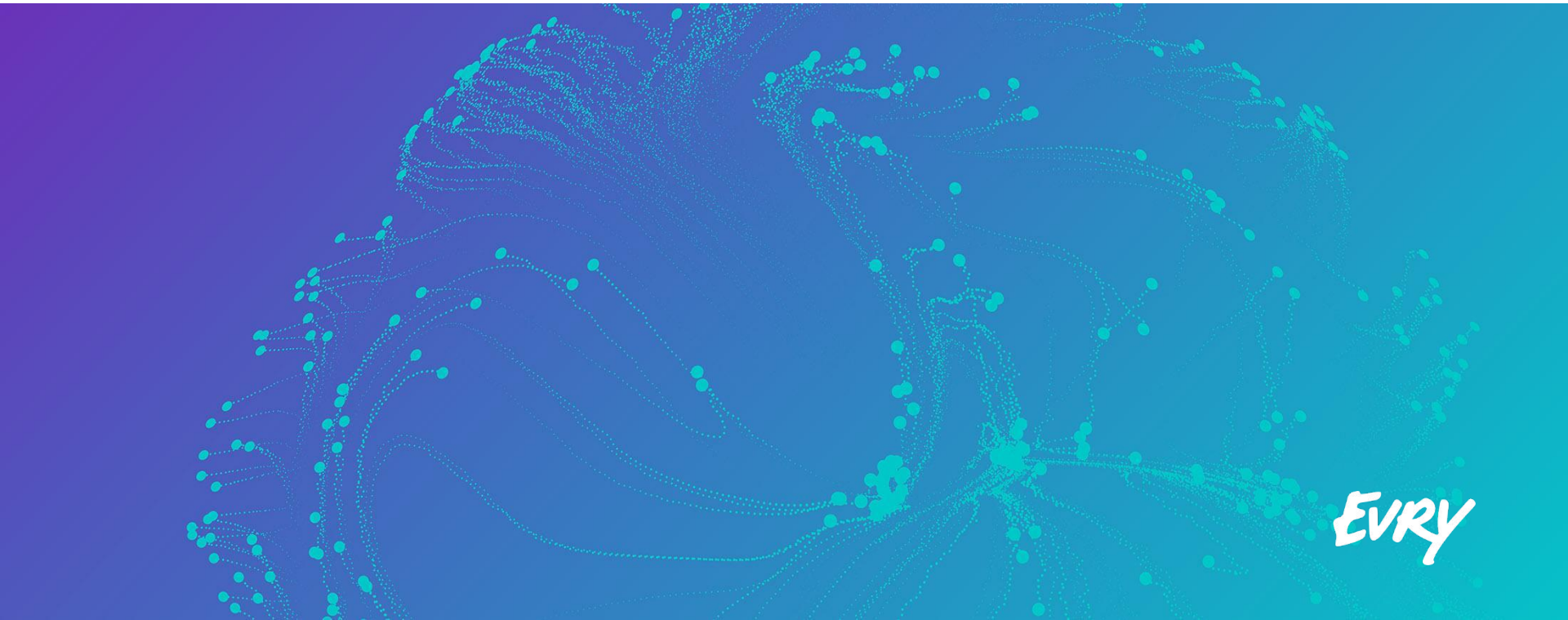
Data Base overview



Agenda

- SQL and DB basis
- IDEF0,1,2,3 diagrams
- ER diagram
- SQL: CREATE, UPDATE, DELETE, SELECT
- Exercises

DB Overview

База данных

База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Систе́ма управле́ния ба́зами да́нных (**СУБД**) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных[1]

