КРОК

ВВЕДЕНИЕ В ЯЗЫК SQL

Банников Сергей
Руководитель группы

Москва, 2017

ВВЕДЕНИЕ

КРОК

- Данный учебный курс посвящен языку структурированных запросов SQL
- Ознакомившись с этим курсом, вы получите общее представление о базах данных и их основных объектах, научитесь использовать язык SQL для выборки требуемых данных на примере популярного продукта Microsoft SQL Server.

СОДЕРЖАНИЕ

КРОК

КРОК

1 ОСНОВНЫЕ ПОНЯТИЯ

БАЗА ДАННЫХ

- **База данных (БД)** — совместно используемый набор логически связанных данных (и описание этих данных)
- **База данных** — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных
- **База данных** — совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними

РЕЛЯЦИОННАЯ БАЗА ДАННЫХ

КРОК

- Реляционная БД — база данных, основанная на реляционной модели данных.
- Реляционная модель данных (РМД) — прикладная теория построения баз данных, использующая для обработки данных теорию множеств и математическую логику
- Программное обеспечение для работы с базой данных, называется СУБД – система управления базами данных

- РМД содержит следующие компоненты:
- Структура – данные в БД представляют собой набор отношений (relation)
- Целостность – данные отвечают определенным условиям целостности
- Обработка – данные можно обрабатывать при помощи операторов манипулирования отношениями

СУБД - СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

КРОК

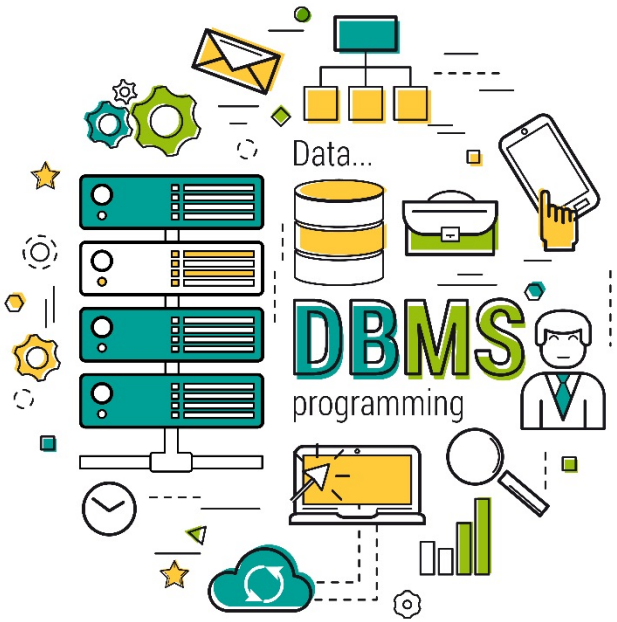
ОСНОВНЫЕ ОБЪЕКТЫ СУБД

КРОК

- Таблицы – фиксированное число столбцов, переменное число строк
- Индексы – служебные структуры для ускорения поиска
- Представления (view) – именованный запрос к таблицам
- Хранимые процедуры – именованная последовательность операторов
- Функции – процедуры, возвращающие значение
  - Скалярные – единичное значение
  - Табличные – набор данных
- Триггеры – возможность обработки событий
  - Перед событием (BEFORE)
  - После события (AFTER)
  - Вместо события (INSTEAD OF)
- Безопасность
  - Пользователи – различные виды авторизации
  - Роли – группы пользователей с одинаковым уровнем доступа

РЕЛЯЦИОННЫЕ СУБД

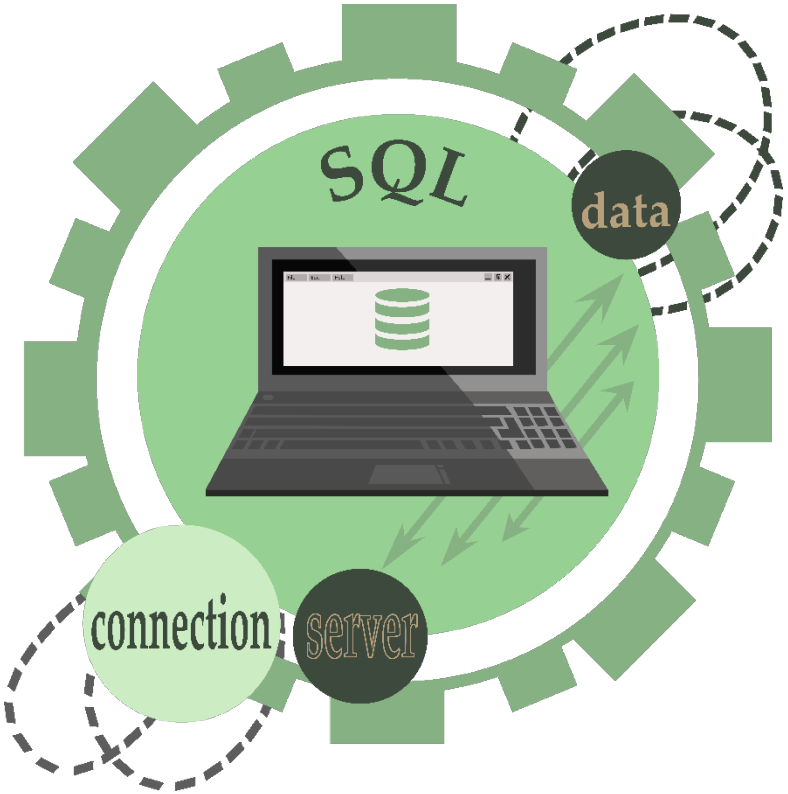
КРОК



Database Management System

- Oracle Database
- Microsoft SQL Server
- MySQL (Oracle)
- IBM DB2
- IBM Informix
- SAP Sybase
- Teradata
- Firebird
- Microsoft Access
- PostgreSQL

ИСТОРИЯ SQL



- SQL – Structured Query Language
- Непроцедурный язык для работы с базами данных
- Первоначальное название – SEQUEL (Structured English Query Language)
- Создан в начале 1970-х годов, авторы – Дональд Чэмбэрлин (Donald D. Chamberlin) и Рэй Бойс (Ray Boyce)
- Первый стандарт: ANSI SQL-86, значительно доработан в 1992 (ANSI SQL-92)
- Последняя версия – SQL:2016 или ISO/IEC 9075:2016

РАЗДЕЛЫ SQL

КРОК

Data Manipulation Language (DML) – язык манипулирования данными

- SELECT
- INSERT
- UPDATE
- DELETE

Transaction Control Language (TCL) – язык управления транзакциями

- BEGIN TRANSACTION / END TRANSACTION
- COMMIT / ROLLBACK

Data Definition Language (DDL) – язык описания данных

- CREATE TABLE, ALTER TABLE, TRUNCATE TABLE, DROP TABLE
- CREATE VIEW, ALTER VIEW, DROP VIEW

Data Control Language (DCL) – язык управления данными

- GRANT
- REVOKE

ТИПЫ ДАННЫХ

КРОК

Тип данных	Описание
CHAR (**n**)	Строка из ровно **n** символов
VARCHAR(**n**)	Строка от 0 до **n** символов
NCHAR(**n**)	Строка из ровно **n** символов (Unicode)
NVARCHAR(**n**)	Строка от 0 до **n** символов (Unicode)
BIT(**n**)	Массив из ровно **n** бит (двоичных разрядов)
BIT VARYING(**n**)	Массив от 0 до **n** бит (двоичных разрядов)

