Базы данных

Москва
2011

Зарплата будет выдана 25 сентября.

Бухгалтерия.

ИНФОРМАЦИЯ

ПРЕДСТАВЛЕНИЕ

СЕМАНТИКА

Въезд запрещен

ИНФОРМАЦИЯ:

Сущность обеспечивающая повышение знаний об окружающем мире ее получателем.

ОЦЕНКА КАЧЕСТВА ИНФОРМАЦИИ:

- Достоверность.
- Интерпретируемость.

Рассинхронизация семантики и представления.

Петров	98
Сидоров	23
Иванов	41

Поддержка ограничений реального мира

Петров	98
Сидоров	-23
Петров	0

Дублирование информации, как источник рассинхронизации

Петров	98
Сидоров	23
Петров	37

1.8

Проблемы, возникающие при хранении информации :

- Рассинхронизация семантики и представления .
- Поддержка ограничений реального мира.
- Дублирование информации, как источник рассинхронизации.

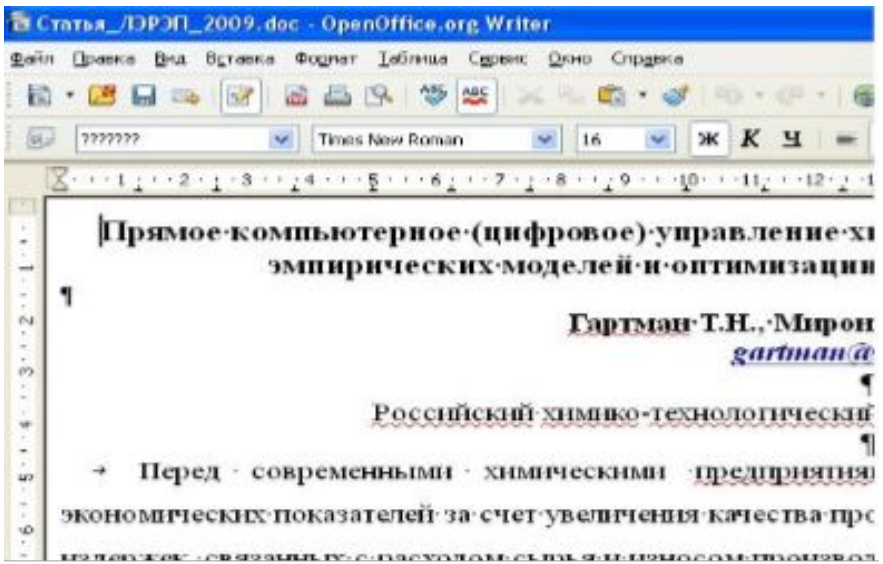
2.1

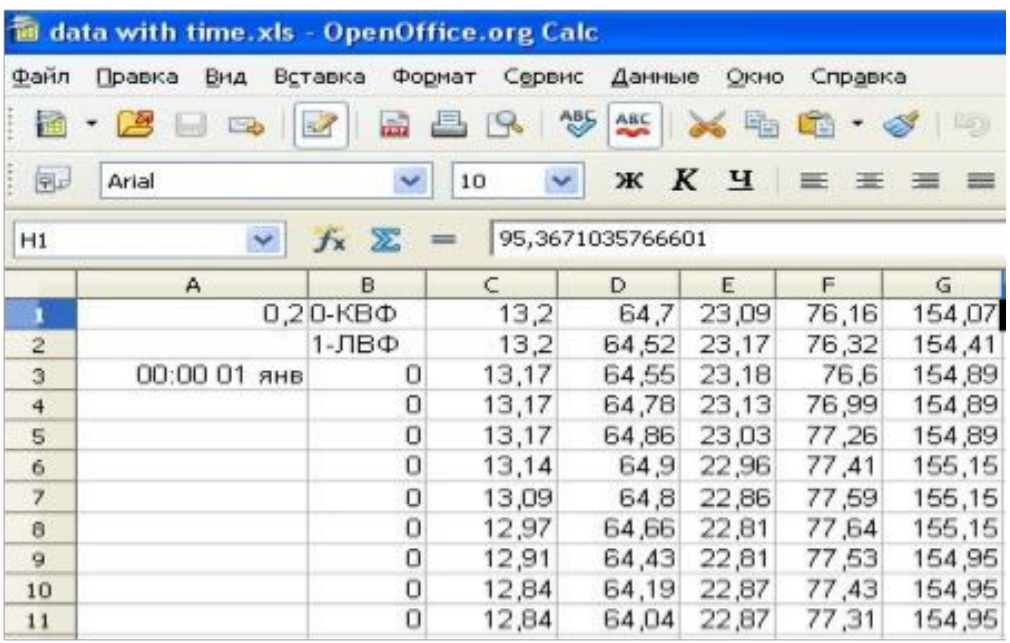
- Базы данных – большой объем информации, предназначенный для хранения в компьютерных системах, таким образом, чтобы облегчить поиск и изменение данных.

Системы управления базами данных

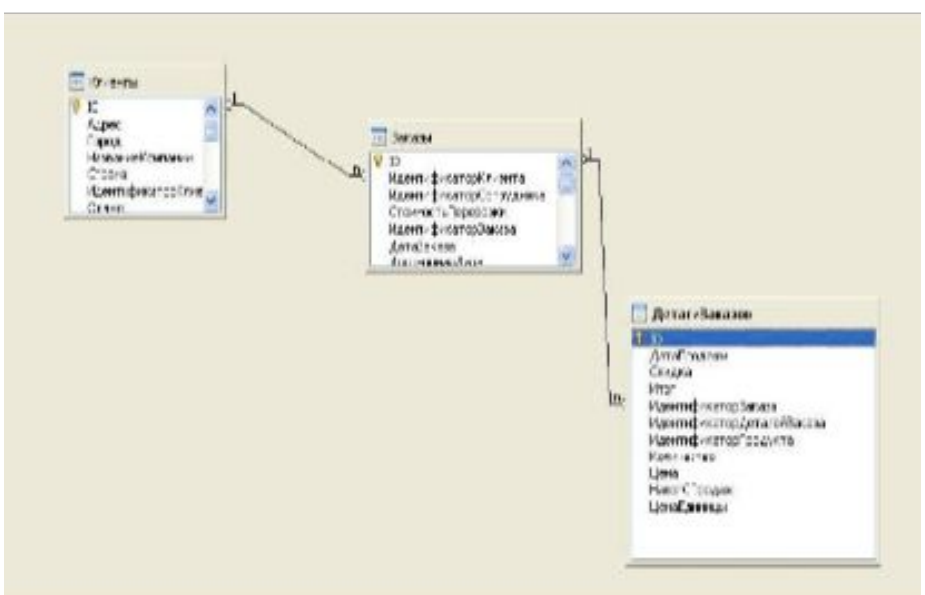
2.2

СУБД?





	A	B	C	D	E	F	G
1	0,2	0-КВФ	13,2	64,7	23,09	76,16	154,07
2		1-ЛВФ	13,2	64,52	23,17	76,32	154,41
3	00:00 01 янв	0	13,17	64,55	23,18	76,6	154,89
4		0	13,17	64,78	23,13	76,99	154,89
5		0	13,17	64,86	23,03	77,26	154,89
6		0	13,14	64,9	22,96	77,41	155,15
7		0	13,09	64,8	22,86	77,59	155,15
8		0	12,97	64,66	22,81	77,64	155,15
9		0	12,91	64,43	22,81	77,53	154,95
10		0	12,84	64,19	22,87	77,43	154,95
11		0	12,84	64,04	22,87	77,31	154,95



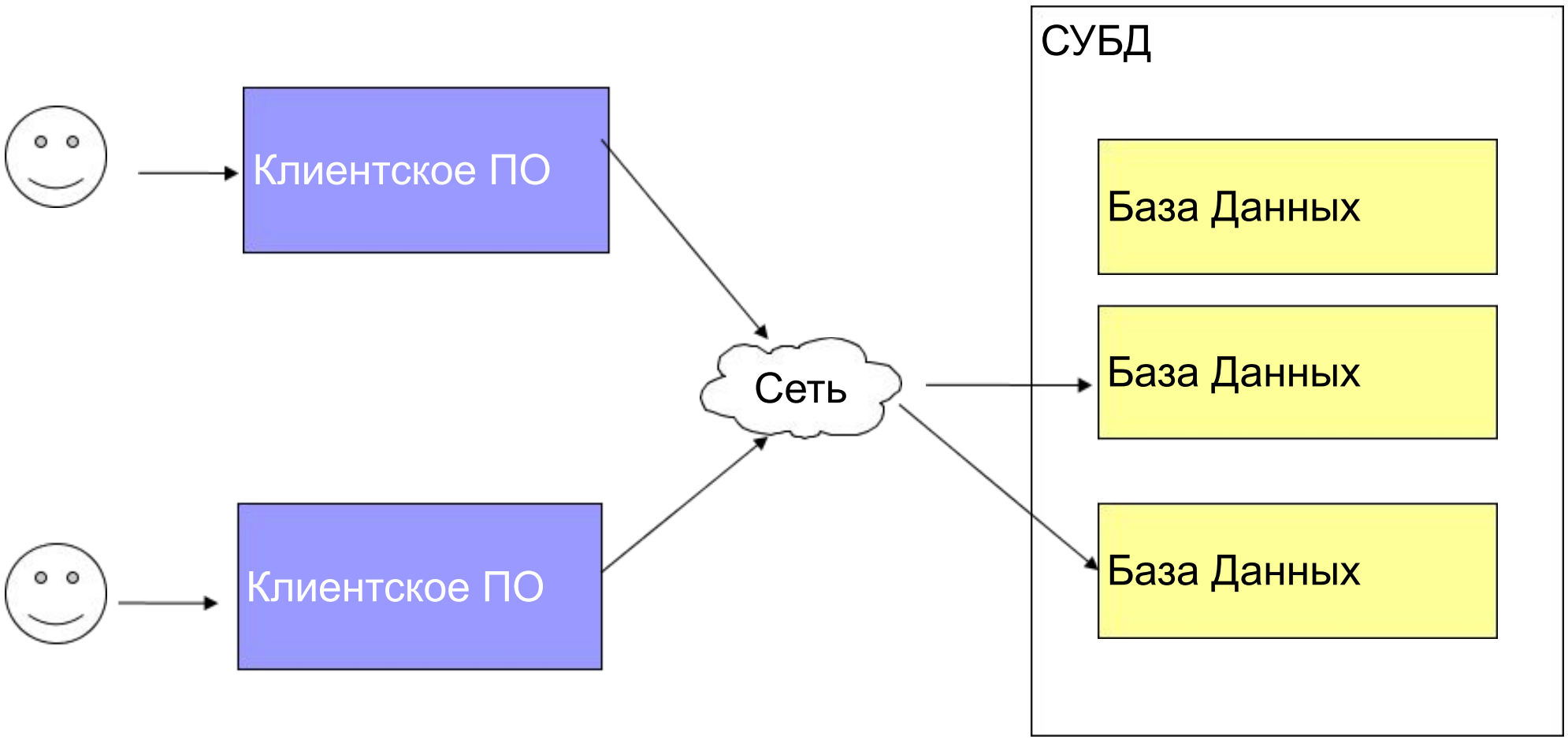
Система управления базами данных.

- Отделение семантики данных от представления. Хранение метаданных о семантике данных:
Сотрудник

ФИО	*Зарплата, тыс. руб.*
Сидоров	23
Петров	37

- Наличие встроенного языка для семантического поиска данных (язык построения запросов):
Вывести ФИО сотрудников с зарплатой больше 30 тыс. руб.

Информационная система на основе базы данных



Разработчик баз данных.

- *Задачи*:
  Проектирование структуры БД, реализация БД в рамках заданной СУБД.

- *Требования к СУБД*:
  - Наличие средств решения проблем, возникающих при работе с информацией.
  - Стандартизованные средства создания БД и манипулирования данными.

Разработчик клиентского ПО.

- *Задачи*:
  Создание внешнего пользовательского интерфейса для работы с БД.

- *Требования к СУБД*:
  - Удобный станадартизованный доступ к данным.
  - Наличие средств построения запросов и манипулирования данными.

Администратор БД.

- *Задачи*:
  Обеспечение бесперебойной работы ИС.
  Поддержание необходимого QoS.

- *Требования к СУБД*:
  - Обеспечение контроля доступа.
  - Надежность.
  - Высокая производительность.
  - Поддержка больших объемов хранимой информации.
  - Масштабируемость системы.

