**Казанский технологический колледж**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **РЕФЕРАТ** | | | | | |
|  | | | | | |
| По дисциплине | | **Технология разработки и защиты баз данных** | | | |
|  | | | | | |
| Студент | | Козынбаев Александр Дмитриевич | | | |
| (фамилия, имя, отчество) | | | | | |
| Группа | 903 | | | |  |
|  | (индекс группы) | | | |  |
| Специальность: | | | 09.02.07 Информационные системы и программирование | | |
| (наименование специальности, уровень СПО) | | | | | |
| Преподаватель | | | | Гришин Сергей Вячеславович | |
| (фамилия, имя, отчество) | | | | | |

2022

Оглавление

[1. Язык SQL: соединение таблиц (внутреннее и внешнее) 3](#_Toc119445366)

[1.1. INNER JOIN (внутреннее соединение) 3](#_Toc119445367)

[1.2. LEFT OUTER JOIN (левое внешнее соединение) 5](#_Toc119445368)

[1.3. RIGHT OUTER JOIN (правое внешнее соединение) 6](#_Toc119445369)

[1.4. FULL OUTER JOIN (полное внешнее соединение) 7](#_Toc119445370)

[1.5. Псевдонимы соединяемых таблиц 8](#_Toc119445371)

[1.6. JOIN и соединение более двух таблиц 9](#_Toc119445372)

[2. Примитивные и не примитивные реляционные операции. 12](#_Toc119445373)

[2.1. Переименование 12](#_Toc119445374)

[2.2. Объединение 12](#_Toc119445375)

[2.3. Пересечение 13](#_Toc119445376)

[2.4. Вычитание 13](#_Toc119445377)

[2.5. Операция присваивания 13](#_Toc119445378)

[2.6. Декартово произведение 13](#_Toc119445379)

[2.7. Выборка (ограничение) 13](#_Toc119445380)

[2.8. Выборка (ограничение) 14](#_Toc119445381)

[2.9. Соединение 14](#_Toc119445382)

[2.10. Деление 14](#_Toc119445383)

1. **Язык SQL: соединение таблиц (внутреннее и внешнее)**

Оператор языка SQL JOIN предназначен для соединения двух или более таблиц базы данных по совпадающему условию. Этот оператор существует только в реляционных базах данных. Именно благодаря JOIN реляционные базы данных обладают такой мощной функциональностью, которая позволяет вести не только хранение данных, но и их, хотя бы простейший, анализ с помощью запросов.

Основные нюансы написания SQL-запросов с оператором JOIN, которые являются общими для всех СУБД (систем управления базами данных).

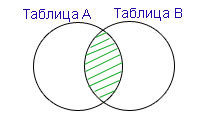
Для соединения двух таблиц оператор SQL JOIN имеет следующий синтаксис:

SELECT ИМЕНА\_СТОЛБЦОВ (1..N)

FROM ИМЯ\_ТАБЛИЦЫ\_1 JOIN ИМЯ\_ТАБЛИЦЫ\_2 ON УСЛОВИЕ

После одного или нескольких звеньев с оператором JOIN может следовать необязательная секция WHERE или HAVING, в которой, также, как в простом SELECT-запросе, задаётся условие выборки. Общим для всех СУБД является то, что в этой конструкции вместо JOIN может быть указано INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN, CROSS JOIN (или, как вариант, запятая).

* 1. INNER JOIN (внутреннее соединение)

Запрос с оператором INNER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON.

То же самое делает и просто JOIN. Таким образом, слово INNER - не обязательное.

Есть база данных портала объявлений - 2. В ней есть таблица Categories (категории объявлений) и Parts (части, или иначе - рубрики, которые и относятся к категориям).

Например, части Квартиры, Дачи относятся к категории Недвижимость, а части Автомобили, Мотоциклы - к категории Транспорт.

Таблицы этой базы данных с заполненными данными имеют следующий вид.

