

Лекция 9_1

Трехуровневая модель данных. Проектирование базы данных

Три уровня данных

Трёхуровневая модель ANSI/SPARC

Трёхуровневая модель ANSI/SPARC была предложена в 1970-х годах для решения проблемы зависимости приложений от изменений в структуре и хранении данных.

Основная проблема:

Ранние СУБД были жёстко связаны с физическим представлением данных — любое изменение в способе хранения (например, формат файла или индекс) требовало переписывания всех приложений, работающих с БД.

Трехуровневая модель ANSI/SPARC

Цель ANSI/SPARC:

Обеспечить **логическую и физическую независимость данных** за счёт разделения системы на три уровня:

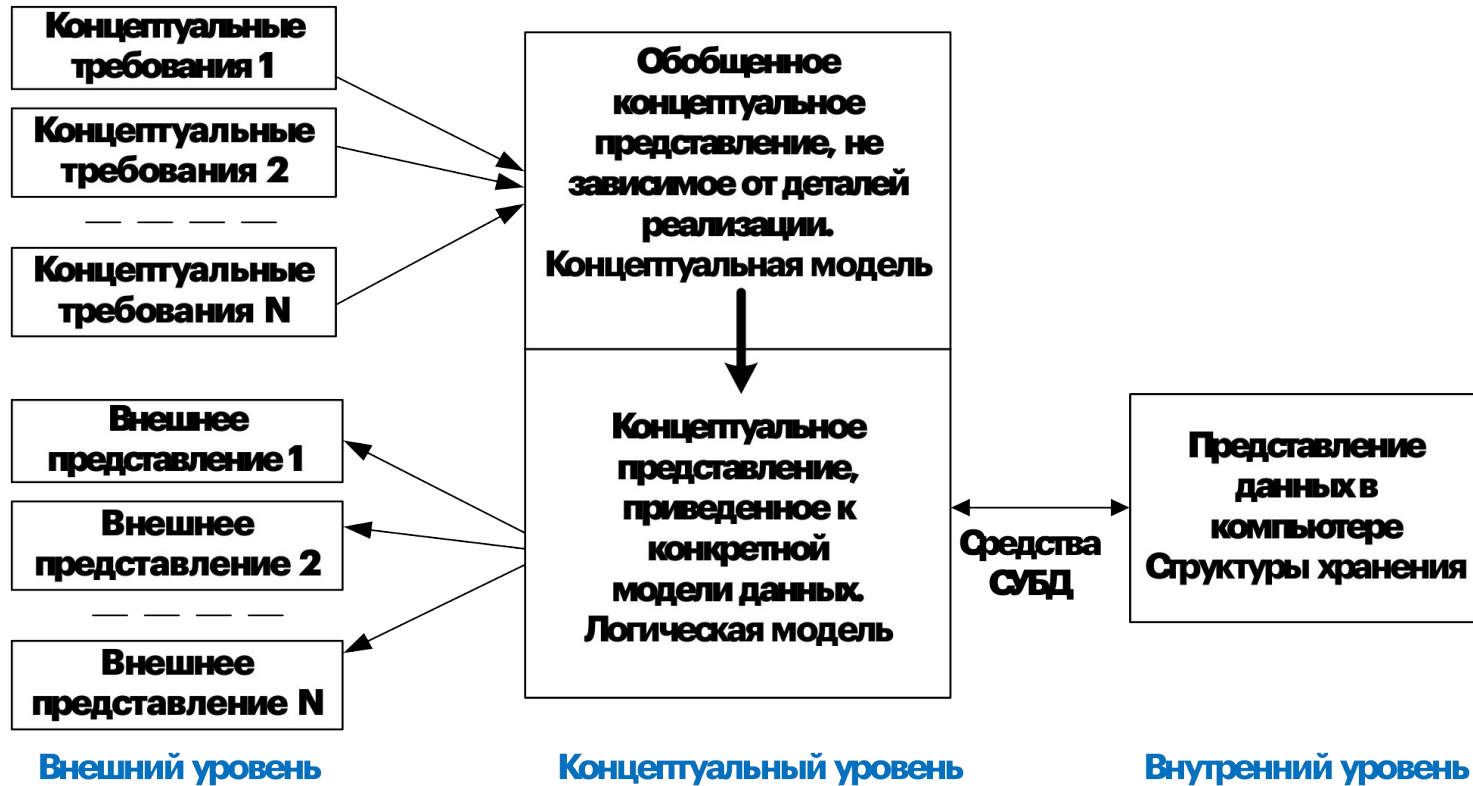
1. **Внешний** — представления для пользователей/приложений,
2. **Концептуальный** — единая логическая модель данных,
3. **Внутренний** — физическое хранение.

Результат:

