БАЗЫ ДАННЫХ И
ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Ст.преп. каф. ИТ Кеберле Наталья Геннадьевна

Лекция 17. Бизнес-логика. Триггеры

На этой лекции

Мы узнаем,

что такое «архитектура клиент-сервер»,

что такое «бизнес-логика»,

разберемся с вариантами реализации бизнес-вычислений: триггер, хранимая процедура, функция

узнаем о модели ECA для триггеров,

т.е. как СУБД обходится без пользователя при выполнении рутинных операций

Архитектура «клиент-сервер»

- Компьютеры и программы, входящие в состав информационной системы, **не являются равноправными**

- Некоторые из них владеют ресурсами:
  - файловая система,
  - процессор,
  - принтер,
  - база данных,
- Другие имеют возможность обращаться к этим ресурсам

Архитектура «клиент-сервер»

- Компьютер (или программу), управляющий ресурсом, называют **сервером** этого ресурса
  - файл-сервер,
  - сервер базы данных,
  - вычислительный сервер

Клиент и сервер какого-либо ресурса могут находиться либо

в рамках одной вычислительной системы,

либо

на различных компьютерах, связанных сетью.

Принцип архитектуры «клиент-сервер»

Основной принцип технологии "клиент-сервер" заключается в разделении функций приложения на **три** группы:

1. **ввод и отображение данных** (взаимодействие с пользователем);

2. **прикладные функции, характерные для данной предметной области;**

3. **функции управления ресурсами** (файловой системой, базой данных и т.д.)

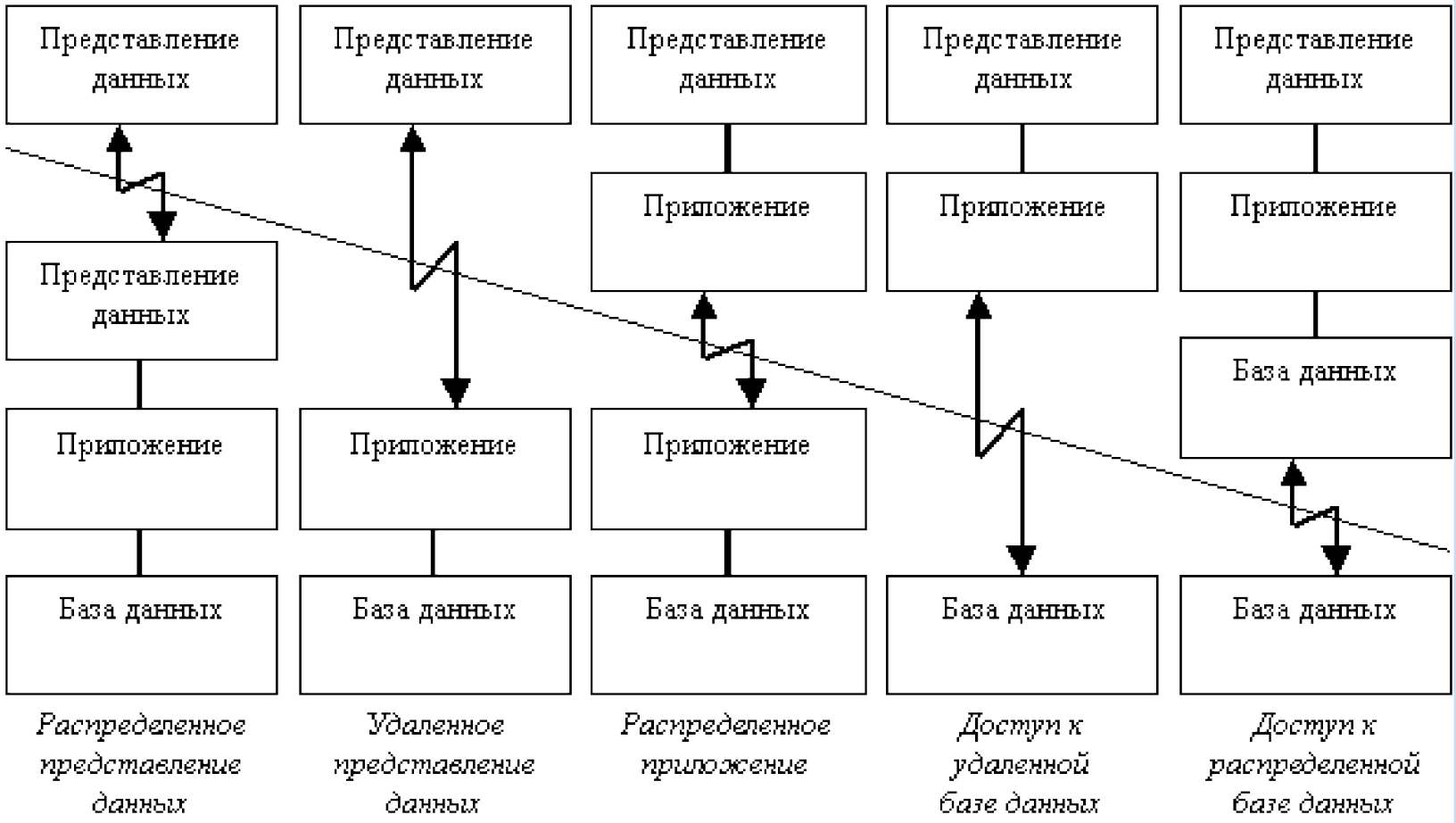
Принцип архитектуры «клиент-сервер»

