Настройка приложений баз данных

Б.А. Новиков
Санкт-петербургский университет
@Business OY

Функциональность и производительность

- «Если производительность недостаточна, подождите год» - ???
  - в университетах – да
  - в реальности - НЕТ
- «Performance is not an issue» - ???
  - в software engineering – да
  - в реальности - НЕТ
- «Вследствие неприемлемого времени ответа скапливаются большие очереди»

@Business OY, Finland

- Изготовление информационных систем среднего размера на заказ
- СУБД: Oracle, MS SQL, IBM DB2, Solid и др.
- Web-клиент, отчеты
- Миграция и интеграция
- Десятки проектов в год

Многие примеры взяты из практики других компаний

Проекты и базы данных

- Разнообразные отрасли народного хозяйства Европы
- Трудоемкость проектов от 3 недель до нескольких лет
- СУБД: высокопроизводительные и недорогие
- Базы данных
  - Объем от единиц до сотен Гб
  - От 100 до 500 таблиц
  - От 300 до 3000 SQL запросов
  - OLTP, reports, OLAP

Настройка

Координатор проекта:

«Заказчики жалуются на медленность системы. Посмотрите, пожалуйста, какие нужно добавить индексы»

Документация по СУБД:

«система, не подготовленная во время разработки, малоэффективна»

Настройка

- Настройкой называется совокупность мер, необходимых для приведения системы в соответствие с требованиями по производительности без изменения функциональности
- Меры по настройке необходимы на всех фазах создания системы, начиная со спецификации требований

Книги по настройке

- Dennis Shasha and Philippe Bonnet: *Database tuning: principles, experiments and troubleshooting techniques.* Morgan-Kaufmann, San Francisco, CA, 2002.
- D. Tow. *{SQL} tuning.* O'Reily, Sebastopol, CA, 2003.

Ограниченность ресурсов

- 2-недельные программисты не предназначены для получения высокой производительности
- Стоимость и время проектов
- Объектные технологии, каркасы, ...
- Управление качеством

Пессимизация на ранних фазах

- Выбор конфигурации на основе рекламных материалов производителей
- Априорные архитектурные решения
- Опережающее проектирование БД
- Эмоциональный выбор программных средств

Проектирование

- Объектная модель и модель БД
- Гибкая структура БД: динамическая схема
- Суррогаты (t_time)
- Нормализация, денормализация, материализация

Тернарная схема БД

```
Create table ObjAttrValue(
  object_id,
  Attr_name,
  Attr_value);
```

Изобреталась и публиковалась многократно

Фатально на сколько-нибудь больших объемах данных

Иногда оптимальна

Настраиваемые записи

```
Create table custom_obj_type (
  obj_id,
  Obj_type_id,
  num_attr1,
  num_attr2,
  …
  string_attr_1,
…);
```

Плохо работает на больших объемах

Пессимизация с помощью суррогатов

Create table mea1(ts datetime, val,cust_id,unit)

Improvement:

```
Create table mea3(ts_id, val,cust_id,unit_id,
   cts, uts, cui, uui, dts, dui…);
Create table t_time(ts_id, ts, ts_properties…)
Create table t_unit( unit_id, …);
```

Дополнительные операции соединения

Потеря кластеризации

Неиспользуемые атрибуты

2Гб => 25 Гб

Нормализация и «денормализация»

- Миф о низкой эффективности нормализованных схем
- Смешивание логического проектирования и проектирования схемы хранения
- Выбор структуры хранения?
  - Кластеризация
  - Материализация

Создание кода приложения

- Java как инструмент пессимизации
- Каркасы
- Объектно-реляционные отображения
- Cursors are death
- Правила гранулярности

Объектно-ориентированное программирование и оптимизация

- Манифест 3-го поколения БД (1990):

- «Программисты должны принять тот факт, что оптимизатор выполнит запрос лучше, чем они»

Объектное программирование может полностью нейтрализовать все возможности оптимизатора запросов

