

Обработка и исполнение запросов в СУБД (Лекция 10)

Объектные и Объектно-Реляционные СУБД

v2

Георгий Чернышев

Академический Университет

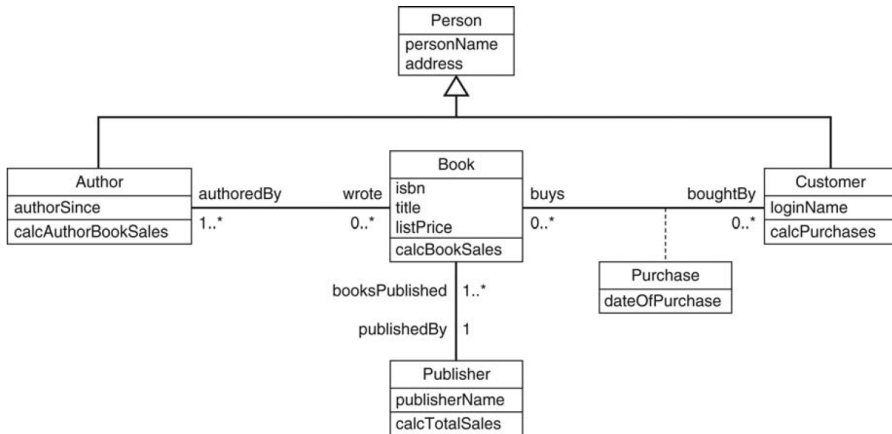
chernishev@gmail.com

30 ноября 2016 г.

- Реляционная модель не всегда подходит (в нишевых системах — CAD, OIS, ...) [Özsu and Valduriez, 2011]:
 - ❶ Иногда нужна возможность задавать пользовательские типы данных (application-specific data types);
 - ❷ Пологая структура иногда плоха, иногда нужно отражать отношение включения что сложно сделать в реляционной модели;
 - ❸ Проблема несоответствия импеданса (impedance mismatch);
- Object-Oriented Database System (OODB):
 - ❶ Object-Oriented Database System Manifesto, середина 1980-х;
 - ❷ Классы, иерархии, OID-ы;
 - ❸ OOP- — единый язык для доступа в базу и разработки приложения, импеданса нет;
- Object-Relational Database (ORDB):
 - ❶ Third generation database system manifesto, 1990-ый;
 - ❷ Первая ORDB — Postgres [Rowe and Stonebraker, 1987];
 - ❸ UDT, таблицы из объектов, ID у записей;

- Сущности представляются объектами, имеют состояние и поведение;
- Состояние — набор структурных свойств объекта;
- Поведение описывается через метды (сигнатуру);
- Объекты с одинаковым интерфейсом объединяются в класс;
- Каждый объект класса получает уникальный внутренний идентификатор — OID, он иммутабелен;
- Связи между объектами реализуются с помощью ссылок;
- Классы можно объединять в иерархии, наследование;

Пример: объектная модель данных, UML



Object Data Models. Figure 1. The publisher object data model.

1

¹Изображение взято из [Urban and Dietrich, 2009]

Объектно-Ориентированная модель

[Urban and Dietrich, 2009]

- Суть OODB: база данных в которой объект это основная единица работы, плюс есть некоторый ОО язык для запросов;
- Object Data Standard — содержит описание модели, язык задания схемы (ODL), язык запросов (OQL), разработан ODMG [Özsu and Valduriez, 2011];
- ODL (Object Definition Language) — язык спецификации объектной схемы: задание свойств объектов и сигнатур методов;
- Свойство это атрибут или отношение между объектами;
- В ODL отношения одно или двунаправленные (тогда СУБД поддерживает целостность);

Пример: OO модель данных, OODB Schema, ODL

<pre>class Person (extent people) {attribute string personName, attribute AddressType address }; class Author extends Person (extent authors) {attribute Date authorSince, relationship set<Book> wrote inverse Book::authoredBy, public float calcAuthorBookSales(); }; class Customer extends Person (extent customers) {attribute string loginName, relationship set<Purchase> buys inverse Purchase::purchasedBy, public float calcPurchases(); }; class Purchase (extent purchases) {attribute Date dateOfPurchase, relationship Book bookPurchased inverse Book::boughtBy, relationship Customer purchasedBy inverse Customer::buys; };</pre>	<pre>class Book (extent books, key isbn) {attribute string isbn, attribute string title, attribute float listPrice, relationship set<Author> authoredBy inverse Author::wrote, relationship set<Purchase> boughtBy inverse Purchase::bookPurchased, relationship Publisher publishedBy inverse Publisher::booksPublished, public float calcBookSales(); }; class Publisher (extent publishers) {attribute string publisherName, relationship set<Book> booksPublished inverse Book::publishedBy, public float calcTotalSales(); };</pre>
--	---

