Базы данных и Информационные системы

15/15 Мета модель

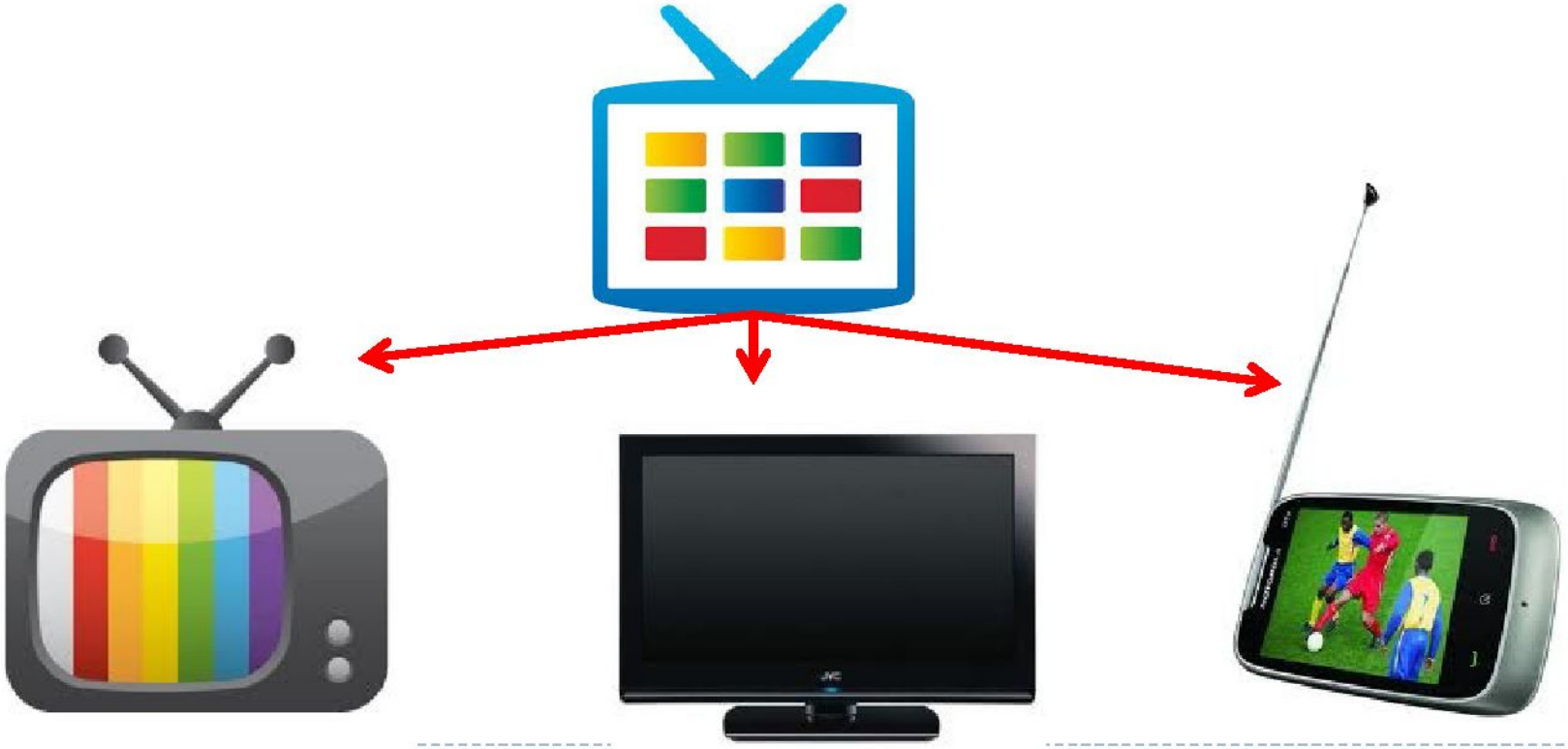
Кузиков Б.О.
Сумы, СумГУ
2013

Сегодня на занятии

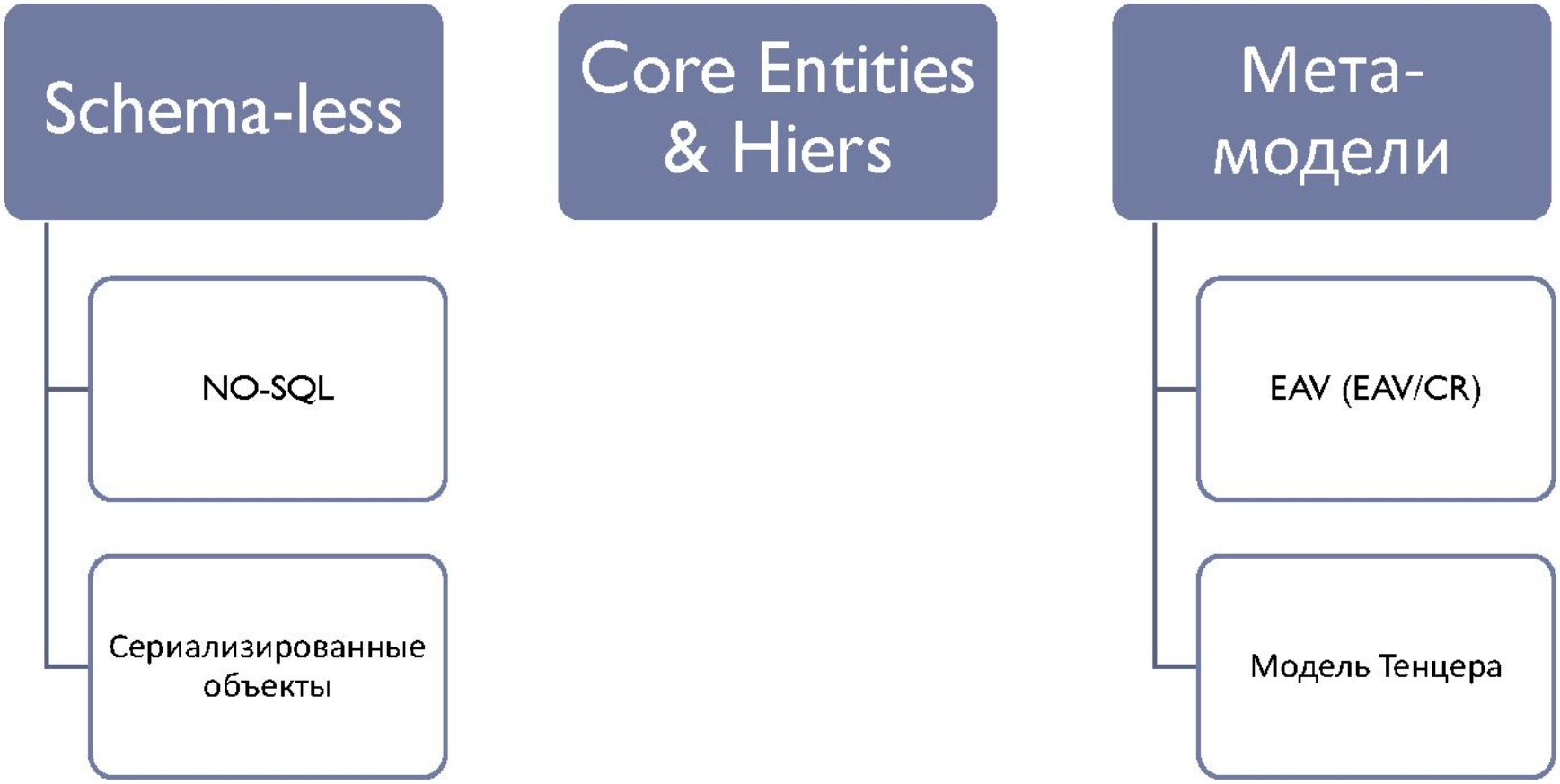
- Мы узнаем:
  - Какие подходы применяются при разработке приложений с часто меняющейся структурой БД.
  - Что такое метамодель
  - Как применить понятие метамодели к разработке БД

