### Лекция 7.

1. Трехзначная логика.
2. Работа с NULL.
3. Использование агрегатных функций: AVG, SUM, COUNT, MAX, MIN и т.д. с NULL значениями.
4. LEFT JOIN, FULL JOIN.
5. Объединение таблиц: UNION

**Практика:**

* Написание запросов с использованием агрегатных функций. Агрегатные функции и NULL.
* Агрегация статистики из 2-х таблиц. Замена UNION на LEFT JOIN

Дом.задание:

1. *Подумать, как при помощи LEFT JOIN и NULL получать разность двух таблиц.*

**Присылаемое ДЗ:**

Домашние задания находятся в архиве на сайте. Домашнее задание включает в себя аналитическую задачу и пакет запросов.

Решения и вопросы по задаче следует присылать на адрес nixlab@nix.ru с заголовком "Парадигмы: задача 3 (4)", в теле письма указать ФИО, группу.

Контрольная пройдет 01.12.16 в 18:00 в 115 КПМ.

**Лекция 7.**

***Использование NULL:***

Для описания отсутствующих (либо неприемлимых) данных в БД используется общий тип данных NULL. С NULL-значениями следует обращаться аккуратно, потому что при неправильном их использовании итог расчетов может оказаться абсолютно неверным.

Обычно NULL-значениями в таблицах бывают еще незаполненные данные, которые в последствии планируется заменить на реальные значения. Пробел, ноль, или еще какая-либо пометка поля, в котором предполагалось нахождение неизвестного значения, не трактуется БД как неизвестное (пустое) значение, так как простановкой символа в поле пользователь как раз определяет значение поля (соответственно Ноль или пробел).

Примером использования NULL-значений в таблицах:

Завели новый товар в таблице товаров, но, например, взвесить (измерить, назначить цену) не успели.

Строки: (Товар Масса)

Мышка 0

и

Мышка NULL

различны, так как в первой строке значение массы определено и равно 0, а во второй строке значение массы не определено и соответственно неизвестно, равно 0 или не равно 0.

***Как сравнивать NULL - значения:*** К NULL - значениям неприменимы обычные операции сравнения. Кроме того, для корректной работы с NULL необходима трехзначная логика. Приведем таблицы истинности для операторов Not, True, False.

Для оператора Not:

|  |  |
| --- | --- |
| X | Not X |
| True | False |
| False | True |
| NULL | NULL |

Для оператора AND:

|  |  |  |  |
| --- | --- | --- | --- |
| AND | TRUE | FALSE | NULL |
| TRUE | TRUE | FALSE | NULL |
| FALSE | FALSE | FALSE | FALSE |
| NULL | NULL | FALSE | NULL |

Для оператора OR:

|  |  |  |  |
| --- | --- | --- | --- |
| OR | TRUE | FALSE | NULL |
| TRUE | TRUE | TRUE | TRUE |
| FALSE | TRUE | FALSE | NULL |
| NULL | TRUE | NULL | NULL |

Есть таблица “Номенклатура”:

|  |  |  |
| --- | --- | --- |
| Артикул | Наименование | Масса |
| 1 | Мышка | 0.02 |
| 2 | Монитор | 5 |
| 3 | Системный блок | 3 |
| 4 | Клавиатура | NULL |
| 5 | Видеокарта | 1 |

Запрос:

SELECT \*

FROM Номенклатура

WHERE Масса = 1

Выведет строки:

|  |  |  |
| --- | --- | --- |
| 5 | Видеокарта | 1 |

Запрос:

SELECT \*

FROM Номенклатура

WHERE Масса <> 1

|  |  |  |
| --- | --- | --- |
| 1 | Мышка | 0.02 |
| 2 | Монитор | 5 |
| 3 | Системный блок | 3 |

Таким образом, видим, что двузначная логика, где масса либо равна, либо не равна конкретному значению, в данном случае неприменима, так как про значение NULL мы не знаем, равно оно или не равно оно конкретному значению.

