Карагандинский технический университет
Кафедра ИВС

Дисциплина «Проектирование и разработка приложений БД»

Тема: «Инфологический этап: сущности, атрибуты, связи — ER-моделирование»

ОП «Информационные системы», «IT - медицина»

Авторы: ст.преп.каф.ИВС Саданова Б.М.
ст.преп.каф.ИВС Олейникова А.В.

Инфологический этап: Сущности, Атрибуты, Связи — ER-моделирование

Концептуальное проектирование - сбор, анализ и редактирование требований к данным. По окончании данного этапа получаем концептуальную модель, инвариантную к структуре базы данных. Часто она представляется в виде инфологической модели "сущность-связь".

Инфологическая модель предметной области представляет собой описание структуры и динамики предметной области, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД.

Что такое инфологическое моделирование и его роль в проектировании БД

Инфологическое моделирование — это этап проектирования, фокусированный на семантике предметной области: какие "вещи" существуют, какие у них свойства, и как они связаны друг с другом. Оно отвечает на вопрос «что» хранить, а не «как» хранить.

Роль в жизненном цикле ПО

Предшествует логическому и физическому моделированию базы данных; служит мостом между предметными экспертами и архитекторами. Результат — ERD и словарь данных, которые становятся спецификацией для реализации.

Ключевые цели

Выявить сущности и атрибуты, формализовать правила целостности данных, описать связи и ограничениями (cardinality, optionality), подготовить модель для нормализации.

Участники

Аналитики, предметные эксперты (SME), дата-архитектор, разработчики и тестировщики — совместная работа обеспечивает полноту и точность модели.

Инфологическая модель удобна для обсуждения с бизнес-стейкхолдерами: она выражена на языке предметной области и не перегружена деталями реализации.

Сущности: определение и методы идентификации

Сущность — это объект или понятие предметной области, о котором необходимо хранить данные (например, Клиент, Заказ, Товар). Сущности описывают «сущностные множества» — набор однородных экземпляров (экземпляр = запись).

Клиент

Человек или организация, для которой ведётся учёт. Содержит контактные и идентификационные атрибуты.

Заказ

Транзакция, отражающая покупку. Обычно привязан к клиенту и содержит дату, статус и суммы.

Товар

Предмет торговли с кодом, ценой и характеристиками. Часто используются для учёта запасов и стоимости.

Как идентифицировать сущности: ищите существительные в описании процесса, события, о которых требуется хранить историю, и объекты, обладающие набором атрибутов и независимо существующие в контексте предметной области.

Атрибуты: виды, свойства и примеры

Атрибут — это отдельная характеристика сущности (например, имя, дата рождения, цена). В инфологической модели атрибуты описывают смысл и ограничения (обязательность, множественность, тип).

По характеру

- Простые (атомарные) — одно значение: `email`
- Составные — состоят из частей: `адрес = (улица, город, почтовый код)`
- Мультизначные — могут иметь несколько значений: телефон(ы)

По роли и свойствам

- Домен/тип: строка, число, дата, булево
- Обязательность: обязательный / опциональный
- По умолчанию и значения по умолчанию
- Уникальность и индексируемость

Документация атрибутов

Для каждого атрибута фиксируем: наименование, тип, ограничения, источник, примечания для бизнес-правил.

В инфологической модели указывайте семантику атрибута: что означает значение, единицы измерения, формат (ISO-дата, валютный код) и допустимые диапазоны — это облегчает валидацию на последующих этапах.

ключевые выводы и дальнейшие шаги

ER-моделирование — базовый инструмент, который превращает бизнес-знания в формальную структуру данных: определение сущностей, атрибутов и связей даёт ясную спецификацию для логического и физического проектирования.

Ключевые выводы

- Ищите сущности через анализ сущностных существительных и бизнес-процессов.
- Атрибуты документируйте с типами и ограничениями — это снижает риск недопонимания.
- Ключи и кардинальности — основа целостности; фиксируйте их на инфологическом уровне.
- M:N связи реализуйте через ассоциативные сущности с явными атрибутами.

