Робота з структурованими файлами з індексно-довільним доступом (без СУБД).

Мета роботи – навчитися працювати з структ. файлами без використання СУБД на основі лише мови C/C++.

В якості предметної області слід вибрати ту, яку ви обрали для ER-моделі. Потрібно обрати два об'єкти з ER-моделі, які сполучені зв'язком з типом відображення 1-∞. На основі об'єкту, до якого спрямована 1, формується master-file, а на основі іншого – slave-file. Програма, яка працює з цими файлами, має включати до свого складу інтерфейс користувача (на основі командного рядка чи віконний), а також наступні функції:

1. функція читання заданого запису та заданих підзаписів на основі прямого індексного доступу; (get-m, get-s)// підзаписи – записи з slave-file.
2. функція вилучення заданого запису чи заданого підзапису; при вилученні запису необхідно вилучати всі його підзаписи; (del-m, del-s)
3. функція оновлення значення заданого поля заданого запису чи заданого підзапису; (update-m, update-s)
4. функція внесення запису та/чи підзаписів в файли (insert-m, insert-s).
5. функція підрахунку кількості записів, підзаписів загалом, а також кількості підзаписів для кожного запису (calc-m, calc-s).
6. утиліти (ut-m, ut-s) читання та друку всіх полів (включаючи службові) master-file та slave-file.

Для пришвидшення здачі лабораторної рекомендується підготувати текстовий з командами для вашої програми з наступним сценарієм:

1. Ввід 5 master-записів.
2. Для 3-х master-записів ввести 1, 2 та підлеглі записи.
3. ut-m, ut-s.
4. Вилучити master-запис з двома підлеглими.
5. Вилучити підлеглий запис від master-запису з 3-ма підлеглими.
6. ut-m, ut-s.
7. Ввід ще одного master-запису та підлеглого до нього запису.
8. ut-m, ut-s.
9. Оновити поле master-запису та поле підлеглого запису.
10. ut-m, ut-s.

Все вище описане можна також зробити і вручну.

Розглянемо вищезгадане завдання на прикладі. Нехай задана ER-модель предметної області «постачальник-деталі» у вигляді 2-х об'єктів (постачальники та деталі) і бінарного зв'язку «поставка» з типом відображення □-□. Відповідна реляційна модель буде мати вигляд:

П(КП,прізв,статус,місто) – постачальники;

Д(КД,назва,вага,колір,місто) – деталі;

ПД(КП,КД,ціна,кількість) – поставки.

На файловому рівні будемо працювати тільки з 2-ма файлами Постачальники(S) та Поставки(SP).

Індексований файл S складається з 2-х частин: власне файл S.fl та індексна таблиця S.ind, яка має 2 стовпчики: значення ключа (тобто КП) і адреса відповідного запису в основному файлі S.fl. Записи індексної таблиці мають бути відсортовані, наприклад, за зростанням значення ключа. При деяких режимах роботи (наприклад, пошуку) індексна таблиця може повністю зберігатися в оперативній пам’яті з наступним перезаписом у зовнішню пам’ять. Її обсяг може бути обмеженим, наприклад, 10 чи 20 записів. Індексну таблицю слід переписати в оперативну пам'ять при старті програми та вивантажити до файлу S.fl при її завершенні.

Оскільки у нашому випадку записи мають фіксовану довжину, то замість адреси в байтах можна використовувати номер запису, а адресу отримувати в результаті множення на довжину запису.

Другий файл SP містить записи поставок деталей, і всі поставки від одного постачальника повинні бути зв’язані у список (у зовнішній пам’яті). Один з варіантів організації такого списку: кожен запис постачальника (файл S.fl) містить спеціальне поле з адресою першого запису такого списку в файлі SP.fl, а далі кожен запис списку містить поле з адресою наступника. Можливо, доцільним буде мати у кожного постачальника (файл S.fl) поле з числом його поставок.

Область збору сміття (ОЗС).

При вилученні запису його не слід вилучати фізично (великі витрати часу при великих об’ємах файлів), а лише - логічно, тобто відмітити в спеціальному службовому полі, що цей запис є вилученим, наприклад, 0 – присутній, а 1 – вилучений. Щоб мати можливість ввести на це місце інший запис при виконанні функції insert адресу (чи його номер) вилученого запису слід зберегти у спеціальній області, яку називають *сміттям*. Вона може розташовуватись на початку файлу розміром 1-2 записи. Сюди також часто вносять адресу (чи номер) наступного за останнім записом файлу для вводу запису при відсутності логічно вилучених записів. Пусті поля області можна заповнювати від’ємними числами, наприклад -1. ОЗС як і індексну таблицю доцільно тримати протягом всього часу роботи програми тримати в оперативній пам’яті у вигляді масива, записуючи до файлу тільки перед завершенням роботи.

