Задания к лабораторным работам

по курсу "Параллельные системы баз данных"

Введение

Цель и задачи практикума

Целью практикума курса "Параллельные системы баз данных" является *разработка прототипа параллельной СУБД*, реализующей фрагментарный параллелизм.

К *основным задачам* практикума относятся: практическое освоение техники проектирования и реализации параллельной СУБД, практическое освоение реализации обменов сообщениями по стандарту MPI.

План практикума

Практикум состоит из двух этапов:

1. *Изучение технического задания*. на разработку прототипа параллельной СУБД, которое приведено в *разделе 1* данного документа. Техническое задание содержит описания модельной базы данных, языка запросов и модульной структуры прототипа.
2. *Разработка* прототипа параллельной СУБД на основе указаний, приведенных в *разделе 2* данного документа. В разработке используется также *описание контрольных тестов* прототипа, вынесенное в *приложение* к данному документу.

Уровень подготовки

Успешное выполнение практикума возможно при наличии:

- знаний материала университетского курса по системам баз данных в объеме учебника [Ульман Дж., Уидом Д. Основы систем баз данных. - М.: ЛОРИ, 2000. - 374 с.] или [Дейт К.Дж. Введение в системы баз данных. - К.: Диалектика, 1998. - 784 с.];
- навыков программирования на языке Си.

Необходимое программное обеспечение

На практикуме используется следующее *программное обеспечение*:

- операционная система Microsoft Windows (версия 98 или выше);
- система программирования Microsoft Visual C++ (версия 6.0 или выше);
- свободно доступная библиотека MPICH (разработка Argonne National Laboratory), реализующая стандарт Message Passing Interface.

Поддерживающие материалы

Помимо текстов лекций и слайдов презентаций лекционной части курса, практикум использует следующие *поддерживающие материалы*:

- исходный *текст реализации головного модуля* прототипа параллельной СУБД;
- исходные *тексты тестовых программ* для автономного тестирования подсистем прототипа параллельной СУБД;
- *контрольные тесты*, т.е. модельная база данных и запросы для проверки прототипа.
- библиотека функций прототипа параллельной СУБД, т.е. объектный код для компоновки с разрабатываемым прототипом (далее *библиотека*);

- справочник по функциям библиотеки прототипа параллельной СУБД в формате HTML (далее *справочник*).

Методические рекомендации

В тексте используются следующие элементы структурирования материала: справочная информация, задания, указания и точки контроля.

Справочная информация не выделяется специальным заголовком, слушателю курса надлежит прочитать данную информацию и разобраться в ней. При наличии затруднений необходимо попытаться найти ответ, используя поддерживающие материалы курса, а если это не удалось – обратиться к преподавателю, ведущему занятия курса.

Задание выделяется соответствующим заголовком и содержит формулировку задачи, которую надлежит выполнить слушателю курса. Задания имеют сквозную нумерацию и предполагают последовательное выполнение. Задание может иметь указания по его выполнению.

Указания по выполнению задания выделяются соответствующим заголовком и представляют собой пошаговые инструкции для выполнения задания.

Точка контроля выделяется соответствующим заголовком и предполагает, что слушатель курса может продолжать работу только *после* самостоятельной проверки промежуточных результатов работы. Точки контроля имеют сквозную нумерацию.

1. Техническое задание на разработку прототипа параллельной СУБД

Данный раздел состоит из четырех частей. В *первой части* описана *модельная база данных* прототипа. Во *второй части* приведены ограничения, наложенные на приложение (исполняемый файл) прототипа. В *третьей части* приводится *описание языка запросов* прототипа параллельной СУБД. В *четвертой части* описана *модульная структура* прототипа.

1.1. Модельная база данных

Ниже приводятся *ограничения*, наложенные на модельную базу данных прототипа параллельной СУБД. Данные ограничения реализованы как *параметры компиляции* соответствующей подсистемы прототипа.