Таблица Categories:

|  |  |  |
| --- | --- | --- |
| Catnumb | Cat\_name | Price |
| 10 | Стройматериалы | 105,00 |
| 505 | Недвижимость | 210,00 |
| 205 | Транспорт | 160,00 |
| 30 | Мебель | 77,00 |
| 45 | Техника | 65,00 |

Таблица Parts:

|  |  |  |
| --- | --- | --- |
| Part\_ID | Part | Cat |
| 1 | Квартиры | 505 |
| 2 | Автомашины | 205 |
| 3 | Доски | 10 |
| 4 | Шкафы | 30 |
| 5 | Книги | 160 |

**Пример 1.** Требуется соединить данные этих двух таблиц так, чтобы в результирующей таблице были поля Part (Часть), Cat (Категория) и Price (Цена подачи объявления) и чтобы данные полностью пересекались по условию. Условие - совпадение номера категории (Catnumb) в таблице Categories и ссылки на категорию в таблице Parts. Для этого пишем следующий запрос:

SELECT Parts.Part, Categories.Catnumb AS Cat, Categories.Price

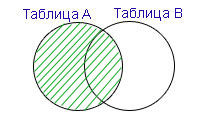
FROM Parts INNER JOIN Categories

ON Parts.Cat = Categories.Catnumb

Результатом выполнения запроса будет следующая таблица:

|  |  |  |
| --- | --- | --- |
| Part | Cat | Price |
| Квартиры | 505 | 210,00 |
| Автомашины | 205 | 160,00 |
| Доски | 10 | 105,00 |
| Шкафы | 30 | 77,00 |

* 1. LEFT OUTER JOIN (левое внешнее соединение)

Запрос с оператором LEFT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из первой по порядку (левой) таблицы, даже если они не соответствуют условию. У записей левой таблицы, которые не соответствуют условию, значение столбца из правой таблицы будет NULL (неопределённым).

**Пример 2.**База данных и таблицы - те же, что и в примере 1.

Для получения результирующей таблицы, в которой данные из двух таблиц полностью пересекаются по условию и дополняются всеми данными из таблицы Parts, которые не соответствуют условию, пишем следующий запрос:

SELECT Parts.Part, Categories.Catnumb AS Cat, Categories.Price

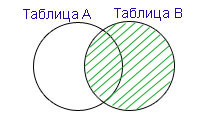
FROM Parts LEFT OUTER JOIN Categories

ON Parts.Cat = Categories.Catnumb

Результатом выполнения запроса будет следующая таблица:

|  |  |  |
| --- | --- | --- |
| Part | Cat | Price |
| Квартиры | 505 | 210,00 |
| Автомашины | 205 | 160,00 |
| Доски | 10 | 105,00 |
| Шкафы | 30 | 77,00 |
| Книги | 160 | NULL |

* 1. RIGHT OUTER JOIN (правое внешнее соединение)

Запрос с оператором RIGHT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из второй по порядку (правой) таблицы, даже если они не соответствуют условию. У записей правой таблицы, которые не соответствуют условию, значение столбца из левой таблицы будет NULL (неопределённым).

**Пример 3.**База данных и таблицы - те же, что и в предыдущих примерах.

Для получения результирующей таблицы, в которой данные из двух таблиц полностью пересекаются по условию и дополняются всеми данными из таблицы Categories, которые не соответствуют условию, пишем следующий запрос:

SELECT Parts.Part, Categories.Catnumb AS Cat, Categories.Price

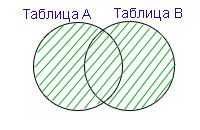
FROM Parts RIGHT OUTER JOIN Categories

ON Parts.Cat = Categories.Catnumb

Результатом выполнения запроса будет следующая таблица:

|  |  |  |
| --- | --- | --- |
| Part | Cat | Price |
| Квартиры | 505 | 210,00 |
| Автомашины | 205 | 160,00 |
| Доски | 10 | 105,00 |
| Шкафы | 30 | 77,00 |
| NULL | 45 | 65,00 |

* 1. FULL OUTER JOIN (полное внешнее соединение)

Запрос с оператором FULL OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из первой (левой) и второй (правой) таблиц, даже если они не соответствуют условию. У записей, которые не соответствуют условию, значение столбцов из другой таблицы будет NULL (неопределённым).