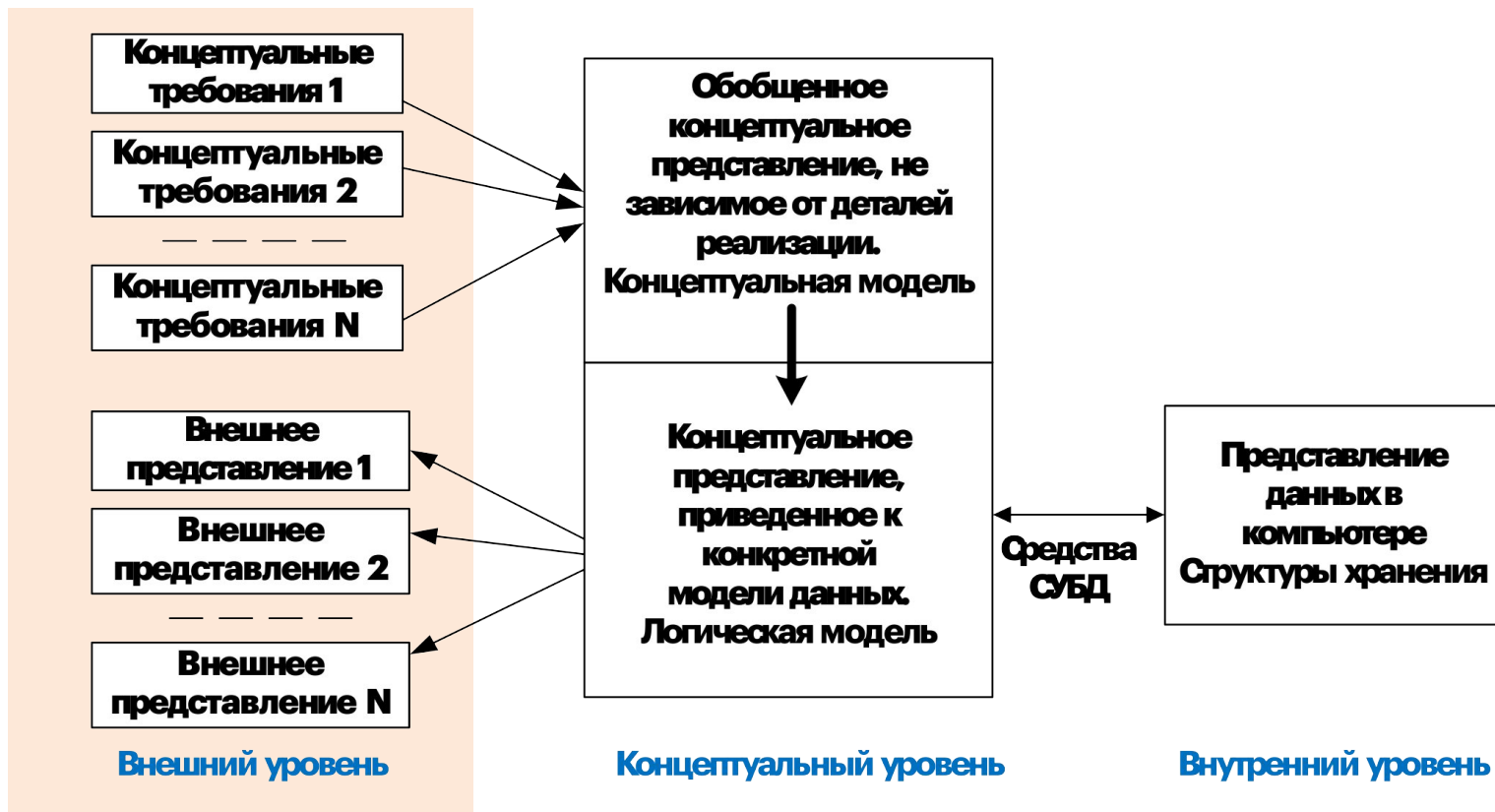
- Изменения на одном уровне **не требуют переделки** других уровней.
- Приложения становятся **устойчивыми к модификациям БД** (например, оптимизации хранения или реорганизации структуры).

Таким образом, ANSI/SPARC ввела **архитектурную гибкость**, став фундаментом современных СУБД.

Трехуровневая модель ANSI/SPARC



Основная цель - отделение пользовательского представления о данных от их физического представления.



Внешнее представление – для специалиста предметной области (пользователя). Работают только с интерфейсом, с элементами управления.

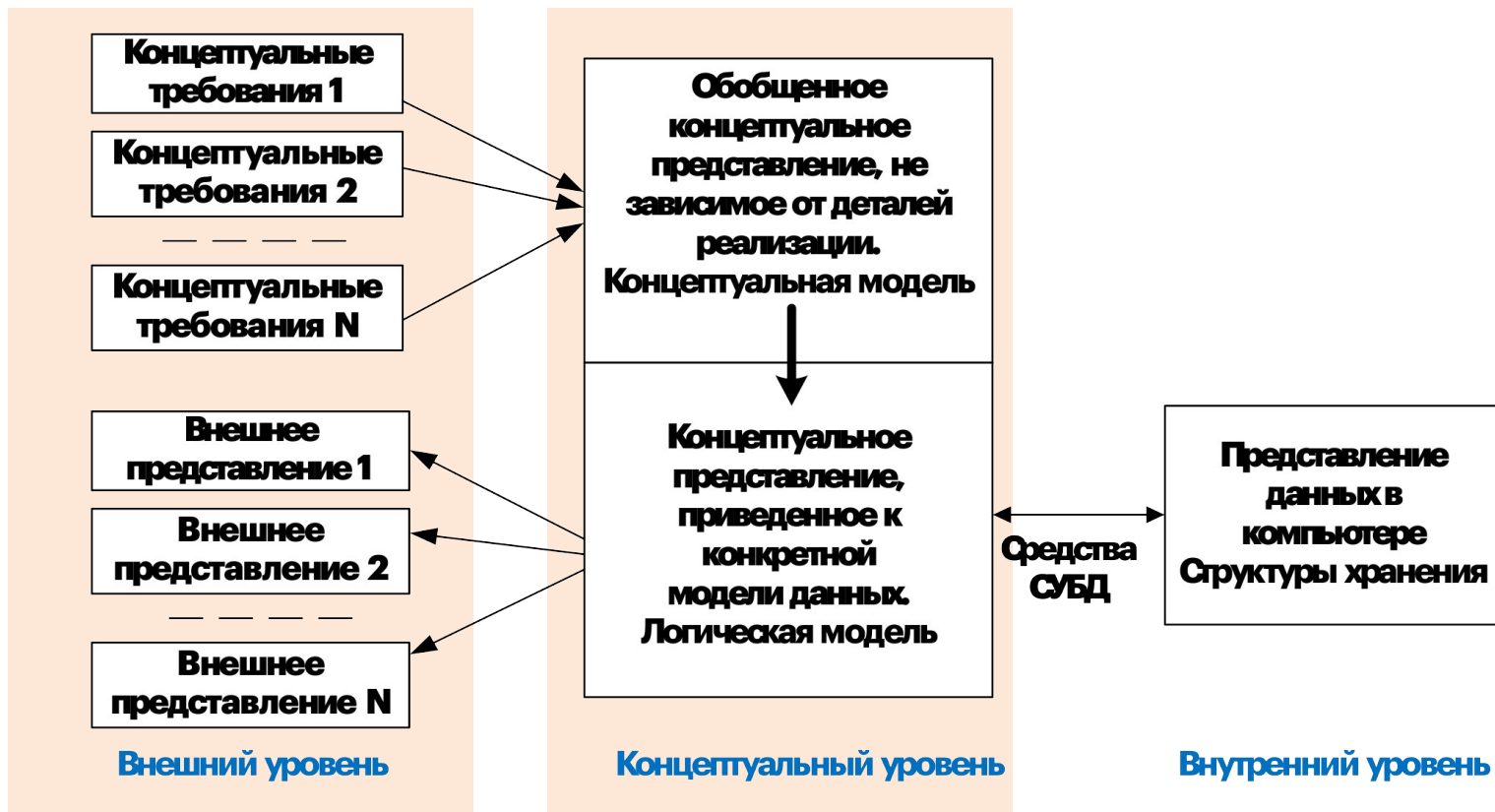
Пользовательский интерфейс должен быть удобным, понятным, связанным с предметной областью.

