БАЗЫ ДАННЫХ

Лекция 1

ПРЕДМЕТ ИЗУЧЕНИЯ

- Системы управления базами данных (СУБД)
- Начинаем с SQL
- На примере Postgres, где-то - MySQL
- Постараемся без ликбеза
- Взаимодействие с приложением

ПРЕДМЕТ ИЗУЧЕНИЯ

- Ближе к концу - NoSQL
- Документные базы (MongoDB)
- Key-Value Storage (Redis, Hazelcast)
- Колоночные базы (Cassandra)
- Вертикальные базы (ClickHouse)
- Если успеем - поисковые, графовые, GIS

ПРАВИЛА КУРСА

- "Внутренняя" оценка - 100 балльная шкала
- (50, 70] - 3 в 5-бальной
- (70, 85] - 4 в 5-бальной
- (85, Inf] - 5 в 5-бальной

СОСТАВНЫЕ ЧАСТИ ОЦЕНКИ

СОСТАВНЫЕ ЧАСТИ ОЦЕНКИ

- 5-6 домашек
- Каждая оценивается на 100 баллов
- На балл влияют: полнота решений, качество (по ревью), соблюдение дедлайнов
- Считаем среднюю по домашкам, отбросив худшую
- В итоговую оценку идет с коэффициентом 0.45

СОСТАВНЫЕ ЧАСТИ ОЦЕНКИ

- Почти после каждого семинара - набор автоматизированных тестов
- Для NoSQL-части - возможна замена на опросы
- Каждый набор тестов или опросник оценивается в 100 баллов
- Считаем среднюю, отбросив 1-2 худших
- В итоговую оценку идет с коэффициентом 0.20

СОСТАВНЫЕ ЧАСТИ ОЦЕНКИ

- Экзамен - как беседа
- Что-то спрошу по домашкам, попрошу написать запрос
- Что-то спрошу на понимание
- Пять вопросов, по 20 баллов каждый
- В итоговую оценку идет с коэффициентом 0.35

МОЖНО БЕЗ ЭКЗАМЕНА

- Первый способ - получить высокие баллы за домашки
- Тогда 0 за экзамен не помешает зачету
- Но балл будет 3
- Второй способ - получить право на автомат
- Выдается за высокую (и разумную) активность на семинаре
- На фоне высоких показателей по домашкам и тестам

ПОДРОБНЕЕ ПРО ТЕСТЫ

- Короткий дедлайн
- Разбор особо интересных задач на следующем семинаре
- В общем случае запоздалые доделки не предусмотрены
- Если пропускается по уважительной причине, то возможно новое задание
- Новое задание выдается в конце семестра

ПОДРОБНЕЕ ПРО ДОМАШКИ

- Дедлайн никто не меряет на секунды
- По мере пропущенных дней набегает понижающий коэффициент
- 1 день - 0.95
- 2 дня - 0.9
- 14 дней 0.3
- Понижается в день на 0.05

ПОДРОБНЕЕ ПРО ДОМАШКИ

- Через 14 дней фиксируется
- Остается 0.3 навсегда
- В случае ухода на пересдачу - не восстанавливается !!!
- Мораль: не забрасывайте домашки !!!

ПОДРОБНЕЕ ПРО ДОМАШКИ

ПОДРОБНЕЕ ПРО ДОМАШКИ

- Если тесты пройдены или не предусмотрены, то начинается code review
- По итогам review балл может быть понижен
- Как правило, замечания можно исправлять
- Как правило, со скидой
- Как правило, одна итерация на исправление

ПОДРОБНЕЕ ПРО ДОМАШКИ

- Возможны бонусы
- За активность на семинаре, доделки заданий с семинаров (будет объявляться)
- За особо красивые решения в тестах
- Бонусные баллы добавляются к релевантным домашкам, но не выше 100 баллов
- Переноса на другие домашки нет

А ЗАЧЕМ ОНИ ВООБЩЕ НУЖНЫ ?

- Есть классические алгоритмы и структуры данных
- Можно записать и прочитать файлы
- Как будто можно обойтись этим
- Зачем умножать сущности ?

