

Язык SQL

Наумов Д.А., доц. каф. КТ

Базы знаний и базы данных, 2021

Содержание лекции

1 Основные понятия

- 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 БД

база данных, основанная на реляционной модели 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



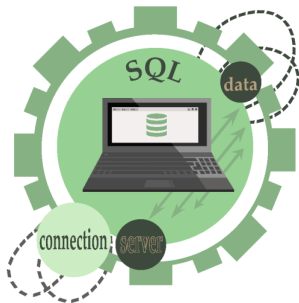
Основные объекты СУБД

- 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, табличные)
- Триггеры – возможность обработки событий
- Пользователи – различные виды авторизации
- Роли – группы пользователей с одинаковым уровнем доступа



Database Management System

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



- 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

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 MySQL

Символьные типы

- CHAR - строка фиксированной длины.
- VARCHAR - строка переменной длины.
- TINYTEXT: текст длиной до 255 байт.
- TEXT: представляет текст длиной до 65 КБ.
- MEDIUMTEXT: представляет текст длиной до 16 МБ
- LONGTEXT: представляет текст длиной до 4 ГБ

Типы для хранения двоичных данных

- TINYBLOB: данные длиной до 255 байт.
- BLOB: данные длиной до 65 КБ.
- MEDIUMBLOB: данные длиной до 16 МБ
- LARGEBLOB: данные длиной до 4 ГБ

Типы данных MySQL

Типы для работы с датой/временем

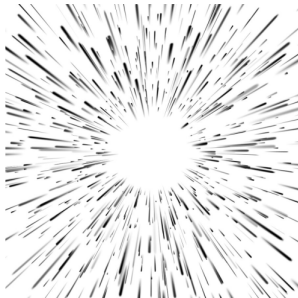
- DATE: хранит даты с 1 января 1000 года до 31 декабря 9999 года (с "1000-01-01" до "9999-12-31").
- TIME: хранит время от -838:59:59 до 838:59:59.
- DATETIME: объединяет время и дату, диапазон дат и времени - с 1 января 1000 года по 31 декабря 9999 года (с "1000-01-01 00:00:00" до "9999-12-31 23:59:59").
- TIMESTAMP: также хранит дату и время, в диапазоне: от "1970-01-01 00:00:01" UTC до "2038-01-19 03:14:07" UTC.
- YEAR: хранит год в виде 4 цифр.

Типы данных MySQL

Числовые типы данных

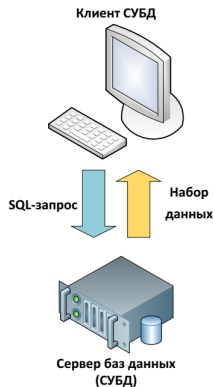
- TINYINT: целые числа от -127 до 128,
- BOOL: TINYINT(1)
- SMALLINT: целые числа от -32768 до 32767
- MEDIUMINT: целые числа от -8388608 до 8388607
- INT: целые числа от -2147483648 до 2147483647
- BIGINT: целые числа от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
- DECIMAL: числа с фиксированной точностью DECIMAL(precision, scale).
- FLOAT: числа с плавающей точкой одинарной точности
- DOUBLE: числа с плавающей точкой двойной точности

NULL-значения



- 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

Клиент и сервер



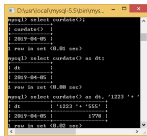
- Клиент и сервер обмениваются данными с использованием определенного программного протокола: ODBC, OLEDB, JDBC, собственный (native) клиент
- Все, что может послать клиент SQL-серверу – это SQL-запрос
- Все, что может послать в ответ SQL-сервер клиенту – набор данных или текстовое сообщение о какой-либо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

