Посібник зі стилю SQL - SQL Style Guide

Вступ

Хочу представити вашій увазі посібник зі стилю написання SQL запитів, процедур, представлень, форматуванню, певних правил по неймінгу та інше.

Рекомендації, описані в цьому посібнику, багато в чому перетинаються з описаними в книзі Джо Селко " Стиль програмування Джо Селко на SQL " (оригінал: SQL Programming Style). Проте були внесені певні зміни під наші реалії.

Тому деякі правила можуть відрізнятися від всесвітньо прийнятих, проте в нашому проєкті вони вже стали фундаментальними

Основні положення

Гарна практика:

- **Ідентифікатори та імена** - зрозумілі, описові та витримані в єдиному стилі;
- **Пробіли та відступи** - логічно розставлені, так щоб було простіше читати (властиві природній мові);
- **Дата та час** - згідно зі стандартом ISO 8601: `YYYY-MM-DD HH:MM:SS.SSSSS` ;
- **Функції SQL** - З міркувань портативності намагайтеся використовувати лише стандартні функції SQL замість функцій постачальника;
- **Код** - лаконічний і без надмірностей як, наприклад: непотрібні лапки або дужки або недоречне використання оператора `WHERE` (який можна отримати інакше);
- **Коментарі** - додавайте коментарі де це необхідно, бажано в стилі C (`/*` коментар `*/`), якщо це неможливо, то використовуйте `--`;
- **CamelCase** - використовуємо в назвах об'єктів;
- **Префікси** - використовуємо в назвах представлення (`v_`), синонімів (`s_`), процедур (`proc_`), функцій (`func_`), тригерів (`tr_`) і т.д.;
- **Стандартні імена стовпців** - використовуйте назви зі списку, щоб вони були зрозумілі для всіх;
- **Форматування запитів** - для цього використовуйте SQL Formatter від RedGate, зі стилем *Right aligned*

Не бажана практика:

- **Угорська нотація**;
- **Множина** - краще використовувати збірні поняття, що природніше звучать. Наприклад, `staff` замість `employees` або `people` замість `individuals`;
- **Ідентифікатори в лапках** - якщо вони обов'язково потрібні, використовуйте подвійні лапки, визначені в стандарті SQL-92 з метою кращої платформонезалежності;
- **Принципи об'єктно-орієнтованого проектування** - не потрібно переносити на SQL або структуру бази даних.

Правила найменування

Загальне

- **Переконайтеся**, що *ім'я є унікальним* і його немає у списку зарезервованих ключових слів;
- **Обмежуйте** довжину імені 30 байт (це *30 символів*, якщо не використовується багатобайтний набір символів);
- **Починайте** імена з великої літери та не закінчуйте символом підкреслення;
- **Використовуйте** лише літери та цифри в іменах;
- **Уникайте** символів підкреслення;
- **Використовуйте** CamelCase там, де ви поставили пробіл в реальному житті (наприклад, `first name` стане `FirstName`);
- **Уникайте скорочень**, якщо їх все ж таки потрібно використовувати, переконайтеся в тому, що вони загальнозрозумілі;
- **Пріоритетне використання** англійських назв, замість транслітерації.

Таблиці

- **Використовуйте** загальні іменники або, уникайте форму множини. Наприклад, `Staff` і `Employees` (за зменшенням переваги);
- **Не використовуйте** описові префікси виду `tbl_` та угорську нотацію загалом;
- **Не допускайте** збігів назви таблиці з назвою будь-якого її стовпців;
- **Уникайте**, по можливості, поєднання назв двох таблиць для побудови таблиці відносин. Наприклад, замість назви `CarsMechanics` краще підійде `Services`.

Стовпці

- Назви завжди **давайте** в однині;
- По можливості **не використовуйте** `id` як первинний ідентифікатор таблиці;
- **Не створюйте** у таблиці стовпців з такою самою назвою, як у неї самої;
- Назви завжди **пишіть** з великої літери.