Трехуровневая модель ANSI/SPARC

1. Внешний уровень (External Level / View Level)

Это **уровень пользователя** — то, как данные видят конечные пользователи или приложения.

- Каждый пользователь может иметь собственное **представление (view)** данных, включающее только ту часть базы, которая ему нужна.
- Например, бухгалтер видит только финансовые данные, а HR — только кадровые.
- Обеспечивает **логическую независимость данных**: изменения в структуре базы (на более низких уровнях) не обязательно влияют на внешние представления.



Внешнее представление и логическая модель – для прикладного программиста.

Программисты разрабатывают:

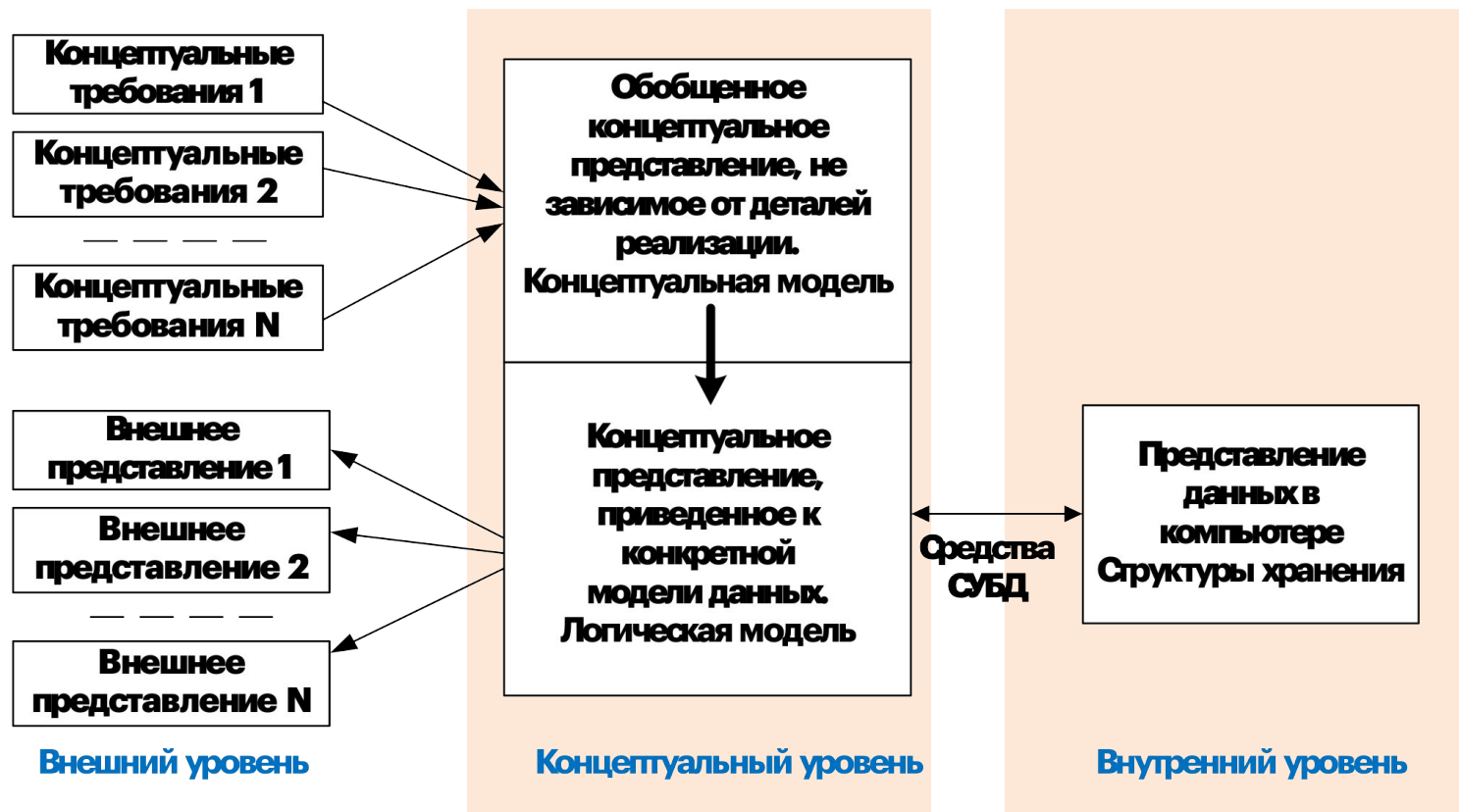
- пользовательский интерфейс и функционал приложения (это связано с предметной областью),
- взаимодействие с логическими структурами данных (абстракция, не связанная с предметной областью).

Трехуровневая модель ANSI/SPARC

2. Концептуальный уровень (Conceptual Level / Logical Level)

Это общее логическое представление всей базы данных.

- Описывает, какие данные хранятся и какие между ними существуют связи (сущности, атрибуты, ограничения целостности).
- Не зависит от физического хранения и от конкретных пользовательских представлений.
- Управляется администратором базы данных (DBA).
- Обеспечивает **логическую целостность** и служит промежуточным звеном между внешним и внутренним уровнями.



Логическая модель и внутреннее представление – для администратора базы данных.

Администратор БД может быть полностью абстрагирован от предметной области.

Трехуровневая модель ANSI/SPARC

3. Внутренний уровень (Internal Level / Physical Level)

Это **физическое представление данных** — как они реально хранятся на диске.

- Включает детали: форматы файлов, индексы, методы сжатия, хеширование, размещение данных и т.д.
- Отвечает за **физическую независимость данных**: изменения в способе хранения (например, переход на SSD или изменение структуры индекса) не должны влиять на концептуальный и внешний уровни.

Трёхуровневая модель ANSI/SPARC

Преимущества трёхуровневой архитектуры:

- Независимость данных (логическая и физическая).
- Гибкость при изменении структуры или хранения.
- Безопасность за счёт ограничения доступа на уровне представлений.
- Упрощённое управление для администраторов и разработчиков.

Эта модель лежит в основе большинства современных реляционных СУБД, таких как PostgreSQL, MySQL, Oracle и др.

