Оптимизация форм и времени доступа в базах данных.

Александр Кошелев
Тбилисский Государственный Университет им. И. Джавахишвили

Аннотация
В работе рассматриваются вопросы оптимизации форм и времени доступа в уже функционирующих системах построенных на основе баз данных. Приводятся доводы в пользу применения промежуточного звена, в частности временных таблиц.

Ключевые слова: Оптимизация базы данных, интернет, SQL

Вступление

К весьма значимой части информационной системы на предприятии, можно отнести подсистему анализа, обработки и отчетности информации. Трудно представить серьезную организацию без аналитического центра, в который входят те или иные отделы, исходя из профессиональной деятельности. В подобных системах одна из основных задач специалистов информационных технологий, уменьшить время ответа системы, и свести его к минимуму при сложившихся обстоятельствах.

Данная статья посвящена именно этой проблеме, в ней приведены тесты, замечания, высказаны некоторые новые идеи. Здесь приведены попытки, наглядно продемонстрировать целесообразность введения дополнительного звена в системах предоставления отчетов и проведения анализа. В конкретном случае, дополнительное звено может представлять собой временные таблицы хранения данных, куда помещаются промежуточные или конечные результаты запросов для последующей обработки. В информационных базах с большим объемом данных и с высоким показателем количества запросов в единицу времени это дает значительное увеличение скорости доступа. В дальнейшем вполне возможна реализация общей автоматизированной системы решающей подобные вопросы.

Об анализе и отчетности данных.

Следует заметить, что имеется в виду под понятиями анализа и отчетности. Отчетность есть вывод различного вида форм и отчетов на основе имеющихся данных, к примеру, вывод информации из базы данных, а под анализом понимается вывод результатов на основе проведенных вычислений над данными.

В условиях стремительного роста интенсивности информационного обмена в современных системах все большее количество крупных, а в последнее время и более мелких предприятий начинают использовать распределенные системы и новейшие информационные технологии в своей профессиональной деятельности. Уже трудно представить успех организации не идущей в ногу со временем в этом направлении.

Рождаются новые и отходят в сторону старые проблемы, связанные с построением распределенных систем. На одной из таких проблем мы остановимся и попытаемся дать ответы и решения, поставить эксперименты, предложить новые идеи.

Представим себе распространенную ситуацию, когда предприятию это может быть банк с множеством филиалов, сеть магазинов, сеть складских помещений, сеть банкоматов, множество контрольно-пропускных пунктов в метрополитене и т.п. нужно иметь централизованное хранилище данных для проведения дальнейшего анализа и получения отчетности.

На данном этапе предполагается, что уже решен вопрос организации работы, предприятия, проложена сеть, размещены локальные базы данных, проведены линии связи и т.п. Представим схематично работу предприятия в виде следующей вполне возможной схемы. (Диаграмма 1)