Для вывода строк с NULL – значениями используется оператор IS c ключевым словом NULL:

SELECT \*

FROM Номенклатура

WHERE Масса is NULL

|  |  |  |
| --- | --- | --- |
| 4 | Клавиатура |  |

Для вывода строк с известным значением поля:

SELECT \*

FROM Номенклатура

WHERE Масса is NOT NULL

Таблица истинности для IS NULL/IS NOT NULL:

|  |  |  |
| --- | --- | --- |
| X | IS NULL | IS NOT NULL |
| TRUE | FALSE | TRUE |
| FALSE | FALSE | TRUE |
| NULL | TRUE | FALSE |

**Что будет, если использовать оператор сравнения с NULL значениями:**

По умолчанию (ISO standard SET ANSI\_NULLS ON) если в запросе прописано условие WHERE column\_name = NULL возвращается 0 строк, даже есть в column\_name есть NULL значения, аналогично, если прописано условие WHERE column\_name <> NULL возвращается 0 строк, даже если есть неNULL значения.

При соединении по полям, содержащим NULL, NULL-значения не выводятся. При этом при сортировке, группировке и distinct NULL значения считаются одинаковыми.

DECLARE @T TABLE (id int, val int)

INSERT INTO @T

VALUES (1, NULL), (2, 1), (3, 2), (4, NULL)

SELECT \*

FROM @T T1 INNER JOIN @T T2 ON T1.val = T2.val

SELECT distinct val

FROM @T

Подробнее:

ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de\_6tsql/html/aae263ef-a3c7-4dae-80c2-cc901e48c755.htm

**Пример**

Что выведет запрос?

SELECT \*

FROM Номенклатура

WHERE Масса NOT IN (1 , 2 , NULL)

***Использование агрегатных функций с NULL.***

Агрегатные функции игнорируют NULL-значения.

То есть запрос:

SELECT Sum(Масса)

FROM Номенклатура

Выдаст: 9,02 , хотя, масса для клавиатуры не определена, и, казалось бы, итоговая сумма должна была бы принять значение NULL.

PS:

Если написать запрос с группировкой:

SELECT Артикул, Sum(Масса) as Масса

FROM Номенклатура

GROUP BY Артикул

То результатом выполнения такого запроса будет:

|  |  |
| --- | --- |
| Артикул | Масса |
| 1 | 0.02 |
| 2 | 5 |
| 3 | 3 |
| 4 |  |
| 5 | 1 |

То есть SUM(NULL) = NULL.

Аналогично, для функций AVG и COUNT

SELECT AVG(Масса)

FROM Номенклатура

Ответ: 2.255

SELECT COUNT(Масса)

FROM Номенклатура

Ответ: 4

В тоже время:

SELECT COUNT(Артикул)

FROM Номенклатура

Даст ответ: 5

Таким образом, если в каком-то запросе требуется пересчитать количество строк в таблице (в подзапросе), следует иметь в виду такую особенность агрегатных функций.

Для того чтобы не задаваться вопросом, есть в поле NULL-значения или нет, можно использовать агрегатную функцию COUNT (\*), которая будет применяться в отличие от обычного COUNT ко всей строке, а не к отдельному полю.

Запрос

SELECT COUNT(\*)

FROM Номенклатура

также даст ответ 5.

***При этом группировка по NULL-значениям производится.***

То есть:

SELECT Масса, count(Артикул)

FROM Номенклатура

GROUP BY Масса

Выведет:

|  |  |
| --- | --- |
| Масса | Count-Артикул |
|  | 1 |
| 0.02 | 1 |
| 1 | 1 |
| 3 | 1 |
| 5 | 1 |

**Функции для работы с NULL-значениями:**

IsNULL() – T-SQL

Coalesce() – T-SQL

Nz() – Access

ifNULL - MySQL

**Соединения таблиц (продолжение)**

LEFT JOIN (левое соединение):

Синтаксис:

SELECT A.\*, B.\* (указывать атрибуты для вывода)

FROM A LEFT JOIN B ON A.поле = B.поле

Левое соединение таблиц А и В включает в себя все строки из таблицы А и те строки из правой таблицы В, для которых обнаружено совпадение. Для строк из таблицы А, для которых не найдено соответствия в таблице В, в поля, извлекаемые из таблицы В, проставляются значения NULL.