2

²Изображение взято из [Urban and Dietrich, 2009]

ОО модель [Urban and Dietrich, 2009]

Замечания по языкам:

- Можно задавать объектную схему на некотором OOP-языке, поддерживаемом в СУБД: C++, Java, ...
- Описание схемы и реализация методов на некотором языке называется language binding;
- Стандарт ODL говорит и о Object Query Language (OQL), подобие SQL, но можно “ходить” по “.” и вызывать методы:

```
1      select b.publishedBy.publisherName
2      from books b
3      where b.isbn = '0-13-042898-1';
```

```
1      select title: b.title, sales: b.calcBookSales()
2      from p in publishers, b in p.booksPublished
3      where p.publisherName = 'Springer-Verlag';
```

- Суть OORB: расширение реляционных СУБД, добавили типизированную таблицу (подобна классу в OODB), основана на UDT;
- Стандарт предложен в SQL3 (SQL1999);
- Можно иметь иерархии объектов;
- У записи (у объекта) такой таблицы есть OID;
- Ссылки в объектах по OID задают отношения между объектами:
 - ① score указывает на таблицу;
 - ② refs are checked — аналог ссылочной целостности, триггеров;

Пример: O-P модель данных, ORDB Schema, SQL Stan-d

<pre>create type personUdt as (personName varchar(15), address varchar(20)) instantiable not final ref is system generated; create table person of personUdt (primary key (personName), ref is personID system generated); create type authorUdt under personUdt as (authorSince varchar(10), wrote ref (bookUdt) scope book array [10] references are checked on delete no action) instantiable not final method calcAuthorBookSales() returns decimal; create table author of authorUdt under person; create type customerUdt under personUdt as (loginName varchar(10), buys ref (purchaseUdt) scope purchase array [50] references are checked on delete no action) instantiable not final method calcPurchases() returns decimal; create table customer of customerUdt under person (unique (loginName)); create type publisherUdt as (publisherName varchar(30), booksPublished ref (bookUdt) scope book array [1000] references are checked on delete set null) instantiable not final ref is system generated method calcTotalSales() returns decimal;</pre>	<pre>create table publisher of publisherUdt (primary key (publisherName), ref is publisherID system generated); create type purchaseUdt as (dateOfPurchase date, purchasedBy ref (customerUdt) scope customer references are checked on delete cascade, bookPurchased ref (bookUdt) scope book references are checked on delete cascade) instantiable not final ref is system generated; create table purchase of purchaseUdt (ref is purchaseID system generated); create type bookUdt as (isbn varchar(30), title varchar(50), listPrice decimal, authoredBy ref (authorUdt) scope author array[5] references are checked on delete no action, boughtBy ref (purchaseUdt) scope purchase array[1000] references are checked on delete set null, publishedBy ref (publisherUdt) scope publisher references are checked on delete no action) instantiable not final ref is system generated method calcBookSales() returns decimal; create table book of bookUdt (primary key (isbn), ref is bookID system generated);</pre>
--	--

3

Запросы в ОР модели [Urban and Dietrich, 2009]

- Запрос, прописываем OID:

```
1  update book set publishedBy =  
2      (select publisherID from publisher  
3       where publisherName = 'Prentice Hall')  
4  where isbn = '0-13-042898-1';
```

- По точкам тоже можно ходить:

```
1  select publishedBy.publisherName  
2  from book  
3  where isbn = '0-13-042898-1';
```

- Можно разыменовывать указатель:

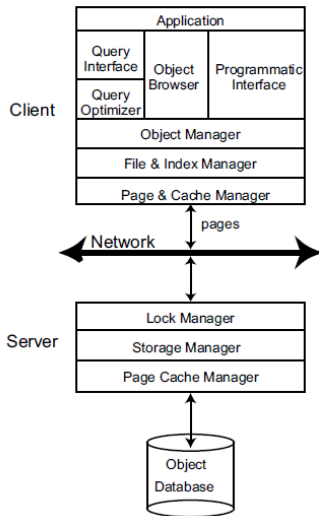
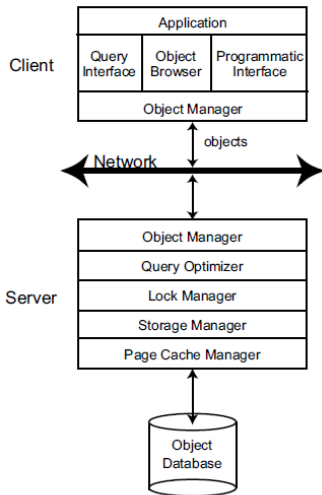
```
1  select deref(publishedBy)  
2  from book  
3  where isbn = '0-13-042898-1';
```