Псевдоніми (аліаси)

- **Повинні** так чи інакше бути пов'язані з об'єктами, чи виразами псевдонімом яких є;
- Ім'я кореляції **зазвичай складається** з перших літер кожного слова імені об'єкта;
- Якщо таке ім'я вже існує, то надайте йому смислового забарвлення, наприклад `[dbo].[FilialMail]` може бути `fmp` - дані по посчатчальнику та `fmf` - дані по філіалу;
- Завжди **використовуйте** ключове слово `AS` для кращого читання;
- Для обчислюваних даних (`SUM()` або `AVG()`) **використовуйте** такі імена, які ви дали б, якби вони були стовпцями в таблиці.

```
SELECT za.SapCounter AS sp,
       SUM(zla.Kolvo) AS KolvoTotal
  FROM ZakazAuto AS za
  JOIN ZakazLinesAuto AS zla
    ON za.id = zla.ZakazID
 GROUP BY za.SapCounter
```

Збережені процедури

- Ім'я **має** містити дієслово;
- **Використовуйте** префікс `proc_`.

Універсальні суфікси

Нижче наведені суфікси універсальні, що гарантує простоту розуміння значення стовпців з коду SQL.

- `...ID` -- унікальний ідентифікатор, наприклад, первинний ключ;
- `...Status` -- прапор або будь-який статус, наприклад `SapStatus`;
- `...Total` - загальна кількість або сума значень;
- `...Name` -- будь-яке ім'я, наприклад `LagerName`;
- `...Date` - колонка, що містить дату;
- `...Addr` - фізична або абстрактна адреса, наприклад `PostAddr`;
- ….

Синтаксис запитів

Зарезервовані слова

Зарезервовані ключові слова завжди пишіть великими літерами, наприклад `SELECT, WHERE`.

Намагайтесь не використовувати скорочений варіант ключового слова, якщо є повний. Наприклад, використовуйте `ABSOLUTE` замість `ABS`.

Намагайтеся не використовувати специфічні для будь-якого постачальника СУБД ключові слова, якщо в ANSI SQL є ключові слова, які виконують такі ж функції. Це зробить ваш код більш платформонезалежним.

```
SELECT TOP 10 ZakazID AS z
  FROM ZakazLinesAuto AS zla
```

Порожній простір (пробіли)

Для кращої зручності читання коду важливо правильно використовувати пробільні символи. Не потрібно нагромаджувати код або видаляти прогалини, властиві природній мові.

Пробіли

Для вибудовування коду потрібно використовувати пробіли, щоб усі ключові слова кореня закінчувалися на одній межі символу. Це утворює додатковий простір посередині, що дозволяє читачам легко переглядати код і відокремлювати ключові слова від деталей реалізації. Такі відступи небажані в типографії, але тут корисні.

```
SELECT zc.lagerid,
       SUM(zc.kolvo_after) AS KolvoTotal,
       zc.rasf,
       SUM(zc.kolvo_after / zc.rasf) AS KolvoRasfTotal
  FROM dbo.ZakazClientsRibZakaz AS zcrz
  JOIN ZakazClients zc
    ON zcrz.ZakazClientsID = zc.ID
 WHERE zc.ZakazClientsID = 96567017
   AND zc.StatusClOrder  = 5
   AND zc.kolvo_after    > 0
 GROUP BY zc.LagerID,
          zc.Rasf
```

Зверніть увагу, що `SELECT`, `FROM` тощо вирівнюються по правому краю, а фактичні назви стовпців та конкретні відомості щодо реалізації вирівнюються за лівим краєм.

Хоча цей список не вичерпний, завжди включайте пробіли:

- до і після знака “дорівнює” (`=`);
- після коми (`,`);
- до відкриваючого і після закриваючого апострофів (`` ` ``), за умови, що вони не в дужках, не з комою та не крапкою з комою;

Міжрядковий інтервал

Завжди робіть перенесення рядка:

- **перед** `AND` чи `OR`;
- **після** крапки з комою (для поділу запитів);
- **після** кожного основного ключового слова;
- **після** коми (при виділенні логічних груп стовпців).

Якщо всі ключові слова вирівняні з правого боку, а значення з лівого боку - у середині запиту створюється рівномірний простір, що також значно полегшує швидке читання та сканування запиту.