Реляционная БД

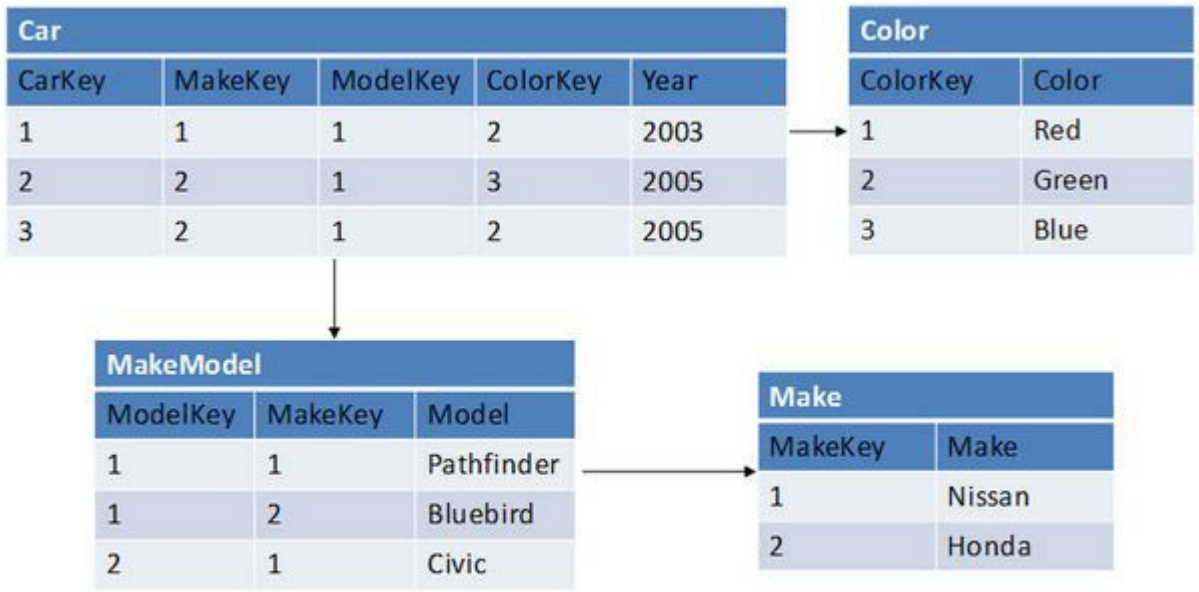
Реляционная база данных представляет собой набор таблиц (сущностей).

Таблицы состоят из колонок и строк (кортежей). Внутри таблиц могут быть определены ограничения, между таблицами существуют отношения.

При помощи SQL можно выполнять запросы, которые возвращают наборы данных, получаемых из одной или нескольких таблиц. В рамках одного запроса данные получаются из нескольких таблиц путем их соединения (JOIN), чаще всего для соединения используются те же колонки, которые определяют отношения между таблицами. Нормализация — это процесс структурирования модели данных, обеспечивающий связность и отсутствие избыточности в данных.

Доступ к реляционным базам данных осуществляется через реляционные системы управления базами данных (РСУБД). Почти все системы баз данных, которые мы используем, являются реляционными, такие как Oracle, SQL Server, MySQL, Sybase, DB2, TeraData и так далее.

Пример реляционной модели данных



Car

CarKey	MakeKey	ModelKey	ColorKey	Year
1	1	1	2	2003
2	2	1	3	2005
3	2	1	2	2005

Color

ColorKey	Color
1	Red
2	Green
3	Blue

MakeModel

ModelKey	MakeKey	Model
1	1	Pathfinder
1	2	Bluebird
2	1	Civic

Make

MakeKey	Make
1	Nissan
2	Honda

Пример типичной реляционной модели данных

Нормальные формы

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется **нормализацией**. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных.[1] Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в базе данных информации.

Назначение процесса нормализации заключается в следующем:

- исключение некоторых типов избыточности;
- устранение некоторых аномалий обновления;
- разработка проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;
- упрощение процедуры применения необходимых ограничений целостности.

Первая нормальная форма

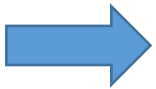
Переменная отношения находится в первой нормальной форме тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов[1].