1. *База данных* состоит из фиксированного количества отношений. Каждое *отношение* идентифицируется порядковым номером, начиная с 0. Например, в базе данных, состоящей из четырех отношений, отношения имеют порядковые номера 0, 1, 2, 3 и *обозначаются* `R0`, `R1`, `R2`, `R3`. Результат выполнения запроса *не сохраняется* в модельной базе данных, а выдается на устройство `stdout`.
2. *Кортеж* каждого отношения базы данных состоит из одного и того же фиксированного количества атрибутов. Каждый *атрибут* идентифицируется порядковым номером, начиная с 0. Например, каждое отношение базы данных состоит из четырех атрибутов, имеющих порядковые номера 0, 1, 2, 3 и *обозначаемых* `A0`, `A1`, `A2`, `A3`.
3. Каждый атрибут определяется на *домене*, который соответствует типу данных `int` в языке программирования Си. Допускаются только неотрицательные значения атрибутов. Значение любого атрибута не превышает некоторой фиксированной константы (например, 100). Атрибут `A0` является *ключевым*, т.е. у кортежей одного от-

ношения его значения уникальны. Значения остальных атрибутов могут быть введены "вручную" или получены с помощью датчика случайных чисел.

4. Каждое отношение разбивается на одно и то же фиксированное количество горизонтальных *фрагментов*. Каждый фрагмент указанного отношения идентифицируется порядковым номером, начиная с 0. Фрагменты *обозначаются* `R0F0`, `R0F1`, …, `R1F0`, `R1F1`, … и т.д. Для каждого отношения фиксируется константа – количество кортежей в каждом фрагменте этого отношения. Например, фрагменты `R0F0`, `R0F1`, … содержат 5 кортежей, фрагменты `R1F0`, `R1F1`, … содержат 6 кортежей и т.д.
5. Физически каждый фрагмент отношения хранится в текстовом файле. Кортеж фрагмента представляет собой строку такого текстового файла, в которой целые неотрицательные числа разделены символами табуляции. Файл, в котором хранится фрагмент отношения, имеет имя
   `R`<номер_отношения>**`F`**<номер_фрагмента>**`.txt`**.
   Например, в случае, если отношения в базе данных состоят из трех фрагментов, то отношение `R0` хранится в файлах `R0F0.txt`, `R0F1.txt` и `R0F2.txt`, отношение `R1` хранится в файлах `R1F0.txt`, `R1F1.txt` и `R1F2.txt` и т.д.
6. Для всех отношений базы данных определяется *одна и та же* функция фрагментации. Для каждого отношения определяется *свой* атрибут фрагментации. *Функция фрагментации* по значению *атрибута фрагментации* заданного кортежа из заданного отношения возвращает номер фрагмента, в котором должен храниться данный кортеж. Например, пусть отношение состоит из трех фрагментов и значение его атрибута фрагментации изменяется в диапазоне от 0 до 99. Тогда функция фрагментации может быть организована таким образом, что кортежи со значением атрибута фрагментации от 0 до 32 включительно попадают в 0-й фрагмент, со значением от 33 до 65 включительно – в 1-й фрагмент, а остальные кортежи – во2-й фрагмент.
7. *Словарь данных* модельной базы данных не хранится в физическом файле. Реализация словаря данных встроена в программный код прототипа (в интерфейсе прототипа имеются соответствующие типы данных и функции доступа к словарю).

Задание 1. Изучение структуры модельной базы данных

Изучите в справочнике по функциям библиотеки прототипа разделы *Генератор базы данных* и *Параметры компиляции базы данных*.

1.2. Прототип параллельной СУБД

Прототип параллельной СУБД представляет собой приложение ОС Windows, на которое наложены следующие ограничения:

1. Прототип работает на *однопроцессорной машине* под управлением ОС Windows, не используя реальные узлы вычислительного кластера. Работа экземпляра СУБД как процесса на отдельном узле вычислительного кластера *имитируется* с помощью *процесса Windows*.
2. Обмен сообщениями между процессами Windows реализуется на основе *стандарта MPI* с помощью *библиотеки MPICH*. Количество процессов Windows фиксируется и совпадает с количеством фрагментов в каждом отношении базы данных.
3. Каждый процесс выполняет одно и то же *консольное Windows-приложение*, обрабатывающее один и тот же запрос к модельной базе данных. Данное приложение рассчитано на *однократную обработку* одного запроса к модельной базе данных (прототип является "однозарядным").