ТИПЫ ДАННЫХ

КРОК

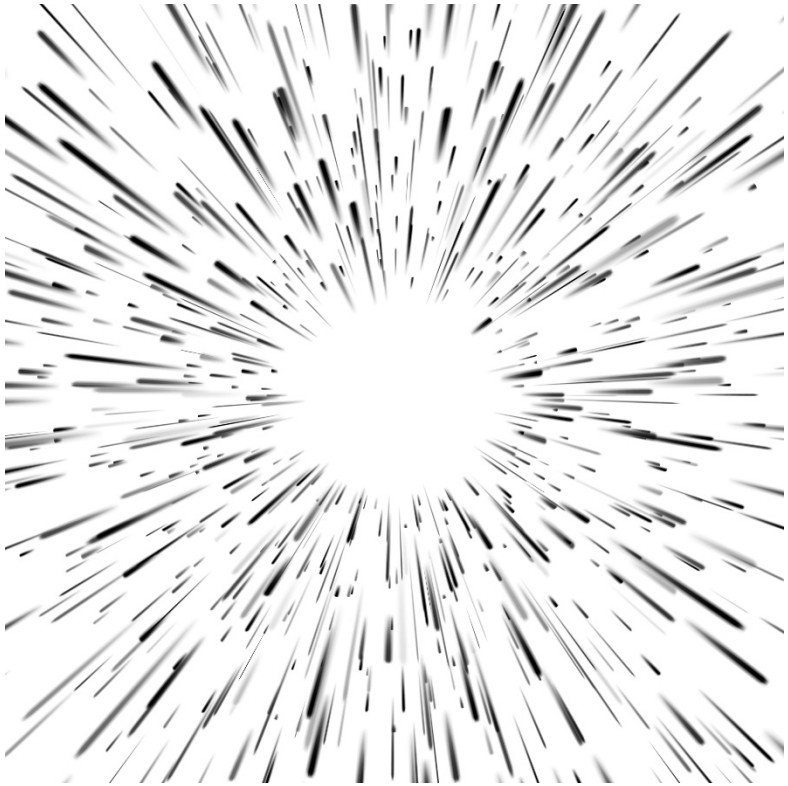
Тип данных	Описание
INTEGER	Целое число
SMALLINT	Целое число уменьшенной разрядности
BIGINT	Целое число увеличенной разрядности
FLOAT	Вещественное число одинарной точности
REAL	Вещественное число
DOUBLE	Вещественное число двойной точности
NUMERIC(**n**,**d**)	Вещественное число – **n** значащих цифр, в том числе **d** знаков после запятой
DECIMAL(**n**,**d**)	

ТИПЫ ДАННЫХ

КРОК

Тип данных	Описание
DATE	Дата
TIME	Время
TIMEZ	Время с учетом часового пояса
TIMESTAMP	Метка времени (дата и время)
TIMESTAMPZ	Метка времени (дата и время) с учетом часового пояса

ПОНЯТИЕ О ЗНАЧЕНИИ NULL

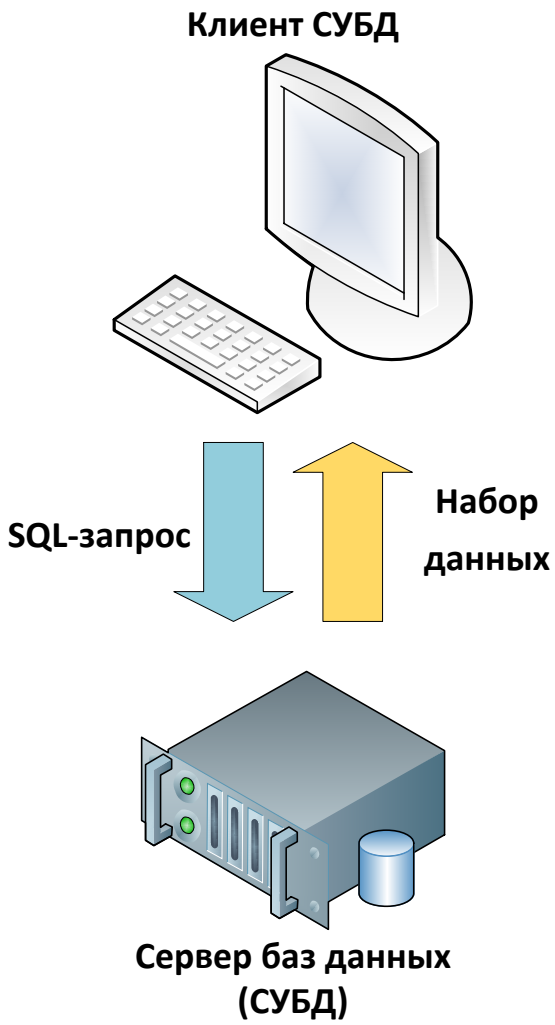


- NULL – ключевое слово SQL, обозначающее отсутствие данных
- Не следует путать значение NULL со значением по умолчанию для какого-либо столбца или типа данных
- Любая арифметическая или логическая операция с NULL приводит к NULL
- NULL добавляет в SQL элементы троичной логики
- Для обработки значения NULL предусмотрены специальные функции и операторы языка SQL

КРОК

2
ПРОСТАЯ ВЫБОРКА ДАННЫХ (SELECT)

КЛИЕНТ И СЕРВЕР



- Клиент и сервер обмениваются данными с использованием определенного программного протокола: ODBC, OLEDB, JDBC, собственный (native) клиент
- Все, что может послать клиент SQL-серверу – это SQL-запрос
- Все, что может послать в ответ SQL-сервер клиенту – набор данных или текстовое сообщение о какой-либо ошибке
- Набор данных – это таблица
- В предельном случае таблица может содержать ровно один столбец и ровно одну строку – тогда говорят о скалярном значении (scalar value)

СТРУКТУРА ОПЕРАТОРА SELECT

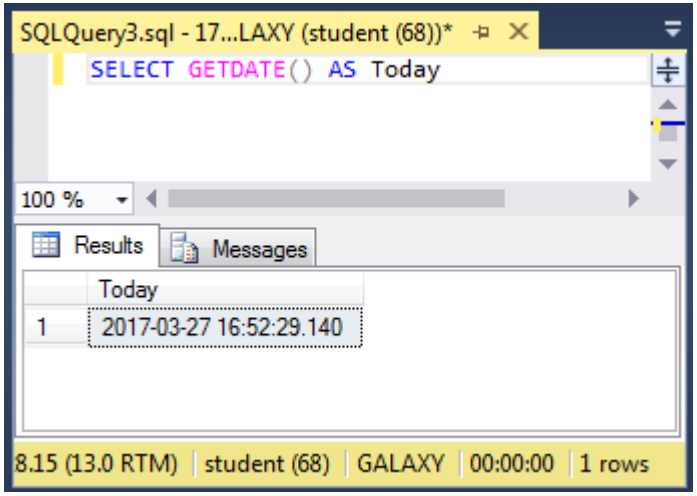
КРОК

Элемент	Содержание	Описание
SELECT	Список полей и выражений	Перечень полей таблиц, а также выражений (формул)
FROM	Список источников	Перечень источников данных – таблиц, представлений, табличных функций
WHERE	Условие	Условие фильтрации исходных строк
GROUP BY	Список для группировки	Перечень полей, на основании которых будет выполняться группировка строк таблицы
HAVING	Условие	Условие фильтрации сгруппированных строк
ORDER BY	Список для сортировки	Перечень полей, по которым будет осуществлена сортировка

ПРОСТЕЙШИЙ ОПЕРАТОР SELECT

КРОК

- SELECT 1
- SELECT 2*2, 3*3
- SELECT GETDATE()
- SELECT GETDATE() Today
- SELECT GETDATE() AS Today

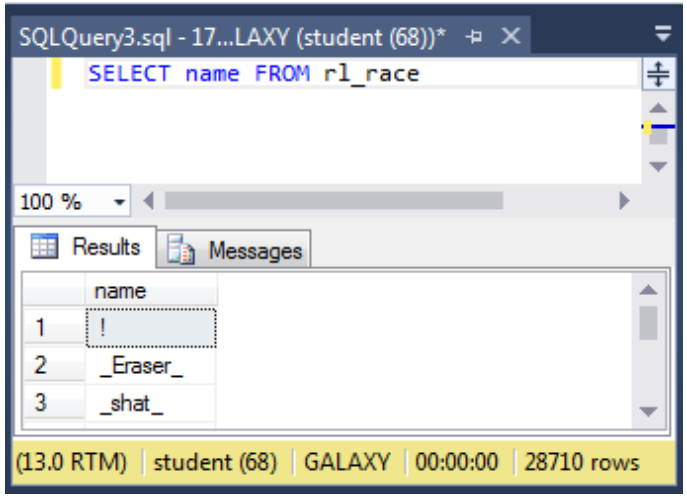


- Простейший оператор SELECT не содержит даже элемента FROM
- Он содержит перечень констант или выражений, включая вызовы скалярных функций
- Для назначения именам столбцов используется ключевое слово AS, но оно является необязательным. Тем не менее, рекомендуется его использовать – это делает запрос более читаемым
- Математические операторы: +, –, *, /, % (остаток от деление по модулю)
- Конкатенация строк: +