Рекомендуемые следующие шаги

1. Провести воркшоп с SME для проверки сущностей и кардинальностей.
2. Составить словарь данных с примерами значений и источниками.
3. Перейти к нормализации модели и подготовке логической схемы.
4. Версионировать ERD и подключить к процессу тестирования и разработки.

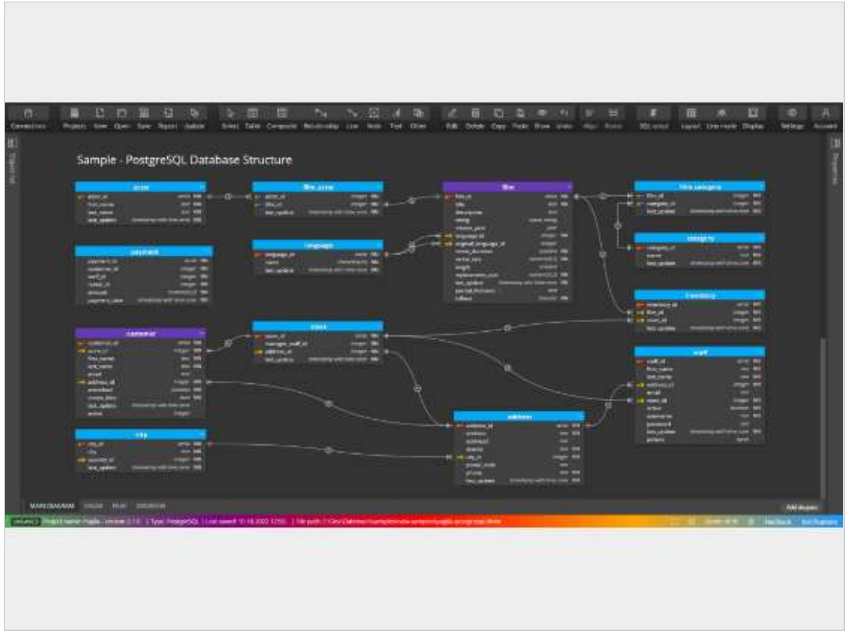
Совет: используйте единую нотацию на всем проекте и храните ER-диаграммы вместе с требованиями — это значительно ускорит онбординг новых участников команды.

Реляционная модель: парадигма

Реляционная модель данных — это логическая модель, основанная на математическом понятии отношений (теория множеств). В этой модели данные представлены в виде набора взаимосвязанных таблиц.

Ключевые преимущества:

- **Логическая независимость:** Изменение структуры хранения не влияет на логическое представление данных пользователю.
- **Ясные правила целостности:** Наличие встроенных механизмов контроля уникальности и связей.
- **Гибкость запросов:** Использование языка SQL для манипуляции данными любой сложности без привязки к путям доступа.



Базис модели: отношение — это таблица; кортеж — это строка; атрибут — это столбец. Каждое значение в ячейке должно быть атомарным.

Трансформация сущностей в таблицы

Методика перехода

На этапе даталогического проектирования каждая **сущность** превращается в отдельную базовую таблицу (отношение). Свойства этой сущности становятся именованными **атрибутами** (столбцами).

Важные шаги проектировщика:

- Определение типов данных для каждого атрибута (целое, строка, дата).
- Установка ограничений обязательности (NOT NULL) для критически важных данных.
- Разбиение составных атрибутов (например, "ФИО") на отдельные поля для удобства поиска и сортировки.

Элемент ER-модели	Физический объект в СУБД
Сущность (Товар)	Отдельная таблица products
Простой атрибут	Колонка в таблице (соответствующий тип)
Связь 1:M (Категория -> Товары)	Внешний ключ (FK) на стороне «Многие»
Связь M:M (Товар <-> Заказ)	Связующая таблица с двумя FK
Пример реализации (SQL)	
CREATE TABLE products (id PK, name, price);	
price DECIMAL(10, 2) NOT NULL	
category_id INT REFERENCES categories(id)	
CREATE TABLE order_items (order_id, product_id)	

Трансформация требует строгого соблюдения соответствия: экземпляр сущности в ER-модели всегда превращается в кортеж (запись) в реляционной таблице.

