**Основы**

Просто хранить данные в файлах неудобно:

* Полное отсутствие гибкости (изменение структуры – изменение кода программы).
* Невозможность нормальной многопользовательской работы с данными.
* Вынужденная избыточность (проще создать еще одну копию данных, чем вносить изменения в десятки программ).

Решение — хранить данные и метаданные (данные о данных) вместе.

База данных — это файлы, снабжённые описанием хранимых в них данных и находящиеся под управлением специальных программных комплексов, называемых СУБД

Способы доступа к БД

* Файл-серверные – данные хранятся на сервере, а СУБД на каждом клиентском компьютере.
* Клиент-серверные – СУБД и данные хранятся на сервере.
* Встраиваемые – СУБД в приложении хранит только его данные и не требует отдельной установки.

Модели данных

* Реляционные - Реляционная база данных хранит данные в таблицах. Таблица состоит из строк и столбцов. Где, строки представляют каждую сущность, в то время как столбцы представляют атрибуты.
* Не реляционные - не используется табличная схема строк и столбцов. В этих базах данных применяется модель хранения, оптимизированная под конкретные требования типа хранимых данных.

Реляционные БД

* Данные хранятся в таблицах
* Каждая таблица состоит из строк, имеющих одну и туже структуру, и имеет уникальное имя.
* Строки имеют фиксированное число полей (столбцов) и значений
* В каждой позиции таблицы всегда имеется в точности одно значение или ничего.
* Строки таблицы отличаются друг от друга хотя бы по одному значению
* Столбцам таблицы присваиваются имена
* В каждом из столбцов размещаются однородные значения данных
* Содержание базы данных представляется в виде явных значений данных. (не существует каких-либо специальных "связей" или указателей, соединяющих одну таблицу с другой).
* При выполнении операций с таблицей ее строки и столбцы можно обрабатывать в любом порядке безотносительно к их содержанию.
* Таблицы базы данных часто группируют в схемы

Изображение выглядит как стол

Автоматически созданное описание

Ключ – минимальный набор атрибутов, по которым можно найти экземпляр сущности.

Внешний ключ - множество атрибутов сущности для организации ссылок на экземпляры другой сущности (должен соответствовать первичному ключу в другой сущности).

Целостность данных

* По сущностям – недопустимо, чтоб какой-то атрибут, участвовавший в первичном ключе, принимал неопределенное значение.
* По ссылкам – внешний ключ должен быть или быть равным значению первичного ключа ассоциируемой сущности или полностью неопределенным
* Пользовательская – уникальность, диапазон значений.

**Архитектура ANSI-SPARC**

Основная цель СУБД – предложить пользователю абстрактоное представление данных, скрыв конкретные особенности хранения и управления ими.

ANSI -SPARK – определяет принцип организации СУБД. Основная цель – отделение пользовательского представления БД от её физического представления. Введение уровней абстракций для пользователей.

Зачем это нада?

* Пользователи должны быть абстрагированы от физической и логической организации данных.
* Разные группы пользователей БД должны обладать разными представлениями и знаниями о ней.
* Администраторы БД должны иметь возможность менять схему данных и не переживать о целостности данных особенности её архитектуры итд.
* Внутренняя структура бд не должна зависеть от аппаратной и программной части.

Уровни абстракции

1. Внешний – уровень представления с которым работают пользователи БД.
2. Концептуальный – логический уровень бд (Отношения, таблицы, атрибуты, типы данных, ключи, ограничение целостности).
3. Внутренний – аспекты физ. Организации

Логическая независимость – защищенность внешних схем от изменений, вносимых в концептуальную схему. Сохранение представления внешнего уровня при изменении схемы данных.