Если кому-то нужно полноценное введение в объекты **с оглядкой на базы данных**, то есть, не ясны понятия: объект, тип, класс, подкласс, наследование, композиция то читайте 15.1 (Distributed Object Database Management) из [Özsu and Valduriez, 2011].

Много всего:

- Варианты клиент/серверной архитектуры;
- Управление буфером на клиенте и на сервере;
- Кеши и их консистентность;
- Управление OID-ами;
- Миграция объектов;
- Кластеризация объектов;
- Распределенный GC;
- Обработка и выполнение запросов в OODB.

Варианты клиент/серверной архитектуры ОСУБД I



4

⁴ Изображение взято из [Özsu and Valduriez, 2011]

В зависимости от гранулярности данных и предоставляемой функциональности бывают:

- Объектные сервера:
 - Сервер делает основную работу;
 - Менеджер объектов (ОМ) предоставляет контекст для исполнения методов, можно выполнять на сервере и на клиенте;
 - Управление OID-ом — ответственность ОМ (удаление, GC);
 - На сервере ОМ отвечает за кластеризацию;
 - На клиенте и сервере кеши из объектов, на сервере страничный;
 - Оптимизацию пользовательских запросов делает сервер, он же синхронизацию транзакции;
- Страничные сервера:
 - Можно рассматривать как улучшенный storage manager: базовая единица обмена данными страница или сегмент, а не объект;

Страничный подход считается лучшим, хотя результаты не финальные:
+ когда кластеризация совпадает с запросом, когда один сервер и т.д.

- Страничные сервера:

- Упрощают код СУБД: страничные кеши на клиенте и сервере, единое представление данных вплоть до пользователя;
- Обновления просто заливать с клиента на сервер;
- Более полное использование ресурсов клиента при выполнении запросов;
- Нагрузку можно балансировать с помощью оптимизатора;
- Можно эффективно использовать ОС, железо (pointer swizzling);

Некоторые соображения на тему:

- Казалось бы, объектная модель по производительности должна иметь преимущества — если сервер будет “понимать” концепцию объекта:
 - Замки и логгирование на уровне объектов, а не страниц;
 - Фильтрация на сервере, чтобы слать меньше + не фильтровать;
- Фильтрация на сервере не так уж однозначно хороша. Нагрузка это смесь из запросов и пообъектной навигации. Для каждого объекта по вызову RPC — плохо (причина победы страничного подхода — навигационные тестовые нагрузки).
- Решение проблемы: слать код на сервер и там исполнять. Появляется другая проблема: безопасность.
- Есть системы умеющие переключаться между режимами: O_2 .

- Клиент. Если page fault информация берется с сервера. Бывает:
 - Объектный — на уровне объекта, (+) высокая гранулярность, конкурентность (-) фрагментация;
 - Страничный — на уровне страницы, (+) нет фрагментации (-) не угадали с шаблоном кластеризации → будем подгружать постоянно;
 - Двойной — есть оба, при сбросе страничного, откопируем полезные объекты в объектный. Копим хорошо кластеризованные страницы и объекты.
- Сервер. Если случается page fault — информация берется с диска.
 - Аналогичен реляционным в части общения с диском. Если нужны группы объектов, конструирует их из страниц и посылает на клиент.
 - MOB (Modified Object Buffer) — амортизация записи на диск.
 - LRU hate hints на сервере (разделять информацию на сервере и клиенте, стремиться чтобы всё обслуживалось клиентом).

Как реализовывать постоянный ID объекта? Объекты бывают постоянные, временные, созданные пользователем или системой.

- Physical identifier (POID) — физический адрес. Стратегия: OID приравнивается к физическому адресу объекта. Например, адрес страницы + смещение.
 - (+) Получаем адрес объекта из OID сразу;
 - (-) Родителей надо обновлять при переезде на новую страницу;
- Logical identifier (LOID) — каждому объекту, в рамках системы дается уникальный ID. Бывает:
 - Чистый LOID, счетчик в рамках системы;
 - Псевдо LOID, сервер + счетчик в рамках сервера;
- Для стратегии LOID надо держать таблицу трансляции в POID;
- Зато не надо трогать LOID при перемещении объекта.

