




ОНЛАЙН-ОБРАЗОВАНИЕ

Онлайн-образование



Меня хорошо видно && слышно?

Ставьте  , если все хорошо
Напишите в чат, если есть проблемы
заодно проверяем, включена ли запись занятия

Включил Юджин запись ли ты



SQL и реляционные СУБД. Введение в PostgreSQL



Аристов Евгений

telegram @AEugene

<https://aristov.tech>

Правила вебинара



Активно участвуем



Задаем вопрос в чат



Вопросы вижу в чате, могу ответить не сразу

Маршрут вебинара



Цели вебинара | После занятия вы сможете

1 объяснить основу реляционной модели данных

2 объяснить назначение языка SQL и его основные конструкции;

3 иметь представление о основных реляционных СУБД

Смысл | Зачем вам это уметь, в результате:

1 говорить на одном языке с разработчиками и архитекторами БД;

2 понимать в каком направлении развивать свои навыки касательно работы со структурированными данными

3 научитесь минимальной работе в GCE

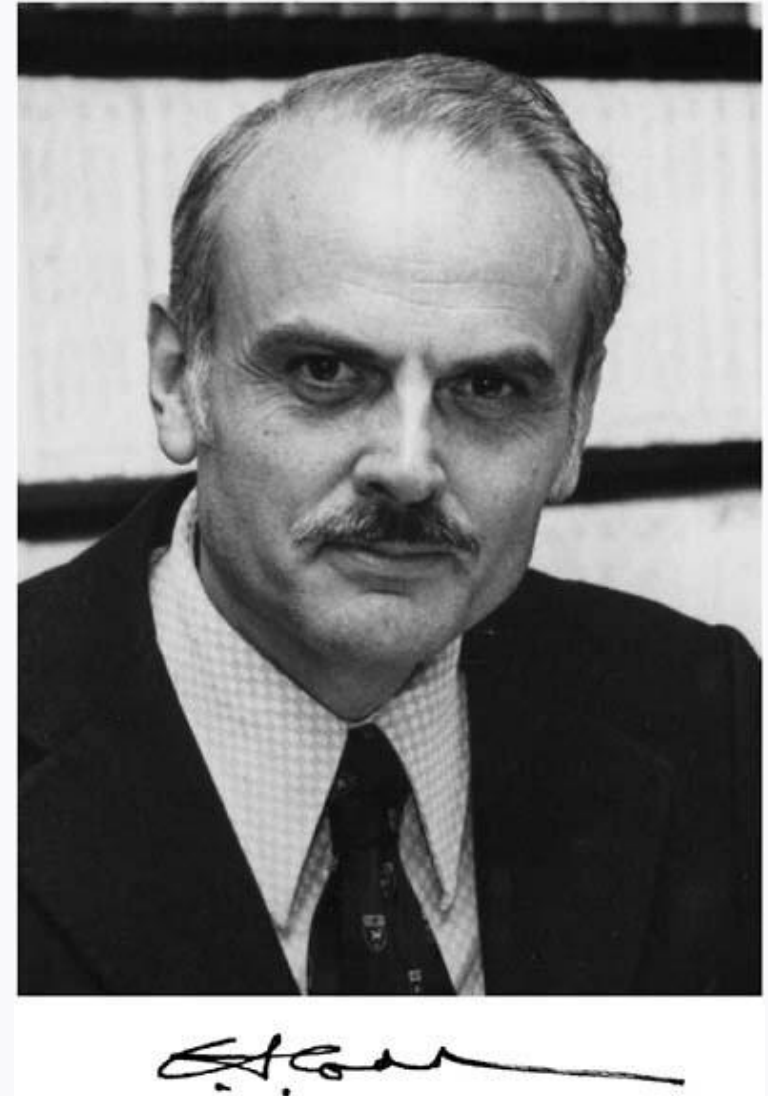
The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this image is a semi-transparent network pattern consisting of numerous small dots connected by thin lines, creating a web-like structure. The text is centered within a horizontal band that transitions from a lighter blue on the left to a darker blue on the right.

