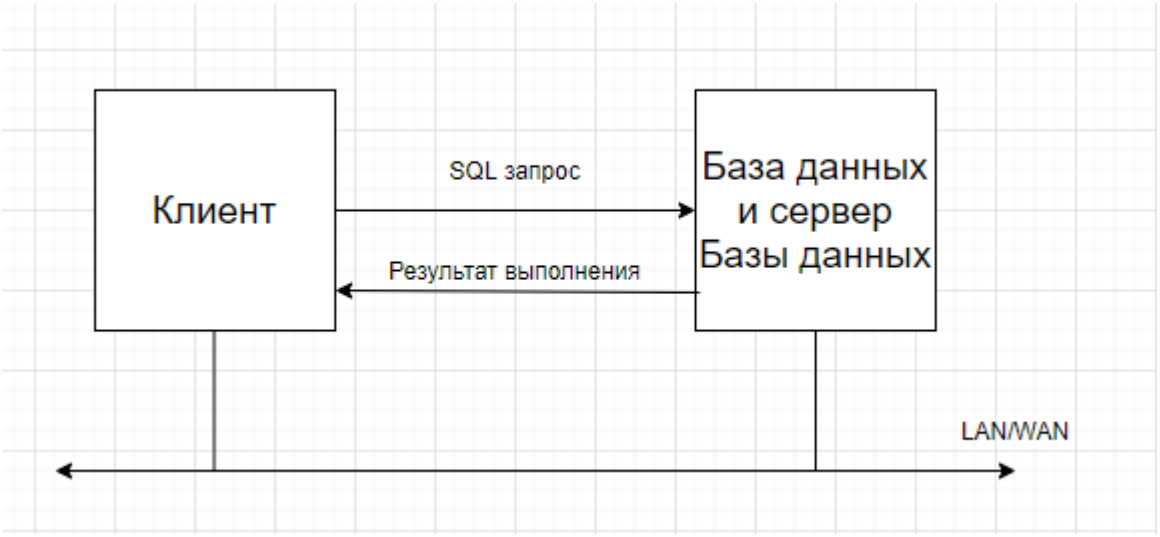
Современные СУБД(На примере СУБД Informix)

Основы организации и функционирования СУБД

Корпорация Informix Software является одним из лидеров рынка программного обеспечения UNIX-систем. Продукты Informix-a - это инструментальные средства разработки распределенных информационных систем, основанных на реляционных базах данных.

В каталоге фирмы Informix нет систем для конечного использования. Пользователи фирмы являются программистами. Программные продукты Informix-a - это, в основном, инструменты для создания прикладных программ. Сами прикладные программы следует искать в каталогах программистских фирм, выбравших продукты Informix-a как средства разработки. Корпорация Informix регулярно выпускает бюллетень с описанием разработанных приложений.

В основе всех этих приложений лежит модель "клиент - сервер". При этом собственно база данных располагается на UNIX-компьютере или файл-сервере NetWork, а прикладная программа(клиент) может выполняться на том же или другом UNIX-компьютере, а также на PC под DOS или MS-Windows. Взаимодействие клиента и сервера ведется на языке SQL. Клиент передает серверу SQL-запросы, а сервер шлет клиенту результат их выполнения. Поддерживаются сетевые протоколы TCP/IP и IPX/SPX.



Программные продукты Informix-a делятся на 3 категории:

- сервера
- средства для построения прикладных программ
- средства связи

SQL-сервера

Informix поставляет два сервера - SB-сервер(Standart Bugine) и Online-сервер. Online-сервер обеспечивает доступ к базе данных под ОС-UNIX и NetWork, SB-сервер может работать под UNIX и DOS.

Разница в SQL связано с данными переменной длины, которые поддерживаются только Online-сервером, а также с некоторыми административными действиями, по расположению пользовательских и системных данных на дисках.

Хранение данных

Основные отличия между серверами определяются способами хранения данных: при использовании SB-сервера база данных представляет собой каталог в файловой системе, таблицы хранятся в файлах, индексы в дополнительных файлах и тд. Online-сервер представляет данные(или их фрагменты) вне файловой системы, создавая на них собственную структуру хранения данных.

Весть ввод-вывод производится страницами, все данные, все индексы, вся системная информация организована с использованием страниц. Размер страницы - 2^n килобайта.

Совокупность физически последовательных страниц называется экстент . Используя то, что страницы в экстентах расположены физически последовательно, Online-сервер может определить адрес требуемой страницы без использования дополнительных ссылок. Хранение всех данных таблицы в одном экстенте наиболее предпочтительно. При росте объемов данных увеличивается количество используемых экстентов. Правильным подбором размеров экстентов для каждой таблицы пользователь может резко увеличить эффективность работы с системой.