Требования к СУБД:

- Наличие стандартизованных средств проектирования структуры БД, устойчивой к проблемам хранения информации.
- Наличие стандартизованных средств построения запросов и манипулирования данными.
- Обеспечение вопросов безопасности и надежности
- Высокая производительность и поддержка хранения больших объемов информации.
- Масштабируемость системы.

1. Табличное представление данных на логическом уровне:

Таблицы
Представление конкретной сущности из предметной области.

Столбцы
Свойства (атрибуты), описывающие сущность.

Строки
Конкретные экземпляры сущности.

Сотрудник

ФИО	*Дата рождения*	*Зарплата, тыс.руб.*	*Специальность*
Петров В.В.	15.07.74	48	бухгалтер
Иванов И.П.	26.05.65	67	Зам. директора

2. Использование первичных ключей для идентификации экземпляров сущности:

Потенциальный ключ (*Candidate Key*)
Минимальный набор
атрибутов однозначно идентифицирующих экземпляр сущности.

Первичный ключ (Primary Key)
Один из потенциальных ключей, выбранный для однозначной идентификации сущности.
Суррогатный первичный ключ.
Уникальный аттрибут для идентфикации экземпляра сущности, не имеющий отображения в предметной области

Сотрудник

Номер сотрудника (PK)	***ФИО***	***Дата рождения***	***Зарплата, тыс. руб.***
15	Петров В.В.	15.07.74	48
27	Иванов И.П.	26.05.65	67

Свойства табличной организации данных.

- Имена сущностей должны быть уникальны в рамках схемы БД.
- Имена атрибутов должны быть уникальны в рамках сущности.
- Адресация атрибутов допускается только по имени атрибута.
  Вывести ***ФИО*** **Сотрудника**
- Адресация экземпляров сущности допускается только по значению атрибутов
  Найти **Сотрудника** *с* ***ФИО*** *Петров В.В*

Сотрудник

Номер сотрудника (PK)	***ФИО***	***Дата рождения***	***Зарплата, тыс. руб.***
15	Петров В.В.	15.07.74	48
27	Иванов И.П.	26.05.65	67

3. Связь сущностей через миграцию первичного ключа:

Внешний ключ (Foreign Key)

Аттрибут (совокупность аттрибутов), совпадающий с первичным ключом внешней сущности, использующийся для установки связи между сущностями.

Сотрудник

Номер сотрудника (PK)	*ФИО*	*Дата рождения*	*Зарплата, тыс.руб.*
15	Петров В.В.	15.07.74	48
27	Иванов И.П.	26.05.65	67

Отпуск

Номер отпуска (PK)	Номер сотрудника (FK)	Дата начала	Длительность
10	15	30.12.2008	14
11	15	13.07.2009	14

Свойства организации связей путем миграции первичного ключа.

- Первичный ключ мигрирует целиком. Составной первичный ключ не может мигрировать отдельными частями.
- Сущность из которой произошла миграция ключа носит название родительской. Сущность в которую произошла миграция ключа носит название дочерней.
- Внешний ключ может входить в состав первичного ключа дочерней сущности. В данном случае связь называется идентифицирующей

Вагон

Номер вагона (PK)	*Тип*
2	Купе
3	Плацкарт

Место

Номер вагона (PK) (FK)	*Номер места (PK)*	*Билет*
2	31	ЯГ 763553
2	33	ЯГ 756873

Правила создания таблиц и связей

Реляционные БД основываются на теории Эдгара Кодда (Edgar F. Codd), которая впоследствии была развита Кристофером Дейтом (Cristopher J. Date).

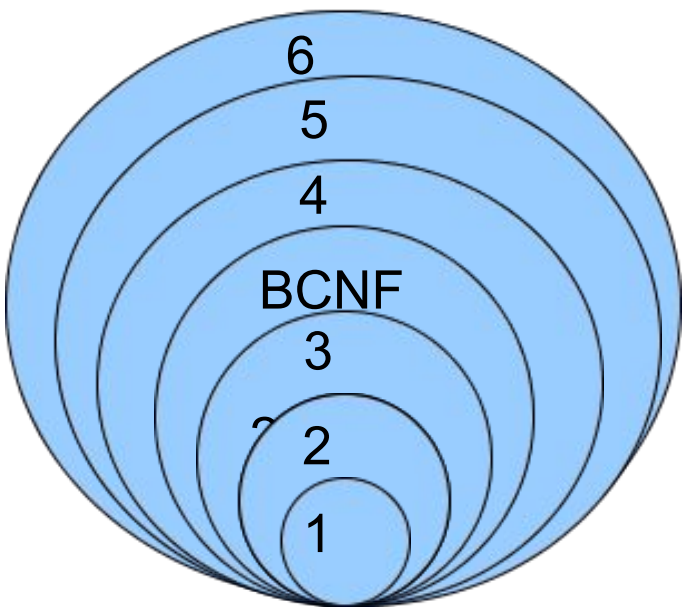
В качестве основы разработки послужили три основных принципа:

- Независимость от порядка расположения записей в физическом представлении. Необходимо добиться независимости порядка записей в представлении пользователю от порядка расположения записей в физическом представлении.
- Независимость от индексирования. Приложения должны оставаться работоспособными при внесении изменений или удаления индексов.
- Независимость от пути доступа. Доступ к данным не должен зависеть от их положения в иерархии

Правила создания таблиц и связей

Для оценки качества схемы реляционной БД было введено понятие нормальных форм.

Нормальная форма.
Совокупность правил, следование которым повышает целостность БД и снижает количество потенциальных аномалий при манипуляции данными.



BCNF — Boyce Codd Normal Form
(Нормальная форма Бойса Кодда)

Первая нормальная форма

E. Codd:
Отношение находится в первой нормальной форме если все его атрибуты являются семантически атомарными.

C. Date:
Отношение находится в первой нормальной форме если оно соответствует следующим условиям:
1) Строки отношения неупорядоченны.
2) Столбцы отношения неупорядочены.
3) Все строки уникальны.
4) Каждая ячейка содержит одно семантически атомарное значение из заданого домена.

Основное различие: Определение Дейта запрещает ячейки с NULL значениями.

Первая нормальная форма

Пример нарушения:

Номер сотрудника (PK)	*ФИО*	*Дата рождения*	*Телефон*
15	Петров В.В.	13.01.1976	543 3232
16	Иванов К.П.	15.09.1962	325 6786, 765 3421

Нормализация

Номер сотрудника (PK)	*ФИО*	*Дата рождения*
15	Петров В. В.	13.01.1976
16	Иванов К. П.	15.09.1962

Номер телефона (PK)	*Номер сотрудника (PK)*	*Телефон*
1	15	543 32 32
1	16	325 6786
2	16	765 3421

3.9

Первая нормальная форма

Пример нарушения (*если требуется поиск по городам проживания*):

Номер сотрудника (PK)	*ФИО*	*Дата рождения*	*Адрес*
15	Петров В.В.	13.01.1976	Москва, ул. Тверская, д.5, кв. 45
16	Иванов К.П.	15.09.1962	Одинцово, ул Ленина, д.9, кв. 76

Нормализация

Номер сотрудника (PK)	*ФИО*	*Дата рождения*	*Город*	*Адрес*
15	Петров В.В.	13.01.1976	Москва	ул. Тверская, д.5, кв. 45
16	Иванов К.П.	15.09.1962	Одинцово	ул Ленина, д.9, кв. 76

Функциональная зависимость.
Пусть **R** - отношение. Множество атрибутов **Y** функционально зависимо от множества атрибутов **X,** тогда и только тогда, когда для любого состояния отношения **R** для любых кортежей $r1, r2 \in R$ из того, что **r1.X = r2.X** следует что **r1.Y = r2.Y**
(т.е. во всех кортежах, имеющих одинаковые значения атрибутов X, значения атрибутов Y так же совпадают в любом состоянии отношения).

Обозначение.
X $\rightarrow$ ***Y***
Y функционально зависит от ***X.***
X функционально определяет ***Y.***
X детерминант,
Y зависимая часть

Следствие.
Если атрибуты **X** составляют потенциальный ключ отношения, то любой атрибут отношения функционально зависит от **X**.

Примеры функциональной зависимости.

Номер телефона (PK)	Номер сотрудника (PK)	Телефон
1	15	543 32 32
1	16	325 6786
2	16	765 3421

{Номер телефона, Номер сотрудника} → Телефон

Номер сотрудника (PK)	ФИО	Дата рождения
15	Петров В.В.	13.01.1976
16	Иванов К.П.	15.09.1962

Номер сотрудника → ФИО

Номер сотрудника → Дата рождения

Вторая нормальная форма

Отношение находится во второй нормальной форме если оно находится в первой нормальной форме и отсутствуют функциональные зависимости неключевых атрибутов от части составного потенциального ключа.
Отношение находится в 2НФ тогда и только тогда, когда оно находится в 1НФ, и каждый его не ключевой атрибут функционально полно зависит от любого возможного ключа этого отношения.

Замечание. Отношения удовлетворяющие первой нормальной форме с простыми первичными ключами удовлетворяют второй нормальной форме без дополнительных проверок.

Пример нарушения второй нормальной формы

№ сотрудника (PK)	*ФИО*	*№ отдела*	*Телефон*	*№ проекта (PK)*	*Название*	*Функции*
1	Петров В. В.	1	978	1	Исследование рынка	Рук.
4	Сидоров Т.О.	2	765	1	Исследование рынка	Исп.
1	Петров В. В.	1	978	3	Рекламная акция	Рук.

Функциональные зависимости, нарушающие вторую нормальную форму :

№ Сотрудника → ФИО, № Проекта→ Название

Проблемы, возникающие при работе с текущим отношением:

- Дублирование информации о сотрудника, отделах и проектах.
- *Аномалии вставки (INSERT):*
  Нельзя вставить нового сотрудника, пока он не назначен на проект.
- *Аномалии обновления (UPDATE):*
  При необходимости изменить название проекта, необходимо изменить его во всех местах где он встречается.
- *Аномалии удаления (DELETE):*
  При удалении информации о проекте удалится информация о сотрудниках Участвующих только в этом проекте. Аналогичная проблема возникает при удалении сотрудников, выполняющих определенный проект целиком

Нормализация до уровня второй нормальной формы

Сотрудник

№ сотрудника (PK)	*ФИО*	*№ отдела*	*Телефон*
1	Петров В.В.	1	978
4	Сидоров Т.О.	2	765

Проект

№ проекта (PK)	*Название*
1	Исследование рынка
3	Рекламная акция

Продолжение схемы на следующем слайде...

Нормализация до уровня второй нормальной формы (продолжение)

Сотрудник проекта

№ сотрудника (PK)	*№ проекта (PK)*	*Функции*
1	1	Рук.
4	1	Исп.
1	3	Рук.

Выводы:

Рассмотренные выше аномалии устранены.

Третья нормальная форма

Отношение **R** находится в третьей нормальной форме если оно находится во второй нормальной форме и отсутствуют транзитивные зависимости атрибутов от потенциального ключа.

Транзитивная зависимость.
Если **X** →**Y** и **Z** →**X**, то зависимость **Z**→**Y** называется транзитивной.

Отношение, нарушающее третью нормальную форму.

Сотрудник

№ сотрудника (PK)	*ФИО*	*№ отдела*	*Телефон*
1	Петров В.В.	1	978
4	Сидоров Т.О.	2	765

Функциональные зависимости, нарушающие третью нормальную форму:

№ сотрудника →№ Отдела, №Отдела → Телефон
Зависимость № Сотрудника → Телефон транзитивна.

Проблемы, возникающие при работе с текущим отношением:

- Дублирование информации об отделах.
- *Аномалии вставки (INSERT):*
  Возможно появление сотрудника с тем же номером отдела,
  но с другим телефоном.
- *Аномалии обновления (UPDATE):*
  При необходимости изменить телефон отдела, необходимо изменить его во всех местах где он встречается.
- *Аномалии удаления (DELETE):*
  При удалении всех сотрудников, принадлежащих определенному отделу исчезает информация об отделе.

Нормализация до уровня третьей нормальной формы

Сотрудник

№ сотрудника (PK)	*ФИО*
1	Петров В.В.
4	Сидоров Т.О.

Отдел

№ отдела (PK)	*Телефон*
1	978
2	765

Продолжение схемы на следующем слайде...

