**34 СОЗДАНИЕ ДОМЕНА**

**34.1 Оператор *create domain*** или другое значение

Если в таблице присутствуют столбцы с одинаковы-ми характеристиками, то можно предварительно описать их тип и поведение с помощью домена. На-пример, создать тип *Pol\_type* для таблицы *Sotr*.

Create domain Pol \_type as char (3) collate PXW\_CYRL;

Create table Sotr (

Fio char (20) not null, Pol Pol\_type, otdel char (10), dolj char (20), primary key (Fio) );

где *Not null* – столбцы ассоциированные с доменом обязательно должны содержать значение.

**34.2 Ограничения на значения домена**

Записываются в определении столбца и начинаются со слова *check*.

<огр\_домена>={

value <оператор> <значение>

| value [not] between <значение 1> and <значение 2> | value [not] like <значение> [escape <значение>]

| value [not] in ( <значение 1> [, <значение 2>…] ) | value is [not] null

| value [not] containing <значение> | not <огр\_домена>

| <огр\_домена> or < огр\_домена> | <огр\_домена> and <огр\_домена>

}

где:

<оператор> = {= |<|>|<=|=>|!<|>!|<>|!=} !< – не меньше, !> – не больше, != – не равно.

*Value* означает,что элементы считаются правильными.

*Create domain ID*\_*Type as integer check* (*value* >=100); *id*\_*Type* >= 100.

*Between* <значение1> *and* <значение2> –значение домена в интервале от значения1до значения2,включая их.

*Like* <значение1> [<значение2>] –значение домена должно"походить"на значение1.

*Like* "%*USD*" –вводимое значение должно оканчиваться на *USD*,все предыдущие значения не име-

ют.

*Like* "\_\_04" ("\_"–единичный символ)вводится значение четырьмя символами,два последних– 04.Если "%" и "\_" нужны как символы в *Like*, то указываются значения в *Escape* и заменяются другими

символами.

Например, *Summa* должна заканчиваться %⇒

*Create domain summa as char* (10) *check* (*like* "%!%" *escape* "!");

После символа ! служебные символы (%) теряют свою силу и становятся обычными символами.

*In* (<значение1>[,<значение2>…]) –значение домена должно совпадать с одним из приведенныхпараметров списка.

*Create domain Pol*\_*type as*

*char* (3) *check* (*value in* ("муж","жен"));

*Containing* <значение> –значение домена должно иметь вхождения параметра<значение>в любомместе.

Например, в наименовании отдела вхождения 041 где угодно "отдел – 041002" или "00304192" и т.

д.

*Create domain otdel*\_*type as*

*varchar* (10) *check* (*value containing* "041" *collate PXW*\_*CYRL*;

*Starting* [*with*] <значение> –значение домена должно начинаться с<значение>,например"041".Большинство условий могут комбинироваться *AND* или *OR* или указывать *NOT*.

*Check* (*value not between* 1 *and* 100);

и т.д.

**34.3 Изменения домена**

Alter domain имя {

[set default {литерал | null | user}] | drop default]

| [add [constraint] check (<огр\_домена>)] | [drop constraint]

};

Для столбца *not null* определение уже нельзя изменить и тип данных тоже. *Set default* –по умолчанию;

*Drop default* –отменяет значение по умолчанию;

*Add* [*constraint*] *check* (<огр\_домена…>)добавляет условие на значения столбцов;

*Drop constraint* –удаляет условия.

Пример

Create domain id\_type as

integer check (value >=100);

create table A (

id id\_type not null, fio varchar (20), primary key (id));

Изменить ограничение на 100 ≤ *x* ≤ 500

1. удалить старое условие *alter domain id*\_*type drop constraint*;
2. добавить новое условие *alter domain id*\_*type check* (*value* >= 100 *and value* <=500).

**35 СОЗДАНИЕ ТАБЛИЦ**

**35.1 Оператор *create table***

Перед созданием таблиц БД необходимо продумать определения всех столбцов таблицы и характе-ристик каждого столбца. При определении таблицы применяются домены. БД, в которую добавляется таблица, должна быть открыта.

Создание таблицы БД осуществляется оператором

Create table имя\_табл [external [file] "<имя файла>"] (<опр\_ст>[, < опр\_ст > | <ограничение>…]);

[external [file] "<имя файла>" – относится к внешним таблицам БД. <опр\_ст> – определение столбца БД имеет формат:

<опр\_ст>= имя\_ст {тип\_данных | computed [BY] (<выражение>) | домен}

[default {литерал | null | user}] [not\_null] [<огранич\_столбца>] [collate collation]

computed [by] (<выражение>) – определение столбца вычисляемых значений. Default – значение столбца по умолчанию, ассоциировано с доменом. <огранич\_столбца> – ограничение на значение столбца.