ВЫБОРКА ИЗ ТАБЛИЦЫ

КРОК

- SELECT * FROM rl_race
- SELECT name FROM rl_race
- SELECT id, name FROM rl_server
- SELECT UPPER(name) FROM rl_race



- Элемент FROM задает таблицу, из которой необходимо извлечь данные
- Звездочка (*) означает «все столбцы»
- Использование звездочки в реальных запросах крайне не рекомендуется
- Столбцы таблицы перечисляются через запятую
- Возможно использование скалярных функций
- Порядок строк в возвращаемом наборе данных не определен и не может быть гарантирован без дополнительных указаний

ВЫБОРКА С УСЛОВИЕМ

КРОК

- SELECT * FROM rl_race
  WHERE id > 1000
- SELECT name FROM rl_race
  WHERE name LIKE ‘A%’
- SELECT name FROM rl_race
  WHERE name BETWEEN ‘A’ AND ‘B’
- SELECT name FROM rl_game
  WHERE server IN (‘msk’, ‘spb’)
- SELECT DISTINCT server
  FROM rl_game
- SELECT race FROM rl_state
  WHERE server = ‘msk’ AND game
  LIKE ‘game%’

- Условие представляет собой логическое выражение, использующее операторы сравнения (<, <=, >, >=, <>, =) и логические операторы – NOT, AND и OR
- Оператор BETWEEN - определение принадлежности значения заданному диапазону
- Оператор IN – определение принадлежности значения заданному набору значений
- Ключевое слово DISTINCT – только уникальные значения
- Оператор LIKE - сравнение со строкой по маске

ПОИСК СТРОКИ ПО ШАБЛОНУ

КРОК

- SELECT name FROM rl_race WHERE name LIKE 'A%‘
- SELECT name FROM rl_race WHERE name LIKE ‘_A%‘
- SELECT name FROM rl_race WHERE name LIKE ‘[A-C]%‘
- SELECT name FROM rl_race WHERE name LIKE ‘[^0-9]%'

- В выражении LIKE можно и нужно использовать подстановочные символы:
- **%** – любая строка длиной от нуля и более символов
- **_** – любой одиночный символ
- **[A-Z]** – любой одиночный символ, содержащийся в заданном диапазоне
- **[^A-Z]** – любой одиночный символ, не содержащийся в заданном диапазоне

СОРТИРОВКА ВЫБОРКИ

КРОК

- SELECT name FROM rl_race
  ORDER BY name
- SELECT server, race, game, state
  FROM rl_state
  ORDER BY server, race, game
- SELECT name FROM rl_race
  ORDER BY name ASC
- SELECT name FROM rl_race
  ORDER BY name DESC
- SELECT name FROM rl_race
  ORDER BY id
- SELECT name FROM rl_race
  ORDER BY 1

- Для сортировки используется ключевое слово ORDER BY, за которым следует перечень полей или выражений
- Для указания направления сортировки (по возрастанию или по убыванию) используется ключевые слова ASC и DESC соответственно
- По умолчанию сортировка осуществляется по возрастанию (ASC)
- Сортировка может осуществляться и по столбцам, не входящим в выборку данных
- Вместо имени колонки допустимо указывать ее порядковый номер, но так делать не рекомендуется

УСЛОВНЫЕ ВЫЧИСЛЕНИЯ

КРОК

- SELECT game,
  IIF (state = '+', 'WINNER', 'OTHER')
  AS StateDescription
  FROM rl_state
  WHERE race = 'Orioner‘
  ORDER BY game

- При необходимости можно осуществлять условные вычисления непосредственно в тексте запроса
- Для этого, например, может быть использована встроенная функция **IIF**, имеющая три аргумента
- Первый аргумент – логическое выражение
- **IIF** возвращает значение второго аргумента, если логическое выражение истинно
- **IIF** возвращает значение третьего аргумента, если логическое выражение истинно

УСЛОВНЫЕ ВЫЧИСЛЕНИЯ

КРОК

- SELECT game,
  CASE WHEN state = '+' THEN
  'WINNER‘
  WHEN state = '-' THEN 'LOSER‘
  WHEN state = '*' THEN 'SURVISOR‘
  WHEN state = '#' THEN 'GAME
  CANCELLED‘
  WHEN state = '!' THEN 'SINGLE
  WINNER‘
  WHEN state = ‘@' THEN 'GROUP
  WINNER‘
  ELSE ''‘
  END AS StateDescription
  FROM rl_state
  WHERE race = 'Orioner‘
  ORDER BY game

- Помимо проверки простого условия, может быть реализована и проверка множества условий
- Для этого используется конструкция CASE WHEN… THEN… ELSE… END
- Ключевые слова WHEN… THEN… могут повторяться столько раз, сколько необходимо реализовать различных альтернатив
- Все оставшиеся варианты могут быть обработаны при помощи ветки ELSE
- Рекомендуется назначать вычислимым столбцам уникальные имена, используя AS

ОБРАБОТКА ЗНАЧЕНИЯ NULL

КРОК

- SELECT game FROM rl_state
  WHERE race = 'Orioner'
  AND state = NULL
  ORDER BY game
- SELECT game FROM rl_state
  WHERE race = 'Orioner'
  AND state IS NULL
  ORDER BY game
- SELECT game FROM rl_state
  WHERE race = 'Orioner'
  AND state IS NOT NULL
  ORDER BY game
- SELECT game FROM rl_state
  WHERE race = 'Orioner'
  AND NOT state IS NULL
  ORDER BY game

- Значение NULL некорректно использовать в логических операциях сравнения – любая операция с NULL приводит к значению NULL
- Для проверки значения на NULL необходимо использовать ключевое слово IS
- Для проверки значения на несоответствие NULL можно использовать конструкции IS NOT NULL или NOT IS NULL

ЗАМЕНА ЗНАЧЕНИЯ NULL

КРОК

- SELECT game,
  CASE WHEN state IS NULL then ‘?'
  ELSE state END
  AS StateDescription
  FROM rl_state
  WHERE race = 'Orioner‘
  ORDER BY game
- SELECT game,
  ISNULL (state, ‘?')
  AS StateDescription
  FROM rl_state
  WHERE race = 'Orioner‘
  ORDER BY game

- Иногда возникает необходимость заменить значение NULL на какое-либо другое значение
- Для этого можно использовать уже знакомую конструкцию CASE
- Но есть более простая альтернатива – функция ISNULL
- Функция ISNULL проверяет свой первый аргумент на соответствие значению NULL
- Если первый аргумент не является NULL, то он возвращается в качестве результата
- Иначе (если первый аргумент – NULL) возвращается второй аргумент

АГРЕГАТНЫЕ ФУНКЦИИ

КРОК

- SELECT COUNT(*)
  AS GameCount
  FROM rl_game
- SELECT AVG (weapon)
  AS AverageWeapon
  FROM rl_state
- SELECT SUM(planets)
  AS TotalPlanets
  FROM rl_state
- SELECT MIN (shield)
  AS MinShield
  FROM rl_state
- SELECT MAX (drive)
  AS MaxDrive
  FROM rl_state

- Агрегатные функции возвращают одиночное значение на основе входного набора данных
- Наиболее распространенные агрегатные функции:
- COUNT – подсчет количества значений (включая значения NULL)
- AVG – среднее арифметическое значений (значения NULL не учитываются)
- SUM – сумма значений (значения NULL не учитываются)
- SUM / COUNT != AVG
- MIN – минимальное значение (значения NULL не учитываются)
- MAX – максимальное значение
- (значения NULL не учитываются)