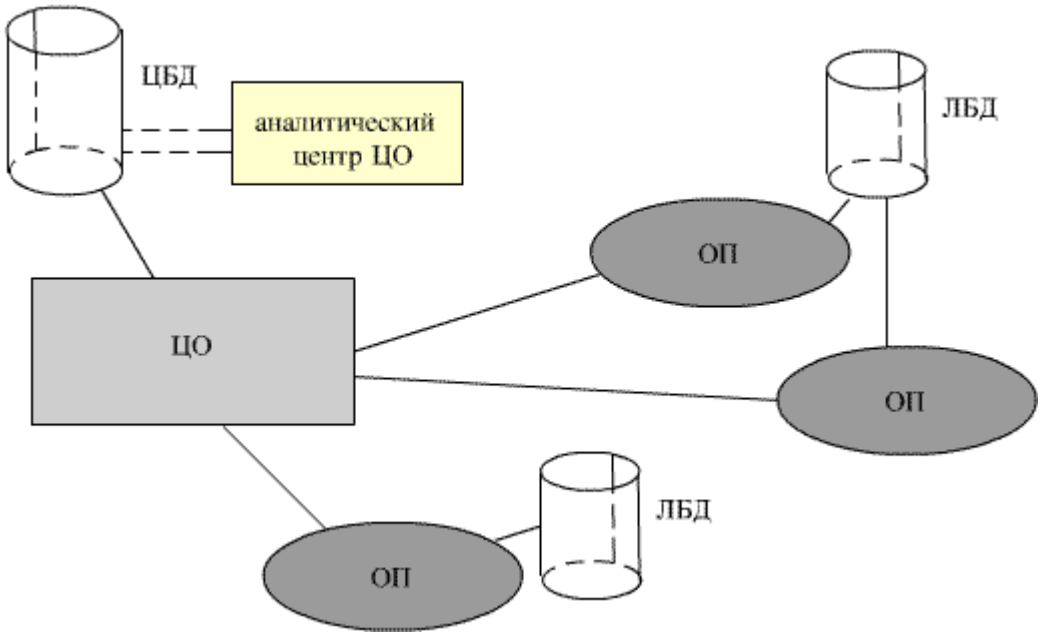

Диаграмма 1

ЦО центральный офис
ОП отделение предприятия
ЛБД локальная база данных
ЦБД центральная база данных

Следующая проблема в развертывании предприятия - определение информационных потоков данных. Но можно заметить, что в подавляющем большинстве случаев требуется организовывать централизованное хранилище данных для проведения анализа, сбора статистики, получения отчетов. Размещение данного хранилища, чаще всего решается в центральном офисе организации.

Теперь о непосредственной организации самого сбора. Здесь, при наличии быстрых каналов связи можно предложить прямое обращение к локальным базам данных и получение необходимых данных (online режим), использование почты (offline режим), и многое другое, но наиболее современное и продуктивное решение представляется в использовании средств репликации.

Не будем останавливаться на вопросах организации репликации, о которых написано немало литературы [1,2], и статей. Будем считать, что наши данные централизованны, и мы подошли к тому, на чем остановимся.

Собрав информацию в централизованное хранилище, остается решить вопрос о достижении реального времени ответа системы (базы данных) на запросы пользователей. Отчетность, обработка и анализ данных должны проводиться во временных рамках удовлетворяющих поставленным целям, но что можно заметить с уверенностью так это то, что время задержки надо стремиться минимизировать. Как же можно это осуществить в уже существующих системах или где искать возможности оптимизации. Наиболее часто встречающийся механизм получения данных можно представить в виде следующей схемы (Диаграмма 2)

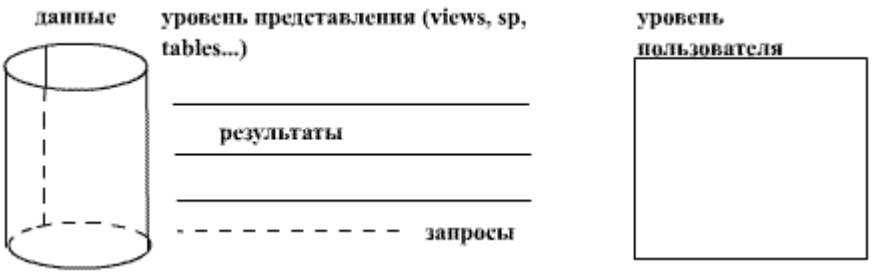

Диаграмма 2

Имеются различные, новые технологии ведения и хранения статистики. Можно отметить технологию (OLAP – Online analytical processing) [3,4,5], но она более подходит к сбору статистики, на основе имеющейся информации, когда строятся многомерные ‘кубы’ данных, измерения которых есть параметры, а на пересечении измерений находятся интересующие значения, при определенных входных параметрах. Скажем, нас интересует объем продаж того или иного продукта во времени. Данные можно организовать в виде двухмерного ‘куба’. (Диаграмма 3)