Відступи

Щоб забезпечити читання SQL, важливо дотримуватися стандартів відступів.

JOIN

Об’єднання (`JOIN`) мають почитнатися з нового рядка та вирівняні по лівому краю слова `FROM`

```
SELECT da.*
  FROM Data.dbo.DogovorHeader AS dh
  LEFT JOIN Data.dbo.DogovorArticles AS da
    ON dh.id = da.DogovorID
 WHERE da.lagerid IN ( 453345 )
   AND dh.DateBegin  < GETDATE()
   AND dh.Datefinish > GETDATE()
```

Підзапити

Підзапити також мають бути вирівняні з правого боку з додатковим відступом, а потім викладені у такому ж стилі, що й будь-який інший запит. Іноді має сенс розмістити дужку, що закриває, на новому рядку в тій самій позиції символу, що і його початковий партнер. Це особливо зручно, якщо у вас є вкладені підзапити.

```
SELECT da.*
  FROM Data.dbo.DogovorHeader AS dh
  LEFT JOIN Data.dbo.DogovorArticles AS da
    ON dh.id = da.DogovorID
 WHERE da.lagerid IN ( 453345 )
   AND dh.DateBegin  < GETDATE()
   AND dh.Datefinish > GETDATE()
   AND (ISNULL(dh.TypeLogistic, 0) NOT IN (   SELECT ValueInt
                                               FROM GlobalParams AS gp
WITH (NOLOCK)
                                               INNER JOIN
GlobalParamsLines AS gpl WITH (NOLOCK)
                                                 ON gp.idParams = gpl.
idParams
                                              WHERE gp.ParamName =
'LOGYSTIC_TYPE_GATHER_TRANS'
                                                AND gp.Enabled   = 1
                                                AND gpl.Enabled  = 1 ))
```

Формальні тонкощі

- **Використовуйте** `BETWEEN`, де можливо, замість нагромадження умов `AND`;
- Так само намагайтеся **використовувати** `IN()` замість `OR`;
- **Використовуйте** `CASE`, якщо значення має бути інтерпретовано до закінчення запиту. За допомогою `CASE` можна формувати складні логічні структури;
- По можливості **уникайте** використання `UNION`;
- **Перевіряйте** стовпці таблиць на наявність `NULL` значень, використовуйте перевірку `ISNULL` (пам'ятайте, що `NULL<>NULL`);
- При побудові складних запитів **використовуйте** темпові таблиці;
- При роботі з темповими таблицями **не забувайте** про індекси.

Рекомендації

Робота з таблицями

- По можливості **не використовуйте** специфічні для тієї чи іншої СУБД тип даних . Це може негативно зашкодити платформонезалежності;
- Стовпці таблиць **мають містити** опис, приклад можна подивитися в таблиці `[AUTOORDER].[dbo].[ZakazClients]`;
- Для роботи з плаваючою точкою **використовуйте** тільки `REAL` або `FLOAT`, а там, де немає потреби в подібних обчисленнях, завжди використовуйте `NUMERIC` та `DECIMAL`. Помилки округлення в операціях з точкою, що плаває, можуть виявитися дуже недоречними;
- **Намагайтеся** створювати таблиці без значень стовпців `NULL`;
- **Використовуйте** значення за замовчуванням (`DEFAULT`), якщо заздалегідь відомо початкове значення;
- Значення за замовченням завжди **повинно збігатися** за типом стовпця. Якщо, скажімо, стовпець оголошено як `DECIMAL`, не потрібно за умовчанням вказувати значення типу `INTEGER`;
- Кожна таблиця **повинна містити** хоча б один ключ для таблиць, які мають реплікуватися це обов'язково;
- По можливості **не використовуйте** поле `ID` в якості ключа таблиці;
- Якщо діапазон числових значень для стовпця відомий, **використовуйте** `CHECK()`, щоб запобігти внесенню в базу некоректних даних або прихованого відсікання частини значення занадто великих даних. Зазвичай перевірка робиться на те, що значення більше від нуля;
- Якщо таблиця використовується в сиспакеті, або відображення даних на клієнтів, не забудьте **перевірити** наявність у системного користувача прав на неї.