КРОК

3
ВЫБОРКА ИЗ НЕСКОЛЬКИХ ТАБЛИЦ

ВЫБОРКА ИЗ НЕСКОЛЬКИХ ТАБЛИЦ

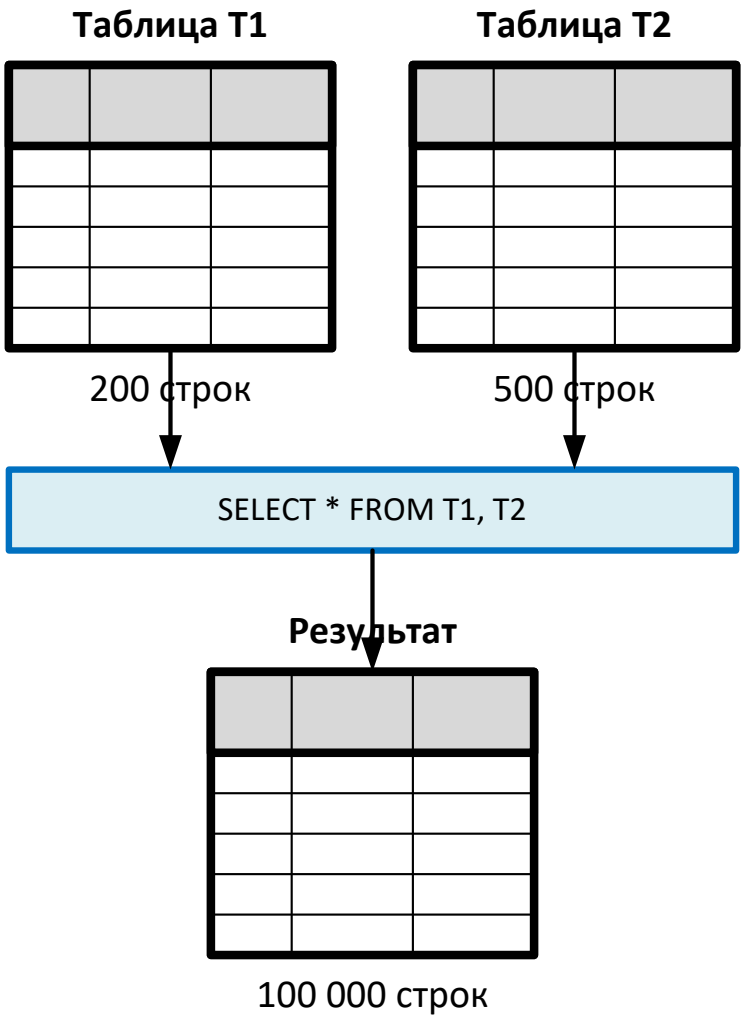
КРОК

- SELECT name, server
  FROM rl_server, rl_game
- SELECT rl_game.name, server
  FROM rl_server, rl_game
- SELECT G.name, server
  FROM rl_server AS S, rl_game AS G

- Простейший вариант выборки из нескольких таблиц – перечисление их в списке FROM
- При совпадении имен столбцов требуется использовать полное имя столбца, включая наименование таблицы
- Для сокращения текста запроса рекомендуется использовать синонимы таблиц, вводимые через ключевое слово AS
- Ключевое слово AS, как и в случае со столбцами, является необязательным, но рекомендуется к использованию

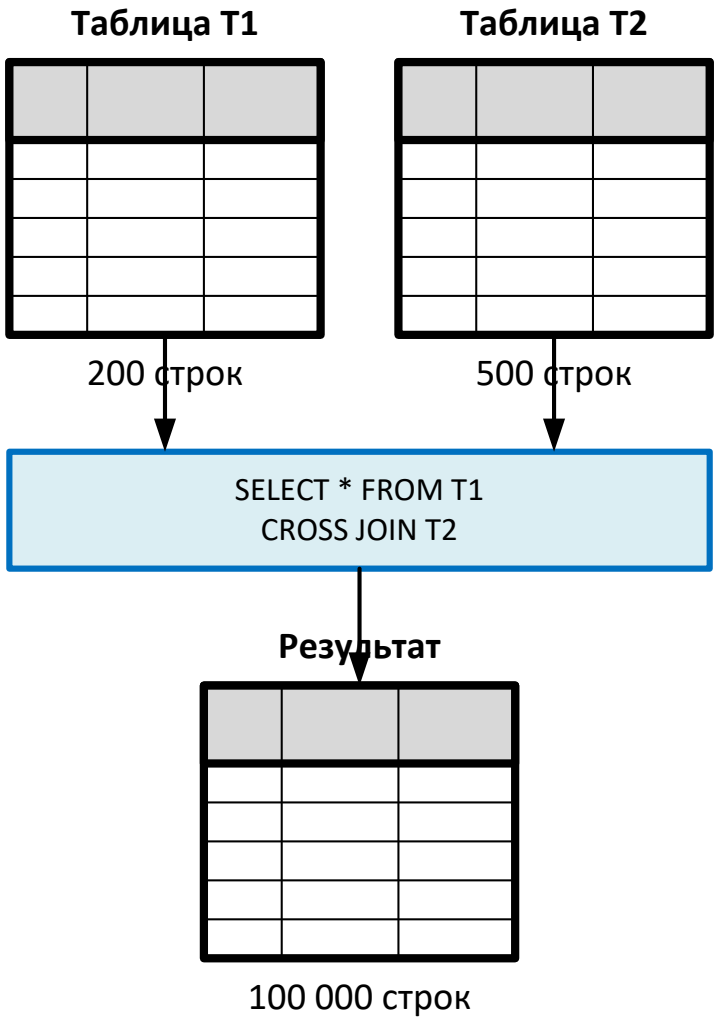
ДЕКАРТОВО ПРОИЗВЕДЕНИЕ

КРОК



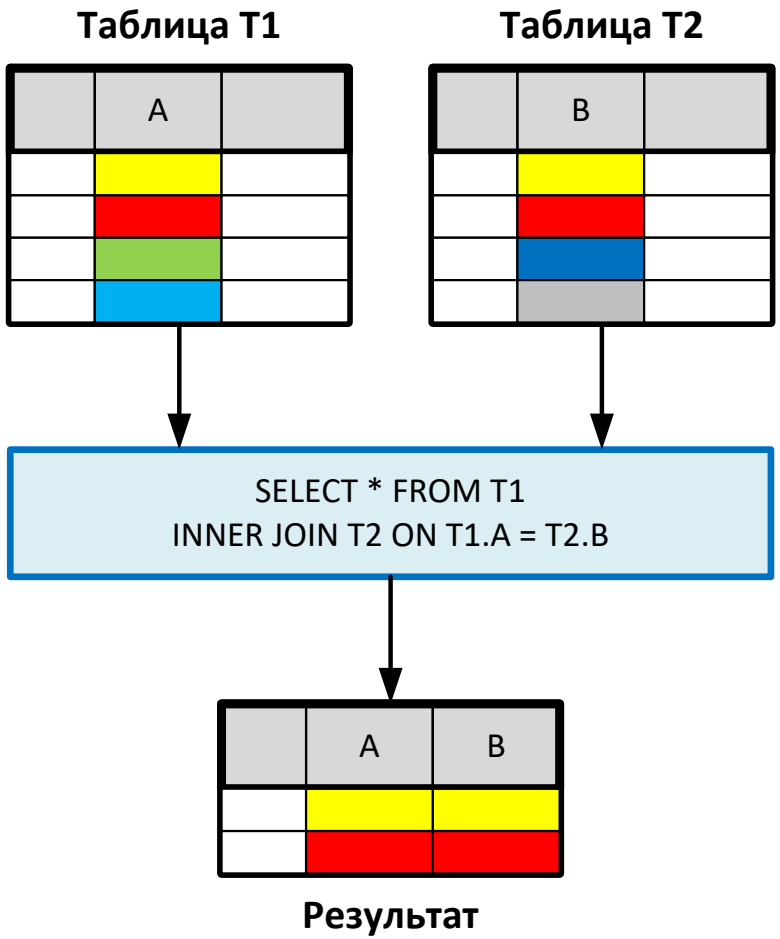
- Выполнение данного запроса приводит к тому, что в результирующей таблице количество строк будет равно произведению количества строк в исходных таблицах
- Это происходит потому, что не накладывается никаких дополнительных условий на соответствие строчек в обоих таблицах
- Строки из обеих таблиц комбинируются по принципу «каждый с каждым»
- В теории множеств это называется декартовым произведением
- Синтаксис введен в стандарте ANSI SQL-89