Физическое место на диске, на котором размещаются экстенты, называется чанком(Chunk). Совокупность чанков образует то пространство , в котором Online сервер размещает базы данных: таблицы, индексы и системную информацию. Понятие чанка является очень существенным для баз данных, увеличивающихся с течением времени и превосходящих рано или поздно любое заранее выделенное количество дисковой памяти. Добавление чанка к существующей системе позволит ей неограниченно продолжать расти.

За счет описанной организации диска повышается скорость работы и надежность хранения. Для доступа нескольких процессов к одним и тем же данным Online-сервер использует разделенную память.

Кроме того, Online-сервер поддерживает типы данных переменной длины, отсутствующие в SB-сервере - строки переменной длины и так называемые "большие двоичные объекты", неструктурированные данные практически неограниченного объема, которые могут содержать графическую информацию, фотографии, звуки и т.д.

Создание и модификация схем баз данных

Каждая база данных - это совокупность отношений (называемых также таблицами). Создание и уничтожение базы данных производится командами **CREATE**

DATABASE/DROP DATABASE, выбор текущей базы данных - командой **DATABASE**. В каждый момент времени может быть только одна текущая база данных. Для смены текущей базы данных необходимо закрыть ее командой **CLOSE DATABASE** и открыть новую командой **DATABASE**. Для работы одновременно с данными в нескольких базах данных существует специальный механизм синонимов.

Создание и изменение структуры таблиц в базе данных производится командами **CREATE TABLE/ALTER TABLE/DROP TABLE**. При добавлении, удалении или изменении длины колонки все данные в таблице сохраняются, дополнительная выгрузка и последующая загрузка не требуется. В отношениях поддерживаются строки фиксированной длины, целые, плавающие числовые данные, интервалы времени, а в Online-сервере поддерживаются строки переменной длины и большие двоичные объекты.

В таблице можно задать целый ряд ограничений на размещенные данные предложением **CHECK**(например, **CHECK** num > 1), можно так же указать, что данные в нескольких колонках должны составлять уникальную комбинацию(**PRIMARY KEY**). Более того, можно описать связь таблиц, имеющих общие колонки.

Например, пусть имеется таблица таблица счетов и таблица субсчетов:

CREATE TABLE accounts(# таблица счетов

acc_num INTEGER, # номер счета

acc_num INTEGER, # тип счета

CHECK(acc_type BETWEEN 1 AND 10), # ограничение типа счета

.... # другие данные

PRIMARY KEY(acc_num, acc_type) # указанная пара уникальна

)

CREATE TABLE sub_acc(# таблица субсчетов

ref_num INTEGER, # номер счета

ref_num INTEGER, # тип счета

sub_acc INTEGER, # номер субсчета

.... # другие данные

FOREIGN KEY(ref_num, ref_type)

REFERENCES accounts(acc_num, acc_type) # связь между таблицами

)

В этом примере говорится, что поля **ref_num** и **ref_type** любой строки таблицы **sub_acc** должны совпадать с полями acc_num и **acc_type** какой либо строки таблицы accounts. Если при этом занести в базу данных запись о субъекте несуществующего счета, то сервер выдаст ошибку.

Команда **CREATE SYNONYM** позволяет создавать новые имена для существующих таблиц. При этом оказывается возможным обращаться к другим базам данных и даже к другим компьютерам в сети.

Например:

CREATE SYNONYM rem FOR test@stankin.user.tab

Запись означает, что любое обращение к таблице rem в текущей базе данных будет автоматически переадресовано на компьютер stankin, в базу данных test, к таблице tab, владельцем которой является пользователь user. Обратите внимание, что оказывается возможным переместить таблицу из одной базы данных в другую, оставив в первой базе ссылку на ее новое местоположение, при этом все необходимые действия для доступа к содержимому таблицы будут сделаны сервером автоматически, и ни одна программа даже не заметит этого.

Данные, описывающие структуру базы данных, хранятся в самой базе и в специальных системных таблицах и могут быть безо всяких проблем считаны.

Все системные таблицы подробно описаны в документации, что дает возможность искать хитроумные программы, аккуратно работающие с базами данных, структура которых определяется в процессе работы самой программы.