В QBE визуально представляется стрелкой от левой таблицы к правой. Создается при выборе параметров соединения в QBE.

**RIGHT JOIN (правое соединение)**

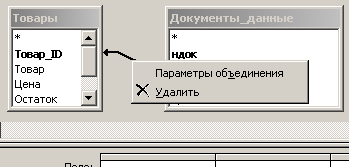
Синтаксис:

SELECT A.\*, B.\* (указывать атрибуты для вывода)

FROM A RIGHT JOIN B ON A.поле = B.поле

Правое соединение таблиц А и В включает в себя все строки из таблицы В и те строки из левой таблицы А, для которых обнаружено совпадение. Для строк из таблицы В, для которых не найдено соответствия в таблице А, в поля, извлекаемые из таблицы А, проставляются значения NULL.

В QBE визуально представляется стрелкой от правой таблицы к левой.



**Пример:**

Документы и Покупатели:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ндок** | **Покупатель\_ID** |  | **ID** | **Покупатель** |
| 1 | 10 |  | 10 | Иванов |
| 2 | 11 |  | 15 | Сидоров |

SELECT \*

FROM Документы LEFT JOIN Покупатели ON

Документы.Покупатель\_ID = Покупатели.ID

|  |  |  |  |
| --- | --- | --- | --- |
| **ндок** | **Покупатель\_ID** | **ID** | **Покупатель** |
| 1 | 10 | 10 | Иванов |
| 2 | 11 |  |  |

Вопрос: Проверьте, что будет, если написать условие WHERE Покупатель.ID > 5 ? Почему?

**Примечание:**

1. Все замечания для INNER JOIN справедливы и для LEFT и RIGHT.
2. При соединении нескольких таблиц с помощью различных типов соединений нужно помнить, что операции LEFT JOIN и RIGHT JOIN могут быть вложены в INNER JOIN, но операция INNER JOIN не может быть вложена в RIGHT или LEFT JOIN.

**Пример 2:**

Посмотрите на результаты работы запросов и подумайте, всё ли понятно.

DECLARE @A TABLE (id int)

INSERT INTO @A VALUES (1), (2)

DECLARE @B TABLE (id int)

INSERT INTO @B VALUES (1), (3)

SELECT \*

FROM @A A LEFT JOIN @B B ON

A.id = B.id

SELECT \*

FROM @A A LEFT JOIN @B B ON

A.id = B.id AND

A.id = 2

SELECT \*

FROM @A A LEFT JOIN @B B ON

A.id = B.id

WHERE A.id = 2

**Пример 3:**

Подумайте, как получить разницу между 2-мя таблицами?

Например, разница между таблицами А и Б в примере 2 будет id = 2.

SELECT A.id

FROM @A A LEFT JOIN @B B ON

A.id = B.id

WHERE B.id is NULL

**Пример 4:**

Что будет результатом следующего запроса:

SELECT \*

FROM A LEFT JOIN B ON

A.id = B.id

На таблицах:

|  |  |  |
| --- | --- | --- |
| **A** |  | **B** |
| **id** |  | **id** |
| 1 |  | 1 |
| 1 |  | 1 |
| 2 |  | 3 |

**Объединение таблиц (или запросов):**

Объединение множеств А и В – множество элементов, каждый из которых принадлежит либо одному из множеств, либо им обоим одновременно. Для таблиц справедливо следующее: множество строк, каждая из которых принадлежит либо одной из таблиц, либо им обеим.