Проблема

- Мы решили продавать ИС для учета кадров. Новые заказчики – новые требования.
- Иллюстрирующий пример: Яндекс-маркет:



Как организовать хранение в БД?



1. Родительские сущности и наследники (Core Entities & Hiers)

Телевизоры

oid	Название	Размеры	Диагональ	Цена	Тип

«Трубочные»

Марка кинескопа	Яркость

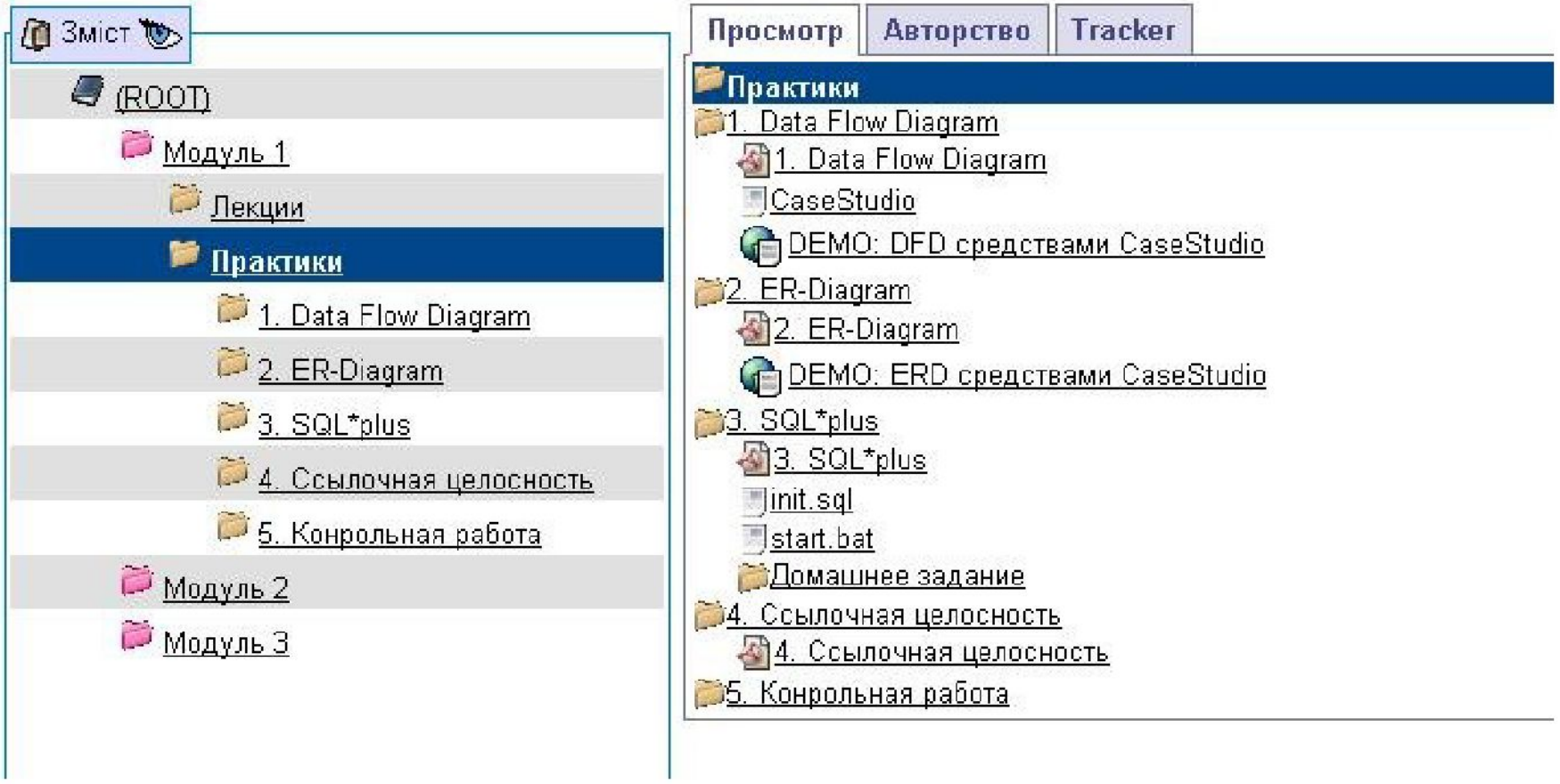
«Плоские»

Тип матрицы	Яркость	Разрешение

«Мобильные»