1.3. Язык запросов

Задание 2. Изучение языка запросов прототипа

1. Изучите описание языка запросов прототипа параллельной СУБД в разделе справочника *Компилятор запросов*.
2. В Табл. 1 приведены выражения реляционной алгебры. Напишите соответствующие запросы на языке запросов прототипа.

Табл. 1. Запросы контрольных тестов (выражения реляционной алгебры)

№ п/п	Запрос
1.	$\sigma_{A2=43}(R0)$
2.	$(\sigma_{A2=80}(R0)) \triangleright\triangleleft_{A1} (\sigma_{A2=21}(R1))$
3.	$(\sigma_{A3=43}(R2)) \triangleright\triangleleft_{A1} (\sigma_{A2=80}(R0))$
4.	$(\sigma_{A2=80}(R0)) \triangleright\triangleleft_{A1} (\sigma_{A3=43}(R2))$

Точка контроля 1

Сравните полученные результаты с ответами, приведенными в приложении.

1.4. Модульная структура прототипа параллельной СУБД

Прототип параллельной СУБД имеет *модульную структуру*, приведенную на Рис. 1.

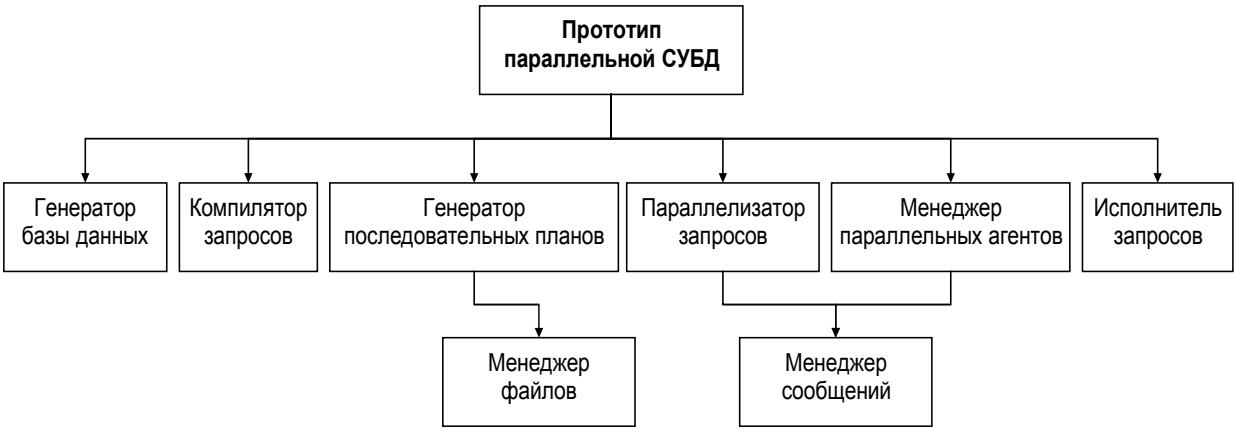

Рис. 1. Модульная структура прототипа параллельной СУБД

Генератор базы данных выполняет создание отношений и словаря данных модельной распределенной базы данных.

Компилятор запросов формирует внутреннее представление запроса к базе данных в виде дерева. Узлами дерева являются операции расширенной реляционной алгебры (соединение, выборка, сканирование отношения и др.).

Генератор последовательных планов назначает низкоуровневые процедуры реализации операций расширенной реляционной алгебры во внутреннем представлении запроса.

Параллелизатор запросов преобразует последовательный план выполнения запроса в параллельный. В частности, параллелизатор запросов добавляет в последовательный план операторы обмена `exchange`.

Менеджер параллельных агентов по параллельному плану выполнения запроса формирует параллельных агентов для узлов вычислительного кластера.

Исполнитель запросов интерпретирует (исполняет) параллельного агента на заданном узле вычислительного кластера.