Робота з процедурами

- Процедура обов'язково **має містити** логування процесу на основі `[Monitoring].[dbo].[proc_LoggingStart]` та `[Monitoring].[dbo].[proc_LoggingRun]`;
- Якщо процедура використовується з FZClient для виведення даних або розрахунків, які запускаються з вікна ПО, то її назва **повинна співпадати** з назвою операції;
- Процедура АО для розрахунків **має починатися** з префіксу `proc_`;
- В тілі процедурами **має бути зафіксований**: автор, дата створення, короткий опис, історія змін та приклад запуску;
- Якщо процедура має вхідні параметри, то вони **мають бути описані** та наведені приклади;
- **Уникайте** багато коментарів, використовуйте першочергово коментарі типу `/* */`;
- Гарною практикою є **використання** `;` по завершенню запита (Microsoft планує зробити це обовьязковим);
- Уважно **слідкуйте** за умовами які накладаються на стовпці формату `[datetime]`, для конвертації в рамках одного процессу використовуйте тільки один вид перетворення `[cast]` або `[convert]`, частіше ми в своєму проєкті використовуємо `[cast]`;
- **Не забувайте** використовувати `WITH RECOMPILE` за необхідності;
- Завжди **використовуйте** `SET NOCOUNT ON`, це дозволить підвищити продуктивність процедури;
- Якщо потрібно виконати декілька послідовних перевірок `ISNULL`, то **використайте** функцію `COALESCE`;
- **Уникайте** Magic Number (параметри зашиті напряму в код, по типу статуса замовлення) в коді, краще все виносити в змінні зі зрозумілою назвчою, коду більше, але його потім простіше супроводжувати.

Список стандартних імен стовпців:

Назва	Опис	Довідник
LagerID	унікальний ідентифікатор артикул	[Data].[dbo].[Lager]
LagerTypeID	тип артикула	[Data].[dbo].[LagerTypes]
SapStatusAssort	сап статус товара на філіалі	[Data].[dbo].[TypeSAPStatusAssortiment]
PostavshikID	унікальний ідентифікатор постачальника. Це універсальне поле, може використовуватися як і для зовнішній та і для внутрішніх постачальників	[Data].[dbo].[Adress] WHERE [Type]=1
PostavshikSAPID	Зовнішній ідентифікатор постачальника. Для Зовнішніх постачальників це ЕДРПО/ІПН, а для внутрішніх SapID	[Data].[dbo].[PropertyAdressChar] WHERE PropertyID = 62
GlobalIDRC	унікальний ідентифікатор Внутрішнього постачальника бізнесу РЦ	
FilID	внутрішній ідентифікатор філіала	[Data].[dbo].[FilialMail]
CodCli	зовнішній ідентифікатор філіала (SapID)	
DateToPost	Дата постачання товару	
DogovorID	унікальний ідентифікатор договору	[Data].[dbo].[DogovorHeader]
RealTypeZakaz	Тип замволення по класифікації Автоордера	[AUTOORDER].[dbo].[ZakazRealType]
TypeLogistic	логістичний тип	[Data].[dbo].[LogisticLevels]
Rasf	розфасовка товару	
RasfID	унікальний ідентифікатор розфасовки	[Data].[dbo].[RASF]
Barcode	штрихкод товару	
LU	логістичний юніт товару	
LV	логістичний варіант композиції товару на палеті	[Data].[dbo].[LogisticVariants]
SapCounter	номер замовлення для зовнішніх систем	[AUTOORDER].[dbo].[SAPCounters]
TypeFil	ідентифікатор типу (бізнеса) філіалу	[Data].[dbo].[TypeFil]
MacroID	ідентифікатор макро групи товару	[Data].[dbo].[MacroGroup]
ActivityID	унікальний ідентифікатор акції	[Data].[dbo].[Activities]

Store	залишок товару	
Reserve	резерв товару	
ShelfLife	термін придатності	