Отсюда получаем требования на объединяемые таблицы:

* + - одинаковое количество соединяемых атрибутов;
    - одинаковые типы данных в соединяемых столбцах (i - й столбец первой таблицы такого же типа, что и i - й столбец второй таблицы).

**Синтаксис:**

SELECT поле1, поле2…, полеN

FROM А

WHERE …

UNION

SELECT поле1, поле2…, полеN

FROM B

WHERE …

Особенностью UNION является то, что при использовании UNION нет дублирующихся строк в выводе. То есть на выходе UNION дает уникальные строки, несмотря на то, что строки могли дублироваться в любом из запросов, либо одна строка выводится и из первого, и из второго.

Если все-таки на выходе хочется иметь дублирующиеся строки, то следует использовать UNION ALL.

Кроме того, учитывать то, что UNION “убивает” индексы в таблицах.

**UNION:**

1. Объединение таблиц из Архива с таблицами из неархива.

2. Пример подсчета оборота по месяцам с учетом аннулирования документов.

Пример:

В Вашей схеме документооборота можно аннулировать документы, в случае, если покупатель отказался от покупки.

Например, 20.10 Покупатель Иванов купил товар за 5000 руб, 21.10.16 Иванов вернулся с претензиями, документ от 20.10 был аннулирован (товар и деньги возвращены) и введен новый (например, с новым товаром) на сумму 2000 руб.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ндок | Дата | Покупатель | Сумма | Аннулирован | аДата |
| 1 | 20.10.16 | Иванов | 5000 | True | 21.10.16 |
| 2 | 21.10.16 | Иванов | 2000 | False |  |

Таким образом, выручка Вашего магазина по дням (денег в кассе на конец дня) должна быть равна:

|  |  |
| --- | --- |
| Дата | Оборот, руб |
| 20.10.16 | 5000 |
| 21.10.16 | -3000 |

Требуется написать запрос, который по дням будет правильно вычислять обороты (чтобы при запуске запроса на следующий день, обороты за предыдущие дни не менялись).

**Пример 5:**

**Пример составления сводных отчетов:**

Постановка: Есть две таблицы, в каждой из которых по покупателям известно, что они купили и что вернули через брак. При этом если покупатель ничего не покупал. То упоминания о нем в таблице Продажи не будет, и если покупатель ничего не сдавал в брак, то упоминания в таблице Брак так же не будет. Нужно написать запрос, который по каждому покупателю покажет его обороты: закупочный и оборот брака.

SELECT Покупатель, sum(Отпущено), sum(Брак)

FROM (

SELECT Покупатель, sum(Колво) as Отпущено, 0 as Брак

FROM Продажи

GROUP BY Покупатель

UNION

SELECT Покупатель, 0, sum(Колво)

FROM Брак

GROUP BY Покупатель

) T

GROUP BY Покупатель

Запрос можно написать без использования UNION, учитывая то, что у нас есть также отдельная таблица с покупателями.

Переписать запрос (из union (пример составления агрегированных отчетов)), без использования UNION, учитывая то, что имеется таблица со всеми покупателями.

SELECT Покупатели.Покупатель, isNULL(Отпущено, 0) Продажи, isNULL(Брак, 0) Брак

FROM Покупатели

LEFT JOIN

(

SELECT Покупатель, sum(Колво) as Отпущено

FROM Продажи

GROUP BY Покупатель

) as Продажи ON Покупатели.Покупатель = Продажи.Покупатель

LEFT JOIN

(

SELECT Покупатель, sum(Колво) as Брак

FROM Брак

GROUP BY Покупатель

) as Брак ON Покупатели.Покупатель = Брак.Покупатель

**Литература:**

1. К.Дж.Дейт, «Введение в системы баз данных»
2. Т.М.Дадашев, В.Ю.Рубаев, О.Л.Белоусов, Д.Р.Гончар «Введение в реляционные базы данных и язык SQL»
3. Д. Селко «SQL для профессионалов»