CROSS JOIN



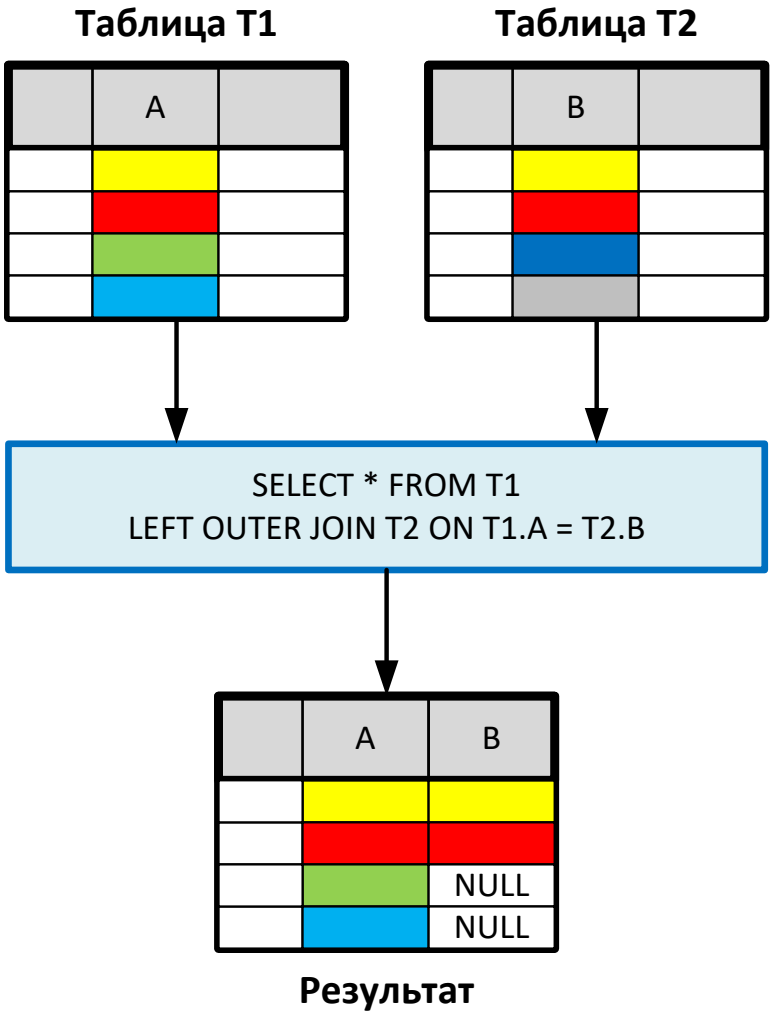
- Полное декартово произведение также реализуется при помощи ключевого слова CROSS JOIN
- Практическое применение носит ограниченный характер, обычно требуется наложить дополнительные условия на строки таблиц, определенным образом сопоставить их друг другу
- Синтаксис CROSS JOIN введен начиная со стандарта ANSI SQL-92 и является рекомендованным к использованию

INNER JOIN



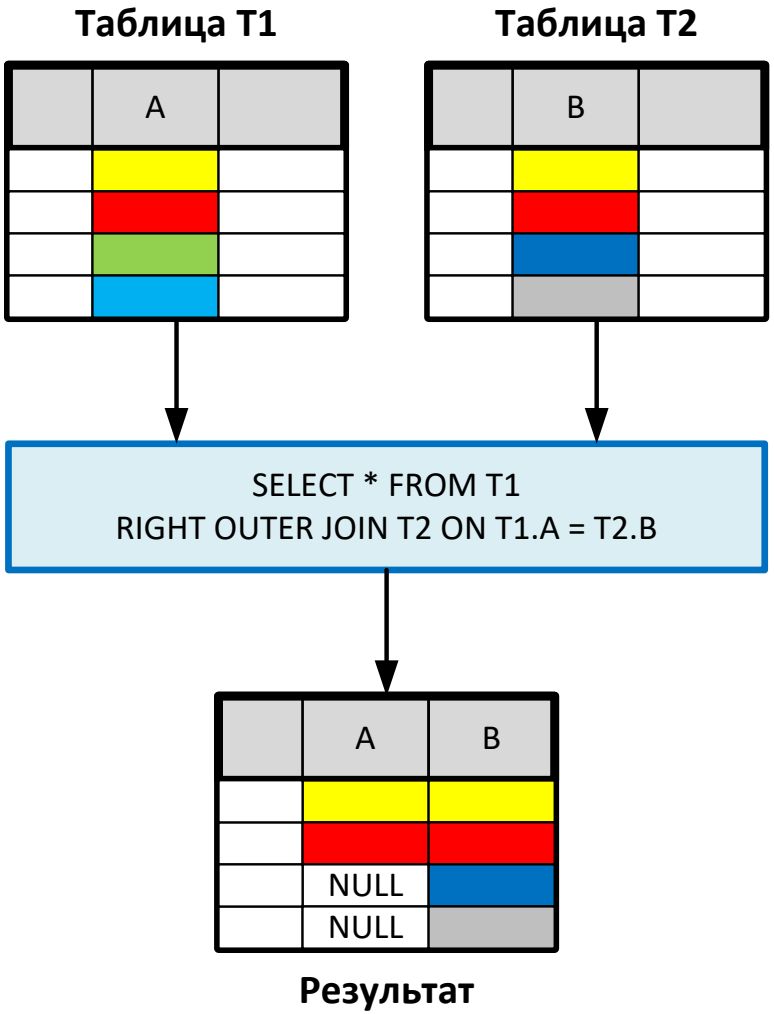
- Требуется дополнить таблицу T1 сведениями, содержащимися в таблице T2
- Оператор INNER JOIN осуществляет проверку условия, заданного после ключевого слова ON
- Все строки из таблицы T2, для который условие выполняется, объединяются со строками из таблицы T1
- Строки обеих таблиц, для которых условие не выполняется, исключаются из результата

LEFT OUTER JOIN



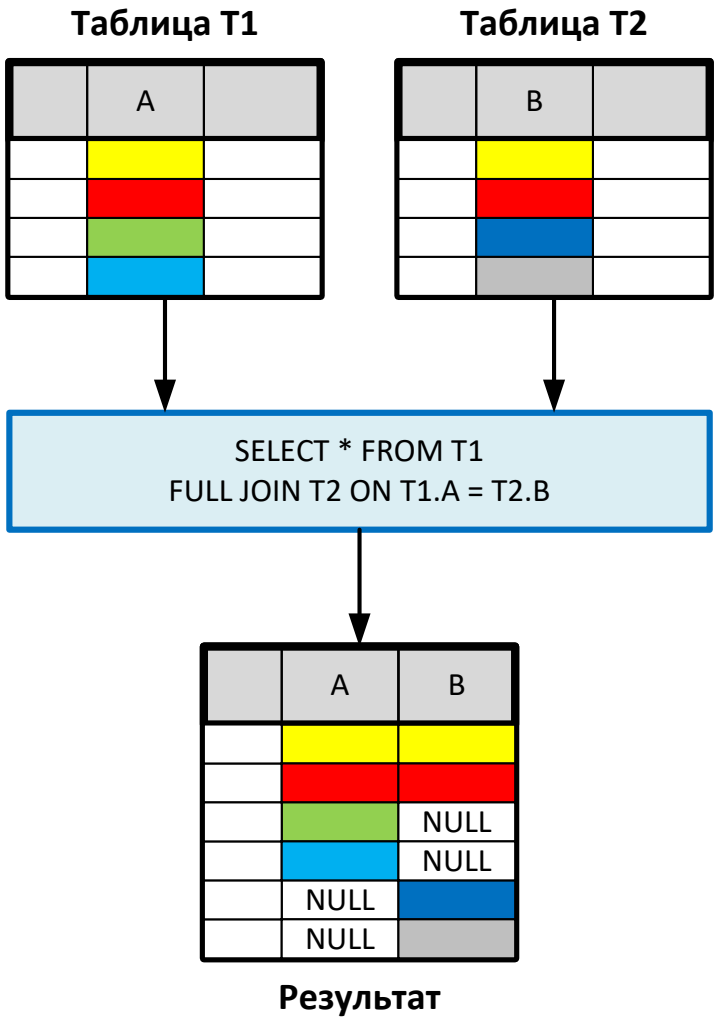
- Требуется дополнить таблицу T1 сведениями, содержащимися в таблице T2
- Оператор LEFT OUTER JOIN осуществляет проверку условия, заданного после ключевого слова ON
- Все строки из таблицы T2, для который условие выполняется, объединяются со строками из таблицы T1
- Если для строки из левой таблицы T1 не находится подходящих по условию ON строк из правой таблицы T2, то такие строки все равно попадают в результат – на место данных из таблицы T2 подставляется значение NULL

