Практическая работа № 1
Создание модели корпоративной базы данных и хранилища данных

Цель работы: изучение принципов построения хранилищ данных корпоративных информационных систем.

Задание:

1) разработать структуру корпоративной базы данных предприятия с использованием СУБД MS SQL Server;

2) написать хранимые процедуры для заполнения базы данных данными, имитирующими деятельность предприятия за несколько лет;

3) разработать структуру хранилища данных для созданной модели корпоративной базы данных.

1.1 Основные положения

Принятие любого управленческого решения в процессе управления крупным предприятием, невозможно не обладая необходимой для этого аналитической информацией, получаемой в процессе сбора, отсеивания и предварительной обработки данных с целью предоставления результирующей информации пользователям для статистического анализа и создания аналитических отчетов. Поэтому, корпоративные информационные системы, как правило, содержат приложения, предназначенные для комплексного многомерного анализа данных, их динамики, тенденций и т.п., называемые системами поддержки принятия решений. Эти приложения основываются на концепции хранилищ данных (Data warehouses).

Ральф Кимбалл (Ralph Kimball), один из авторов концепции хранилищ данных, описывал хранилище данных как "место, где люди могут получить доступ к своим данным". Он же сформулировал и основные требования к хранилищам данных:

поддержка высокой скорости получения данных из хранилища;

поддержка внутренней непротиворечивости данных;

возможность получения и сравнения так называемых срезов данных (slice and dice);

наличие удобных утилит просмотра данных в хранилище;
полнота и достоверность хранимых данных;
поддержка качественного процесса пополнения данных.
Удовлетворять всем перечисленным требованиям в рамках одного и того же продукта зачастую не удается. Поэтому для реализации хранилищ данных обычно используется несколько продуктов, одни их которых представляют собой собственно средства хранения данных, другие – средства их извлечения и просмотра, третьи – средства их пополнения и т.д.
Типичное хранилище данных, как правило, отличается от обычной реляционной базы данных. Во-первых, обычные базы данных предназначены для того, чтобы помочь пользователям выполнять повседневную работу, тогда как хранилища данных предназначены для принятия решений. Например, продажа товара и выписка счета производятся с использованием базы данных, предназначенной для обработки транзакций, а анализ динамики продаж за несколько лет, позволяющий спланировать работу с поставщиками, – с помощью хранилища данных.
Во-вторых, обычные базы данных подвержены постоянным изменениям в процессе работы пользователей, а хранилище данных относительно стабильно: данные в нем обычно обновляются согласно расписанию (например, еженедельно, ежедневно или ежечасно – в зависимости от потребностей). В идеале процесс пополнения представляет собой просто добавление новых данных за определенный период времени без изменения прежней информации, уже находящейся в хранилище.
И в-третьих, обычные базы данных чаще всего являются источником данных, попадающих в хранилище. Кроме того, хранилище может пополняться за счет внешних источников, например статистических отчетов.
Системы поддержки принятия решений обычно обладают средствами предоставления пользователю агрегатных данных для различных выборок из исходного набора в удобном для восприятия и анализа виде. Как правило, такие агрегатные функции образуют многомерный (и, следовательно, нереляционный) набор данных (нередко называемый гиперкубом или метакубом), оси которого содержат параметры, а ячейки — зависящие от них агрегатные данные - причем храниться такие данные могут и в реляционных таблицах, но в данном случае мы говорим о логической организации данных, а не о физической реализации их хранения). Вдоль каждой оси данные

могут быть организованы в виде иерархии, представляющей различные уровни их детализации. Благодаря такой модели данных пользователи могут формулировать сложные запросы, генерировать отчеты, получать подмножества данных.

