Организация распределенного каталога

Глобальный каталог (ССД)

Некоторые варианты хранения глоюального системного каталога:

1. **Централизованный каталог**. Весь каталог хранится в одном месте, т.е. на центральном узле.

2. **Полностью реплицированный каталог**. Весь каталог полностью хранится на каждом узле.

3. **Секционированный каталог**. На каждом узле содержится его собственный каталог для объектов, хранимых на этом узле. Общий каталог является объединением всех разъединенных локальных каталогов.

4. **Комбинация первого и третьего вариантов**. На каждом узле хранится собственный локальный каталог (как в п. 3), кроме того, на одном центральном узле хранится унифицированная копия всех этих локальных каталогов (как в п. 1).

Для каждого подхода характерны определенные недостатки и проблемы.

В реальных СУБД используются и другие подходы.

Именование объектов в System R*

Один из самых распространенных подходов (System R, Oracle и др.):

имя_пользователя.имя_объекта

В System R* используется развитие этого подхода. Системное имя объекта включает четыре компонента:

1. Идентификатор пользователя – создателя объекта;
2. Идентификатор узла сети, в котором выполнялась операция создания объекта;
3. Локальное имя объекта, присвоенное ему при создании;
4. Идентификатор узла, в котором объект располагался непосредственно после своего создания (объект может перемещаться из одного узла в другой при выполнении операции MIGRATE).

В запросе на SQL можно использовать системные имена объектов, но разрешается использовать и короткие локальные имена (либо локальное имя, квалифицированное именем пользователя).

Например, системное имя MARYLIN @ NEW YORK . STATS @ LONDON идентифицирует объект (например, хранимое отношение) с локальным именем STATS, созданный пользователем Marylin на узле в Нью-Йорке и изначально хранившийся на узле в Лондоне. Такое имя **гарантировано от каких-либо изменений** даже при перемещении этого объекта на другой узел.

Организация распределенного каталога в System R*

При указании локального имени в System R* возможны две интерпретации локального имени:

1. Имя интерпретируется как часть системного имени, и в этом случае по умолчанию дополняется до системного, исходя из идентификатора узла, в котором производится компиляция, и идентификатора пользователя, от имени которого она производится (если имя пользователя не указано явно).
2. Локальное имя рассматривается как ранее определенный синоним системного имени.

Для определения синонимов SQL в System R* расширен оператором вида:

CREATE SYNONYM <relation-name> FOR <system-wide-name>

При выполнении такого предложения в локальный каталог заносится соответствующая информация.

Таким образом, при компиляции запроса всегда можно определить системные имена всех употребляемых в нем отношений: либо они явно указаны, либо могут быть получены на основе информации из локального каталога.

Организация распределенного каталога в System R*

После создания синонима

create synonym mstats for marylin @ new york . stats @ london;

можно использовать одно из следующих выражений:

SELECT . . . FROM STATS . . . ;

SELECT . . . FROM MSTATS . . . ;

В первом случае используется локальное имя.

Во втором случае системное имя определяется помощью опроса соответствующей **таблицы синонимов**.

Эти таблицы рассматриваются как первый компонент каталога, а каждый узел содержит набор таблиц всех пользователей, известных на данном узле, с отображением синонимов пользователя на системные имена.

Кроме таблицы синонимов на каждом узле поддерживаются:

1. Элемент каталога для каждого объекта, **созданного** на этом узле.
2. Элемент каталога для каждого объекта, **хранимого** в данный момент на этом узле.

Т.о., обращение к любому объекту потребует не более 2-х удаленных обращений: сначала – на узел-создатель, затем – на узел, на котором сейчас хранится объект.

Для этого при перемещении объекта **X**, который был создан на узле *A*, с узла *B* на узел *C* производятся следующие действия:

1) Добавляется элемент **X** в каталог узла *C*, куда перемещен объект.

2) Удаляется элемент **X** из каталога узла *B*, с которого перемещен объект.

3) В каталоге узла *A* изменяется информация о текущем месте хранения объекта **X** (с *B* на *C*).

Организация распределенного каталога в System R*

Концепция распределенного каталога System R* основана на наличии у каждого объекта РБД уникального системного имени. Принято следующее соглашение: информация о размещении любого объекта базы данных (идентификатор текущего узла, в котором размещен объект) сохраняется в локальном каталоге того узла, в котором объект располагался непосредственно после создания (родового узла). Для получения полной информации об отношении в общем случае необходимо сначала воспользоваться локальным каталогом узла, в котором происходит компиляция, затем обратиться к удаленному каталогу родового узла данного отношения и в заключение воспользоваться каталогом текущего узла. Применяется некоторая оптимизация этой процедуры. В локальном каталоге узла могут храниться копии элементов каталога других узлов (своего рода кэш-каталог). Согласованность копий элементов каталога не поддерживается. Эта информация используется на первой стадии компиляции запроса, а затем, на второй стадии, если информация, касающаяся некоторого объекта, оказалась неточной, она уточняется на основе локального каталога того узла, в котором объект хранится в настоящее время. Обнаружение некорректности копии элемента каталога производится за счет наличия при каждом элементе каталога номера версии. Если учесть достаточную инерционность системной информации, эта оптимизация может оказаться существенной.