RIGHT OUTER JOIN



- Требуется дополнить таблицу T1 сведениями, содержащимися в таблице T2
- Оператор RIGHT OUTER JOIN осуществляет проверку условия, заданного после ключевого слова ON
- Все строки из таблицы T2, для который условие выполняется, объединяются со строками из таблицы T1
- Если для строки из правой таблицы T2 не находится подходящих по условию ON строк из левой таблицы T1, то такие строки все равно попадают в результат – на место данных из таблицы T1 подставляется значение NULL

FULL JOIN



- Требуется дополнить таблицу T1 сведениями, содержащимися в таблице T2
- Оператор FULL JOIN осуществляет проверку условия, заданного после ключевого слова ON
- Все строки из таблицы T2, для который условие выполняется, объединяются со строками из таблицы T1
- Строки, для которых не нашлось соответствия, добавляются в результат. На месте отсутствующих данных подставляется значение NULL
- Дублирующие строки удаляются автоматически (неявный DISTINCT)

КРОК

4
ПОДЗАПРОСЫ И СЛОЖНЫЕ ЗАПРОСЫ

ПОДЗАПРОС КАК УСЛОВИЕ

КРОК

- SELECT name, server
  FROM rl_game
  WHERE tpw =
  (
  SELECT MIN (tpw)
  FROM rl_game
  WHERE tpw > 0
)
- SELECT name, alias, realname
  FROM rl_race
  WHERE name IN
  (
  SELECT race
  FROM rl_state
  WHERE state = '+'
)

- Иногда требуется использовать в условии запроса величину, которая в свою очередь сама определяется при помощи другого запроса
- В таком случае на помощь приходят подзапросы
- Подзапрос заключается в скобки
- Если подзапрос возвращает скалярное значение, его можно использовать как аргумент при сравнении
- Если подзапрос возвращает набор данных, требуется использование ключевого слова IN

ПОДЗАПРОС КАК ИСТОЧНИК

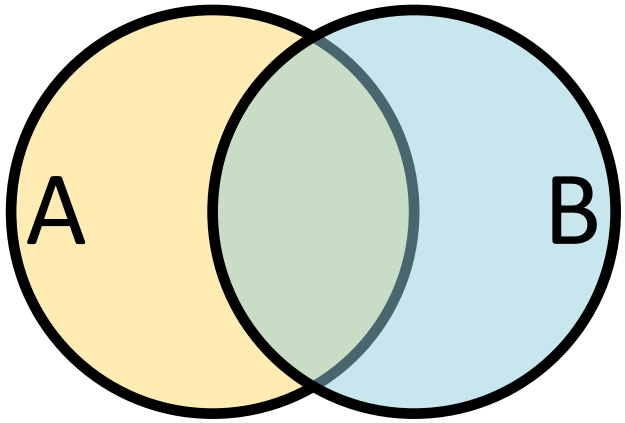
КРОК

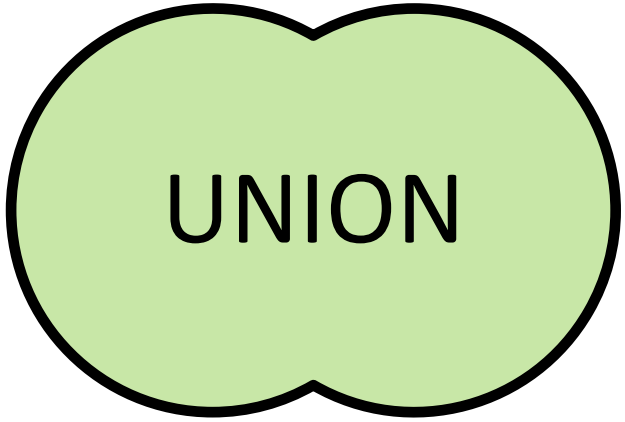
- SELECT A.* FROM
  (
  SELECT game, race
  FROM rl_state
  WHERE state = '*'
) AS A

- Подзапрос можно также использовать вместо таблицы после ключевого слова FROM или JOIN
- В таком случае для подзапроса обязательно должен быть описан синоним с использованием ключевого слова AS (или без него, так как AS является опциональным)

ОБЪЕДИНЕНИЕ ТАБЛИЦ

КРОК

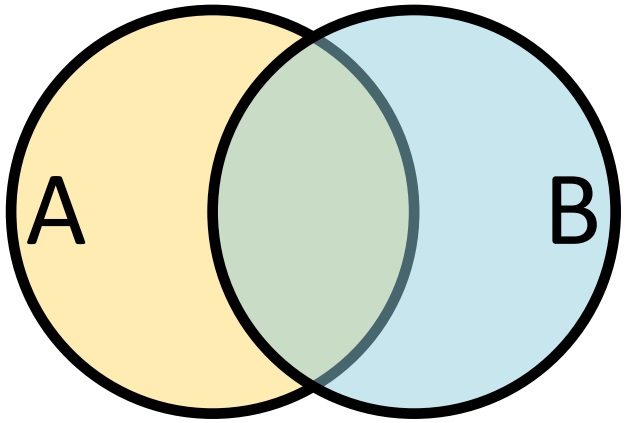


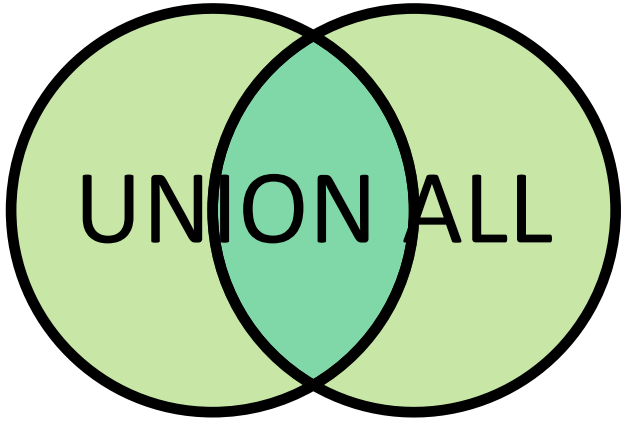


- Ключевое UNION выполняет объединение двух таблиц и возвращает суммарный набор строк
- Дублирующие строки удаляются
- Столбцы в первой и второй выборке должны совпадать по количеству и типу, но не обязательно по имени
- Сортировка может быть задана только в конце запроса
- SELECT name FROM rl_race
  UNION
  SELECT alias FROM rl_race WHERE alias IS NOT NULL
  ORDER BY name

