**Нормализация БД**

*Нормализация БД* – это процесс преобразования БД к виду отвечающей нормальным формам.

*Нормальная форма* – НФ(NF) – Свойство отношения реляционной модели данных характеризующей его с точки зрения избыточности данных потенциально приводящей к логически ошибочным результатам выборки.

*Метод нормальных форм* – может применяться при проектировании БД и должен применяться для устранения аномалий и упрощения процедуры применения ограничения целостности.

Виды аномалий:

* Аномалия удаления – можно потерять данные не связанные с удаляемой записью. (удаление конкретной строчки удаляет и общую строчку)
* Аномалия обновления(модификации) – при необходимости изменить одно значение потребуется просмотреть все строки для изменения значения во всех ячейках.
* Аномалия вставки(включения) – нельзя вставить запись из-за нехватки инфы

Виды нормальных форм:

1. Первая нормальная форма (1НФ)
2. Вторая нормальная форма (2НФ)
3. Третья нормальная форма (3НФ)
4. 3,5 НФ или Нормальная форма Бойса-Кодда или БКНФ
5. Четвертая НФ
6. Пятая НФ
7. Шестая НФ

Нормальные формы отличаются ограничениями, которые на них накладывают:

**1 группа ограничений:**

* 1НФ, 2НФ, 3НФ – Ограничивают зависимости между ключевыми и не ключевыми атрибутами.

**2 группа ограничений:**

* Бойса-кодда – Ограничивает зависимость ключевых атрибутов.

**3 группа ограничений:**

* 4НФ – Формирует ограничение на виды многозначной зависимости.

Виды зависимостей:

1. Функциональная зависимость(А -> B) – Зная А, знаем B. (номер зачетки -> ФИО, специальность). (серия, номер паспорта -> ФИО, ДР, адрес
   1. Функциональная взаимозависимость (A <-> B) – Зная А, знаем Б, знаем Б, знаем А (© Степан). (номер зачетки <-> серия, номер паспорта) (СНИЛС – ИНН)
   2. Частичная функциональная зависимость – зависимость не ключевого атрибута от части первичного ключа.
   3. Полная функциональная зависимость – зависимость не ключевого атрибута от всего первичного ключа. (пример полной зависимости – серия, номер паспорта -> ФИО).
2. Транзитивная зависимость – (A -> B -> C | A -> B, B ->C) или (A ->> B | C) – “С” транзитивно зависит от A. (номер зачетки - > группа -> специальность) (ИНН -> серия, номер паспорта -> семейное положение).
3. Многозначная зависимость – Отношение с атрибутами A B C, есть многозначная зависимость B от A, если множество атрибутов B соответствующих паре атрибутов A и С, зависит от значения A и не зависит от значения C.

*Детерминант* – это атрибут от которого функционально зависит другие атрибуты ( A -> B, A – детерминант).

*Декомпозиция* – это совокупность отношений R1, R2, R3, … являющихся проекцией отношения R и при соединение дающее отношение R, для соединения используется общий атрибут, обычный, первичный, внешний ключ.

*Вложенность нормальных форм* – каждая нормальная форма соответствует ограничению предыдущих:



Отношения находятся в первой НФ если каждое значение является атомарным (т.е. в каждой ячейке только одно значение).

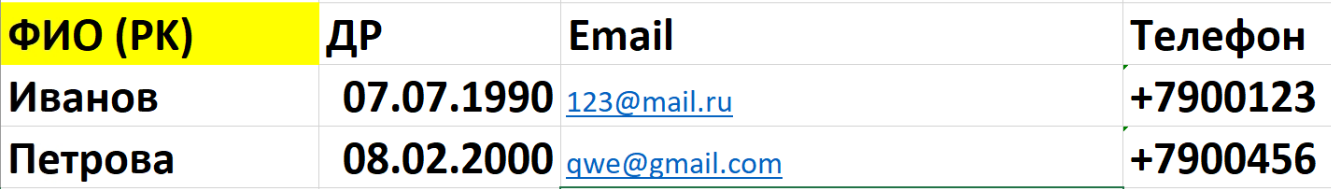
Возможны следующие варианты, нарушающие 1НФ**:**

1. В одной ячейке разнотипные данные, для приведения к 1НФ столбец делится на несколько, первичный ключ не меняется.