Нормализация до уровня третьей нормальной формы (продолжение)

Сотрудник отдела

№ Сотрудника отдела (PK)	*№ сотрудника (FK)*	*№ отдела (FK)*	*Дата зачисления*	Дата перевода
13	1	1	13.01.2005	NULL
14	4	2	12.09.2007	1.12.2008

Выводы:

Рассмотренные выше аномалии устранены.

Нормальная форма Бойса Кодда

*Отношение **R** находится в нормальной форме Бойса Кодда если оно находится в третьей нормальной форме и все функционально определяющие совокупности атрибутов являются потенциальным ключом.*

Нормализованное отношение находится в НФБК тогда и только тогда, когда каждый детерминант является потенциальным ключом.

До определения НФБК, предполагалось, что в отношении один потенциальный ключ, который и является первичным. НФБК может нарушаться в случае, когда потенциальных ключей несколько.

3.21

Отношение, нарушающее нормальную форму Бойса Кодда.

Сотрудник проекта

№ сотрудника (РК ?)	*ФИО (РК ?)*	*№ проекта (РК ?)*	*Функции*
1	Петров В.В.	1	Рук.
4	Сидоров Т.О.	1	Исп.
1	Петров В.В.	3	Рук.

В данном отношении два потенциальных ключа:
{№Сотрудника, № Проекта} и {ФИО, № Проекта}

Поэтому функциональных зависимостей нарушающих 2НФ и 3НФ нет.

Функциональные зависимости нарушающие форму Бойса Кодда:
№ Сотрудника →ФИО и ФИО → № Сотрудника, т.к их детерминанты не является частью потенциального ключа

Проблемы, возникающие при работе с текущим отношением:

- Дублирование информации о фамилиях сотрудников.
- *Аномалии вставки (INSERT):*
  Отсутствуют
- *Аномалии обновления (UPDATE):*
  При необходимости изменить фамилию сотрудника,
  нужно провести изменения в нескольких местах.
- *Аномалии удаления (DELETE):*
  Отсутствуют.

Нормализация до уровня нормальной формы Бойса Кодда

Сотрудник

№ сотрудника (PK)	*ФИО*
1	Петров В.В.
4	Сидоров Т.О.

Сотрудник проекта

№ сотрудника (PK)	*№ проекта (PK)*	*Функции*
1	1	Рук.
4	1	Исп.
1	3	Рук.

В каждой из новых сущностей потенциальный ключ один, следовательно нарушения НФБК нет. Данный потенциальный ключ выбирается в качестве первичного ключа.

Сравнение сильно и слабо нормализованных отношений

Критерий	Слабая нормализация (1НФ, 2НФ)	Сильная нормализация (3НФ, НФБК)
Адекватность базы данных, соответствующей предметной области	Хуже	Лучше
Удобство разработки и сопровождения	Сложнее	Легче
Скорость операций вставки, обновления, удаления	Медленнее	Быстрее
Скорость операций выборки	Быстрее	Медленнее

Сравнение сильно и слабо нормализованных отношений

Операция выборки, - одна из основных операций в БД, так как ИС на основе БД предполагают использование данных для проведения аналитики и построения отчетов.

В связи с этим на практике, как правило, проводят денормализацию сущностей для увеличения скорости операций построения отчетов.

Один из пунктов денормализации, - включение вычисляемых полей в сущности.

Требования к СУБД:

- Наличие стандартизованных средств проектирования структуры БД, устойчивой к проблемам хранения информации.
- Наличие стандартизованных средств построения запросов и манипулирования данными.
- Обеспечение вопросов безопасности и надежности
- Высокая производительность и поддержка хранения больших объемов информации.
- Масштабируемость системы.

Запросы на получение и манипуляцию данными.

- Доступ к данным разрешен только путем исполнения текстовых запросов на языку SQL.
- Язык SQL является языком декларативного программирования (мы описываем результат, а не последовательность действий, приводящих к его получению).
- Все СУБД поддерживают начальную часть SQL92 и дополнительные несовместимые расширения SQL.
  Microsoft: Transact SQL
  Oracle: PL/SQL
  IBM: DB2 SQL

Проблема №1.

Клиент

№ Клиента (PK)	ФИО	Дата рождения
13	Петров	12.08.1975

Вставка нового клиента (№ Клиента:13, ФИО:Сидоров,
Дата рождения: 11.03. 1982)

4.4

НЕПРОТИВОРЕЧИВОСТЬ (CONSISTENCY)

- Целостным состоянием БД называется состояние при котором выполняются все явные и неявные ограничения.
- *Все операции должны переводить базу данных из одного целостного состояния в другое*.
- Операции, нарушающие целостность БД не фиксируются (т.е. результаты их выполнения не сохраняются в БД).

Проблема № 2.

Счет

№ счета (PK)	№ клиента (FK)	№ валюты (FK)	Сумма
1	13	840	25 000
2	7	840	13 500
3	13	978	10 000

Операция начисления зарплаты:

1. Счет №2. Сумма := Сумма + 3500

СБОЙ

ДОЛГОВЕЧНОСТЬ (DURABILITY)

Если операция закончилась успешно, то результаты ее выполнения должны быть зафиксированы в БД вне зависимости от последующих аппаратных или программных сбоев.

Операция начисления зарплаты:
1. Счет №2. Сумма := Сумма + 3500
2. Сохранение результата в БД.
3. Подтверждение успешности выполнения операции

СБОЙ

Проблема №3

Счет

№ счета (PK)	№ клиента (FK)	№ валюты (FK)	Сумма
1	13	840	25 000
2	7	840	13 500
3	13	978	10 000

Операция перевода денег клиентом № 13 с долларового счета в евро:

1. Счет № 1. Сумма := Сумма — 5000

2. Новая_Сумма_в Евро := Конвертация 5000 долларов в евро.

СБОЙ

3. Счет №3. Сумма =Сумма + Новая_Сумма_в Евро.

АТОМАРНОСТЬ (ATOMICITY)

Атомарный блок операций должен выполнится целиком. Если в ходе выполнения произошла ошибка или сбой, то результаты всех операций блока должны быть отклонены

1. Начать атомарный блок.

2. Счет № 1. Сумма := Сумма — 5000

3. Новая_Сумма_в Евро := Конвертация 5000 долларов в евро.

СБОЙ

4. Счет №3. Сумма =Сумма + Новая_Сумма_в Евро.

5. Завершить атомарный блок.

Проблема №4

Счет

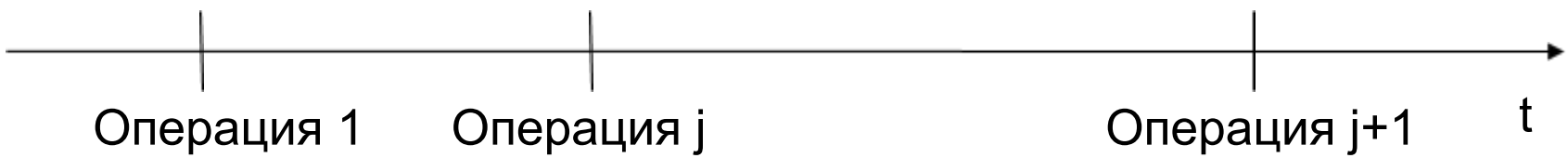
№ счета (PK)	№ клиента (FK)	№ валюты (FK)	Сумма
1	13	840	25 000
2	7	840	13 500
3	13	978	10 000

Операция снятия денег со счета:
1. Начать атомарный блок
2. Счет №2.Сумма = Сумма - 1000
.
.
.
Ошибка.
Отмена атомарного блока.
.
.
N. Конец атомарного блока

Операция снятия денег со счета:
1. Начать атомарный блок
.
.
i. Счет №2.Сумма = Сумма - 300
.
.
.
N. Конец атомарного блока

ИЗОЛИРОВАННОСТЬ (ISOLATION)

График выполнения операций — расположение операций на временной шкале по времени выполнения



Результат выполнения операций не должен зависеть от графика выполнения операций.

Свойства операций над базой данных

4.10

ТРАНЗАКЦИЯ

Единица работы в БД, объединяющая набор операций, гарантирующая соблюдение принципов атомарности, непротиворечивости, изоляции и долговечности работы операций.

Транзакция – это последовательность команд SQL, которые БД обрабатывает как единое целое (Все результаты транзакции или целиком сохраняются (фиксируются), или.целиком отменяются (откатываются назад))

ACID

Atomicity **C**onsistency **I**solation **D**urability

SQL

BEGIN TRANSACTION — начать транзакцию

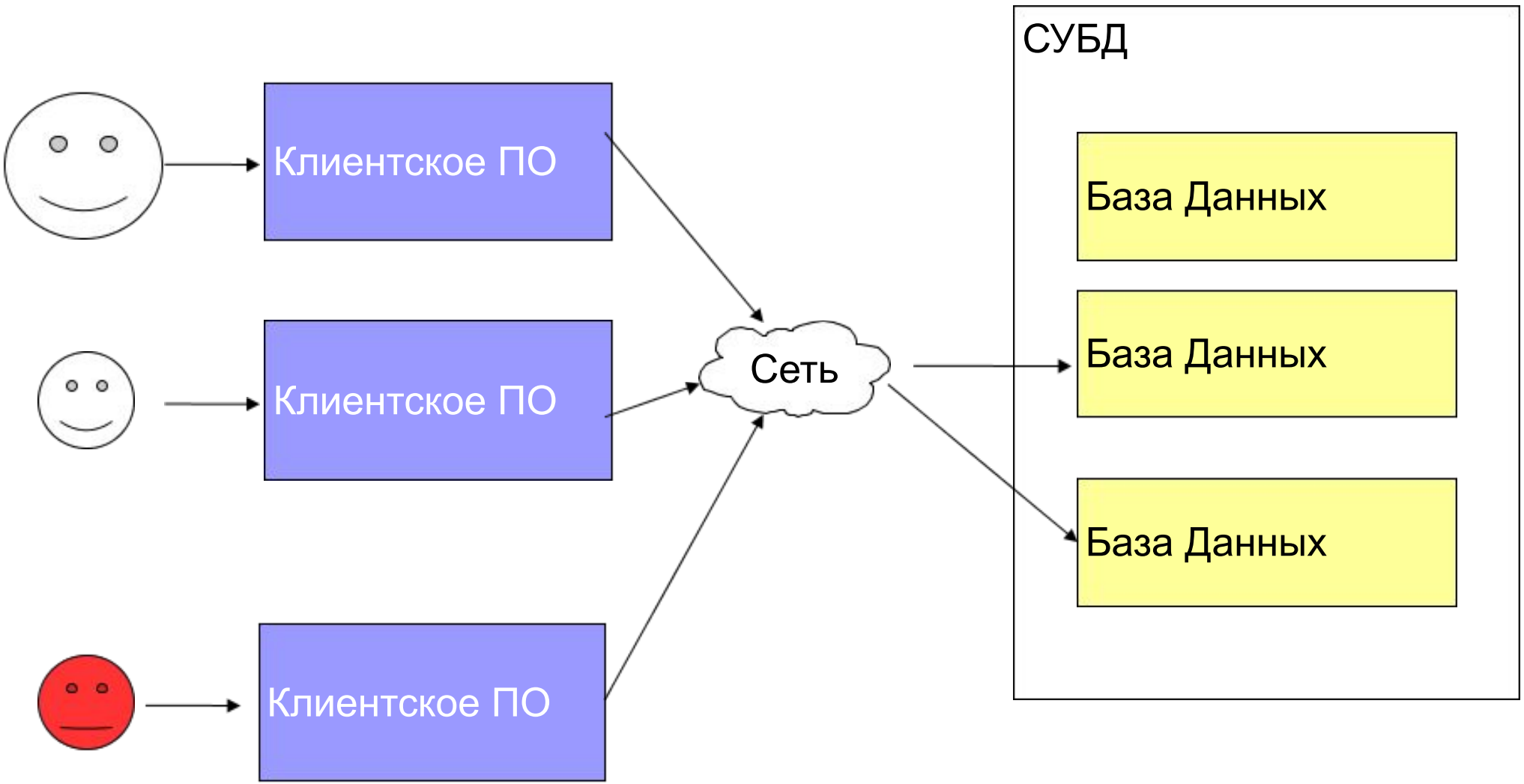
COMMIT — зафиксировать работу транзакции

ROLLBACK — откатить транзакцию (отменить транзакцию)

Требования к СУБД:

- Наличие стандартизованных средств проектирования структуры БД, устойчивой к проблемам хранения информации.
- Наличие стандартизованных средств построения запросов и манипулирования данными.
- Обеспечение вопросов безопасности и надежности
- Высокая производительность и поддержка хранения больших объемов информации.
- Масштабируемость системы.