В любом приложении выделяются следующие компоненты:

- компонент представления данных
- прикладной компонент
- компонент управления ресурсом

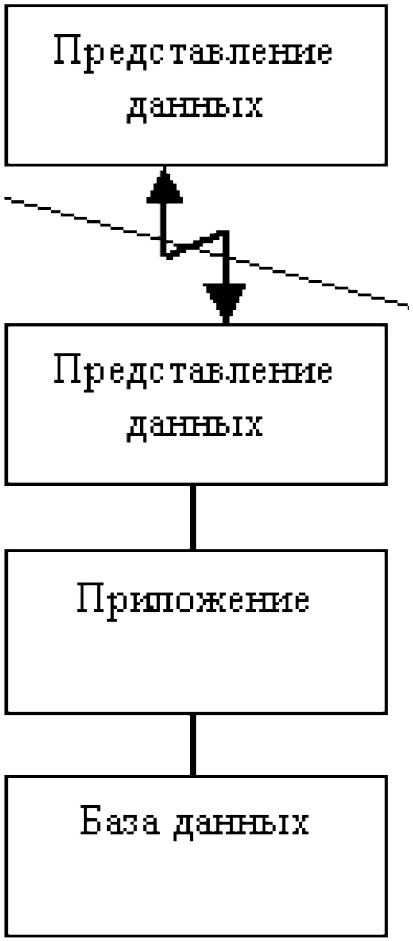
Связь между компонентами осуществляется по определенным правилам, которые называют "**протокол взаимодействия**".

Модели взаимодействия «клиент-сервер»



Источник рисунка: http://www.mstu.edu.ru/study/materials/zelenkov/ch_7_1.html

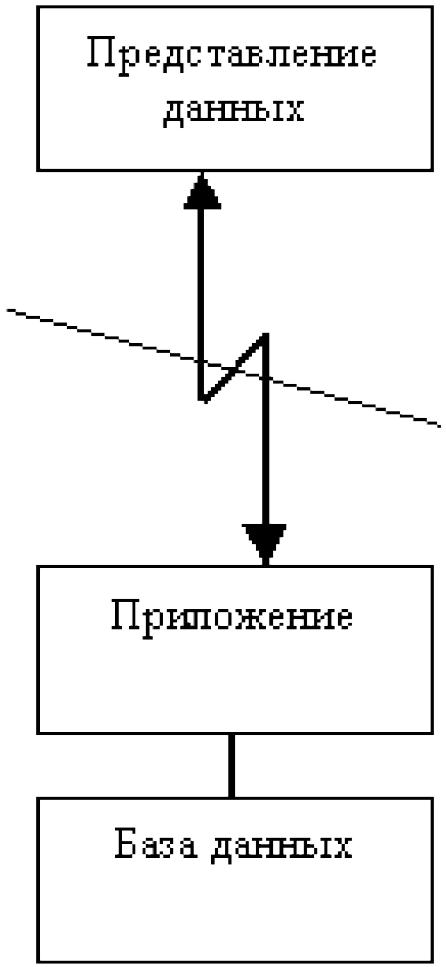
Распределенное представление данных



Распределенное представление данных

- Исторически первая модель
- Реализовывалась на универсальной ЭВМ (**mainframe**) с подключенными к ней неинтеллектуальными терминалами.
- Управление данными и взаимодействие с пользователем - в одной программе,
- на терминал передавалась только "картинка", сформированная на центральном компьютере