ОБЪЕДИНЕНИЕ ТАБЛИЦ

КРОК

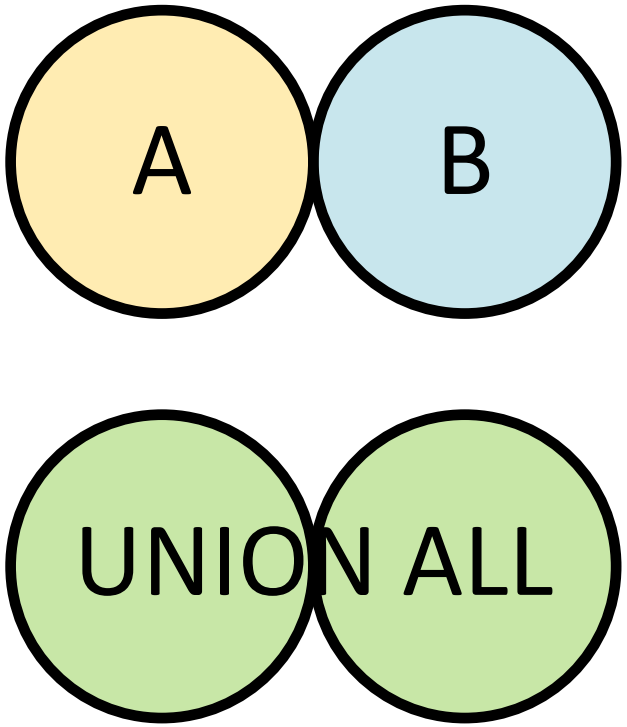




- Ключевое UNION ALL выполняет объединение двух таблиц и возвращает суммарный набор строк
- Дублирующие строки сохраняются
- Столбцы в первой и второй выборке должны совпадать по количеству и типу, но не обязательно по имени
- Сортировка может быть задана только в конце запроса
- SELECT name FROM rl_race
  UNION ALL
  SELECT alias FROM rl_race WHERE alias IS NOT NULL
  ORDER BY name

ОБЪЕДИНЕНИЕ ТАБЛИЦ

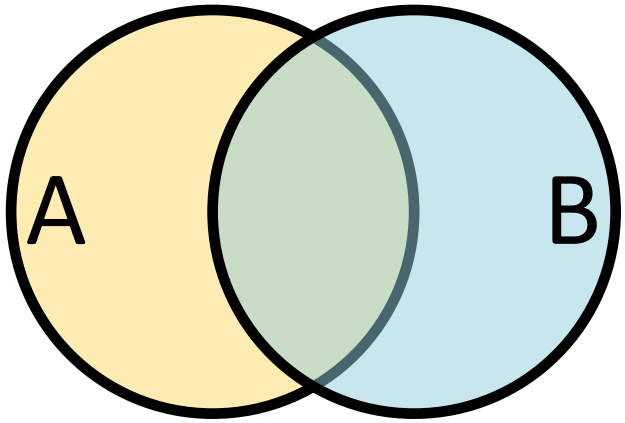
КРОК

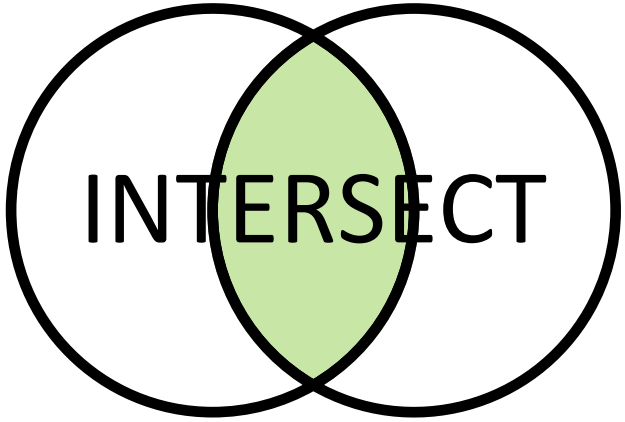


- Ключевое UNION ALL выполняет объединение двух таблиц и возвращает суммарный набор строк
- Дублирующие строки сохраняются
- Столбцы в первой и второй выборке должны совпадать по количеству и типу, но не обязательно по имени
- Сортировка может быть задана только в конце запроса
- SELECT name FROM rl_race
  UNION ALL
  SELECT alias FROM rl_race WHERE alias IS NOT NULL
  ORDER BY name

ПЕРЕСЕЧЕНИЕ ТАБЛИЦ

КРОК

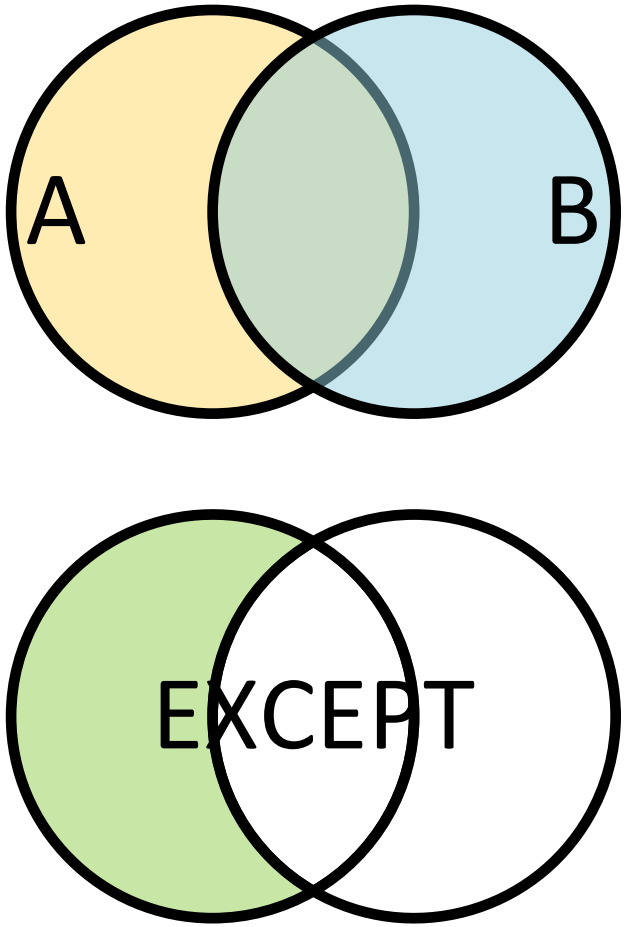




- Ключевое INTERSECT выполняет пересечение двух таблиц и строки, присутствующие только в обеих таблицах
- Столбцы в первой и второй выборке должны совпадать по количеству и типу, но не обязательно по имени
- Сортировка может быть задана только в конце запроса

ВЫЧИТАНИЕ ТАБЛИЦ

КРОК



- Ключевое EXCEPT выполняет вычитание двух таблиц.
- Возвращаются только строки, присутствующие в первой и отсутствующие во второй таблице
- Порядок таблиц имеет значение
- Столбцы в первой и второй выборке должны совпадать по количеству и типу, но не обязательно по имени
- Сортировка может быть задана только в конце запроса

КРОК

5
ОБРАБОТКА ДАННЫХ

ОПЕРАТОРЫ ОБРАБОТКИ ДАННЫХ

КРОК

- SELECT – выборка данных (не изменяет данные)
- INSERT – добавление данных в таблицу
- UPDATE – изменение данных в таблице
- DELETE – удаление данных из таблицы

INSERT

КРОК

- INSERT INTO rl_game (name) VALUES (‘game1’)
- INSERT INTO rl_game (name) VALUES (‘game1’), (‘game2’)
- INSERT INTO rl_game (name) SELECT name FROM rl_state WHERE server = ‘msk’

- Оператор INSERT добавляет данные в заданную таблицу
- Осуществляется проверка на корректность данных (первичные ключи, внешние ключи, уникальные индексы)
- Дополнительные контроль данных может быть реализован при помощи триггеров
- Значения могут быть заданы непосредственно в виде констант
- Значения также могут быть получены как результат выполнения запроса

UPDATE

КРОК

- UPDATE rl_state
  SET state = NULL
- UPDATE rl_state
  SET state = NULL
  WHERE state = ‘’
- UPDATE rl_state
  SET pop = pop + 10
  WHERE race = ‘Orioner’
  AND game = ‘game16’
- UPDATE Person
  SET department = D.name
  FROM Departs AS D
  INNER JOIN Person AS P
  ON P.depart_id = D.id

- Оператор UPDATE позволяет изменять определенные значения в заданной таблице
- Оператор может содержать условия WHERE
- Для вычисления нового значения может использоваться существующее
- В качестве источника данных может использоваться специальный подзапрос вида UPDATE … FROM
- При использовании конструкции UPDATE…FROM необходимо обратить внимание на то, чтобы подзапрос возвращал скалярное значение, иначе результат не определен