Понятие и структура схемы отношений

Техническая спецификация

Схема отношений — это совокупность заголовков отношений, входящих в состав базы данных. Она является метаданными, фиксирующими структуру до заполнения её данными.

Обязательные элементы схемы:

- Имена таблиц и уникальные имена полей.
- Домены (множества допустимых значений) для каждого столбца.
- Описание первичных и внешних ключей.
- Ограничения CHECK (например, возраст > 18).

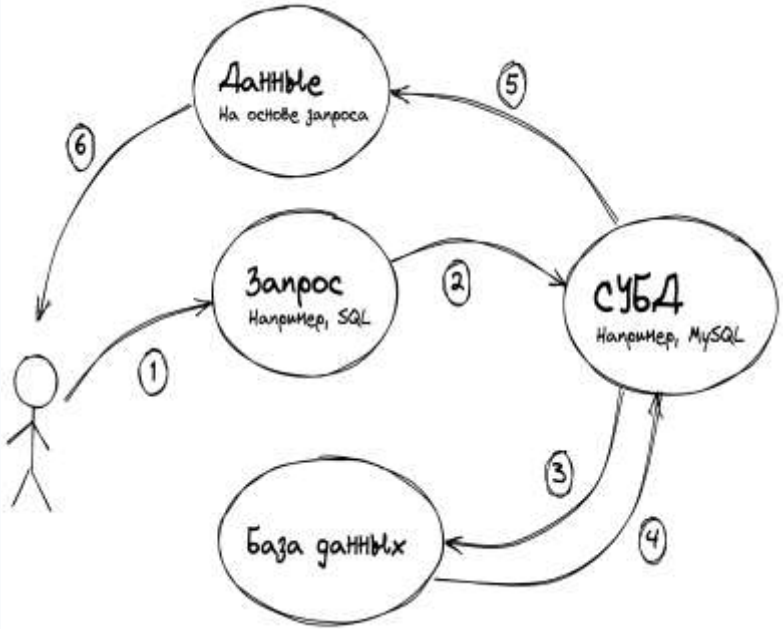


Схема служит мостом между логическим проектированием и физической реализацией на

Первичные ключи: идентификация

Первичный ключ (Primary Key) — это минимальный набор атрибутов, значения которых никогда не повторяются в рамках одной таблицы и не могут быть пустыми (NULL).

Виды ключей:

- **Естественные:** Имеют смысл в предметной области (ИНН, номер паспорта, SKU).
- **Суррогатные:** Искусственно созданные идентификаторы (ID, UUID, Auto-increment), не несущие смысловой нагрузки.

Совет: Предпочтительнее использовать суррогатные ключи, так как они стабильнее и не меняются при изменении бизнес-логики.

⚠Требования к PK:

- **Уникальность:** Исключает дублирование записей.
- **Неизменность:** Ключ не должен обновляться в течение жизни записи.
- **Минимальность:** Не должен содержать лишних полей, если достаточно одного.

Внешние ключи и связи таблиц

Внешний ключ (Foreign Key) — это атрибут в дочерней таблице, значения которого ссылаются на первичный ключ в родительской таблице. Это основной механизм связывания данных.

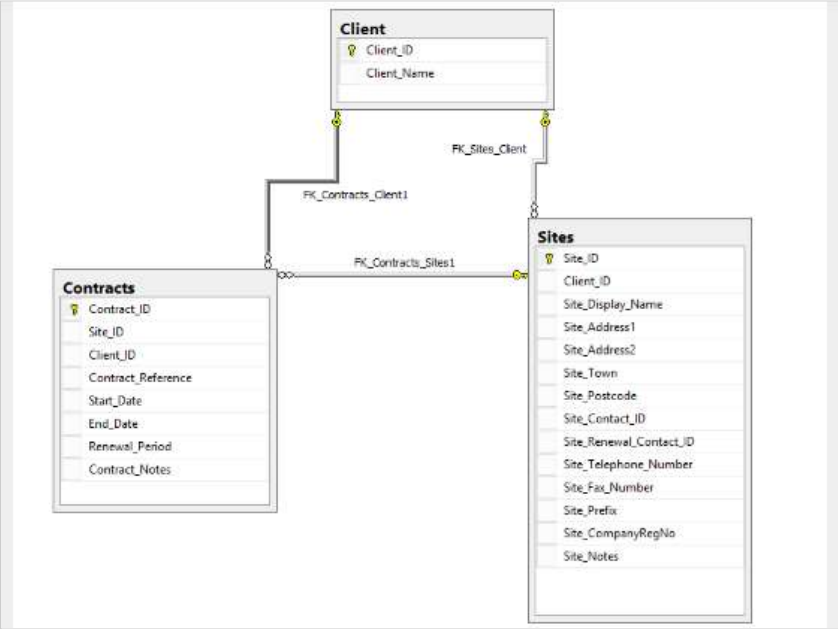
Ссылочная целостность:

FK предотвращает появление "битых" ссылок. СУБД контролирует операции удаления и обновления:

CASCADE: Удаление родителя удаляет всех детей.

SET NULL: При удалении родителя ссылки в дочерних записях обнуляются.

RESTRICT: Запрет удаления родителя, пока есть связанные дети.



Внешние ключи позволяют "собирать" данные из разных таблиц в единый отчет с помощью операции JOIN.

Классификация связей

Один к одному (1:1)

Каждая запись в табл. A соответствует строго одной записи в табл. B. Используется для разделения больших таблиц или хранения конфиденциальных данных отдельно.

Один ко многим (1:N)

Запись в A может иметь много связей в B. Пример: "Один Департамент — Много Сотрудников". FK всегда ставится на стороне "Многих" (в таблицу сотрудников).

Многие ко многим (M:N)

Много A связано со многими B. Пример: "Студенты и Курсы". Реализуется через третью промежуточную таблицу (таблица связки), содержащую FK обеих сторон.

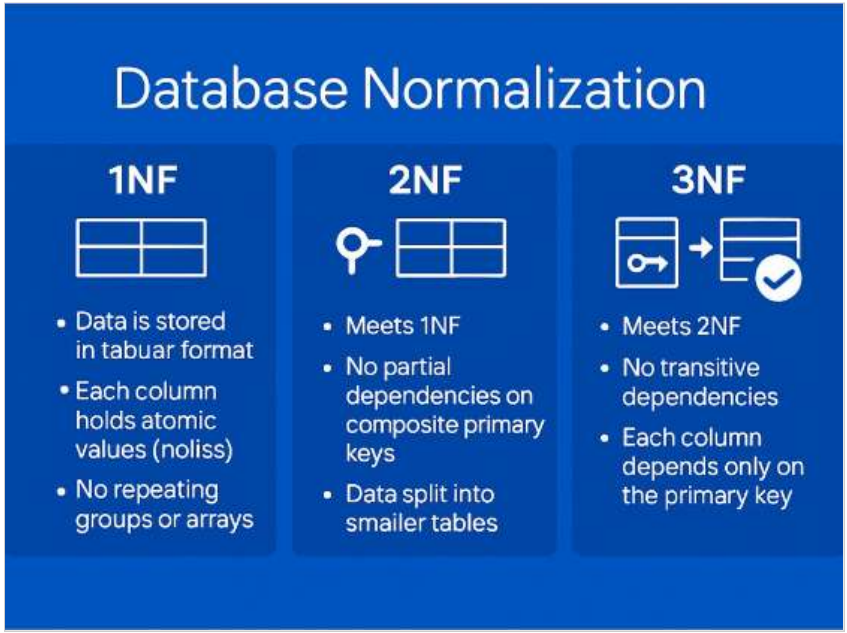
Теория нормализации данных

Нормализация — метод проектирования, сводящий к минимуму дублирование данных и защищающий от аномалий (ошибок при вставке, удалении или обновлении).

1NF (Первая): Все атрибуты атомарны. Нет повторяющихся групп полей.

2NF (Вторая): Выполнена 1NF + все неключевые атрибуты зависят от всего составного ключа целиком.