**Пример 4.**База данных и таблицы - те же, что и в предыдущих примерах.

Для получения результирующей таблицы, в которой данные из двух таблиц полностью пересекаются по условию и дополняются всеми данными как из таблицы Parts, так и из таблицы Categories, которые не соответствуют условию, пишем следующий запрос:

SELECT Parts.Part, Categories.Catnumb AS Cat, Categories.Price

FROM Parts FULL OUTER JOIN Categories

ON Parts.Cat = Categories.Catnumb

Результатом выполнения запроса будет следующая таблица:

|  |  |  |
| --- | --- | --- |
| Part | Cat | Price |
| Квартиры | 505 | 210,00 |
| Автомашины | 205 | 160,00 |
| Доски | 10 | 105,00 |
| Шкафы | 30 | 77,00 |
| Книги | 160 | NULL |
| NULL | 45 | 65,00 |

* 1. Псевдонимы соединяемых таблиц

В предыдущих запросах мы указывали с названиями извлекаемых столбцов из разных таблиц полные имена этих таблиц. Такие запросы выглядят громоздко: одно и то же слово повторяется несколько раз. Для этого следует использовать псевдонимы таблиц - их сокращённые имена. Псевдоним может состоять и из одной буквы. Возможно любое количество букв в псевдониме, главное, чтобы запрос после сокращения был понятен Вам самим.

Общее правило: в секции запроса, определяющей соединение, то есть вокруг слова JOIN нужно указать полные имена таблиц, а за каждым именем должен следовать псевдоним таблицы.

**Пример 5.**Переписать запрос из примера 1 с использованием псевдонимов соединяемых таблиц.

Запрос будет следующим:

SELECT P.Part, C.Catnumb AS Cat, C.Price

FROM Parts P INNER JOIN Categories C

ON P.Cat = C.Catnumb

Запрос вернёт то же самое, что и запрос в примере 1, но он гораздо компактнее.

* 1. JOIN и соединение более двух таблиц

Реляционные базы данных должны подчиняться требованиям целостности и не избыточности данных, в связи с чем данные об одном бизнес-процессе могут содержаться не только в одной, двух, но и в трёх и более таблицах.

В этих случаях для анализа данных используются цепочки соединённых таблиц: например, в одной (первой) таблице содержится некоторый количественный показатель, вторую таблицу с первой и третьей связывают внешние ключи - данные пересекаются, но только третья таблица содержит условие, в зависимости от которого может быть выведен количественный показатель из первой таблицы.

И таблиц может быть ещё больше. При помощи оператора SQL JOIN в одном запросе можно соединить большое число таблиц. В таких запросах за одной секцией соединения следует другая, причём каждый следующий JOIN соединяет со следующей таблицей таблицу, которая была второй в предыдущем звене цепочки.

Таким образом, синтаксис SQL запроса для соединения более двух таблиц следующий:

SELECT ИМЕНА\_СТОЛБЦОВ (1..N)

FROM ИМЯ\_ТАБЛИЦЫ\_1 JOIN ИМЯ\_ТАБЛИЦЫ\_2

ON УСЛОВИЕ

JOIN ИМЯ\_ТАБЛИЦЫ\_3

ON УСЛОВИЕ

...