Паническая боязнь SQL

- Декларативный силь труден для массового программирования
- Учебники О-О рекомендуют использование только примитивных запросов
- Функции СУБД дублируются в коде приложения
- Ущербные объектные оболочки для SQL
- Каркасы

Универсальный интерфейс БД

- Реализуется парой функций для извлечения и записи скалярных значений

```
Get_value(db, table, column, id)
Set_value(db, table, column, id, value)
```

- Разрабатывался во многих организациях

Замкнутость объектных систем

- Программисты не закрывают курсоры, рассчитывая на сборку мусора
- СУБД закрывает курсоры при завершении сеанса
- Сеансы никогда не закрываются, а используются для других приложений
- Ошибка присутствует в большинстве приложений, написанных на java

Cursors are Death

- (Глава из книги D. Shasha по настройке БД)
- Создание курсора включает несколько сетевых обменов с ожиданием ответа
- Передача результатов выполнения курсора выполняется асинхронно
- Объектные программы обычно используют большое количество мелких запросов
- 16000 запросов для построения формы, содержащей около 100 полей – 6 минут

Правила гранулярности запросов

- Количество запросов не должно быть пропорционально числу получаемых строк
- Если из таблицы читаются данные, то все необходимые атрибуты должны быть получены в том же запросе
- Условия селекции должны включаться в запрос, а не проверяться в коде приложения

Настройка сервера БД

- Единственная общепризнанная задача администратора БД
- Практически не требуется для недорогих СУБД, так как отсутствуют альтернативы
- Высокопроизводительные системы обычно обладают автоматическими средствами конфигурации и вмешательство требуется только в редких случаях

Автоматическая настройка SQL

- Исследования и реклама
- В большинстве случаев оптимизаторы высокопроизводительных СУБД вырабатывают высококачественные планы
- Мощные средства анализа и диагностики
- Адаптивные методы оптимизации
- И тем не менее в трудных случаях: «consider SQL re-writing»

Мифы, связанные с SQL

- Не следует использовать более 5 операций соединения в запросе (может быть оправданно только в недорогих СУБД)
- Представления (view) материализуются при выполнении запросов полностью
- Операции соединения выполняются вложенными циклами (только в недорогих СУБД)

Настройка SQL: стратегия

- Получение оценок снизу для времени выполнения запроса и сопоставление с требованиями
- Анализ плана и выявление операций, вызывающих неудовлетворительную производительность
- Переписывание запроса
- Построение индексов и материализаций
- Использование подсказок оптимизатору

Короткие запросы

- Результат зависит от небольшой доли строк больших таблиц
- Обычно необходимы индексы
- Обычно наиболее эффективны соединения row NL
- Группировка и сортировка практически не вляют на время выполнения

Эквивалентные преобразования SQL

- Компенсируют слабость оптимизатора
- UNION вместо OR
- NOT EXISTS вместо MAX или MIN
- Вложенные подзапросы с агрегированием
- Транзитивность равенства в соединениях с общим ключом:

```
t1.a = t2.a = t3.a
```

Неэквивалентные преобразования

- Оптимизатор не имеет права на подобные трансформации
- Устранение избыточных соединений (например, вложенных подзапросов)
- Введение избыточных условий на основе семантики приложения
- Использование избыточности схемы (перенос условий селекции)

Большие запросы

- Полный просмотр необходим для получения результата
- Как правило, индексы бесполезны
- Обычно хорошо работает hash join
- Основной прием настройки – исключение многократных просмотров больших таблиц

Каскадное агрегирование

- Обычно оптимизаторы выполняют агрегирование после всех операций соединения
- Раннее выполнение частичного агрегирования может существенно сократить время выполнения запроса

Другие аспекты настройки

- Транзакции
- Параллелизм и очень большие базы данных
- XML как средство пессимизации

Вместо заключения

Изобретательность программистов предела не имеет, поэтому задачи настройки возникают почти в каждом проекте.