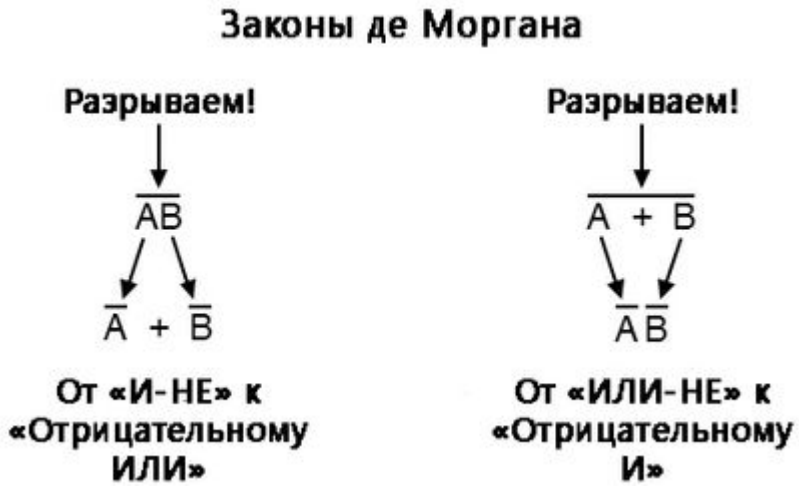
ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

С помощью оператора IN легко и просто выбрать записи, которые совпадают с одним из переданных значений.
Часто это используется и для подзапросов.

```
SELECT поля
FROM таблица
WHERE поле IN (<список значений>/<подзапрос>)
```

В случае с подзапросом есть одно узкое место, которое часто ставит разработчиков в тупик.

Давайте посмотрим на примере.



# Виды соединения таблиц

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Таблицы в базах данных могут быть соединены двумя способами:

1. По вертикали
2. По горизонтали

Вертикальное соединение:

- UNION
- UNION ALL
- EXCEPT (MINUS)
- INTERSECT

Горизонтальное соединение:

- CROSS JOIN
- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- NATURAL JOIN
- * Self JOIN

# Квиз

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

1. Есть две таблицы, в одной из них 10 записей, в другой – 100. Какое минимальное и максимальное количество записей можно получить используя разные виды соединений?

# Квиз



2. Соедините две таблицы всеми видами соединений (без CROSS JOIN).
Соединение происходит следующим шаблоном запроса:

*SELECT Table1.ID ID_1, Table2.ID ID_2*
*FROM Table1*
*<вид соединения> JOIN Table2*
*ON Table1.ID = Table2.ID;*

Table1

| ID |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| NULL |

Table2

| ID |
|---|
| 2 |
| NULL |
| 3 |
| 3 |
| 5 |
| NULL |

# Квиз

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

2. Соедините две таблицы всеми видами соединений (без CROSS JOIN).
Соединение происходит следующим шаблоном запроса:

*SELECT Table1.ID ID_1, Table2.ID ID_2*
*FROM Table1*
*<вид соединения> JOIN Table2*
*ON Table1.ID = Table2.ID;*

Table1

| ID |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| NULL |

Table2

| ID |
|---|
| 2 |
| NULL |
| 3 |
| 3 |
| 5 |
| NULL |

Result (INNER)

| ID_1 | ID_2 |
|---|---|
| 2 | 2 |
| 3 | 3 |
| 3 | 3 |

# Квиз

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

2. Соедините две таблицы всеми видами соединений (без CROSS JOIN).
Соединение происходит следующим шаблоном запроса:

```
SELECT Table1.ID ID_1, Table2.ID ID_2
FROM Table1
    <вид соединения> JOIN Table2
    ON Table1.ID = Table2.ID;
```

Table1

| ID |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| NULL |

Table2

| ID |
|---|
| 2 |
| NULL |
| 3 |
| 3 |
| 5 |
| NULL |

Result (LEFT)

| ID_1 | ID_2 |
|---|---|
| 2 | 2 |
| 3 | 3 |
| 3 | 3 |
| 1 | NULL |
| 4 | NULL |
| NULL | NULL |

# Квиз

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

2. Соедините две таблицы всеми видами соединений (без CROSS JOIN).
Соединение происходит следующим шаблоном запроса:

*SELECT Table1.ID ID_1, Table2.ID ID_2*
*FROM Table1*
*<вид соединения> JOIN Table2*
*ON Table1.ID = Table2.ID;*

Table1

| ID |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| NULL |

Table2

| ID |
|---|
| 2 |
| NULL |
| 3 |
| 3 |
| 5 |
| NULL |

Result (RIGHT)

| ID_1 | ID_2 |
|---|---|
| 2 | 2 |
| 3 | 3 |
| 3 | 3 |
| NULL | NULL |
| NULL | 5 |
| NULL | NULL |

# Квиз



2. Соедините две таблицы всеми видами соединений (без CROSS JOIN).
Соединение происходит следующим шаблоном запроса:

```
SELECT Table1.ID ID_1, Table2.ID ID_2
FROM Table1
    <вид соединения> JOIN Table2
    ON Table1.ID = Table2.ID;
```

