Оптимизация UDF

Задание 1

1. Функция F_EMPLOYEE_FULLNAME

- Эта функция вызывает другую функцию (F_EMPLOYEE_GET) для получения идентификатора сотрудника, а также делает выборку из таблицы Employee для построения полного имени. Каждый раз, когда эта функция вызывается, выполняется подзапрос, что может негативно сказаться на производительности, особенно при большом количестве записей.

- Если поле ID_EMPLOYEE не имеет индекса, это может значительно замедлить выборку, особенно в больших таблицах.

- Стоит устранить повторные запросов - использовать один SELECT для получения всех необходимых данных.

2. Функция F_WORKITEMS_COUNT_BY_ID_WORK

- Функция использует подзапрос для фильтрации по id_analiz. Это может быть неэффективно, если таблица analiz содержит большое количество строк. Возможно стоит использовать JOIN для улучшения производительности.

- Если в таблице workitem нет индексов на этих полях, это также может замедлить выполнение запросов.

3. Функция F_WORKS_LIST

- В функции несколько раз вызываются другие функции для подсчета элементов (F_WORKITEMS_COUNT_BY_ID_WORK) и формирования полного имени (F_EMPLOYEE_FULLNAME). Каждый из этих вызовов увеличивает время выполнения функции, особенно если результат не кэшируется.

4. Некоторые поля имеют типы данных VARCHAR, которые могут быть уменьшены по размеру. Например, varchar(50) может быть слишком большим для поля Name, если имена обычно короче. Это может привести к излишнему расходу памяти.

Задание 2.

Для повышения скорости выполнения запросов (помимо оптимизации функции F_WORKS_LIST) стоит добавить индексы на поля, которые часто используются для фильтрации и сортировки. К таким полям, к примеру, относятся:

1. Works.IS_DEL: это поле участвует в фильтрации, и добавление индекса может ускорить выборку актуальных записей, исключая удалённые заказы.

2. WorkItem.is_complit: это поле может использоваться для фильтрации завершенных и незавершенных исследований.

3. Analiz.is_group: это поле используется для фильтрации групповых и индивидуальных спецификаций.

Задание 3.

Если данные не обновляются слишком часто, можно внедрить механизм кэширования результатов запроса dbo.F_WORKS_LIST(). Это может быть реализовано с помощью промежуточных таблиц в базе данных, в которых хранятся результаты запроса, обновляемые через определённые интервалы времени.

Потенциальные недостатки: если данные меняются часто, кэш может устаревать, и поддержание актуальности данных станет дополнительной задачей. Это также увеличит сложность системы.