Время автономной работы

Применение подхода



1. Родительские сущности и наследники

Плюсы

- Вся мощь реляционных БД
- Простой контроль полноты данных

Минусы

- Высокие требования к проектированию
- Сложность добавления новых типов и атрибутов
- «Раздувание БД»

2. Schema-less (в БД)

Oid	Атирбуты(в XML, JSON, YML, INI, plain-text)
1	
2	
3	
4	

Тип данных XMLType

```
CREATE TABLE  storage(
empno NUMBER,
body XMLTYPE);

insert into storage values (2300,
 '<?xml version="1.0"?>
   <employe>
     <ename>King</ename>
     <sal>3000</sal>
    </employe>‘));

insert into storage values (2310,
'<?xml version="1.0"?>
  <employe>
     <ename>Kong</ename>
     <sal>3500</sal>
   </employe>‘));
```

```
SELECT empno
FROM storage
WHERE
  extractvalue(body,
  '//sal') > 3000;
```

```
EMPNO
---------
2310
```

2. Schema-less (в БД)

Плюсы

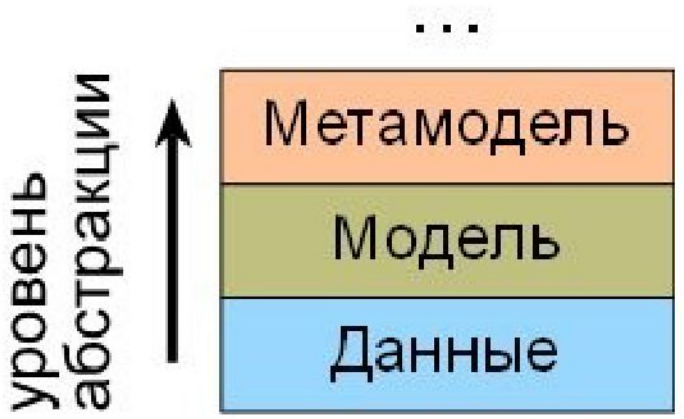
- Гибкая структура
- Снижает требования к предварительному проектированию БД
- Возможно использовать индексы для поиска

Минусы

- Какие атрибуты есть у объектов типа X?
- В каком диапазоне находятся цены для объектов типа У?
- Сложности конкурентного доступа
- Избыточность структуры
- Ограничения на значения параметров
- Одинаковые параметры с разными именами

Понятие метамодели

- Метамодели — это средства построения моделей (например, формальные языки или графические нотации для описания структуры классов, свойств и связей).
- Модели — это описание структуры данных.
- Данные — это множество простейших единиц информации, которые касаются не абстрактных, а конкретных сущностей.



http://ru.wikibooks.org/wiki/Метамоделирование

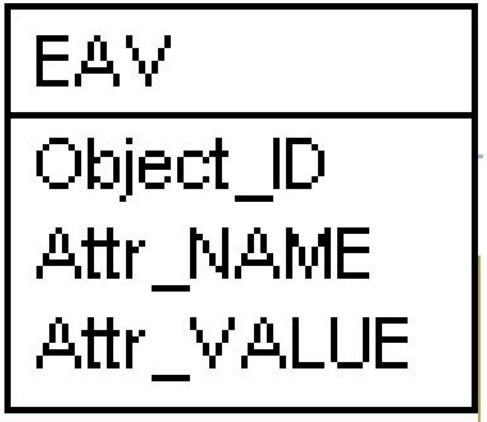
Модель Entity-Attribute-Value

EAV
Object_ID
Attr_NAME
Attr_VALUE

```
CREATE TABLE EAV (
Object_ID Varchar2(20),
Attr_NAME Varchar2(20),
Attr_VALUE Varchar2(20));
```

```
INSERT INTO EAV
  VALUES (1,'TYPE','TV');
INSERT INTO EAV
  VALUES (1,'MANUFACTOR', 'LG');
INSERT INTO EAV
  VALUES (1,'COST', '5000');
INSERT INTO EAV
  VALUES (1,'DIAGONAL', '32');
```

Типичные запросы



- Что известно об объекте №1?

```
SELECT attr_name, attr_value
FROM eav
WHERE object_id = 1;
```

- Какого типа есть объекты?

```
SELECT DISTINCT attr_value
FROM eav
WHERE attr_name= 'TYPE';
```

- Какие объекты типа «TV» есть?

```
SELECT object_id
FROM eav
WHERE attr_name= 'TYPE'
  AND attr_value='TV'
```

Типичные запросы

EAV
Object_ID
Attr_NAME
Attr_VALUE

- Какие атрибуты есть у объектов типа «TV»?

```
SELECT DISTINCT b.attr_name
FROM eav a
JOIN eav b using (object_id)
WHERE a.attr_name='TYPE'
  AND a.attr_value='TV'
```

- Максимальная и минимальная цена объекты типа «TV»?

```
SELECT max(b.attr_value),min(b.attr_value)
FROM eav a
JOIN eav b on (a.object_id = b.object_id)
WHERE a.attr_name='TYPE'
  AND a.attr_value='TV'
  AND b.attr_name='COST'
```

Модель Entity-Attribute-Value

Плюсы

- Реляционное представление
- Полностью гибкая схема
- Не надо хранить атрибуты, где не установлено значение
- Простая схема, простой доступ к отдельному атрибуту
- Легко добавить хранение версий атрибутов

Минусы

- Сложность запросов
- Избыточность на хранение имени атрибута
- Неопределённость типа атрибута
- Неопределённость допустимых значений атрибута

Дополнительное чтение

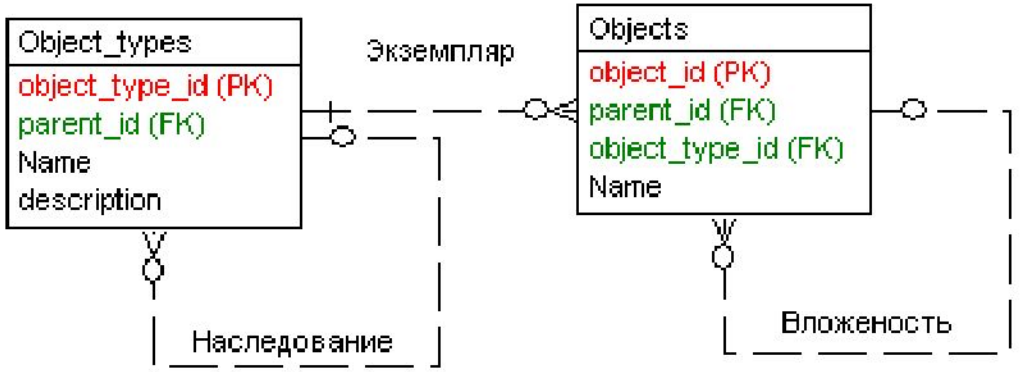
- http://ycmi.med.yale.edu/nadkarni/eav_CR_contents.htm
  - детальное описание + введение иерархий объектов и связей между ними.
- http://habrahabr.ru/post/45935/ - «Разворачивание широкой таблицы в столбец (EVA pattern)»
- http://en.wikipedia.org/wiki/Entity-Attribute-Value_model
  - словарная статья в Википедии