Информационная система на основе базы данных



Контроль доступа (Access Control)

- Информационная система предназначена для работы различных категорий пользователей (от операторов ввода данных до генерального директора).
- Каждая категория пользователей имеет определенный уровень доступа к информации.
- Создание *пользователей*(users) и ролей(*roles*) в терминологии СУБД позволяет разграничить доступ к данным.
- Разграничение доступа ведется на уровне выдачи привилегий на выполнение **CRUD** операций (**C**reate **R**ead **U**pdate **D**elete) для каждого объектов БД (таблицы, схемы, процедуры и т.д.)

Аутентификация (Autentification)

- Пользователи клиентского ПО должны быть сопоставлены с зарегистрированными *пользователями БД* и ролями.
- Логин и пароль как средства подтверждения факта, что текущий клиент действительно является зарегистрированными *пользователем БД*.
- Хранение паролей, даже в зашифрованном виде является плохой практикой. Вместо пароля хранится хеш пароля.
- Хеш — односторонняя функция, не позволяющая восстановить значение аргумента по результату работы. (MD5, SHA-2, ГОСТ Р 31.11-94)

Шифрование (Encryption)

- Контроль доступа отвечает только за доступ через стандартные средства СУБД (SQL запросы).
- Данные физически хранятся в файле доступ к которому регламентируется средствами ОС и лежит вне области контроля СУБД.
  (Уязвимости несанкционированного чтения и модификации данных)
- Шифрование физического представления данных позволяет частично снизить зависимость от защищенности ОС.
- Шифрование не должно использоваться как средство контроля доступа.
- Основные алгоритмы: 3DES, AES и их вариации.
  Возможно использование как симметричных,
  так и ассиметричных протоколов

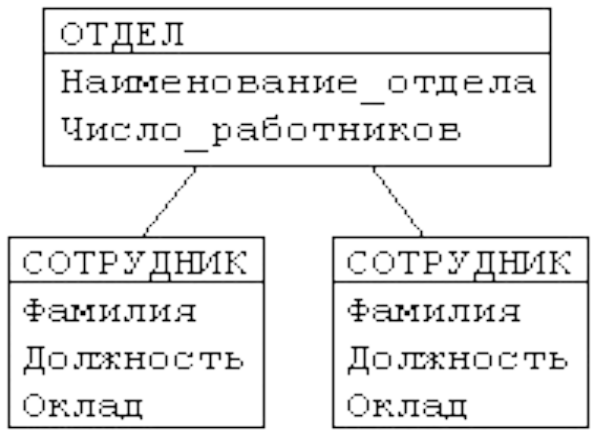
Аудит (Auditing)

- Человек, - самое уязвимое звено любой системы безопасности
- Администратор базы данных (Database Administrator, DBA), - человек с очень высоким уровнем доступа к базе данных.
- Максимально возможный доступ к БД является необходимым для работы администратора, однако сильно снижает защищенность системы
- Подсистема аудита фиксирует действия всех пользователей и является средством с доступом только на чтение для администратора БД.

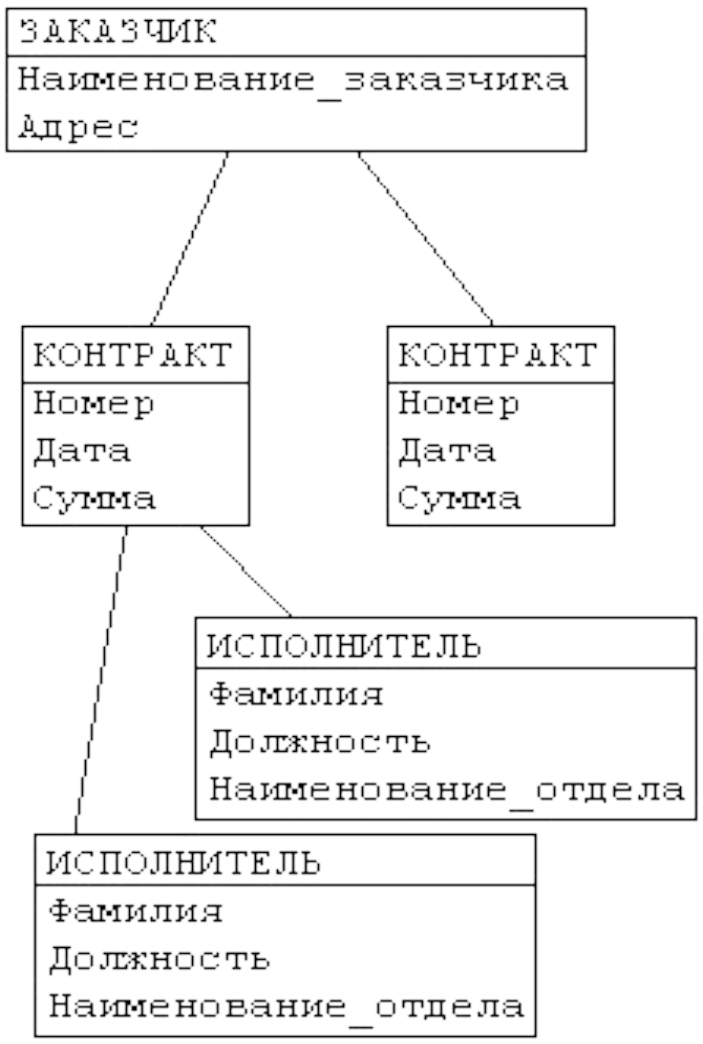
6.1 Модели данных

Модель данных	Преимущества	Недостатки	Программные продукты
Файлы	- Простота - Удобство для простых задач. - Встроенные средства доступа в большинстве языков. программирования - Бесплатность	Основные проблемы, возникающие при работе с данными не решены: - Рассинхронизация семантики и представления. - Дублирование информации	Любая файловая система
Навигационные СУБД	- Хранение и доступ к метаданным - Поддержка транзакций	Доступ к данным требует полной спецификации пути к ним	IBM’s IMS (www.ibm.com)
Реляционные СУБД	- Развитая математическая база. - Наличие крупных поставщиков. - Наличие стандартов для языка запросов (SQL) и организации внешнего доступа ODBC.	Трудность в построении концептуальной модели.	Oracle, Sybase, IBM DB2, Microsoft SQL Server MySQL, PostgreSQL

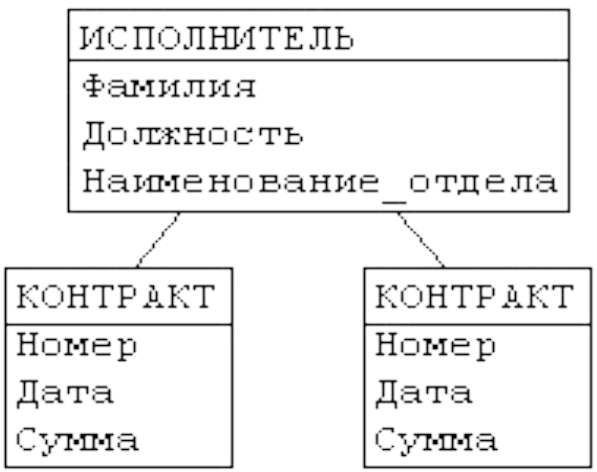
ИЕРАРХИЧЕСКАЯ МОДЕЛЬ ДАННЫХ



(a)



(b)



(c)

6.3 Модели данных

Модель данных	Преимущества	Недостатки	Программные продукты
Объектно-ориентированные СУБД	- Прозрачная работа с объектами - Удобство работы с БД заточенными под конкретные приложения - Единый подход к хранению и обработке данных. - Более гибкая система по сравнению с реляционной.	- Отсутствие общепринятых стандартов (OQL только развивается) - Развивающаяся технология	Computer Associate's Jasmine (www.ca.com) Versant Developer Suite (www.versant.com) Object Design's ObjectStore (www.objectdesign.com) Objectivity/DB (www.objectivity.com) Poet (www.poet.com)
XML СУБД	- Прозрачная поддержка XML. - Снижение затрат на парсинг XML.	- Возвращение к проблемам навигационных СУБД. - Больший объем трафика. - Проблемы производительности	dbXML (www.dbxml.org) OpenLink Software's Virtuoso (www.openlinksw.com) Software AG's Tamino (www.softwareag.com) X-Hive/DB (www.x-hive.com)

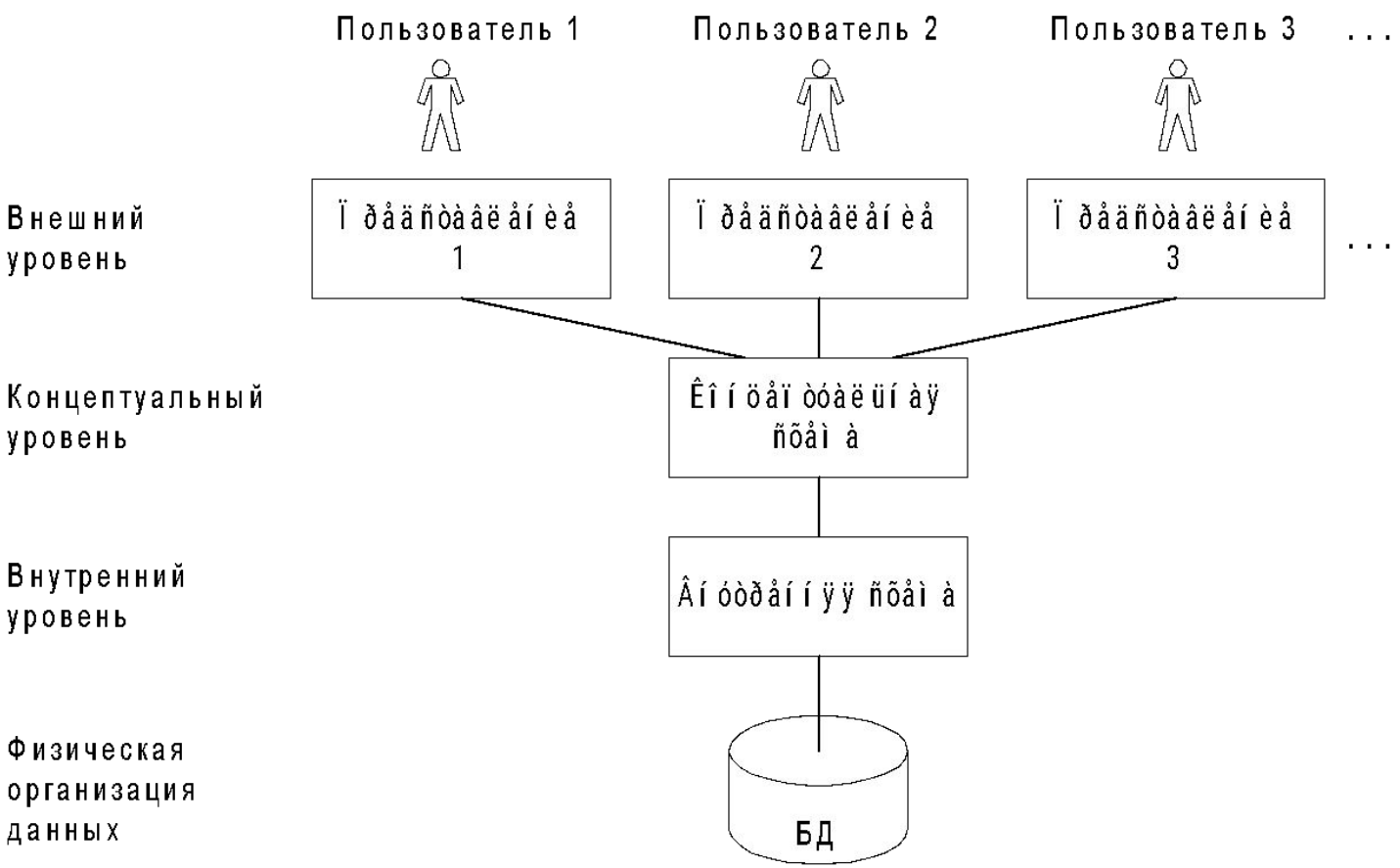
6.4 Модели данных

Модель данных	Преимущества
Файлы	Хранение настроек приложения, хранения журнала событий, легкость копирования
Навигационные СУБД	Поддержка уже существующих систем
Реляционные СУБД	Поддержка средних и больших объемов данных с транзакционным доступом. Поддержка средних и больших объемов данных. Реализация высоконадежных, хорошо масштабируемых систем, работающих в режиме 24/7 с высоким объемом операций
Объектно-ориентированные СУБД	Реализация сложных структур данных с большим количеством взаимосвязей. Поддержка сложных запросов
XML СУБД	Поддержка гетерогенных систем