Менеджер файлов реализует низкоуровневые операции с отношениями базы данных: открыть, закрыть, сканировать отношение, выдать очередной кортеж отношения и др.

Менеджер сообщений реализует обмен сообщениями (кортежами) между узлами вычислительного кластера на основе интерфейса MPI.
исправления.

Задание 3. Сборка и запуск прототипа

1. Создайте в среде MS Visual C++ новый проект `pdbms` для разработки прототипа параллельной СУБД. Настройте проект `pdbms`, обеспечив возможность компоновки библиотеки прототипа и библиотек пакета MPICH.
2. Изучите исходные тексты головного модуля прототипа (используйте также справочник).
3. Добавьте исходные тексты головного модуля в созданный проект. Выполните компиляцию и сборку прототипа.
4. Выполните запуск прототипа *на запросах 1, 2, 3* контрольных тестов (используйте архив базы данных и запросов контрольных тестов в каталоге поддерживающих материалов курса). Убедитесь, что полученные результаты выполнения запросов совпадают с указанными в приложении.

2. Разработка прототипа параллельной СУБД

В ходе разработки прототипа предполагается реализация функций, составляющих библиотеку прототипа. Выполняя компиляцию и сборку прототипа, следите за тем, что при этом используются исходные тексты Вашей реализации, а не объектный код библиотеки.

Задание 4. Реализация Исполнителя запросов

Изучите в справочнике по функциям библиотеки прототипа раздел *Исполнитель запросов*. Выполните реализацию данной подсистемы. указания к заданию

Указания к заданию 4

1. Разработайте алгоритм и выполните реализацию функции `en_ExecuteQuery` ("Выполнить запрос").
2. Выполните компиляцию и сборку прототипа.

Точка контроля 2

Выполните запуск прототипа *на запросах 1, 2, 3* контрольных тестов. Убедитесь, что полученные результаты совпадают с указанными в приложении. В противном случае выполняйте приведенные выше указания данного задания до устранения ошибок.

Задание 5. Реализация Менеджера параллельных агентов

Изучите в справочнике по функциям библиотеки прототипа раздел *Менеджер параллельных агентов*. Выполните реализацию данной подсистемы.

Указания к заданию 5

1. Разработайте алгоритмы и выполните реализацию функций `ResetSTORE` ("Начать операцию сохранения кортежей") и `ExecuteSTORE` ("Выполнить операцию сохранения кортежей").

2. Разработайте алгоритм и выполните реализацию функции `am_MakeAgent` ("Создать параллельного агента").
3. Выполните автономное тестирование Менеджера параллельных агентов, используя соответствующую тестовую программу; при наличии ошибок выполняйте отладку и тестирование данной подсистемы до их устранения.
4. Выполните компиляцию и сборку прототипа.

Точка контроля 3

Выполните запуск прототипа *на запросах 1, 2, 3* контрольных тестов. Убедитесь, что полученные результаты совпадают с указанными в приложении. В противном случае выполняйте приведенные выше указания данного задания до устранения ошибок.

Задание 6. Реализация Параллелизатора запросов

Изучите в справочнике по функциям библиотеки прототипа раздел *Параллелизатор запросов*. Выполните реализацию данной подсистемы.

Указания к заданию 1

1. Разработайте алгоритм реализации функции `pl_ParallelizeQuery` ("Построить дерево параллельного плана исполнения запроса").
   Рекомендуем выполнить "сухую" отладку данного алгоритма (без сборки и запуска прототипа) на запросах контрольных тестов.
2. Выполните реализацию функции `pl_ParallelizeQuery`.
3. Выполните автономное тестирование Параллелизатора запросов, используя соответствующую тестовую программу; при наличии ошибок выполняйте отладку и тестирование данной подсистемы до их устранения.
4. Выполните компиляцию и сборку прототипа.

Точка контроля 4

Выполните запуск прототипа *на запросах 1, 2, 3* контрольных тестов. Убедитесь, что полученные результаты совпадают с указанными в приложении. В противном случае выполняйте приведенные выше указания данного задания до устранения ошибок.