JOIN ИМЯ\_ТАБЛИЦЫ\_M

ON УСЛОВИЕ

**Пример 6.**База данных - та же, что и в предыдущих примерах. К таблицам Categories и Parts в этом примере добавится таблица Ads, содержащая данные об опубликованных на портале объявлениях. Приведём фрагмент таблицы Ads, в котором среди записей есть записи о тех объявлениях, срок публикации которых истекает 2018-04-02.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A\_Id | Part\_ID | Date\_start | Date\_end | Text |
| 21 | 1 | '2018-02-11' | '2018-04-20' | "Продаю..." |
| 22 | 1 | '2018-02-11' | '2018-05-12' | "Продаю..." |
| ... | ... | ... | ... | ... |
| 27 | 1 | '2018-02-11' | '2018-04-02' | "Продаю..." |
| 28 | 2 | '2018-02-11' | '2018-04-21' | "Продаю..." |
| 29 | 2 | '2018-02-11' | '2018-04-02' | "Продаю..." |
| 30 | 3 | '2018-02-11' | '2018-04-22' | "Продаю..." |
| 31 | 4 | '2018-02-11' | '2018-05-02' | "Продаю..." |
| 32 | 4 | '2018-02-11' | '2018-04-13' | "Продаю..." |
| 33 | 3 | '2018-02-11' | '2018-04-12' | "Продаю..." |
| 34 | 4 | '2018-02-11' | '2018-04-23' | "Продаю..." |

Представим, что сегодня '2018-04-02', то есть это значение принимает [**функция CURDATE() - текущая дата**](https://function-x.ru/sql_current_date.html). Требуется узнать, к каким категориям принадлежат объявления, срок публикации которых истекает сегодня.

Названия категорий есть только в таблице CATEGORIES, а даты истечения срока публикации объявлений - только в таблице ADS. В таблице PARTS - части категорий (или проще, подкатегории) опубликованных объявлений. Но внешним ключом Cat\_ID таблица PARTS связана с таблицей CATEGORIES, а таблица ADS связана внешним ключом Part\_ID с таблицей PARTS.

Поэтому соединяем в одном запросе три таблицы и этот запрос можно с максимальной корректностью назвать цепочкой.

Запрос будет следующим:

SELECT C.Cat\_name FROM Categories C JOIN Parts P

ON P.Cat=C.Catnumb JOIN ads A ON A.Part\_id=P.Part\_id

WHERE A.Date\_end=CURDATE()

|  |
| --- |
| Cat\_name |
| Недвижимость |
| Транспорт |

Результат запроса - таблица, содержащая названия двух категорий - "Недвижимость" и "Транспорт":

1. Примитивные и не примитивные реляционные операции.

Реляционная алгебра — замкнутая система операций над отношениями в реляционной модели данных. Операции реляционной алгебры также называют реляционными операциями.

Первоначальный набор из 8 операций был предложен Э. Коддом в 1970-е годы и включал как операции, которые до сих пор используются (проекция, соединение и т. д.), так и операции, которые не вошли в употребление (например, деление отношений).

В процессе развития реляционной теории и практики было предложено несколько новых реляционных операций, например полусоединение (SEMI-JOIN) и полуразность, или анти-полусоединение (ANTI-SEMI-JOIN), CROSS APPLY и OUTER APPLY, транзитивное замыкание (TCLOSE) и др.

Поскольку многие операции выразимы друг через друга, в составе реляционной алгебры можно выделить несколько вариантов базиса (набора операций, через который выразимы все остальные). Наиболее известный и строго определённый базис (алгебра А) предложен Кристофером Дейтом и Хью Дарвеном.

Реляционная алгебра и реляционное исчисление эквивалентны по своей выразительной силе. Существуют правила преобразования запросов между ними.

Основное применение реляционной алгебры — предоставить теоретическую основу для реляционных баз данных, особенно языков запросов для таких баз данных, главным из которых является SQL.

* 1. Переименование

Результатом применения операции переименования атрибутов является отношение с изменёнными именами атрибутов.

Синтаксис:

*R* RENAME *Atr1, Atr2,*… AS *NewAtr1, NewAtr2,*…

где

*R* — отношение

*Atr1, Atr2,*… — исходные имена атрибутов

*NewAtr1, NewAtr2,*… — новые имена атрибутов.