Реляционная модель и SQL

Реляционная модель

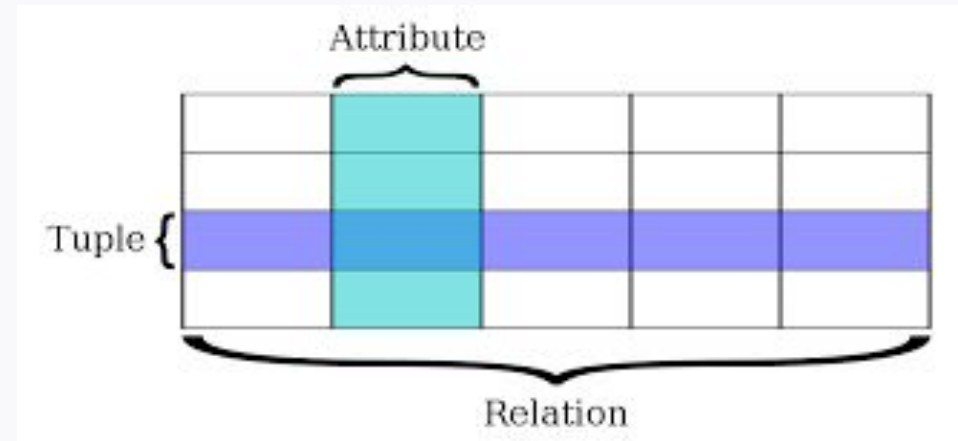
Реляционная модель:

- начало 1970-х
- основоположник Эдгар Франк Кодд (Edgar Frank Codd)
- в рамках программы исследований IBM
- A Relational Model of Data for Large Shared Data Banks
- NULL, view, нормализация, select и т.д.
- позднее предложил «12 правил Кодда»



Реляционная модель

- данные представлены в виде кортежей (tuples)
- объединенных в отношения (relations)
- декларативный способ представления данных и запросов
- пользователь пишет запрос на понятном ему языке
- позволяя СУБД выполнить всю работу по обработке этих запросов



Реляционная модель

- [Нормализация отношений. Шесть нормальных форм / Хабр](#)
- минимизация логической избыточности
- сокращает объем хранимых данных
- увеличивает производительности
- в разумных формах полезна для транзакционных БД
- вредна для аналитических БД

Реляционная модель и SQL

SQL

- так же был разработан в IBM
- в начале 1970-х
- в рамках проекта СУБД IBM System R (потом стал DB2) и на основе работ Кодда
- SEQUEL, Structured English QUery Language
- стал стандартом ANSI в 1986-м и ISO в 1987-м
- последний на данный момент времени стандарт SQL:2016, в 2019 внесена корректировка, пересматривается каждые 5 лет