Модель Анатолия Тенцера

- http://www.compress.ru/article.aspx?id=11515&iid=452 – «**База данных — хранилище объектов**» Анатолий Тенцер, КомпьютерПресс 8'2001

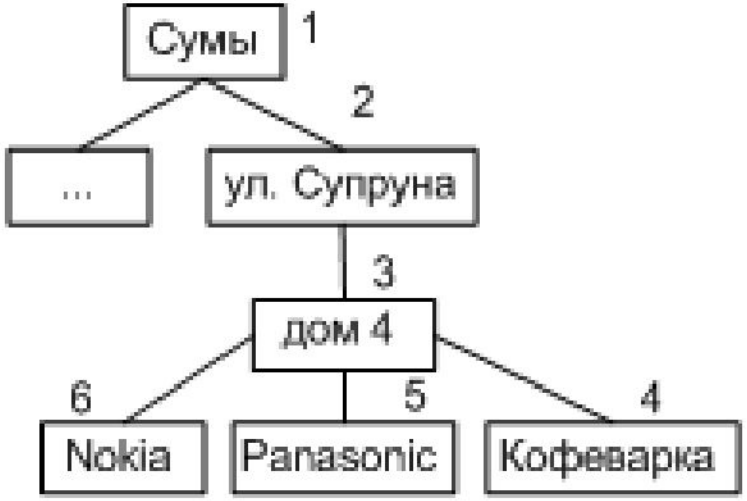
1. Каждая сущность, информация о которой хранится в БД, — это объект.
2. Каждый объект уникален в пределах БД и имеет уникальный идентификатор.
3. Объект имеет свойства (строковые, числовые, временные, перечислимые), которые описывают атрибуты сущности.
4. Объекты могут быть связаны между собой произвольным образом. Связь характеризуется связанными объектами и типом связи.
5. Объект может быть хранилищем. В этом случае допускается хранение в нем других объектов (например, товара на складе).

Шаг 1: Объекты и классы

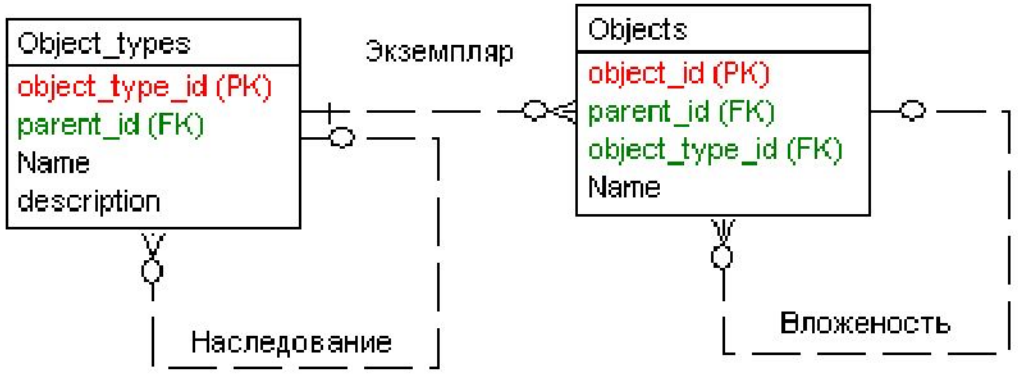


- Пример Классы

- Пример Объекты



Классы



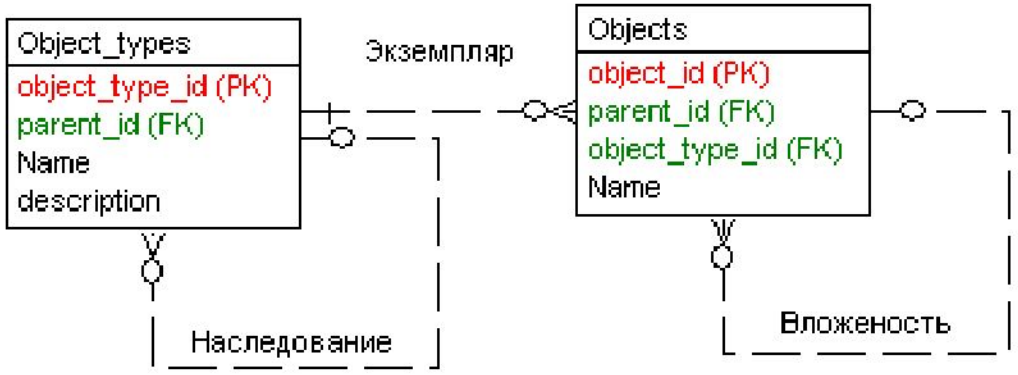

```
create table object_types (
 object_type_id number
    primary key,
 parent_id number,
 name varchar2(20) unique ,
 description varchar2(20),
) ;
alter table object_types add foreign
key (parent_id) references
object_types (object_type_id)
```