* 1. Объединение

Отношение с тем же заголовком, что и у совместимых по типу отношений *A* и *B*, и телом, состоящим из кортежей, принадлежащих или *A*, или *B*, или обоим отношениям.

Синтаксис:

*A* UNION *B*

## 2.3. Пересечение

Отношение с тем же заголовком, что и у отношений *A* и *B*, и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям *A* и *B*.  
Синтаксис:

*A* INTERSECT *B*

### 2.4. Вычитание

Отношение с тем же заголовком, что и у совместимых по типу отношений *A* и *B*, и телом, состоящим из кортежей, принадлежащих отношению *A* и не принадлежащих отношению *B*.  
Синтаксис:

*A* MINUS *B*

### 2.5. Операция присваивания

Операция присваивания (:=) позволяет сохранить результат вычисления реляционного выражения в существующем отношении.

### 2.6. Декартово произведение

Отношение, заголовок (*A*1, *A*2, …, *An*, *B*1, *B*2, …, *Bm*) которого является сцеплением заголовков отношений *A*(*A*1, *A*2, …, *An*) и *B*(*B*1, *B*2, …, *Bm*), а тело состоит из кортежей, являющихся всеми вариантами сцеплений кортежей отношений *A* и *B*: (*a*1, *a*2, …, *an*, *b*1, *b*2, …,*bm*),

таких, что

(*a*1, *a*2, …, *an*) ∈ *A*,

(*b*1, *b*2, …, *bm*) ∈ *B*.

Синтаксис:

*A* TIMES *B*

### 2.7. Выборка (ограничение)

Отношение с тем же заголовком, что и у отношения *A*, и телом, состоящим из кортежей, значения атрибутов которых при подстановке в условие *c* дают значение ИСТИНА. *c* представляет собой логическое выражение, в которое могут входить атрибуты отношения *A* и/или скалярные выражения.  
Синтаксис:

*A* WHERE *c*

Выборка записывается как {\displaystyle \sigma \_{a\theta b}(R)} или {\displaystyle \sigma \_{a\theta v}(R)} где:

* *a* и *b* — имена атрибутов
* *θ* — одна из бинарных операций {\displaystyle \{\;<,\leq ,=,\neq ,\geq ,\;>\}}
* *v* — постоянная величина
* *R* отношение

### 2.8. Выборка (ограничение)

Отношение с тем же заголовком, что и у отношения *A*, и телом, состоящим из кортежей, значения атрибутов которых при подстановке в условие *c* дают значение ИСТИНА. *c* представляет собой логическое выражение, в которое могут входить атрибуты отношения *A* и/или скалярные выражения.  
Синтаксис:

*A* WHERE *c*

Выборка записывается как {\displaystyle \sigma \_{a\theta b}(R)} или {\displaystyle \sigma \_{a\theta v}(R)} где:

* *a* и *b* — имена атрибутов
* *θ* — одна из бинарных операций {\displaystyle \{\;<,\leq ,=,\neq ,\geq ,\;>\}}
* *v* — постоянная величина
* *R* отношение

### 2.9. Соединение

Операция соединения отношений *A* и *B* по предикату *P* логически эквивалентна последовательному применению операций декартова произведения *A* и *B* и выборки по предикату *P*. Если в отношениях имеются атрибуты с одинаковыми наименованиями, то перед выполнением соединения такие атрибуты необходимо переименовать.

Синтаксис:

(*A* TIMES *B*) WHERE *P*

### 2.10. Деление

Отношение с заголовком *(X1, X2, …, Xn)* и телом, содержащим множество кортежей *(x1, x2, …, xn)*, таких, что для всех кортежей *(y1, y2, …, ym) ∈ B* в отношении *A(X1, X2, …, Xn, Y1, Y2, …, Ym)* найдется кортеж *(x1, x2, …, xn, y1, y2, …, ym)*.

Синтаксис:

*A* DIVIDEBY *B*