MySQL

Извлечение строк

Извлечение строк

- Оператор **SELECT** применяется для извлечения строк, выбранных из одной или нескольких таблиц
- Выражение select_expression задает столбцы, в которых необходимо проводить выборку
- При указании ключевых слов следует точно соблюдать порядок их следования

Конструкция Select

- SELECT [STRAIGHT_JOIN]
- [SQL_SMALL_RESULT | SQL_BIG_RESULT]
- SQL_BUFFER_RESULT]
- [SQL_CACHE | SQL_NO_CACHE]
- [SQL_CALC_FOUND_ROWS]
- [HIGH_PRIORITY]
- [DISTINCT | DISTINCTROW | ALL]
- select_expression,...
- [INTO {OUTFILE | DUMPFILE} 'file_name' options]
- FROM table_references
- [WHERE where_definition]
- [GROUP BY value [ASC | DESC], ...]
- [HAVING where_definition]
- [ORDER BY value [ASC | DESC], ...]
- [LIMIT [offset,] rows]

Урезанная конструкция

- SELECT
- select_expression,…
- FROM table_references
- [WHERE where_definition]
- [GROUP BY value [ASC | DESC], …]
- [HAVING where_definition]
- [ORDER BY value [ASC | DESC], …]
- [LIMIT [offset,] rows]

Жесткое объединение

- При указании параметра **STRAIGHT_JOIN** оптимизатор будет объединять таблицы в том порядке, в котором они перечислены в выражении FROM

Обработка выдачи

- **SQL_SMALL_RESULT** – для ускорения обработки запросов применяются временные таблицы
- **SQL_BIG_RESULT** – предупреждает оптимизатор о том, что ожидается большой результат выборки
- **SQL_BUFFER_RESULT** – заставляет оптимизатор поместить результирующую выборку во временную таблицу

Кэширование выдачи

- Параметр **SQL_CACHE** предписывает MySQL сохранять результат запроса в кэше запросов
- Параметр **SQL_NO_CACHE** запрещает MySQL хранить результат запроса в кэше запросов

Суммарная выборка

- Параметр **SQL_CALC_FOUND_ROWS** возвращает количество строк, которые вернул бы оператор SELECT, если бы не был указан LIMIT

Изменение приоритета

- При указании параметра **HIGH_PRIORITY** содержащий его оператор SELECT будет иметь более высокий приоритет, чем команда обновления таблицы

Работа с дублями

- Параметры (опции) **DISTINCT**, **DISTINCTROW** и **ALL** указывают, должны ли возвращаться дублирующиеся записи
  - ALL – возвращаются все встречающиеся строки
  - DISTINCT и DISTINCTROW – являются синонимами и указывают, что дублирующиеся строки в результирующем наборе данных должны быть удалены

Выбор столбцов

- Выражение **select_expression** задает участвующие в выборке столбцы
- Может принимать значения:
  - * – все столбцы
  - `tbl_name`.* – все столбцы таблицы
  - `tbl_name`.`col_name` – столбец таблицы
  - `col_name` – столбец
  - ‘abc’ – константа
  - 1 + 1 – результат выражения

выгрузка в файл

- **INTO OUTFILE** осуществляет запись выбранных строк в файл:
  - FIELDS TERMINATED BY – задает разделитель между столбцами
  - OPTIONALLY ENCLOSED BY – задает экранирование для строковых значений
  - LINES TERMINATED BY – задает разделитель строк
- **INTO DUMPFILE** запишет в файл только одну строку без разделителей

FROM

- Выражение **FROM** table_references задает таблицы, из которых надлежит извлекать строки
- Если указано имя более чем одной таблицы, следует выполнить объединение

Условия

- В выражении **WHERE** указываются условия для выборки строк из таблиц(ы)
- Можно использовать любую из функций, поддерживаемых в MySQL
- Не может накладывать условия на столбцы, полученные путем применения агрегатных функций (AVG, SUM, COUNT, MAX, MIN)

Группировка результатов

- Выражение **GROUP BY** группирует результаты выборки по заданным столбцам

Having

- В выражении **HAVING** указываются условия для выборки строк из таблиц(ы)
- Можно использовать любую из функций, поддерживаемых в MySQL
- Накладывает условия на столбцы, полученные путем применения агрегатных функций (AVG, SUM, COUNT, MAX, MIN)