Удалённое представление данных



Удаленное представление данных

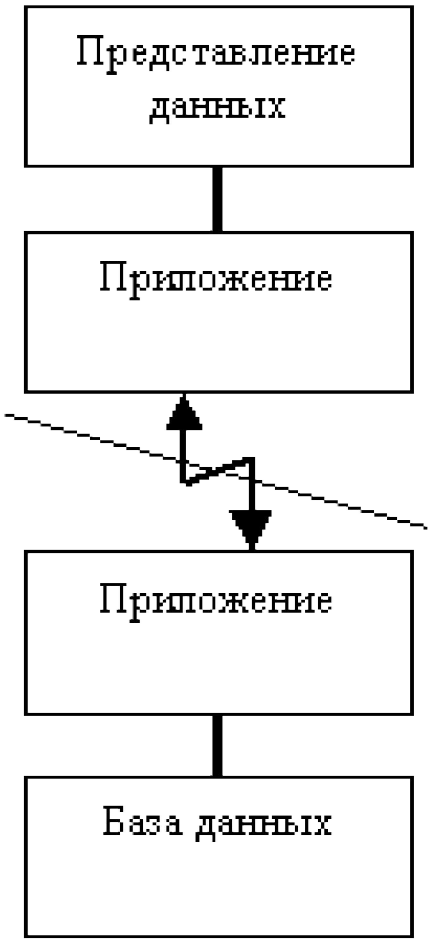
- Обусловлено появлением ПК и локальных сетей.
- Некоторое время базовой для сетей ПК была архитектур**а файлового сервера:**

один из компьютеров является файловым сервером,

на клиентах выполняются приложения, в которых совмещены компонент представления и прикладной компонент (СУБД и прикладная программа).

- Протокол обмена при этом представляет набор низкоуровненых вызовов операций файловой системы.
- Архитектура реализуется с помощью персональных СУБД (Access, dBase, FoxPro, Clipper)
- Имеет недостатки - высокий сетевой трафик и отсутствие унифицированного доступа к ресурсам

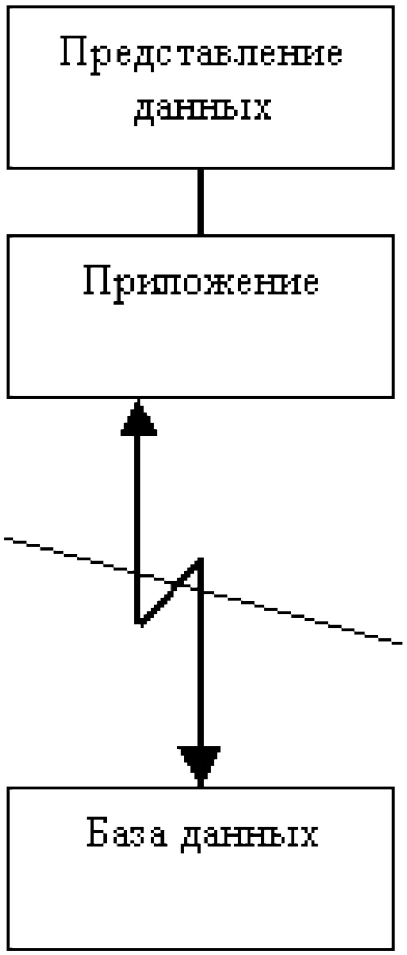
Распределенное приложение



Распределенное приложение

- Требуются специализированные сервера баз данных
- Ядро СУБД функционирует на сервере, протокол обмена обеспечивается с помощью языка SQL.
- Такой подход по сравнению с файловым сервером ведет к уменьшению загрузки сети и унификации интерфейса "клиент-сервер".
- Сетевой трафик остается достаточно высоким
- невозможно хорошее администрирование приложений, поскольку в одной программе совмещаются различные функции.