стандарт SQL выделяет 5 видов операций

DML - modification

- INSERT
- UPDATE
- DELETE

DDL - definition

- ALTER
- CREATE
- DROP

DCL - control

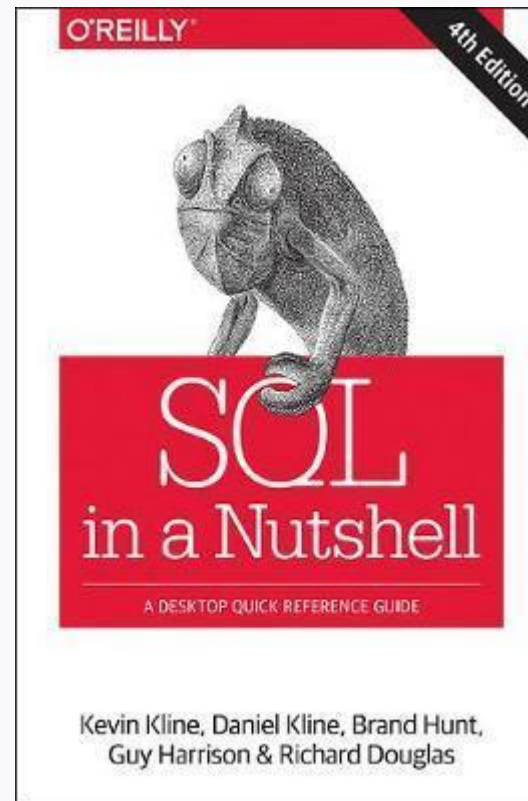
- GRANT
- REVOKE

DQL - query

- SELECT

TCL - transaction control

- BEGIN
- COMMIT
- ROLLBACK



реляционная модель и SQL

SQL constraints

- правила описывающие корректность данных
- работают на уровне DML
- любимы DBA
- не любимы разработчиками
- давайте им имена уж если вы их используете
- ANSI:
 - PRIMARY KEY
 - FOREIGN KEY
 - UNIQUE
 - CHECK

The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band that contains a white network or mesh pattern of interconnected lines and dots. Centered within this band is the word "Вопросы?" in a large, white, sans-serif font.

Вопросы?

The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this is a semi-transparent network pattern consisting of numerous small dots connected by thin lines, creating a web-like effect. The text is centered within a horizontal band that transitions from a lighter blue on the left to a darker blue on the right.

OLTP, ACID, MVCC, ARIES

Множественная параллельная нагрузка

- реляционная теория и SQL позволяет абстрагироваться от конкретной реализации СУБД
- но есть одна непростая проблема
- как обеспечить **параллельную** работу множества сессий (**concurrency**)
- которые **модифицируют** данные
- так чтобы они **не мешали** друг другу
- ни с точки зрения **чтения** ни с точки зрения **записи**
- и обеспечивали **целостность** данных - т.н. **consistency**
- и их **надежность** - т.н. **durability**

ACID

ответ - транзакционные системы, OLTP - **O**nline **T**ransaction **P**rocessing

ACID

- **Atomicity** — Атомарность
- **Consistency** — Согласованность
- **Isolation** — Изолированность
- **Durability** — Долговечность

транзакция (**transaction**)

- называется множество операций, выполняемое приложением
- которое переводит базу данных из одного корректного состояния в другое корректное состояние (согласованность)
- при условии, что транзакция выполнена полностью (атомарность)
- и без помех со стороны других транзакций (изолированность)

ACID - хорошо, а как блокировки?

- **ARIES** - **A**lgorithms for **R**ecovery and **I**solation **E**xploiting **S**emantics
 - logging
 - undo
 - redo
 - checkpoints
- **MVCC** - **M**ultiversion **C**oncurrency **C**ontrol
 - copy-on-write
 - каждый пользователь работает со снимком БД
 - вносимые пользователем изменения не видны другим до фиксации транзакции

The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band across the middle, which contains a white network pattern of dots and lines. The word "Вопросы?" is centered in this band in a large, white, sans-serif font.

Вопросы?

The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band that contains a white network diagram. This diagram consists of numerous small dots connected by thin white lines, creating a complex web-like structure that suggests connectivity and data flow. The text 'Современные РСУБД' is centered within this band in a large, white, sans-serif font.

Современные РСУБД

Современные РСУБД

- поддержка SQL:2008 как минимум
- ACID: ARIES/MVCC
- поддержка Linux
- наличие облачных сервисов (cloud managed services)
- работа в Docker и Kubernetes

современные РСУБД

	Oracle Database	SQL Server	MySQL	PostgreSQL	MariaDB
Тип	закрытая	закрытая	открытая	открытая	открытая
Год основания	1979	1989	1995	1989	2009
Текущая версия	21c	2019	8.0	15	10
ANSI/ISO SQL	SQL:2016	SQL:2016	SQL:2016	SQL:2016	SQL:2016
Процедурное расширение	PL/SQL	T-SQL	SQL/PSM	PL/pgSQL, PL/Tcl, PL/Perl, PL/Python	SQL
ACID	ARIES/MVCC	ARIES, MVCC-ready	ARIES/MVCC	ARIES/MVCC	ARIES/MVCC
Linux	Oracle, RedHat, SuSE	RedHat, SuSE, Ubuntu	Any	Any	Any
Cloud	AWS, Oracle	AWS, Azure, GCP	AWS, Azure, GCP, Oracle	AWS, Azure, GCP	AWS
Docker/Kubernetes	cloud	+	+	+	+
Цена	космос	чуть меньше	free	free	free