Трехуровневая архитектура ANSI/SPARC

ANSI – American National Standard Institute,

SPARC – Standards Planning and Requirements Committee



Общая характеристика моделей данных

Модель данных *– это интегрированный набор понятий для описания данных, связей между ними и ограничений, накладываемых на данные в некоторой организации.*

МОДЕЛЬ ДАННЫХ

Сильно типизированные

Слабо типизированные

Компоненты:

- категория
- свойства категории
- связи между категориями

Схема:

ВОДИТЕЛЬ (*Имя*, *Возраст*, *Стаж работы*)

АВТОМОБИЛЬ (*Модель*, *Гос. номер*, *Дата приобретения*)

УПРАВЛЯЕТ (ВОДИТЕЛЬ, АВТОМОБИЛЬ)

Множества: домены, атрибуты

Множество *– это собрание правильно идентифицированных объектов, удовлетворяющих правилу принадлежности*

МНОЖЕСТВО

Интенсионал(intentional)

{2, 8, 16, 46}, {12, 10, 8, 100, 32}, {2, 4, 8, 16, 32, 64}

Экстенсионал

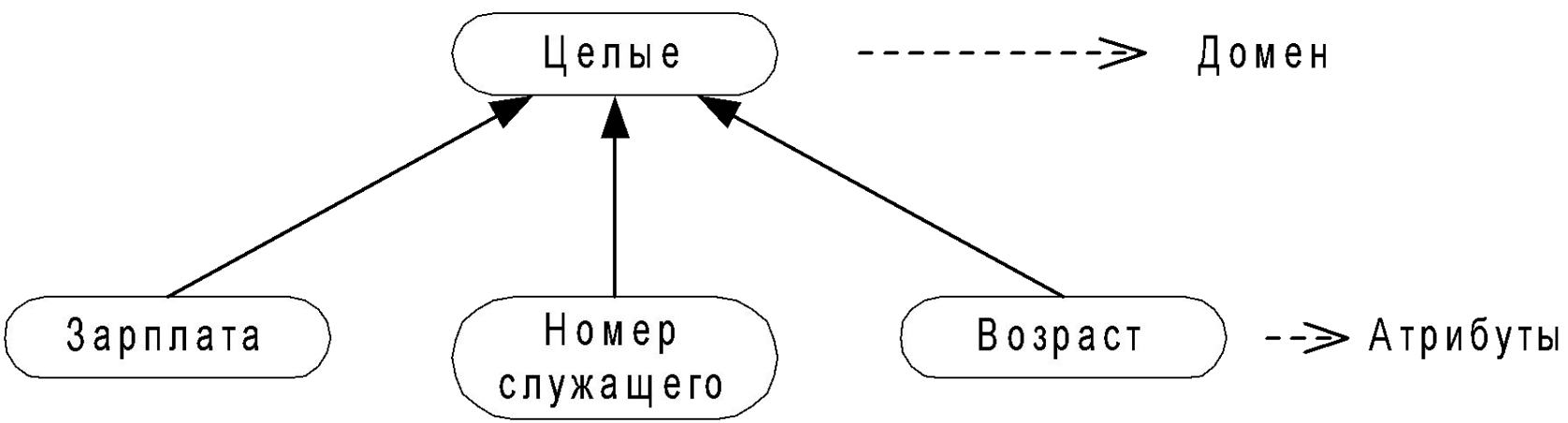
{2, 4, 8, 16, 32, 64}

Домены *– это множества, элементы которых более или менее однородны.*

Множества: домены, атрибуты

Атрибуты *– это именованные домены, представляющие семантически значимые объекты.*

Атрибуты определяются на доменах и представляют собой интенсионал именованного домена. Значения атрибута – это экстенсионалы.



Отношения: сущности

Агрегат, построенный на множествах, определяется как ***отношение***

Пусть дана некоторая совокупность доменов D1, D2, ..., Dm, не обязательно различных. Отношение, определенное на доменах D1, D2, ..., Dm, есть множество упорядоченных кортежей <d1, d2, ..., dm>, таких, что d1 ∈ D1, d2 ∈ D2, ..., dm ∈ Dm.

Отношения: сущности

Пример:

Даны множества:
D1 = {d1i | d1i – строчная буква английского алфавита} – интенсионал множества, его экстенсионал, например, {a, b, c, d, e}
D2 = {d2j | d2j – десятичная цифра} – интенсионал множества, его экстенсионал, например, {1, 3, 5}
Определим на этих доменах отношение R:
R = {<d1i, d2j> | d1i ∈ D1, d2j ∈ D2} – интенсионал отношения; задает двух символьные кортежи, в которых первый символ – буква, второй – десятичная цифра. Экстенсионалом данного отношения может быть конкретное множество
R1 = {<a,3>, <a,1>, <c,1>}.

	D1
d1i	a
.	b
.	c
.	d
dm	e

	D2
d2j	1
.	2
.	3
.	4
dm	5

R1

	D1	D2
d1	a	3
d2	a	1
d3	c	1

Отношения: сущности

Степень отношения *(или арность кортежа) – характеристика, относящаяся к интенсионалу отношения; количество образующих данное отношение множеств.*

Мощность отношения *– характеристика, относящаяся к экстенсионалу отношения; количество элементов в конкретной реализации отношения*

Модель данных	Реляционная база данных
Отношение	Таблица
Заголовок отношения	Заголовок таблицы
Атрибут отношения	Наименование столбца таблицы
Кортеж отношения	Строка таблицы
Степень (-арность) отношения	Количество столбцов таблицы
Мощность отношения	Количество строк таблицы
Домены и типы данных	Типы данные в ячейках таблицы

Отношения: сущности

Схема отношения *– это именованный список пар <имя атрибута>:<имя домена>, имя которого задает имя отношения: R(A1:D1, A2:D2, ..., Am:Dm).*

Отношения: связи

*Агрегат, построенный на других отношениях, рассматривается как **связь между этими отношениями***

$R = \{<s1i, s2j> \mid s1i \in S1, s2j \in S2\}$

Два отображения:
прямое – R : S1 □ S2
обратное – R-1 : S2 □ S1

***Кардинальное число** отображения определяется количеством элементов одного множества, связанных с одним элементом другого множества.*

R (S1 (m1, n1) : S2 (m2, n2))

*каждый элемент из **S1** связан минимум с m2, максимум с n2 элементами из S2,*
*каждый элемент из **S2** связан минимум с m1, максимум с n1 элементами из S1.*

Отношения: связи

Минимальное и максимальное кардинальные числа не определены:
R (S1 (0, ∞) : S2 (0, ∞)), или R (S1 : S2).

Типы отображений:

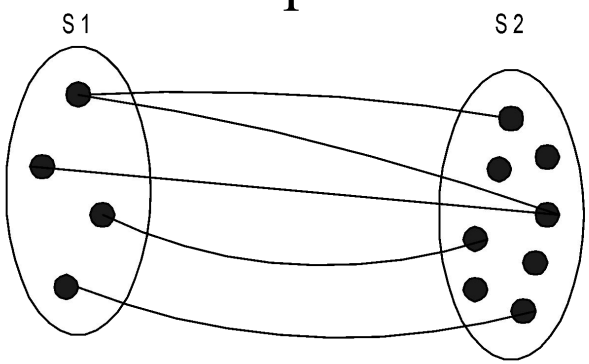


Рис. 1. Полностью определенное отображение на S1

R (S1 (0, ∞) : S2 (1, ∞))

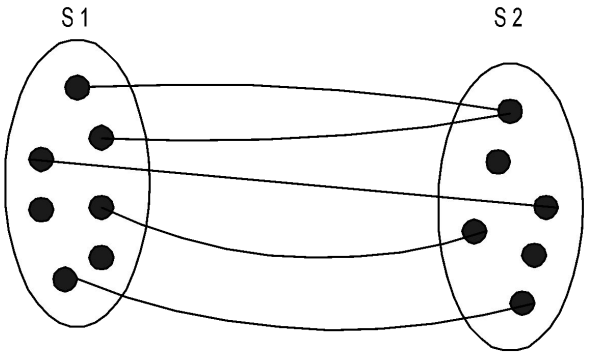


Рис. 2 Неполное функциональное отображение

R (S1 (0, ∞) : S2 (1, 1))

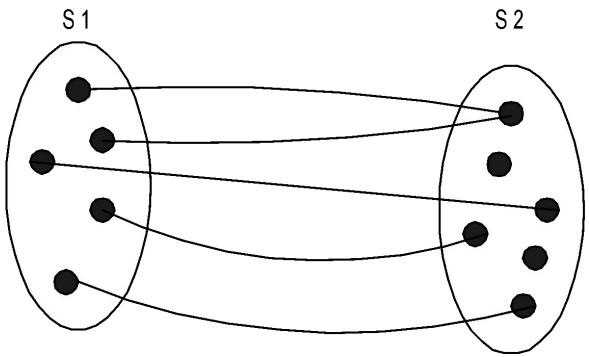


Рис. 2. Полное функциональное отображение

(S1 (0, ∞) : S2 (0, 1))

Связи, для которых хотя бы одно отображение является функциональным (полным или неполным), часто называют связями типа **1 : n,** или **«один ко многим».** Связи, в которых оба отображения являются нефункциональными, называют связями типа **n : n**, или **«многие ко многим».**

Общая характеристика ограничений целостности

Структурными компонентами модели данных являются отношения и связи между ними;
Отношения задаются своими схемами;
Схема отношения определяется через атрибуты, определенные на доменах.

Логические ограничения, накладываемые на данные, называются ***ограничениями целостности***.

Ограничения целостности реализуется средствами языка описания ограничений (ЯОО, или CDL – Constraints Definition Language)

Реляционная модель данных

Реляционная модель данных (РМД) была разработана сотрудником IBM Э. Ф. Коддом (E.F. Codd) в 1969-70 г.г. на основе математической теории отношений.

Реляционная модель данных

Структурная и целостная части

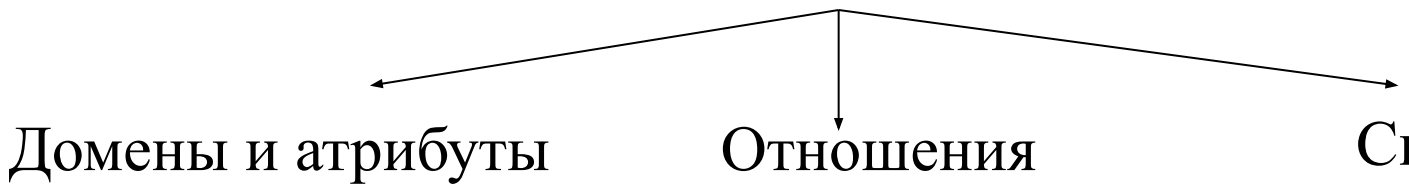
Манипуляционная часть

Язык описания данных (ЯОД)
Data Definition Language (DDL)

Язык манипулирования данными (ЯМД)
Data Manipulation Language (DML)

Особенности реляционной модели данных:
определена манипуляционная часть – конкретный набор операций, функциональные возможности,
имеются конкретные языки описания данных, ограничений, накладываемых на данные, и манипулирования данными,
современные реляционные СУБД используют единый язык – SQL, в котором объединены и ЯОД, и ЯМД.

Базовые структурные компоненты реляционной модели данных



Домен *– множество элементов одного типа.*

Пример:

Простой домен: ГОД = {1985, 2003, 2000}; ДЕНЬГИ = {500, 1000, 850}

Составной домен: ИСТОРИЯ ЗАРПЛАТЫ = {{<1985, 500>, <2000, 1000>}, {<2000, 850>}, {<1985, 850>, <2000, 500>, <2003, 1000>}}

Пусть дана совокупность множеств D1, D2, …, Dn, не обязательно различных. Тогда отношение R, определенное на этих множествах, есть множество упорядоченных кортежей<d1, d2, …, dn> таких, что di $\in$ Di для каждого i из [1:n].

В реляционной модели данных множества Di представляют собой домены.

Базовые структурные компоненты реляционной модели данных

Пример:

Даны два домена D1 = {a, b, c} и D2 = {1, 2}.