5. Разработайте алгоритмы реализации следующих функций:
   - `ResetMERGE` ("Начать операцию MERGE"), `ExecuteMERGE` ("Выполнить операцию MERGE");
   - `ResetGATHER` ("Начать операцию GATHER"), `ExecuteGATHER` ("Выполнить операцию GATHER");
   - `ResetSPLIT` ("Начать операцию SPLIT"), `ExecuteSPLIT` ("Выполнить операцию SPLIT");
   - `ResetSCATTER` ("Начать операцию SCATTER"), `ExecuteSCATTER` ("Выполнить операцию SCATTER").

   Рекомендуем выполнить "сухую" отладку данных алгоритмов (без сборки и запуска прототипа) по крайней мере на одном запросе контрольных тестов.
6. Выполните реализацию функций `ResetMERGE`, `ExecuteMERGE`, `ResetGATHER`, `ExecuteGATHER`, `ResetSPLIT`, `ExecuteSPLIT`, `ResetSCATTER`, `ExecuteSCATTER`.
7. Выполните компиляцию и сборку прототипа.

Точка контроля 5

Выполните запуск прототипа *на запросах 1, 2, 3* контрольных тестов. Убедитесь, что полученные результаты совпадают с указанными в приложении. В противном случае выполняйте приведенные выше указания данного задания до устранения ошибок.

Задание 7. Реализация Генератора последовательных планов

Изучите в справочнике по функциям библиотеки прототипа раздел *Генератор последовательных планов*. Выполните реализацию данной подсистемы.

Указания к заданию 7

1. Разработайте алгоритмы и выполните реализацию функций `ResetSCAN` ("Начать операцию сканирования отношения") и `ExecuteSCAN` ("Выполнить операцию сканирования отношения").
2. Разработайте алгоритмы и выполните реализацию функций `ResetRESTRICT` ("Начать операцию выборки") и `ExecuteRESTRICT` ("Выполнить операцию выборки").
3. Выполните компиляцию и сборку прототипа.

Точка контроля 6

Выполните запуск прототипа *на запросах 1, 2, 3* контрольных тестов. Убедитесь, что полученные результаты совпадают с указанными в приложении. В противном случае выполняйте приведенные выше указания данного задания до устранения ошибок.

4. Разработайте алгоритм функции `ExecuteJOIN_NestedLoops` ("Выполнить операцию соединения методом вложенных циклов").
   Рекомендуем выполнить "сухую" отладку данных алгоритмов (без сборки и запуска прототипа).
5. Выполните реализацию функции `ExecuteJOIN_NestedLoops`.
6. Выполните компиляцию и сборку прототипа.

Точка контроля 7

Выполните запуск прототипа *на запросах 1, 2, 3 и 4* контрольных тестов. Убедитесь, что результаты выполнения *запросов 1, 2* и *либо 3, либо 4* совпадают с указанными в приложении (в зависимости от того, какое из соединяемых отношений находится во внутреннем цикле). В противном случае выполняйте приведенные выше указания данного задания до устранения ошибок.

Задание 8. Реализация Менеджера сообщений

Изучите в справочнике по функциям библиотеки прототипа раздел *Менеджер сообщений*. Выполните реализацию данной подсистемы.

Указания к заданию 8

1. Определите функции стандарта MPI, на основе которых будет реализован Менеджер сообщений. Разработайте алгоритмы и выполните реализацию функций Менеджера сообщений.
2. Выполните автономное тестирование Менеджера сообщений, используя соответствующую тестовую программу; при наличии ошибок выполняйте отладку и тестирование данной подсистемы до их устранения.
3. Выполните компиляцию и сборку прототипа.

Точка контроля 8

Выполните запуск прототипа *на запросах 1, 2, 3* контрольных тестов. Убедитесь, что полученные результаты совпадают с указанными в приложении. В противном случае выполняйте приведенные выше указания данного задания до устранения ошибок.

ПРИЛОЖЕНИЕ. Описание контрольных тестов

Отношения модельной базы данных

Замечания.

