

SQL



Тора
Самисько



Тора Самисько

GameDev, OAI



План занятия

1. [Общие сведения](#)
2. [Операторы SQL](#)
3. [Операторы определения данных](#)
4. [Операторы манипуляции данных](#)
5. [Операторы доступа к данным](#)
6. [Операторы управления транзакциями](#)
7. [Первичные и внешние ключи](#)
8. [Сложные выборки данных](#)
9. [Индексы](#)
10. [Explain](#)
11. [Итоги](#)
12. [Домашнее задание](#)



Общие сведения

Общие сведения

SQL (structured query language) - декларативный язык программирования, применяемый для создания, модификации, управления данными в реляционной базе данных.

Базовые операции SQL:

- создание в базе данных новой таблицы;
- добавление в таблицу новых записей;
- изменение записей;
- удаление записей;
- выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
- изменение структур таблиц.

Язык SQL представляет собой совокупность операторов, инструкций, вычисляемых функций.

Общие сведения

Преимущества:


- **Независимость от конкретной СУБД**
SQL-запросы могут быть достаточно легко перенесены из одной СУБД в другую.
- **Наличие стандартов**
Наличие стандартов и набора тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту
- **Декларативность**
В SQL-запросах описываются только операции над данными. Реализация запросов осуществляется средствами СУБД.



Операторы SQL

Операторы SQL

Тип оператора	Оператор	Назначение
Определение данных	CREATE	Создание объекта
	ALTER	Изменение объекта
	DROP	Удаление объекта
Манипуляция данными	SELECT	Выборка данных
	INSERT	Вставка данных
	UPDATE	Изменение данных
	DELETE	Удаление данных
Доступ к данным	GRANT	Предоставление прав
	REVOKE	Отзыв прав
	DENY	Запрет на действие
Управление транзакциями	COMMIT	Применение транзакции
	ROLLBACK	Откат изменений
	SAVEPOINT	Деление транзакции



Операторы определения данных

Операторы определения данных

CREATE

- Создание БД

CREATE DATABASE завод;

- Создание таблицы

CREATE TABLE kadry (nomerceh INT, tabnom SERIAL, fio CHAR(20) UNIQUE);

- Создание псевдотаблицы

CREATE VIEW poor AS SELECT tabnom, fio FROM kadry WHERE tabnom < 120;

- Создание индекса

CREATE UNIQUE INDEX indkdtb ON kadry (tabnom);

- Создание синонима имени таблицы

CREATE SYNONYM t1 FOR завод.kadry;

Операторы определения данных

ALTER

- Изменить имя БД
ALTER DATABASE завод MODIFY NAME = factory;
- Изменение имени таблицы
ALTER TABLE кадры RENAME TO persons;
- Изменение столбцов в таблице
*ALTER TABLE кадры ADD (dolzhnost CHAR(20) BEFORE fio),
DROP(tabnom);*
- Упорядочивание таблицы по индексу
ALTER INDEX indkdtb TO CLUSTER;

Операторы определения данных

DROP

- Удалить БД
DROP DATABASE zawod;
- Удалить таблицу
DROP TABLE kadry;
- Удалить индекс
DROP INDEX indkdtb;
- Удалить синоним
DROP SYNONYM t1;
- Удалить псевдотаблицу
DROP VIEW poor;



Операторы манипуляции данных

Операторы манипуляции данными

SELECT

- Вывести все записи таблицы
*SELECT * FROM persons;*
- Вывести количество записей в таблице
SELECT COUNT() FROM persons;*
- Вывести определенные столбцы из таблицы
SELECT fio, tabnom from persons;
- Вывести данные по условию
SELECT fio, tabnom from persons where tabnom>100;
- Вывести только уникальные значения
SELECT DISTINCT fio, tabnom from persons;
- Вывести упорядоченные данные по признаку (ASC - по возрастанию, DESC - по убыванию)
SELECT fio, tabnom from persons ORDER BY tabnom ASC;
- Вывести сгруппированные значения по признаку
SELECT COUNT() from persons GROUP BY fio;*

Операторы манипуляции данными

INSERT

- Вставка данных в таблицу
INSERT INTO persons VALUES (1, 123, "Пупкин");
- Вставка данных в таблицу с указанием столбца
INSERT INTO persons (nomerch, tabno, fio) VALUES (1, 123, "Пупкин");

Операторы манипуляции данными

UPDATE

- Изменение поля в конкретной строке

UPDATE persons SET fio = 'Alfred Schmidt' WHERE tabno = 1;

- Изменение поля во всей таблице

UPDATE persons SET fio = 'Alfred Schmidt';

Важно!

