Содержание

Введение 3

1 Использование БД в прикладном программировании 4

1.1 Области использования прикладного программирования 4

1.2 Локальные и удаленные БД. Архитектура клиент-сервер 4

2 Использование SQL в прикладном программировании 5

2.1 Назначение SQL 5

2.2 Статический и динамический SQL 5

2.3 Влияние логики прикладного приложения на эффективность SQL 5

2.4 Средства разрабоки приложений 5

3 Пример использования SQL в прикладном программировании – программа PrintMonitorLite 7

Заключение 10

Список использованных источников 11

1 Использование БД в прикладном программировании

1.1 Области использования прикладного программирования

1.2 Локальные и удаленные БД. Архитектура клиент-сервер

функциональные прикладные программы, ориентированные в первую очередь на использование базы данных и активную работу с ней. Эта среда разработки имеет развитые механизмы по управлению данными, скрывающие детали доступа к данным, но также содержит возможность написания собственных запросов SQL.

Все запросы к БД хранятся в семействе QueryDefs. Имеется возможность динамического создания запросов, возможность модификации существующих или создания временных запросов, удаляемых автоматически сразу после закрытия объекта QueryDef. Использование параметров запроса не поддерживается; также нет возможности предварительной компиляции запроса с созданием плана выполнения. Использование страничной блокировки записей накладывает значительные ограничения на масштабируемость разработки и не позволяет организовывать эффективную параллельную обработку данных.

В целом средство разработки Microsoft Access подходит для создания простых прикладных программ с несложным алгоритмом обработки данных через SQL-запросы, либо как справочная БД с возможностью выборки данных через SQL-операторы по различным критериям.

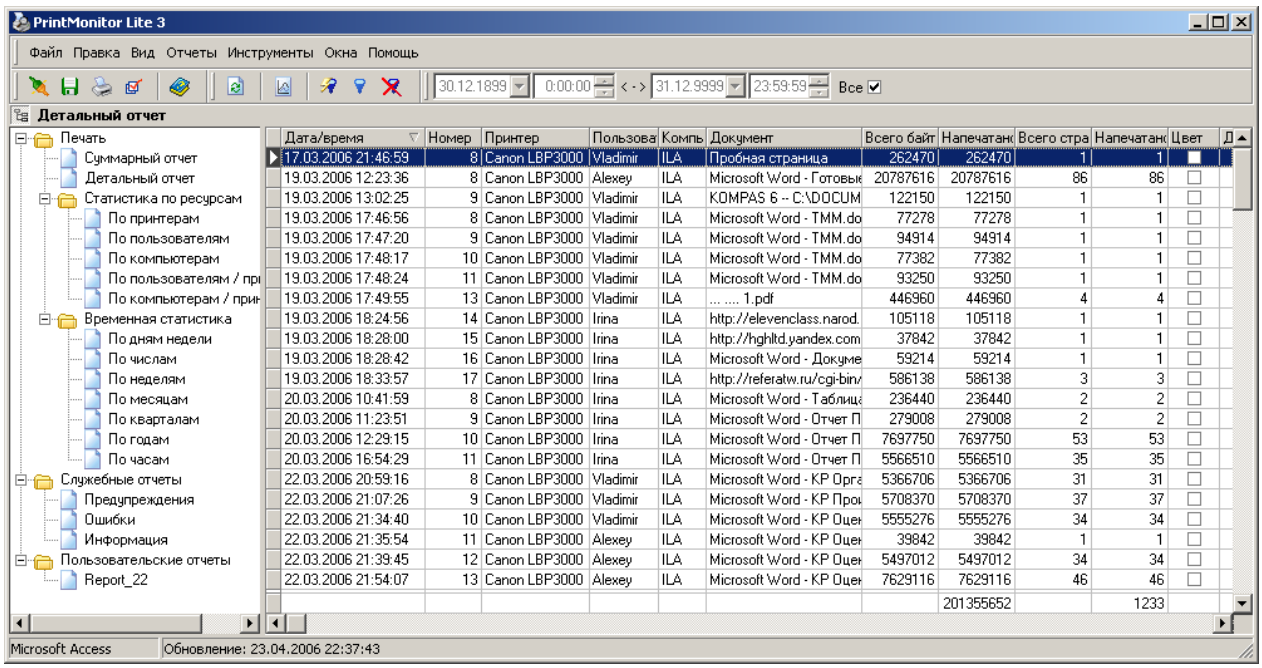

Рисунок 3.1 – Отчет об использовании принтера

Очевидно, что для домашнего компьютера с одним принтером большинство из формируемых запросом полей можно опустить. Тогда, изменяя SQL-оператор, получим совершенно новый отчет, который не содержит избыточной информации:

```
SELECT [_pm_jobs].job_date, [_pm_users].name as [user],
       iif([_pm_jobs].job_name, [_pm_jobs].ppages
FROM [_pm_users]
INNER JOIN ([_pm_printers]
INNER JOIN [_pm_jobs]
ON [_pm_printers].id = [_pm_jobs].idPrinter)
ON [_pm_users].id = [_pm_jobs].idUser
WHERE ([_pm_jobs].IsDelete = 0) AND ([_pm_jobs].dt >= %TIME-START%) AND
([_pm_jobs].dt <= %TIME-STOP%);
```

Отчет, формируемый этим запросом, приведен на рисунке 3.2.

Таким образом, использование SQL для прикладного программирования позволяет конечным пользователям решать задачи, не предусмотренные программистом на этапе разработки прикладного приложения без дополнительных затрат со стороны разрабочика.

Заключение