Представление 1. «Лист растения»



Представление 2. «Змея»



Представление 3. «Дерево»



Представление 4. «Веревка»



Представление 1. «Лист растения»



Представление 2. «Змея»



Представление 3. «Дерево»

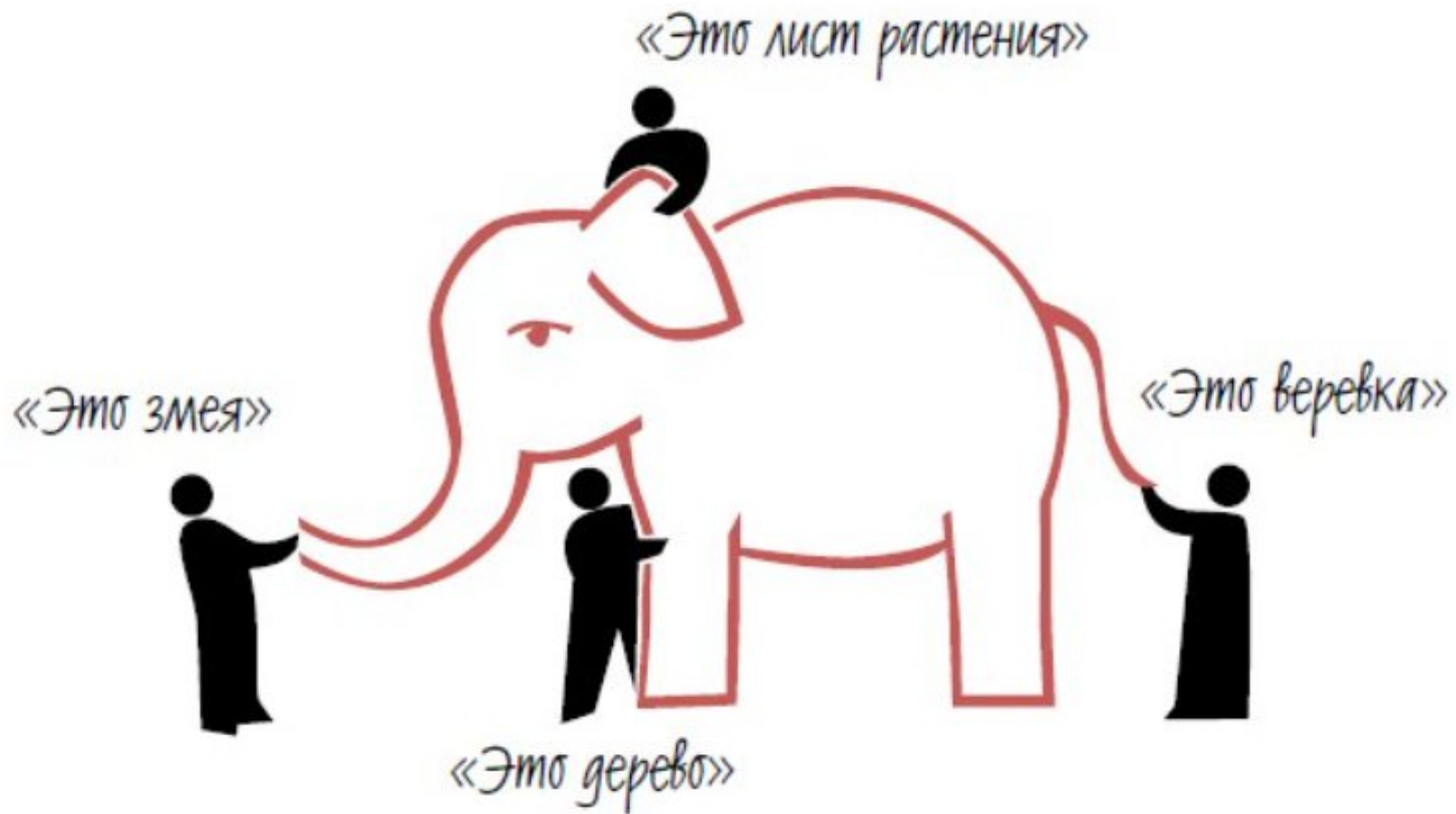


Представление 4. «Веревка»



О чем/о ком эти
представления?

Уровни представления о слоне



Слон и база данных

Представим, что слон — это база данных в целом, а каждый слепой — пользователь или разработчик, взаимодействующий с ней на своём уровне:

1. Внешний уровень (External level) → «Слон — это верёвка / веер / стена»

- Это то, что видит (или "чувствует") конечный пользователь или приложение.
- У каждого — своё **представление (view)**: бухгалтер видит только финансовые данные, логист — только заказы и склады, HR — только сотрудников.
- Как слепые, они взаимодействуют лишь с **фрагментом** системы, но для них это и есть «вся правда».

📌 *Они не знают (и не обязаны знать), как устроен слон целиком.*

2. Концептуальный уровень (Conceptual level) → «Это слон»

- Это **единая, целостная модель** предметной области: все сущности, связи, бизнес-правила.
- Здесь понятно, что хобот, нога, ухо и бок — **части одного организма**.
- Этот уровень создаётся аналитиками и архитекторами, чтобы **согласовать разные точки зрения** и избежать противоречий.

“ Если бы слепые могли "увидеть" слона целиком, они бы поняли: все правы частично, но реальность сложнее.”

3. Внутренний уровень (Internal level) → «Как устроен слон изнутри: кости, мышцы, нервы»

- Это **физический уровень**: как данные хранятся на диске, как организованы индексы, буферы, файлы.
- Пользователи даже не подозревают о нём — как слепые не знают о скелете или кровеносной системе слона.
- Отвечает за **производительность, надёжность, масштабируемость**.

“ 📌 *Даже если слона перекрасить или надеть на него попону (изменить внешние представления), его внутреннее устройство останется тем же. ”*



-  Почему это иллюстрирует независимость уровней?
- Можно изменить внешнее представление (например, дать бухгалтеру новую форму отчёта — «теперь слон похож на шланг») — **концептуальная и физическая модель не пострадают.**
 - Можно оптимизировать хранение (перейти на SSD или сжать данные — «усилить мышцы слона») — **пользователи ничего не заметят.**
 - Можно уточнить бизнес-логику (добавить сущность «Контрагент» — «понять, что у слона есть ещё и хвост») — **физическое хранение можно адаптировать позже, не ломая интерфейсы.**

Вывод

“Притча о слепых и слоне идеально отражает суть трёхуровневой архитектуры:

каждый уровень даёт своё корректное, но неполное представление о данных.

Задача проектировщика базы данных — сохранить целостность “слона”, позволяя каждому “слепому” взаимодействовать с ним так, как ему нужно, — без путаницы, дублирования и противоречий. ”

Таким образом, ANSI/SPARC — это не просто техническая схема, а философия согласования разных точек зрения в единую, устойчивую систему.  

