Лабораторная работа No 10/11. Блокировки чтения/записи и условные переменные

Задача — обеспечить конкурентный доступ к файлу по чтению/записи, используя метод ISAM (Index Sequential Access Method — метод индексного последовательного доступа), со стороны нескольких процессов.

Файл имеет базовую структуру простого склада товаров.

Этот файл, назовем его PRODUCT (префикс PROD), содержит заголовок и записи фиксированной длины. Каждая запись включает минимально следующие поля:

Поле	**Ключ**	**Примечание**
ID	P	Внутренний идентификатор товара (главный индекс)
Name	SM	Наименование товара
Code	SU	Код товара
Amount		Количество всего
Reserved		Зарезервировано

P — первичный ключ, уникальное значение, автоинкремент при добавлении.

SU — вторичный ключ, уникальное значение.

SM — вторичный ключ, неуникальное значение.

С каждым из ключей связан индексный файл, содержащий записи фиксированной длины с двумя полями, позволяющий по значению ключа получать прямой доступ к записи в файле за один или два шага чтения. Структура индексных файлов:

Индексный файл	Ключ	Значение	Примечание
PROD_MASTER	`PROD::ID`	Порядковый номер записи	
PROD_Name	`PROD::Name`	Главный индекс	
PROD_Code	`PROD::Code`	Главный индекс	

Индексный файл PROD_MASTER может быть упорядочен либо сортировкой, либо хешированием, поскольку поддерживает только прямой доступ. Индексные файлы PROD_Name и PROD_Code упорядочиваются сортировкой, поскольку помимо прямого доступа должны поддерживать и последовательный.

На этих файлах должен поддерживаться ряд атомарных операций:

	Операция	**Описание**
1	ADD_RECORD	Добавить запись
2	DEL_RECORD	Удалить запись
3	GET_RECORD	Получить запись по главному индексу
4	PUT_RECORD	Изменить запись в главном индексе
5	GET_PRIMARY	Получить главный индекс по имени или коду
6	SET	Установить курсор доступа в порядке вторичного индекса
7	NEXT	Получить следующую запись в порядке вторичного индекса
8	PREV	Получить предыдущую запись в порядке вторичного индекса

В рамках лабораторной 10 реализуются только операции 1, 2, 3, 4 и 5.

Несколько процессов (копии программы в разных виртуальных терминалах) должны используя вышеуказанные операции читать и писать в файл PRODUCT, не мешая друг другу.

Детальное описание операций

ADD_RECORD — добавить запись

Запись добавляется в файл PRODUCT, а также в индексные файлы в порядке соответствующих ключей. Добавление выполняется как атомарная операция над всеми измененными файлами.

Запись добавляется на место удаленной ранее записи, если такие есть, либо в конец файла. Для добавления новой записи на место удаленной можно поддерживать файл учета удалений с организацией в виде стека, в который помещаются порядковые номера удаленных записей.

Заголовок файла содержит поле, которое используется для автоинкрементной генерации первичного ключа и поле количества записей в файле, которые должны атомарно изменяться при добавлении.

DEL_RECORD — удалить запись

Запись удаляется из файла PRODUCT, а также из индексных файлов в порядке соответствующих ключей. Удаление выполняется как атомарная операция над всеми измененными файлами.

Порядковый номер удаленной записи заносится в файл учета удалений.

Поле количества записей в заголовке файла изменяется.

GET_RECORD — получить запись по главному индексу

Возвращает запись из файла PRODUCT, либо ошибку.

PUT_RECORD — изменить запись в главном индексе

Записывает запись, ранее полученную из файла PRODUCT либо в операции GET или NEXT/PREV, на прежнее место в главном индексе и перестраивает все затронутые вторичные индексы. Главный индекс при этом не меняется и соответствующий файл перестройки не требует. Например, если менялось PROD::Name, перестраивается индекс PROD_Name, если поменялось «Amount», ничего не перестраивается.

Возвращает 0, либо ошибку.

GET_PRIMARY — получить главный индекс по имени или коду

Возвращает главный индекс из соответствующего вторичного индекса, либо ошибку, если нет такого значения в поле ключа.

Замечания

Индексные файлы кешируются — отображаются на память в виде контейнеров std::map.

При изменении индекса в кеше (вставка/удаление в контейнере) соответствующий индексный файл приводится в согласованное с кешем состояние.

Для уведомление других процессов об изменении индексного файла можно использовать fcntl(F_NOTIFY).

Протокол изменения записи

1) получаем запись для изменения в REC;

2) сохраняем копию в SAV;

3) делаем что-нибудь с полями, не затрагивая ID (поскольку будем ее возвращать на старое место в первичном индексе);

4) блокируем запись на время для изменения

5) получаем ту же самую запись в REC2 для проверки;

6) если SAV != REC2 (запись конкурентно изменена), снимаем блокировку и идем на пункт 1

7) записываем REC назад и снимаем блокировку.

SET — установить курсор

NEXT — получить следующую запись в порядке вторичного индекса

Возвращает следующую запись, значение ключа которой равно или больше указанного в вызове. в порядке вторичного индекса, значение главный индекс из соответствующего вторичного индекса, либо ошибку, если нет такого значения в поле ключа.

PREV — получить предыдущую запись в порядке вторичного индекса