Доступ к удалённой базе данных



Доступ к удаленной базе данных

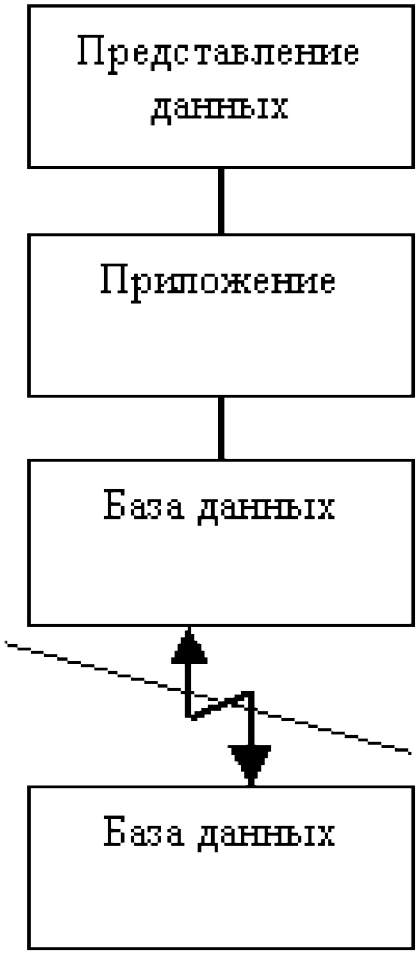
- Используется концепция активного сервера, который использует **механизм хранимых процедур**.
- Часть прикладного компонента переносится на сервер
- Процедуры хранятся в словаре базы данных, разделяются между несколькими клиентами и выполняются на том же компьютере, что и SQL-сервер.
- Плюсы:

  \+ возможно централизованное администрирование прикладных функций,

  \+ значительно снижается сетевой трафик (т.к. передаются не SQL-запросы, а вызовы хранимых процедур).
- Минусы:

  \- ограниченность средств разработки хранимых процедур по сравнению с языками общего назначения (C и Pascal).

Доступ к распределенной базе данных



Доступ к распределенной базе данных

- На практике сейчас обычно используются смешанный подход:

  - простейшие прикладные функции выполняются хранимыми процедурами на сервере

  - более сложные реализуются на клиенте непосредственно в прикладной программе

Концепция “активный сервер”

- Активные СУБД объединяют технологии баз данных с программированием «на логических правилах» для придания базам данных возможности ***реагирования* на** некоторые (возможно, внешние) стимулы, называемые ***событиями***.

Активность - это способность БД выполнять некоторые действия над данными опосредованно, без явного вмешательства пользователя.

Активная база данных

Активная база данных состоит из

(пассивной) базы данных

и

множества активных правил (active rules)

[Widom , Ceri, 1995]

Наиболее популярной формой активных правил являются так называемые **ECA-правила**

ECA-rules,

“Event-Condition-Action”,

правила “Событие Условие Действие”

[Chakravarthy et al., 1989]

Правила Event-Condition-Action

- Каждое ECA-правило описывает

  действие (**action**), которое должно быть выполнено при возникновении одного или более событий (**events**),

  при условии выполнения некоторого требования или группы требований (**conditions**)

- Когда нужное событие происходит, говорят, что *правило срабатывает*, после срабатывания правило рассматривается на предмет выполнения необходимых требований, и если эти требования выполняются, то правило исполняет предписанные ему действия.

Средства реализации активных правил

- Чаще всего они реализуются двумя путями:

  с помощью **триггеров** и **хранимых процедур**.

- Примерный список СУБД, поддерживающих активные правила :

  Oracle, Postgres, Starburst, Interbase,

  Informix, Ingres, Sybase, MS SQL Server, MySQL, Allbase,

  а также объектно-ориентированные СУБД, придерживающиеся стандарта SQL3.

Отличия в системе реализации ECA

- (Event-Condition Time Coupling) проверка требований, связанных с правилом – сразу после срабатывания правила (immediate) или в отложенном режиме (delayed);
- (Condition–Action Time Coupling) исполнение правила сразу после проверки условий или в отложенном режиме;
- (Action Execution) атомарное («все или ничего») исполнение правила или с возможностью прерывания исполнения правила (interruptable);
- (Event/Action Link) исполнения правила после (after), перед (before) или вместо (instead of) события, вызвавшего срабатывание триггера.