Отношение R1 = {<a, 2>, <c, 1>}. Отношение R2 = {<a, 2>, <b, 2>, <a, 1>}.

Свойства отношения:
кортежи отношения не упорядочены,
домены внутри кортежей упорядочены.

Атрибуты *задают способ использования домена внутри отношения.*

Схема отношения *– это именованная совокупность пар <имя атрибута : имя домена>.*
Пример: *ОТДЕЛ (Номер отдела: ЧИСЛО, Название: СТРОКА)*

Базовые структурные компоненты реляционной модели данных

В задании схемы отношения могут использоваться и составные домены.

Пример, реализации (экстенсионал) отношения СОТРУДНИК :

Номер сотрудника	Имя	Зарплата
1	Иванов	1985, 500 2000, 1000
2	Петров	1985, 850 2000, 500 2003, 1000

В данном примере значением одного элемента составного домена является множество пар вида<ГОД, ДЕНЬГИ>

Нормализованное отношение *– это отношение, в котором каждое значение атрибутов является атомарным.*

Базовые структурные компоненты реляционной модели данных

Нормализация: составной домен заменяется составляющими его простыми доменами, а в реализации отношения значения атрибутов, определенных на других доменах, повторяются для каждого кортежа составного домена

Номер сотрудника	Имя	Зарплата
1	Иванов	1985, 500
1	Иванов	2000, 1000
2	Петров	1985, 850
2	Петров	2000, 500
2	Петров	2003, 1000

Свойства реляционной модели данных

1. *Каждый атрибут отношения имеет уникальное в данном отношении имя.*
2. *Каждый атрибут определен на каком-то одном домене.*
3. *На одном и том же домене может быть определено несколько атрибутов.*
4. *Имя атрибута может совпадать с именем домена.*
5. *Порядок следования атрибутов не устанавливается.*
6. *В отношении нет совпадающих кортежей (каждый кортеж уникален).*
7. *Порядок следования кортежей не устанавливается.*
8. *Отношение имеет имя, которое в схеме базы данных отличается от имен всех других отношений.*

Представление сущности

Представление сущности *означает возможность уникальной идентификации каждого отдельного кортежа отношения по значениям его атрибутов*

Ключ *– это совокупность атрибутов, которая однозначно идентифицирует каждый кортеж данного отношения.*

Первичный ключ (PK – Primary Key) *– не избыточный набор атрибутов, значения которых однозначно определяют кортеж отношения.*

Первичный ключ не избыточен, если:

состоит из одного атрибута,

состоит из нескольких атрибутов, но ни один из этих атрибутов не является лишним для однозначной идентификации каждого кортежа.

Представление сущности

Отношение может иметь только один первичный ключ.
Если в отношении можно выделить несколько наборов атрибутов для первичного ключа, выбирается один, остальные определяются как альтернативные **(AK – Alternate Key).**

Пример отношения:

КАФЕДРА (*Номер кафедры*, *Название*)

Номер кафедры – PK

Название - AK

Связи

Связи между сущностями отражают взаимосвязи между конкретными экземплярами сущностей. Эти взаимосвязи представляются с помощью внешних ключей

Внешний ключ (FK – Foreign Key) *– это атрибут или некоторое множество атрибутов отношения R1, которые не являются собственными атрибутами отношения R1, но их значение совпадает со значениями первичного ключа некоторого отношения R2 (возможность идентичности R1 и R2 не исключается).*

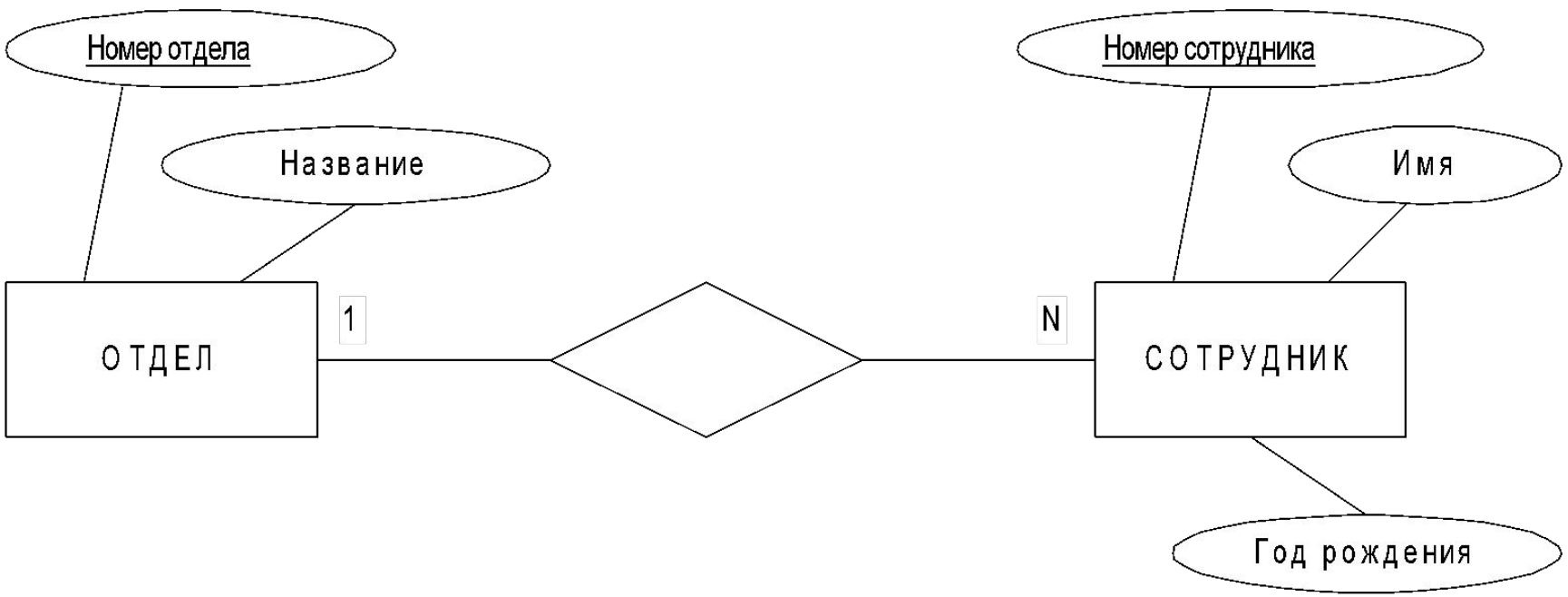
Основными типами связей между сущностями являются: 1 : n и n : n

Связи

Пример, есть отношения:

СОТРУДНИК с атрибутами *Номер сотрудника* (первичный ключ), *Имя* и *Год рождения*

ОТДЕЛ с атрибутами *Номер отдела* (первичный ключ) и *Название*.



Связи

Схемы отношений:

ОТДЕЛ (*Номер отдела*, *Название* (АК))

СОТРУДНИК (*Номер сотрудника*, *Имя*, *Год рождения*, *Номер отдела* (FK))

ОТДЕЛ

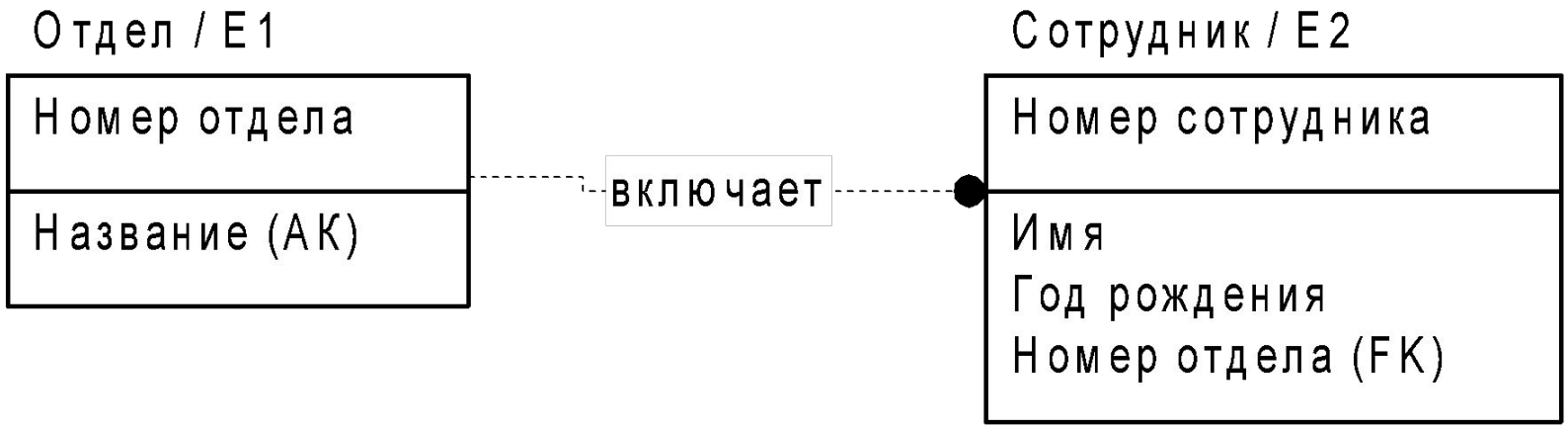
Номер отдела	**Название**
Первичный ключ	*Альтернативный ключ*
1	Бухгалтерия
2	АХО
3	Библиотека

СОТРУДНИК

Номер сотрудника	**Имя**	**Год рождения**	**Номер отдела**
Первичный ключ			*Внешний ключ*
1	Иванов	1953	1
2	Петров	1970	3

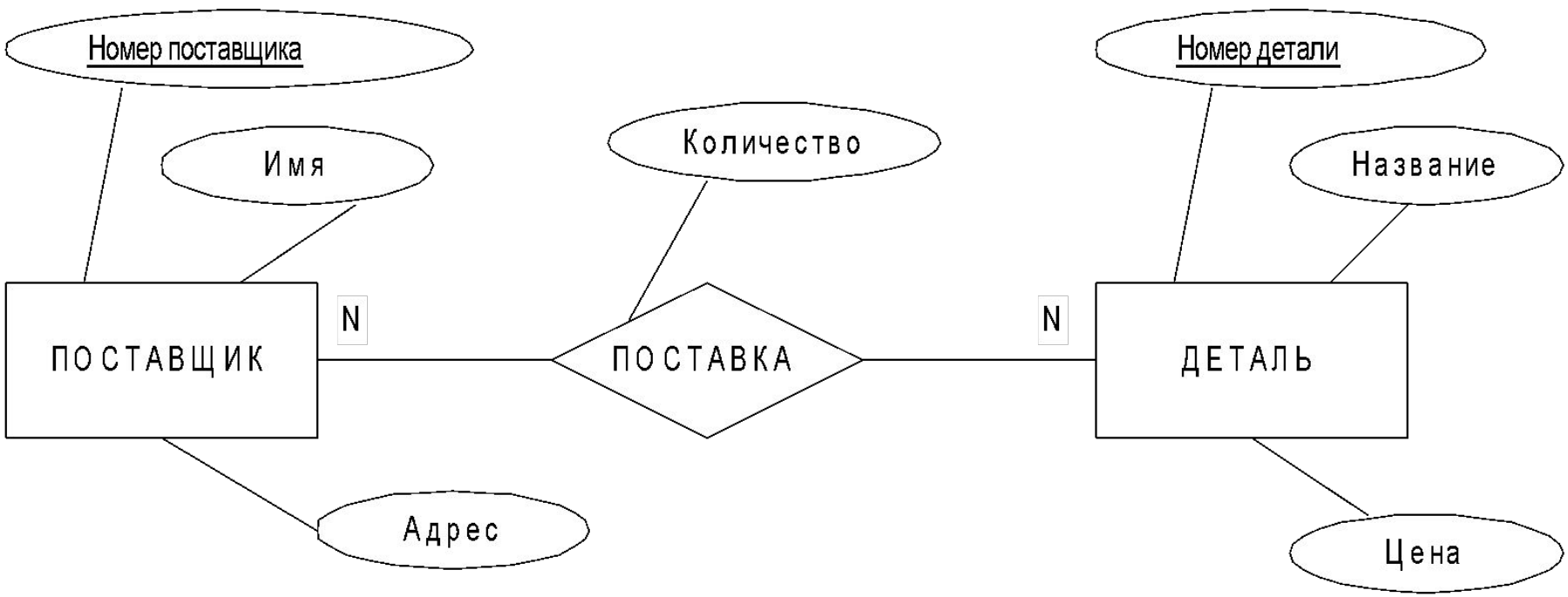
Связи

Представление связи 1:n в IDEF1X:



Связи

Пример, есть отношения:
ПОСТАВЩИК с атрибутами *Номер поставщика* (первичный ключ), *Имя* и *Адрес*, и отношение ДЕТАЛЬ с атрибутами *Номер детали* (первичный ключ), *Название* и *Цена*.



