9_SQL Server - основа информационной системы предприятия или организации.

Технология "клиент-сервер"

- "Клиент-сервер" - модель взаимодействия компьютеров в сети: компьютер, **управляющий** ресурсом, называют **сервером** ресурса, а компьютер, использующий ресурс - **клиентом**. В технологии "клиент-сервер" часть функций реализована в программе-клиенте, другая - в программе-сервере.
- В модели «Файловый сервер» протокол обмена представляет собой набор низкоуровневых вызовов, обеспечивающих приложению доступ к **файловой** системе на файл-сервере.
- В модели "клиент-сервер" общезначимая часть логики оформлена как набор **хранимых процедур и триггеров** и функционирует на сервере БД.
- Доступ к информационным ресурсам обеспечивается операторами языка SQL. Язык хранимых процедур представляет **процедурное расширение** языка запросов SQL и отличается для конкретной СУБД.
- SQL сервер позволяет работать большому числу пользователей, каждый из которых запускает свои запросы в параллельных сессиях.

Серверы баз данных

Термин «сервер БД» используется для обозначения всей СУБД, основанной на архитектуре клиент-сервер, включая серверную и клиентскую часть. Обычно одна БД целиком хранится в одном узле сети и поддерживается сервером в сервере БД. Доступ к БД из прикладной программы осуществляется с использованием клиентской части системы.

В качестве основного интерфейса между клиентскими и серверными частями выступает язык SQL. Запросы на языке SQL до своего реального выполнения могут подвергаться компиляции. Компиляция запросов может производиться на стадии предкомпиляции прикладной программы, написанной на обычном традиционном языке программирования с включением предложений SQL, или в процессе выполнения транзакций с использованием инструкции языка SQL. С точки зрения пользователя процесс компиляции приводит к следующим результатам: для каждого предложения на SQL образуется программа в машинных кодах, вызовы которой помещаются в текст исходной прикладной программы, однако в действительности процесс компиляции запроса намного более сложен из-за наличия сложных сетевых взаимодействий, которые требуются при реальном выполнении транзакции. Серверы БД, интерфейс которых основан на языке SQL, обладают преимуществами и недостатками.

Преимущества: стандартный открытый интерфейс, т. е. клиентская часть любой ориентированной СУБД может работать с любым SQL-сервером независимо от того, когда компания его разработала.

Недостатки. При высоком уровне интерфейса между клиентской и серверной частями системы со стороны клиента работает слишком мало программ СУБД. Если клиентский компонент обладает достаточной мощностью, то часто возникает необходимость возложить на него большие функции управления БД и разгрузить сервер, который в этом случае является узким местом этой системы. Одним из корректных направлений СУБД является гибкая конфигурированная система, при которой распределяются функции между клиентской и пользовательской системами.

Архитектура сервера

- использует для хранения БД набор файлов операционной системы, при этом для каждой из них создается собственный файл.
- Первичный файл данных (**Primary data file**)— отправная точка БД. Всякая БД имеет только один первичный файл данных. Рекомендуемое расширение — .mdf.
- Вторичные файлы данных (**Secondary data files**) являются необязательными и могут хранить все данные и объекты, не вошедшие в первичный файл данных. Некоторые БД могут вообще не иметь вторичных файлов данных, а другие иметь множество таких файлов. Рекомендуемое расширение — .ndf.
- Файлы журнала (**Log files**) - фиксируется вся информация о транзакциях, которая используется для восстановления БД. Каждая БД имеет, по крайней мере, один файл журнала. Рекомендуемое расширение — .ldf.
- При создании БД все входящие в ее состав файлы "обнуляются" (заполняются нулями), чтобы стереть все данные, которые остались на диске от ранее удаленных файлов. Приводит к увеличению продолжительности создания БД, но избавляет Windows NT от необходимости очистки файлов при записи в них данных (поскольку они уже "обнулены") во время нормальной работы с БД, что повышает производительность системы.
- Данные таблиц хранятся в наборе страниц данных. Каждая страница имеет заголовок, который содержит такую системную информацию, как идентификатор владеющей данной страницей таблицы и указатели на следующую и предыдущую страницы в связанном списке. В конце страницы расположена таблица смещений строк, остальное пространство страницы занято строками данных.

- **1988** —Microsoft и Ashton-Tate анонсировали первую версию SQL Server — РСУБД для локальных вычислительных сетей. Новый продукт носил название Ashton-Tate/Microsoft SQL Server и представлял собой версию Sybase DataServer для OS/2. Роль Ashton-Tate заключалась в том, что фирма предоставила dBASE IV для разработки приложений.
- **1989** —В мае увидела свет первая версия Ashton-Tate/Microsoft SQL Server.
- **1990** — выпущен SQL Server v1.1 с поддержкой как OS/2, так и новой графической оболочки фирмы — Microsoft Windows 3.0.
- **1991** — Microsoft получила доступ к исходному коду SQL Server и начала работу над новой версией продукта.
- **1992** —Выпущен SQL Server 4.2 — 16-разрядная СУБД, результат совместной работы Microsoft и Sybase. В СУБД были реализованы клиентские библиотеки для MS-DOS, Windows и OS/2, помимо этого в нее впервые были включены средства администрирования с графическим интерфейсом под управлением Windows. Microsoft приняла решение сосредоточиться на развитии версий SQL Server только для Windows NT и остановить развитие версий для Unix. В октябре была выпущена бета-версия SQL Server для Windows NT.
- **1994** —закончилось сотрудничество Microsoft и Sybase, и далее эти две компании стали разрабатывать свои серверные СУБД независимо друг от друга. В конце года был выпущен сервер Sybase SQL Server System 10.
- **1996** —SQL Server 6.5, обладавший встроенной поддержкой Web-приложений, средствами распределенного администрирования, наличием динамических блокировок.
- **1998** —Microsoft SQL Server 7.0 с радикально измененной архитектурой. Это была первая версия SQL Server, не содержавшая унаследованного кода, оставшегося со времен сотрудничества с Sybase. Особо стоит отметить появление в этой версии OLAP-служб в составе продукта (до этого серверные OLAP-средства, производимые поставщиками серверных СУБД, включая и Oracle, продавались исключительно как отдельные продукты и относились к категории весьма дорогостоящего программного обеспечения).
- **2000** — выпущен Microsoft SQL Server 2000, поддерживающий Web-приложения, XML

Стандарты SQL

- ANSI – Американский национальный институт стандартов, ISO – Международная организация стандартов
- Стандарт SQL1 был впервые опубликован в 1986 г. - обеспечивал минимальную функциональность, обновлялся в 1989 – механизм поддержания ссылочной целостности
- в 1992 (SQL2) - расширенная функциональность
- в 1999 (SQL3) – интеграция с объектно-ориентированным подходом

SQL-серверы

СУБД	Производитель	URL
Oracle	Oracle Corp.	www.oracle.com
MS SQL	Server- Microsoft	www.microsoft.com
Informix	Informix	www.informix.com
Sybase	Sybase	www.sybase.com
DB2	IBM	www.4.ibm.com

Обработка распределенных данных

- Главная проблема больших систем - организация обработки распределенных данных. Данные находятся на компьютерах различных моделей и производителей, функционирующих под управлением различных ОС, а доступ к данным осуществляется разнородным программным обеспечением. Сами компьютеры территориально удалены друг от друга и находятся в различных географических точках планеты. Ответом стали две технологии: **технология распределенных БД (транзакций)** и **технология тиражирования данных.**
- Под **распределенной** БД подразумевают БД, включающую **фрагменты из нескольких БД, которые располагаются на различных узлах сети компьютеров**, и, возможно, **управляются различными СУБД**. Распределенная БД выглядит с точки зрения пользователей и прикладных программ как обычная локальная БД. В распределенных базах транзакция, выполнение которой заключается в обновлении данных на нескольких узлах сети, называется глобальной или распределенной транзакцией. Для пользователя распределенной БД глобальная транзакция выглядит как обычная. Для этого в современных СУБД предусмотрен так называемый протокол двухфазной фиксации транзакций:
  - Фаза 1 начинается, когда транзакция фиксируется. Сервер распределенной БД направляет уведомление "подготовиться к фиксации" всем серверам локальных БД, выполняющим распределенную транзакцию. Если все серверы приготовились к фиксации, сервер распределенной БД принимает решение о фиксации. Если хотя бы один из серверов не откликнулся на уведомление в силу каких-либо причин, будь то аппаратная или программная ошибка, то сервер распределенной БД откатывает локальные транзакции на всех узлах, включая даже те, которые подготовились к фиксации и оповестили его об этом.
  - Фаза 2 - сервер распределенной БД направляет команду "зафиксировать" всем узлам, затронутым транзакцией, и гарантирует, что транзакции на них будут зафиксированы. Если связь с локальной БД потеряна в интервал времени между моментом, когда сервер распределенной БД принимает решение о фиксации транзакции, и моментом, когда сервер локальной БД подчиняется его команде, то сервер продолжает попытки завершить транзакцию, пока связь не будет восстановлена.

Технология тиражирования данных

- Принципиальное отличие технологии тиражирования данных от технологии распределенных БД заключается в **отказе от распределенных данных**. Ее суть состоит в том, что любая БД (как для СУБД, так и для работающих с ней пользователей) всегда является локальной; данные всегда размещаются локально на том узле сети, где они обрабатываются; все транзакции в системе завершаются локально.
- **Тиражирование данных** - это асинхронный перенос изменений объектов **исходной** БД в БД, принадлежащим различным узлам распределенной системы. Функции тиражирования данных выполняет специальный модуль СУБД - сервер тиражирования данных, называемый **репликатором**. Его задача - поддержка идентичности данных в принимающих БД данным в исходной БД. В качестве базиса для тиражирования выступает транзакция к БД.
- Технология распределенных БД и технология тиражирования данных - антиподы. Принцип первой - синхронное завершение транзакций одновременно на нескольких узлах распределенной системы, то есть синхронная фиксация изменений в распределенной БД. "Ахиллесова пята" этой технологии - жесткие требования к производительности и надежности каналов связи. Если БД распределена по нескольким территориально удаленным узлам, объединенным медленными и ненадежными каналами связи, а число одновременно работающих пользователей составляет десятки и выше, то вероятность того, что распределенная транзакция будет зафиксирована в обозримом временном интервале, становится чрезвычайно малой. В таких условиях обработка распределенных данных практически невозможна.
- Реальной альтернативой технологии распределенных БД становится технология тиражирования данных, не требующая синхронной фиксации изменений. В действительности не во всех задачах требуется обеспечение идентичности БД на различных узлах в любое время. Достаточно поддерживать тождественность данных в определенные критичные моменты времени, т.е. можно накапливать изменения в данных в виде транзакций в одном узле и периодически копировать их на другие узлы.

- Просуммируем очевидные преимущества технологии тиражирования данных:
  - данные всегда расположены там, где они обрабатываются - следовательно, скорость доступа к ним существенно увеличивается.
  - передача только операций, изменяющих данные (а не всех операций доступа к удаленным данным, как в технологии распределенных БД), и к тому же в асинхронном режиме, позволяет значительно уменьшить трафик.
  - со стороны исходной БД для принимающих БД репликатор выступает как процесс, инициированный одним пользователем, в то время как в физически распределенной среде с каждым локальным сервером работают все пользователи распределенной системы, конкурирующие за ресурсы друг с другом.
  - никакой продолжительный сбой связи не в состоянии нарушить передачу изменений. Дело в том, что тиражирование предполагает буферизацию потока изменений (транзакций); после восстановления связи передача возобновляется с той транзакции, на которой тиражирование было прервано.
- Технология тиражирования обладает и недостатками. Например, невозможно полностью исключить конфликты между двумя версиями одной и той же записи. Он может возникнуть, когда вследствие все той же асинхронности два пользователя на разных узлах исправят одну и ту же запись в тот момент, пока изменения в данных из первой БД еще не были перенесены во вторую. Следовательно, при проектировании распределенной среды с использованием технологии тиражирования данных необходимо предусмотреть конфликтные ситуации и запрограммировать репликатор на какой-либо вариант их разрешения.

Просмотр списка баз SQL Server

В среде **Query Analyser** наберем команду и, выделив ее, нажмем клавишу **F5**

SQL Query Analyzer

Edit Query Tools Window Help

master

ect Browser

KERBEROS(KERBEROS\Ад

- KERBEROS
  - CompShop
  - master
  - model
  - msdb
  - Northwind
  - Notes
  - proba_1c
  - pubs
  - tempdb
  - АбитуриентSQL
- Common Objects
  - Configuration Functions
  - Cursor Functions
  - Date and Time Functions
  - Mathematical Functions
  - Aggregate Functions
  - Metadata Functions

Query - KERBEROS.master.KERBEROS\Администратор - Untitled1*

```
exec sp_helpdb
```

	name	db_size	owner	dbid	created	status	compatibili
1	CompShop	1.49 MB	KERBEROS\ASPNET	9	Jul 19 2008	Status=ONLINE...	80
2	master	13.88 MB	sa	1	Aug 6 2000	Status=ONLINE...	80
3	model	1.25 MB	sa	3	Aug 6 2000	Status=ONLINE...	80
4	msdb	14.00 MB	sa	4	Aug 6 2000	Status=ONLINE...	80
5	Northwind	4.25 MB	sa	6	Aug 6 2000	Status=ONLINE...	80
6	Notes	2.00 MB	KERBEROS\Адм...	8	Oct 23 2007	Status=ONLINE...	80
7	proba_1c	15.25 MB	KERBEROS\Адм...	7	Jul 25 2007	Status=ONLINE...	80
8	pubs	2.50 MB	sa	5	Aug 6 2000	Status=ONLINE...	80
9	tempdb	8.75 MB	sa	2	Aug 25 2008	Status=ONLINE...	80
10	АбитуриентSQL	2.56 MB	KERBEROS\Адм...	10	Aug 23 2008	Status=ONLINE...	80

Просмотр таблиц БД NorthWind

Выполним команду *Query – Change database* и выберем БД NorthWind. В окне запроса наберем команду exec sp_tables. Увидим список таблиц БД **NorthWind**.

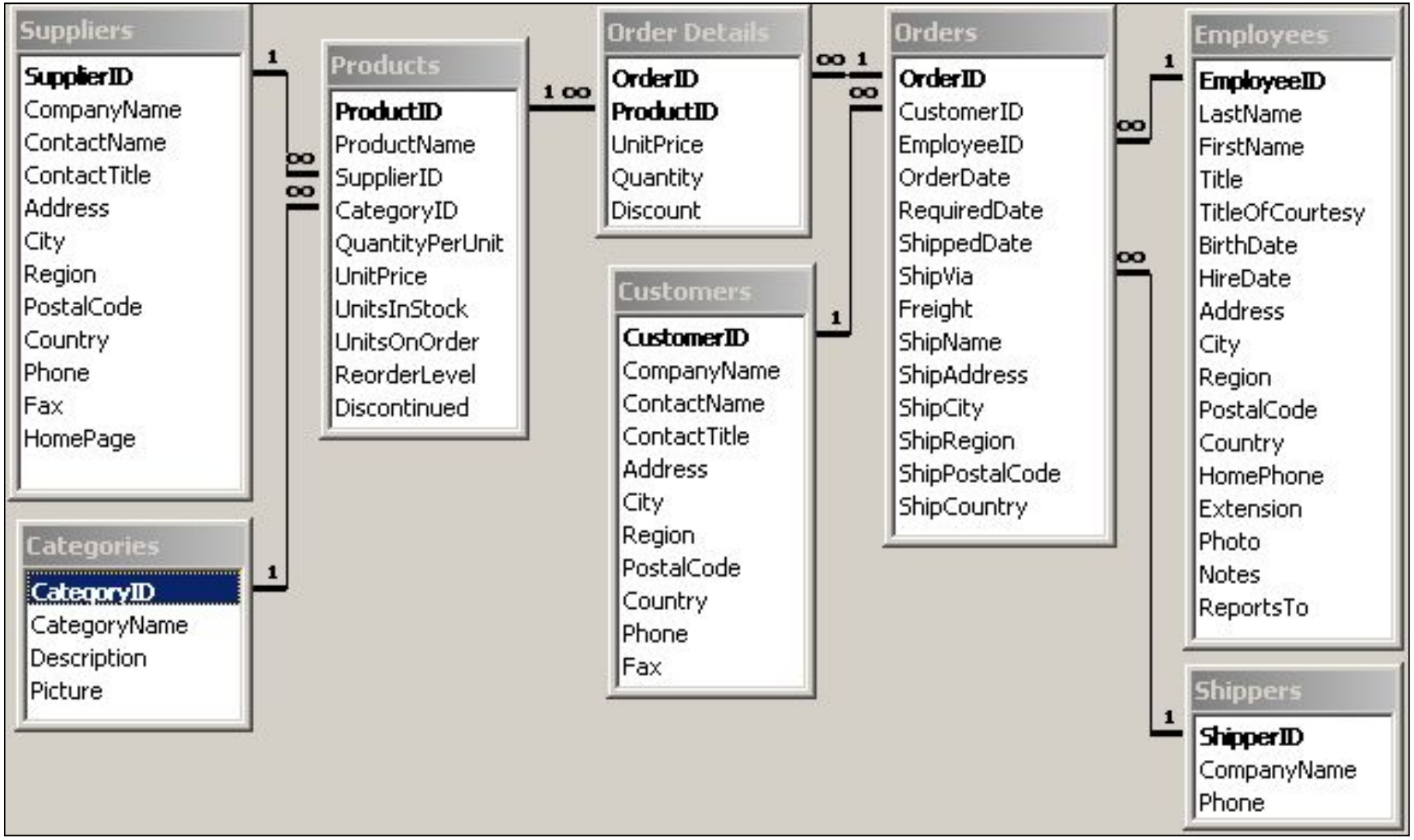
Query - KERBEROS.Northwind.KERBEROS\Администратор - Untitled1*

```
exec sp_tables
```

	TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	TABLE_TYPE	REMARKS
1	Northwind	dbo	syscolumns	SYSTEM TABLE	NULL
2	Northwind	dbo	syscomments	SYSTEM TABLE	NULL

Схема БД Northwind

Состав таблиц БД: **Supplier** – поставщик, **Products** – товар, **Order** – счет, **Customer** – покупатель, **Employeer** – продавец, **Shipper** - перевозчик



Тип данных MS Access	Тип данных SQL Server
Логический	bit
Числовой (Байт)	tinyint
Числовой (Целое)	smallint
Числовой (Длинное целое)	int
Числовой (Одинарное с плавающей точкой)	real
(отсутствует)	bigint
Числовой (Двойное с плавающей точкой)	float
Денежный	money , smallmoney
Действительное/числовой	decimal , numeric
Дата, время	datetime , smalldatetime
Счетчик	int (с определенным свойством **Идентификация**)
Текстовый (n)	varchar(n) , nvarchar(n)
Поле MEMO	text
Объекты OLE	image
Код реплики (глобальный уникальный идентификатор (GUID))	uniqueidentifier
Гиперссылка	char, nchar, varchar, nvarchar (со свойством **Гиперссылка**, имеющим значение «Да»)
(отсутствует)	varbinary
(отсутствует)	smallint
(отсутствует)	timestamp
(отсутствует)	char , nchar
(отсутствует)	sql_variant
(отсутствует)	определяемый пользователем

Словарь SQL

Два типа запросов:

- Возвращающий строки: Select

SELECT [List of Fields or *]
FROM [Table(s)] {Optionally Join Syntax}
WHERE [Criteria]
GROUP BY [List of Fields]
ORDER BY [List of Fields]

- Не возвращающие строки: Action Queries
  - *Update* : изменение записей
  - *Insert* : вставка новой записи
  - *Delete* : удаление записи

Типы данных

- Для указания даты используется знак **#** (в стандарте ANSI – апостроф, т.е. **'2/17/94 13:00'**: **#5/2/62#** **#4:12 am#**
- Значение NULL обозначает отсутствие данных в поле. Null это не 0 и не пустая строка. Сравнение выполняется с помощью оператора *Is NULL.*

Примеры:

1. Все строки таблицы Authors

*SELECT * FROM Authors*

2. Все столбцы и те строки, для которых в столбце PubID = 1213

*SELECT * FROM Publishers WHERE PubID = 1213*

3. Два столбца и те строки, для которых верно условие

```
SELECT LastName, PlaceofBirth
FROM Customers
WHERE ((AGE > 30) and (SEX = 'M'))
ORDER BY LastName, PlaceOfBirth
```

Для выборки данных по шаблону можно использовать оператор **LIKE** с заменителями - **%** или ***.**

Примеры оператора LIKE

- (MS **Access использует для указания любого символа знак *, ANSI SQL - %**):

...Where ((LastName Like ‘SM’) or (Name Like ‘sm*’) or (Name Like ‘Sm*’))*

- Оператор LIKE выполняется быстрее, если указан в конце оператора WHERE.
- ANSI SQL использует круглые скобки ( ). MS Access использует также [ ], поэтому желательно для преемственности кода заменить скобки на круглые.
- Для указания в запросе источника данных используется символ точка(.)

Database.Table.Field

- Оператор IN используется в операторе WHERE для указания подмножества, к которому может относиться проверяемое поле записи.
- Подмножеством может быть список или результат выполнения запроса (в этом случае подзапрос должен возвратить значения одного поля)

Select Name, YearBorn From Authors Where YearBorn In (1962, 1963, 1964)

Select Name, YearBorn From Authors Where YearBorn In (Select Year From HoleInOne)

- Asterisk (*)
  - SELECT authorID, firstName, lastName FROM Authors WHERE lastName LIKE ‘D*’
- Question mark (?)
  - SELECT authorID, firstName, lastName FROM Authors WHERE lastName LIKE ‘?I*’
  - DELETE FROM Authors WHERE firstName Like 'Chan%‘ (ANSI SQL)
  - DELETE FROM Authors WHERE firstName Like 'Chan*‘ (MS Access)

```
SELECT Titles.title, Titles.isbn, Authors.firstName, Authors.lastName,
    Titles.copyright, Publishers.publisherName
FROM (Publishers INNER JOIN Titles ON Publishers.publisherID =
    Titles.publisherID)
INNER JOIN (Authors INNER JOIN AuthorISBN ON Authors.authorID =
    AuthorISBN.authorID) ON Titles.isbn = AuthorISBN.isbn
ORDER BY Titles.title;
```

Оптимизация команды SELECT

- Не указывайте лишние столбцы в запросе
- Используйте не перечисление полей, а символ * (все поля).

Команда DELETE

DELETE * FROM таблица WHERE условие

Примеры:

Delete * From Authors Where Dead = TRUE
Delete * From Publishers Where PubID > 30
Delete * From MooCows
Delete *

Команда UPDATE

UPDATE таблица SET поле = значение [, поле = значение ...] WHERE условие

Примеры:

Update Authors Set Commissions = (Sales * 0.1)
Update Authors Set Address = '123 Maple' Where (AuID = 3121)
Update Authors Set Dead=False, Stupid=True Where ((Sales>100000) and (Commissions=0))

Если команда содержит вычисления, то они будут выполняться на стороне сервера и это хорошо.

Команда INSERT

INSERT INTO таблица (поле, поле) VALUES (значение, значение)

Примеры:

INSERT INTO authors (Name, Address, Sales) VALUES (‘Smith, Frank’, ‘123 Main St’, 35232.06)

INSERT INTO publishers (Name, ABACODE, Paperbacks) VALUES (‘Smith Books’, 1311, TRUE)

ANSI SQL: True это не ноль, обычно –1, False это ноль

Связывание таблиц

Для выборки из связанных таблиц используется оператор JOIN.

Связи между таблицами бывают двух типов:

Внутренние **INNER**: запрос содержит совпадающие по ключевым полям строки.

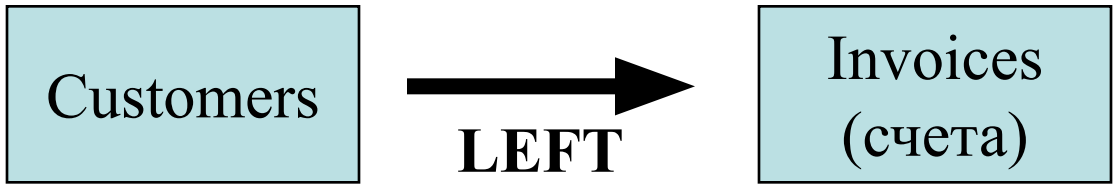
Внешнее **OUTER**: запрос может включать пустые (NULL) поля.

Некоторые СУБД используют слово FULL

```
SELECT [поля]
FROM таблицаA {INNER | LEFT | RIGHT} JOIN таблицаB
ON (таблицаA.поле1 = таблицаB.поле2)
WHERE [условие]
ORDER BY [поля]
```

Выбор внешнего соединения – левое или правое?

- Внешнее соединение используется для
  - Выявления несовпадений в ключевых полях таблиц
  - Выявления пустых полей
- Левое соединение LEFT JOIN выбирает все записи левой таблицы и совпадающие по ключевому полю записи правой таблицы. Правое соединение – наоборот. Соединение **Full** JOIN выберет все записи в обеих таблицах, в том числе с совпадающими ключевыми полями



Все Покупатели с учетом и без учета Счетов

Все Счета с учетом и без учета Покупателей

Найти всех авторов без книг:

```
SELECT DISTINCTROW Authors.Au_ID, Authors.Author
FROM Authors LEFT JOIN [Title Author] ON Authors.Au_ID = [Title Author].Au_ID
WHERE ((([Title Author].Au_ID) Is Null))
```

Оператор HAVING

- Оператор HAVING фильтрует результаты после их группировки
- Оператор WHERE фильтрует все записи
- Оператор WHERE лучше, хотя может быть неизбежно использование оператора HAVING.

Оператор Group By

- Используется для группировки записей
- Все поля, перечисляемые в части SELECT, должны упоминаться и в части GROUP BY, кроме тех полей, что участвуют в вычислениях в операторе SELECT.
- С помощью оператора AS можно дать имена вычисляемым полям, в которых могут использоваться агрегатные функции (вычисляющие итоги): COUNT, SUM, AVE, MIN, MAX, STDEV, VAR, FIRST, LAST

```
SELECT Titles.PubID, Titles.[Year Published], Count(Titles.Title) AS Count
FROM Titles
GROUP BY Titles.PubID, Titles.[Year Published]
```

PubID	Year Published	Count
3	1994	31
3	1995	46
3	1996	27
4	1980	1
4	1986	1

Подзапросы

- Полезны для сложной выборки данных
- Могут быть именованными
- Могут использоваться для поиска дубликатов записей:

```
SELECT DISTINCTROW Titles.Title, Titles.[Year Published], Titles.ISBN, Titles.PubID
FROM Titles
WHERE ( ((Titles.Title) In          'Альтернативное использование оператора IN
(SELECT [Title] FROM [Titles] As Tmp GROUP BY [Title] HAVING Count(*)>1 )) )
ORDER BY Titles.Title
```

Сводные (перекрестные) таблицы

TRANSFORM aggfunction selectstatement PIVOT pivotfield [IN (value1[, value2[, ...]])]

```
TRANSFORM Count(Titles.ISBN) AS [The Value]
SELECT Titles.[Year Published], Count(Titles.ISBN) AS [Total Of ISBN]
FROM Titles
GROUP BY Titles.[Year Published]
PIVOT Titles.PubID
```

Извлечение данных

```
Use NorthWind
SELECT ProductName AS [Название продукта]
from Products
```

	Название продукта
1	Alice Mutton
2	Aniseed Syrup

Указание на обращение к таблицам БД может быть указано явно командой **USE**. Полям таблицы можно задать псевдонимы, т.е. заголовки.

```
Use NorthWind
SELECT productname AS [название товара],
unitprice AS [Цена] FROM Products
WHERE unitprice BETWEEN 10 AND 12
```

	название товара	Цена
	Aniseed Syrup	10.0000
	Sir Rodney's Scones	10.0000
	Spegesild	12.0000
	Longlife Tofu	10.0000

Выбор товаров, цена которых лежит в пределах от 10 до 12.

```
Use NorthWind
SELECT productname AS [название товара],
unitprice AS [Цена] FROM Products
WHERE unitprice in (10,18)
```

Выбор товаров, цена которых или 10 или 18

	название товара	Цена
	Chai	18.0000
	Aniseed Syrup	10.0000
	Sir Rodney's Scones	10.0000
	Steeleye Stout	18.0000
	Chartreuse verte	18.0000
	Longlife Tofu	10.0000
	Lakkalikööri	

```
Use NorthWind
SELECT DISTINCT Unitprice from Products
```

	Unitprice
1	2.5000
2	4.5000

Для исключения повторов в **столбце** используется ключ **DISTINCT**

Функции

Для работы с датами используются функции извлечения года (**YEAR**), месяца (**MONTH**), дня (**DAY**)…

```
USE northwind
SELECT firstname,birthdate FROM employees
WHERE MONTH(birthdate)=7
```

firstname	birthdate
Michael	1963-07-02 00:00:00.000

Выбрать компании, у которых не указан регион

```
USE northwind
SELECT region, companyname FROM suppliers
WHERE region IS NOT NULL
```

region	companyname
LA	New Orleans Cajun Delights
MI	Grandma Kelly's Homestead

Сравнение со строкой - оператор **LIKE** со знаками **%** или **?** (т.е. любое количество символов. В MS Access знак “*”.)

```
USE northwind
SELECT productname FROM products
WHERE productname LIKE '%TOU%'
```

productname
Steeleye Stout
Tourtière

_ один любой символ;

[-…] один символ из диапазона:

```
USE northwind
SELECT productname FROM products
WHERE productname LIKE '%Ch_g%'
```

productname
Queso Manchego La Pastora
Schoggi Schokolade

```
USE northwind
SELECT productname FROM products
WHERE productname LIKE '%gu[ld]%'
```

productname
Gudbrandsdalsost
Gula Malacca

[…] один из символов **в скобках**;

Товары, в названии которых встречается комбинация букв “gu”, за которой может стоять буква “l” или “d”

[^…] любой символ **не в скобках**;

Товары, в названии которых есть комбинация букв “gu“, после которых не следует буква “a”

```
USE northwind
SELECT productname FROM products
WHERE productname LIKE '%gu[^a]%'
```

productname
Chef Anton's Gumbo Mix
Gudbrandsdalsost
Gula Malacca

Упорядочение записей, подсчет итогов

Упорядочение записей по значению поля (полей) выполняется с помощью оператора **ORDER BY**. Для упорядочения по возрастанию используется ключ **ASC** (по умолчанию), по убыванию - **DESC**.

```
USE northwind
SELECT productname FROM products
ORDER BY productname
```

productname
Alice Mutton
Aniseed Syrup
Boston Crab Meat
Camembert Pierrot

Наименование товара в алфавитном порядке

```
USE northwind
SELECT productname, Unitprice FROM products
ORDER BY unitprice DESC
```

productname	Unitprice
Côte de Blaye	263.5000
Thüringer Rostbratwurst	123.7900
Mishi Kobe Niku	97.0000

Список из наименования и цены товара, отсортированные по убыванию цены.

Выборка первых N записей с помощью ключа **TOP**. Отсортировав записи можно выбрать наилучшую (наихудшую) выборку товаров.

Десятка наиболее дорогих товаров

```
USE northwind
SELECT TOP 10 productname, Unitprice
FROM products ORDER BY unitprice DESC
```

productname	Unitprice
Côte de Blaye	263.5000
Thüringer Rostbratwurst	123.7900
Mishi Kobe Niku	97.0000
Sir Rodney's Marmalade	81.0000
Carnarvon Tigers	62.5000
Raclette Courdavault	55.0000
Manjimup Dried Apples	53.0000
Tarte au sucre	49.3000
Ipoh Coffee	46.0000
Rössle Sauerkraut	45.6000

Подсчет статистики по столбцам - функции: **Max, Min, SUM, AVG** (ср. знач.), **COUNT** (количество), **STDEV** (стандартное отклонение), **VAR** (дисперсия)

```
USE northwind
SELECT sum(Unitprice)
FROM products
```

(No column name)
2222.7100

```
USE northwind
SELECT max(Unitprice)
FROM products
```

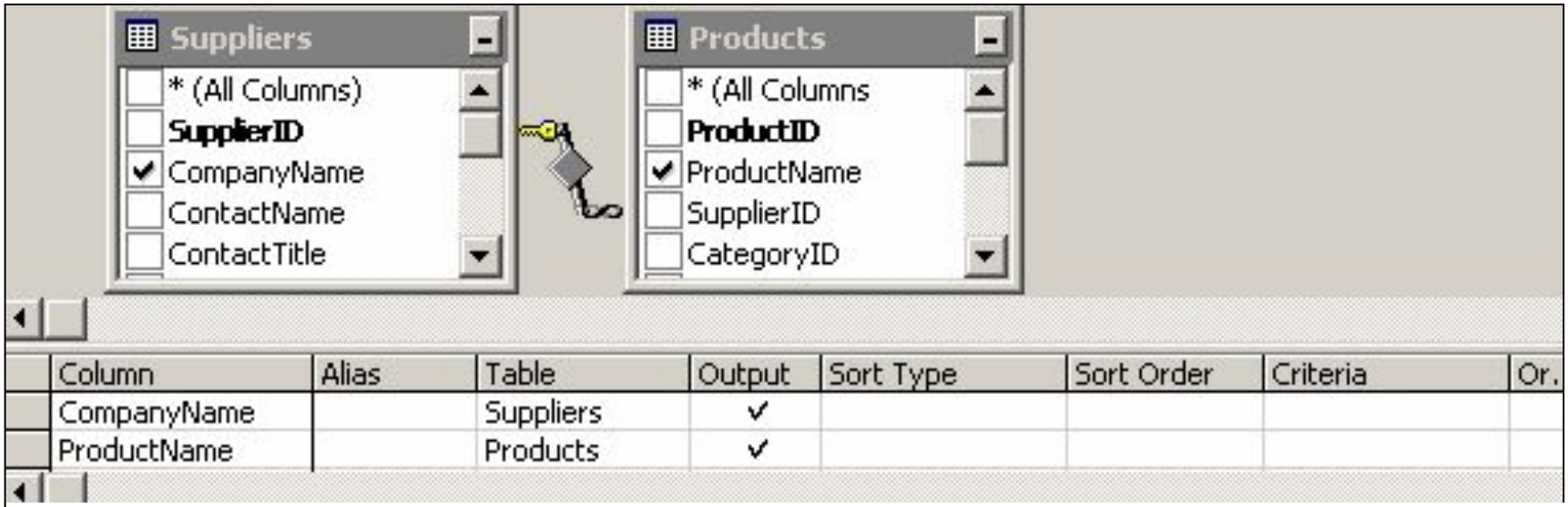
(No column name)
263.5000

Количество товара, цена которого менее 50

```
USE Northwind
SELECT COUNT(*) AS [Кол-во товара]
FROM products WHERE unitprice<50
```

Кол-во товара
70

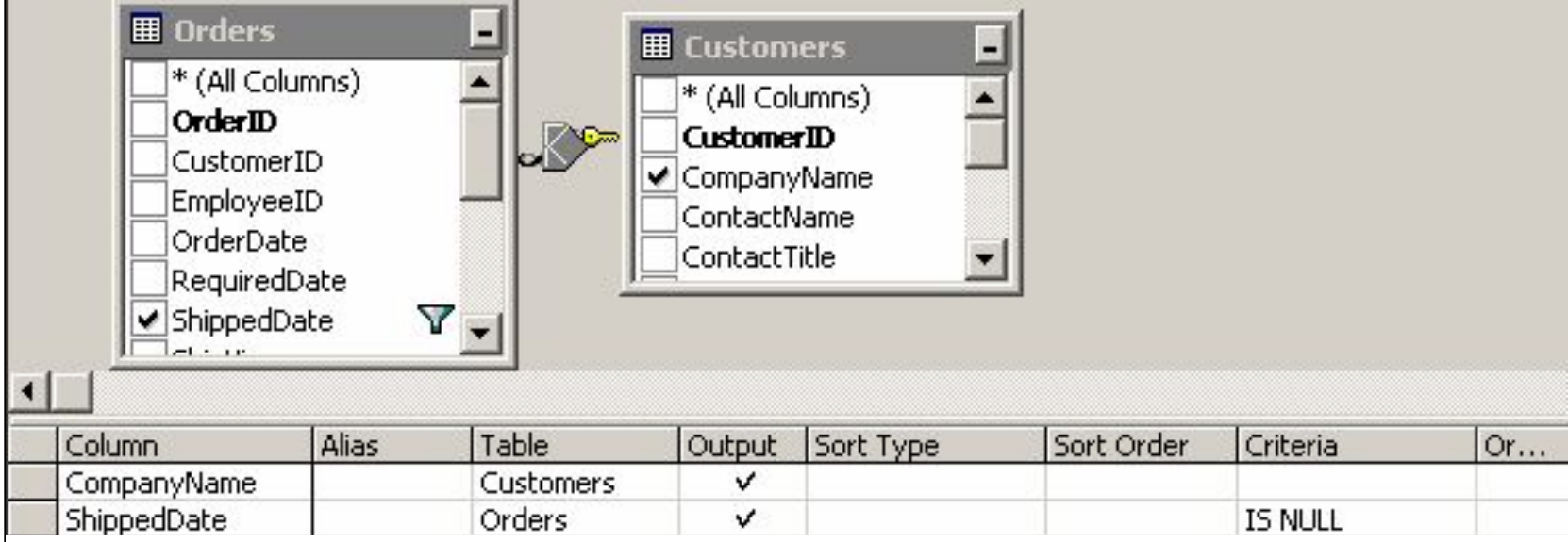
Соединение таблиц задается в секции **FROM**. Условия выборки задаются в конструкции **WHERE** (при группировке **GROUP BY -** в конструкции **HAVING)**. Типы соединений: внутреннее (выбираются те строки, которые соответствуют условию соединения), внешнее (возвращаются все строки главной таблицы - левой или правой или обоих, участвующих в соединении с учетом условия выборки).



Column	Alias	Table	Output	Sort Type	Sort Order	Criteria	Or.
CompanyName		Suppliers	✓				
ProductName		Products	✓				

```
SELECT dbo.Suppliers.CompanyName, dbo.Products.ProductName
    FROM dbo. Suppliers INNER JOIN dbo.Products
        ON dbo.Suppliers.SupplierID = dbo.Products.SupplierID
```

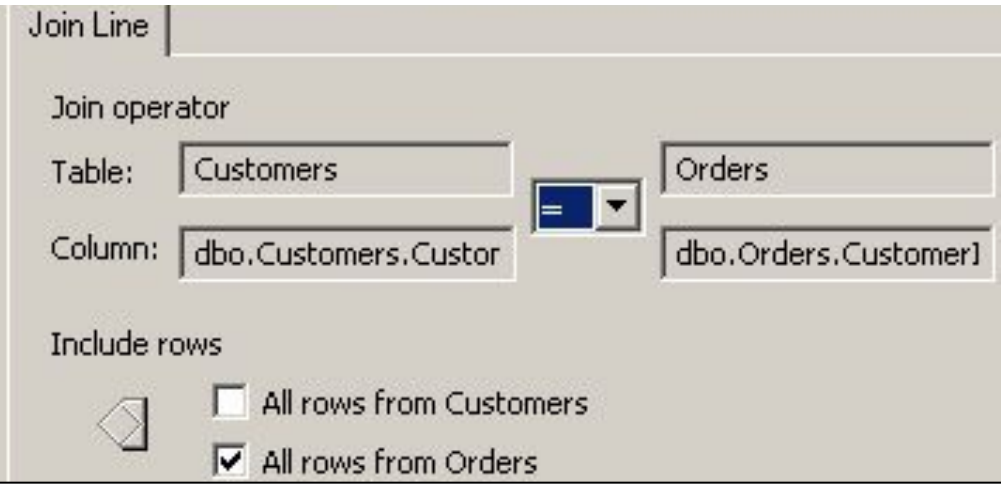
CompanyName	ProductName
Exotic Liquids	Chai
Exotic Liquids	Chang
Exotic Liquids	Aniseed Syrup
New Orleans Cajun	Chef Anton's Cajur
New Orleans Cajun	Chef Anton's Gumb
Grandma Kelly's Ho	Grandma's Boysent
Grandma Kelly's Ho	Uncle Bob's Organic



Column	Alias	Table	Output	Sort Type	Sort Order	Criteria	Or...
CompanyName		Customers	✓				
ShippedDate		Orders	✓			IS NULL	

```
SELECT dbo.Customers.CompanyName, dbo.Orders.ShippedDate
    FROM dbo. Customers RIGHT OUTER JOIN dbo.Orders
        ON dbo.Customers.CustomerID = dbo.Orders.CustomerID
        WHERE (dbo.Orders.ShippedDate IS NULL)
```

CompanyName	ShippedDate
Ernst Handel	<NULL>
Rancho grande	<NULL>
LINO-Delicateses	<NULL>
Great Lakes Food M	<NULL>
Bottom-Dollar Mark	<NULL>
La maison d'Asie	<NULL>
Cactus Comidas pa	<NULL>
Blauer See Delikate	<NULL>
Ricardo Adocicados	<NULL>
Great Lakes Food M	<NULL>
Reggiani Caseifici	<NULL>



Правое соединение – RIGHT OUTER JOIN, левое – LEFT OUTER JOIN, полное – FULL OUTER JOIN. В среде **Query Analyser** слово **OUTER** (внешний) не обязательно.

Таблицы Поставщики и Продукты связаны условием **INNER JOIN** по полю **SupplierID** (код поставщика).

```
USE Northwind
SELECT suppliers.companyname, products.productname
FROM suppliers INNER JOIN products
ON suppliers.supplierID=products.supplierID
```

companyname	productname
Exotic Liquids	Chai
Exotic Liquids	Chang
Exotic Liquids	Aniseed Syrup
New Orleans Cajun Delights	Chef Anton's Cajun

Для уникальных полей названия таблиц указывать не обязательно.
Можно также использовать псевдонимы **таблиц**

```
USE Northwind
SELECT s.companyname, p.productname
FROM suppliers AS s INNER JOIN products AS p
ON s.supplierID=p.supplierID
```

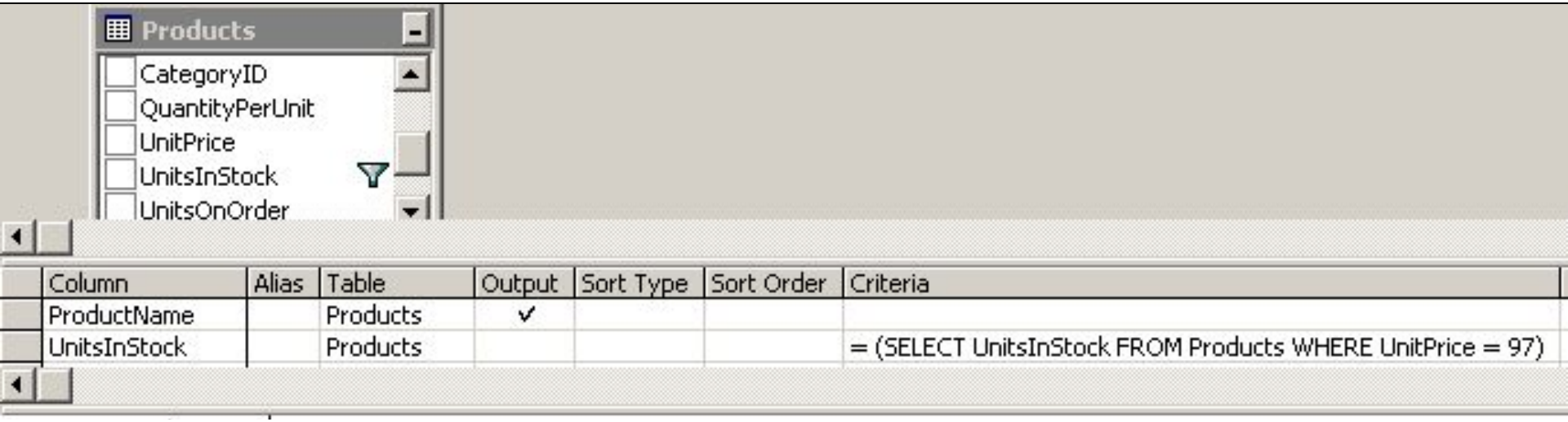
```
USE Northwind
SELECT Customers.CompanyName, Orders.ShippedDate
FROM Customers RIGHT JOIN Orders
ON Customers.CustomerID = Orders.CustomerID
WHERE Orders.ShippedDate Is Null
```

CompanyName	ShippedDate
Ernst Handel	NULL
Rancho grande	NULL

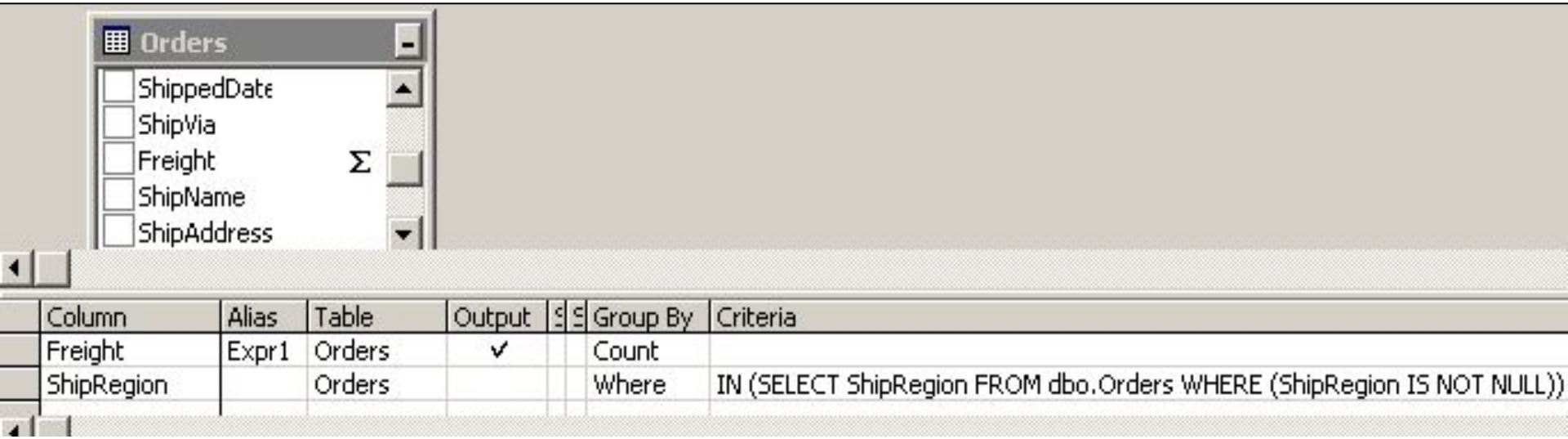
Покупатели, кому товар еще не доставлен (дата доставки – поле ShippedDate таблицы Orders (счета)).

Подзапрос - запрос, вложенный во внешний оператор **SELECT, INSERT, UPDATE, DELETE.** Возвращает **одно** значение. Подзапросы могут содержать ключи **IN, ANY, ALL, EXISTS**.

Найти товар, запас (UnitsInStock) которого совпадает с выборкой товаров с ценой=97.




```
SELECT  ProductName FROM dbo. Products
     WHERE  (UnitsInStock =
     (SELECT UnitsInStock FROM Products WHERE UnitPrice = 97))
```




```
SELECT  COUNT(Freight) AS Expr1 FROM dbo.Orders
   WHERE (ShipRegion IN
      (SELECT ShipRegion FROM dbo.Orders
          WHERE (ShipRegion IS NOT NULL)))
```

Expr1
323

Количество элементов в поле *Freight* таблицы *Orders* для которых в поле *ShipRegion* нет пустых значений.

Найти сумму цен 5 дешевых товаров (создадим подзапрос, затем в основной запрос включим текст подзапроса):

Column	Alias	Table	Output	Sort Type
UnitPrice		Products	✓	Ascending

```
SELECT TOP 5 UnitPrice
    FROM dbo.Products
        ORDER BY UnitPrice
```

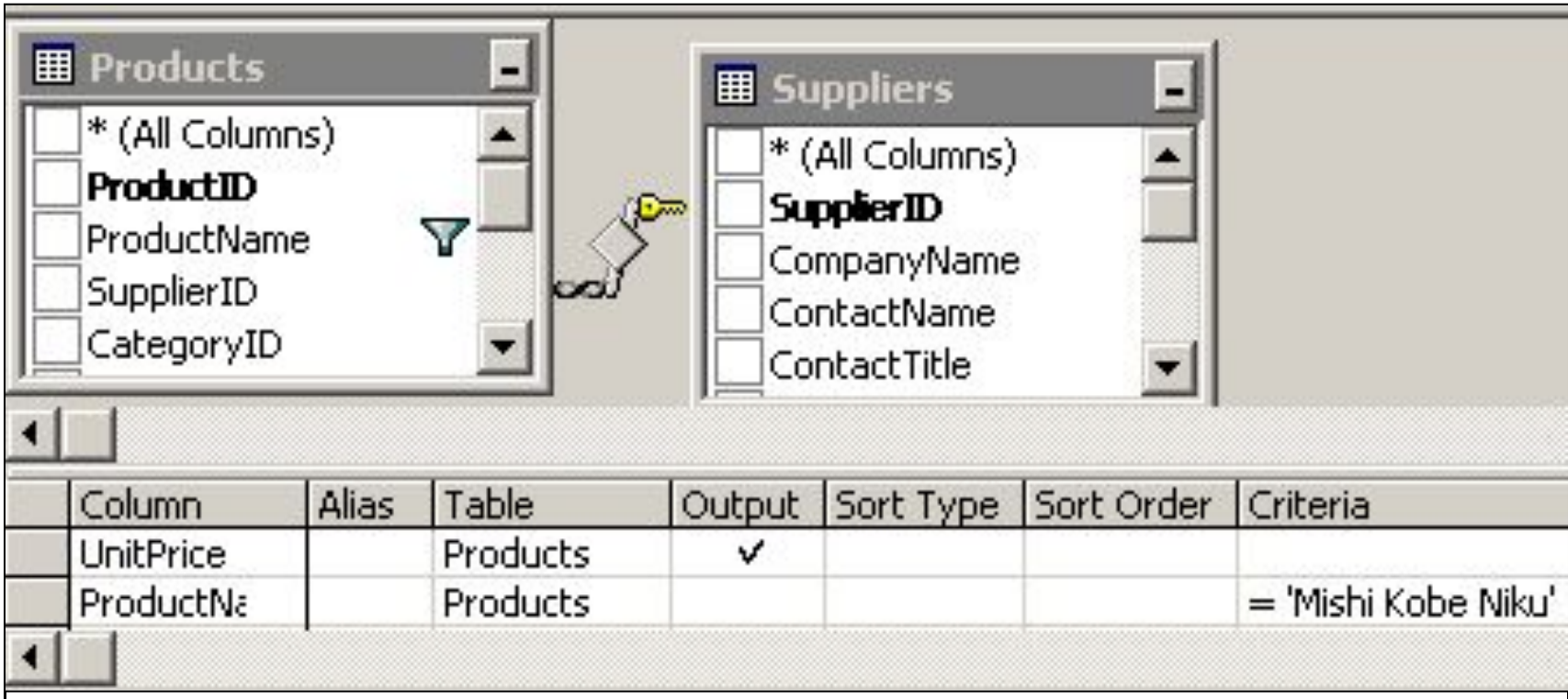
UnitPrice
2,5
4,5

Column	Alias	Table	Output	Sort Type	Sort Order	Group By	Criteria
UnitPrice		Products				Where	IN (SELECT TOP 5 UnitPrice FROM dbo.Products ORDER BY UnitPrice)
UnitPrice	SumP	Products	✓			Sum	

```
SELECT SUM(UnitPrice) AS SumP
    FROM dbo.Products
        WHERE (UnitPrice IN
            (SELECT TOP 5 UnitPrice FROM dbo. Products ORDER BY UnitPrice))
```

SumP
27,45

Найти товары, цена за единицу которых больше, чем у продукта “Mishi Kobe Niku” (создадим подзапрос, затем включим его в запрос)




```
SELECT dbo.Products.UnitPrice
    FROM dbo.Products INNER JOIN dbo.Suppliers
        ON dbo.Products.SupplierID = dbo.Suppliers.SupplierID
            WHERE (dbo.Products.Product Name = 'Mishi Kobe Niku')
```

UnitPrice
97

Ключевые слова **ALL** и **ANY** сравнивают скалярное значение с набором значений **одного столбца**.

Ключ **ALL** применяется ко всем значениям, **ANY** – как минимум к одному.

Column	Table	Output	Sort	Criteria
ProductName	Product:	✓		
UnitPrice	Product:			> ANY (SELECT dbo.Products.UnitPrice FROM dbo.Pro

```
SELECT dbo.Priducts.ProductName FROM  dbo.Products
WHERE  (dbo.Products.UnitPrice > ANY
  (SELECT dbo.Products.UnitPrice
    FROM dbo.Products INNER JOIN dbo.Suppliers
      ON dbo.Products.SupplierID = dbo.Suppliers.SupplierID
        WHERE  (dbo.Products.Product Name = 'Mishi Kobe Niku')))
```

ProductName
Thüringer Rostbrat
Côte de Blaye

Товары, цена за единицу которых больше, чем у продукта “Mishi Kobe Niku”

Ключевое слово **EXISTS** проверяет наличие атрибута.

Оператор **WHERE** внешнего запроса проверяет, существуют ли строки, соответствующие подзапросу. Результат конструкции WHERE – **TRUE** или **FALSE.**

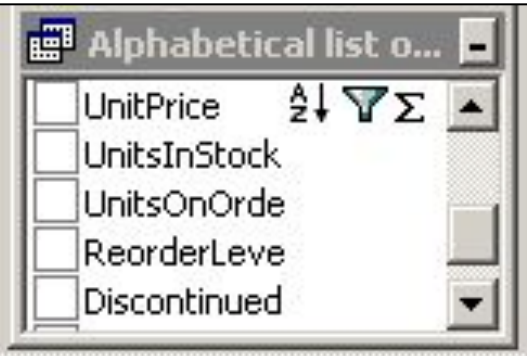
Column	Table	Output	Criteria
CompanyName	Suppliers	✓	
EXISTS (SELECT * FROM products WHERE suppliers.supplierid = products.supplierid AND unitprice = 1			= TRUE

```
SELECT DISTINCT dbo.Suppliers.CompanyName
  FROM dbo.Products INNER JOIN dbo.Suppliers
    ON dbo.Products.SupplierID = dbo.Suppliers.SupplierID
      WHERE EXISTS (SELECT * FROM products
        WHERE suppliers.supplierID = products.supplierID AND unitprice = 14)
```

CompanyName
Bigfoot Breweries
Heli Süßwaren Gmb
Leka Trading

Поставщики, цена товара которых = 14. Попутно выполняется проверка наличия таких продуктов от поставщиков.

Группировка строк



Выбрать 100 категорий тех товаров из запроса *Alphabetical list of products,* у которых средняя (**AVG**) цена находится в интервале от 10 до 50.

Column	Alias	Table	Output	Sort Type	Sort Order	Group By
CategoryName		[Alphabetical list of products]	✓			Group By
UnitPrice	AVGPrice	[Alphabetical list of products]	✓	Ascending	1	Avg

```
SELECT TOP 100 CategoryName, AVG(UnitPrice) AS AVGPrice
    FROM dbo.[Alphabetical list of products]
        WHERE  (UnitPrice BETWEEN 10 AND 50)
            GROUP BY CategoryName ORDER BY AVG(UnitPrice)
```

CategoryName	AVGPrice
Beverages	20
Seafood	20,0675
Produce	21,0833
Confections	22,738
Condiments	23,2181
Meat/Poultry	24
Grains/Cereals	27,9375
Dairy Products	28,725

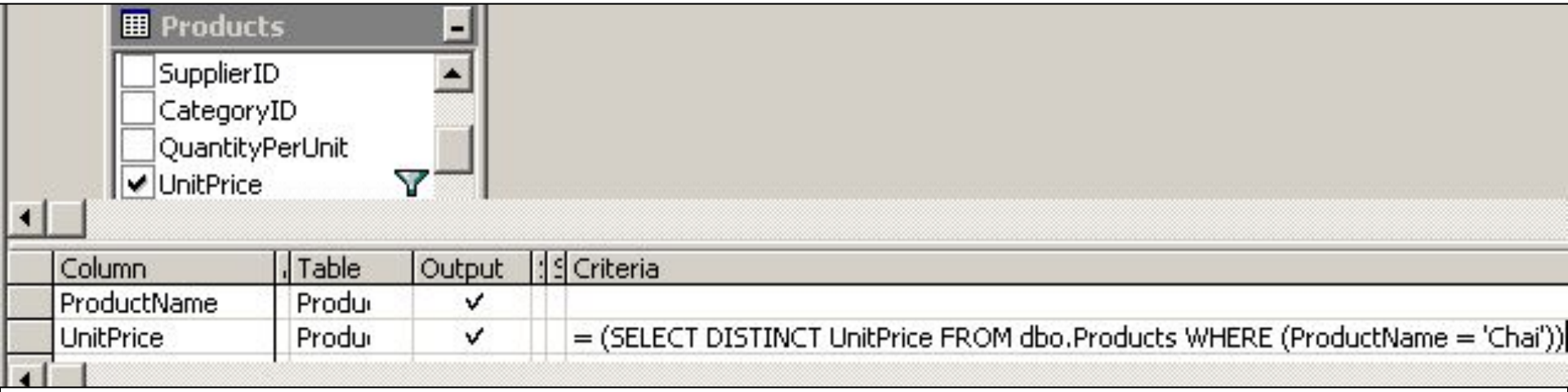
Дополнительную фильтрацию выполним с помощью выражения **HAVING**.

Column	Table	Output	S	S	Group By	Criteria
CategoryName	[Alphabetical list of product:	✓			Group By	
UnitPrice	[Alphabetical list of product:	✓			Avg	< 25

```
SELECT CategoryName, AVG(UnitPrice) AS AVGPrice
    FROM  dbo.[Alphabetical list of products]
        GROUP BY CategoryName
            HAVING (AVG(UnitPrice) < 25)
```

CategoryName	AVGPrice
Condiments	23,2181
Grains/Cereals	21,2916
Meat/Poultry	15,725
Seafood	20,6825

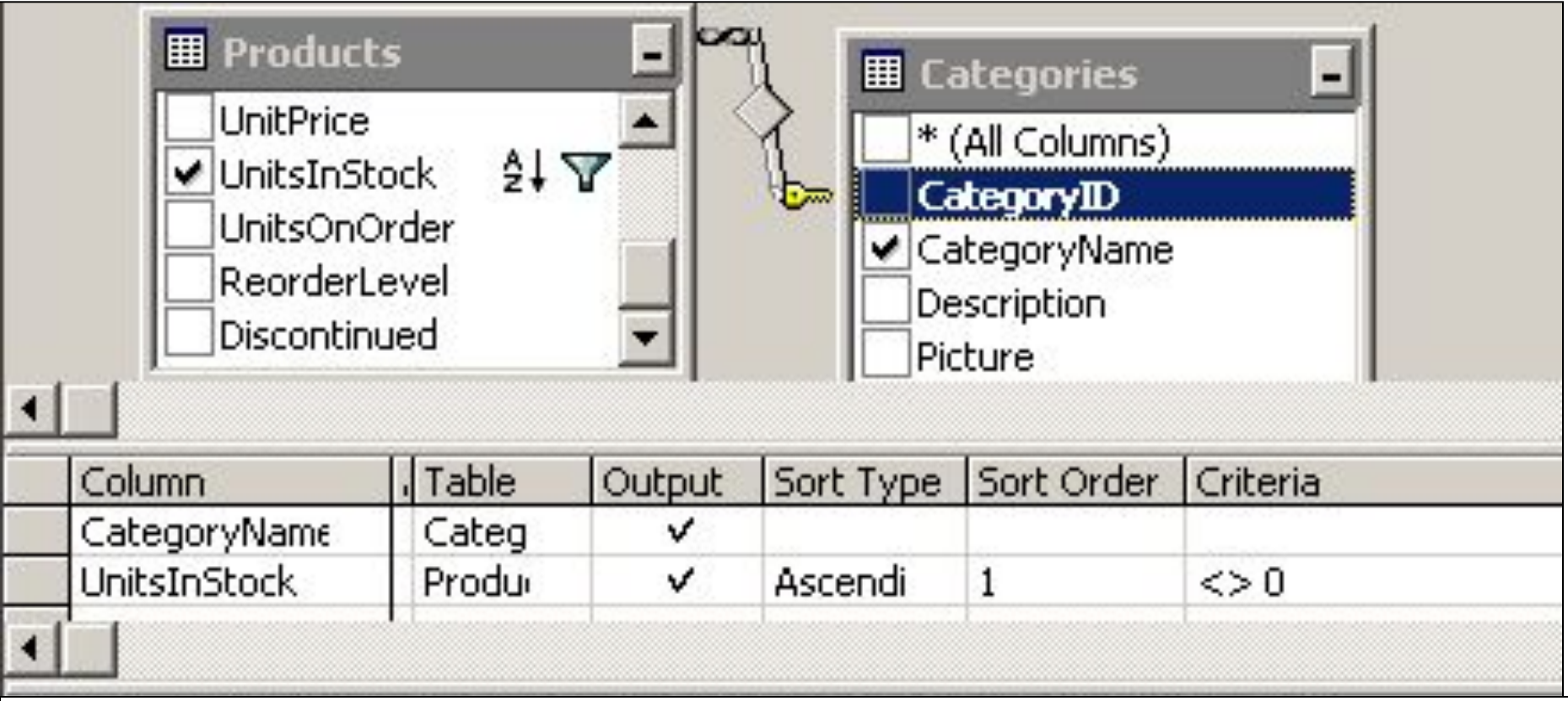
Товары, средняя цена которых < 25




```
SELECT  ProductName, UnitPrice  FROM  dbo.Products
  WHERE  (UnitPrice =
     (SELECT DISTINCT UnitPrice  FROM   dbo.Products
         WHERE  (ProductName = 'Chai')))
```

ProductName	UnitPrice
Chai	18
Steeleye Stout	18
Chartreuse verte	18
Lakkalikööri	18

Названия и цена продуктов, совпадающих по цене с товаром “Chai”.

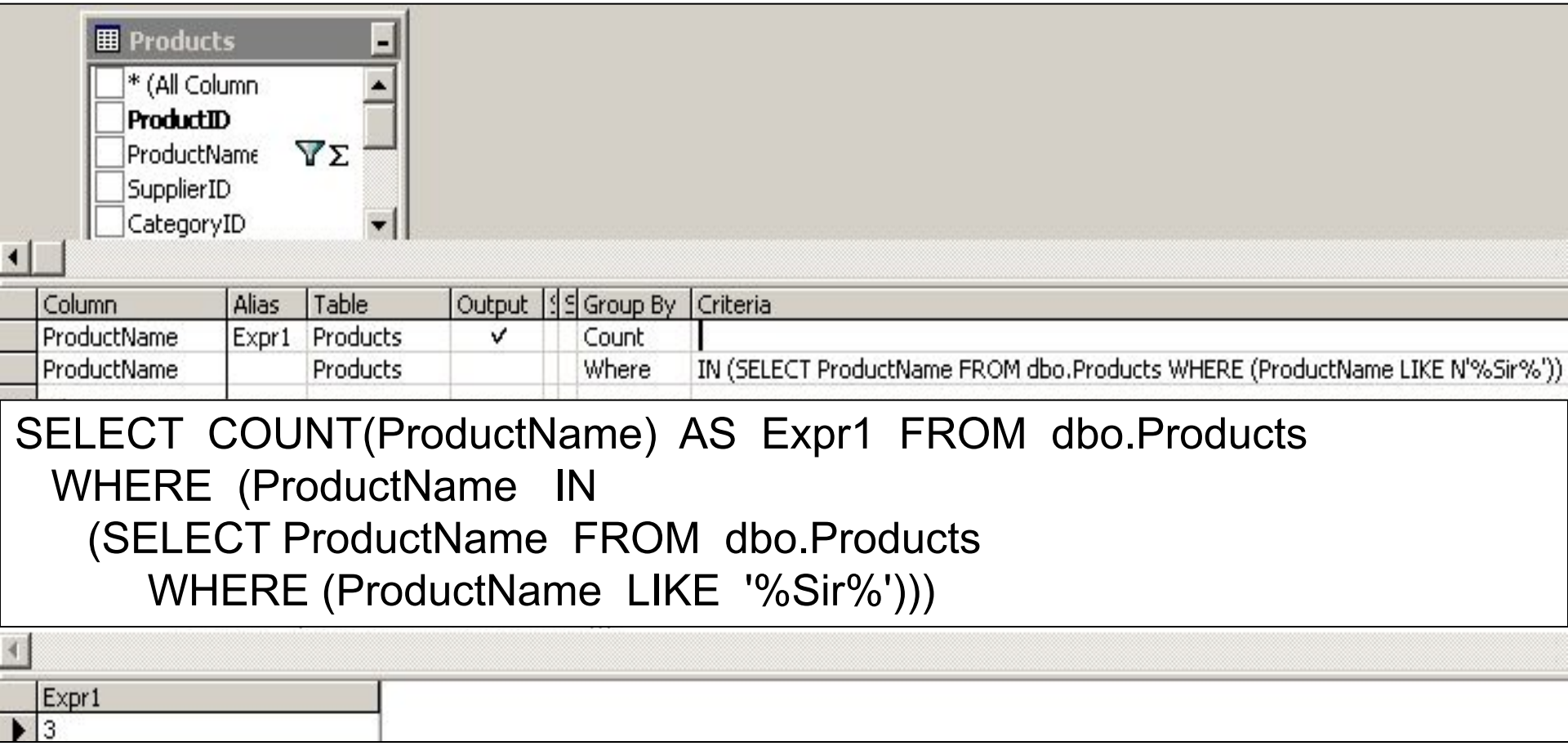


Column	Table	Output	Sort Type	Sort Order	Criteria
CategoryName	Categ	✓			
UnitsInStock	Produ	✓	Ascendi	1	<> 0

```
SELECT TOP 3 dbo.Categories. CategoryName, dbo.Products.UnitsInStock
  FROM dbo.Products INNER JOIN dbo.Categories
    ON dbo.Products.CategoryID = dbo.Categories.CategoryID
      WHERE (dbo.Products.UnitsInStock <> 0)
        ORDER BY dbo.Products.UnitsInStock
```

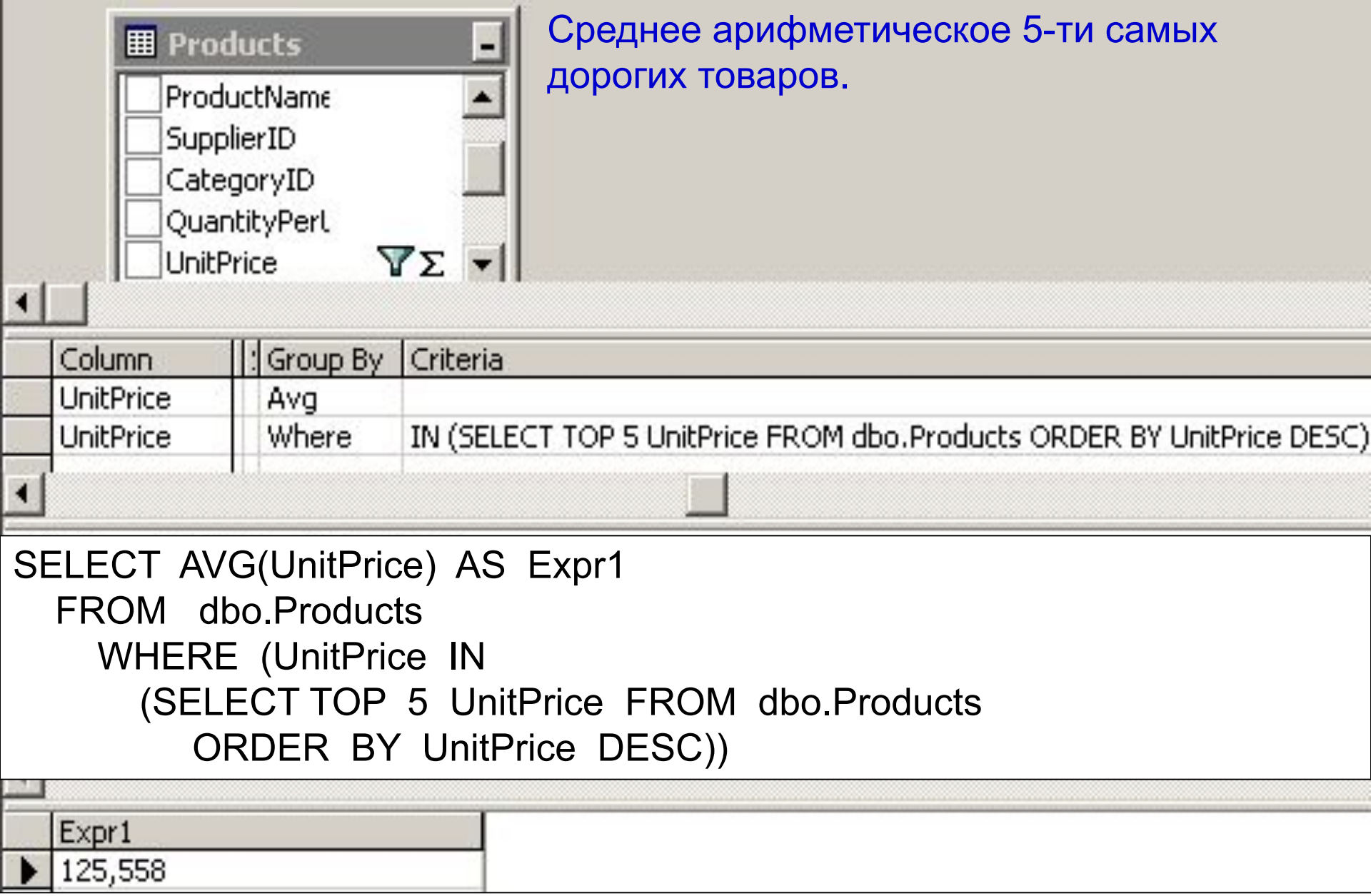
CategoryName	UnitsInStock
Confections	3
Condiments	4
Produce	4

3 категории товаров, запасы которых минимальны (сортировка по убыванию значения запаса)



Количество продуктов, в названии которых встречается слово ‘Sir’

Среднее арифметическое 5-ти самых дорогих товаров.




```
SELECT  AVG(UnitPrice)  AS  Expr1
   FROM   dbo.Products
      WHERE  (UnitPrice  IN
         (SELECT TOP  5  UnitPrice  FROM  dbo.Products
            ORDER  BY  UnitPrice  DESC))
```

Expr1
125,558

Запрос после конструктора можно подкорректировать вручную.
Необходимо проверять результат, например, в MS Excel

5 кафедр, число сотрудников которых максимально

```
SELECT TOP 5
сотрудник.[Код кафедры], Count(сотрудник.ФИО) AS число_сотрудников
        FROM сотрудник
            GROUP BY сотрудник.[Код кафедры]
                ORDER BY Count(сотрудник.ФИО) DESC;
```

ФИО и премии сотрудников

```
SELECT сотрудник.ФИО, [оклад]*0.5 AS Премия FROM сотрудник;
```

Средний оклад сотрудников

```
SELECT AVG(сотрудник.оклад) AS Ср_оклад FROM сотрудник;
```

Список сотрудников, у которых нет детей

```
SELECT DISTINCTROW сотр.ФИО FROM сотр LEFT JOIN дети
      ON сотрудник.Код=дети.Код_сотр
         WHERE (((дети.Код_сотр) Is Null));
```

Сотрудники с окладами в диапазоне от 1000 до 2000

```
SELECT  сотрудник.ФИО, сотрудник.оклад FROM сотрудник
     WHERE (((сотрудник.оклад) Between 1000 And 2000));
                    или Between 2000 And 1000
```

Сотрудники с окладами менее 1000 и более 15000

```
SELECT  сотрудник.ФИО, сотрудник.оклад FROM сотрудник
   WHERE (((сотрудник.оклад)<1000 Or (сотрудник.оклад)>15000));
```

Сотрудники с окладами вне диапазона от 1000 до 2000

```
SELECT  сотрудник.ФИО, сотрудник.оклад FROM сотрудник
   WHERE (((сотрудник.оклад) Not Between 1000 And 2000));
```

Сотрудники с ФИО, начинающимися на букву “Д”

```
SELECT  сотрудник.ФИО FROM сотрудник
      WHERE (((сотрудник.ФИО) Like "Д*"));
```

Кафедры с числом сотрудников более 5 человек

```
SELECT  сотр.[Код кафедры], Count(сотр.ФИО) AS [Число_сотрудников]
    FROM сотр
       GROUP BY сотр.[Код кафедры] HAVING (((Count(сотр.ФИО))>5));
```

*Таблицы фирма, сотрудники и экзамены. Сколько аттестованных сотрудников есть на **каждой** фирме (один сотрудник может быть аттестован по нескольким пунктам, поэтому **DISTINCT**. Предварительно для каждой фирмы проверяется наличие аттестованных сотрудников).*

```
SELECT сотр.фирма, Count(сотр.сотр) AS аттест_сотр  FROM сотр
  WHERE (EXISTS (SELECT DISTINCT сотр.фирма, сотр.сотр
      FROM сотр INNER JOIN экзам ON сотр.сотр = экзам.сотр)) <>False
        GROUP BY сотр.фирма;
```

Удалить записи с ФИО “Бурлак”

```
DELETE * FROM студент WHERE студент.ФИО="Бурлак";
```

Минимальная, максимальная и средняя зарплата

```
SELECT MIN(зарплата), MAX(зарплата), AVG(зарплата)
    FROM сотрудники;
```

Самая короткая фамилия

```
SELECT MIN(ФИО) FROM сотрудники;
```

```
SELECT COUNT(*), COUNT(налог) FROM сотрудники;
```