Конкретнее, рассматриваемая таблица должна удовлетворять следующим пяти условиям:

- Нет упорядочивания строк сверху вниз (другими словами, порядок строк не несет в себе никакой информации).
- Нет упорядочивания столбцов слева направо (другими словами, порядок столбцов не несет в себе никакой информации).
- Нет повторяющихся строк.
- Каждое пересечение строки и столбца содержит ровно одно значение из соответствующего домена (и больше ничего).
- Все столбцы являются обычными[1].

Исходная ненормализованная (то есть не являющаяся правильным представлением некоторого отношения) таблица:

Сотрудник	Номер телефона
Иванов И. И.	283-56-82 390-57-34
Петров П. П.	708-62-34



Сотрудник	Номер телефона
Иванов И. И.	283-56-82
Иванов И. И.	390-57-34
Петров П. П.	708-62-34

Вторая нормальная форма

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут *неприводимо* зависит от её потенциального ключа.[1]

Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.[1] Для неприводимой функциональной зависимости часто используется эквивалентное понятие «полная функциональная зависимость».[1]
Если потенциальный ключ является простым, то есть состоит из единственного атрибута, то любая функциональная зависимость от него является неприводимой (полной). Если потенциальный ключ является составным, то согласно определению второй нормальной формы в отношении не должно быть неключевых атрибутов, зависящих от *части* составного потенциального ключа.
Вторая нормальная форма по определению запрещает наличие неключевых атрибутов, которые *вообще не зависят* от потенциального ключа. Таким образом, вторая нормальная форма в том числе запрещает создавать отношения как несвязанные (хаотические, случайные) наборы атрибутов.

Пример второй нормальной формы

R

Филиал компании	**Должность**	**Зарплата**	**Наличие компьютера**
Филиал в Томске	Уборщик	20000	Нет
Филиал в Москве	Программист	40000	Есть
Филиал в Томске	Программист	25000	Есть

Допустим, что зарплата зависит от филиала и должности, а наличие компьютера зависит только от должности.

Существует функциональная зависимость *Должность* → *Наличие компьютера*, в которой левая часть (детерминант) является лишь частью первичного ключа, что нарушает условие второй нормальной формы.

Для приведения к 2NF исходное отношение следует декомпозировать на два отношения:

R1

Филиал компании	**Должность**	**Зарплата**
Филиал в Томске	Уборщик	20000
Филиал в Томске	Программист	25000
Филиал в Москве	Программист	40000

R2

Должность	**Наличие компьютера**
Уборщик	Нет
Программист	Есть

IDEF0

IDEF0 — методология функционального моделирования (англ. *function modeling*) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов.
Функциональная модель компании

Функциональная модель IDEF0 представляет собой набор блоков, каждый из которых представляет собой «черный ящик» со входами и выходами, управлением и механизмами, которые детализируются (декомпозируются) до необходимого уровня. Наиболее важная функция расположена в верхнем левом углу. А соединяются функции между собой при помощи стрелок и описаний функциональных блоков. При этом каждый вид стрелки или активности имеет собственное значение. Данная модель позволяет описать все основные виды процессов, как административные, так и организационные.

Стрелки могут быть:

Входящие – вводные, которые ставят определенную задачу.
Исходящие – выводящие результат деятельности.
Управляющие (сверху вниз) – механизмы управления (положения, инструкции и пр).
Механизмы (снизу вверх) – что используется для того, чтобы произвести необходимую работу.

Входящие и исходящие стрелки точнее было бы называть вводящими и выводящими, так как по-английски они называются Input и Output соответственно. Но особенности перевода и привычные названия выглядят уже так, как сложилось. И все же для правильного понимания терминов важно помнить их значение в данном случае. Это подтверждается еще и тем, что данная нотация создана прежде всего для разработки ПО, и термины переводить правильнее в этой точки зрения.

IDEF0

Стрелки подписываются при помощи имен существительных (опыт, план, правила), а блоки – при помощи глаголов, т.е. в них описываются действия, которые производятся (создать товар, заключить договор, произвести отгрузку).