Моделирование и проектирование базы данных

- **Модель предметной области** — это концептуальное представление сущностей, их атрибутов, поведения и взаимосвязей в рамках определённой области деятельности, отражающее бизнес-правила и логику без привязки к технической реализации.
- **Модель базы данных** — это способ описания структуры базы данных с помощью формализованного (в том числе графического) языка на некотором уровне абстракции (концептуальном, логическом или физическом), фиксирующий сущности, атрибуты, типы данных, ограничения и связи, необходимые для хранения и обработки информации в СУБД.

Связь между моделью предметной области и базой данных

- **Модель предметной области служит основой для моделирования базы данных.**

На этапе проектирования базы данных разработчики используют концептуальные сущности и связи из предметной модели и преобразуют их в таблицы, столбцы, внешние ключи и другие элементы, соответствующие выбранной СУБД.

- **Моделирование БД — это техническая реализация предметной модели.**

Например, сущность «Покупатель» из предметной области может стать таблицей `customers`, а связь «Покупатель делает заказ» — внешним ключом `customer_id` в таблице `orders`.

Моделирование и проектирование

- **Моделирование базы данных** — это процесс создания абстрактного представления структуры данных, их взаимосвязей и ограничений. Обычно включает построение концептуальной, логической и физической моделей (например, с использованием ER-диаграмм).
- **Проектирование базы данных** — более широкий процесс, включающий не только моделирование, но и выбор СУБД, определение индексов, планирование репликации, резервного копирования, нормализацию/денормализацию, а также учёт требований производительности и безопасности.

Основные задачи моделирования БД

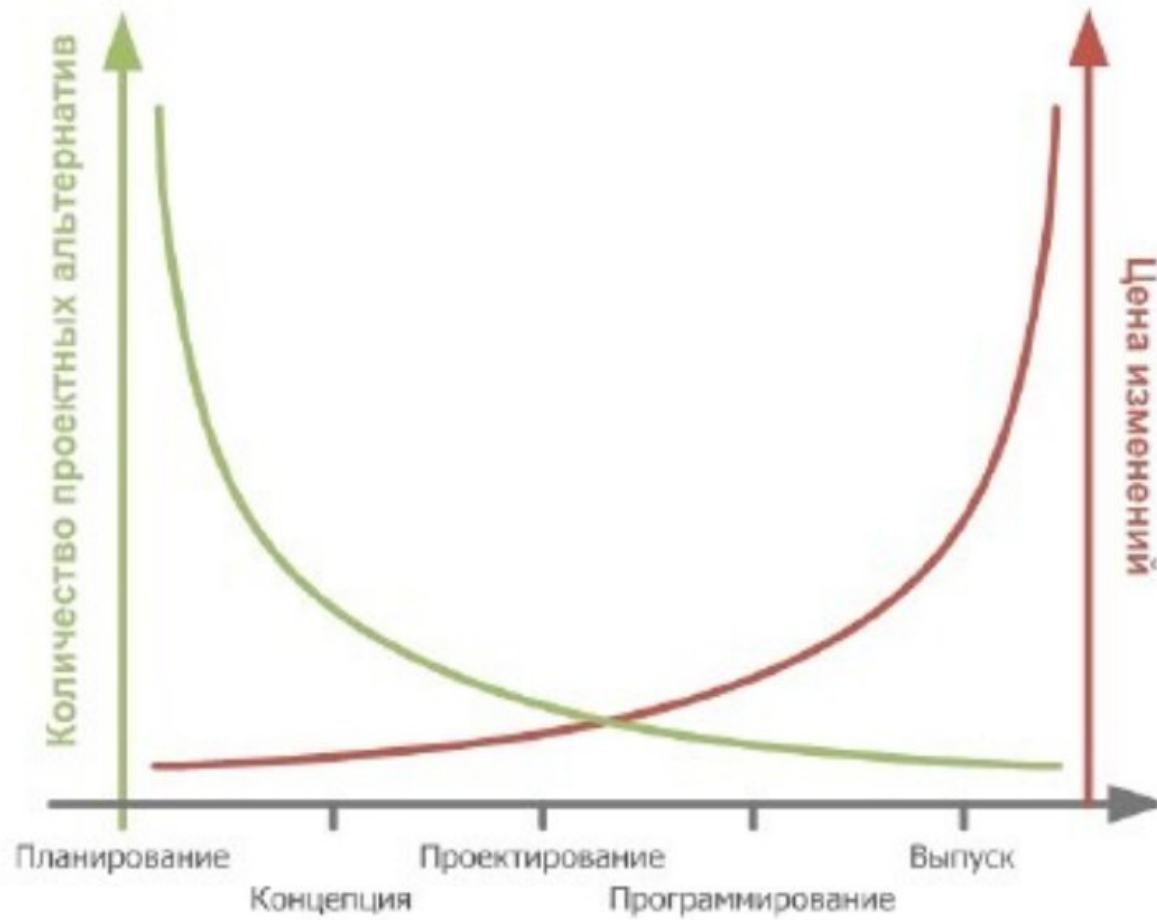
- Хранение в БД всей необходимой информации.
- Получение данных по всем необходимым запросам.
- Сокращение избыточности и дублирования данных.
- Обеспечение целостности данных (исключение противоречий, предотвращение потерь и т.д.).

Важность проектирования базы данных

Строим без плана



Проектирование БД



1. **Целостность данных** — правильно спроектированная база обеспечивает согласованность и точность данных (например, через ограничения, внешние ключи, нормализацию).
2. **Производительность** — хорошая структура упрощает и ускоряет выполнение запросов, особенно при работе с большими объёмами данных.
3. **Масштабируемость** — грамотное проектирование позволяет системе расти без необходимости полной перестройки архитектуры.

4. **Упрощение сопровождения** — понятная и логичная структура облегчает поддержку, модификацию и отладку.
5. **Безопасность** — проектирование включает продумывание уровней доступа, ролей и защиты данных.
6. **Снижение избыточности** — устранение дублирования данных экономит место и уменьшает риск ошибок.