Если не указывать конкретную строку (через оператор WHERE) - изменения затронут всю таблицу.

Операторы манипуляции данными

DELETE

- Удалить все данные в таблице
DELETE FROM persons;
- Удалить конкретную строку (или набор строк)
DELETE FROM persons WHERE tabno=1;

Важно!

Если не указывать конкретную строку (через оператор WHERE) - изменения затронут всю таблицу.

Операторы манипуляции данными

ИТОГ

- Синтаксис создания записи:

INSERT INTO table_name VALUES (value1, value2, value3, ...);

INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);

- Синтаксис получения записи:

*SELECT * FROM table_name;*


SELECT column1, column2, ... FROM table_name;

- Синтаксис обновления записи:

UPDATE table_name SET column1 = value1, ... WHERE condition;

- Синтаксис удаления записи:

DELETE FROM table_name WHERE condition;



Операторы доступа к данным

Операторы доступа к данным

GRANT

Синтаксис выдачи прав выглядит следующим образом:

```
GRANT privilege_name ON object_name to {user_name | public |  
role_name};
```

Пример:

```
GRANT ALL ON customer TO iwanow, petrow;  
GRANT UPDATE(fname,lname,company, city),SELECT ON customer TO  
PUBLIC;
```

Операторы доступа к данным

REVOKE

Синтаксис отзыва прав выглядит следующим образом:

```
REVOKE privilege_name ON object_name FROM {user_name | public |  
role_name};
```

Пример:

```
REVOKE ALL ON customer FROM iwanow, petrow;  
REVOKE UPDATE(fname,lname,company, city),SELECT ON customer FROM  
PUBLIC;
```

Операторы доступа к данным

DENY


Синтаксис запрета выглядит следующим образом:

```
DENY privilege_name ON object_name TO {user_name | public |  
role_name};
```

Пример:

```
DENY ALL ON customer TO iwanow, petrow;
```

```
DENY UPDATE(fname,lname,company,city),SELECT ON customer TO PUBLIC;
```



Операторы управления транзакциями

Операторы управления транзакциями

Пример транзакции для PostgreSQL

начало транзакции

BEGIN;

обновляем данные

UPDATE accounts SET balance = balance - 100.00 WHERE name = 'Alice';

ставим точку сохранения

SAVEPOINT my_savepoint;

обновляем данные

UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Bob';

допустили ошибку, возвращаемся к my_savepoint

ROLLBACK TO my_savepoint;

теперь правильно обновляем данные

UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Wally';

завершаем транзакцию

COMMIT;

Операторы управления транзакциями

Пример транзакции для MySQL

начало транзакции

START TRANSACTION;

обновляем данные

UPDATE accounts SET balance = balance - 100.00 WHERE name = 'Alice';

ставим точку сохранения

SAVEPOINT my_savepoint;

обновляем данные

UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Bob';

допустили ошибку, возвращаемся к my_savepoint

ROLLBACK TO my_savepoint;

теперь правильно обновляем данные

UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Wally';

завершаем транзакцию

COMMIT;