Было:



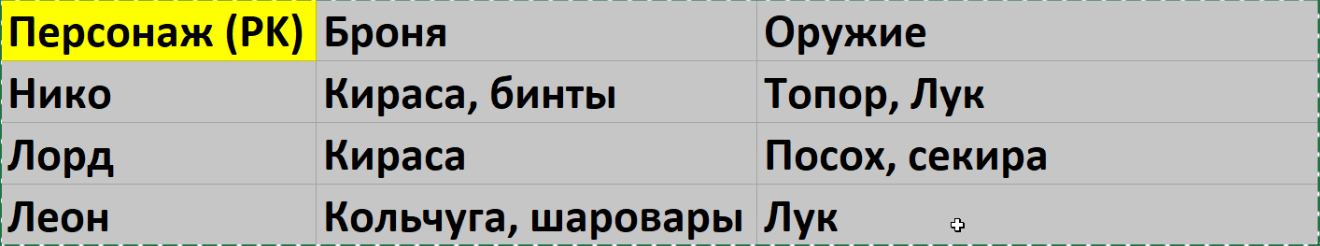
Стало:



1. В одной ячейке однотипные данные, для приведения к 1НФ атрибут в котором однотипные данные включается в первичный ключ, строка ячейки в которой много значений повторяется столько раз, сколько есть значений и в каждой ячейке вместо набора значений остается одно.

Было:

Отношение не соответствует 1НФ, т.к. в столбце “Броня” не атомарные атрибуты



Стало:

Броня теперь первичный ключ. Но оружие не находится в 1НФ, так как значение находится не в атомарной форме.



Конечный результат:

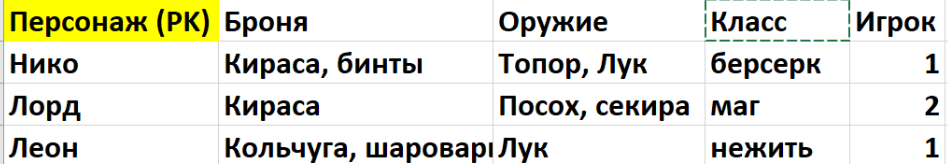
Теперь все значения входят в первичный ключ.



**2НФ** – Отношения находятся во 2НФ, если оно находится в 1НФ и *каждый не ключевой атрибут* находится в полной функциональной зависимости (т.е. зависит от *всего* PK).

Если PK состоит из одного атрибута, то отношение находится во 2НФ, если из двух и более, то требуется проверка.

Исходная таблица:



Приведение к 1НФ: R {Персонаж, Броня Оружие, Класс, Игрок}

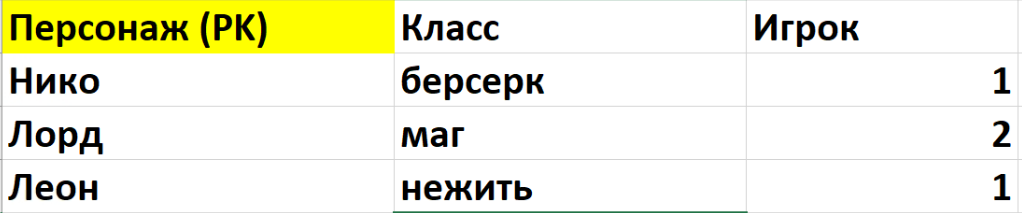


**Алгоритм приведения ко 2НФ:**

1. Найти зависимости между ключевыми и не ключевыми атрибутами.
   1. Персонаж -> Класс, Игрок
   2. Персонаж, Броня, Оружие -> (ничего нету)
2. Создать столько отношений, сколько есть детерминантов (наборов атрибутов слева от стрелки), в новых отношениях детерминанты становятся PK
3. Для связи всех отношений добавляются FK, чтобы при объединении отношений получить исходное, FK будет в составном PK.
4. Удалить дублирующийся строки (Для удаления дубликатов в Excel: Данные -> Работа с данными -> Удалить дубликаты).

**Результат приведения ко 2НФ:**

Было: R1 {Персонаж, Класс, Игрок}

****

Стало: R1 {Персонаж, Броня, Оружие}

****

**3НФ –** Отношение находится в 3НФ, если оно находится во 2НФ и *нет транзитивной зависимости между ключевыми и не ключевыми атрибутами.* (т.е. нет функциональных зависимостей между не ключевыми атрибутами). Если не ключевых атрибутов меньше двух, то отношение находится в 3НФ.