КЛЮЧЕВЫЕ ПРОБЛЕМЫ

- Эффективный доступ
- При том, что объем данных превышает возможности памяти
- Но есть более популярные данные и менее популярные
- И особенности дисковой памяти
- Популярные надо кешировать

КЛЮЧЕВЫЕ ПРОБЛЕМЫ

- Надежное обновление
- Питание может отключиться
- Хочется избегать наполовину внесенных изменений
- Конфликт между кеширование и обновлением

КЛЮЧЕВЫЕ ПРОБЛЕМЫ

- Параллельная обработка
- Хотим делать целостные изменения
- Чтобы другие запросы не видели промежуточного состояния
- И откатываться назад, если передумали

СУБД: КРАТКАЯ ИСТОРИЯ

- Появляются как решение обозначенных проблем
- Возникает доминирующая модель - реляционная
- И стандарт языка запросов
- И сопутствующая теория проектирования
- Несколько вендоров закрытых СУБД

POSTGRES

- На рубеже 80-90 появляется Postgres - университетская СУБД
- Много амбизиозных идей, например - объектно-оринтированность
- Появляются пользователи, необходимость поддержки
- Кодовую базу отдают сообществу
- Open-Source проект без жесткого управления
- Есть компании на поддержке (PostgresPro)

MYSQL

- Изначально - шведский стартап
- OpenSource + support-контракты
- Компанию покупает Sun Microsystems
- Которую потом покупает Oracle
- OpenSource под корпоративным управлением

SQLITE

- Легкая реляционная СУБД
- Реализована как библиотека
- Удобна для использования внутри приложения
- Удобна для изучения

ДРУГИЕ ИГРОКИ

- Oracle DB
- MS SQL
- Ретро: MS Access, Ingres
- Игрушечные: Derby, H2

NOSQL

- В нулевых годах начинается NoSQL-движение
- Часто понимается как NotOnly-SQL
- СУДБ фокусируются на каком-то аспекте реляционной модели
- Жертвуя какими-то другими

РЕЛЯЦИОННАЯ МОДЕЛЬ

- В основе - математическая абстракция кортежа
- Набор полей (элементов)
- У каждого поля - свой допустимый набор значений
- Тпп кортежа - упорядоченный набор типов полей

РЕЛЯЦИОННАЯ МОДЕЛЬ

- Набор кортежей одного типа составляет таблицу (отношение)
- Пример кортежа: таблица данных о транспортных картах
- Поля: id карты, ФИО владедьца (если привязана к льготам и т.п.), сумма на счете
- Другой пример: данные о поездках по карте
- Поля: id карты, время оплаты, данные об оплате (возможны варианты)

РЕЛЯЦИОННАЯ МОДЕЛЬ

- Не задается вопросами создания и заполнения отношений
- Определяет операции с готовыми отношениями
- Переименование полей
- Объединение, пересечение, вычитание
- Декартово произведение
- Выборка, проекции

SQL: ОСНОВНЫЕ ОПЕРАЦИИ

- CRUD: Create, Request, Update, Delete
- Create - это про записи
- Есть еще создание таблиц - но это уже "метаоперация"
- Создание простейшей таблицы: CREATE TABLE cards (id INT, name VARCHAR(20), balance INT);

SQL: ОСНОВНЫЕ ОПЕРАЦИИ

- Традиция - писать ключевые слова заглавными, имена - строчными
- У таблица есть имя и перечень полей
- У каждого поля есть тип
- В нашем случае - два целочисленных поля и одно строковое

SQL: ОСНОВНЫЕ ТИПЫ

- CHAR(10), VARCHAR(200), TEXT - строковые типы
- CHAR(10) - выделяем на поле фиксированное количество символов
- Если значение короче, то заполняем свободные позиции нулевыми байтами
- Пример - код страны

SQL: ОСНОВНЫЕ ТИПЫ

- VARCHAR(200) - храним фактическое количество символов
- Гарантируем лимит на длину
- Пример - твит
- TEXT - храним фактическое количество символов, без каких-либо гарантий

ЗАПРОСЫ НА ДОБАВЛЕНИЕ ДАННЫХ

- Основной вариант - INSERT
- Изготовили карточку, хотим добавить ее в базу
- INSERT INTO cards VALUES (1, 'Vasily Petrov', 0);
- Можно и так: INSERT INTO cards VALUES (1, 'Vasily Petrov');
- Но эффект будет другой