Технология комплексного многомерного анализа данных получила название OLAP (On-Line Analytical Processing). OLAP – это ключевой компонент организации хранилищ данных. Концепция OLAP была описана в 1993 году Эдгаром Коддом, известным исследователем баз данных и автором реляционной модели данных. В 1995 году на основе требований, изложенных Коддом, был сформулирован так называемый тест FASMI (Fast Analysis of Shared Multidimensional Information — быстрый анализ разделяемой многомерной информации), включающий следующие требования к приложениям для многомерного анализа:

предоставление пользователю результатов анализа за приемлемое время (обычно не более 5 с), пусть даже ценой менее детального анализа;

возможность осуществления любого логического и статистического анализа, характерного для данного приложения, и его сохранения в доступном для конечного пользователя виде;

многопользовательский доступ к данным с поддержкой соответствующих механизмов блокировок и средств авторизованного доступа;

многомерное концептуальное представление данных, включая полную поддержку для иерархий и множественных иерархий (это — ключевое требование OLAP);

возможность обращаться к любой нужной информации независимо от ее объема и места хранения.

Следует отметить, что OLAP-функциональность может быть реализована различными способами, начиная с простейших средств анализа данных в офисных приложениях и заканчивая распределенными аналитическими системами, основанными на серверных продуктах. Но прежде чем говорить о различных реализациях этой функциональности, давайте рассмотрим, что же представляют собой кубы OLAP с логической точки зрения.

1.2 Порядок выполнения

1.2.1 Создание базы данных.

Атрибуты колонок таблиц SQL-Server 2000:

1. Column Name – имя колонки

2. Data Type – тип данных
3. Length – длина n для символьных видов
4. Allow Nulls – разрешение значения NULL
5. Description – описание
6. Default Value – значение по умолчанию (н-ер: NewId() для *uniqueidentifier*, GetDate() для *datetime*)
7. Precision – точность (общее количество знаков), p – для *decimal* и *numeric*
8. Scale – масштаб (количество знаков после запятой), s – для *decimal* и *numeric*
9. Identity – признак счетчика
10. Identity Seed – начальное значение счетчика
11. Identity Increment – шаг приращения счетчика
12. Is RowGuid – признак глобального идентификатора
13. Formula – для вычисляемых столбцов (н-ер: Цена * Количество)
14. Collation – сопоставление для сравнения и сортировки строк

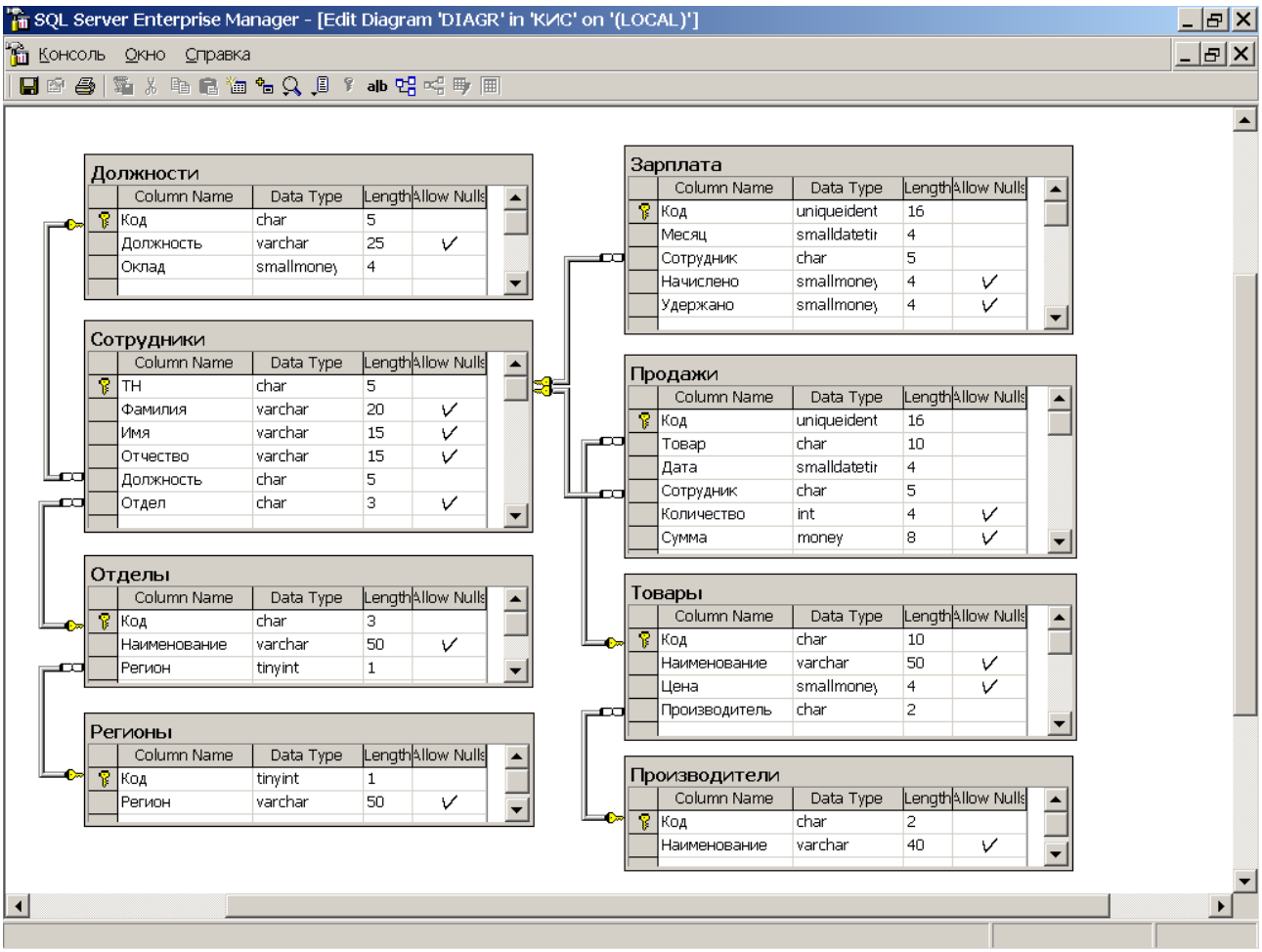
Типы данных SQL-Server 2000:

Вид	**Тип**	**Интервал значений**	**Размер**
Двоичные	*binary*(n) *varbinary*(n) *image*	до 8 000 байт до 8 000 байт до 2 Гбайт	n n
Символьные	*char*(n) *varchar*(n) *nchar*(n) *nvarchar*(n)	до 8 000 байт до 8 000 байт до 4 000 байт (Unicode) до 4 000 байт (Unicode)	n n n n
Текст	*text* *ntext*	до 2 Гбайт до 1 Гбайт (Unicode)	
Дата и время	*datetime* *smalldatetime*	01.01.1753-31.12.9999г. до 3,33 мс 01.01.1900-06.06.2079г. до 1 мин.	8 байт 4 байта
Точное представление чисел	*decimal*(p, s) *numeric*(p, s)	p <= 38, s <= p p <= 38, s <= p	При p = 2 – 2 байта, при p = 38 – 17 байт
Числа с плавающей точкой	*float*(n) *real*	$1,8*10^{308}$ $1,8*10^{308}$	n = 1 – 53 n = 24 n – число бит мантиссы

Целочисленные типы	*int* *smallint* *tinyint* *bigint*	$2*10^{10}$ 32 767 0 – 255 $9*10^{19}$	4 байта 2 байта 1 байт 8 байт
Денежные типы	*money* *smallmoney*	$9*10^{15}$, 4 знака после запятой 214 748.3648	8 байт 4 байта
Специальные	*bit* *timestamp* *uniqueidentifier* *sysname* *sql_variant* *table* *cursor*	0 или 1 (логический) отслежив.последоват.изм. записей NewID()	1 бит 8 байт 16 байт

Установка формата даты: SET DATEFORMAT dmy. Первого дня недели: SET DATEFIST 1 (1 – понедельник, 7 – воскресенье).
Преобразование типов данных:
CAST(expr AS type)
CONVERT(type, expr, [style]) style – при преобразовании в дату: 4 – dd.mm.yy, 104 – dd.mm.yyyy

Структура примера исходной базы данных корпоративной информационной системы «КИС»:



1.2.2 Создание хранимых процедур для заполнения базы данных

Ниже приведены примеры процедур и функций, заполняющие таблицу заработной платы сотрудников предприятия и случайным образом генерирующие данные о продажах товаров этими сотрудниками в течении заданного периода времени.

```
CREATE FUNCTION ПервыйДеньМесяцаПоДате (@Дата
SmallDateTime)
RETURNS SmallDateTime
AS
BEGIN
Select @Дата = @Дата - Day(@Дата) + 1
return @Дата
END

CREATE FUNCTION ПроверкаТабельногоНомера (@Код char(4))
RETURNS bit
AS
BEGIN
```

```
Declare @Yes bit
if exists(select ТабельныйНомер from Сотрудники where
ТабельныйНомер = @Код)
Set @Yes = 1
else
Set @Yes = 0
return @Yes
END

CREATE PROCEDURE dbo.НачислениеЗП (@Месяц SmallDateTime)
AS
insert into dbo.Зарплата
select NewId() as Код,
@Месяц as Месяц,
dbo.Сотрудники.ТН,
dbo.Должности.Оклад as Начислено,
dbo.Должности.Оклад*0.13 as Удержано
from
dbo.Сотрудники inner join dbo.Должности
on dbo.Сотрудники.Должность = dbo.Должности.Код
GO

CREATE PROCEDURE dbo.MassCalculate (@НачДата SmallDateTime)
AS
declare @Дата SmallDateTime
declare @i TinyInt
set @i = 1
set @Дата = Convert(SmalldateTime, '01 ' + DateName(Month,
@НачДата)
+ ' ' + Cast(DatePart(Year, @НачДата) as Char(4)), 107)
while @i < 120
begin
select @Дата = DateAdd(Month, 1, @Дата)
exec dbo.НачислениеЗП @Дата
select @i = @i + 1
end
GO

CREATE PROCEDURE dbo.Продажи10лет (@ДатаНач
SmallDateTime) AS
```

```
Declare @iМесяц Int, @iСотр Int, @iТов Int
Declare @СотрВсего Int, @Сотр Int
Declare @ТНСотр Char(5)
Declare @ТовВсего Int, @Тов Int, @КолТов Int
Declare @КодТовара Char(5)
Declare @ЦенаТовара SmallMoney
Declare @iПродаж Int, @КолПродаж Int
Declare КурсорСотр Cursor local scroll for Select ТН from Сотрудники
Declare КурсорТовары Cursor local scroll for Select Код, Цена from
Товары
Open КурсорСотр
Open КурсорТовары
Select @СотрВсего = Count(Сотрудники.ТН) From Сотрудники
Select @ТовВсего = Count(Товары.Код) From Товары
Delete from Продажи
Select @ДатаНач = ПервыйДеньМесяцаПоДате(@ДатаНач)
Set @iМесяц = 1
While @iМесяц <= 120
begin
Set @iСотр = 1
While @iСотр <= @СотрВсего
begin
Select @Сотр = Round(Rand() * @СотрВсего, 0)
Fetch Absolute @Сотр from КурсорСотр Into @ТНСотр
Set @iПродаж = 1
Select @КолПродаж = Floor(Rand() * 50)
While @iПродаж <= @КолПродаж
begin
Select @КолТов = Floor(Rand() * 5)
Select @Тов = Round(Rand() * @ТовВсего, 0)
Fetch Absolute @Тов from КурсорТовары Into
@КодТовара,@ЦенаТовара
Insert Into Продажи (Код,Дата,Сотрудник,Товар, Количество,Сумма)
Values
(NewId(),@ДатаНач,@ТНСотр,@КодТовара,@КолТов,@ЦенаТовара*
@КолТов)
Select @iПродаж = @iПродаж + 1
end
Set @iСотр = @iСотр + 1
end
```

```
Select @ДатаНач = DateAdd(Month, -1, @ДатаНач)
Select @iМесяц = @iМесяц + 1
end
Close КурсорСотр
Close КурсорТовары
Deallocate КурсорСотр
Deallocate КурсорТовары
GO
```

1.2.3 Создание хранилища данных

Структура хранилища данных «КИС_ХД» для приведенной выше базы данных: