

# Введение в моделирование данных, базы данных и SQL

Лекции 8,9

# Представления

# Представления

- Представление – это «виртуальная таблица», определенная в терминах содержания других таблиц и представлений.

**CREATE VIEW** <ИМЯ>

**AS**

<запрос>

# MS SQL – create view

## CREATE VIEW

[ schema\_name . ] view\_name

[ (column [ ,...n ] ) ]

[...]

**AS** select\_statement [WITH CHECK OPTION]

- если *column* не указан
  - столбцам представления назначаются имена столбцов в SELECT
  - *лучше указывать*
- *column* требуется
  - если в SELECT не назначены имена столбцам, формируемым на основе арифметического выражения (тогда всем предшествующим)
  - если столбцу представления требуется назначить имя, отличное от имени столбца SELECT.
- необходимы соответствующие разрешения доступа к объектам, указанным в предложении SELECT.

# Сценарии использования представлений

- Концентрация внимания на определенных данных
  - ненужные или конфиденциальные данные выводятся из рассмотрения с помощью представлений.
- Упрощение обработки данных
  - часто используемые запросы определяются как представления, не нужно расписывать каждый раз
  - дополнительной условия можно указать для представления
- Обеспечение обратной совместимости
  - изменилась схема данных, старые таблицы реализуются представлениями
- Фильтрация, видимость данных
  - пользователям с разными интересами и разной квалификацией – разные представления

# Пример: определения представления

- **CanDrink(drinker, beer)** представление  
“содержащее” пары (персона, пиво) такие, что  
персона посещает хотя бы один бар,  
продающий пиво:

```
CREATE VIEW CanDrink AS  
  SELECT drinker, beer  
  FROM Frequents, Sells  
  WHERE Frequents.bar = Sells.bar;
```

Sells(bar, beer, price)

Frequents(bar, drinker)

# Пример: Обращение к представлению

- Можно обращаться к представлениям, как к реальным таблицам
- Пример:

```
SELECT beer FROM CanDrink  
WHERE drinker = 'Алекс';
```

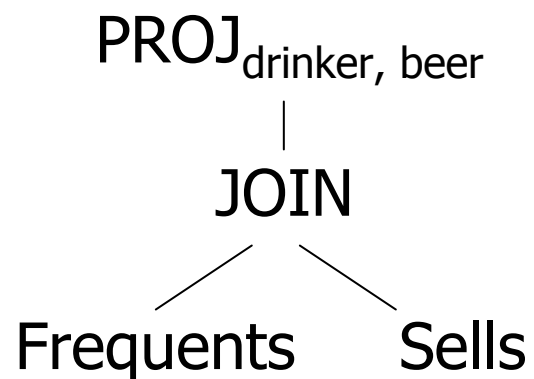
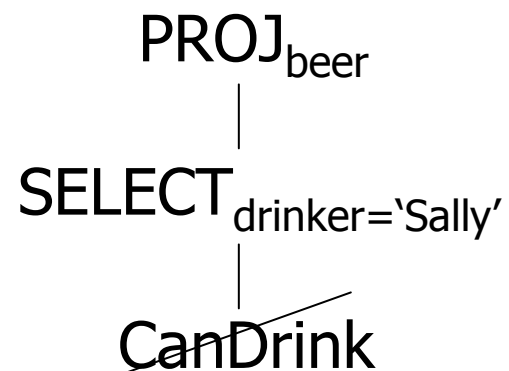
- имеются ограниченные возможности **по изменению** данных представлений

# Что происходит в случае использования представления?

- СУБД начинает интерпретировать запрос как будто это реальная таблица
  - Обычно СУБД транслирует запрос в промежуточный язык, напоминающий язык реляционной алгебры – суперпозиция операций.
- Запросы, определяющие представления, используемые в исходном запросе, **заменяются** их алгебраическими эквивалентами, «встраиваются» в дерево выражения запроса
  - затем СУБД оптимизирует запрос, преобразовывая алгебраическое выражение, чтобы повысить скорость его исполнения



# Пример: «разыменование» представления



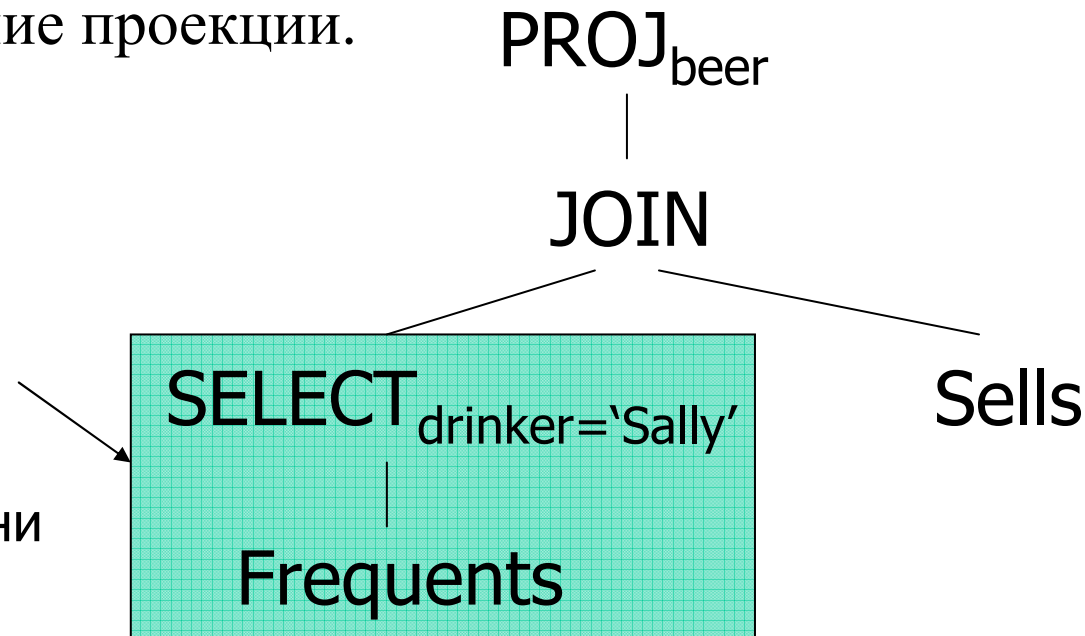
# Пример: Оптимизация

- Ключевые оптимизации:

1. Спустить отбор кортежей ниже по дереву

2. Удалить лишние проекции.

большинство  
кортежей  
устраняются из  
Frequents  
до выполнения  
емкого по времени  
соединения



Sells(bar, beer, price)

Frequents(bar, drinker)

## Другой пример

Person(name, city)

Purchase(buyer, seller, product, store)

Product(name, maker, category)

```
CREATE VIEW Moscow-view AS

SELECT buyer, seller, product, store
FROM   Person, Purchase
WHERE  Person.city = "Москва" AND
       Person.name = Purchase.buyer
```

Получаем виртуальную таблицу:

Moscow-view(buyer, seller, product, store)

# «Разыменование» представления

```
SELECT name, Moscow-view.store  
FROM Moscow-view, Product  
WHERE Moscow-view.product = Product.name AND  
Product.category = “обувь”
```



```
SELECT name, Purchase.store  
FROM Person, Purchase, Product  
WHERE Person.city = “Москва” AND  
Person.name = Purchase.buyer AND  
Purchase.product = Product.name AND  
Product.category = “обувь”
```

# Изменение данных представления

Как вставить кортеж в таблицу, которой нет?

`Employee(ssn, name, department, project, salary)`

```
CREATE VIEW Developers AS  
SELECT name, project  
FROM Employee  
WHERE department = "СМО"
```

Если выполнить вставку:

```
INSERT INTO Developers  
VALUES("Алекс", "Аналитика")
```

Она становится:

```
INSERT INTO Employee  
VALUES(NULL, "Алекс", NULL, "Аналитика", NULL)
```

изменение имеет смысл **как изменение реальной** таблицы

# Неизменяемые представления

```
CREATE VIEW Moscow-view AS  
  SELECT seller, product, store  
  FROM   Person, Purchase  
  WHERE  Person.city = "Москва" AND  
         Person.name = Purchase.buyer
```

Как добавить кортеж

("Алекс", "Туфли модель 12345", "Модельная обувь")?

В начале добавить "Алекс" к Person.

Одну или несколько копий?

- Чтобы сделать представление обновляемым следует использовать триггеры INSTEAD OF.
- Триггер INSTEAD OF выполняется вместо инструкции модификации данных, для которой он определен.

# MS SQL - обновляемые представления

- СУБД должно однозначно проследить изменения от определения представления до одной базовой таблицы
  - изменяемые столбцы представления должны *непосредственно ссылаться* на столбцы базовой таблицы, не должны сформироваться, например, вычисляться
  - любые изменения (UPDATE, INSERT и DELETE) должны ссылаться на столбцы только одной базовой таблицы, скрытые представлением столбцы должны иметь значения по умолчанию.
  - GROUP BY, HAVING и DISTINCT не должны влиять на изменяемые столбцы, ими пользоваться.

## MS SQL – примеры обновляемый

- Представление возвращает значения из *двух таблиц*, но инструкция UPDATE выполнена *успешно*, потому что изменялись столбцы *только одной* из базовых таблиц
- инструкция INSERT была выполнена успешно, поскольку были *указаны столбцы только одной* базовой таблицы, а другие столбцы в базовой таблице имеют *значения по умолчанию* или допускают значений NULL
- Для успешного удаления строки представления это удаление *должно удовлетворять всем* FOREIGN KEY ограничениям связанных таблиц



# MS SQL - материализованные представления

- можно повысить производительность использования представления, создав для него уникальный кластеризованный индекс.
  - результирующий набор будет храниться в базе данных подобно базовой таблице.
- изменения данных базовых таблицах
  - отражаются в данных, хранимых в индексированном представлении.
- при создании индекса для представления, что оптимизатор СУБД будет использовать этот индекс в запросах, где представление не упоминается непосредственно.
  - создание индекса для представления может привести к повышению эффективности уже имеющихся запросов.

# Триггеры

# Триггеры: мотивация

- условия на атрибуты и кортежи предоставляют **ограниченные возможности**
- assert-ы более общие, однако их **сложно эффективно реализовать**
  - СУБД должна иметь хороший интеллект, чтобы избегать проверок утверждений, где возникновение нарушений невозможно

# Решение: триггеры

- Триггер позволяет пользователю указать, когда проверка должна выполняться
- Аналогично assert-у, триггер позволяет указать условие общего вида, так же можно выполнить любую последовательность SQL модификаций БД
- можно создавать множественные триггеры на одну таблицу
  - например, таблица может иметь два триггера на INSERT, три на UPDATE и один на DELETE для более четкого разделения логики в триггерах.
- триггеры могут вызываться рекурсивно

# Правила

## «Событие – Условие – Действие»

- Другое наименование “триггера”  
– *ЕСА правило* (Event-Condition-Action)
- **Событие** : обычно вид модификации БД, например, “INSERT ON Sells”
- **Условие** : любое логическое SQL выражение
- **Действие** : любые SQL операторы

# Пример: триггер

- Имеется множество деталей в употреблении триггеров
- рассмотрим некоторый «сценарий».
  - Вместо использование ограничений внешнего ключа и **отклонения** вставок в Sells(bar, beer, price) кортежей с неизвестным сортом пива, триггер может **добавлять** такой сорт к Beers(name, manf) с неизвестным (NULL) производителем

# Пример: определение триггера

**CREATE TRIGGER BeerTrig**

**AFTER INSERT ON Sells**

← событие

**REFERENCING NEW ROW AS NewTuple**

**FOR EACH ROW**

**WHEN** (NewTuple.beer **NOT IN**  
(**SELECT** name **FROM** Beers))

← условие

**INSERT INTO** Beers(name)  
**VALUES**(NewTuple.beer);

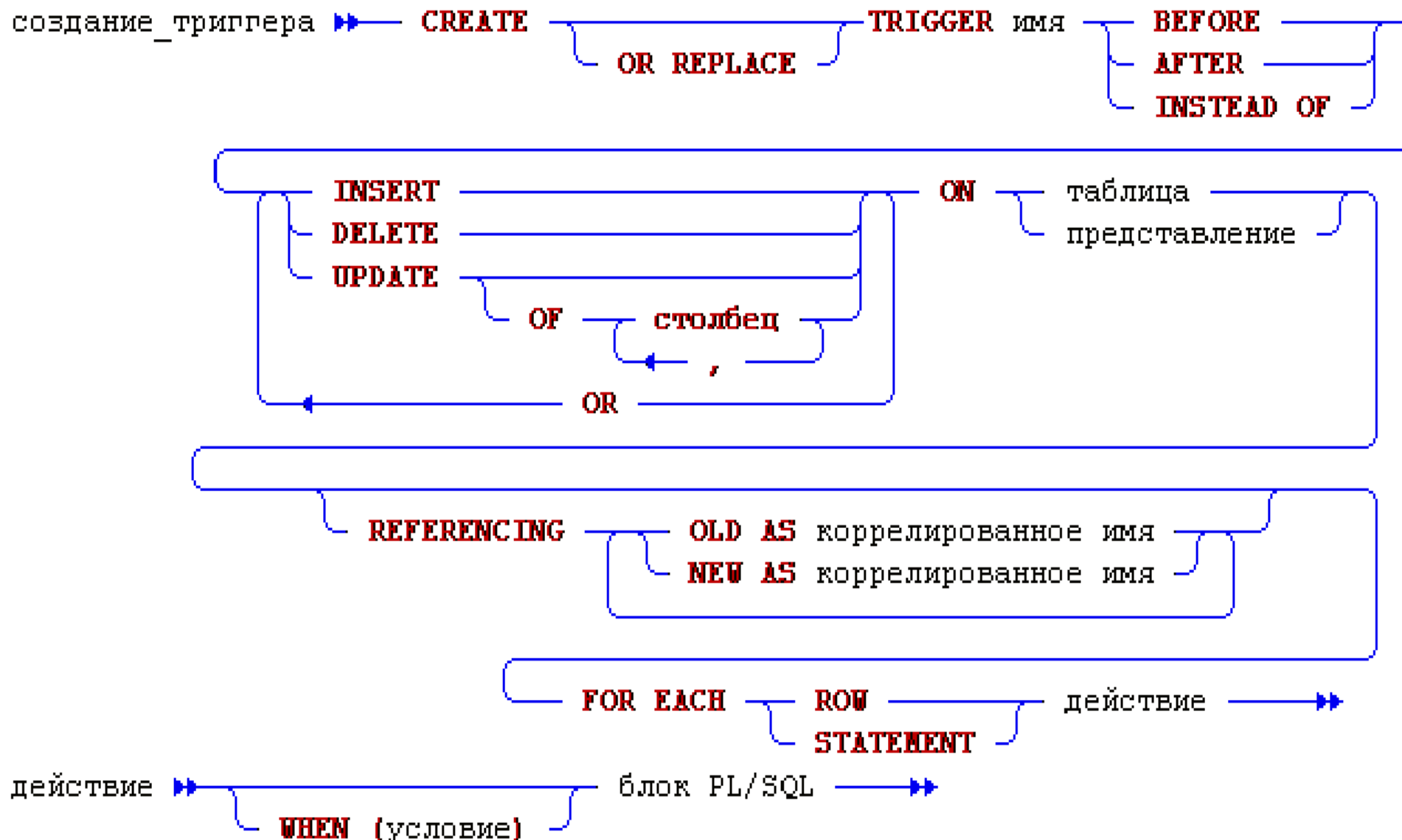
← действие

# CREATE TRIGGER

- **CREATE TRIGGER** <ИМЯ>
  - *Событие*
    - {AFTER | BEFORE | INSTEAD OF}  
{INSERT | DELETE | UPDATE}
  - *Условие*
    - WHEN
  - *Действие*
    - BEGIN ... END
- **CREATE OR REPLACE TRIGGER** <ИМЯ>
  - когда, возможно, имеется одноименный триггер и мы желаем его модифицировать
- **ALTER TRIGGER** <ИМЯ>
- **DROP TRIGGER** <ИМЯ>



# Оператор CREATE TRIGGER (Oracle)



# Условия

- AFTER или BEFORE
  - INSTEAD OF если отношение – это представление
    - **перехватывает** команды SQL, направленные на обновление представления, и выполняет соответствующее действие
    - отличный способ обеспечения модификации любых **представлений**: иметь триггеры, транслирующие обращения в модификации соответствующих базовых таблиц
- INSERT или DELETE или UPDATE.
  - может быть UPDATE . . . ON на **некоторый атрибут**

# REFERENCING - 1

- REFERENCING

{NEW | OLD} {ROW | TABLE} AS <имя>...

- определяет коррелированные имена, по которым действие триггера может обращаться
  - к столбцам изменяемых строк
  - к изменяемой таблице
- OLD - имя, представляющее **изменяемую** строку/таблицу до изменения
- NEW - имя, представляющее **изменяемую** строку/таблицу после изменения
  - MS SQL: **inserted, deleted**

# REFERENCING - 2

- INSERT подразумевает
  - новый кортеж (ориентированные на кортежи) или
  - совокупность новых кортежей (ориентированные на SQL-операторы).
- DELETE подразумевает
  - существующий кортеж или
  - таблицу
- UPDATE подразумевает
  - и то и другое

# Область действия триггера

- Триггеры ориентируются
  - либо на кортежи (ROW)
  - либо на SQL-операторы (STATEMENT)
- **FOR EACH ROW**
  - указывает ориентацию на кортежи
  - если опущено, то ориентация на SQL-операторы
- Триггеры, ориентированные на кортежи,
  - выполняются для **каждого** модифицируемого кортежа
- Триггеры, ориентированные на SQL-операторы - **FOR EACH STATEMENT**,
  - выполняются для **SQL оператора**, вне зависимости сколько кортежей он модифицирует.

# Условия

- Приемлемо любое логическое SQL-выражение
- Проверяется до или после События, в зависимости от того, что указано для события - **BEFORE** или **AFTER**
- Можно ссылаться на новой/существующий кортеж или совокупность кортежей с помощью имени указанного в **REFERENCING** предложении

# Действие

- В действии можно указать несколько SQL операторов
  - заключив их BEGIN . . . END
- Но в этом случае запросы здесь имеют мало смысла, то есть это только модификации БД

# Пример

- Необходимо формировать список баров поднимающих цену пива более чем на 10
- Воспользуемся отношением

`Sells(bar, beer, price)`

и унарным отношением

`RipoffBars(bar),`

созданным для этой цели



# The Trigger

```
CREATE TRIGGER PriceTrig
```

```
AFTER UPDATE OF price ON Sells
```

```
REFERENCING
```

```
OLD ROW as old
```

```
NEW ROW as new
```

```
FOR EACH ROW
```

```
WHEN(new.price > old.price + 10)
```

```
INSERT INTO RipoffBars
```

```
VALUES(new.bar);
```

Событие –  
только изменения  
цены

Введем наименования  
для новых и  
существующих  
кортежей

Для каждого  
изменения цены

Условие:  
при  
цене > 10

При таком изменении  
цены добавлять бар  
в RipoffBars


# Триггеры для представлений

- В общем случае - невозможно модифицировать представления
- но **INSTEAD OF** триггер позволяет интерпретировать изменения представлений надлежащим образом
- Пример: Создадим представление **Synergy**(drinker, beer, bar), включающее кортежи такие, что бар продает сорт пива, любитель посещает бар и любит это пиво

# Пример: представление

```
CREATE VIEW Synergy AS
```

Выберем значения  
атрибутов



```
SELECT Likes.drinker, Likes.beer, Sells.bar
```

```
FROM Likes, Sells, Frequents
```


```
WHERE Likes.drinker = Frequents.drinker
```

```
AND Likes.beer = Sells.beer
```

```
AND Sells.bar = Frequents.bar;
```

Натуральное соединение

Likes(drinker, beer), Sells(bar, beer, price) и  
Frequents(drinker, bar)



# Интерпретация вставки в представление

- не можем вставлять в Synergy - представление
- используем **INSTEAD OF** триггер, чтобы кортеж (drinker, beer, bar) отобразить в **три** вставки кортежей отношений Likes, Sells и Frequents.
  - Sells.price будет устанавливаться в NULL.

# Триггер для представления

```
CREATE TRIGGER ViewTrig
  INSTEAD OF INSERT ON Synergy
  REFERENCING NEW ROW AS n
  FOR EACH ROW
  BEGIN
    INSERT INTO LIKES VALUES(n.drinker, n.beer);
    INSERT INTO SELLS(bar, beer) VALUES(n.bar, n.beer);
    INSERT INTO FREQUENTS VALUES(n.drinker, n.bar);
  END;
```

# Microsoft SQL Server - документация

На сайтах:

- <http://msdn.microsoft.com/ru-ru/library/>
- <http://technet.microsoft.com/ru-ru/library/>

по SQL Server 2005

- Документация - [ms203721\(v=sql.90\).aspx](#)
- SQL Server Database Engine - [ms187875\(v=sql.90\).aspx](#)
- Построение проектов БД - [ms190668\(v=sql.90\).aspx](#)

Проектирование и создание баз данных

- Таблицы - [ms189104\(v=sql.90\).aspx](#)
- Представления - [ms190706\(v=sql.90\).aspx](#)
- Триггеры DDL - [ms190989\(v=sql.90\).aspx](#)
- Триггеры DML - [ms191524\(v=sql.90\).aspx](#)
- Таблицы inserted и deleted - [ms191300.aspx](#)
- Различия между триггерами DDL и DML - [ms189599\(v=sql.90\).aspx](#)
- Хранимые процедуры - [ms190782\(v=sql.90\).aspx](#)
- Индексы - [ms189271\(v=sql.90\).aspx](#)

Книга

- [http://download.microsoft.com/download/3/D/9/3D956EA9-F211-4E66-929D-3431F8617127/SqlServer2K5\\_BOL\\_Jan2009\\_ru-RU.msi](http://download.microsoft.com/download/3/D/9/3D956EA9-F211-4E66-929D-3431F8617127/SqlServer2K5_BOL_Jan2009_ru-RU.msi)

# **Безопасность и санкционирование доступа**

# Аутентификация и авторизация

- Доступ к данным и возможностям СУБД **персонифицирован**.
  - Для каждого действия, выполняемого в системе, определено, каким пользователем оно выполняется.
  - Процедура аутентификации (authentication) состоит в подтверждении подлинности пользователя при установлении соединения с базой данных.
  - Подтверждение подлинности обеспечивается «**паролем**» (password)
- Связь/соединение с базой данных устанавливается приложением при помощи SQL-оператора
  - **CONNECT TO** {[имя\_сервера.]имя\_базы\_данных} [**AS** имя\_соединения] **USER** [**логин**[.пароль] | **\$integrated**]



# Виды аутентификации

- внутренние, внешние методы аутентификации:
  - **внешние** - при соединении с базой данных аутентификация производится по тому же имени и паролю, которые используются для его аутентификации в **операционной системе или сетевых службах**.
  - **внутренние** - данные о пользователях являются объектами базы данных, имена и пароли хранятся в **СУБД**
- **{CREATE|ALTER|DROP} USER**
- **sp\_addlogin** [ @loginame = ] 'имя'  
[ , [ @passwd = ] 'пароль' ]  
[ , [ @defdb = ] 'база данных' ]  
**sp\_droplogin** [ @loginame = ] 'имя'
  - **ms189751(v=sql.90).aspx**
  - **ms182669(v=sql.90).aspx**
  - **ms173768(v=sql.90).aspx**

# Авторизации, привилегии

- Привилегия
  - некоторый поддерживаемый системой признак, который определяет, **разрешение** выполнения какой-либо конкретной операции (общей или относящейся к конкретному **объекту**), которая будет выполняться от имени какого-то **субъекта**.
- Каждое действие, выполняемое пользователем в базе данных, сопровождается процедурой **авторизации (authorization)**,
  - проверка того, может ли данный пользователь выполнять запрошенное действие над данным объектом,
  - т.е., **имеет ли** данный пользователь данную **привилегию**.
- По результатам авторизации запрос на доступ выполняется или отклоняется.

# Авторизация доступа

- Авторизация (защита) доступа к данным
  - к отношениям и их полям
  - **предопределенный** набор привилегий
    - для любого отношения БД и любого его атрибута
    - для выполнения любого действия пользователь должен обладать соответствующей привилегией
    - возможные действия описываются фиксированным стандартным набором привилегий
  - **транзакция → идентификатор пользователя**
    - неявно связывается
    - от имени которого она выполняется
  - средствами языка SQL

# SQL GRANT / REVOKE

- Создание нового отношения
  - все привилегии принадлежат только пользователю-создателю отношения
- **SQL GRANT**
  - привилегия передачи всех или части привилегий другому пользователю
  - привилегия на передачу привилегий
  - [ms187965\(v=sql.90\).aspx](#)
- **SQL REVOKE**
  - привилегия изъятия всех или части привилегий у пользователя, которому они ранее были переданы
  - [ms187728\(v=sql.90\).aspx](#)
- Проверка полномочности доступа к данным
  - происходит на основе полномочий, существующих во время компиляции SQL-оператора

# SQL GRANT

- Привилегии предоставляются с помощью предложения **GRANT** (предоставить)
  - **GRANT** привилегии **ON** объект **TO** пользователи
- **GRANT** select, update (manf) **ON** Beers **TO** user3;

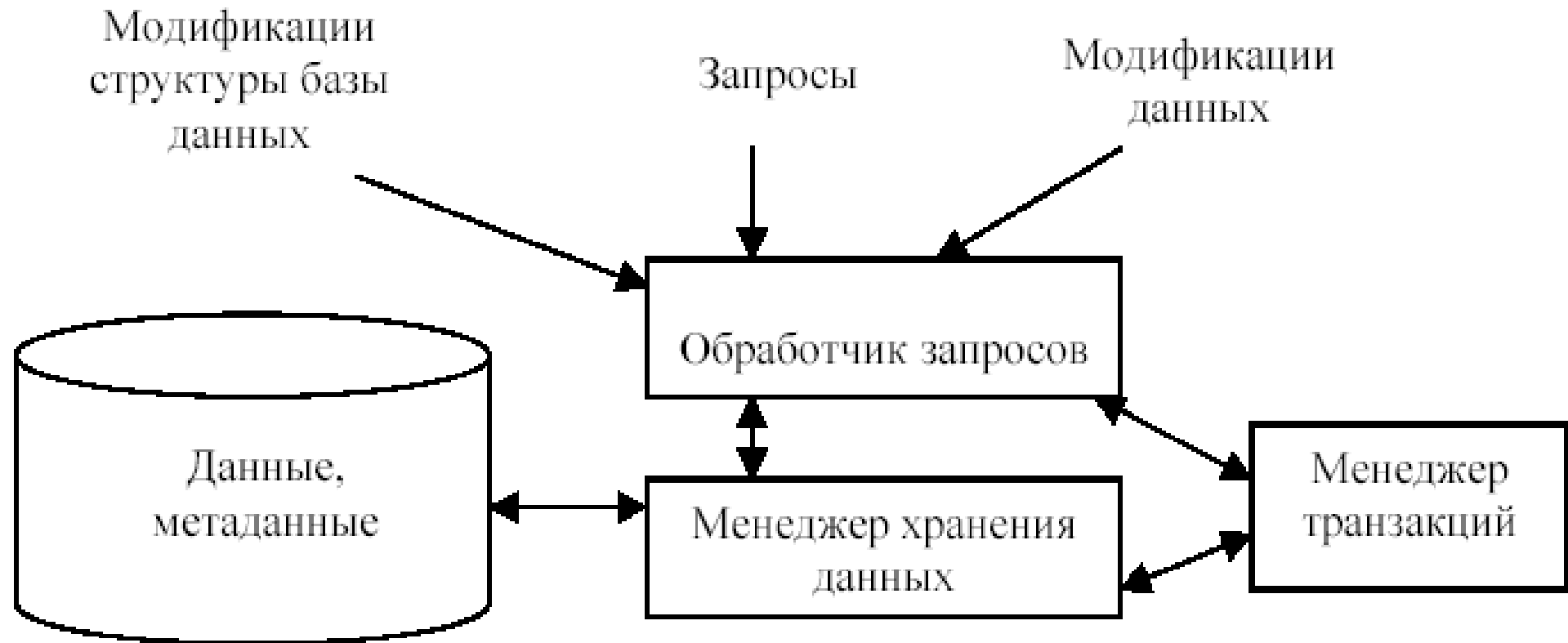
Тип объекта	Возможные команды
Таблица	SELECT, UPDATE, DELETE, INSERT, REFERENCE
Столбец	SELECT, UPDATE
Представление	SELECT, UPDATE, DELETE, INSERT
Хранимая процедура	EXECUTE

# SQL REVOKE

- в последствии отменить все или некоторые из этих привилегий. Отмена осуществляется с помощью предложения REVOKE (отменить),
  - **REVOKE** привилегии **ON** объект **FROM** пользователи;
  - **REVOKE** update (manf) **ON** Beers **FROM** user3;

# *Прикладное программирование*

# Укрупненная архитектура СУБД в общем виде



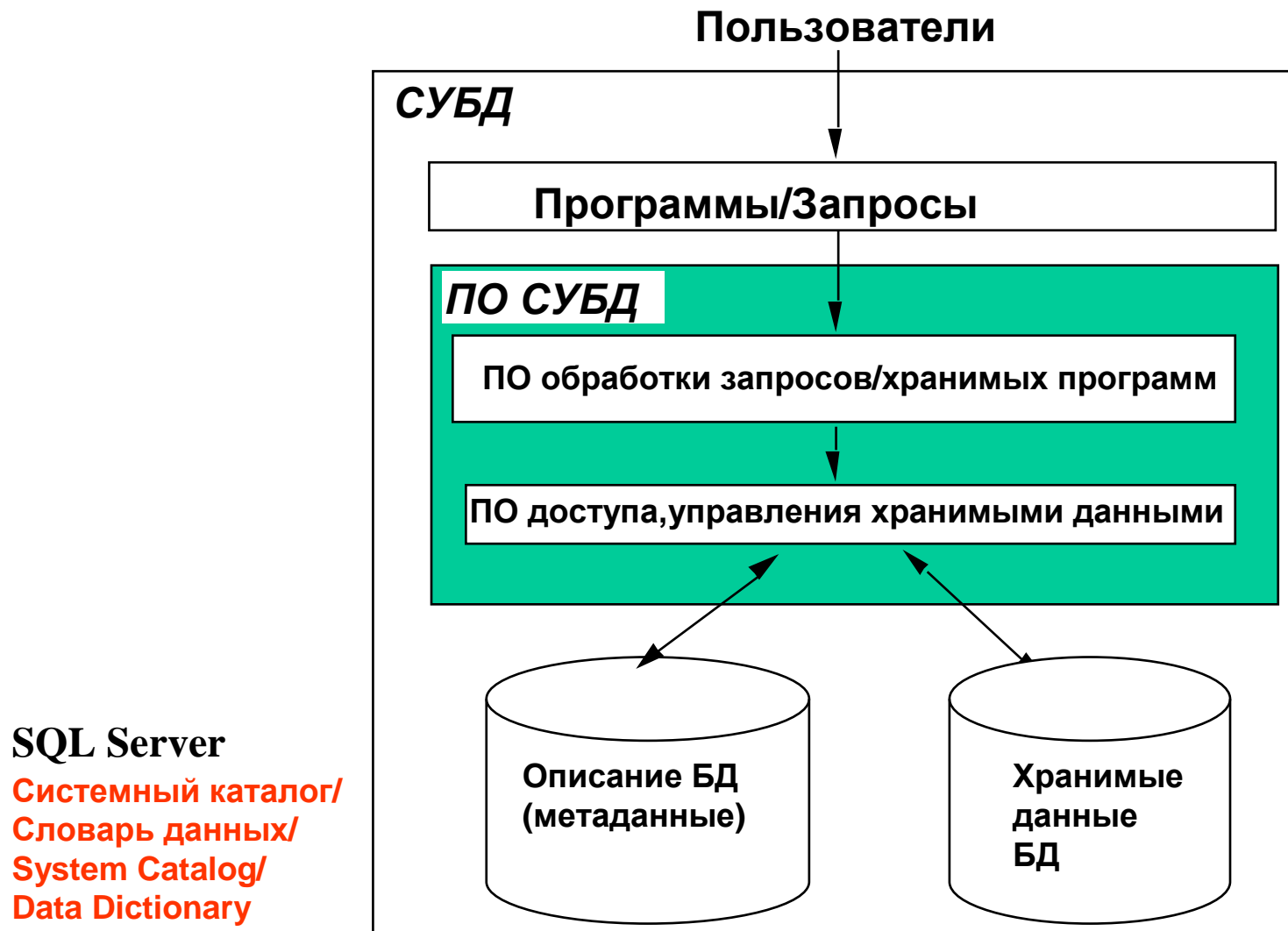


# Составные части

## РМД

- **Структурная (Structure)**
  - Как хранятся?
    - Структура данных подсказывает, как данные **организованы**, как они хранения данных в БД.
    - РМД представляет метод, который должен использоваться для хранения всех данных в БД.
- **Целостная (Integrity)**
  - Какие **ограничения** накладываются данные?
    - Представляет способ, с помощью которого корректность и качество данных БД может быть обеспечено.
    - РМД представляет метод, который должен использоваться для обеспечения *корректности* данных в БД.
- **Манипуляционная (Manipulation)**
  - Какие операции используются для **манипулирования** данными?
    - Представляет возможности по доступу к данным хранимым в БД.
    - РМД представляет метод, который должен использоваться для извлечения всех данных из БД.

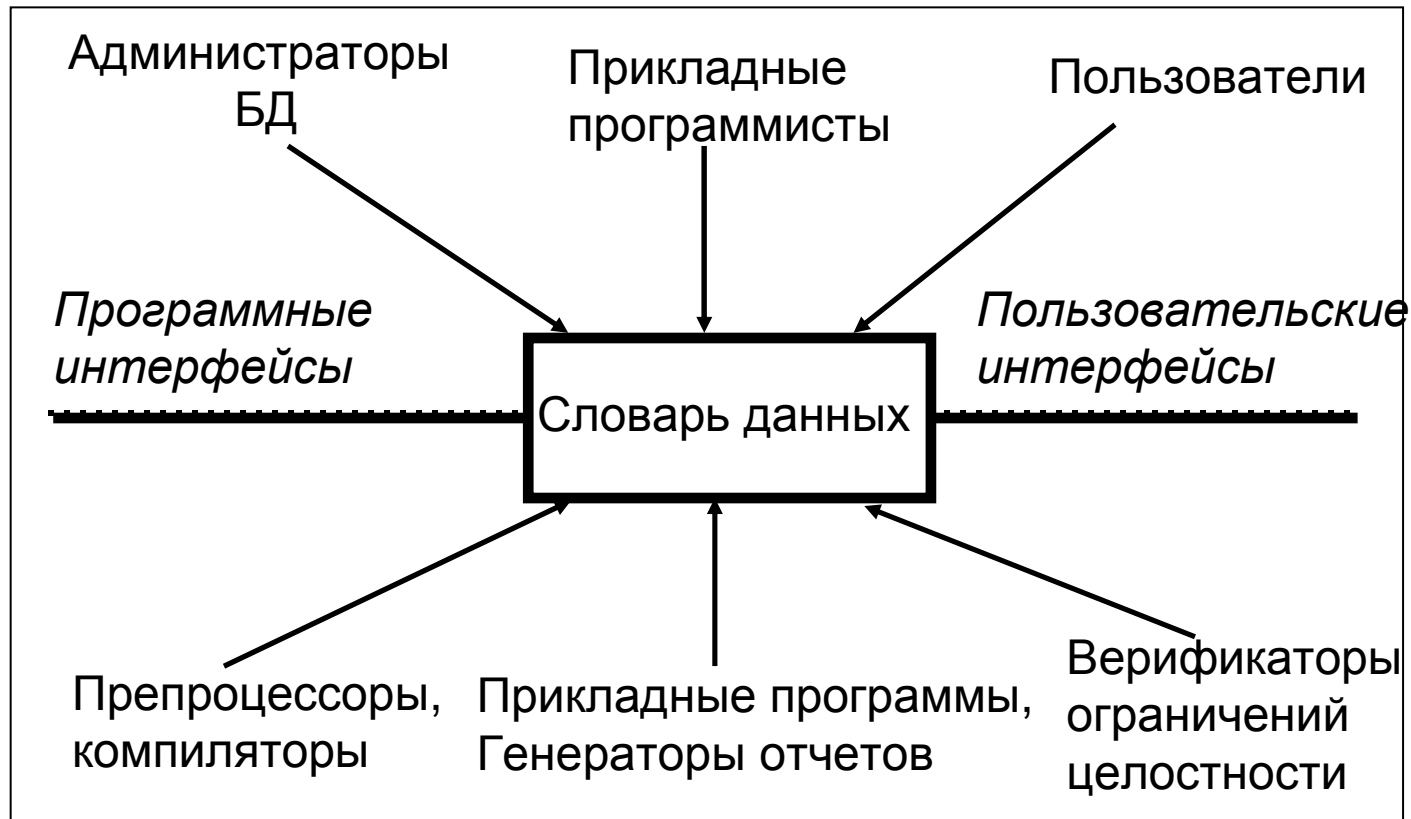
# Логическая структура СУБД



# Роль системного каталога

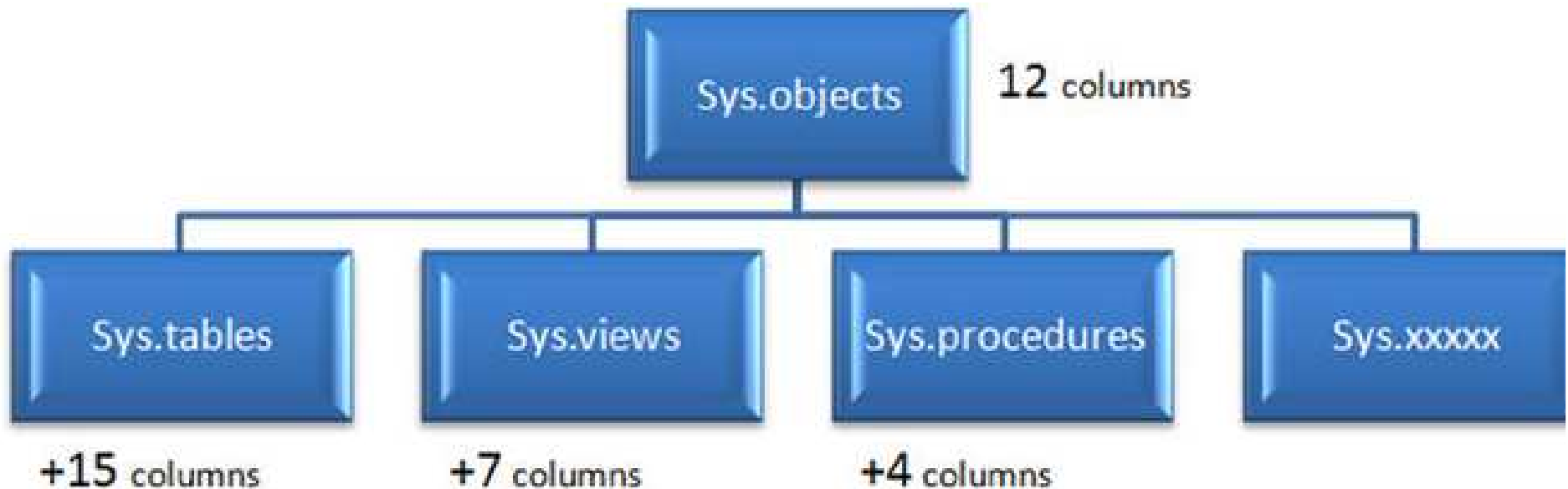
- Набор таблиц и представлений, используемый СУБД в качестве справочного руководства по работе с данными, хранящимся в базе данных, ее файлах.
- Каждый пользователь применяет, хранящиеся **Словаре данных (Data Dictionary)**.
  - неявно - выполняя запросы
  - явно - обращаясь к словарю данных, к его таблицам, вьюшкам
- Словарь данных хранит:
  - имена тех, кому разрешен доступ к БД,
    - их роли, привилегии,
  - описания доменов, таблиц, ограничений целостности, представлений,
  - триггеров, хранимых процедур/функций,
  - индексов, кластеров и т. д.

# Словарь данных



# Метаданные

- <http://www.mssqltips.com/sqlservertip/1934/understanding-catalog-views-in-sql-server-2005-and-2008> (**Understanding Catalog Views** in SQL Server)



# Microsoft SQL Server - справочник

На сайтах:

- <http://msdn.microsoft.com/ru-ru/library/>
- <http://technet.microsoft.com/ru-ru/library/>

по SQL Server 2005

- Документация - [ms203721\(v=sql.90\).aspx](#)
- SQL Server Database Engine - [ms187875\(v=sql.90\).aspx](#)
- Построение проектов БД - [ms190668\(v=sql.90\).aspx](#)

Метаданные

- Системные таблицы - [ms179932\(v=sql.90\).aspx](#)
  - Таблицы плана обслуживания БД - [ms190275\(v=sql.90\).aspx](#)
- Системные представления - [ms177862\(v=sql.90\).aspx](#)
  - Представления каталога - [ms174365\(v=sql.90\).aspx](#)
  - Представления совместимости - [ms187376\(v=sql.90\).aspx](#)
- Запрос к системному каталогу - [ms189082\(v=sql.90\).aspx](#)
  - Настройка видимости метаданных - [ms187113\(v=sql.90\).aspx](#)
  - Часто задаваемые вопросы о запросах к системному каталогу сервера SQL Server - [ms345522\(v=sql.90\).aspx](#)

Книга

- [http://download.microsoft.com/download/3/D/9/3D956EA9-F211-4E66-929D-3431F8617127/SqlServer2K5\\_BOL\\_Jan2009\\_ru-RU.msi](http://download.microsoft.com/download/3/D/9/3D956EA9-F211-4E66-929D-3431F8617127/SqlServer2K5_BOL_Jan2009_ru-RU.msi)

# Встроенный SQL

- встраивание операторов SQL в языки программирования
- проблема
  - SQL операции со **множествами**
  - языки программирования **скалярные** операции
- специальные понятия, операторы
  - поддерживающие встраивание операторов SQL
  - покортежный доступ к результату запроса к БД
  - **курсор** оператора выборки
  - параметризация операторов SQL значениями переменных программы

# Динамический SQL

- интерактивные SQL-системы
- откомпилировать и выполнить любой оператор SQL во время выполнения транзакции
- **PREPARE**
  - динамическая компиляция операторов SQL
- **DECLARE, OPEN, FETCH, CLOSE**
  - аппарат курсоров для оператора выборки
- **EXECUTE**
  - выполнение других операторов



# Курсор

- **DECLARE** имя\_курсора **CURSOR**  
**FOR** подзапрос
- **OPEN** имя\_курсора
- **FETCH**        имя\_курсора  
**INTO**   переменная ,переменная} ...
- **CLOSE** имя\_курсора
- **DROP CURSOR** имя\_курсора

# Bce