**Алгоритм приведения к 3НФ:**

1. Найти зависимости между не ключевыми атрибутами.
2. Создать столько отношений сколько есть детерминантов + одно с исходным PK, то он становится первичным ключом, из отношения с исходным PK убираются атрибуты, которые зависят от детерминанта.
3. Внешний ключ добавляется в исходное отношение.

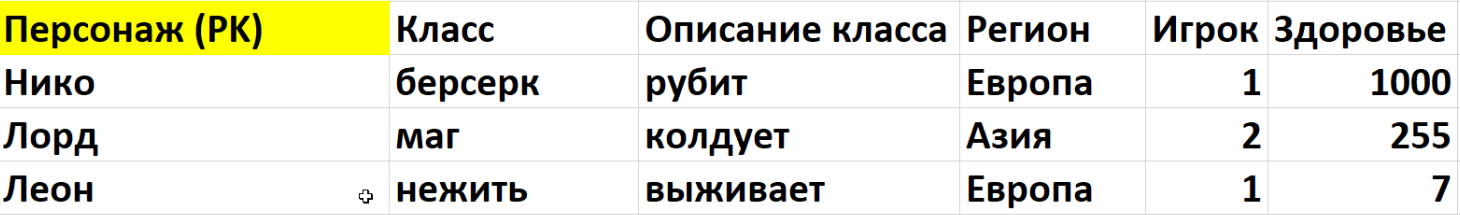
**Результат приведения к 3НФ:**

Было:

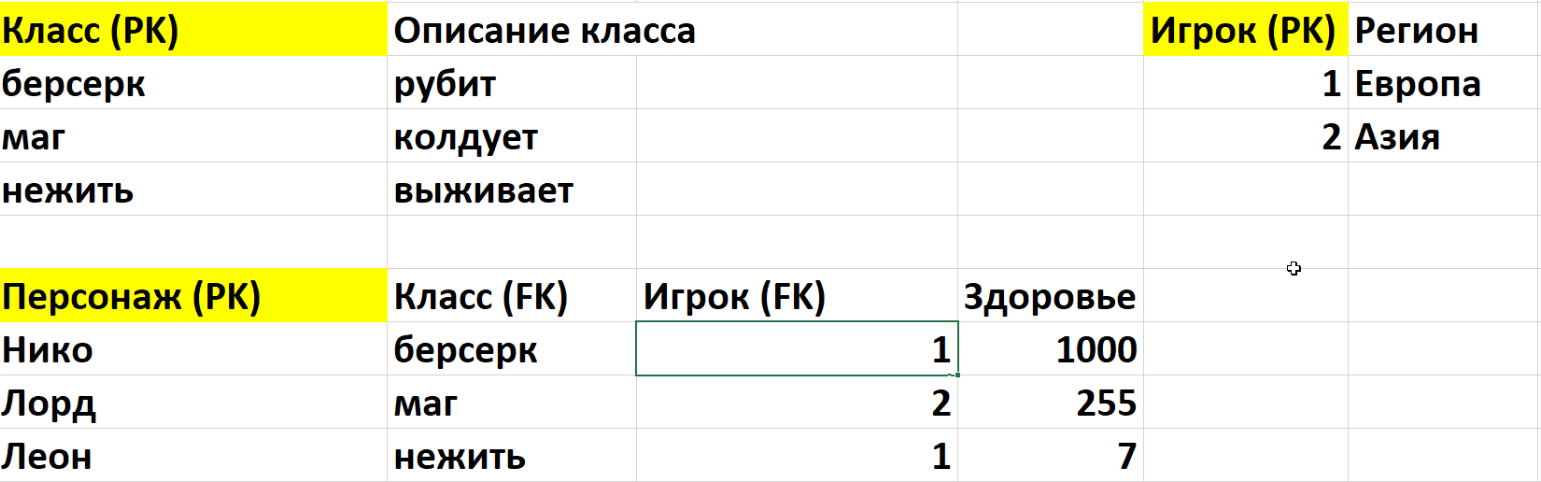
Класс -> Описание класса

Игрок -> Регион

Персонаж -> Класс, Игрок, Здоровье



Стало:



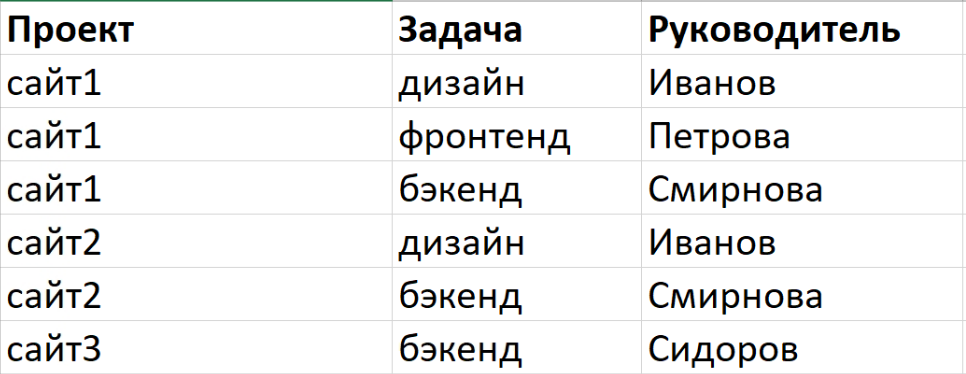
**НФ Бойса-Кодда** – Отношение находится в БКНФ, если оно находится в 3НФ и каждый детерминант является потенциальным ключом.

**Проверка на БКНФ**, возможны след. случаи:

1. В отношении два или более потенциальных ключей
2. Два потенциальных ключа являются составными
3. Два потенциальных ключа пересекаются, т.е. имеют общий набор атрибутов

Следствие: Если в отношении **НЕТ** потенциальных ключей, то оно находится в БКНФ.

Каждый руководитель отвечает только за одну деятельность.

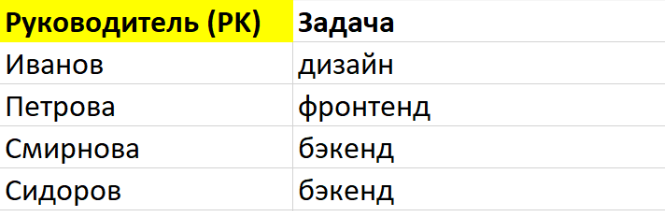


Выбираем PK: Здесь задача зависит от руководителя, т.е. PK неправильно подобран, т.к. от не ключевого атрибута зависит ключевой атрибут.

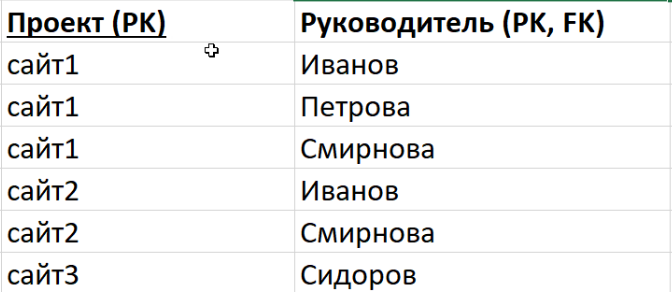


**Алгоритм перевода в БКНФ**:

1. Выносим отдельно отношение и его не ключевой атрибут и убираем дубли.



1. В исходном отношении не ключевой атрибут становится частью первичного ключа и внешним ключом.