IDEF0 – это очень простой и одновременно наглядный язык описания бизнес-процессов. С помощью этого стандарта возможна передача информации между разработчиками, консультантами и пользователями. Стандарт очень тщательно разрабатывался, он удобен для проектирования, универсален. Для работы с ним существует множество инструментов, например, VISIO, BPWIN, ERWIN, Bussines studio и т.д.

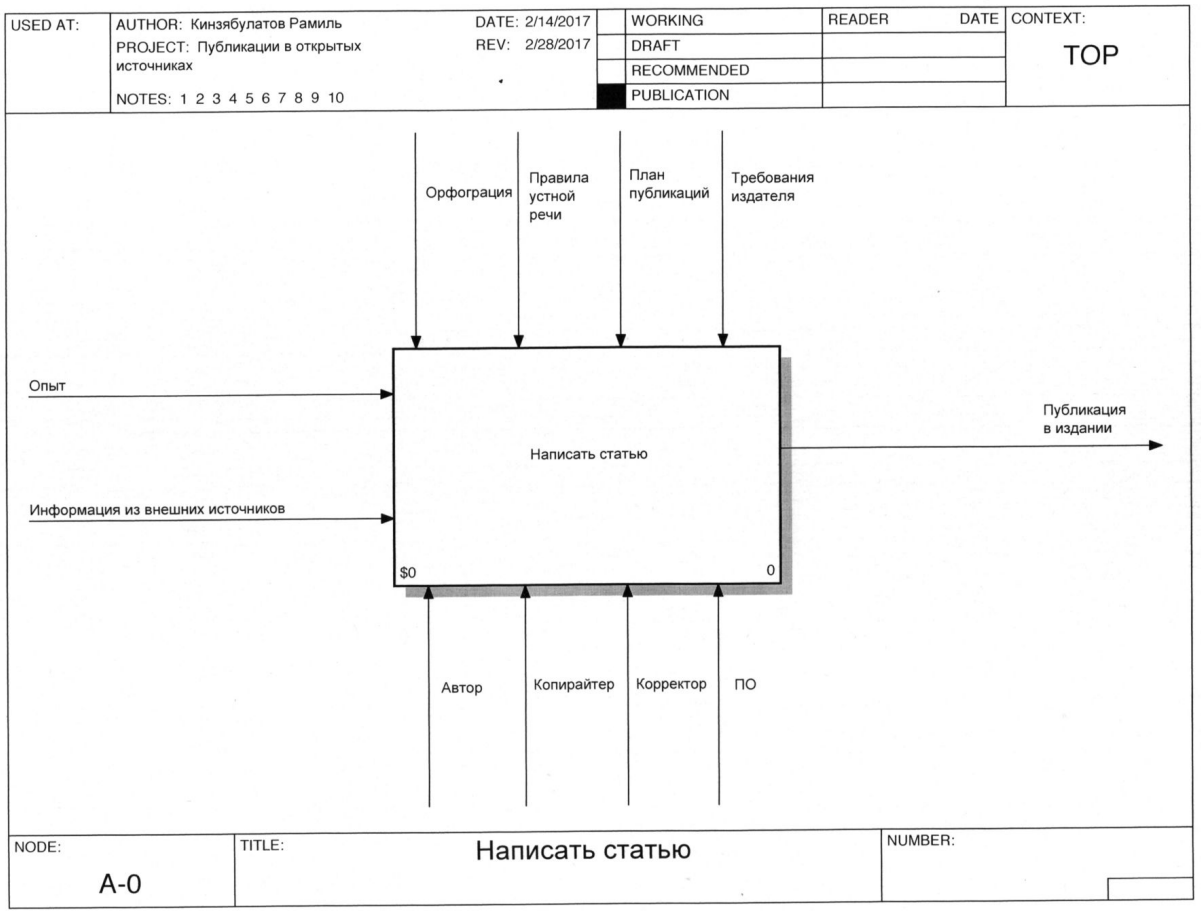
Кроме того, использование для создания бизнес-моделей IDEF0 — это не только удобно, это еще и правильно. Этот инструмент был разработан для бизнес-аналитики, он прошел длительную и тщательную отладку и шлифовку. А потому при помощи IDEF0 создать функциональную модель без ошибок намного проще, чем без применения этого стандарта.