Table1

| ID |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| NULL |

Table2

| ID |
|---|
| 2 |
| NULL |
| 3 |
| 3 |
| 5 |
| NULL |

Result (FULL)

| ID_1 | ID_2 |
|---|---|
| 2 | 2 |
| 3 | 3 |
| 3 | 3 |
| 1 | NULL |
| 4 | NULL |
| NULL | NULL |
| NULL | NULL |
| NULL | 5 |
| NULL | NULL |

# Виды соединения таблиц

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Таблицы в базах данных могут быть соединены двумя способами:

1. По вертикали
2. По горизонтали

Вертикальное соединение:

- UNION
- UNION ALL
- EXCEPT (MINUS)
- INTERSECT

Горизонтальное соединение:

- CROSS JOIN
- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- NATURAL JOIN
- * Self JOIN

# Аналитические (оконные) функции

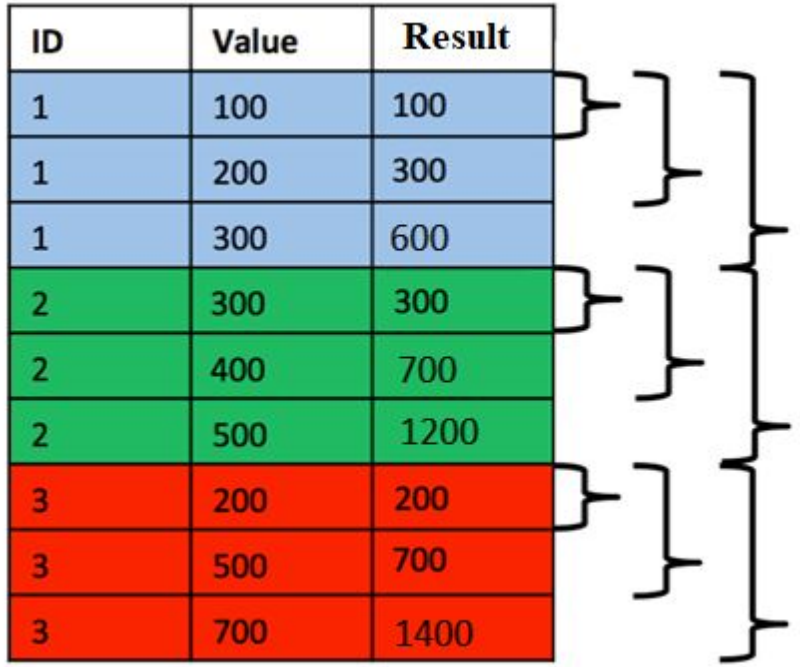
ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Синтаксис оконной функции:

```
analytic_function( <value> ) OVER (
  [PARTITION BY expression1, expression2,...]
  [ORDER BY expression1 [ASC|DESC], expression2,... ])

SELECT
  SUM( value ) OVER( ORDER BY value ) result
FROM
  OrderDetails;

SELECT
  SUM(value) OVER(PARTITION BY id ORDER BY value) result
FROM
  OrderDetails;
```



| ID | Value | Result |
|---|---|---|
| 1 | 100 | 100 |
| 1 | 200 | 300 |
| 1 | 300 | 600 |
| 2 | 300 | 300 |
| 2 | 400 | 700 |
| 2 | 500 | 1200 |
| 3 | 200 | 200 |
| 3 | 500 | 700 |
| 3 | 700 | 1400 |

<u>Наиболее употребляемые аналитические функции:</u>
AVG, MIN, MAX, SUM, COUNT, FIRST_VALUE, LAST_VALUE, LAG, LEAD, ROW_NUMBER, RANK, DENSE_RANK

# Агрегатные (стат) функции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Синтаксис агрегатной функции:

*analytic_function( <value> ) WITHIN GROUP (*
*[ORDER BY expression1 [ASC|DESC], expression2,... ])*

```
SELECT
    AVG(salary) avg_salary,
    MODE() WITHIN GROUP (ORDER BY salary),
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY salary),
    CORR(salary, coalesce(comission_pct,0))
FROM employees;
```

Наиболее употребляемые агрегатные функции:
MODE, PERCENTILE_CONT, PERCENTILE_DISC,
RANK, DENSE_RANK

# Что еще есть в базе данных?



Вот список объектов, которые есть в большинстве реляционных баз данных:

- Таблицы (managed, temp, external)
- Представления и Материализованные представления
- Последовательности
- Индексы
- Хранимые процедуры и функции
- Триггеры
- Роли
- ...

# Представления

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Представление (на сленге «вьюха») – SQL код, сохраненный в справочнике базы данных под своим собственным именем.
Данный код исполняется каждый раз при обращении к представлению.
Обратите внимание, что представление само по себе не хранит данные, только SQL код.

Представления используются для двух целей:
- замаскировать нежелательные данные (best practice)
- спрятать в представление бизнес-логику (worth practice)

Во втором случае возможны нежелательные side-эффекты, например в виде потери производительности.

```
CREATE VIEW <view_name> AS
   SELECT * FROM <table_name> ;

CREATE OR REPLACE VIEW <view_name> AS
   SELECT * FROM <table_name> WHERE 1=1;

DROP VIEW <view_name> ;
```

Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

онлайн университет

# Спасибо за внимание