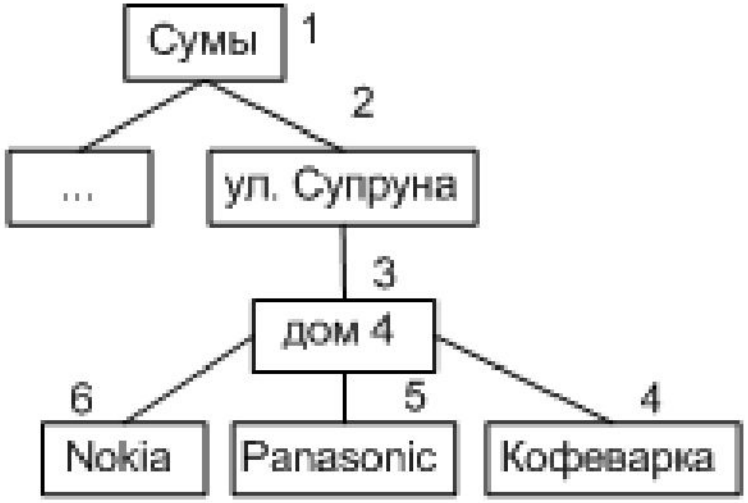
```
Insert into object_types values
(1,null,'контейнер','родительски
й класс');
```

```
Insert into object_types values
(2,1,'жилье','класс для
помещений');
```

```
Insert into object_types values
(3,1,'оборудование','класс для
оборудования');
```

Объекты






```
Create table Objects (
 object_id Number primary key ,
 parent_id Number,
 object_type_id Number NOT NULL,
 name Varchar2(20)
);
Alter table Objects add  foreign key
(parent_id) references Objects
(object_id);
Alter table Objects add  foreign key
(object_type_id) references
Object_types (object_type_id);
```

```
Insert into objects
values (1,null,1,'Сумы');
```

```
Insert into objects
values (2,1,1,'ул. Супруна');
```

```
Insert into objects
values (3,2,2,'дом 4');
```

```
Insert into objects
values (4,3,3,'Кофеварка');
```

Типичные запросы

- Какого типа есть объекты?

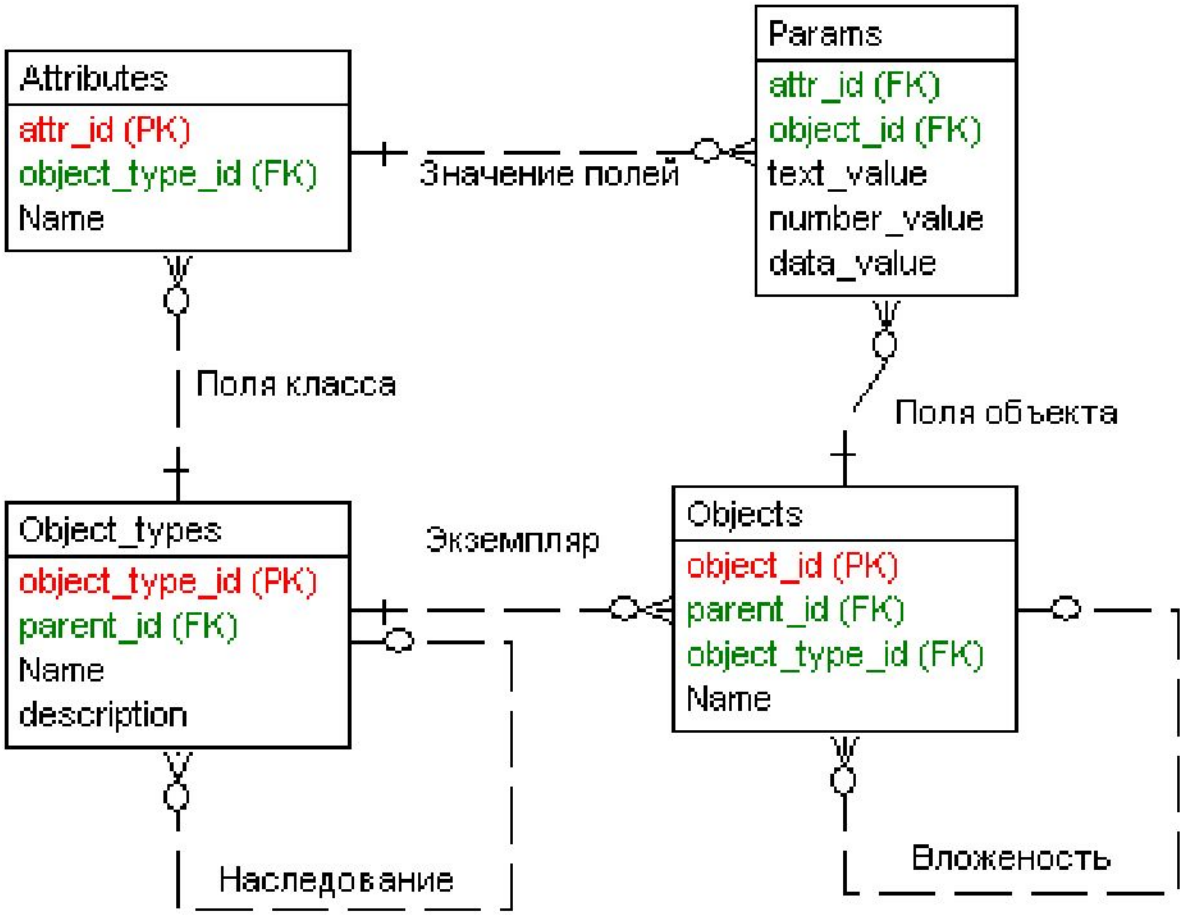
```
SELECT name
FROM object_types;
```

```
SELECT DISTINCT object_types.name
FROM objects
JOIN object_types using (object_type_id)
```

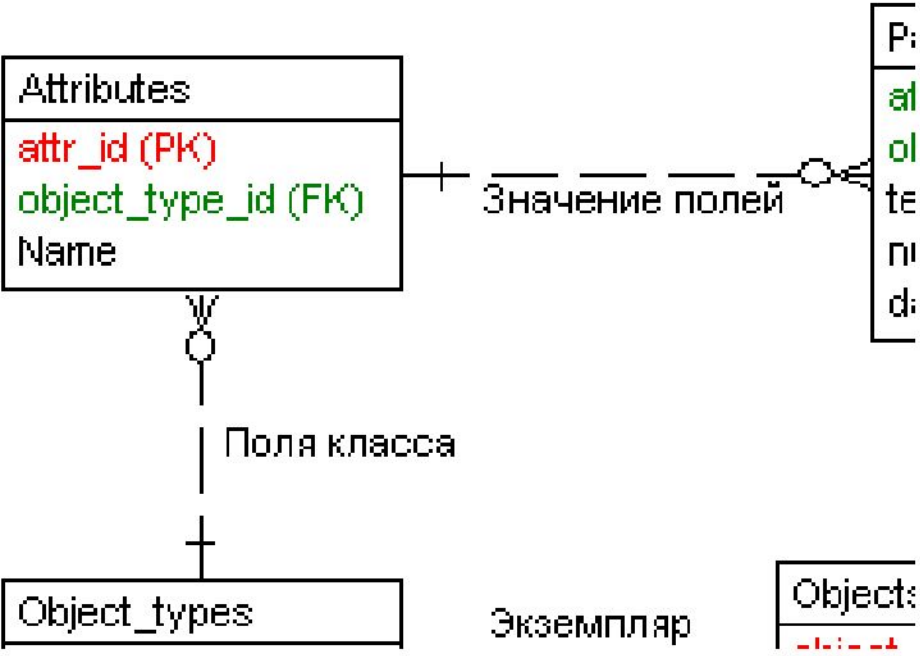
- Какие объекты типа «TV» есть?

```
SELECT objects.name
FROM objects
JOIN object_types using (object_type_id)
WHERE object_types.name = 'TV';
```