Как известно, забивать гвозди лучше всего молотком. Конечно, вы можете для этого применять и другие инструменты, но молоток — наиболее функционален и с его помощью проще всего забить гвоздь аккуратно и точно. Так и с применением IDEF0 — этот инструмент был создан для функционального моделирования, и с его помощью вы намного быстрее и точнее сможете получить нужный результат.

Пример

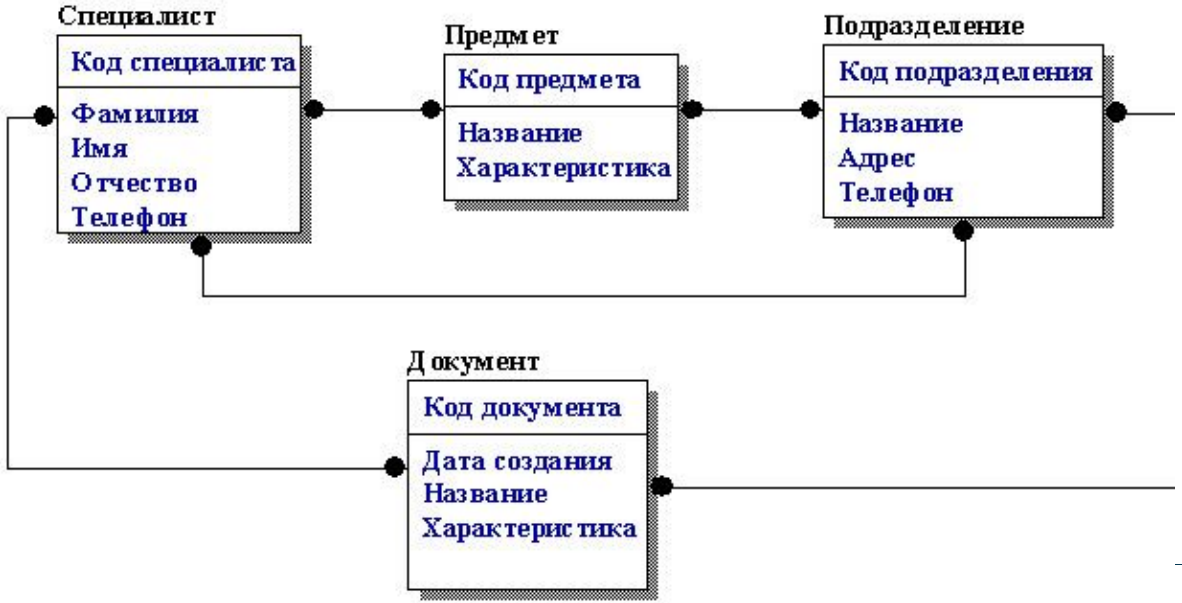
.

Основной блок – «Написать статью».



IDEF1

IDEF1 (*integration definition for information modeling*) — одна из методологий семейства IDEF. Применяется для построения информационной модели, которая представляет структуру информации, необходимой для поддержки функций производственной системы или среды.



ER- диаграмма

ER-модель (от англ. *entity-relationship model*, *модель «сущность — связь»*) — модель данных, позволяющая описывать концептуальные схемы предметной области.

ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время проектирования баз данных происходит преобразование ER-модели в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.).
ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена ***диаграмма «сущность — связь»*** (англ. *entity-relationship diagram, ERD*, *ER-диаграмма*).
Понятия *ER-модель* и *ER-диаграмма* часто не различают, хотя для визуализации ER-моделей могут быть использованы и другие графические нотации, либо визуализация может вообще не применяться (например, использоваться текстовое описание).

ER-диаграмма

Базовые понятия ERD

- **Сущность** (таблица, отношение) — это представление набора реальных или абстрактных объектов (людей, вещей, мест, событий, идей, комбинаций и т. д.), которые можно выделить в одну группу, потому что они имеют одинаковые характеристики и могут принимать участие в похожих связях. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Каждая сущность в модели изображается в виде прямоугольника с наименованием.