НЕКОТОРЫЕ НЮАНСЫ

- Значением по уиолчанию будет NULL
- Это не то же, что и 0
- Это означает "неопределенное значение"
- Может приводить к неожиданным результатам в момент SELECT

НЕКОТОРЫЕ НЮАНСЫ

- Можем запретить NULL-значения
- CREATE TABLE cards (id INT, name VARCHAR(20), balance INT NOT NULL)
- Тогда "неполный" INSERT сломается
- Можем указать значение по умолчанию
- CREATE TABLE cards (id INT, name VARCHAR(20), balance INT NOT NULL DEFAULT 0)

ПОРОЖДЕНИЕ УНИКАЛЬНОГО ЗНАЧЕНИЯ

- Получили такое требование
- Каждому пользователю хотим назначать уникальный ID
- Чтобы он был номером карты
- Нужно знать состояние таблицы для создания новой записи

ПОРОЖДЕНИЕ УНИКАЛЬНОГО ЗНАЧЕНИЯ

- Вариант 1: хранить "состояние" в приложении
- Возможны гонки
- Вариант 2: делать SELECT + INSERT в транзакции
- Дорого

ПОРОЖДЕНИЕ УНИКАЛЬНОГО ЗНАЧЕНИЯ

- СУБД предлагают встронный механизм
- В Postgres - тип SERIAL
- В MySQL - атрибут AUTO_INCREMENT
- CREATE TABLE cards (id SERIAL, name VARCHAR(20), balance INT NOT NULL DEFAULT 0)

ПОРОЖДЕНИЕ УНИКАЛЬНОГО ЗНАЧЕНИЯ

- Есть модификация INSERT, позволяющая указывать подмножество полей
- INSERT INTO cards (name, balance) VALUES ('Ivan Sidorov', 100);
- Кстати, никто не запрещает явно указать значение типа SERIAL
- Последствия - на совести того, кто так делает
- Это не влияет на счетчик

УДАЛЕНИЕ ДАННЫХ

- Самое радикальное - удалить базу: DROP DATABASE name;
- Можно удалить таблицу: DROP TABLE cards;
- Можно удалить все записи в таблице: DELETE FROM cards;
- Можно удалить по выбору: DELETE FROM cards WHERE balance < 0;

ОГРАНИЧЕНИЯ НА ПОЛЯ

- Есть сложная система ограничений на поля
- Есть особые ограничения - на них выделены отденнльные ключеные слова (NOT NULL)
- Еще одно - UNIQUE
- Поля с таким ограничителем обязаны быть уникальными

ОГРАНИЧЕНИЯ НА ПОЛЯ

- Пример: записи о людях
- Может быть поле "инн"
- На нем может быть ограничение UNIQUE, но не быть ограничения NOT NULL
- Это может быть база сотрудников
- Человека добавили, а ИНН он еще не принес

ОГРАНИЧЕНИЯ НА ПОЛЯ

- Часто эти два ограничения идут вместе
- Мы создаем поле специально, чтобы через него ссылаться на запись
- Пример: создание аккаунта в соцсети
- Должны дать уникальный идентификатор
- Без идентификатора запись не имеет смысла

ОГРАНИЧЕНИЯ НА ПОЛЯ

- Бывают более сложные ограничения
- Например, мы не ожидаем, что карта работает в долг
- Можем указать ограничения на диапазон значений
- CREATE TABLE cards (id SERIAL, name VARCHAR(20), balance INT NOT NULL DEFAULT 0 (CHECK balance >= 0))
- Двойная польза: не ошибетесь и поможете в оптимизации

ПОСМОТРИМ НА ТАБЛИЦУ

- \d+ cards
- Увидим поля, их типа, UNIQ/NOT NULL
- Отдельно увидим произвольные ограничения
- Они именованы

ИЗМЕНЕНИЕ ТАБЛИЦ

- Созданную таблицу можно поменять
- Есть группа команд ALTER TABLE
- В частности - можно убрать ограничения
- И тогда сошлемся на них по имени
- Подробности - позже

ВАРИАНТЫ ЗАДАНИЯ ОГРАНИЧЕНИЙ

- Можно после поля задать анонимное ограничение
- Postgres придумает ему имя
- Можно в конце определения таблицы задать именованное ограничение
- Через ключевое слово CONSTRAINT