**Руководство по стилю SQL**

**Общие рекомендации по написанию кода:**

* **Идентификаторы и имена должны быть осмысленными и написанными в единой стилистике;**
* **Пробелы и отступы должны быть логично расставлены;**
* **Даты и время должны соответствовать стандарту ISO 8601 (ГГГГ-ММ-ДД ЧЧ:ММ:СС.ССССС.);**
* **Функции SQL должны быть стандартными, а не специфичными;**
* **Код должен быть лаконичным и без излишеств;**
* **Комментарии должны стоять либо перед блоком кода на отдельной строке в стиле языка C (/\* *комментарий* \*/), либо после кода на той же строке (--*комментарий*).**

|  |  |
| --- | --- |
| 1  2  3 | **SELECT** name *–– ФИО*  **FROM** student  **WHERE** name = 'Гарифуллина Валерия Руслановна'; |

|  |  |
| --- | --- |
| 1  2  3  4 | */\* Обновление данных студента \*/*  **UPDATE** student  **SET** name = 'Максудов Эмиль Альбертович'  **WHERE** name = 'Гарифуллина Валерия Руслановна'; |

**Не следует использовать:**

* Префиксы и венгерскую нотацию;
* Множественное число;
* Идентификаторы в кавычках;
* Принципы **объектно-ориентированного проектирования.**

**Соглашения о наименованиях**

**Общее:**

* Имя должно быть уникальным и не должно совпадать с ключевыми словами;
* Используйте в именах только буквы, цифры и символ подчёркивания;
* Имя не должно начинаться и заканчиваться подчёркиванием;
* **Избегайте** многократного повтора подчёркиваний;
* **По возможности не используйте** сокращения.

|  |  |
| --- | --- |
| 1  2 | **SELECT** first\_name, last\_name  **FROM** teacher; |

**Таблицы:**

* Используйте собирательные названия;
* Не допускайте совпадений названия таблицы с названиями её столбцов.

**Столбцы:**

* Названия должны иметь форму единственного числа;
* Название не должно совпадать с названием таблицы;
* Названия должны начинаться с строчной буквы.

**Псевдонимы:**

* Имя псевдонима должно быть связано с объектом по смыслу;
* Имя псевдонима может собираться из первых букв каждого слова наименования объекта;
* Добавьте цифру, если псевдоним уже существует;
* Для данных, получаемых при помощи агрегирующих функций, используйте имена, похожие на наименования столбцов таблицы.

|  |  |
| --- | --- |
| 1  2  3  4 | **SELECT** first\_name **AS** fn  **FROM** teacher **AS** s1  **JOIN** student **AS** s2  **ON** s2.id = s1.id; |

|  |  |
| --- | --- |
| 1  2 | **SELECT** **SUM**(s.monitor\_tally) **AS** monitor\_total  **FROM** staff **AS** s; |

Название хранимых процедур должно содержать в себе глагол.

**Универсальные суффиксы для наименований:**

* \_id — уникальный идентификатор;
* \_status — флаг или статус;
* \_total — общее количество;
* \_num — число;
* \_name — имя;
* \_seq — непрерывная последовательность значений;
* \_date — дата;
* \_tally — счётчик;
* \_size — размер или величина;
* \_addr — адрес.

**Синтаксис запросов**

Ключевые слова пишутся прописными буквами, например SELECT, WHERE, AS и др.

Не рекомендуется использовать сокращённый вариант слова, если существует его полный вариант. Также желательно не использовать специфичные ключевые слова для каких-либо СУБД. Это сделает ваш более код переносимым.

|  |  |
| --- | --- |
| 1  2  3 | **SELECT** age  **FROM** student **AS** p  **WHERE** p.birthday > '2003-02-05'; |

Для опрятности кода и удобного чтения оного можно и даже нужно использовать пробелы. Однако следует избегать дубликатов пробела. Пробелами выделяют важные ключевые слова, чтобы выровнять их по правому краю.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | (**SELECT** f.species\_name,  **AVG**(f.height) **AS** average\_height, **AVG**(f.diameter) **AS** average\_diameter  **FROM** flora **AS** f  **WHERE** f.species\_name = 'Банксия'  **OR** f.species\_name = 'Шеос'  **OR** f.species\_name = 'Плетень'  **GROUP** **BY** f.species\_name, f.observation\_date)  **UNION** **ALL**  (**SELECT** b.species\_name,  **AVG**(b.height) **AS** average\_height, **AVG**(b.diameter) **AS** average\_diameter  **FROM** botanic\_garden\_flora **AS** b  **WHERE** b.species\_name = 'Банксия'  **OR** b.species\_name = 'Шеос'  **OR** b.species\_name = 'Плетень'  **GROUP** **BY** b.species\_name, b.observation\_date); |

Также следует расставлять пробелы:

* вокруг знака равно;
* после запятых;
* вокруг кода, написанного внутри апострофов, если тот не внутри скобок, или без последующих запятой или точки с запятой.