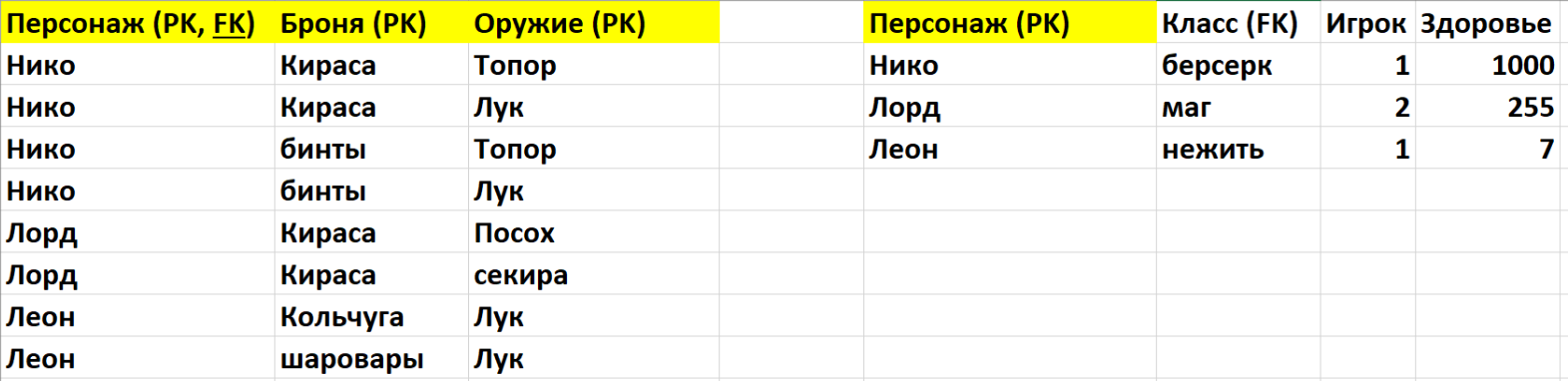


Это две таблицы ↑

**4НФ** – Отношение находится в 4НФ, когда оно находится в БКНФ и не имеет многозначных зависимостей.

Следствие: Если PK состоит из 1 и более атрибутов, то отношение находится в БКНФ.

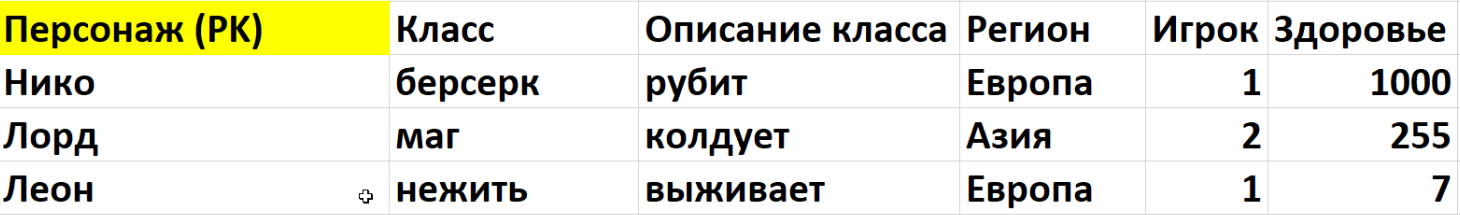
Было: R { персонаж, броня, оружие } <- Исходная таблица



Результат преобразования: Добавилось две таблицы отдельно с бронёй и оружием для персонажа. R1 { **персонаж**, **броня** }; R2 { **персонаж**, **оружие** } <- Декомпозиция.



Описание исходной таблицы:



R { **персонаж**, броня, оружие, класс, описание класса, регион, игрок, здоровье }

1. **1НФ** - Для приведения к 1НФ атрибуты броня и оружие включены в первичный ключ.

R { **персонаж**, **броня**, **оружие**, класс, описание класса, регион, игрок, здоровье } <- 1НФ

1. **2НФ** - Т.к. в исходном отношении все атрибуты зависели от атрибута персонаж, отношения не находились в 2НФ.

**Персонаж** -> класс, описание класса, регион, игрок, здоровье

Делаем декомпозицию:

R1 { **персонаж(PK)**, **броня**, **оружие** }

R2 { **персонаж**, класс, описание класса, регион, игрок, здоровье }

1. **3НФ** – Отношение R1 { **персонаж(PK)**, **броня**, **оружие** } находится в 3НФ, т.к. в нём нету не ключевых атрибутов.

Функциональные зависимости между не ключевыми атрибутами.

**класс** -> описание класса

**игрок**-> регион

Отношение R2 { **персонаж**, класс, описание класса, регион, игрок, здоровье } не находится в 3НФ, нужно провести декомпозицию:

R3 { **класс**, описание класса }

R4 { **игрок**, регион }

R5{ **персонаж**, класс, игрок, здоровье }

1. **БКНФ**:

R1 { **персонаж(PK)**, **броня**, **оружие** } <- Есть в БКНФ

R3 { **класс**, описание класса } <- Есть в БКНФ

R4{ **игрок**, регион } <- Есть в БКНФ

R5{ **персонаж**, класс, игрок, здоровье } <- Есть в БКНФ

Все они в БКНФ, т.к. нету потенциальных ключей.

1. **4НФ**:

R1 { **персонаж(PK)**, **броня**, **оружие** } <- Составной PK, проводим декомпозицию.