Первичные и внешние ключи



Первичные и внешние ключи

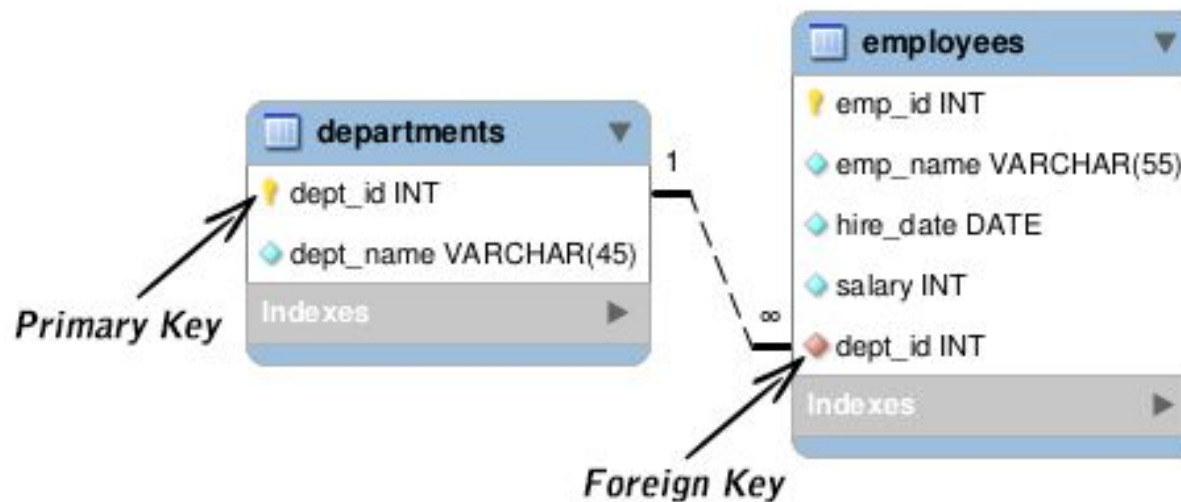
Ключи - это некие сущности, созданные для установления определенных ограничений, которые поддерживают целостность и доступность данных в таблицах баз данных.

Ключи в sql созданы для того, чтобы указать дополнительную функциональность столбца. Будь то уникальность или то, что столбец ссылается на другую таблицу (внешний ключ).

Первичный ключ или PRIMARY KEY означает, что в таблице значение колонки primary key не может повторяться. То есть устанавливает уникальность данных.

Внешний ключ или FOREIGN KEY устанавливает взаимосвязь между данными в разных таблицах.

Первичные и внешние ключи





Сложные выборки данных

Сложные выборки данных

Выборка данных из нескольких таблиц не является тривиальной процедурой.

Типы таких выборок можно разделить на:

- выборка с объединением;
- выборка с использованием вложенного запроса;
- выборка с использованием JOIN.

Сложные выборки данных

Выборку с объединением можно осуществить посредством оператора **UNION**.

Например, у нас есть 2 таблицы:

table 1		
id	name	country
1	John	England
2	Bob	USA

table 2		
id	name	language
3	Alice	Assembler
4	Sindy	C++

Тогда запрос на выборку имен может выглядеть следующим образом:

```
SELECT name FROM table_1 UNION SELECT name FROM table_2;
```

Важно! *UNION* можно применять, только если объединяющиеся выборки совпадают по столбцам.

Сложные выборки данных

Выборка с вложенным запросом производится с использованием оператора **WHERE**.

Например, у нас есть 2 таблицы:

customer		
id	name	order_id
1	John	3
2	Bob	4

orders		
id	title	price
3	Tea	10
4	Phone	9999

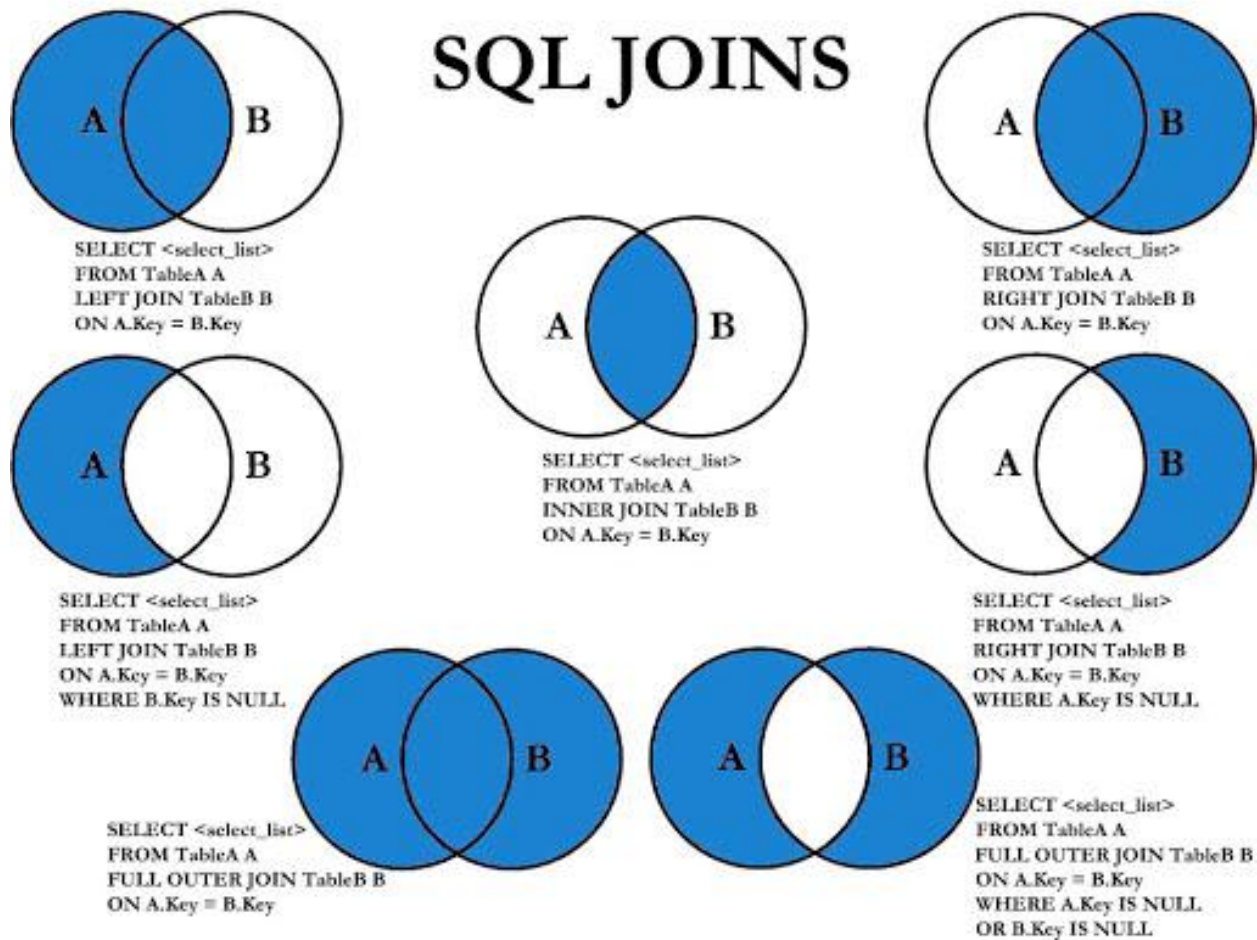
Тогда найти имя покупателя, купившего чай можно следующим образом:
SELECT name FROM customer WHERE order_id IN (SELECT id FROM orders WHERE title="TEA");

Сложные выборки данных

Выборки с использованием **JOIN** можно подразделить на следующие типы:

- Внутреннее присоединение (INNER JOIN)
- Внешнее правое присоединение (RIGHT OUTER JOIN)
- Внешнее левое присоединение (LEFT OUTER JOIN)
- Внешнее присоединение (FULL OUTER JOIN)
- Левое множество, исключая правое
- Правое множество, исключая левое
- Множества, исключая пересечение

Сложные выборки данных



Сложные выборки данных

Пусть у нас есть 2 таблицы (persons, positions):

id	name	post_id
1	Владимир	1
2	Татьяна	2
3	Александр	6
4	Борис	2

id	name
1	Дизайнер
2	Редактор
3	Программист

Сложные выборки данных

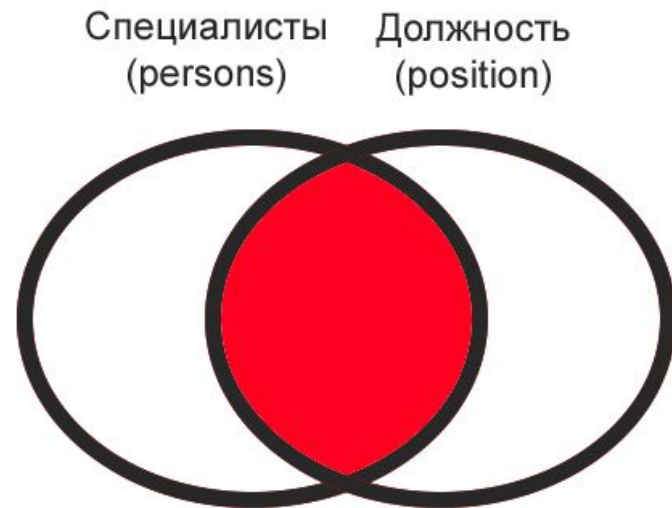
INNER JOIN

```
SELECT p.id, p.name 'Имя сотрудника', ps.id 'pos.id', ps.name 'Должность'
```

```
FROM 'persons' p
```

```
INNER JOIN 'positions' ps ON ps.id = p.post_id
```