|  |  |
| --- | --- |
| 1  2  3  4 | **SELECT** b.title, b.author  **FROM** books **AS** b  **WHERE** b.title = 'Леса Финляндии'  **OR** b.title = 'Гонки на байдарках'; |

Не пренебрегайте переносом строки:

* перед AND или OR;
* после точки с запятой (для разделения запросов);
* после каждого основного ключевого слова;
* после запятой (при выделении логических групп столбцов).

|  |  |
| --- | --- |
| 1  2  3 | **INSERT** **INTO** albums (title, release\_date)  **VALUES** ('Леса Финляндии', '1990-01-01 01:01:01.00000'  ('Гонки на байдарках', '2008-01-01 01:01:01.00000'); |

|  |  |
| --- | --- |
| 1  2  3 | **UPDATE** albums  **SET** release\_date = '1990-01-01 01:01:01.00000'  **WHERE** title = 'Гонки на байдарках'; |

|  |  |
| --- | --- |
| 1  2  3  4  5 | **SELECT** b.title,  b.release\_date, b.production\_date *-- сгруппированные даты*  **FROM** books **AS** b  **WHERE** b.title = 'Леса Финляндии'  **OR** b.title = 'Гонки на байдарках'; |

Также очень важно придерживаться стандартам расстановки отступов. Объединения и подзапросы должны располагаться по правую часть «коридора», при этом в подзапросах действуют все те же правила написания кода, что и в любых других запросах.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | **SELECT** r.name  **FROM** riders **AS** r  **INNER** **JOIN** cars **AS** c  **ON** r.car\_vin\_num = c.vin\_num  **AND** c.engine\_tally > 5  **INNER** **JOIN** crew **AS** **cr**  **ON** r.crew\_chief\_name = **cr**.name  **AND** **cr**.chief = 'Да'; |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | **SELECT** r.name,  (**SELECT** **MAX**(**YEAR**(championship\_date))  **FROM** champions **AS** **c**  **WHERE** **c**.name = r.name  **AND** **c**.confirmed = 'Да') **AS** last\_championship\_year  **FROM** riders **AS** r  **WHERE** r.name **IN**  (**SELECT** **c**.name  **FROM** champions **AS** **c**  **WHERE** **YEAR**(championship\_date) > '2008'  **AND** **c**.confirmed = 'Да');  **AND** **c**.chief = 'Да'; |

**Советы:**

* Вместо частого использования условий AND, используйте BETWEEN;
* Вместо частого использования условий OR, используйте IN();
* Используйте CASE, если значение должно быть интерпретировано до окончания выполнения запроса;
* Избегайте временных таблиц и UNION.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | **SELECT** **CASE** postcode  **WHEN** 'МСК1' **THEN** 'Москва'  **WHEN** 'СПБ1' **THEN** 'Санкт-Петербург'  **END** **AS** city  **FROM** locations  **WHERE** country = 'Россия'  **AND** opening\_time **BETWEEN** 8 **AND** 10  **AND** postcode **IN** ('МСК1', 'СПБ1', 'ЕКБ1', 'КЗН1'); |

**Создание таблиц (CREATE)**

При создании столбцов таблицы старайтесь не использовать специфичные типы данных. Это может навредить вам при переносе или при переходе на более старую версию этой же СУБД.

Для работы с числами с плавающей точкой используйте типы данных FLOAT или REAL, в иных случаях – DECIMAL или NUMERIC.

Тип данных значения по умолчанию должен совпадать с типом данных столбца.

**Ключи:**

* Ключ обязан быть уникальным;
* Все ключи в системе должны быть согласованы по типу данных;
* Ключ должен быть простым;
* Рекомендуется использовать числовые ключи;
* У каждой таблицы должен быть хотя бы один ключ;
* Первичный ключ должен быть объявлен первым полем в таблице.

**Ограничения:**

* Ограничениям следует давать адекватные имена, из которых можно понять его логику;
* Ограничения должны быть объявлены строго ниже столбца, с которым они связаны;
* В случае ограничений, затрагивающих несколько столбцов, старайтесь **объявлять** их как можно ближе к описанию последнего из них;
* Если диапазон числовых значений для столбца известен, используйте CHECK() для ограничения внесения в базу некорректных данных;

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | **CREATE** **TABLE** student (  **PRIMARY** **KEY** (student\_num),  student\_num INT(5) **NOT** **NULL**,  name VARCHAR(100) **NOT** **NULL**,  exam\_results INT(3) **NOT** **NULL**,  **CONSTRAINT** exam\_results\_range  **CHECK**(exam\_results **BETWEEN** 0 **AND** 310)  ); |

Не применяйте объектно-ориентированные принципы, поскольку они далеко не всегда оптимально ложатся на реляционную модель базы данных.

Не разбивайте данные, логически принадлежащие одной таблице, по разным таблицам на основании условностей, например архивации по времени или географическому расположению. Впоследствии для работы с несколькими подобными таблицам придётся часто использовать UNION вместо простых запросов к одной таблице, что крайне не рекомендуется.