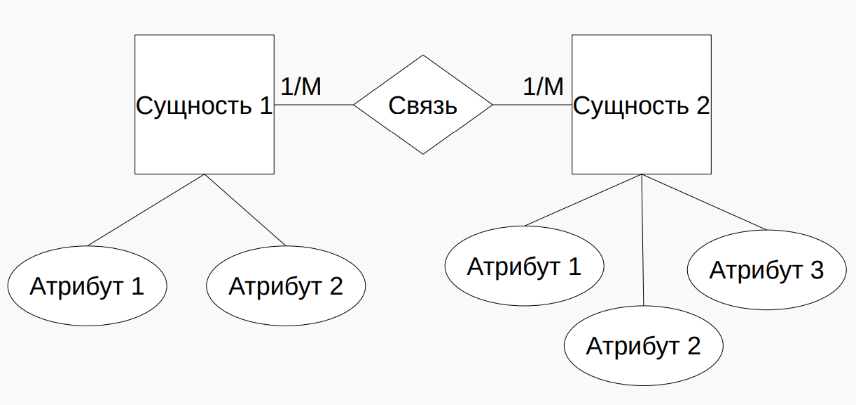
Физическая независимость – защищенность концептуальной схемы от изменений, вносимых во внутреннюю схему. Файловые системы и системные функции организация хранения данных.

**Модель "Сущность-Связь". Классификация сущностей. Виды связей. Ограничения целостности.**

* Инфологическая модель – Имеет обобщенное представление предметной области, собирается на основе анализа пользовательских представлений, не зависит от физ. Аспектов бд.
* Даталогическая модель – инфологическая модель + модель данных + детали СУБД.

Один из вариантов для построения инфологической модели: ER-диаграммы (Entity-Relationship - сущность-связь).

Основные элементы

* Сущность — класс объектов, фактов, явлений, предметов, элементы которых будут храниться в базе данных.
* Экземпляр сущности относится к конкретной вещи в наборе. Например, типом сущности (сущностью) может быть СТУДЕНТ , а экземпляром — Иван Иванов и т. д.
* Атрибут — важная характеристика (свойство) сущности, которой присваивается имя.
* Связь — ассоциирование двух или более сущностей, выражающая форму взаимодействия между ними.

Виды связи между сущностями

* Один к одному - В каждый момент времени каждому экземпляру первой сущности соответствует 1 или 0 экземпляров второй сущности. Группа – староста
* Один ко многим - Одному экземпляру первой сущности соответствуют 0, 1 или несколько экземпляров второй сущности. Группа – студены
* Многие к одному - Обратная связь к 1:М. Студенты – группа.
* Многие ко многим - Одному экземпляру первой сущности соответствуют 0, 1 или несколько экземпляров второй сущности; Одному экземпляру второй сущности соответствует 0, 1 или несколько экземпляров первой сущности. Студент – Экзамен.

Сложные связи – когда между сущностями может быть несколько связей.

Тернарная связь - отражает связи, охватывающие более двух множеств сущностей. Актер – фильм – студия -> контракт.

Рекурсивная связь - Рекурсивная связь – связь, в которой один и тот же тип сущности участвует более одного раза в разных ролях.

Классификация сущностей

* Стержневая – базовая сущность
* Ассоциативная сущность – связь вида N ко многим, между двумя и более сущностями.
* Характеристическая сущность – связь вида 1 к N нужна для описания или уточнения некоторой сущности.
* Сущность сильного тип– тип сущности, существование которого не зависит от какого-то иного типа сущности.
* Сущность слабого типа – тип сущности/существование которого зависит от какого-то другого типа сущности.
* Тип сущности (entity type) – группа объектов с одинаковыми свойствами, которая рассматривается в конкретной предметной области как имеющая независимое существование.

Ключ — минимальный набор атрибутов, по значениям которых однозначно определяется требуемый экземпляр сущности.

Суррогатный ключ - автоматически сгенерированный атрибут для однозначной идентификации экземпляра сущности в рамках данной сущности.

**SQL основы**

Объекты базы данных - таблицы, представления, процедуры, триггеры. Для работы с объектами БД используется SQL.

SQL — это язык структурированных запросов (Structured Query Language), позволяющий хранить, манипулировать и извлекать данные из реляционных баз данных.