id	Имя сотрудника	pos.id	Должность
1	Владимир	1	Дизайнер
2	Татьяна	2	Редактор
4	Борис	2	Редактор



Сложные выборки данных

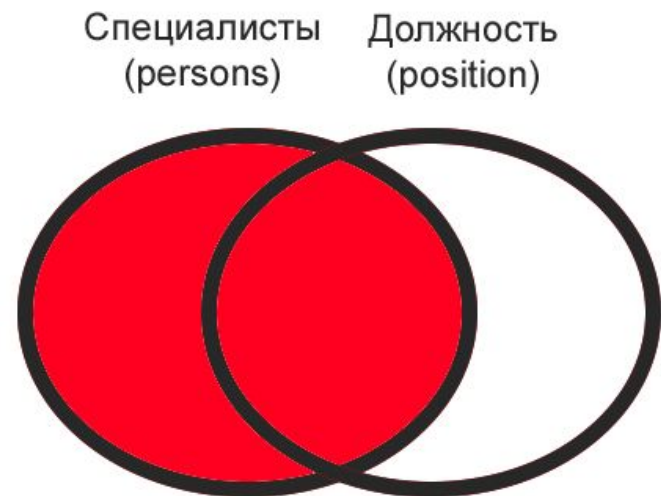
LEFT OUTER JOIN

```
SELECT p.id, p.name 'Имя сотрудника', ps.id 'pos.id', ps.name 'Должность'
```

```
FROM 'persons' p
```

```
LEFT OUTER JOIN 'positions' ps ON ps.id = p.post_id
```

id	Имя сотрудника	pos.id	Должность
1	Владимир	1	Дизайнер
2	Татьяна	2	Редактор
4	Борис	2	Редактор
3	Александр	NULL	NULL

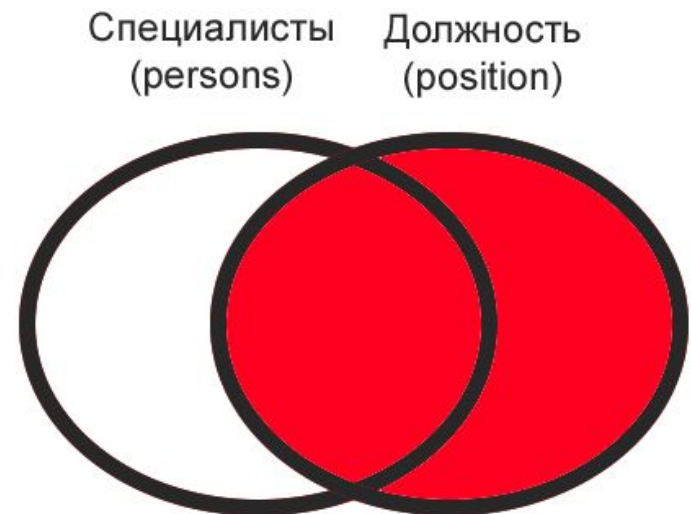


Сложные выборки данных

RIGHT OUTER JOIN

```
SELECT p.id, p.name 'Имя сотрудника', ps.id 'pos.id', ps.name 'Должность'  
FROM `persons` p  
RIGHT OUTER JOIN `positions` ps ON ps.id = p.post_id
```

id	Имя сотрудника	pos.id	Должность
1	Владимир	1	Дизайнер
2	Татьяна	2	Редактор
4	Борис	2	Редактор
NULL	NULL	3	Программист