DELETE

КРОК

- DELETE FROM rl_user
- DELETE FROM rl_user
  WHERE id = 1
- DELETE FROM rl_race
  WHERE name NOT IN
  (
  SELECT race FROM rl_state
)
- DELETE FROM rl_race
  WHERE NOT EXISTS
  (
  SELECT 1 FROM rl_state
  WHERE race = rl_race.name
)

- Оператор DELETE удаляет данные из таблицы
- Если не указать условие удаление, будут удалены все данные из таблицы. Такой оператор эквивалентен оператору TRUNCATE TABLE
- Условие удаления может быть задано непосредственно при помощи WHERE
- Альтернативный способ – использование подзапроса и ключевого слова IN или NOT IN
- В подзапросе для построения условия можно использовать ссылки на поля той таблицы, из которой следует удалить данные

ТРАНЗАКЦИИ

КРОК

- BEGIN TRANSACTION
  DELETE FROM rl_race
  WHERE NOT EXISTS
  (
  SELECT 1 FROM rl_state
  WHERE race = rl_race.name
)
  COMMIT TRANSACTION
- BEGIN TRAN
  DELETE FROM rl_race
  WHERE NOT EXISTS
  (
  SELECT 1 FROM rl_state
  WHERE race = rl_race.name
)
  ROLLBACK

- Для обеспечения целостности данных при последовательных вставках, изменениях и удалениях предусмотрен механизм транзакций
- Транзакция – это логическая операция, которая осуществляет сложную модификацию базы данных. При этом транзакция может быть либо выполнена целиком, либо не выполнена вовсе – частичное исполнение транзакции не допускается
- Транзакция может быть подтверждена при помощи оператора COMMIT или отменена при помощи оператора ROLLBACK

ОБРАБОТКА ОШИБОК

КРОК

- BEGIN TRY
  BEGIN TRANSACTION
  INSERT INTO rl_race (id)
  VALUES (**12345**)
  INSERT INTO rl_race (id)
  VALUES (**NULL**)
  COMMIT
  END TRY
  BEGIN CATCH
  PRINT 'Error‘
  ROLLBACK
  END CATCH

- Транзакция может комбинироваться с блоком обработки ошибок
- В Microsoft SQL Server для перехвата и обработки ошибок используется блок TRY…CATCH
- В конце блока TRY должен размещаться оператор COMMIT для успешного завершения транзакции
- В блоке CATCH выполняется откат транзакции при помощи оператора ROLLBACK

КРОК

6 ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ

ГРУППИРОВКА ВЫБОРКИ

КРОК

- SELECT server, COUNT(*)
  FROM rl_state
  GROUP BY server
- SELECT server,
  MAX (drive) as MaxDrive,
  MIN (weapon) as MaxWeapon,
  AVG (shield) as AvgShield,
  SUM (cargo) as SumCargo
  FROM rl_state
  GROUP BY server
  ORDER BY server

- Группировка используется для объединения нескольких строк исходной таблицы и вычисления агрегирующих значений (суммы, среднего и пр.)
- После группировки исходные строки таблицы преобразуются, и столбцы, которые не перечислены в условии группировки, можно использовать только в агрегирующих функциях
- Сортировка результата осуществляется после группировки

АГРЕГАТНЫЕ ФУНКЦИИ

КРОК

- SELECT server, COUNT(*)
  FROM rl_state
  GROUP BY server
- SELECT server, AVG(weapons)
  FROM rl_state
  GROUP BY server
- SELECT server, SUM(planets)
  FROM rl_state
  GROUP BY server
- SELECT server, MIN(shield)
  FROM rl_state
  GROUP BY server
- SELECT server, MAX(drive)
  FROM rl_state
  GROUP BY server

- Наиболее распространенные агрегатные функции:

- COUNT – подсчет количества значений (включая значения NULL)
- AVG – среднее арифметическое значений (значения NULL не учитываются)
- SUM – сумма значений (значения NULL не учитываются)
- SUM / COUNT != AVG
- MIN – минимальное значение (значения NULL не учитываются)
- MAX – максимальное значение
- (значения NULL не учитываются)

УСЛОВИЕ ДЛЯ ГРУППЫ

КРОК

- SELECT server,
  AVG (shield) as AvgShield
  FROM rl_state
  GROUP BY server
  HAVING AVG (shield) > 300
- SELECT server,
  AVG (shield) as AvgShield
  FROM rl_state
  WHERE shield < 500
  GROUP BY server
  HAVING AVG (shield) > 300

- Оператор HAVING определяет условие выборки для группы
- Оператор HAVING указывается после оператора GROUP BY
- Оператор HAVING требует указания агрегирующего выражения (указание псевдонима недопустимо, так как псевдонимы назначаются на последнем этапе обработки запроса)
- В одном запросе допускается применение условия как к столбцу, так и к группе

ФУНКЦИИ РАБОТЫ С ДАТАМИ

КРОК

- SELECT GETDATE()
- SELECT DATEADD(DAY, 1, GETDATE())
- SELECT DATEDIFF(YEAR, GETDATE(), '2025-01-01')

- Функция GETDATE() возвращает текущую системную дату и время
- Функция DATEADD() возвращает новое значение datetime, добавляя к указанной дате промежуток времени
- Функция DATEDIFF() возвращает разницу в секундах, минутах, часах, днях, месяцах, годах между двумя датами

ФУНКЦИИ РАБОТЫ С ДАТАМИ

КРОК

- SELECT DAY(GETDATE())
- SELECT MONTH(GETDATE())
- SELECT YEAR(GETDATE())
- SELECT DATEPART (WEEKDAY, GETDATE())

- Функции DAY(), MONTH(), YEAR() возвращают значение дня, месяца, года от указанной даты соответственно
- Функция DATEPART () возвращает заданную часть даты и времени – год, квартал, месяц, число, номер недели, день недели, час, минуту, секунду

ФУНКЦИИ РАБОТЫ С ДАТАМИ

КРОК

- SELECT DAY(DATEADD(Month, 1, GETDATE()) - DAY(DATEADD(Month, 1, GETDATE())))

- SELECT CAST(DATEDIFF(DAY, DATEPART(WEEKDAY, GETDATE()) - 1, GETDATE()) AS DATETIME)

- Возвращает количество дней в текущем месяце

- Возвращает начало недели (BOW – begin of week)

ТРАНСПОНИРОВАНИЕ ТАБЛИЦ

КРОК

- SELECT game,
  [skull], [svin], [msk], [spb]
  FROM [GALAXY].[dbo].[rl_state]
  PIVOT
  (
  AVG(shield)
  FOR server
  IN([skull], [svin], [msk], [spb])
) pvt

- Оператор PIVOT разворачивает возвращающее табличное значение выражение, преобразуя уникальные значения одного столбца выражения в несколько выходных столбцов, а также, в случае необходимости, объединяет оставшиеся повторяющиеся значения столбца и отображает их в выходных данных

	server	shield
1	skull	174
2	svin	272
3	msk	527
4	spb	460

	game	skull	svin	msk	spb
1	2x2_01	NULL	NULL	NULL	NULL
2	2x2_01	NULL	NULL	NULL	NULL

ТРАНСПОНИРОВАНИЕ ТАБЛИЦ

КРОК

- SELECT server, game, stat, value
  FROM [GALAXY].[dbo].[rl_state]
  UNPIVOT
  (
  value
  FOR stat IN ([vote], [drive],
  [weapon], [shield])
) unpvt

- Оператор UNPIVOT производит действия, обратные PIVOT, то есть представляет данные, записанные в строке таблицы, в одном столбце

	server	game	vote	drive	weapon	shield
1	spb	dg1051	37	505	150	239

	server	game	stat	value
1	spb	dg1051	vote	37
2	spb	dg1051	drive	505
3	spb	dg1051	weapon	150
4	spb	dg1051	shield	239

КРОК

СПАСИБО
ЗА ВНИМАНИЕ

Банников Сергей
Руководитель группы

111033, Москва, ул. Волочаевская, д.5, к.1
Т: (915) 040 67 93
Ф: (495) 974 22 77
sbannikov@croc.ru

croc.ru