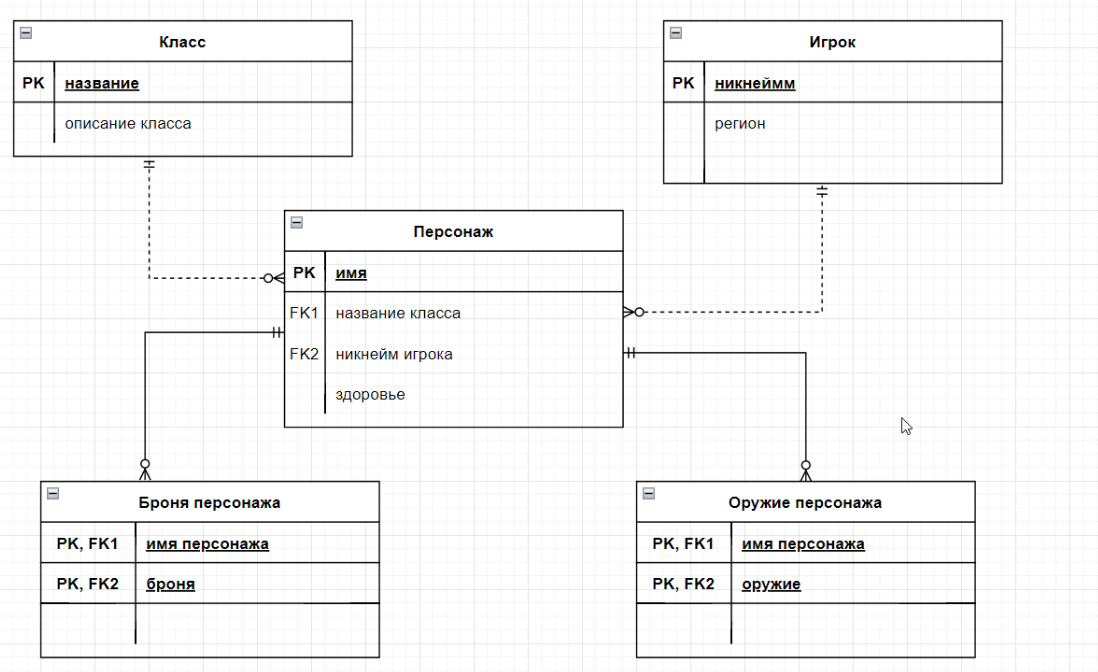
R3 { **класс**, описание класса } <- В 4НФ, т.к. PK не составной

R4{ **игрок**, регион } <- В 4НФ, т.к. PK не составной

R5{ **персонаж**, класс, игрок, здоровье } <- В 4НФ, т.к. PK не составной

R6 { **персонаж**, **броня** }

R7 { **персонаж**, **оружие**}



В броне персонажа и оружии линии между PK НЕТУ.

**Денормализация**

**Денормализация** - это процесс **осознанного** приведения БД к виду, не отвечающему НФ, проводится с целью повышения производительности, обработки данных БД и ускорения чтения данных за счет добавления избыточных данных.

Обычно можно денормализовать до 2НФ

**Способы денормализации**:

1. Внесение в родительское отношение атрибутов из дочернего.
2. Внесение в отношение вычисляемых полей ( цена, НДС, цена с НДС)
3. Вынесение в отдельную таблицу редко изменяемых атрибутов.
4. Внесение в отдельную таблицу атрибутов с объемными данными, т.е. разбиение объемной таблицы на несколько новых со связью 1:1.

**Физическое проектирование**

**Физическое проектирование** - это создание схемы БД для конкретной СУБД с учетом ее специфики.

**Специфика включает в себя**:

1. Ограничение на наименование БД и ее объектов
2. Ограничение на поддерживаемые типы данных
3. Создание индексов
4. Обеспечение ограничения целостности
5. Выбор решений, связанных с физическим хранением данных

**Этапы физического проектирования БД:**

1. Создание таблиц и столбцов таблиц
2. Добавление PK в таблицу
3. Создание связей и FK в таблицу
4. Настройка ограничений целостности (CHECK или CK), то есть условия для проверки корректности данных
5. Указание значений по умолчанию (DEFAULT или DF) (AUTOINCREMENT или AI)