- Объектные системы предпочитают LOID, из-за динамичности окружения в котором работает СУБД;
- Трюк: для маленьких значений не разделяемых по ссылке, хранить их значение в OID (меньше таблица, быстрее получение);
- Генерация LOID:
 - чистый LOID дорог: сетевые задержки, плюс допнагрузка на узел;
 - номер генерируется секвенцей, не переиспользуется при удалении.
- Трансляция LOID-POID:
 - Если чистый LOID и клиент подключен к нескольким серверам то таблица на клиенте;
 - Если псевдо LOID, то только на серверах;
 - На клиенте держать плохо: не масштабируемо;
 - Таблица хранится в виде B^+ -дерева или хеш-таблицы.

Pointer Swizzling I

При вычислении Path Expressions вида `c.engine.manufacturer.name` надо подгружать данные с диска. На диске OID, но в памяти лучше использовать указатели. Подход динамической подмена на указатели — `pointer swizzling` (словарь: “настройка по адресам”).

Методы:

- Хардварный: если приходит от ОС `page fault` загружаем страницу, конвертируем на ней все OID в указатели на несуществующие страницы. Резервируем место на эти страницы. Эти страницы подгружаются только когда их, в свою очередь, “тронут” (перехватываем `page fault` от ОС).
- Софтверный: используется таблица объектов, в ней подменяется указатель.
- Бывают `eager` и `lazy` софтверные схемы.

- Хардварный: лучше когда ходим по одной группе, меньше косвенных обращений (к таблице объектов);
- Плохая кластеризация — хардварный будет страдать;
- Удаления обрабатывать сложнее;
- При хардварной может кончиться память (постоянно резервируем);
- При хардварной трудно работать на уровне объектов.

Исполнение запросов в ОСУБД

[Özsu and Valduriez, 2011]

Базируется на принципах реляционных СУБД, однако есть отличия:

- Более сложная алгебра операций:
 - оперируем над наборами объектов;
 - наборы семантически разные: список, множество, bag;
 - \rightarrow необходим сложный механизм вывода типов.
- В реляционных стоимости основываются на АМ, физическом расположении данных. Информация легко доступна оптимизатору
 - стоимость выполнения методов сложнее оценить;
 - так как ЯП общего назначения, то методы надо оптимизировать;
 - инкапсуляция мешает, иногда оптимизатор игнорирует ее.
- Сложная (ссылочная) структура на объектах:
 - центральная проблема: оптимизация path expressions;
 - доступ к атрибутам через граф наследования тоже надо оптимизировать.

Пример: `c.engine.manufacturer.name`. Интересные моменты:

- Занимались с начала 80-х, проблема: могут быть и методы!
- Соединения на графе;
- Path index: аналог join index, можно строить не только для цепочки вызовов, но и для графа наследования;
- Access support relation: структура данных для хранения избранных path expressions.

Если кому-то интересно про оптимизацию в объектных системах то смотрите [Mitchel, 1993], [Ozsu and Blakeley, 1990], [Straube and Ozsu, 1990].

А начать стоит с дочитывания 15.6 (Distributed Object Database Management) из [Özsu and Valduriez, 2011].



Gail Anne Mitchel. Extensible Query Processing in an Object-Oriented Database. Technical Report TR93-16. Department of Computer Science, Brown University.
<ftp://ftp.cs.brown.edu/pub/techreports/93/cs93-16.pdf>



M. Tamer Özsu and Jose A. Blakeley. Query Processing in Object-Oriented Database Systems. 19 pages.
<https://cs.uwaterloo.ca/~tozsu/publications/odbms/wonkim/kimchap.pdf>



David D. Straube and M. Tamer Özsu. 1990. Queries and query processing in object-oriented database systems. ACM Trans. Inf. Syst. 8, 4 (October 1990), 387–430. DOI=<http://dx.doi.org/10.1145/102675.102678>



Rowe L. and Stonebraker M. The Postgres Data Model. In Proc. 13th Int. Conf. on Very Large Data Bases. 1987.



Susan D. Urban, Suzanne W. Dietrich. Object Data Models. Encyclopedia of Database Systems. Springer US, 2009. 1929–1935.
http://dx.doi.org/10.1007/978-0-387-39940-9_249



Özsu M.T. and Valduriez P. Principles of Distributed Database Systems, 3rd ed. Prentice-Hall, 2011.