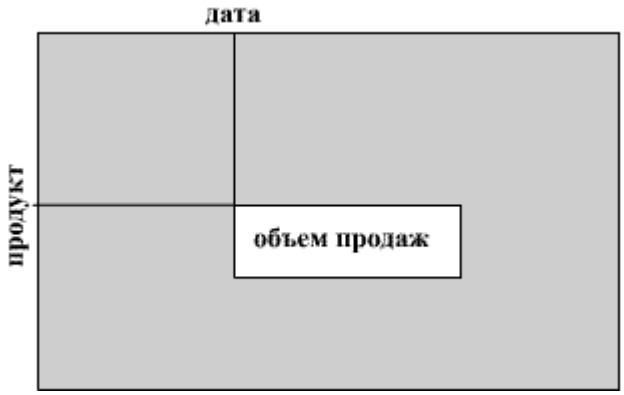

Диаграмма 3

Зачем же изобретаются такого рода технологии? Несомненно, что они так же решают по своему один и тот же вопрос организации хранения и минимизации времени доступа.
Надо заметить, что при всем своем преимуществе они совершенно не подходят к системам оперативной обработки транзакций (OLTP - Online Transaction Processing) [3, 4,5] где требуется в реальном маштабе времени обрабатывать огромное количество транзакций и в то же время выдавать отчеты и проводить необходимый анализ.

Поставим один небольшой эксперимент, который наведет нас на вполне возможный путь решения поставленной проблемы.

Предположим, к рассмотрению две таблицы. Это могут быть номера счетов (таблица temp1 поле ID) и их оборот (таблица temp2) или описание продукта и количество продаж по регионам, соответственно и многое другое. То есть, имеем вполне типичный, “нормализованный” фрагмент базы данных.(Диаграмма 4)

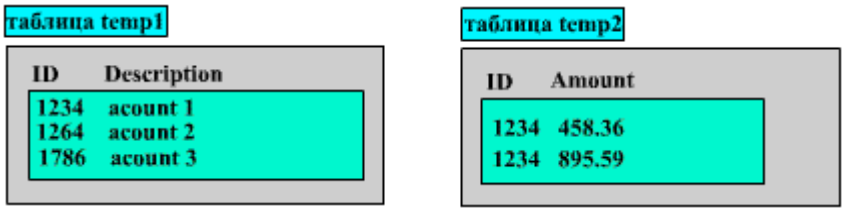

Диаграмма 4

Какой же запрос типичен в подобной ситуации? На этот вопрос однозначного ответа дать не представляется возможным, но можно предположить, что наиболее вероятным является (исходя из рассмотренного нами конкретного фрагмента) получение сумм из таблицы temp2 сгруппированных по ID из таблицы temp1.

Что же можно предложить для минимизации времени запроса в случае такого запроса. Было решено применить следующее, ввести временную таблицу test, поместить туда данные предыдущего запроса, и затем использовать обычную выборку из временной таблицы.