Этапы проектирования базы данных

Этапы проектирования

- **Сбор данных**
- **Концептуальное проектирование** - создание концептуальной модели данных, исходя из представлений пользователей о предметной области
- **Логическое проектирование** - преобразование концептуальной модели на основе выбранной модели данных в логическую модель, не зависящую от особенностей СУБД
- **Физическое проектирование** - описание конкретной реализации базы данных в выбранной СУБД

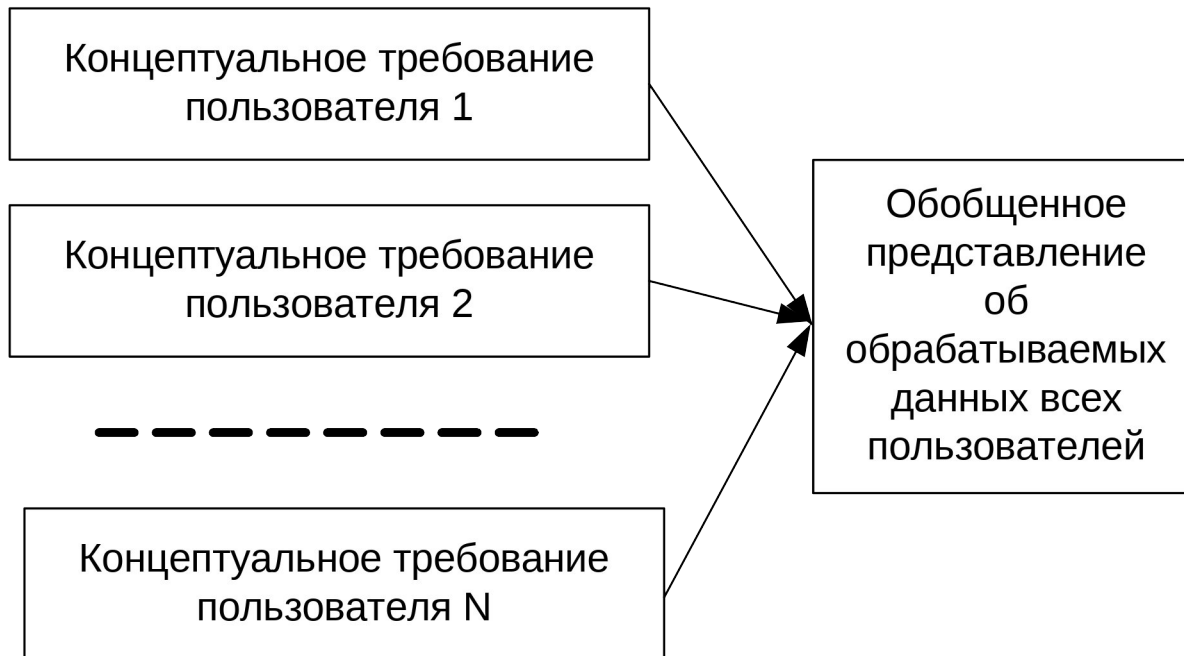
Уровни моделирования базы данных

Уровень		Что описывает	Чем оперирует
Увеличение степени детализации ↓ Логический уровень	Инфологический (концептуальный) уровень	Предметная область без привязки к виду баз данных	Сущности, атрибуты, некоторые связи
	Даталогический (логический) уровень	Предметная область с привязкой к виду базы данных или даже конкретной СУБД	Сущности, атрибуты, связи, ключи, некоторые индексы и представления
	Физический уровень	Технические аспекты реализации базы данных под управлением конкретной СУБД	Сущности, атрибуты, связи, ключи, индексы, представления, триггеры, хранимые подпрограммы, методы доступа, кодировки, права доступа и т.д. и т.п.

Концептуальная модель. ER-диаграммы

Концептуальная модель базы данных

Информационное описание предметной области с учетом логических взаимосвязей.



Концептуальная модель базы данных — это абстрактное, высокоуровневое описание структуры данных предметной области, включающее:

- основные сущности (объекты, понятия),
- их атрибуты (свойства),
- а также логические связи и ограничения между ними,

без привязки к конкретной СУБД или способу физического хранения.

Она отражает суть предметной области с точки зрения бизнес-логики и предназначена для:

- взаимопонимания между аналитиками, разработчиками и заказчиками,
- обеспечения общей картины данных до перехода к логическому и физическому проектированию.

Примеры представления:

- Диаграммы «сущность–связь» (ER-диаграммы),
- UML-диаграммы классов (в объектно-ориентированном подходе).

Терминология

Сущность (Entity) или объект – то, о чем будет накапливаться информация в информационной системе.

Правильная (сильная) сущность не зависит от существования других объектов. **Слабая сущность** не может существовать при отсутствии некоторой другой сущности.

Атрибут – поименованное свойство (характеристика) сущности. Совокупность атрибутов = **запись об объекте**.

Конкретная сущность = **экземпляр сущностей**. Экземпляры сущностей должны однозначно идентифицироваться по одному или нескольким атрибутам.

Связь (отношение) описывает имеющуюся в предметной области логическую связь между сущностями.

Сущность: СТУДЕНТ

Атрибуты:

- номер зачетной книжки,
- фамилия,
- дата рождения,
- место рождения

Экземпляр сущностей: {123456, Иванов, 01.01.2000, Саранск}

Этап 1. Концептуальное проектирование

1 этап - концептуальное проектирование

- Определение сущностей
- Определение связей между сущностями
- Создание ER-модели предметной области
- Определение атрибутов (имя, тип, значение по умолчанию, NULL)
- Определение наборов допустимых значений атрибутов
- Определение первичных ключей для сущностей

Определение сущностей:

- Понимание, какая информация должна храниться и обрабатываться в виде сущности.
- Присвоение сущности имени.
- Выявление атрибутов сущности и присвоение им имени.
- Исключение синонимов и ононимов имен.
- Определение уникального идентификатора сущности.

Определение связей:

- Выявление «естественных» связей экземпляров одной сущности с экземплярами другой.
- Установка связей, позволяющих ответить на все возможные запросы пользователей.
- Присвоение имен связям и определение их типа.

Результат 1-го этапа проектирования БД

- **Концептуальная модель данных**, представленная в виде **ER-диаграммы** (в нотации Чена, Crow's Foot, UML или IDEF1X — в зависимости от принятых стандартов проекта).
- Дополнительно — **гlossарий терминов** и **описание сущностей, атрибутов, связей и доменов**.

Эта модель:

- Не зависит от СУБД.
- Отражает суть предметной области.
- Согласована с заинтересованными сторонами (заказчиками, аналитиками, бизнес-пользователями).
- Служит основой для логического проектирования.

Диаграмма сущностей-связей (Entity-Relationship)

ER-модель — это **графическое и/или текстовое представление** структуры данных, объединяющее сущности, их атрибуты и связи. Она:

- Абстрагирована от технических деталей реализации.
- Понятна как аналитикам, так и заказчикам (даже без IT-подготовки).
- Служит основой для последующих этапов проектирования.