[Fraternali, Tanca, 1995]

Триггеры

Триггер – это специальный тип хранимых процедур, запускаемых сервером **автоматически** при выполнении некоторых действий с данными таблицы

- Каждый триггер привязывается к конкретной таблице
- Когда пользователь пытается изменить данные в таблице, сервер автоматически запускает триггер и, только если он завершается успешно, разрешается выполнение изменений
- Все производимые триггером модификации данных рассматриваются как **одна транзакция**
- В случае обнаружения ошибки или нарушении целостности данных происходит **откат этой транзакции**

Применение триггеров

- Сложная проверка целостности данных (не хватает штатных средств СУБД)
- Каскадное обновление/удаление связанных данных
- Автоматическое архивирование некоторых действий над данными

Виды триггеров по типу вызова

Триггеры различаются по типу команд, на которые они реагируют. Существуют три типа триггеров.

- **INSERT TRIGGER**. Триггеры этого типа запускаются при попытке вставки данных с помощью команды INSERT;
- **UPDATE TRIGGER**. Триггеры этого типа запускаются при попытке изменения данных с помощью команды UPDATE;
- **DELETE TRIGGER**. Триггеры этого типа запускаются при попытке удаления данных с помощью команды DELETE.

Виды триггеров по времени выполнения

- **AFTER**. Триггер выполняется после успешного выполнения команды, изменяющей данные в таблице. Если же команда по каким-либо причинам не может быть успешно завершена, то триггер также не выполняется.
- **INSTEAD OF**. В этом случае триггер вызывается вместо команд, назначенных для запуска триггера (в MySQL – нет, в MS SQL Server - есть).
- **BEFORE**. Триггер выполняется перед фактическим выполнением команды, вызвавшей его (в MySQL – есть).

Создание триггера

```
CREATE TRIGGER триггер
{BEFORE | AFTER} {DELETE | INSERT | UPDATE}
ON { таблица } FOR EACH ROW
    действие [. . . n]
```

Как триггер отслеживает изменения

- Для каждой таблицы, к которой приписан триггер, создаются две временные таблицы, OLD и NEW (MySQL), deleted и inserted (MS SQL Server).
- Эти таблицы содержат списки записей, которые были /будут добавлены/удалены/изменены операцией с базой данных.
- Структура таблиц такая же, как и у исходной таблицы

Как триггер отслеживает изменения(2)

- При выполнении команды INSERT INTO... новые строки будут в таблице NEW
- При выполнении команды DELETE FROM... удаленные строки будут в таблице OLD
- При выполнении команды UPDATE... Старые значения строк будут в таблице OLD, новые – в NEW.

Пример триггера

рейсы (	рейс	пункт_ отправл	пункт_ прибытия	время_ вылета	время_ прибыт	)
	83	Нью-Йорк	Чикаго	10:15	12:28	
	84	Чикаго	Нью-Йорк	14:15	16:30	
	109	Нью-Йорк	Лос-Анджелес	21:50	2:52	
	213	Нью-Йорк	Бостон	11:43	12:45	
	214	Бостон	Нью-Йорк	14:20	15:12	
—	117	Атланта	Бостон	06:30	09:35	

```
DELETE FROM рейсы
WHERE рейс=117;
```

Предположим, нам нужно сохранять данные об удаляемых рейсах в архиве

Пример триггера (архивирование)

рейсы Архив (	код_записи	рейс	пункт_ отправл	пункт_ прибытия	…	дата_ удален	)
	1	117	Атланта	Бостон	…	01.03.13	

```
DELETE FROM рейсы
WHERE рейс=117;
```

Создадим вначале такую архивную таблицу:

```
CREATE TABLE рейсыАрхив (
код_записи NOT NULL PRIMARY KEY,
рейс int,
...
дата_удален timestamp);
```

Пример триггера (архивирование)

```
CREATE TRIGGER тр_удал_рейс_архив
AFTER DELETE
ON рейсы FOR EACH ROW
BEGIN
    DECLARE delete_date timestamp;
    DECLARE record_id int;
    SET delete_date = NOW();
    SET record_id = (SELECT MAX(код_записи) + 1
            FROM рейсы_архив );
    INSERT INTO рейсы_архив (код_записи, рейс,
    пункт_отправ, ..., дата_удален)
    SELECT record_id, OLD.рейс,
    OLD.пункт_отправ, ..., delete_date;
END//
```

Итого:

- Триггеры – способ задать работу сервера **в ответ на** некоторые действия пользователя
- В СУБД должен быть язык программирования с использованием команд SQL
- Триггер выполняется слитно с транзакцией, вызвавшей действия триггер
- Триггер использует специальные таблицы OLD и NEW