Ниже приводится таблица результатов и сам script на sql [6] (протестировано на Microsoft SQL Server).

```
/* создание таблиц  */
CREATE TABLE [dbo].[temp1] (
        [ID] [int] NOT NULL ,
        [Description] [varchar] (50) NULL
) ON [PRIMARY]
GO
CREATE TABLE [dbo].[temp2] (
        [ID] [int] NOT NULL ,
        [Amount] [money] NULL
) ON [PRIMARY]
GO
CREATE TABLE [dbo].[test] (
        [uid] [uniqueidentifier] NOT NULL ,
        [ID] [int] NOT NULL ,
        [Amount] [money] NULL ,
        [Description] [varchar] (50) NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[temp1] WITH NOCHECK ADD
        CONSTRAINT [PK_temp1] PRIMARY KEY  NONCLUSTERED
        (
                [ID]
        )  ON [PRIMARY]
GO
 CREATE  INDEX [IX_ID] ON [dbo].[temp2]([ID]) ON [PRIMARY]
GO
 CREATE  INDEX [IX_uid] ON [dbo].[test]([uid]) ON [PRIMARY]
GO
 CREATE  INDEX [IX_ID] ON [dbo].[test]([ID]) ON [PRIMARY]
GO

/* заполняем таблицы данными */
declare @constant  int
set     @constant = 10000  /* здесь указываем значение константы */
set nocount on
create table #temp1(ID Int)
declare @counter int

set @counter = 1
while @counter <= @constant
```

```
  begin
   insert #temp1(ID) values(@counter)
   set @counter = @counter + 1
  end

insert into temp1(ID, Description)
select ID, 'test' + convert(varchar(50), ID) as Description from #temp1
drop table #temp1
set nocount off

set nocount on
create table #temp2(ID Int, Amount money)
declare @counter int,
        @counter_ins int

set @counter = 1
while @counter <= @constant
  begin
   set @counter_ins = 1
   while @counter_ins <= 10
    begin
     insert #temp2(ID, Amount) values(@counter, rand(@counter_ins)*@counter)
     set @counter_ins = @counter_ins + 1
    end
   set @counter = @counter + 3
  end

insert into temp2(ID, Amount)
select ID, Amount from #temp2
drop table #temp2
set nocount off

/* возращение  информации без использования временных таблиц */
select t1.ID, t2.Amount, t1.Description from
(
 (select ID, Description from temp1) t1 inner join
  (select ID, sum(Amount) as Amount from temp2 group by ID) t2
   on t1.ID = t2.ID
)
/* возращение  информации с использованием временных таблиц */
set nocount on
declare @uid uniqueidentifier
set @uid = newid()
insert into test(uid, ID, Amount, Description)
select @uid as uid, t1.ID, t2.Amount, t1.Description from
(
 (select ID, Description from temp1) t1 inner join
  (select ID, sum(Amount) as Amount from temp2 group by ID) t2
   on t1.ID = t2.ID
)
set nocount off
select * from test where uid = @uid
```

Вот результаты эксперимента: (Таблица 1)

@constant	заполнение temp1	заполнение temp2	заполнение test	select 'join'	select test
10000	15 секунд	16 секунд	1 сек.	1сек.	1сек.
100000	24 секунд	55 секунд	22 сек.	13сек.	9сек.
1000000	2м. 26 сек.	7м. 57 сек.	7м. 30 сек.	5м 2с top 100000	26с top 100000

Таблица 1

Первый столбец есть количество обрабатываемых строк, столбцы со второго по четвертый - время заполнения соответствующих таблиц, четвертый и пятый – время, затраченное на выдачу результатов без использования и с применением временных таблиц соответстенно.

Тестирование проводилось на ПК со следующими характеристиками:
аппаратная платформа: материнская плата MSI K7Pro2-A (MS 6330) , процессор AMD Athlon (Thunderbird) 650Mh, жесткий диск Qvantum Fireball LB/15GB/, ОЗУ 128 Mb.
программное обеспечение: ОС Windows 98, MS SQL Server 7.0

Из приведенных результатов видно, что при увеличении количества обрабатываемых строк при повторяюшихся запросах, целесообразно вводить временные таблицы.

Проведение этого теста, привело на мысль об организации промежуточного звена в предыдушей схеме (Диаграмма 5)

