Разработка приложений БД

Графеева Н.Г.

2014

Основные принципы

- успех или неудача разработки приложения базы данных определяется тем, как оно использует базу данных;
- в команде разработчиков должно быть ядро "программистов базы данных", обеспечивающих согласованность логики работы с базой данных и настройку производительности системы.

Для успешного создания приложения необходимо понимать

- архитектуру конкретной СУБД, ее компоненты и алгоритмы работы;
- особенности управления одновременным доступом в конкретной СУБД;
- глубокое знание реализации SQL в конкретной СУБД.
- как настраивать приложение с момента введения его в эксплуатацию;
- как реализованы определенные компоненты СУБД и чем эта реализация отличается от обычно предполагаемой;
- какие возможности реализованы в самой СУБД и почему, как правило, лучше использовать предоставляемые СУБД функции, а не реализовать их самостоятельно (не надо изобретать велосипед);

Особенности управления одновременным доступом

Незнание особенностей приведет к тому, что:

- будет нарушена целостность данных;
- приложение будет работать медленнее, чем предусмотрено, даже при небольшом количестве пользователей;
- будет потеряна возможность масштабирования для большего числа пользователей.

Пример.Реализация блокировок

- Типичный код, использованный разработчиками для резервирования неких ресурсов:

- create table resources (resource_name varchar2(25) primary key, ...);
- create table schedules (resource_name varchar2(25) references resources, start_time date, end_time date);

В чем ошибка?

- select count(*) from schedules where resource_name = :room_name and (start_time between :new_start_time and :new_end_time or end_time between :new_start_time and :new_end_time)

- …. Если выдавалось значение 0 – ресурс резервировался…..

Как правильно (ORACLE)

- select * from resources where resource_name = :room_name FOR UPDATE;
- select count(*) from schedules where resource_name = :room_name and (start_time between :new_start_time and :new_end_time or end_time between :new_start_time and :new_end_time)

- …. Если будет получено значение 0 – ресурс действительно можно резервировать…..

Зависят ли простейшие запросы от СУБД?

- Требуется перенести с Transact SQL на PL/SQL следующий код:

- declare l_some_variable varchar2(25);
- begin
- if (some_condition)
- then l_some_variable := f(...);
- end if;
- for x in
- (select * from T where x = l_some_variable) loop ...

- Нужно ли здесь что-то переписывать?

Проблема - значение NULL

- Разные СУБД по разному обрабатывают значение NULL. Sybase, SQL Server допускают сравнение (=) со значением NULL, а ORACLE – нет.

Пример

- select * from dual;
- ----------
- select * from dual where null=null;
- no rows selected
- ----------
- select * from dual where null<>null;
- no rows selected

Как правильно переписать запрос?

- select * from T where x = l_some_variable (SQL Server)
- select * from t where nvl(x,-1) = nvl(l_some_variable,-1) (ORACLE)
- Индекс –
- create index t_idx on t(nvl(x,-1));

Стандарты SQL

- Если все СУБД соответствуют стандарту SQL92, они должны быть одинаковы…
- SQL92 — это стандарт ANSI/ISO для СУБД. Он является развитием стандарта ANSI/ISO SQL89. Этот стандарт задает язык (SQL) и поведение (транзакции, уровни изоляции и т.д.) для СУБД. Многие коммерческие СУБД ФОРМАЛЬНО соответствуют стандарту SQL92. А знаете ли, как немного это значит для переносимости запросов и приложений?

Стандарт SQL92 (4 уровня)

- Начальный уровень
- Переходный
- Промежуточный
- Полный

Начальный уровень

- Именно этому уровню соответствует большинство промышленных СУБД. Этот уровень является незначительным развитием предыдущего стандарта, SQL89. Ни одна СУБД не сертифицирована по более высокому уровню. Более того, фактически Национальный институт стандартов и технологий (National Institute of Standards and Technology — NIST), агентство, сертифицировавшее соответствие стандартам SQL, сертификацией больше не занимается. В стандарт начального уровня не входят такие конструкции, как внешние соединения и т.д.

Переходный

- С точки зрения поддерживаемых возможностей это что-то среднее между начальным и промежуточным уровнем. Переходный уровень требует поддержки соответствующего синтаксиса внешнего и внутреннего соединения.

Промежуточный

- Этот уровень добавляет много возможностей, в том числе:
  - динамический SQL;
  - каскадное удаление для обеспечения целостности ссылок;
  - типы данных **DATE** и **TIME**;
  - символьные строки переменной длины;
  - выражения **CASE**;
  - функции **CAST** для преобразования типов данных.

Полный

- Добавляет следующие возможности:
  - управление подключением (CONNECT);
  - тип данных **BIT** для битовых строк;
  - отложенная проверка ограничений целостности;
  - производные таблицы в конструкции **FROM**;
  - подзапросы в конструкции **CHECK**;
  - временные таблицы.

Возможности и функции

- Нужно хорошо понимать, что именно предлагает конкретная СУБД, и полностью использовать ее возможности.
- Не надо придумывать универсальные механизмы репликаций, систем обмена сообщениями, заданий и т.п. Они есть почти во всех СУБД, только совсем непохожи друг на друга…

Как ограничить количество подключений пользователя к системе?

- Разработчики, не читающие документацию умудряются придумывать очень изощренные решения этой проблемы. А между тем (в ORACLE)….

Пример (как ограничить количество подключений в ORACLE)

- create profile one_session limit sessions_per_user 1;
- alter user scott profile one_session;
- alter system set resource_limit=true;

Производительность приложения

- О производительности необходимо думать уже на уровне проекта, а затем непрерывно проверять в процессе разработки. Это нельзя откладывать на потом. Нельзя поставлять приложения только с первичными ключами, вообще без дополнительных индексов. Необходимо проводить нагрузочное тестирование с большими объемами данных и имитацией реального количества пользователей.

Взаимоотношения администраторов БД и разработчиков

- В основе большинства успешно работающих информационных систем лежит плодотворное взаимодействие между АБД и разработчиками приложений.

Разграничение обязанностей

- Разработчик не обязан знать, как устанавливать и конфигурировать программное обеспечение. Этим должен заниматься АБД и, возможно, системный администратор.
- Основная обязанность АБД - резервное копирование и восстановление базы данных.
- Разработчик может менять ряд параметров сеанса, но за параметры уровня базы данных отвечает АБД.

- Разработчики могут не знать, как запустить СУБД, но должны уметь работать в ней.
- АБД связывается с разработчиком, заметив, что запросы потребляют слишком много ресурсов, а разработчик обращается к АБД когда не знает, как ускорить работу системы.
- Хороший разработчик обычно очень плохой АБД, и наоборот. У них разные навыки и опыт, а также (как правило), разное устройство ума и личностные характеристики.

Резюме

Для создания хороших приложений необходимо:

- понимать архитектуру СУБД;
- понимать как выполняется управление одновременным доступом;
- не воспринимать СУБД как черный ящик, устройство которого понимать не обязательно;
- не изобретать “велосипед”;
- решать проблемы как можно проще, максимально используя возможности конкретной СУБД.