Именование объектов в Oracle

Обращение к сервисам базы данных (серверу БД, очереди печати, серверу электронной почты и т.д.) происходит по уникальному имени (global name).

Для БД оно состоит из основного имени $ORACLE_SID, назначаемого ей при создании (длиной не более 8-и символов) и сетевого домена БД, например:

sales.parts@east.compworld

– обращение к таблице PARTS базы данных SALES, расположенной на сервере east.compworld.

Для получения доступа к удаленной БД нужно установить связь с этой БД с помощью специальной команды языка SQL – LINK. При формировании связи могут учитываться учетные сведения пользователя для обеспечения безопасности данных, но это требует дополнительных усилий для распространения учетных сведений пользователя в сети: во-первых, нужно создать учетный раздел пользователя на удаленном сервере; во-вторых, пакеты регистрации в сети желательно шифровать, т.к. сеть не защищена от доступа посторонних лиц.

Связи в распределенной БД Oracle

Примеры.

Локальная база данных – HQ.ACME.COM.

Удаленная база данных – SALES.ACME.COM.

Создание общей связи баз данных к удаленной базе данных SALES:

```
CREATE PUBLIC DATABASE LINK sales.acme.com USING 'dbstring';
```

Создание личной связи баз данных для создателя этой связи:

```
CREATE DATABASE LINK sales CONNECT TO scott
    IDENTIFIED BY tiger;
```

Фраза CONNECT TO специфицирована явно. При установлении сессии в удаленной базе данных через эту связь баз данных будет использоваться идентификация SCOTT/TIGER.

Фраза USING опущена. Поэтому, когда используется эта личная связь баз данных, должна существовать одноименная общая или сетевая связь баз данных, содержащая строку базы данных для установления соединения с удаленной базой данных.