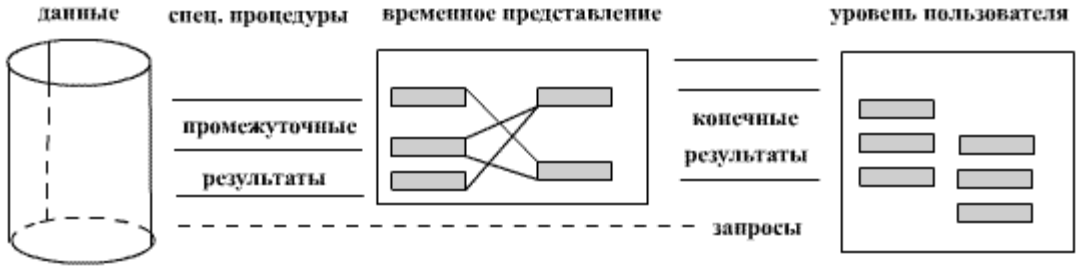

Диаграмма 5

Можно заметить из практики, что разумное использование данного подхода, принесло немало пользы, система с очень большим временем ответа стала вполне приемлемой и не давала сбои.

Далее приводится один из возможных путей внедрения и использования данного подхода.
1. Выявление критических процедур и запросов – выполнение которых необходимо для получения исходного материала для последуюшей обработки или получения отчетности.
2. Составление полного списка обнаруженных на предыдущем этапе "запросов" с указанием их масс (масса есть цена реализации конкретного запроса), в нее могут входить время выполнения, время использования процессора, количество операции ввода вывода и т.п.
3. Добавление специального, для ведения статистики кода в критические запросы. Кода который будет сохранять лог при обрашении к данным запросам.
4. Анализ статистики. Выявление наиболее используемых запросов.
5. Получение результатов и внедрение временных таблиц.

Далее приводится возможная реализация данного алгоритма для Microsoft SQL Server.

```
/* скрипт */
/* создание таблиц для сохранения статистики */
CREATE TABLE [dbo].[Analyse_Log] (
        [ID] [int] IDENTITY (1, 1) NOT NULL ,
```

```
        [When_Started] [datetime] NULL ,
        [When_Ended] [datetime] NULL ,
        [Who] [varchar] (50) NOT NULL ,
        [What] [int] NOT NULL
) ON [PRIMARY]
GO
CREATE TABLE [dbo].[DB_Elements] (
        [ID] [int] NOT NULL ,
        [DB_EL_Name] [varchar] (50) NOT NULL ,
        [DB_EL_ID] [int] NULL ,
        [Criterion_Weight] [money] NOT NULL ,
        [DB_EL_TYPE_ID] [int] NOT NULL
) ON [PRIMARY]
GO
CREATE TABLE [dbo].[DB_Elements_Types] (
        [ID] [int] NOT NULL ,
        [ElName] [varchar] (50) NOT NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[DB_Elements] WITH NOCHECK ADD
        CONSTRAINT [DF_DB_Elements_Criterion_Weight] DEFAULT (0) FOR
[Criterion_Weight],
        CONSTRAINT [PK_DB_Elements] PRIMARY KEY  NONCLUSTERED
        (
                [ID]
        )  ON [PRIMARY]
GO
ALTER TABLE [dbo].[DB_Elements_Types] WITH NOCHECK ADD
        CONSTRAINT [PK_DB_Elements_Types] PRIMARY KEY  NONCLUSTERED
        (
                [ID]
        )  ON [PRIMARY]
GO

 CREATE  INDEX [IX_DB_EL_ID] ON [dbo].[DB_Elements]([DB_EL_ID]) ON [PRIMARY]
GO
/* процедура заполнения уникальными идентификаторами критических запросов */
CREATE PROCEDURE SET_DB_ELEMENTS_ID
AS
SET NOCOUNT ON

BEGIN TRAN UPD_EL_ID
UPDATE DB_Elements
SET DB_EL_ID = OBJECT_ID(DB_EL_Name)

IF @@ERROR <> 0 GOTO ROLL_BACK
COMMIT TRAN DB_EL_ID

SET NOCOUNT OFF
RETURN(0)

ROLL_BACK:
```

```
ROLLBACK TRAN UPD_EL_ID
RETURN(1)

/* код добавляемый в критические запросы */
SET NOCOUNT ON
DECLARE @trans_id INT,
        @object_id INT

SET @object_id = 741577680 -- уникальный идентификатор
INSERT INTO Analyse_Log(When_Started, Who, What)
VALUES(getdate(), system_user, @object_id)
SET @trans_id = @@identity

-- запрос, который остается неизменным

select top 100 t1.ID, t2.Amount, t1.Description from
(
 (select ID, Description from temp1) t1 inner join
  (select ID, sum(Amount) as Amount from temp2 group by ID) t2
   on t1.ID = t2.ID
)

UPDATE  Analyse_Log
SET When_Ended = getdate()
WHERE ID = @trans_id
SET NOCOUNT OFF
RETURN(0)
```