1. Атрибуты фрагментации отношений выделены штриховкой.
2. Все отношения фрагментированы с использованием функции фрагментации `db_FragmentByDefault` (см. справочник по функциям прототипа).

Отношение R0

Фрагменты	Атрибуты			
	A0	**A1**	**A2**	**A3**
F0	0	*0*	43	67
	1	*0*	21	22
	2	*0*	80	71
	3	*0*	43	45
	4	*0*	80	58
F1	5	*1*	43	71
	6	*1*	6	23
	7	*1*	89	58
	8	*1*	80	24
	9	***1***	**20**	**77**
F2	11	*2*	80	67
	12	*2*	33	58
	13	*2*	44	43
	14	*2*	43	77
	15	*2*	99	55

Отношение R1

Фрагменты	Атрибуты			
	A0	**A1**	**A2**	**A3**
F0	100	*0*	21	22
	101	*0*	45	33
	102	*0*	70	44
	103	*0*	34	55
	104	*0*	80	66
F1	106	*1*	80	58
	107	*1*	21	28
	108	*1*	40	67
	109	*1*	63	39
	110	*1*	74	55
F2	112	*2*	99	6
	113	*2*	21	3
	114	*2*	12	1
	115	*2*	41	77
	116	*2*	21	23

Отношение R2

Фрагменты	Атрибуты			
	A0	A1	A2	A3
F0	200	0	*0*	11
	201	2	*0*	43
	202	80	*0*	9
	203	11	*0*	89
	204	1	*0*	43
F1	207	34	*1*	43
	208	0	*1*	15
	209	45	*1*	43
	210	43	*1*	4
	211	2	*1*	32
F2	214	80	*2*	43
	215	55	*2*	67
	216	0	*2*	77
	217	1	*2*	43
	218	87	*2*	0

Запросы контрольных тестов

№ п/п	Запрос	
	Выражение реляционной алгебры	Язык запросов прототипа
1.	$\sigma_{A2=43}(R0)$	`0 R 2 = 43 #0`

2.	$(\sigma_{A2=80}(R0)) \triangleright\triangleleft_{A1} (\sigma_{A2=21}(R1))$	`0 J 1 1 2`
		`1 R 2 = 80 #0`
		`2 R 2 = 21 #1`

3.	$(\sigma_{A3=43}(R2)) \triangleright\triangleleft_{A1} (\sigma_{A2=80}(R0))$	`0 J 1 1 2`
		`1 R 3 = 43 #2`
		`2 R 2 = 80 #0`

4.	$(\sigma_{A2=80}(R0)) \triangleright\triangleleft_{A1} (\sigma_{A3=43}(R2))$	`0 J 1 1 2`
		`1 R 2 = 80 #0`
		`2 R 3 = 43 #2`

Результаты выполнения запросов

№ запроса п/п	Результирующее отношение				
1.	№ кортежа п/п	Атрибуты			
		A0	A1	A2	A3
	1	0	0	43	67
	2	3	0	43	45
	3	5	1	43	71
	4	14	2	43	77

№ запроса п/п	Результирующее отношение							
2.	№ кортежа п/п	Атрибуты						
		A0	A1	A2	A3	A4	A5	A6
	1	2	0	80	71	100	21	22
	2	4	0	80	58	100	21	22
	3	8	1	80	24	107	21	28
	4	11	2	80	67	113	21	3
	5	11	2	80	67	116	21	23
3.	№ кортежа п/п	Атрибуты						
		A0	A1	A2	A3	A4	A5	A6
	1	201	2	0	43	11	80	67
	2	204	1	0	43	8	80	24
	3	213	1	1	43	8	80	24
	4	217	1	2	43	8	80	24
	5	220	0	2	43	2	80	71
	6	220	0	2	43	4	80	58
4.	№ кортежа п/п	Атрибуты						
		A0	A1	A2	A3	A4	A5	A6
	1	2	0	80	71	220	2	43
	2	4	0	80	58	220	2	43
	3	8	1	80	24	204	0	43
	4	8	1	80	24	213	1	43
	5	8	1	80	24	217	2	43
	6	11	2	80	67	201	0	43