Введение в PostgreSQL. Практика

введение в PostgreSQL

- автор: Майк Стоунбрейкер, Бёркли
- начало 1970-х: Ingres — INteractive Grafic REtrieval System
- середина 1980-х: Postgres - Post Ingres
- Postgres95 - добавлен SQL
- 1996: postgresql.org
- сейчас развивается PostgreSQL Global Development Group
- компании контрибьюторы
 - 2ndQuadrant
 - EnterpriseDB
 - Crunchy Data
 - Postgres Professional

release notes PostgreSQL

<https://www.postgresql.org/docs/release/10.0/>

[Встречаем PostgreSQL 10. Перевод Release Notes / Блог компании Авито / Хабр](#)

<https://www.postgresql.org/docs/release/11.0/>

[PostgreSQL 11 Released!](#)

<https://www.postgresql.org/docs/release/12.0/>

[PostgreSQL 12 Released!](#)

<https://www.postgresql.org/docs/release/13.0/>

[PostgreSQL 13 Released!](#)

<https://info.crunchydata.com/blog/why-postgresql-13-is-a-lucky-release>

<https://pgpedia.info/postgresql-versions/postgresql-14.html>

<https://www.postgresql.org/docs/release/15.0/>

PostgreSQL Extensions

<https://postgrespro.ru/docs/postgresql/14/contrib>

PostgreSQL Pro

<https://postgrespro.ru/docs/enterprise/14/intro-pgpro-vs-pg>

<https://postgrespro.com/products/postgrespro/enterprise>

<https://postgrespro.ru/docs/enterprise/14/multimaster>

Документация PostgreSQL

<https://www.postgresql.org/files/documentation/pdf/14/postgresql-14-A4.pdf>

кто угадает сколько страниц?)

Практика

Развернем VM



МИНИТЕСТ

<https://forms.gle/xXKGKVdHWjSPx3ks5>

2-3 минуты



Уровни изоляции транзакций

Уровни изоляции транзакций

Стандарт SQL допускает четыре уровня изоляции, которые определяются в терминах аномалий, которые допускаются при конкурентном выполнении транзакций на этом уровне:

- «Грязное» чтение (dirty read). Транзакция T1 может читать строки измененные, но еще не зафиксированные, транзакцией T2. Отмена изменений (ROLLBACK) в T2 приведет к тому, что T1 прочитает данные, которых никогда не существовало.
- Неповторяющееся чтение (non-repeatable read). После того, как транзакция T1 прочитала строку, транзакция T2 изменила или удалила эту строку и зафиксировала изменения (COMMIT). При повторном чтении этой же строки транзакция T1 видит, что строка изменена или удалена.
- Фантомное чтение (phantom read). Транзакция T1 прочитала набор строк по некоторому условию. Затем транзакция T2 добавила строки, также удовлетворяющие этому условию. Если транзакция T1 повторит запрос, она получит другую выборку строк.
- Аномалия сериализации - Результат успешной фиксации группы транзакций оказывается несогласованным при всевозможных вариантах исполнения этих транзакций по очереди.

Уровни изоляции транзакций

	«грязное» чтение	неповторяю- щееся чтение	фантомное чтение	аномалия сериализации
Read Uncommitted	Допускается, но не в PG	да	да	да
Read Committed	-	да	да	да
Repeatable Read	-	-	Допускается, но не в PG	да
Serializable	-	-	-	-

на всех уровнях не допускается потеря зафиксированных изменений

<https://habr.com/ru/company/otus/blog/501294/>

<https://postgrespro.ru/docs/postgrespro/13/transaction-iso>

Уровни изоляции транзакций

Аномалия сериализации

class | value

-----+-----

1 | 10

1 | 20

2 | 100

2 | 200

сериализуемая транзакция А вычисляет:

SELECT SUM(value) FROM mytab WHERE class = 1; добавляем запись с суммой и class 2

сериализуемая транзакция В вычисляет:

SELECT SUM(value) FROM mytab WHERE class = 2; добавляем запись с суммой и class 1

что произойдет при попытке фиксации изменений?