После проведения первых трех пунктов алгоритма следует анализ статистики и на основе полученной информации внедрение временных таблиц.

В дальнейшем, вполне возможно и полезно составление специализированных, автоматических средств внедрения указанных возможностей увеличения скорости доступа.

Об Интернет.

Поговорим об Интернет. Несомненно это чудо двадцатого века которое подарила нам компьютеризация. Что это – для некоторых развлечение, средство зарабатывать деньги, возможность завоевывать новые рынки (реклама), для других найти спутника жизни, друга и многое другое, но все разнообразие можно заключить в две возможности : предоставлять и получать информацию. (Диаграмма 6)

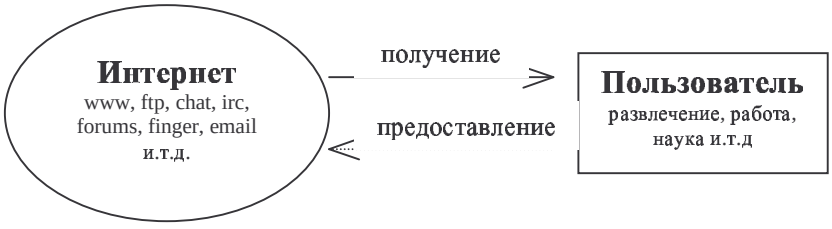

Диаграмма 6

Чем то напоминает доступ к базе данных. Можно продолжить проведение аналогии, сам Интернет это физический уровень, поисковые системы порталлы типа www.yahoo.com,

www.av.com, www.google.com, это концептуальный уровень доступа к физическому (Интернету).

Ну и что же сейчас делает пользователь, - он ведет себя также как программист некоторый период назад и пытается в большинстве случаев обойти концептуальный и обратится напрямую к физическому, запросы наподобие (www.sony.com, www.jvc.com и.т.д.) (Диаграмма 7)

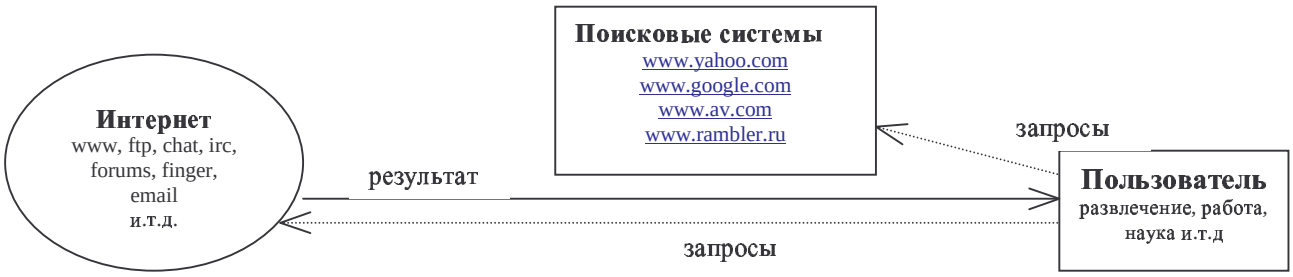

Диаграмма 7

Да и сами поисковые системы не совсем справляются со своими функциями – часто дублируют себя и предоставляют ссылки на одни и те же ресурсы. А что уж там говорить если они даже не в состоянии обработать и вернуть информацию от своего имени.