3NF (Третья): Выполнена 2NF + неключевые атрибуты зависят только от ключа, а не друг от друга (нет транзитивности).



Важно: Чрезмерная нормализация может замедлить работу БД из-за обилия JOIN. Оптимальный уровень для большинства систем — 3NF.

Физический этап проектирования баз данных: хранение, индексы, кластеризация

Физический этап объясняет, как данные организованы на диске, какие структуры ускоряют чтение и запись, и какие архитектурные решения помогают масштабировать и поддерживать производительность в реальных системах.

Физическая модель содержит полную информацию, необходимую для генерации объектов базы данных.

Физический проект описывает базовые отношения, определяет организацию файлов и состав индексов для обеспечения эффективного доступа к данным, а также регламентирует все ограничения целостности данных и меры защиты данных.

Хранение данных: основные принципы и структуры

БД хранит данные в файлах, разбитых на страницы/блоки фиксированного размера. Основные принципы: экономия IO за счёт чтения/записи страниц, локальность данных для кэширования и согласованность через журнал. Формат записи (row-oriented vs column-oriented) выбирается в зависимости от рабочей нагрузки.

Строковый (Row)

Оптимален для OLTP: быстрые вставки/обновления одной строки.

Колонночный (Column)

Оптимален для аналитики: эффективное сжатие и выбор отдельных столбцов.

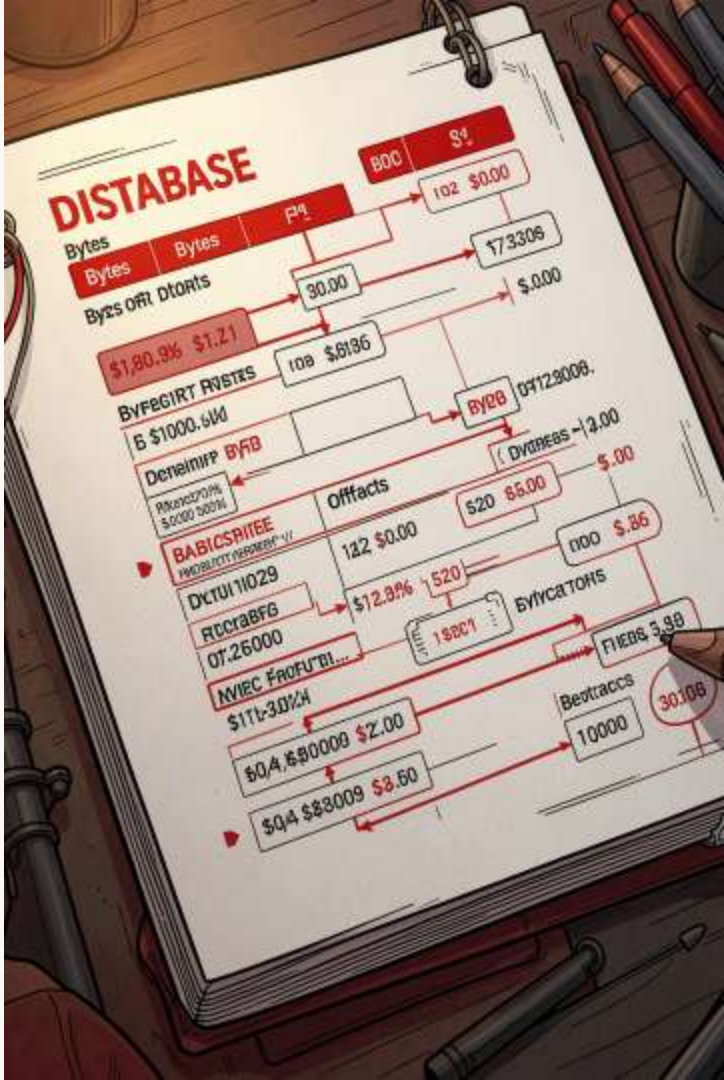
Файловая организация

Однофайловая vs многофайловая — влияет на управление диском и parallel I/O.

Страницы и блоки: как данные организуются физически

Данные группируются в страницы (обычно 4–16 КБ). Страница содержит заголовок, записи и свободное пространство. Поведенческие последствия: фрагментация, необходимость compaction/garbage collection и влияние на производительность случайных I/O.