Функція get-m – по коду постачальника знаходимо в індексній таблиці номер запису, позиціонуємось за цим номером у файлі S.fl і читаємо відповідний запис. Потім виводяться всі поля користувача; більш складний варіант: пропонувати користувачу вказати ім'я поля, значення якого потрібно вивести. Доцільно також передбачити режим виводу всіх записів файлу.

Функція get-s – спочатку знаходимо запис master-file (функція get-m), потім, використовуючи номер 1-ої поставки (із запису постачальника) у списку поставок послідовним перебором знаходимо потрібний підзапис, наприклад, по коду деталі. Щоб уникнути послідовного перебору, тут можна теж використати індексну таблицю для файлу SP.fl. Це, звичайно, пришвидшить пошук, але потребує додаткових зусиль для підтримки цієї індексної таблиці.

Функція del-m : спочатку виконуємо get-m; в індексній таблиці (оскільки вона має бути відсортована) зміщуємо хвіст на один рядок вгору; в файлі S.fl можемо взяти останній запис і перемістити його на місце, яке звільняється, з відповідною корекцією в індексній таблиці. Крім того, необхідно виконати функцію del-s по всім підлеглим підзаписам, бо не може бути поставок від неіснуючого постачальника.

Функція del-s: спочатку виконуємо get-s; в файлі SP.fl можемо взяти останній запис і перемістити його на місце, яке звільняється, з відповідною корекцією 2 –х списків.

При вилученнях можна використати технологію «**збору сміття**»: у всіх записах (і підзаписах) має бути спеціальне поле, по якому можна визначити чи існує відповідний запис логічно (бо фізично він завжди присутній), наприклад 0 у цьому полі означає логічно відсутній, а 1 – присутній; тоді при вилученнях ми просто змінюємо 1 на 0 без фізичного вилучення, а при пошуку ігноруємо записи з нулями. Періодично проводиться реорганізація таких файлів з фізичним вилученням записів, які раніше були вилучені тільки логічно, з відповідною корекцією індексних таблиць і списків підлеглих записів. Для зменшення складності реорганізації, а також для підтримки якомога більшої компактності файлів до реорганізації часто використовують, так звані, *сміттєві зони* – це можуть бути окремі файли, чи спеціально зарезервоване місце на початку файлу – сюди заносяться номери логічно (але не фізично) вилучених записів, щоб мати можливість використати їх при виконанні функцій insert для нових записів.

Функції update використовуються для оновлення значень тільки для неключових полів записів; оновлення ключового поля здійснюється шляхом вилучення запису і внесення нового з новим значенням ключового поля.

Функція update-m - спочатку знаходимо постачальника (функція get-m), потім змінюємо значення у заданого поля і заносимо запис на теж саме місце.

Функція update-s - спочатку знаходимо поставку (функція get- s), потім змінюємо значення у заданого поля і заносимо запис на теж саме місце.

Функція insert-m – наповнюємо структуру, яка відповідає запису файлу S.fl, і записуємо в цей файл; необхідно зафіксувати адресу (чи номер) запису і внести її разом із значенням ключа до індексної таблиці; при необхідності індексну таблицю необхідно відсортувати за зростанням значення ключа і теж записати у свій файл.

Функція insert-s - наповнюємо структуру, яка відповідає запису файлу SP.fl; виконуємо get-m по ключу постачальника і заносимо значення поля посилання (з нього) на ланцюг підлеглих записів поставок в структуру запису поставок (тим самим ставимо її на початок ланцюга); заносимо запис поставок в файл SP.fl (в кінець) і запам’ятовуємо її адресу (чи номер), потім цю адресу переписуємо в прочитаний запис постачальника і заносимо цей запис до файлу S.fl.

Більш детально про роботу з нетекстовими (бінарними) файлами (мова С) можна почитати у відповідних розділах:

Вінник В.Ю. Алгоритмічні мови та основи програмування: Мова С — Житомир: ЖДТУ, 2007. — 328 с.

www.cyb.univ.kiev.ua/uk/library.school-guides.html