И зачем пользователю дали возможность набирать адрес в браузере. Попытаемся представить, а что можно сделать для улучшения? Одна из возможных попыток решить проблему приведена на диаграмме 8.

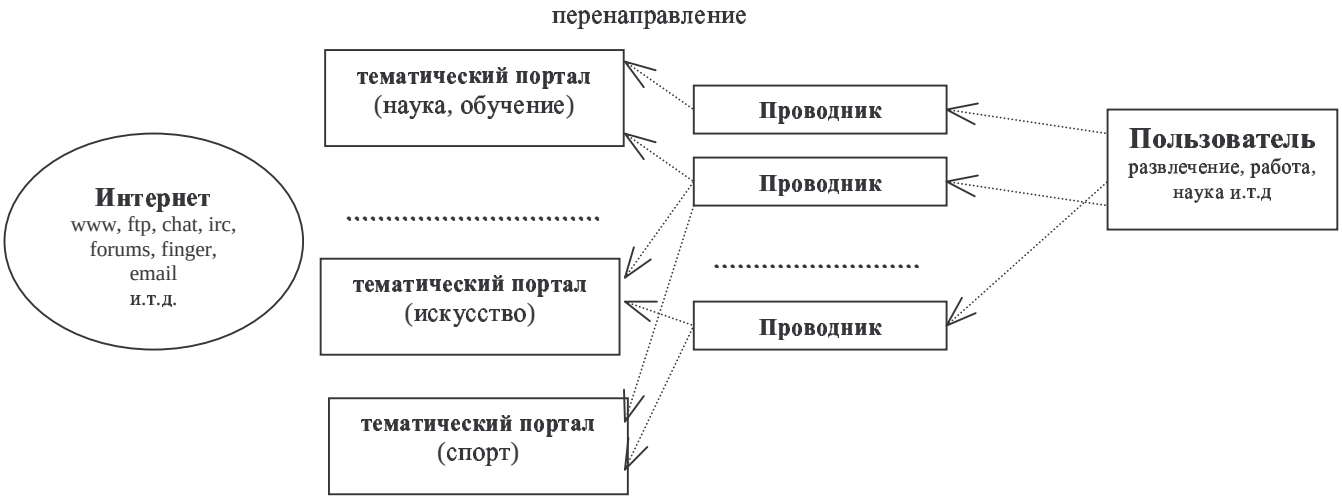

Диаграмма 8

И что мы видим , что в принципе вопрос подымаемый в данной статье позволил бы нам подойти даже к таким темам как Интернет……

Заключение

Из выше сказанного можно заметить, что целесообразно использовать промежуточное звено (к примеру в виде временных таблиц) для оптимизации скорости доступа в уже функционирующих системах на основе СУБД. Сложность заключается в том , что в каждом конкретном случае надо провадить отдельный анализ и трудно дать общий четкий алгоритм, хотя общие замечания и были приведены в статье. Можно надеятся, что в скором будущем данный вопрос полностью разрешится, созданием универсальной системы по оценке и нахождению возможных путей оптимизации производительности информационных систем.

Список цитированной литературы:

1. Тихомиров Ю.В. “Microsoft SQL Server 7.0” изд. BHV Санкт-Петербург, 1999. - 720 стр.
2. Винкоп, Стефан . “Использование Microsoft SQL Server 7.0” Специальное издание. изд. дом “Вильямс”, 1999. - 816 стр.
3. Дейв Энсор, Йен Стивенсон. “Oracle Проектирование баз данных” изд. BHV Киев, 2000.-560 стр
4. Джо Грин.“Oracle 8/8i Server.Энциклопедия пользователя”изд.ДиаСофт 2000.–576 стр.
5. Silberschatz , H. Korth , S. Sudarsnan. “Database system concepts”, McGraw-Hill Companies, Inc, 3rd ed. 1999. – 821 l.