Уровни изоляции транзакций

ОШИБКА: не удалось сериализовать доступ из-за зависимостей чтения/записи между транзакциями

причем если бы одна из транзакций была бы repeatable read, все бы успешно закоммитилось

Уровни изоляции транзакций

Практика

Вложенные транзакции

реализация SAVEPOINT

каталог \$PGDATA/pg_subtrans/

<https://postgrespro.ru/docs/postgresql/14/sql-savepoint>

The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band that contains a white network or mesh pattern of interconnected lines and dots. Centered within this band is the word "Вопросы?" in a large, white, sans-serif font.


Вопросы?

The image features a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City, with numerous skyscrapers and buildings. A semi-transparent blue band with a white geometric network pattern of dots and lines runs horizontally across the center of the image. The Russian word "Порефлексируем" is written in white, bold, sans-serif font across this band.

Порефлексируем

Вопросы?

- Кто что запомнил за сегодня?
- Что узнали нового?



ДЗ

ДЗ

ДЗ сдаем в виде миниотчета на github в формате markdown

- создать новый проект в Google Cloud Platform, например postgres2022-, где уууумм год, месяц вашего рождения (имя проекта должно быть уникально на уровне GCP), **Яндекс облако или на любых ВМ, докере**
- далее создать инстанс виртуальной машины с дефолтными параметрами ОС **Ubuntu**
- добавить свой ssh ключ в GCE metadata
- зайти удаленным ssh (первая сессия), не забывайте про ssh-add
- поставить PostgreSQL
- зайти вторым ssh (вторая сессия)
- запустить везде psql из под пользователя postgres
- выключить auto commit
- сделать в первой сессии новую таблицу и наполнить ее данными

```
create table persons(id serial, first_name text, second_name text);
insert into persons(first_name, second_name) values('ivan', 'ivanov');
insert into persons(first_name, second_name) values('petr', 'petrov');
commit;
```
- посмотреть текущий уровень изоляции: show transaction isolation level
- начать новую транзакцию в обеих сессиях с дефолтным (не меняя) уровнем изоляции
- в первой сессии добавить новую запись

```
insert into persons(first_name, second_name) values('sergey', 'sergeev');
```


ДЗ

- сделать `select * from persons` во второй сессии
 - видите ли вы новую запись и если да то почему?
 - завершить первую транзакцию - `commit`;
 - сделать `select * from persons` во второй сессии
 - видите ли вы новую запись и если да то почему?
 - завершите транзакцию во второй сессии
 - начать новые но уже `repeatable read` транзакции - `set transaction isolation level repeatable read`;
 - в первой сессии добавить новую запись
`insert into persons(first_name, second_name) values('sveta', 'svetova');`
 - сделать `select * from persons` во второй сессии
 - видите ли вы новую запись и если да то почему?
 - завершить первую транзакцию - `commit`;
 - сделать `select * from persons` во второй сессии
 - видите ли вы новую запись и если да то почему?
 - завершить вторую транзакцию
 - сделать `select * from persons` во второй сессии
 - видите ли вы новую запись и если да то почему?
- `\echo :AUTOCOMMIT`
`\set AUTOCOMMIT OFF`


ДЗ

Выполнение ДЗ: 10 баллов
+ 2 балла за красивое решение

- 2 балла за рабочее решение, и недостатки указанные преподавателем не устранены



Заполните, пожалуйста,
опрос о занятии по ссылке в чате
<https://otus.ru/polls/51725/>



Спасибо за внимание!
Приходите на следующие вебинары

Аристов Евгений