ER-модель может быть создана вручную или с помощью CASE-инструментов (например, ERwin, PowerDesigner, Lucidchart, Draw.io и др.).

Диаграмма сущностей-связей (Entity-Relationship)



Нотация Чена

Предложена Питером Ченом (Peter Chen) в 1976 году — автором самой концепции ER-моделирования.

Особенности:

- Сущности изображаются прямоугольниками.
- Атрибуты — овалами, соединёнными с соответствующей сущностью или связью.
- Связи (relationships) — ромбами, соединяющими сущности.
- Кратность (кардинальность) указывается рядом с линиями связи (например, «1», «N»).

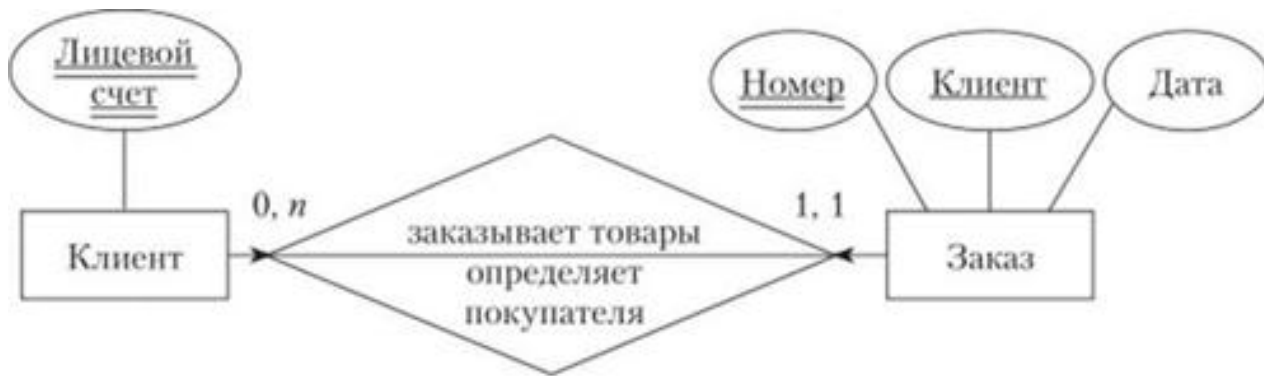
Преимущества:

- Наглядно отражает концептуальную структуру.
- Подчёркивает различие между сущностями, атрибутами и связями.

Недостатки:

- Громоздка при большом количестве атрибутов.
- Редко используется в промышленной практике сегодня (чаще — в учебных целях).

Нотация Чена



Нотация Мартина (Crow's Foot)

Разработана Джеймсом Мартином (James Martin) в 1980-х годах как часть методологии информационного проектирования.

Особенности:

- Сущности представлены прямоугольниками, внутри которых перечислены атрибуты (часто с указанием первичного ключа).
- Связи — линиями между сущностями.
- Кардинальность обозначается специальными графическими символами на концах линий:
 - «галочка» (crow's foot) → «много»,
 - перпендикулярная черта → «ровно один»,
 - кружок → «ноль».

Нотация Мартина (Crow's Foot)

Примеры обозначений:

- ○|— : «ноль или один»
- |— : «обязательно один»
- |<— : «один или много»
- ○<— : «ноль или много»

Преимущества:

- Компактна и удобна для проектирования реляционных БД.
- Широко поддерживается в CASE-инструментах (например, Oracle Data Modeler, ERwin, MySQL Workbench).
- Легко читаема разработчиками.

Недостатки:

- Менее наглядна для демонстрации атрибутов связей (в отличие от нотации Чена).

Нотация Мартина (Crow's Foot)



One



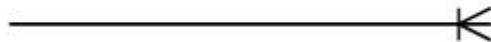
Many



One (and only one)



Zero or one



One or many



Zero or many

Этап 2. Логическое проектирование БД

2 этап - логическое проектирование БД

- Выбор модели данных (реляционная, иерархическая, сетевая)
- Определение наборов структурированных данных.
Нормализация таблиц (для реляционной модели)
- Проверка модели данных на предмет выполнения всех транзакций бизнес-процессов
- Определение требований поддержки целостности данных

Логическое проектирование БД

- **Сущности** концептуальной модели описываются как записи, состоящие из полей.
- Каждый **атрибут** описывается как поле с типом и характеристиками, возможными в выбранной модели данных.
- **Связи** концептуальной модели описываются в понятиях, соответствующих выбранной модели данных.
- Определяется порядок реализации запросов пользователей к базе данных с помощью типовых операций СУБД, поддерживающих выбранную модель данных, и т.д.

Результат 2-го этапа проектирования БД

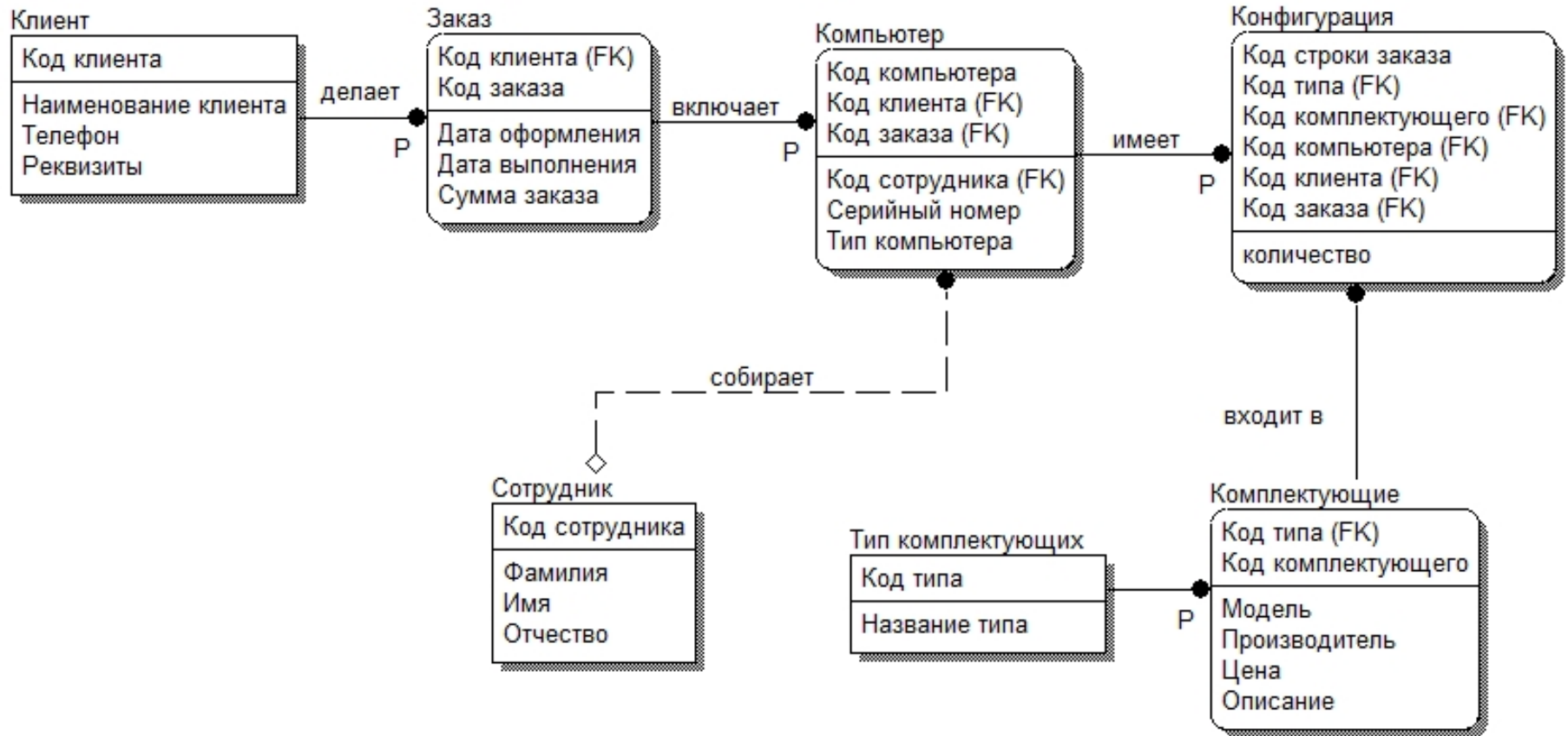
Результатом этапа является **логическая модель данных**, которая:

- не зависит от конкретной СУБД,
- полностью отражает структуру предметной области,
- содержит все сущности, атрибуты, связи и ограничения целостности,
- готова к трансляции в физическую схему.

Часто такая модель оформляется в виде **диаграммы в нотации IDEF1X** (или в ER-нотации). **IDEF1X** — стандарт, ориентированный именно на реляционные базы данных, и позволяет чётко выразить:

- сильные и слабые сущности,
- обязательные и необязательные связи,
- зависимости по идентификатору,
- ключи и атрибуты.

Нотация IDEF1X



Нотация IDEF1X

IDEF1X (читается как «ай-деф икс») — это стандартизированная методология и нотация для моделирования данных, разработанная в рамках программы IDEF (Integration Definition for Function Modeling) Министерства обороны США (DoD). IDEF1X специализируется на концептуальном и логическом проектировании реляционных баз данных и отличается строгой семантикой, ориентированной на бизнес-правила и целостность данных.

♦ История и происхождение

- **Разработана:** в 1980-х годах на основе работ Дж. Питера Чена (ER-модель) и методологии ICAM (Integrated Computer-Aided Manufacturing).
- **Стандартизирована:** как FIPS PUB 184 (Federal Information Processing Standard) в 1993 году.
- **Цель:** обеспечить единый, точный и проверяемый способ описания структуры данных в сложных информационных системах, особенно в оборонной и государственной сферах.

♦ Основные принципы IDEF1X

1. Ориентация на сущности и связи, но с чётким разделением на:
 - Независимые сущности (strong/identifier-independent entities)
 - Зависимые сущности (weak/identifier-dependent entities)
2. Все связи — бинарные и обязательные для интерпретации (нет «многозначных» связей).
3. Атрибуты не изображаются отдельно, а перечисляются внутри блоков сущностей.
4. Связи не являются отдельными объектами (в отличие от нотации Чена) — они выражаются через зависимости идентификаторов.

1. Сущности

- **Независимая сущность** (может существовать сама по себе):
 - прямоугольник с **сплошной рамкой**, имя — в верхнем регистре.Пример: `PERSON` , `PRODUCT` .
- **Зависимая сущность** (не может быть идентифицирована без другой сущности):
 - прямоугольник с **закруглёнными углами** или **двойной рамкой** (в разных реализациях), имя — также в верхнем регистре.Пример: `ORDER_ITEM` зависит от `ORDER` .

2. Связи

IDEF1X определяет два типа связей:

а) Связь идентификации (Identifying Relationship)

- Родительская сущность участвует в первичном ключе дочерней.
- Обозначается сплошной линией.
- Дочерняя сущность — зависимая (с закруглёнными углами).
- Пример: `ORDER` → `ORDER_LINE` (без заказа позиция не существует).

б) Связь неидентификации (Non-Identifying Relationship)

- Родительская сущность не входит в первичный ключ дочерней.
- Обозначается пунктирной линией.
- Дочерняя сущность — независимая (прямоугольник с прямыми углами).
- Пример: `CUSTOMER` → `ORDER` (заказ ссылается на клиента, но идентифицируется своим ID).

Нотация IDEF1X

♦ Преимущества IDEF1X

- Высокая точность: модель однозначно транслируется в реляционную схему.
 - Поддержка нормализации: естественно отражает функциональные зависимости.
 - Бизнес-ориентированность: связи выражают реальные правила предметной области.
 - Стандартизация: подходит для официальной документации и аудита.
-

♦ Недостатки

- Сложность для новичков: требует понимания зависимостей идентификаторов.
- Менее визуально интуитивна, чем Crow's Foot.
- Ограниченная поддержка в современных open-source инструментах (чаще встречается в промышленных CASE-системах, например, ERwin, PowerDesigner).

Этап 3. Физическое проектирование БД

3 этап - физическое проектирование БД

- Проектирование таблиц средствами выбранной СУБД
- Реализация бизнес-правил в среде выбранной СУБД
- Проектирование физической организации СУБД
- Разработка стратегии защиты БД
- Организация мониторинга функционирования БД и ее настройка

Результат 3-го этапа проектирования БД

- Полностью готовая к развертыванию схема базы данных, включая:
 - SQL-скрипты создания таблиц, индексов, представлений, триггеров и хранимых процедур.
 - Настройки безопасности (роли, привилегии, политики аудита).
 - Документация по физической структуре и рекомендациям по эксплуатации.
- План мониторинга и оптимизации производительности.
- Стратегия резервного копирования и восстановления.

Эти артефакты позволяют развернуть рабочую базу данных в продакшене, обеспечив соответствие требованиям по функциональности, производительности, безопасности и надёжности.