Можно сказать, что **Сущности** представляют собой множество реальных или абстрактных вещей (людей, объектов, событий, идей и т. д.), которые имеют общие **атрибуты** или характеристики.

- **Экземпляр сущности** (запись, кортеж)- это конкретный представитель данной сущности.
- **Атрибут сущности** (поле, домен) — это именованная характеристика, являющаяся некоторым свойством сущности.
- **Связь** — это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою. Связи позволяют по одной сущности находить другие сущности, связанные с ней.

Ключ или *потенциальный ключ* – это минимальный набор атрибутов, по значениям которых можно однозначно выбрать требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждая сущность должна но не обязана обладать хотя бы одним возможным ключом.

ER-диаграмма

Каждая **связь** может иметь один из следующих **типов связи**:

- **Один-к-одному, многое-ко-многим, один-ко-многим**.

Связь типа **один-к-одному** означает, что **один экземпляр первой сущности** (левой) связан с **одним экземпляром второй сущности** (правой). Связь **один-к-одному** чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Связь типа **многое-ко-многим** означает, что **каждый экземпляр первой сущности** может быть связан с **несколькими экземплярами второй сущности**, и каждый **экземпляр второй сущности** может быть связан с **несколькими экземплярами первой сущности**. Тип связи **много-ко-многим** является **временным** типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа **один-ко-многим** путем создания промежуточной сущности.

Связь типа **один-ко-многим** означает, что **один экземпляр первой сущности**(левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны «один») называется **родительской**, правая (со стороны «много») — **дочерней**.

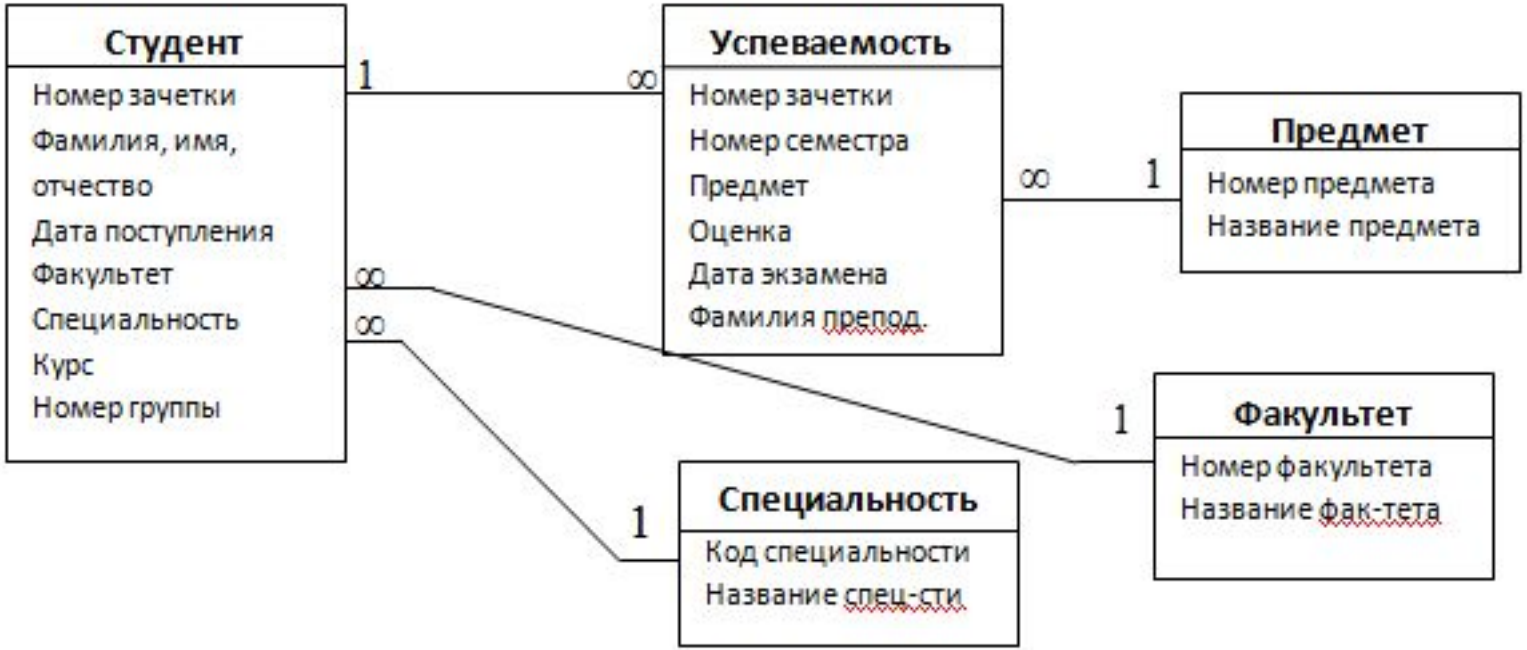
При разработке **ER-моделей** необходимо обследовать предметную область (организацию, предприятие) и выявить:

1) Сущности, о которых хранятся данные в организации (предприятии), например, люди, места, идеи, события и т.д., (будут представлены в виде блоков);
2) Связи между этими сущностями (будут представлены в виде линий, соединяющих эти блоки);
3) Свойства этих сущностей (будут представлены в виде имен атрибутов в этих блоках).

Пример

Краткая постановка задачи: главная задача системы – сбор и обработка информации об основных участниках учебного процесса: студентах и преподавателях, формирование необходимых печатных форм (документов), используемых преподавателями в период зачётной недели и экзаменационной сессии, генерация сводных отчётов по результатам сессии для работников деканатов, института. При разработке системы следует учитывать, что она основывается на документации, поступающей из приёмной комиссии, деканатов и других подразделений института. Информация об успеваемости студентов должна накапливаться и храниться в течение всего периода обучения. В системе должен использоваться справочник специальностей и дисциплин (предметов), изучаемых студентами.

Диаграмма


Студент
Номер зачетки
Фамилия, имя, отчество
Дата поступления
Факультет
Специальность
Курс
Номер группы
Успеваемость
Номер зачетки
Номер семестра
Предмет
Оценка
Дата экзамена
Фамилия препод.
Предмет
Номер предмета
Название предмета
Факультет
Номер факультета
Название фак-тета
Специальность
Код специальности
Название спец-сти
1
∞
1
∞
∞
1
∞
1

SQL

Some of The Most Important SQL Commands

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database
- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

CREATE & INSERT

CREATE DATABASE IF NOT EXISTS article;

CREATE TABLE `table1` (`user_id` INT(5) **NOT** NULL AUTO_INCREMENT, `username` VARCHAR(50), PRIMARY **KEY**(`user_id`), **INDEX**(`username`)); **CREATE TABLE** `table2` (`phone_id` INT(5) **NOT** NULL AUTO_INCREMENT, `user_id` INT(5) **NOT** NULL, phone_number INT(10) **NOT** NULL, PRIMARY **KEY** (`phone_id`), **INDEX**(`user_id`, `phone_number`)); **CREATE TABLE** `table3` (`room_id` INT(5) **NOT** NULL AUTO_INCREMENT, `phone_id` INT(5) **NOT** NULL, `room_number` INT(4) **NOT** NULL, PRIMARY **KEY**(`room_id`), **INDEX**(`phone_id`, `room_number`));

INSERT INTO table2 (user_id, phone_number) **VALUE** ('2','200');

SELECT

https://www.w3schools.com/sql/sql_select.asp

Задания

1. Выбрать все Продукты с именем саплаера и категории ProductID, ProductName, SupplierName, CategoryName

2. Сколько всего заказов?

3. Посчитать стоимость заказа 10248

Discussion

Thank you for attention!

EVRY Digital +\dvantage