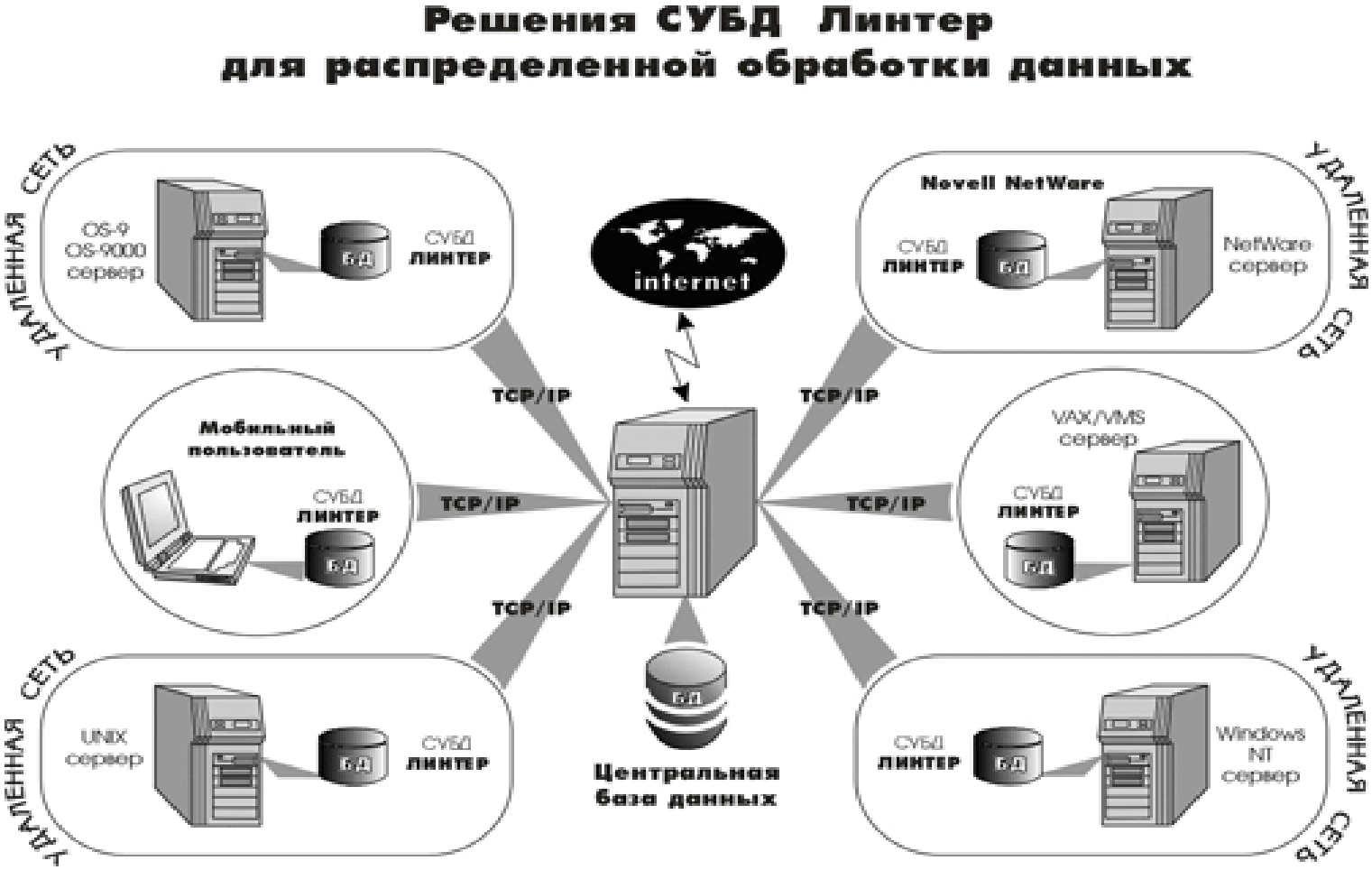
Глобальный словарь в Ingres

В Ingres глобальный словарь реализуется с помощью компоненты Ingres/Star. Эта компонента извлекает информацию из всех локальных словарей данных и выполняет оптимизацию запросов. Недостатком такого подхода является то, что все данные словарей собираются на центральном узле Ingres/Star и при его отключении или сбое теряется доступ к остальным узлам распределенной БД. Для восстановления доступа придется создавать другой центральный узел.

Глобальный словарь в ЛИНТЕР

- Концепция распределённости СУБД ЛИНТЕР позволяет прозрачно обрабатывать запросы к данным, находящимся в различных базах данных вне зависимости от их физического расположения и обеспечивает равноправный доступ пользователей к разным базам данных, расположенным в различных узлах вычислительной сети.
- Основным понятием концепции является **сетевое** или **логическое ЛИНТЕР-имя**, которое представляет собой восьмисимвольный идентификатор, хранящийся в специальном файле ЛИНТЕР-имен - **nodetab**. В этом файле ЛИНТЕР-имена связаны с сетевыми параметрами узла запуска ядра ЛИНТЕР.
- ЛИНТЕР-имя однозначно определяет в текущей операционной среде базу данных, для которой будет запущено независимое ядро СУБД ЛИНТЕР. Части выполняемого запроса, которые необходимо выполнить именно в данной базе будут переправляться по указанному в nodetab адресу с использованием ЛИНТЕР-имени.
- Список баз данных (ЛИНТЕР-имён), которые могут участвовать в процессе выполнения распределённых запросов, содержится в специальной системной таблице - **SERVERS**. Эта таблица входит в системный словарь базы данных и подчиняется стандартным правилам работы со словарями, принятым в СУБД ЛИНТЕР.

Распределенная обработка данных в ЛИНТЕР



Глобальный словарь в ЛИНТЕР

В **SERVERS** хранится имя, по которому работает СУБД внутри и которое будет передано вовне для определения, на какой из удалённых серверов нужно пересылать запрос за недостающими данными. В **SERVERS** не хранится связь имени с сетевой конфигурацией. Таким образом, при изменениях конфигурации (через файл nodetab) трансляция имен внутри базы данных останется неизменной.

Создание/удаление узла распределённой базы данных производится специальным SQL-запросом CREATE/DROP NODE.

Главная компонента аппарата распределённости - **loltp** - процесс, который работает с удаленным сервером. Он принимает запрос от локального ядра во внутреннем формате и формирует из него SQL-запрос к удалённому серверу. Соответственно, при этом используется сетевой клиентский драйвер.
Другая модификация системы позволяет создавать многомашинные СУБД комплексы на основе асинхронной репликации.
Такие системы используются чаще всего в WEB-приложениях, где запросы на поиск гораздо более часты, нежели запросы на модификацию (поисковые системы, интернет-магазины и т.п.). В таких приложениях можно достичь практически пропорционального масштабирования при увеличении числа машин в комплексе, например, при двукратном увеличении числа ЭВМ, мы получим практически двукратное увеличение скорости обработки общего потока запросов.