Структура оператора SELECT

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

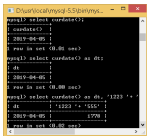
- SELECT 1
 - SELECT 2*2,
3*3
 - SELECT
CURDATE() AS
Today
 - SELECT '1' + '3'
- Простейший оператор SELECT не содержит даже элемента FROM
 - Он содержит перечень констант или выражений, включая вызовы скалярных функций
 - Для назначения именам столбцов используется ключевое слово AS, но оно является необязательным. Тем не менее, рекомендуется его использовать – это делает запрос более читаемым
 - Математические операторы: +, -, *, /
 - Конкатенация строк: CONCAT(,)



```
mysql> select curdate();
+-----+
| curdate() |
+-----+
| 2019-04-05 |
+-----+
mysql> select curdate() as dt;
+-----+
| dt |
+-----+
| 2019-04-05 |
+-----+
mysql> select curdate() as dt, '123' + '456';
+-----+-----+
| dt | 123456 |
+-----+-----+
| 2019-04-05 | 1799 |
+-----+-----+
```


Выборка данных из таблицы

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



```

SQL> select carddate()
carddate()
-----
2019-04-05
SQL> row in out (0.01 sec)
SQL> select carddate() as dt;
dt
--
2019-04-05
SQL> row in out (0.00 sec)
SQL> select carddate() as dt, '123' as
dt      '123' * '145'
-----
dt      1770
SQL> row in out (0.00 sec)
  
```

- 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

- `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

- `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

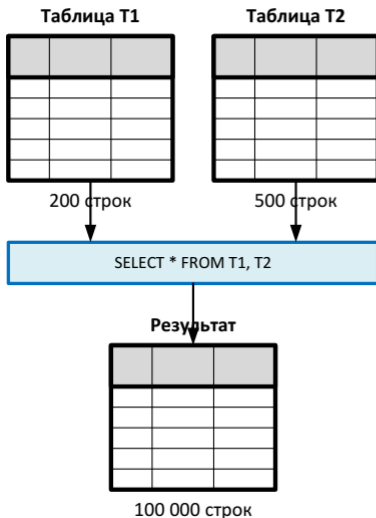
- `SELECT name FROM rl_race
ORDER BY name`
- `SELECT server, race, game, state
FROM rl_state
ORDER BY server, race, game`
- `SELECT name FROM rl_race
ORDER BY name ASC`
- `SELECT name FROM rl_race
ORDER BY name DESC`
- `SELECT name FROM rl_race
ORDER BY id`
- `SELECT name FROM rl_race
ORDER BY 1`
- Для сортировки используется ключевое слово `ORDER BY`, за которым следует перечень полей или выражений
- Для указания направления сортировки (по возрастанию или по убыванию) используются 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

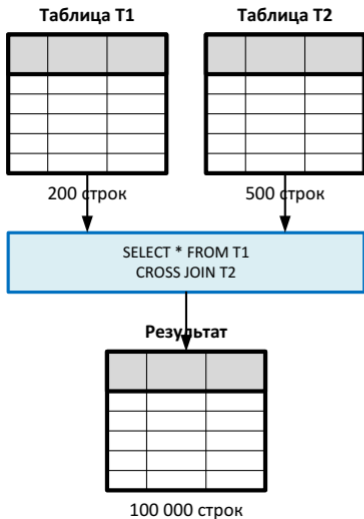
```
SELECT game,  
  IF (state = '+', 'WINNER', 'OTHER')  
  AS StateDescription  
FROM rl_state  
WHERE race = 'Orioner'  
ORDER BY game
```

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

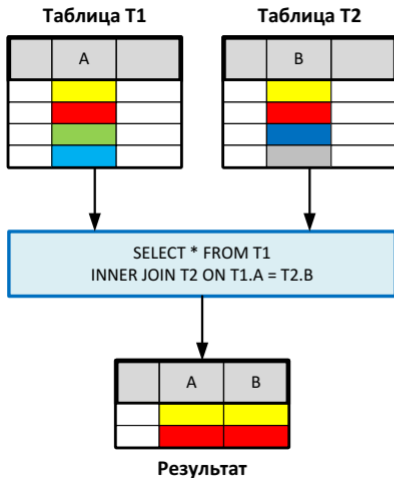
- `SELECT game FROM rl_state
WHERE race = 'Orioner'
AND state = NULL
ORDER BY game`
- `SELECT game FROM rl_state
WHERE race = 'Orioner'
AND state IS NULL
ORDER BY game`
- `SELECT game FROM rl_state
WHERE race = 'Orioner'
AND state IS NOT NULL
ORDER BY game`
- `SELECT game FROM rl_state
WHERE race = 'Orioner'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