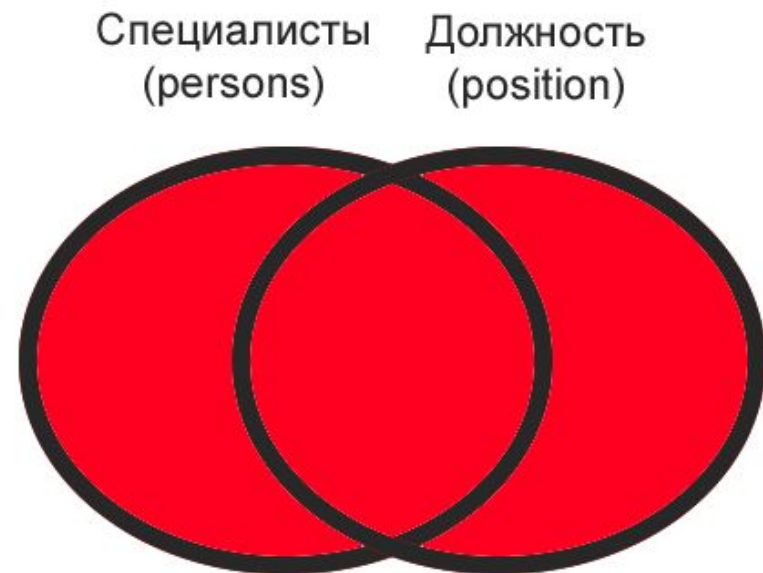


Сложные выборки данных

FULL OUTER JOIN

*SELECT * FROM ps FULL JOIN p ON ps.pos_id = p.id;*

id	Имя сотрудника	pos.id	Должность
1	Владимир	1	Дизайнер
2	Татьяна	2	Редактор
4	Борис	2	Редактор
3	Александр	NULL	NULL
NULL	NULL	3	Программист



Сложные выборки данных

Левое множество, исключая правое

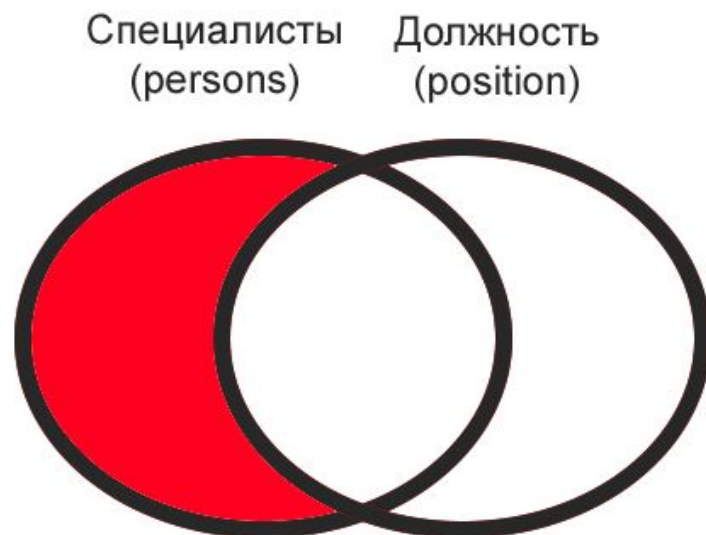
```
SELECT p.id, p.name 'Имя сотрудника', ps.id 'pos.id', ps.name 'Должность'
```

```
FROM `persons` p
```

```
LEFT OUTER JOIN `positions` ps ON ps.id = p.post_id
```

```
WHERE ps.id is NULL
```

id	Имя сотрудника	pos.id	Должность
3	Александр	NULL	NULL



Сложные выборки данных

Правое множество, исключая левое

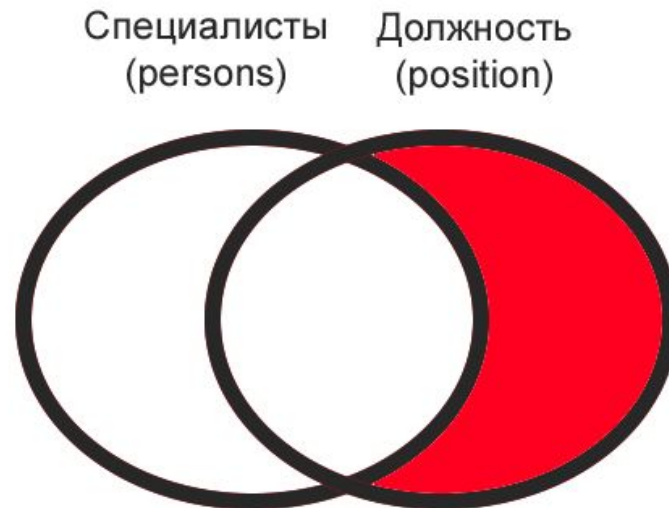
```
SELECT p.id, p.name 'Имя сотрудника', ps.id 'pos.id', ps.name 'Должность'
```

```
FROM `persons` p
```

```
RIGHT OUTER JOIN `positions` ps ON ps.id = p.post_id
```

```
WHERE p.id is NULL
```