* Предложения - команда, которая состоит из ключевых и зарезервированных слов, определяемых пользователем в соответствии с синтаксическими правилами SQL
* Идентификаторы (имена) – Создаются при определении объекта. Нужны для обращений. SELECT \* FROM **STUDENTS**;
* Константы - Любые значения, которые не являются идентификаторами или ключевыми словами. Числа, строки, время, булевы значения.
* Операторы - Служат для обозначения действия над одним или несколькими выражениями. Делятся на категории: арифметические, логические, присваивания, сравнения и тд. > =
* Зарезервированные слова - Слова и фразы для задания конструкций и использования возможностей языка SQL. **SELECT** **\* FROM** STUDENTS **WHERE** AGE > 19;

**DDL**

Предложения определения данных

CREATE создает объект БД (саму базу, таблицу, представление, пользователя)

* CREATE TABLE STUDENTS ( STUD\_ID integer, STUD\_NAME text, BIRTH\_DATE date );

ALTER изменяет объект

* ALTER TABLE STUDENTS ADD COLUMN STUD\_GROUP text;
* ALTER TABLE STUDENTS DROP COLUMN BIRTH\_DATE;

DROP удаляет объект.

* DROP TABLE STUDENTS;

Типы данных

* Числовые типы
  + smallint – 2 байта
  + integer – 4 байта
  + bigint – 8 байт
* Символьные типы
  + varchar(n) – ограничение сверху
  + char(n) – фиксированная длинна (пробелы)
  + text – без ограничений
* Логический тип
  + Состояния true / false / unknown – через null
  + true – TRUE/true/t/yes/y/on/1
  + false – FALSE/false/f/no/n/off/0
* Временной тип
  + time – время суток (00:00:00 — 828:59:59)
  + timestamp – дата и время (1970-01-01 00:00:00 — 2038-12-31 00:00:00)
  + date – дата (1000-01-01 — 9999-12-31)
  + interval – временной интервал
* NULL – спец. значение (несуществующее), которое может быть записано в поле таблицы бд.
  + Значения типа NULL не равны друг другу
  + Столбец, содержащий значение NULL , игнорируется при вычислениях агрегатных значений
  + В предложениях с DISTINCT, ORDER BY, GROUP BY, значения NULL не отличаются друг от друга

Ограничения целостности

* Мы можем задавать значение по умолчанию. GR\_COUNT integer **DEFAULT** 0
* **CHECK** - Позволяет задать для определённой колонки, выражение, которое будет осуществлять проверку, помещаемого в эту колонку значения. Выражение должно возвращать логическое значение. Все ок если true/null.

FAILED\_COURSES integer CHECK (FAILED\_COURSES >= 0)

* Колонка не должна содержать значение null. ST\_ID integer **NOT NULL**
* Данные в колонке или группе колонок являются уникальными по отношению к данным из той же колонки/группы. ST\_ID integer NOT NULL **UNIQUE**
* Первичный ключ означает, что колонка (группа колонок) используются как уникальный идентификатор строки в таблице. Ограничение первичного ключа эквивалентно комбинации ограничений уникальности и не-null. Таблица может иметь не более одного первичного ключа. ST\_ID integer **PRIMARY KEY**

Внешний ключ

Мы можем создать внешний ключ таблицы при помощи слова REFERENCES. GR\_ID integer REFERENCES GROUP, где GROUP – название таблицы, откуда мы вытаскиваем PRIMARY\_KEY.

Однако при удалении одной записи, мы потеряем связь, и мы не сможем получить запись из другой таблицы. Варианты:

1. Ничего не делать.
2. Удалить каскадно. GR\_ID integer REFERENCES GROUP ON DELETE **CASCADE**
3. Установить в null. GR\_ID integer REFERENCES GROUP ON DELETE **SET NULL**
4. Установить значение по умолчанию. GR\_ID integer REFERENCES GROUP ON DELETE **SET DEFAULT**

**DML**

Предложения манипуляции данными.