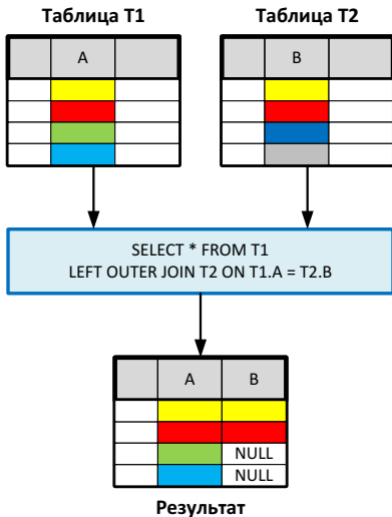
- Выполнение данного запроса приводит к тому, что в результирующей таблице количество строк будет равно произведению количества строк в исходных таблицах
- Это происходит потому, что не накладывается никаких дополнительных условий на соответствие строчек в обеих таблицах
- Строки из обеих таблиц комбинируются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



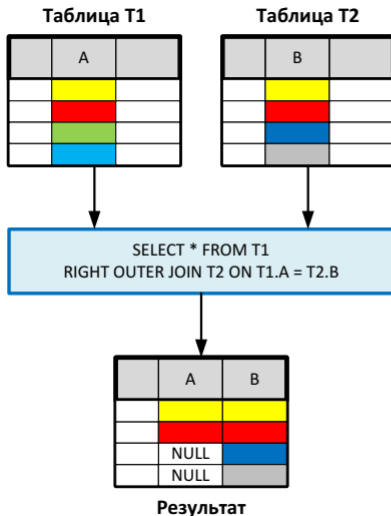
- Полное декартово произведение также реализуется при помощи ключевого слова CROSS JOIN
- Практическое применение носит ограниченный характер, обычно требуется наложить дополнительные условия на строки таблиц, определенным образом сопоставить их друг другу
- Синтаксис CROSS JOIN введен начиная со стандарта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



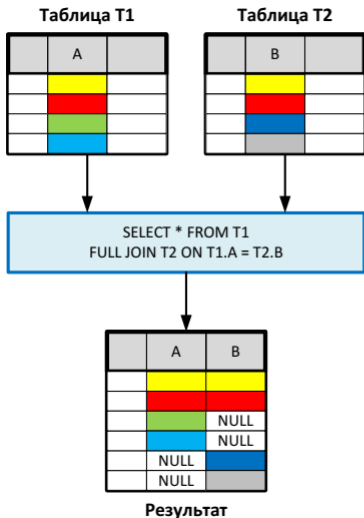
- Требуется дополнить таблицу T1 сведениями, содержащимися в таблице T2
- Оператор INNER JOIN осуществляет проверку условия, заданного после ключевого слова ON
- Все строки из таблицы T2, для которых условие выполняется, объединяются со строками из таблицы T1
- Строки обеих таблиц, для которых условие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



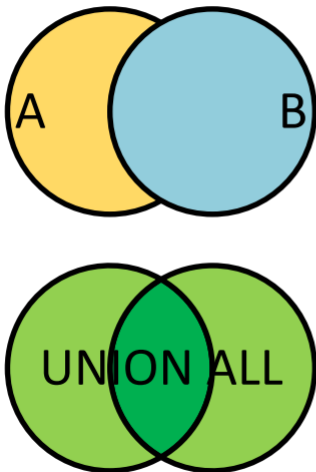
- Требуется дополнить таблицу T1 сведениями, содержащимися в таблице T2
- Оператор LEFT OUTER JOIN осуществляет проверку условия, заданного после ключевого слова ON
- Все строки из таблицы T2, для которых условие выполняется, объединяются со строками из таблицы T1
- Если для строки из левой таблицы T1 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



- 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 которых 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



- 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 которых 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)



- 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
```

- `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 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 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



- `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

- `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)

- `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