Collate collation – порядок сортировки символов.

**35.2 Столбцы вычисляемых значений**

Значение таких столбцов не вводится, а вычисляется согласно выражению, например в таблице *А* есть столбцы номера квартала *N*\_*Q*, количество продаж в данном квартале в прошлом году *kol*\_*s* и те-кущем году *kol*\_*n* и прирост продаж за квартал *prirost*:

*Create table A* ( *N*\_*Q integer not null*, *Kol*\_*s integer*, *Kol*\_*n integer*,

*prirost computed by* ( *kol*\_*n* – *kol*\_*s*), *primary key* (*N*\_*Q*));

**35.3 ОГРАНИЧЕНИЯ ЦЕЛОСТНОСТИ**

Бывают двух уровней, на уровне столбца или на уровне всей таблицы. Наложение ограничения це-лостности на отдельный столбец следует за его именем и типом:

*Tovar varchar* (20) *not null primary key*, *create table*… (

*tovar varchar* (20) *not null*,

…,

*primary key* (*tovar*));

*primary key* –это первичный ключ построений по столбцу или столбцам.

Столбцы должны иметь значение *not null*. Первичный ключ служит для установления связи в внеш-ним ключом (*foreign key*) дочерней таблицы и определяет ссылочную целостность между родительской и дочерней таблицами.

**36.1 Уникальный ключ**

Строится по столбцу (столбцам ) когда столбец не входит в первичный ключ и имеет уникальное значение /нет одинаковых значений/

*Create table klient* (

*Imja*\_*klienta varchar* (20) *not null primary key*, *Nom*\_*scheta varchar* (50) *not null*,

*Unique* (*nom*\_*scheta*));

Или *nom*\_*scheta varchar* (50) *not null unique*.

**36.2 Внешний ключ**

Строится в дочерней таблице для соединения с родительской. Форматы:

*Foreign key* (<список\_ст\_внешнего\_ключа>) *references* <имя\_род\_табл>[<*c*писок\_ст\_род\_табл>]

[*on delete* {*no action* | *cascade* | *set default* | *set null*}] [*on update* {*no action* | *cascade* | *set default* | *set null*}]

список\_ст\_внешнего\_ключа – столбцы дочерней таблицы;

имя\_род\_табл – таблица, в которой описан первичный ключ (или столбец с атрибутом *unique*); список\_ст\_род\_табл – не обязателен при ссылке на первичный ключ родительской таблицы, в дру-

гих случаях необходим.

*On delete*, *on update* –способ изменения записей дочерней таблицы при удалении или измененииполя связи в родительской таблице.

*No action* –запрет;

*Cascade* –название;

*Set default* –в поле дочерней таблицы заносится значение,определенное ранее по умолчанию; *Set null* –заносится значение *null*.

*Пример*:определим две таблицы.

Родительская *detal* с полями *name*\_*det* (имя детали) и *zena*\_*ed* (цена за единицу), первичный ключ по полю *name*\_*det*.

Дочерняя *prihod* (приход со склада) с полями *n*\_*prihod* (номер прихода), *date*\_*prihod* (дата прихода ), *name*\_*det* (имя детали), *kolvo* (количество деталей в приходе).Первичный ключ по *n*\_*prihod*,внешний– *name*\_*det*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *Detal* |  | *Prihod* |  |  |
| *Name*\_*det Zena*\_*ed* | | *N*\_*prihod* | *Date*\_*prihod Name*\_*det kolvo* | | |
| … | … | … | … | … | … |
| … | … | … | … | … | … |

*Create table detal* (

*Name*\_*det varchar* (20) *not null*1, *Zena*\_*ed integer not null*, *primary key* (*name*\_*det*));

*Create table prihod* (

*N*\_*prihod integer not null primary key*, *Date*\_*prihod date not null*,

*Name*\_*det varchar* (20) *not null*1,

1 Должны быть описаны одинаково.

*Kolvo integer not null*

*Foreign key* (*name*\_*det*) *references detal*).

**36.3 Именование ссылочной целостности**

Ссылочная целостность может именоваться следующим образом: [*constraint* <имя ссылочной целостности>]

*foreign key* (<список столбцов внешнего ключа>) *references* <имя родительской таблицы>[<список столбцов родительской таблицы>]

Необязательное имя ссылочной целостности присутсвует в смешанных сообщениях и может ис-пользоваться при анализе БД.

Для конкретного примера в таблицу *prihod* добавим:

*Kolvo integer not null*, *Constraint po*\_*detaly*

*foreign key* (*name*\_*det*) *references detal*;

**37.1 Требования к значениям столбцов**

Определяются как на уровне отдельного столбца так и на уровне всей таблицы.

Например, для таблицы параметров человека (*parametr*) рост (*rost*) должен быть больше веса (*ves*). Ограничения можно записать:

*Create table parametr* (

*Id integer not null primary key*, *Rost integer not null*,

*Ves integer not null check* (*rost* > *ves*));

или

*Create table parametr* ( *Id integer not null*, *Rost integer not null* , *Ves integer not null*, *primary key* (*id*),

*check* (*rost* > *ves*));

Ограничения, накладываемые на столбец, определяются предложением *check*, имеющим формы: *Check* (<условия\_поиска>)

где

<условия\_поиска> = {<значение><оператор>{<значение1 > | (<выбор\_одного>)} | <значение> [*not*] *between* <значение 1> *and* <значение 2>

| <значение> [*not*] *like* <значение> [ *escape* <значение>]

| <значение> [*not*] *in* <значение 1>[, <значение 2>…] | <выбор\_многих>) | <значение> *is* [*not*] *null*

| <значение> {[*not*] {=| < | > } >= | <= } {*all* | *some* | *any* } (<выбор многих>) | *exists* (<выражение\_выбора>)

| *singular* (<выражение\_выбора>)

| <значение> [*not*] *containing* <значение 1>

| <значение> [*not*] *starting* [*with*] <значение 1> | (<условие\_поиска>)

| *not* (<условие\_поиска>)

| <условие\_поиска> *or* <условие\_поиска> | <условие\_поиска> *and* <условие\_поиска>

где

<значение >={столбец | <константа> | <выражение> | <функция> | *null* | *user* | *RDB*$*DB*\_*key* константа = число | "строка"

функция = {

*count* (\* | [*all*] <значение> | *distinct* <значение>)

*sum* ([*all*] <значение> | *distinct* <значение>) | *avg* ([*all* ] <значение> | *distinct* <значение>) *max*([*all*] <значение> | *distinct* <значение>) *min*([*all*] <значение> | *distinct* <значение>) *cast* (<значение> *as* <тип\_данных>)

*upper* (<значение>)

| *gen*\_*id* (генератор, <значение>)

}

выбор\_одного – *select* возвращает одно значение;

выбор\_многих – *select* возвращает список или ни одного выражения выбора *select;* *singular* – *true* для списка из одной строки;

*exists* – *true* если вписок не пустой.

**38 ИЗМЕНЕНИЕ ТАБЛИЦЫ**

**38.1 Оператор *alter table***

Команда **ALTER TABLE** не часть стандарта ANSI; но это – широко доступная, и довольно содер-жательная форма, хотя ее возможности несколько ограничены. Она используется, чтобы изменить оп-ределение существующей таблицы. Обычно, она добавляет столбцы к таблице. Иногда она может уда-лять столбцы или изменять их размеры, а также в некоторых программах добавлять или удалять огра-ничения. Типичный синтаксис чтобы добавить столбец к таблице:

alter table <имя\_табл> ADD <имя\_столбца> <data type> <size>;

Столбец будет добавлен со значением NULL для всех строк таблицы. Новый столбец станет по-следним по порядку столбцом таблицы . Вообще то, можно добавить сразу несколько новых столбцов, отделив их запятыми, в одной команде. Имеется возможность удалять или изменять столбцы. Наиболее часто, изменением столбца может быть просто увеличение его размера, или добавление (удаление) ог-раничения. Ваша система должна убедиться, что любые изменения не противоречат существующим данным – например, при попытке добавить ограничение к столбцу, который уже имел значение, при на-рушении которого ограничение будет отклонено. Лучше всего дважды проверить это.

Оператор *alter table* позволяет:

− добавить определение нового столбца; − удалить столбец из таблицы;

− удалить атрибуты целостности таблицы или отдель-ного столбца;

− добавить новые атрибуты целостности. Пусть имеем таблицу:

*Create table sotr* (

*Id*\_*sotr integer not null primary key*, *Fio char* (10),

*Otdel varchar* (10), *Doljnost char* (10));

Пусть необходимо изменить характеристики столбца *fio c char* (10)на *varchar*(25):

1. добавим в таблицу новый временный столбец *fio*\_*tmp*: *alter table sotr*

*add fio*\_*tmp char* (10);

1. копируем данные из *fio* в *fio*\_*tmp*:

*update sotr*

*set fio*\_*tmp* = *fio*;

1. удаляем столбец *fio*: *alter table sotr*

*drop fio*;

1. создаем новый стобец *fio*: *alter table sotr*

*add fio varchar* (25);

1. переписываем данные: *update sotr*

*set fio* = *fio*\_*tmp*;

1. удаляем временный столбец: *alter table sotr*

*drop fio*\_*tmp*.

**38.2 Изменение атрибутов столбца**

Добавление нового столбца в таблицу БД:

*Alter table* <имя таблицы> *add* <определение столбца>;Добавление новых ограничений целостности:

*Alter table* <имя таблицы> *add* [*constraint* <имя ограничения>] <определение целостности>;Удаление столбца (столбцов) из таблицы:

*Alter table* <имя таблицы> *drop* <имя столбца1> [, <имя столбца2>…];Удаление ограничений целостности (уровень таблицы):

*Alter table* <имя таблицы> *drop* <ограничения целостности>;Пример. Для таблицы *prihod*:

*Create table prihod* (

*N*\_*prihod integer not null primary key*, *Date*\_*prihod date not null*,

*Name*\_*det varchar* (20) *not null*, *Kolvo integer not null Constraint po*\_*tovaru*

*foreign key* (*name*\_*det*) *references detal*);

Удалить целостность *po\_tovaru*:

*Alter table prihod drop po*\_*tovaru*.

**38.3 Удаление таблицы**

Удаление таблицы целиком: *Drop table* <имя таблицы>;

Удаление может быть блокировано для родительских таблиц, если есть дочерние. Удаление табли-цы разрушит ссылочную целостность.

**39 РАБОТА С ИНДЕКСАМИ**

Первичный и внешний ключи строятся для обеспечения ссылочной целостности реляционно-связанных таблиц. Кроме этого , первичный ключ, выполняет функции уникальности своих значений. Для этих же целей используется и просто уникальный ключ.

Индексы, в отличие от ключей, создаваемые оператором *create index*, служат для сортировок и оп-тимизации доступа к данным. В конечном счете, ключи и индексы преобразуются в *физические индексы*

– специальный механизм быстрого доступа к данным.

**39.1 Необходимость создания индексов**

Создаются в случаях, когда

1-часто производится поиск в БД (столбец (столбцы)) часто перечисляется в предложении *where* оператора *select*);

2-часто строится объединение таблиц;

3-часто производится сортировка (*order by* в операторе *select*).

Не рекомендуется строить индексы по столбцам или группами столбцов, которые:

4-редко используются для поиска;

5-часто меняют значение (надо часто обновлять индекс);

6-содержит небольшое число вариантов значения.

**39.2 Создание индекса, оператор *create index***

Оператор

*Create* [*unique*] [*asс*[*ending*] | *desс* [*ending*]]

*Index* <имя индекса> *on* <имя таблицы> (столбец1,…); *Unique* –уникальный индекс,не допускает одинаковых значений

*Asс*[*ending*] –сортировка полей индекса по возрастанию(по умолчанию) *desс*[*ending*] –сортировка полей индекса по убыванию.

Пример. Для таблицы *prihod*:

*Create table prihod* (

*N*\_*prihod integer not null primary key*, *Date*\_*prihod date not null*,

*Name*\_*det varchar* (20) *not null*, *Kolvo integer not null*);

Создать индекс в порядке убывания значений *date*\_*prihod* и *name*\_*det*:

*Create desc index D*\_*P*

*On prihod* (*date*\_*prihod*, *name*-*det*);

После многократного внесения изменений в таблицу индексы могут быть разбалансированы, "глу-бина" индекса возрастает.

Это приводит к увеличению времени поиска. Необходимо время от времени:

1. выполнять балансировку индекса оператором *alter index*;
2. переписывать выбираемость индекса оператором *set statistics*;
3. уничтожать и вновь создавать индекс операторами *drop index* и *create index*.

**39.3 Пересоздание и балансировка индекса**

1. *alter index* <имя индекса> *deactivate*;
2. *alter index* <имя индекса> *activate*;

Нельзя:

1. перестроить индекс, находящийся в запросах,
2. перестроить индекс, построенный по первичному, внешнему ключу.

**39.4 Удаление индекса**

*Drop index* <имя индекса>;

**40 ЗАПРОСЫ В SQL**

**40.1 Оператор *select***

Позволяет производить выборки данных, преобразовывать полученные результаты, реализует сложные условия выбора.

Формат оператора:

*Select* [ *distinct* | *all* {\* | <значение1> [, <значение2>… ]}

*from* <таблица1 > [, <таблица2 >… ][ *where* <условие\_поиска>]

[ *group by* столбец [*collate collation*] [, столбец 1 [*collate collation*]…]

[ *having* <условия\_поиска>] [ *union* <оператор\_*select*>]

[ *plan* <план\_выполнения\_запроса>] [ *order by* <список\_столбцов>];

В простейшем случае, когда требуется просмотреть все записи одной или нескольких таблиц, опе-ратор имеет вид:

*Select* {\* | <значение1> [, <значение2>…]} *from* <таблица1> [, <таблица2>…];

<значение 1>, <значение 2>… – имя столбца возвращаемого оператором, \* – все столбцы <таблица 1>, <таблица 2>… – имя таблицы, из которой происходит выборка данных. Например, создать набор данных, состоящий из всех столбцов:

*Select* \* *from prihod*;

Такой же набор данных можно получить:

*Select n*\_*prihod*, *date*\_*prihod*, *name*\_*det*, *kolvo*, *from prihod*;

Предложение *where* используется для включения в БД лишь нужных записей, удовлетворяющих ус-ловию:

*Select* { \* | <значение1> [, <значение2>… ]} *from* <таблица1> [, <таблица2>…]

*where* <условия поиска>;

**40.2 Сравнение с константой**

При сравнении значения столбца с константой условие поиска имеет вид: <имя\_столбца> < оператор> < константа> <оператор> – =, <, >, <= (!>), >=(!<), <>(!=).

Например, выбрать из таблицы *prihod* все операции приема товара объемом 20 единиц:

*Select* \* *from prihod*

*where kolvo*=20;

**40.3 Внутреннее соединение таблиц**

При сравнении значения столбца одной таблицы со значением столбца другой таблицы условие по-иска имеет вид:

<имя\_столбца\_табл1> <оператор> <имя\_столбца\_табл2>.

Например, выбрать все записи о приходе деталей из таблицы *prihod* и для каждой детали указать его цену из таблицы *detal*:

*Select prihod* .\*, *detal*.*zena*\_*ed from prihod*, *detal*

*where prihod*.*name*\_*det* = *detal*.*name*\_*det*;

Для каждой записи из таблицы *prihod* ищется запись в таблице *detal*, у которой значение в поле *name*\_*det* совпадает со значением *name*\_*det* текущей записи таблицы *prihod*.Порядок перечисления вусловии поиска значения не имеет:

*Prihod*.*name*-*det* = *detal*.*name*\_*det*

или

*Detal*.*name*\_*det* = *prihod*.*name*\_*det*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *N*\_*priho* | *Date*\_*prih* | *Name*\_*de* | *Kolv* | *Name*\_ | *Zena*\_*e* |
| *d* | *od* | *t* | *o* | *det* | *d* |
| 1 | 20.09.00 | *D*1 | 100 | *D*1 | 1.00 |
| 2 | 29.09.00 | *D*1 | 100 | *D*2 | 50.00 |
| 3 | 1.10.00 | D*2* | 200 | *D*3 | 30.00 |
| 4 | 1.10.00 | *D*1 | 200 | *D*4 | 100.00 |
| 5 | 2.10.00 | *D*2 | 200 | *D*5 | 35.00 |
| 6 | 4.10.00 | *D*3 | 100 | *D*6 | 50.00 |
| 7 | 4.10.00 | *D*3 | 110 | *D*7 | 55.00 |

**41.1 Использование псевдонимов таблиц**

Идентификация столбцов через имя таблицы неудобно из-за громоздкости обозначений. Лучше при-своить каждой таблице краткое имя.

Такие имена называются *псевдонимами* таблиц. Они отделяются пробелом от фактического имени таблицы в списке *from*:

*Select* …

*from* <таблица1псевдоним1> [, <таблица2псевдоним2> …] *where* … ;

Например:

*Select prihod*.\*, *detal*.*zena*\_*ed from prihod P*, *detal D*

*where P*.*name*-*det* = *D*.*name\_det*;

**41.2 Определение сортировки** *order by*

Результирующий HД можно упорядочить с помощью предложения: *Order by* <список\_столбцов>.

Если в списке столбцов указано больше одного столбца, то первый будет использоваться для гло-бальной сортировки, второй – для сортировки внутри группы, определенной единым значением первого столбца, и т.д.

Например, показать все записи приема деталей, отранжировать по имени детали:

*Select name*\_*det*, *zena*\_*ed from detal*

*where zena*-*ed* >= 35; *order by name*\_*det*;

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Name*\_*det* | *Zena*\_*ed* |  |
|  | *D*2 | 50.00 |  |
|  | *D*4 | 100.06 |  |
|  | *D*5 | 35.00 |  |
|  | *D*6 | 50.00 |  |
|  | *D*7 | 55 |  |

или

*Select n*\_*prihod*, *date*\_*prihod*, *name*\_*det*, *kolvo from prihod*

*order by name\_det*, *kolvo*, *date*\_*prihod*;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *N*\_*priho* | *Date*\_*prihod* | *Name*\_*det* | *Kolvo* |  |
| *d* |  |
|  |  |  |  |
| 1 | 20.09.00 | *D*1 | 100 |  |
| 2 | 29.09.00 | *D*1 | 100 |  |
| 3 | 4.10.00 | *D*1 | 100 |  |
| 4 | 1.10.00 | *D*1 | 200 |  |
| 5 | 1.10.00 | *D*2 | 200 |  |
| 6 | 2.10.00 | *D*2 | 200 |  |
| 7 | 4.10.00 | *D*3 | 110 |  |

**42.1 Расчет вычисляемых столбцов**

Ключевое слово *Distinct*. Повторяющимися считаются записи, содержащие идентичные значения во всех столбцах результирующего HД. Если в результирующем HД нужно вносить все записи, то указы-вают ключевое слово *All* (по умолчанию *All*).

Например, получить наименование всех деталей, полученных на склад:

*Select distinct name*\_*det from prihod*;

**41.3 Устранение повторяющихся записей**

Для расчета вычисляемых столбцов результирующего HД используются арифметические выраже- ния:

*Select* [*distinct* | *аll* ] { \* | <столбец1> [, <выражение1>… ]} *from* <таблица1> [, <таблица2>… ];

Если столбцу надо присвоить нестандартное имя, то оно может быть указано за выражением при помощи ключевого слова *As*.

Например: рассчитать общую стоимость полученных деталей для каждого факта получения:

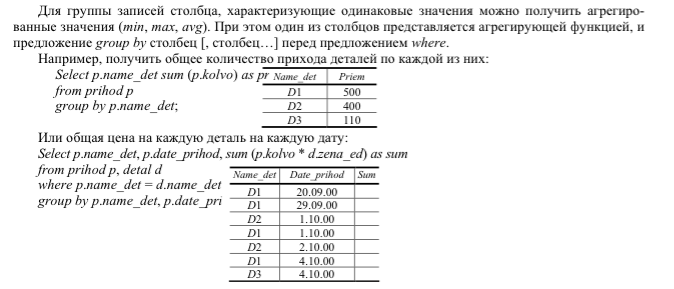
*Select p*.\*, *d*.*zena*\_*ed*, *p*.*kolvo* \* *d*.*zena*\_*ed as stoim*

*from prihod p*, *detal d*

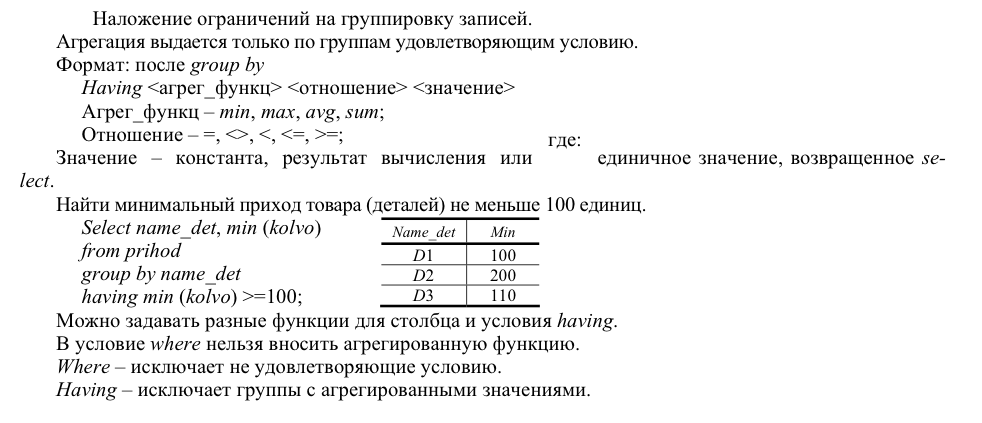
*where p*.*name*\_*det* = *d*.*name*\_*det*;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *N*\_*prih* | *Date*\_*prih* | *Name*\_ | *Kolvo* | *Zena*\_*e* | *Stoim* |  |
| *od* | *od* | *det* | *d* |  |
|  |  |  |
| 1 | 20.09.00 | *D*1 | 100 | 1.00 | 100.00 |  |
| 2 | 29.09.00 | *D*1 | 100 | 1.00 | 100.00 |  |
| 3 | 1.10.00 | *D*2 | 200 | 50.00 | 10000. |  |
| 00 |  |
|  |  |  |  |  |  |
| 4 | 1.10.00 | *D*1 | 200 | 1.00 | 200.00 |  |
| 5 | 2.10.00 | *D*2 | 200 | 50.00 | 10000. |  |
| 00 |  |
|  |  |  |  |  |  |
| 6 | 4.10.00 | *D*1 | 100 | 1.00 | 100.00 |  |
| 7 | 4.10.00 | *D*3 | 110 | 30.00 | 3300.0 |  |
| 0 |  |
|  |  |  |  |  |  |

42.2 Группировка



**42.3 Предложение *having***



**43.1 Использование подзапросов**

Часто невозможно решить поставленную задачу путем использования единственного запроса. На-пример, в тех случаях, когда при использовании условия поиска в предложении *where* значение с кото-рым надо сравнить, далее не определено, а вычисляется оператором *select*.

В таких случаях применяют вложенные запросы или подзапросы. Оператор *select* имеет вид:

*Select* … *from* …

*where* <сравниваемое значение> <оператор> (*select*);

Пусть надо найти дату, на которую приходится максимальный приход деталей. Тогда запрос может быть записан так:

|  |  |
| --- | --- |
| *Kolvo* | *Date*\_*prihod* |
| 200 | 1.10.00 |
| 200 | 2.10.00 |

*Select kolvo*, *date*\_*prihod*

*from prihod*

*where kolvo* = ( *select max* (*kolvo*) *from prihod*);

Оператор *select* возвращает не одно значение, а список. Поэтому может возникнуть ошибка. Чтобы ее избежать надо заменить оператор = на оператор выбора из нескольких возможных значений (*in*).

Синтаксис вложенного запроса ничем не отличается от синтаксиса основного запроса. Это значит, что в подзапрос может быть вложен другой подзапрос и т.д.

Например, составим список получения деталей от поставщика, который в свое время поставил мак-симальную партию любой детали:

*Select p*0.\* *from prihod p*0 *where p*0.*post in* (*select p*1.*post*

*from prihod p*1 *where kolvo in* (*select max* (*p*2.*kolvo*)

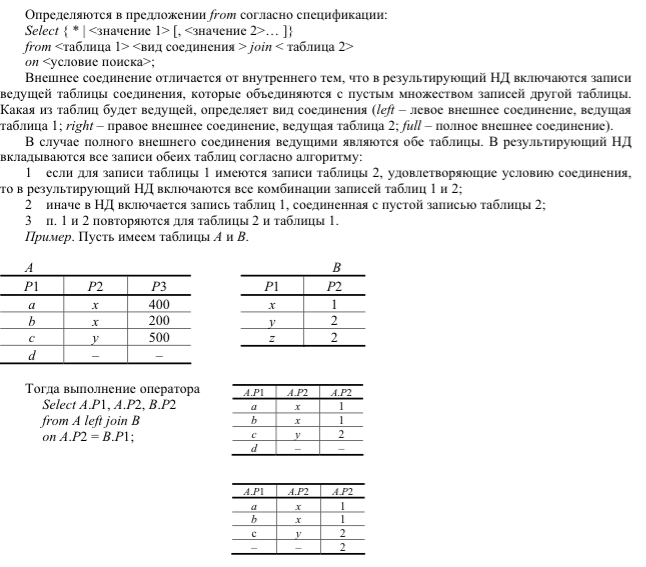
*from prihod p*2));

Сначала определим *max*, далее имя поставщика, а затем все записи, связанные с данным поставщи-ком.

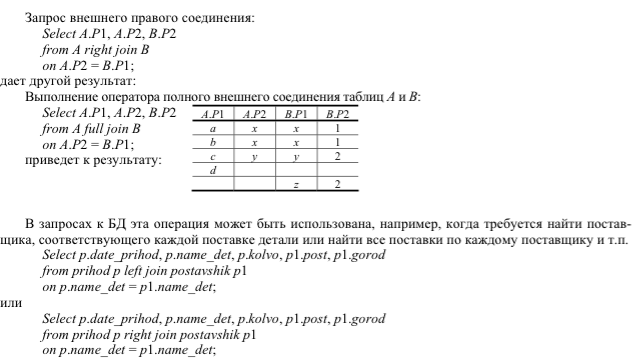
Таблица *prihod* имеет дополнительно столбцы поставщика:

*Post* –имя поставщика; *Gorod* –город поставщика; *Addr* –адрес поставщика.

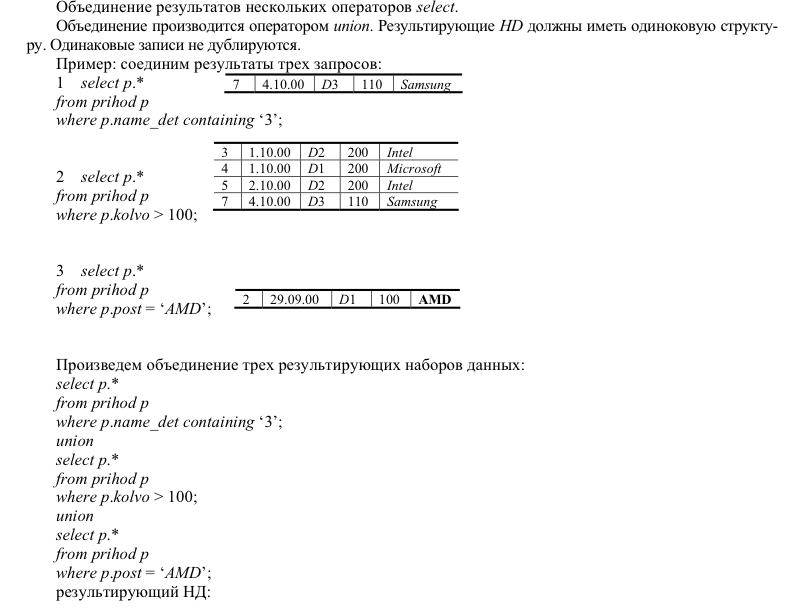
**43.2 Внешние соединения**



**9**



**44 Объединение результатов нескольких операторов Select**



**45 ДОБАВЛЕНИЕ, ИЗМЕНЕНИЕ, УДАЛЕНИЕ ЗАПИСЕЙ**

*Insert into* <объект>[ (столбец1 [,столбец2 …])]

{*values* (<значение 1> [, <значение 2>…]) | Формат оператора: < оператор *select*> };

Если значения присваиваются всем столбцам по порядку, то имена столбцов можно не писать, в противоположном случае миена столбцов надо указать.

*Insert into prihod*

*values* (8, "12-*oct*-2000", "*D*4", 50, "*AMD*");

Вместо явного указания номера прихода, можно воспользоваться генератором уникального ключа:

*Insert into prihod*

*values* (*gen*\_*ID* (*prihod*\_*n*\_*prihod*, 2), "12-*oct*-2000", "*D*4", 50, "*AMD*");

Применение оператора *select* не нарушает порядка присваивания значений столбцам. Пусть определена таблица *prihod*\_*data* аналогичная *prihod*:

*Create table prihod*\_*data* ( *N*\_*prihod integer not null*, *Date*\_*prihod date not null*, *Name*\_*det varchar* (20) *not null*, *Kolvo intreger not null*,

*Post varchar* (20), *Primary key* (*n*\_*prihod*));

Пусть в эту таблицу надо ежедневно копировать все записи о полученных деталях, например для передачи в бухгалтерию. Тогда выгрузка записей из *prihod* в таблицу *prihod*\_*data* будет реализована оператором:

*Insert into prihod*\_*data Select* \*

*from prihod*

*where date*\_*prihod* = "13-*oct*-2000";

Оператор *update* применяется для изменения значения в группе записей или одной записи. Формат:

*Update* <объект>

*set* столбец1 = <значение1> [,столбец2 = <значение2>… ][ *where* <условие поиска>]

Если опустить *where* <условие поиска>, то будут изменены все записи, в противном случае только записи, удовлетворяющие условию.

Например, заменить дату 10.10.00 и увеличить количество товара на 10 единиц для всех записей с датой 4.10.00 в таблице *prihod*:

*Update prihod*

*set date* \_*prihod* = "10-*oct*-2000", *kolvo* = *kolvo*+10,

*where date*\_*prihod* = "4-*oct*-2000";

Оператор *delete* применяется для удаления групп записей из объекта (таблиц или *view*). Формат:

*Delete from* <объект>[*where* <условие поиска>];

Удаляются все записи если нет *where* <условие поиска>, иначе только записи, удовлетворяющие условию.

Пример: удалить все записи детали *D*1 за 10.10.2000.

*Delete from prihod*

*where* (*name*\_*det* = "*D*1") *and*

(*date*\_*prihod* = "10-*oct*-2000");