Шаг 2: Усложняем модель



Атрибуты класса

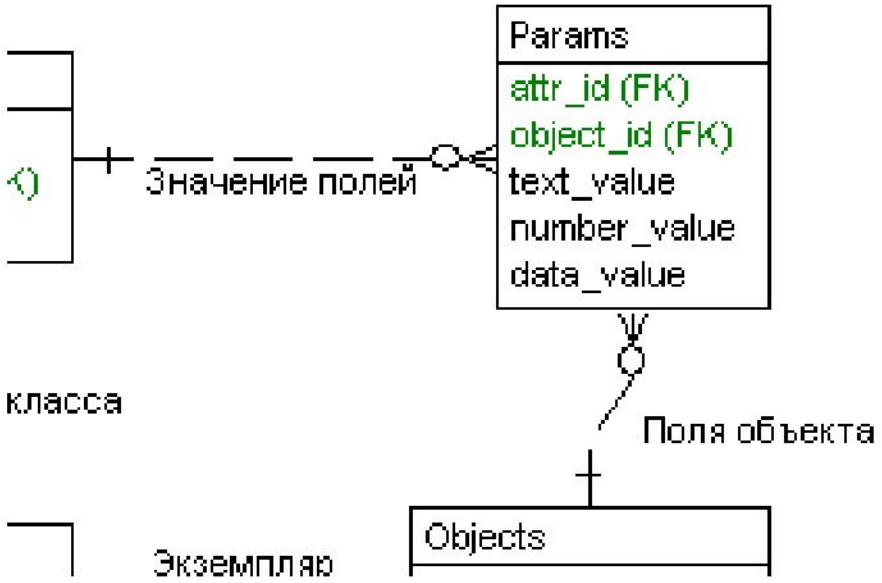



```
Create table Attributes (
 attr_id Number PRIMARY KEY,
 object_type_id Number NOT NULL,
 name Varchar2(20)
);


Alter table Attributes add
unique (attr_id,Name);


Alter table Attributes add
foreign key (object_type_id)
references
 Object_types(object_type_id);
```

Параметры объекта




```
Create table Params (
  attr_id Number NOT NULL,
  object_id Number NOT NULL,
  text_value Varchar2(20),
  number_value Number,
  date_value Date
);

Alter table Params add
foreign key (object_id)
references Objects (object_id)
on delete cascade;

Alter table Params add
foreign key (attr_id)
references Attributes
(attr_id) on delete cascade;
```

Соотношение понятий

ООП	Реляционная модель	Метамодель
Класс	Таблица	Запись в таблице Object_types
Объект (Экземпляр класса)	Запись в таблице	Запись в таблице Objects
Поле, свойство (property)	Атрибут (столбец)	Запись в таблице Attributes
Значение поля	Значение атрибута	Запись в таблице Params

Типичные запросы

- Какие параметры у объекта №1?

```
SELECT attr.name, params.text_value, params.number_value,
params.date_value
FROM params
JOIN attributes attr using (attr_id)
WHERE object_id = 1;
```

```
SELECT attr.name, params.text_value, params.number_value,
params.date_value
FROM objects o
JOIN attributes attr
   using (object_type_id)
LEFT JOIN params
    on (     params.attr_id= attr.attr_id
         AND o.object_id   = params.object_id)
WHERE o.object_id = 1;
```

Типичные запросы

- Цена объекта №1?

```
SELECT params.number_value
FROM objects o
JOIN attributes attr using (object_type_id)
LEFT JOIN params
    on (     params.attr_id= attr.attr_id
         AND o.object_id   = params.object_id)
WHERE object_id = 1
  AND attr.name='COST'
```

```
SELECT params.number_value
FROM objects o
JOIN params           using (object_id)
JOIN attributes attr using (attr_id)
WHERE object_id = 1
  AND attr.name='COST'
```

Типичные запросы

Какая максимальная и минимальная цена у объектов «TV»

```
SELECT max(params.number_value), min(params.number_value)
FROM attributes attr
JOIN object_types ot using (object_type_id)
JOIN params  using (attr_id)
WHERE attr.name = 'COST'
  AND ot.name = 'TV';
```