**Рекомендации по физическому проектированию БД:**

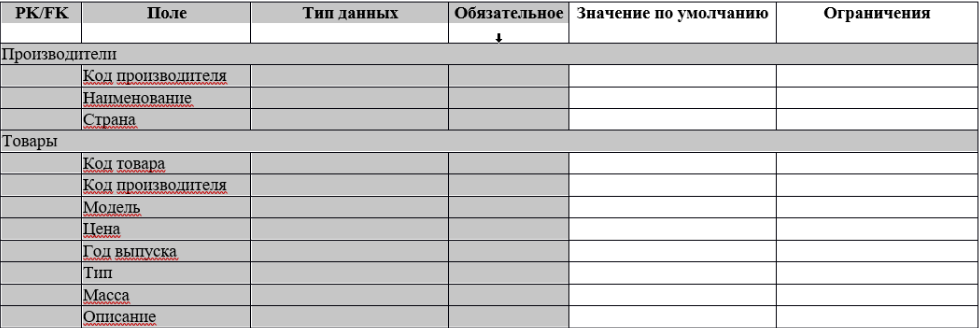
1. Вместо строковых ключей, следует использовать числовые
2. Потенциальные строковые ключи следует сделать уникальным индексом (UNIQUE или UQ)
3. Суррогатные числовые ключи лучше делать автоинкрементными, они позволяют уменьшить количество хранящейся в БД информации, если они используются в других таблицах как FK

**Способы представления физической модели БД:**

1. ERD с указанием типов данных, желательно что бы названия были такие же как в БД
2. Data Dictionary – Обычно представляется в виде таблицы описывающей БД, каждая таблица может быть описана отдельной таблицей в документе или все таблицы могут быть описаны в одной таблице. Если данные в одной таблице, то в строке объединяют ячейки и пишут название таблицы

**Пример списка столбцов словаря данных:**

1. Указываем PK и FK
2. Указываем поле (имя атрибута или столбца)
3. Указываем тип данных
4. Обязательное или Not Null (+/- или Да/Нет)
5. Значение по умолчанию (AI или DF (1.5, “ИСПП-21”, GetDate(), Now(), Year(GetDate())))
6. Ограничение целостности (> 10000, от 10000 до 50000, ‘09.02.07’ или '09.02.01') (UQ +/-)
7. Примечание или комментарии, дополнительная информация



**Особенности SQL:**

1. Регистронезависимый (id=ID, a=A)
2. Из-за стандартизации типы данных в СУБД примерно одинаковые
3. MSSQL: CamelCase / PascalCase (idGame)
4. MySQL: snake\_case (id\_game)

**Типы данных SQL**

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип данных** | **MS SQL Server** | **MySQL** | **Примечание** |
| Строка переменной длины | **varchar**(длина)  **nvarchar**(длина) – поддержка русского и др. языков  Если нужна длина  > 4000, то пишется MAX (nvarchar(MAX)) | **varchar**(длина) | Подходит для хранения строки, длиной не больше указанной. Если длина строки больше, строка обрезается справа |
| Строка фиксированной длины | **char**(длина)  **nchar**(длина) – поддержка русского и др. языков | **char**(длина) | Подходит, если все хранимые значения одинаковой длины.  Если длина значения меньше, дополняется пробелами |
| Перечисление | *Отсутствует.*  *Реализуется при создании ограничений целостности* | **еnum**(‘X’, ‘Y’, ‘Z’, …) | Позволяет указать список допустимых строковых значений |
| Логический | **bit** | **bit** | True/False или 1/0 |
| Байт (целое) | **tinyint** | **tinyint** | 0-255 |
| Короткое целое | **smallint** | **smallint** | -32768/+32767 |
| Целое | **int** | **int** | +/-2^31 (+/- 2 млрд) |
| Длинное целое | **bigint** | **bigint** | +/-2^63 |
| Вещественное | **decimal**(всего цифр, вещественная часть) | **decimal**(всего цифр, вещественная часть) | Пример: Для **decimal**(5,2) максимум 999,99(всего 5 цифр, 2 – после запятой) |
| Денежный | **money**  Допустимо: **decimal**(n,2) | **decimal**(n,2) | Подходит для хранения денежных значений |
| Время | **time** | **time** | Формат: ‘чч:мм:сс’ |
| Дата | **date** | **date** | Формат: ‘ГГГГ-ММ-ДД’ |
| Дата и время | **datetime** | **datetime** | Формат: ‘ГГГГ-ММ-ДД чч:мм:сс’ |
| Двоичные данные | **varbinary**(n) – до 8000 байт n – кол-во байт  **varbinary**(MAX) – до 2ГБ | **BLOB**  **TEXT** | Позволяет хранить файлы в БД  Недостаток: размер БД увеличится  Альтернатива: хранить в БД пути или ссылки на файлы  Недостаток: могут указывать на отсутствующий файл |

**Пример словаря данных:**