Связи

Схемы отношений :

ПОСТАВЩИК (*Номер поставщика*, *Имя*, *Адрес*)

ДЕТАЛЬ (*Номер детали*, *Название*, *Цена*)

ПОСТАВКА (*Номер поставщика* (FK1), *Номер детали* (FK2), *Количество*)

Поставщик

№ поставщика	*Имя*	*Адрес*
PK		
S1	Иванов	Москва
S2	Сидоров	С.Петербург
S3	Петров	Тюмень

Деталь

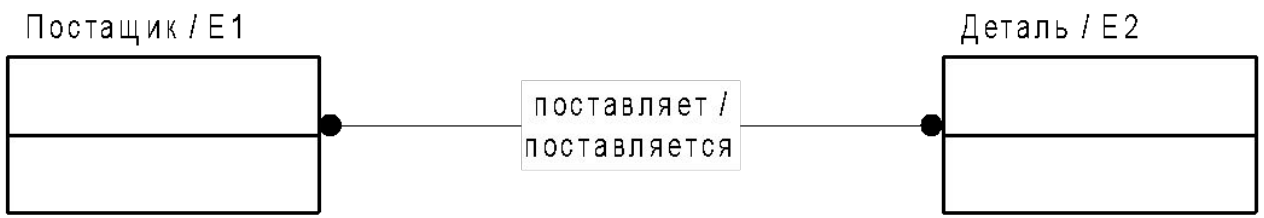
Номер детали	*Название*	*Цена*
PK		
P1	Болт	18
P2	Винт	14
P3	Гайка	10

Поставка

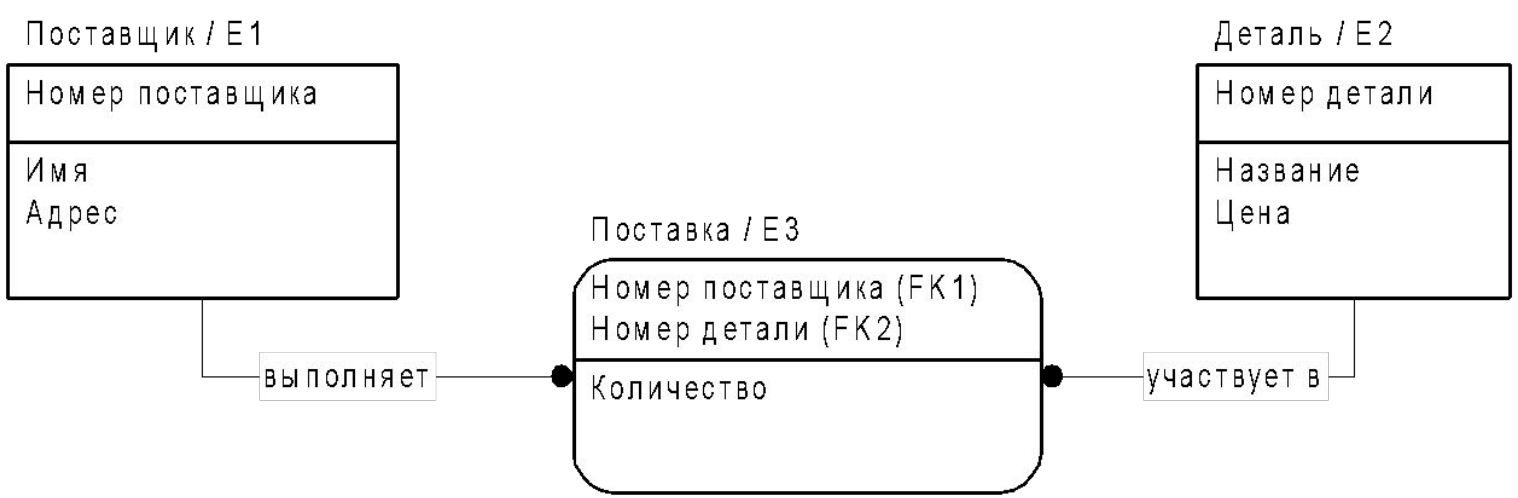
Первичный ключ отношения связи		*Количество*
Номер поставщика	*Номер детали*	
Внешний ключ отношения ПОСТАВЩИК	Внешний ключ отношения ДЕТАЛЬ	Собственный атрибут связи
S1	P1	100
S1	P2	200
S2	P3	150

Связи

Представление и разрешение связи n:n в IDEF1X:



а) неопределенная связь



б) преобразование в определенную связь

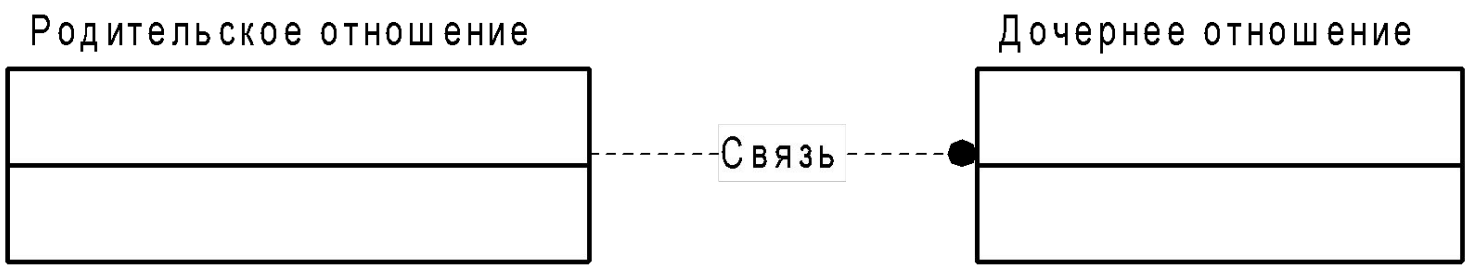
Ссылочная целостность

Пример отношения:

ОТДЕЛ (*Номер отдела*, *Название* (АК))

СОТРУДНИК (*Номер сотрудника*, *Имя*, *Год рождения*, *Номер отдела* (FK))

***Требование ссылочной целостности**: значение атрибута внешнего ключа в любом кортеже дочернего отношения должно соответствовать значению атрибута первичного ключа в некотором кортеже родительского отношения.*



Языковые средства описания данных

Язык определения данных (DDL) для реляционной модели данных включает следующие возможности:

- *создание домена,*
- *создание отношения,*
- *определение ограничений целостности.*

Соответствие между компонентами РМД и элементами базы данных

Структурный компонент реляционной модели данных	**Элемент реляционной базы данных**
Отношение	Таблица
Домен	Тип данных
Атрибут	Колонка таблицы
Кортеж отношения	Строка таблицы

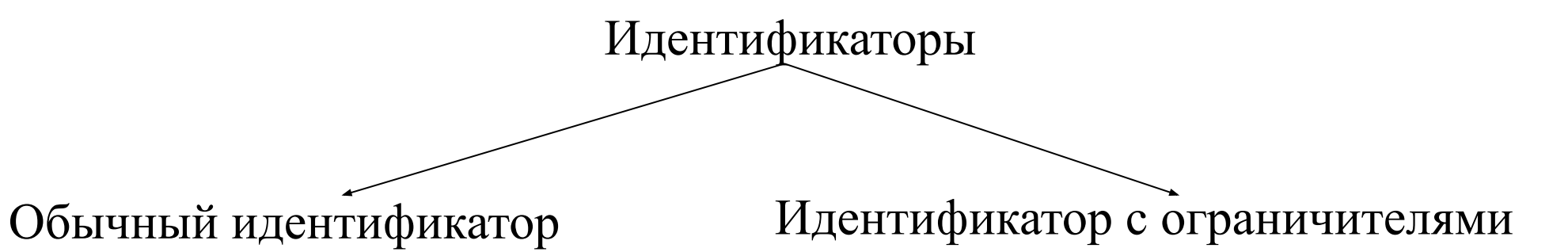
Типы ограничений целостности

Тип ограничения целостности	Представление средствами SQL
Уникальность значений	PRIMARY KEY или UNIQUE
Обязательность значений	NULL или NOT NULL
Допустимость значений	CHECK
Ссылочное ограничение	FOREIGN KEY

Основные предложения DDL(ЯОД):

CREATE *тип_объекта* – создать соответствующий объект базы данных,
ALTER *тип_объекта* – изменить соответствующий объект базы данных,
DROP *тип_объекта* – удалить указанный объект базы данных.

Способы именования объектов



Последовательность букв, цифр и символов подчеркивания (_), начинающаяся с буквы. Не чувствителен к регистру. Не совпадает с зарезервированными словами.
Примеры: A12 Tab_Name DeptID

Текст заключен в двойные кавычки. Чувствительны к регистру, можно использовать зарезервированные слова.

Примеры: ”WKLY-SAL” ”Department Name” ”UNION”

Типы данных

Числовые *- используются для представления целых, вещественных и десятичных чисел.*

Название	Диапазон	Описание
SMALLINT	от -32 768 до 32 767 (от -215 до 215 – 1).	короткое целое; двух байтное целое
INTEGER (INT)	от -2 147 483 648 до 2 147 483 647 (от -231 до 231 – 1)	целое; четырех байтное целое
REAL	$1{,}175*10^{-37}$ до $3{,}402*10^{+38}$.	вещественные числа обычной точности
DECIMAL (*p,s*)	от $-10^{31} + 1$ до $10^{31} - 1$	Точность *p* задает общее количество цифр в записи числа, $p \leq 31$. Масштаб *s* задает количество десятичных цифр в дробной части

Типы данных

Строковые типы данных *используются для представления символьных строк*

Название	Диапазон	Описание
CHARACTER[(*len*)] (**CHAR**[(*len*)])	$1 \le len \le 254$	символьные строки фиксированной длины
VARCHAR[(*len*)]	$1 \le len \le 32672$	строки переменной длины

Типы данных дата – время *используются для представления даты и времени*

Название	Диапазон	Описание
DATETIME	От 00:00:00 до 23:59:590,997	Значение по умолчанию 1900-01-01 00:00:00
TIMESTAMP	1970-01-01 00:00:00 — 2038-12-31 00:00:00	Внутреннее представление даты и времени.

Основные операции

Операция конкатенации (CONCAT или || или +) *- можно использовать с операндами строкового типа. Операция выполняет сцепление двух строковых операндов и образует строковое выражение.*

Арифметические операции *можно использовать с операндами числовых типов.*

Бинарные операции (/, *, +, -) определяют сложение, вычитание, умножение и деление, соответственно

Операции над датой и временем*. Значения даты и времени можно увеличивать, уменьшать и вычитать. Эти операции могут включать десятичные числа –* ***продолжительность****.*
Продолжительность *– это число, представляющее интервал времени.*

Создание таблиц

CREATE TABLE *имя_таблицы*
(
имя_колонки *тип_данных* [*ограничение_обязательности*]
[*ограничение_целостности_на_колонку*]
[*спецификация_генерируемого_значения*]
[, …]
[, *ограничение_целостности_на_таблицу*]
[, …]
)

имя_таблицы – идентификатор, задает имя таблицы;
имя_колонки – идентификатор, уникальный в пределах данной таблицы;
тип_данных – тип данных для колонки;
ограничение_обязательности – задает ограничение обязательности значения данной колонки;
ограничение_целостности_на_колонку – ограничение целостности, накладываемое на данную колонку;

Создание таблиц

ограничение_целостности_на_таблицу – ограничение целостности, накладываемое на одну или несколько колонок создаваемой таблицы.
генерируемое_значение – значения по умолчанию, если в операции вставки значение данной колонки не указано

Ограничения целостности:

[**CONSTRAINT** *имя_ограничения*] *ограничение*

имя_ограничения – задает имя ограничения. Уникально в пределах данной таблицы;

ограничение – задает конкретное ограничение целостности.