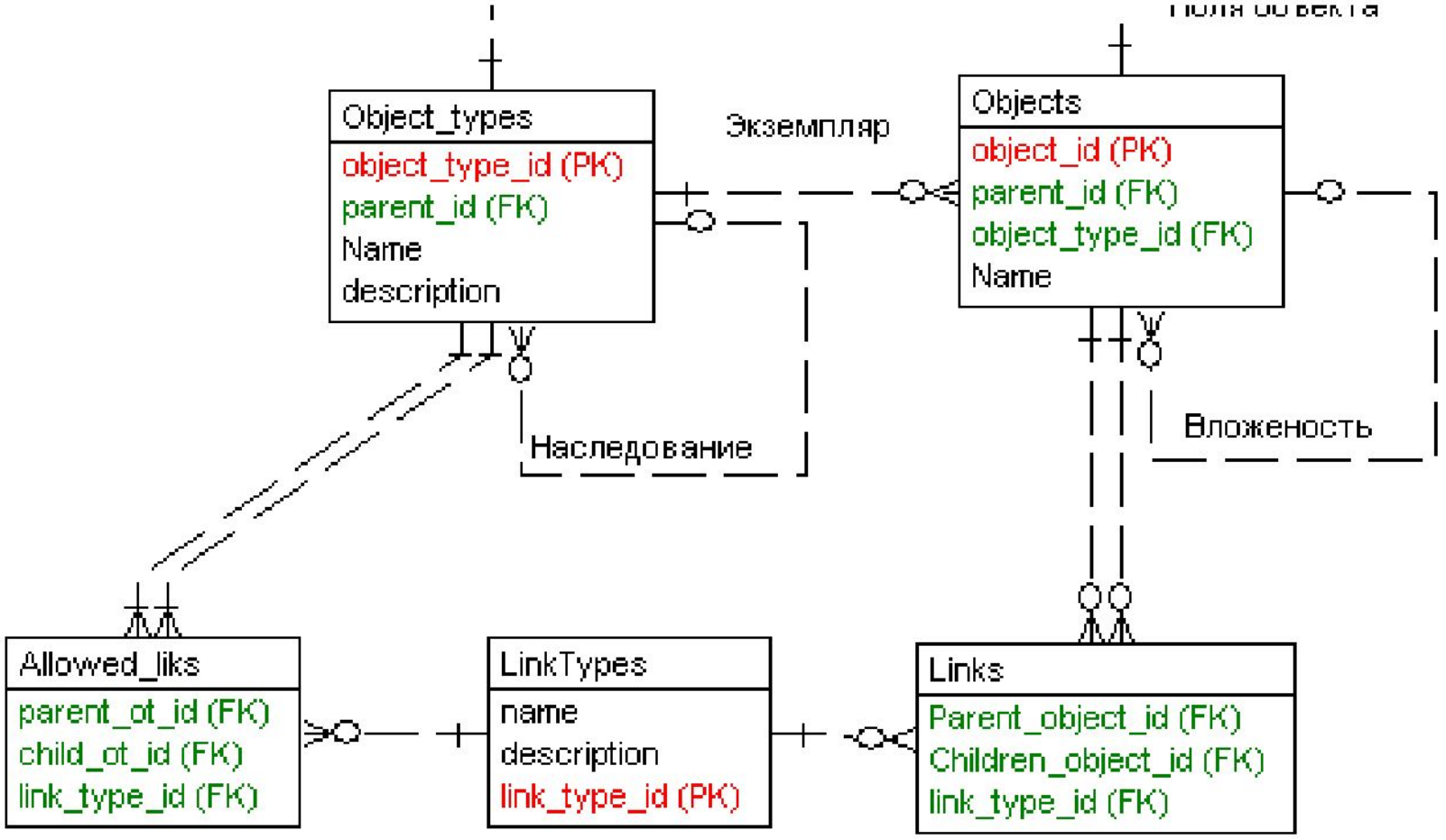
Вернемся к EMP и DEPT

```
SELECT ename, job, deptno
  FROM Emp
  WHERE job='CLERK‘ and depno = 5;
```

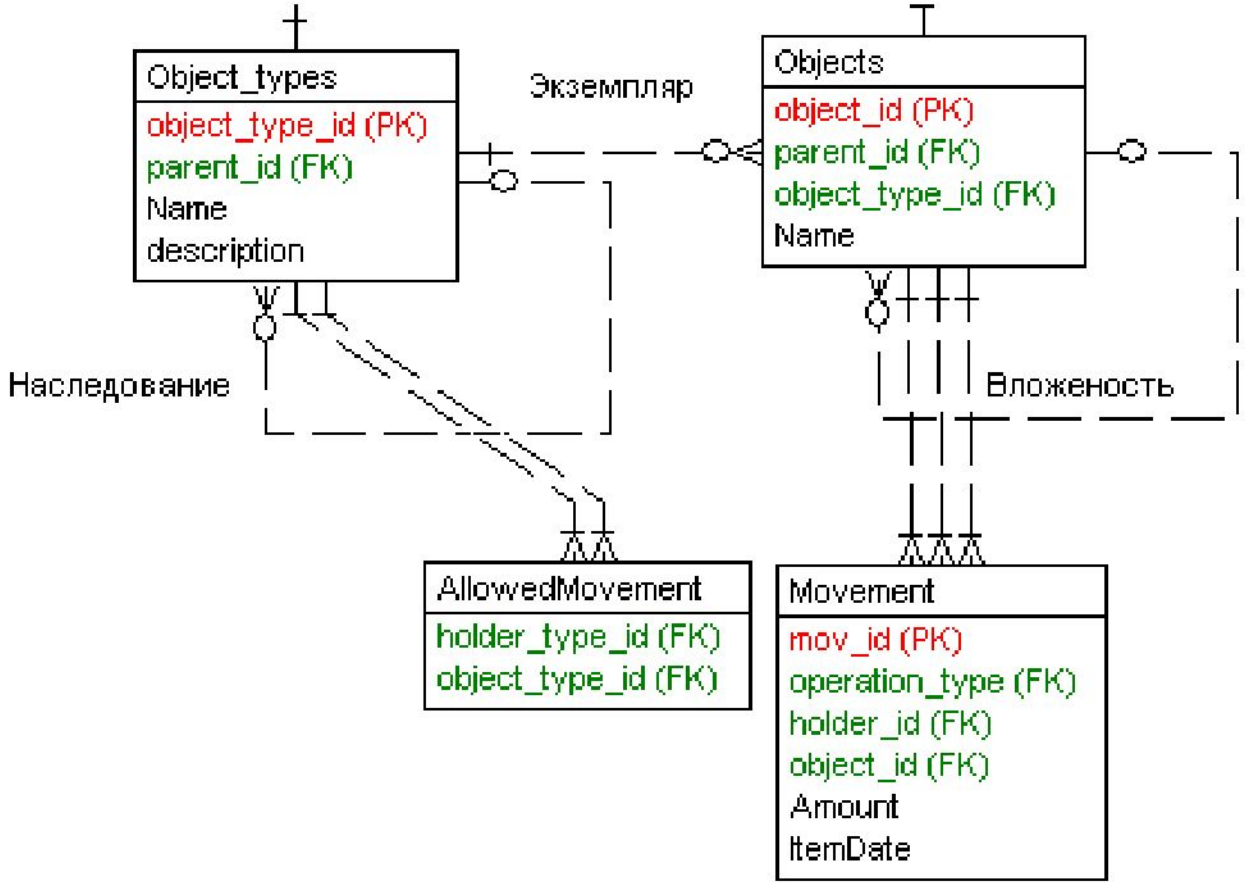
- Теперь ее эквивалент выглядит так:

```
SELECT Obj.name, Job.text_value, Deptno.number_value
  FROM objects Obj
JOIN params Job ON     Job.object_id  = Obj.object_id
                   AND Job.text_value = ‘CLERCK’
JOIN attributes j_attr ON     j_attr.attr_id = Job.attr_id
                          AND j_attr.name = ‘job’
JOIN params Deptno ON     Deptno.object_id = Obj.object_id
                      AND Deptno.number_value = 5
JOIN attributes d_attr ON     d_attr.attr_id = Deptno.attr_id
                          AND d_attr.name = ‘deptno’
```