Сортировка

- Выражение **ORDER BY** сортирует результат выполнения запроса по указанным столбцам
  - ASC – прямая сортировка
  - DESC – обратная сортировка

Ограничение на выборку

- Выражение **LIMIT** используется для ограничения количества строк, возвращенных командой SELECT
- LIMIT принимает один или два числовых аргумента
  - начало первой возвращаемой строки
  - количество возвращаемых строк
- Аргументы должны быть целочисленными константами

Пример

- SELECT `teacher_surname`, `teacher_name`
  INTO OUTFILE 'C:\\2.txt'
  FIELDS TERMINATED BY ',' OPTIONALLY
  ENCLOSED BY '"'
  LINES TERMINATED BY "\n"
  FROM `teacher`
  WHERE `teacher_id` IS NOT NULL
  GROUP BY `teacher_id`
  ORDER BY `teacher_surname` ASC
  LIMIT 0, 5

Внутреннее объединение таблиц

- Для создания внутреннего соединения между таблицами, необходимо в блоке WHERE указать через «=» столбцы по которым будут объединяться таблицы
- При указании условия название столбца пишется после названия таблицы, в которой этот столбец находится (через точку)
- `tab_1`.`col_id` = `tab_2`.`col_id`

явное объединение таблиц

- Для создания явной связки между таблицами используется оператор **JOIN**
- Операция соединения предназначена для обеспечения выборки данных из двух таблиц и включения этих данных в один результирующий набор
- При необходимости соединения не двух, а нескольких таблиц, операция соединения применяется несколько раз (последовательно)

Отличительные особенности

- в схему таблицы-результата входят столбцы обеих исходных таблиц (таблиц-операндов), то есть схема результата является «сцеплением» схем операндов
- каждая строка таблицы-результата является «сцеплением» строки из одной таблицы-операнда со строкой второй таблицы-операнда

Виды оператора JOIN

- Выделяют 4 вида оператора JOIN:
  - INNER JOIN – симметричное соединение таблиц
  - RIGHT JOIN – правостороннее соединение таблиц
  - LEFT JOIN – левостороннее соединение таблиц
  - CROSS JOIN – симметричное соединение таблиц по принципу каждый с каждым

Таблицы

City (Города)

Id	Name
1	Москва
2	Санкт-Петербург
3	Казань

Person (Люди)

Name	CityId
Андрей	1
Леонид	2
Сергей	1
Григорий	4

INNER JOIN

- **SELECT** *
- **FROM** Person
- **INNER JOIN**
  - City **ON** Person.CityId = City.Id

Person.Name	Person.CityId	City.Id	City.Name
Андрей	1	1	Москва
Леонид	2	2	Санкт-Петербург
Сергей	1	1	Москва

LEFT JOIN

- **SELECT** *
- **FROM** Person -- *Левая таблица*
- **LEFT OUTER JOIN** City -- *Правая таблица*
  - **ON** Person.CityId = City.Id

Person.Name	Person.CityId	City.Id	City.Name
Андрей	1	1	Москва
Леонид	2	2	Санкт-Петербург
Сергей	1	1	Москва
Григорий	4	NULL	NULL

RIGHT JOIN

- **SELECT** *
- **FROM** Person -- *Левая таблица*
- **RIGHT OUTER JOIN** City -- *Правая таблица*
  - **ON** Person.CityId = City.Id

Person.Name	Person.CityId	City.Id	City.Name
Андрей	1	1	Москва
Сергей	1	1	Москва
Леонид	2	2	Санкт-Петербург
NULL	NULL	3	Казань

CROSS JOIN

- **SELECT** *
- **FROM** Person
- **CROSS JOIN** City

- *ИЛИ*

- **SELECT** *
- **FROM** Person, City

Результат

Person.Name	Person.CityId	City.Id	City.Name
Андрей	1	1	Москва
Андрей	1	2	Санкт-Петербург
Андрей	1	3	Казань
Леонид	2	1	Москва
Леонид	2	2	Санкт-Петербург
Леонид	2	3	Казань
Сергей	1	1	Москва
Сергей	1	2	Санкт-Петербург
Сергей	1	3	Казань
Григорий	4	1	Москва
Григорий	4	2	Санкт-Петербург
Григорий	4	3	Казань