id	Имя сотрудника	pos.id	Должность
NULL	NULL	3	Программист

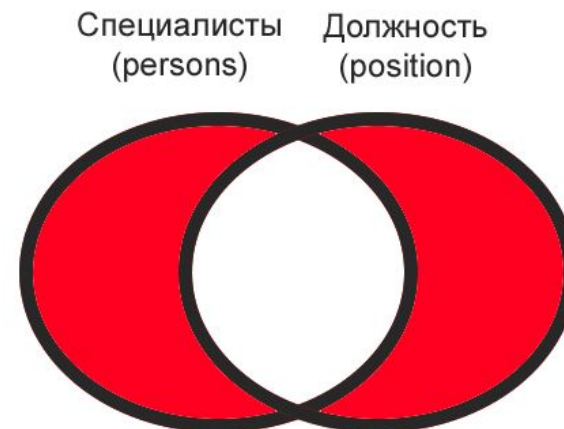


Сложные выборки данных

Множества, исключая пересечение

```
(SELECT p.id, p.name 'Имя сотрудника', ps.id 'pos.id', ps.name 'Должность'  
FROM `persons` p LEFT OUTER JOIN `positions` ps ON ps.id = p.post_id  
WHERE ps.id is NULL) UNION ALL
```

```
(SELECT p.id, p.name 'Имя сотрудника', ps.id 'pos.id', ps.name 'Должность'  
FROM `persons` p RIGHT OUTER JOIN `positions` ps ON ps.id = p.post_id  
WHERE p.id is NULL)
```





Индексы



Индексы

Индексы - это специальные структуры в базах данных, которые позволяют ускорить поиск и сортировку по определенному полю или набору полей в таблице, а также используются для обеспечения уникальности данных.

Количество индексов увеличивает скорость выборок в БД.

При переизбытке количества индексов падает производительность операций изменения данных и увеличивается размер БД.

Индексы

Общие принципы, связанные с созданием индексов:

- Индексы необходимо создавать для столбцов, которые используются в JOIN операциях, по которым часто производится поиск и операции сортировки.
- Для столбцов, на которые наложено ограничение уникальности индекс создается в автоматическом режиме;
- Индексы лучше создавать для тех полей, в которых - минимальное число повторяющихся значений и данные распределены равномерно.
- При внесении изменений в таблицы автоматически изменяются и индексы, наложенные на эту таблицу. В результате индекс может быть сильно фрагментирован, что сказывается на производительности.

Индексы

Индексы можно охарактеризовать следующим образом:

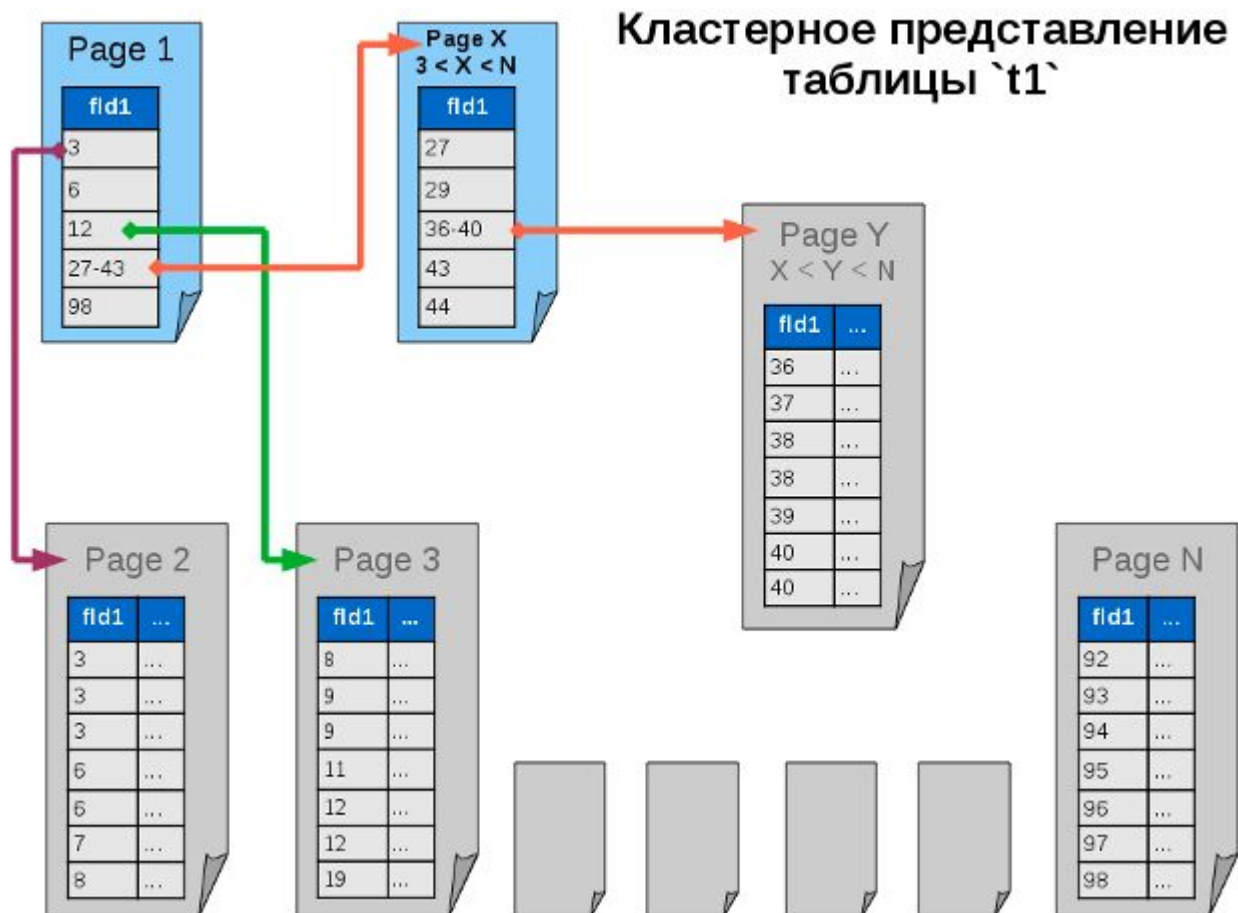
- **Кластерные**

Кластерный индекс представляет из себя древовидную структуру, где такой индекс соответствует набору значений или “смешан” с данными.

- **Некластерные**

Некластерный индекс присваивает каждой записи уникальное значение, позволяющее производить быстрый поиск по таблице.

Индексы





Explain

Explain

Explain - это оператор SQL, предоставляющий полную информацию выполнения запроса.

Синтаксис применения:

EXPLAIN {запрос};

Пример запроса:

*EXPLAIN SELECT * FROM table_name;*

```
***** 1. row *****
id: 1
select_type: SIMPLE
table: categories
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 4
Extra:
1 row in set (0.00 sec)
```

Explain

Описание параметров вывода (пример для MySQL):

Параметр	Назначение
id	порядковый номер для каждого SELECT'а внутри запроса (когда имеется несколько подзапросов)
select_type	тип запроса SELECT
table	таблица, к которой относится выводимая строка
type	тип связи используемых таблиц
possible_keys	индексы, которые могут быть использованы для нахождения строк в таблице
key	использованный индекс
key_len	длина индекса
ref	столбцы или константы, которые сравниваются с индексом, указанным в поле key
rows	число записей, обработанных для получения выходных данных
Extra	содержит дополнительную информацию, относящуюся к плану выполнения запроса



Итоги

Итоги

В данной лекции мы:

- узнали что такое SQL;
- ознакомились с базовыми операторами SQL;
- рассмотрели механизм связи таблиц в БД;
- научились делать сложные выборки;
- изучили механизм индексации данных;
- узнали, как профилировать SQL запросы.

Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Тора Самисько