Расширение 1: Связи



Расширение 2:Приход-расход



Модель Тенцера

Плюсы

- Снижение избыточности по сравнению с EAV
- Все плюсы реляционного представления данных
- Простое версионирование значений
- Легкая расширяемость

Минусы

- Ослабленный контроль за целостностью данных
- Относительно сложные для зпаписи/понимания запросы
- Понижение производительности запросов

-1: Ослабленный контроль за целостностью данных

Нарушение ограничений домена	Добавление таблицы с типами атрибутов и ограничениями на них.
Вставка не всех/лишних атрибутов в объект	Создание хранимых процедуры для вставки типовых объектов с контролем их целостности
Удаление/изменение связанных объектов	Создание триггеров на удаление/изменение объектов/атрибутов

-2: Снижение производительности

```
SELECT O.name, O.name as ename,
       S1.number_value AS MGR, S2.number_value AS sal,
       S3.number_value AS comm, s4.date_value AS hiredate
FROM Objects O
JOIN Object_Types OT ON OT.Object_type_id = O.Object_type_id
LEFT JOIN params S1 ON O.object_id = S1.object_id
LEFT JOIN attributes SD1 ON S1.attr_id = SD1.attr_id
LEFT JOIN params S2 ON O.object_id = S2.object_id
LEFT JOIN attributes SD2 ON S2.attr_id = SD2.attr_id
LEFT JOIN params S3 ON O.object_id = S3.object_id
LEFT JOIN attributes SD3 ON S3.attr_id = SD3.attr_id
LEFT JOIN params S4 ON O.object_id = S4.object_id
LEFT JOIN attributeS SD4 ON S4.attr_id = SD4.attr_id
WHERE  OT.name = 'emp'
   AND SD1.name = 'mng'
   AND SD2.name = 'sal'
   AND SD3.name = 'comm'
   AND SD4.name = 'hiredate’
```

-3:Сложности в записи запросов

```
CREATE VIEW vObjects AS
SELECT O.*, OT.name as class
FROM Objects O
JOIN Object_Types OT
ON o.object_type_id = ot.object_type_id

CREATE VIEW vNumber AS
SELECT object_id,
 number_value as value, attr.Name
FROM Params
JOIN Attributes attr using(attr_id)

CREATE VIEW vDate AS
SELECT object_id,
 date_value as value, attr.Name
FROM Params
JOIN Attributes attr using(attr_id)
```

```
SELECT O.name, O.name as ename,
       S1.value AS MGR,
       S2.value AS sal,
       S3.value AS comm,
       s4.value AS hiredate
FROM vObjects O
LEFT JOIN vNumber S1
    ON O.object_id = S1.object_id
LEFT JOIN vNumber S2
    ON O.object_id = S2.object_id
LEFT JOIN vNumber S3
    ON O.object_id = S3.object_id
LEFT JOIN vDate   S4
    ON O.object_id = S4.object_id
WHERE  O.class = 'emp'
   AND S1.name = 'mng'
   AND S2.name = 'sal'
   AND S3.name = 'comm'
   AND S4.name = 'hiredate'/
```

Сравните:

```
SELECT O.name, O.name as ename,
       S1.number_value AS MGR,
S2.number_value AS sal,
       S3.number_value AS comm
FROM Objects O
JOIN Object_Types OT ON
OT.Object_type_id = O.Object_type_id
LEFT JOIN params S1 ON O.object_id =
S1.object_id
LEFT JOIN attributes SD1 ON S1.attr_id
= SD1.attr_id
LEFT JOIN params S2 ON O.object_id =
S2.object_id
LEFT JOIN attributes SD2 ON S2.attr_id
= SD2.attr_id
LEFT JOIN params S3 ON O.object_id =
S3.object_id
LEFT JOIN attributes SD3 ON S3.attr_id
= SD3.attr_id
WHERE  OT.name = 'emp'
   AND SD1.name = 'mng'
   AND SD2.name = 'sal'
   AND SD3.name = 'comm'
```

```
SELECT O.name, O.name as ename,
       S1.value AS MGR,
       S2.value AS sal,
       S3.value AS comm,
FROM vObjects O
LEFT JOIN vNumber S1
   ON O.object_id = S1.object_id
LEFT JOIN vNumber S2
   ON O.object_id = S2.object_id
LEFT JOIN vNumber S3
   ON O.object_id = S3.object_id
WHERE  O.class = 'emp'
   AND S1.name = 'mng'
   AND S2.name = 'sal'
   AND S3.name = 'comm'
```

Выводы

- Сегодня мы разобрали способы реализации схемы БД если она часто меняется:
  - Schema-less
  - Core Entities & Hiers
  - Entity-Attribute-Values
  - Модель Тенцера

Дополнительное чтение

- http://www.rsdn.ru/forum/db/4714713 - Обсуждение статьи о универсальной модели данных
- http://habrahabr.ru/post/164803/ - Реализация Schema-less решения в БД
- http://www.interface.ru/home.asp?artId=24052 - Универсальная модель данных. Оценка производительности запросов к БД, если вся БД представлена мета-моделью.
- http://www.compress.ru/article.aspx?id=11515&iid=452 – «**База данных — хранилище объектов**»
- **СРАВНИТЕЛЬНЫЙ АНАЛИЗ НЕКОТОРЫХ МЕТОДОВ О – R-ПРЕОБРАЗОВАНИЯ** *О.А. Змеев, А.Н. Моисеев*