Целостность данных

Для обеспечения целостности данных используются логическое и физическое протоколирование. С помощью протоколов база данных приводится в корректное состояние после дискового сбоя или своя системы, с помощью протоколов реализовано управление транзакциями. Чтобы воспользоваться преимуществами, предоставляемыми системой протоколирования, требуется создать базу данных с протоколированием следующей командой:

CREATE DATABASE name WITH LOG

Для баз данных с протоколированием имеется возможность управления транзакциями. Эта возможность позволяет выполнить серию операций над базой данных как единое действие. Каждая серия операций при этом заканчивается либо подтверждением транзакции, что гарантирует внесение всех изменений в базу, либо отменой транзакции, гарантирующей, что ни одно из изменений в базу не внесено.

Для начала транзакции используется предложение **BEGIN WORK**, для подтверждения - **COMMIT WORK**, для отката - **ROLLBACK WORK**. При откате транзакции в Online-

сервере откатываются также операции определения типов данных(например, **CREATE TABLE**)

Если не использовать явного определения транзакций, каждое предложение SQL будет представлять собой неявную транзакцию. При этом все успешно выполненные предложения подтвердят свою неявную транзакцию, а неуспешные откатят ее.

В обоих серверах имеется механизм восстановления, использующий протоколы для восстановления баз данных на момент последней завершенной транзакции. Содержимое протоколов накапливается в оперативной памяти до тех пор, пока транзакции не подтвердятся. В этот момент происходит запись протокола на диск, гарантирующая восстановление транзакции в случае аппаратного или программного сбоя. В случае Online сервера восстановление системы после сбоя происходит автоматически. В случае стандартного сервера из за использования системы для хранения данных в некоторых случаях может оказаться необходимым восстановление базы данных из архива-копии.

Для увеличения производительности системы протоколы можно буфферизировать следующей командой:

CREATE DATABASE name WITH BUFFERED LOG

Уровни изоляции и одновременный доступ

Online-сервер обеспечивает несколько способов, с помощью которых один процесс может защититься от конфликтов с другими процессами, которые работают с теми же строками той же базы данных. Эти способы называются уровнями изоляции. Реализация уровней изоляции производится с помощью замков. Замки позволяют нескольким процессам считать одни и те же данные, но не позволяют эти данные изменить. Уровни изоляции относятся только к чтению данных, они не имеют никакого отношения к их изменению. Если строка изменена, она защищается; все остальные процессы не могут получить к ней доступ до завершения транзакции.

Предоставлено 4 уровня изоляции

- грязное чтение
- подтвержденное чтение
- стабильное чтение
- повторяемое чтение

Грязное чтение означает отсутствие изоляции от других процессов. При этом записи вообще не используются. Если строка получена грязным чтением, то, например, нет гарантии, что эта строка не была в создана транзакции, которая впоследствии была изменена.

Подтвержденное чтение гарантирует, что извлеченная строка была создана подтвержденной транзакцией. Система ставит на строку замок, гарантируя, что в данный момент никто эту строку не модифицирует. Однако замок сразу же снимается,

поэтому во время использования строки тем, кто ее считал, она может быть модифицирована или даже удалена из базы.

Стабильное чтение - это такой уровень изоляции, когда система ставит на строку замок и не снимает его до тех пор пока не будет запрошена другая строка из базы. никакой другой процесс не сможет удалить или изменить данную строку.

При повторяемом чтении система ставит замки на каждую строку, извлеченную в ранних транзакциях. Другие процессы могут просматривать те же строки, но не могут изменить их. Все замки снимаются только по завершении транзакции.

Для установки требуемого уровня изоляции используют команду:

SET ISOLATION

Для баз данных без протоколирования единственным допустимым уровнем изоляции является грязное чтение.

Манипуляции с данными

Над данными в таблице можно выполнять любые манипуляции командами: **INSERT, SELECT, UPDATE, DELETE**. Это стандартные операции SQL, с их помощью производятся все операции непосредственно с данными.

Из нескольких таблиц с помощью команды **SELECT** можно построить вид(**VIEW**), а потом использовать его как новую таблицу. При некоторых более или менее естественных ограничениях вид можно использовать для модификации данных в базе.

Сохраняемые процедуры

Механизм сохраняемых процедур позволяет хранить в самой базе данных программы в скомпилированном виде. Процедуры пишутся на языке SPL. Это достаточно развитый язык, имеющий локальные и глобальные переменные, массивы, условия, циклы. Сохраняемые процедуры создается создаются командой **CREATE PROCEDURE**, а выполняются командой

EXECUTE PROCEDURE name INTO возвращаемое значение.

Сохраняемые процедуры представляют собой мощное средство программирования сервера. Они позволяют повысить производительность системы.