Диаграмма показывает: при записи новой строки может понадобиться split/overflow страницы; при обновлении — в-place или copy-on-write; при активных транзакциях ключевую роль играет управление свободным пространством и порядок flush.



Индексы: ускорение доступа к данным — зачем они нужны?

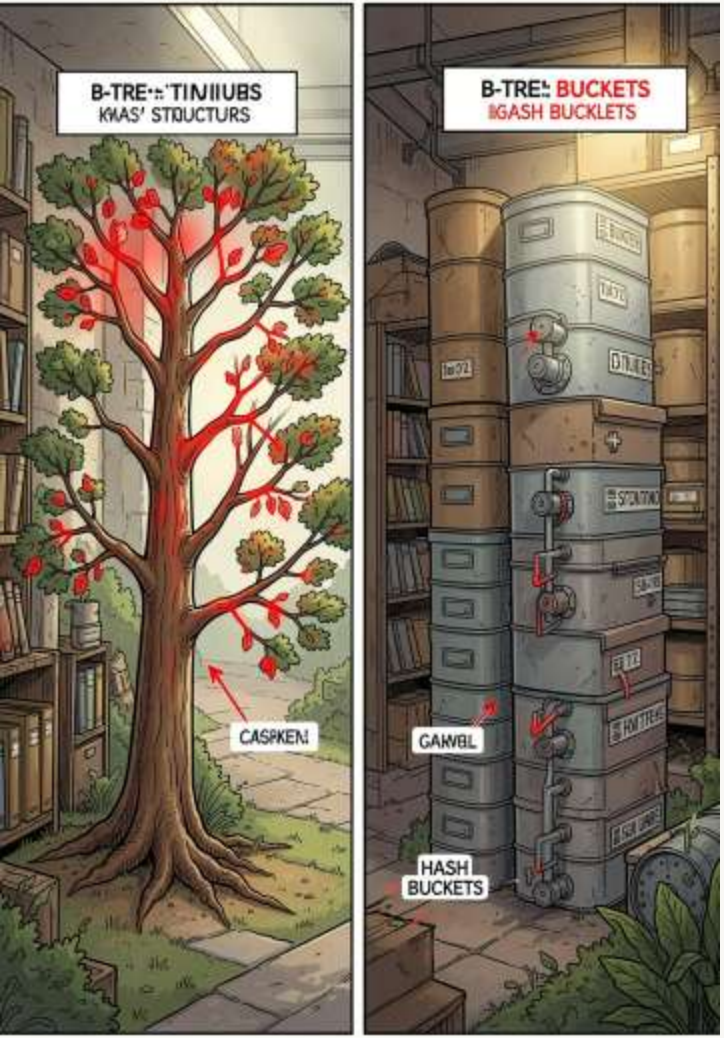
Индексы уменьшают количество считываемых страниц, превращая потенциальный full table scan в быстрый поиск. Они повышают производительность чтения, но стоят дополнительным местом и замедляют записи (insert/update/delete). Балансировка — ключевой аспект.

Плюсы

Быстрое выполнение точечных и диапазонных запросов, улучшенная сортировка и выборка по условиям WHERE/ORDER BY.

Минусы

Дополнительное пространство, накладные расходы при обновлениях, риск неэффективного использования при плохом селективности.



Типы индексов: B-деревья, хеш-индексы и другие

Короткий обзор распространённых типов: B-деревья (B+tree) — универсальны, поддерживают диапазонные запросы; хеш-индексы — быстры для точечных запросов, но не для диапазонов; GiST/GIN — для полнотекстового поиска и сложных типов данных; bitmap-индексы — эффективны при низкой кардинальности.

B-дерево / B+tree

Индексация с многократно сбалансированными узлами, эффективны при больших наборах и диапазонных запросах.

Хеш

Константное время для точечных выборок, чувствительны к коллизиям, не подходят для ORDER BY по ключу.

GIN / GiST

Индексы для массивов, JSONB, полнотекстового поиска и пространственных данных.