Создание таблиц. Правила задания ограничений

Ограничения обязательности

NULL **NOT NULL**

Ограничения уникальности(всегда Not Null)

UNIQUE **PRIMARY KEY**

Ограничение уникальности на колонку : **UNIQUE** или **PRIMARY KEY**.
Ограничение уникальности на таблицу : **UNIQUE(***имя_колонки*, …**)** или **PRIMARY KEY(***имя_колонки*, …**).**
UNIQUE **–** определяет альтернативный ключ**,** может быть несколько
PRIMARY KEY **–** определяет первичный ключ, может быть только одно

Создание таблиц. Правила задания ограничений

Ограничение допустимости значений

CHECK(*условие***)**

Условие, записанное в ограничении на колонку, может содержать имя только данной колонки;

Условие, записанное в ограничении на таблицу, может содержать любые (одно или более) имена колонок из данной таблицы.

Условие

Предикаты:
BETWEEN, IN, LIKE
Результат предиката:
TRUE, FALSE, NULL

Логические операции:
NOT, AND, OR

Создание таблиц. Правила задания ограничений

Правила вычисления логических операций

P	Q	NOT P	P AND Q	P OR Q
true	true	false	true	true
true	false	false	false	true
true	NULL	false	NULL	true
false	true	true	false	true
false	false	true	false	false
false	NULL	true	false	NULL
NULL	true	NULL	NULL	true
NULL	false	NULL	false	NULL
NULL	NULL	NULL	NULL	NULL

Создание таблиц. Правила задания ограничений

Основной предикат:
выражение операция_отношения выражение

операции_отношения: операции = (равно), <> (не равно), < (меньше), > (больше), <= (меньше или равно), >= (больше или равно).

Предикат BETWEEN
выражение1 [**NOT**] **BETWEEN** *выражение2* **AND** *выражение3*
должно выполняться соотношение*: выражение2* ≤ *выражение3*

Пример:
выражение1 ***BETWEEN*** *выражение2* ***AND*** *выражение3*,
тоже самое что и:
выражение1 >= *выражение2* **AND** *выражение1* <= *выражение3*.

Создание таблиц. Правила задания ограничений

Предикат IN
*выражение [**NOT**] **IN** (список_выражений)*
список_выражений - перечисление выражений через запятую в произвольном порядке

*Результат вычисления **IN**:*
«истина», если значение выражения совпадает хотя бы с одним значением из списка;
«ложь», если значение выражения не совпадает ни с одним значением из списка, и в списке нет неопределенных(NULL) значений;
***не определено(NULL)**,* если значение выражения не совпадает ни с одним значением из списка, и в списке есть неопределенные значения.

*Результат значения **NOT IN*** противоположен ***IN***.

Создание таблиц. Правила задания ограничений

Предикат LIKE

выражение [NOT] LIKE шаблон [ESCAPE escape-символ]

Предикат LIKE проверяет, соответствует ли строка, являющаяся результатом вычисления *выражения*, заданному *шаблону*.

Результат вычисления ***LIKE*** *:*

«истина», если значение выражения удовлетворяет шаблону,

«ложь», если не удовлетворяет,

не определено, если хотя бы один из аргументов имеет значение NULL

Результат значения ***NOT LIKE*** противоположен ***LIKE***.

Шаблон это строка использующая специальные метасимволы:

_ (символ подчеркивания) – соответствует одному (любому) символу,

% – соответствует любой (возможно, пустой) цепочке символов.

Создание таблиц. Правила задания ограничений

Ссылочное ограничение

На колонку:

REFERENCES *имя_родительской_таблицы* [(*имя_колонки_PK*, …)] [**ON DELETE** *правило_удаления*] [**ON UPDATE** *правило_обновления*]

На таблицу :

FOREIGN KEY (*имя_колонки_FK*, …) **REFERENCES** *имя_родительской_таблицы* [(*имя_колонки_PK*, …)] [**ON DELETE** *правило_удаления*] [**ON UPDATE** *правило_обновления*]

имя_колонки_FK – имена колонок внешнего ключа в создаваемой (дочерней) таблице. Порядок перечисления принципиален;

имя_родительской_таблицы – имя родительской таблицы, на которую ссылается создаваемая дочерняя таблица;

имя_колонки_PK – имена колонок первичного ключа в родительской таблице.

Создание таблиц. Правила задания ограничений

Ссылочное ограничение

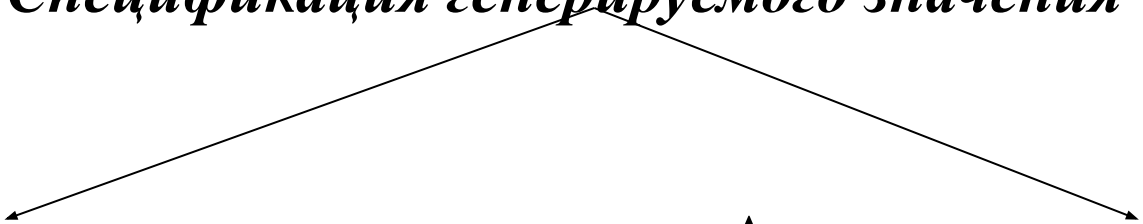
правило_удаления *– определяет, какие действия должны быть выполнены при удалении записи из родительской таблицы. Допускаются значения:* ***RESTRICT ИЛИ NOACTION, CASCADE, SET NULL***

правило_обновления *– определяет, какие действия должны быть выполнены при модификации первичного ключа в родительской таблице. Допускаются значения* ***RESTRICT или NO ACTION***

RESTRICT выполняется перед проверкой всех остальных ограничений
NO ACTION выполняется после проверки всех ограничений

Создание таблиц. Правила задания ограничений

Спецификация генерируемого значения



Значение по умолчанию

DEFAULT *значение*

Автоинкрементное значение

IDENTITY [*опции*]
Опции: начальное значение, шаг, максимальное, минимальное, повторное выполнение

Создание таблиц.

Пример:

Схемы отношений :
ПОСТАВЩИК (*Номер поставщика*, *Имя*, *Адрес*)
ДЕТАЛЬ (*Номер детали*, *Название*, *Цена*)
ПОСТАВКА (*Номер поставщика* (FK1), *Номер детали* (FK2), *Количество*)

б) преобразование в определенную связь

Создание таблиц.

Пример:

Схема отношений :
ДЕТАЛЬ (*Номер детали*, *Название*, *Цена*)

```
CREATE TABLE P(
P_ID INT NOT NULL CONSTRAINT P_PK PRIMARY KEY, --1
PNAME VARCHAR(20) NOT NULL CONSTRAINT P_UQ_01 UNIQUE,--2
PRICE DECIMAL(6,0) NOT NULL CONSTRAINT P_CH_01
CHECK(PRICE > 0)    --3
)
```

1 – ограничения первичного ключа и обязательности значения;
2 – ограничения уникальности и обязательности значения;
3 – условие: значение атрибута должно быть строго положительным.

Создание таблиц.

Пример:

Схема отношений :
ПОСТАВЩИК (*Номер поставщика*, *Имя*, *Адрес*)

```
CREATE TABLE S(
S_ID INT NOT NULL,
SNAME VARCHAR(30) NOT NULL,
ADDRESS VARCHAR(80),
CONSTRAINT S_PK PRIMARY KEY(S_ID) -- 4
)
```

4 – ограничения первичного ключа на таблицу;

Создание таблиц.

Пример:

Схема отношений :
ПОСТАВКА (*Номер поставщика* (FK1), *Номер детали* (FK2), *Количество*)

```
CREATE TABLE SP(
S_ID INT NOT NULL CONSTRAINT SP_FK_01 REFERENCES S, --5
P_ID INT NOT NULL,
QTY INT NOT NULL CONSTRAINT SP_CH_01 CHECK(QTY > 0),
CONSTRAINT SP_PK PRIMARY KEY(S_ID, P_ID),          --6
CONSTRAINT SP_FK_02 FOREIGN KEY(P_ID) REFERENCES P      --7
)
```

5 – внешний ключ(ограничения на колонку)

6 – ограничение первичного ключа(ограничение на таблицу)

7 –определяется внешний ключ (ограничение на таблицу).

Удаление и модификация таблиц.

DROP TABLE *имя_таблицы* – удаление таблицы

Модификация таблиц

ALTER TABLE *имя_таблицы операция* [*операция* …]

Операции:
Добавление новых колонок:

ADD [**COLUMN**] *определение_колонки*
определение_колонки тоже что и в CREATE TABLE

Пример:
ALTER TABLE *EQUIPMENT*
ADD COLUMN *Equip_QTY* ***SMALLINT***
NOT NULL WITH DEFAULT *1*

Удаление и модификация таблиц.

Операции:

Изменение характеристик колонок:

ALTER COLUMN *имя_колонки тип_данных*

Примеры:

ALTER TABLE *EQUIPMENT* ***ALTER COLUMN*** *Equip_Desc* ***VARCHAR****(50)* ***–*** *изменится длину строки в колонке на 50*

ALTER TABLE *EQUIPMENT* ***ALTER COLUMN*** *имя_колонки* ***DROP DEFAULT*** *– удаляет установленное для колонки текущее значение по умолчанию;*

ALTER COLUMN *имя_колонки* **DROP IDENTITY** *– удаляет для указанной колонки установленный для нее атрибут автоинкрементного значения.*

Удаление и модификация таблиц.

Операции:

Добавление ограничений целостности:

ADD [CONSTRAINT *имя_ограничения] ограничение*

Ограничение задается точно так же, как и табличное ограничение в предложении CREATE TABLE

Пример1:

```
ALTER TABLE EQUIPMENT
   ADD CONSTRAINT DeptQuip
      FOREIGN KEY(Equip_Owner)
      REFERENCES DEPT
      ON DELETE SET NULL
```

Удаление и модификация таблиц.

Пример2:

```
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT Revenue
    CHECK(Salary + Comm > 30000)
```

Удаление ограничений целостности

DROP *тип_ограничения* *имя_ограничения*

тип_ограничения: FOREIGN KEY, UNIQUE, CHECK, CONSTRAINT

Пример:

```
ALTER TABLE EMPLOYEE
  DROP CONSTRAINT Revenue
```

Предложения языка манипулирования данными

Предложение INSERT

INSERT INTO *имя_таблицы* [(*имя_колонки*, …)] **VALUES** (*значение*, …), (…), …

имя_таблицы –указывает существующую таблицу базы данных, в которую вставляются новые значения (строки).
(*имя_колонки*, …) – указывает имена колонок, для которых в данном предложении представляются значения.

Свойства:

Одно и то же имя колонки не должно указываться дважды;

Если список имен колонок опущен, порядок колонок определяется в соответствии с CREATE TABLE;

Для неуказанных колонок определяется атрибут NULL или DEFAULT;

Предложения языка манипулирования данными

Предложение INSERT

Свойства:

Количество значений, указанных в каждой строке, должно соответствовать явному или неявному списку имен колонок;

Тип вставляемых значений должен соответствовать типу соответствующих им колонок

Строки, добавляемые в таблицу, получены в результате запроса:
INSERT INTO *имя_таблицы* [(*имя_колонки*, …)] *запрос*

Количество колонок в результате выполнения запроса должно совпадать с количеством колонок, указанным (явно или неявно) в списке.

Предложения языка манипулирования данными

Предложение INSERT

Пример1:

DeptNo	DeptName	MgrNo	AdmrDept
CHAR(3) NOT NULL	VARCHAR(29) NOT NULL	CHAR(6)	CHAR(3) NOT NULL
Номер отдела	Название отдела	Номер сотрудника-руководителя	Номер отдела, которому подчиняется данный отдел

1) **INSERT INTO** DEPARTMENT
   VALUES (’E31’, ’ARCHITECTURE’, ’00390’, ’E01’);
2) **INSERT INTO** DEPARTMENT **(**DeptNo, DeptName, AdmrDept**)**
   VALUES (’E31’, ’ARCHITECTURE’, ’E01’**);**
3) **INSERT INTO** DEPARTMENT
   VALUES (’E31’, ’ARCHITECTURE’, NULL, ’E01’**)**

Предложения языка манипулирования данными

Предложение INSERT

Пример1:

DeptNo	DeptName	MgrNo	AdmrDept
CHAR(3) NOT NULL	VARCHAR(29) NOT NULL	CHAR(6)	CHAR(3) NOT NULL
Номер отдела	Название отдела	Номер сотрудника-руководителя	Номер отдела, которому подчиняется данный отдел

4) INSERT INTO DEPARTMENT (DeptNo, DeptName, AdmrDept**)**
VALUES (’B11’, ’PURCHASING’, ’B01’**),**
(’E41’, ’DATABASE ADMINISTRATION’, ’E01’**)**

Предложения языка манипулирования данными

Предложение INSERT

Пример2:

EmpNo	ProjNo	ActNo	EmpTime	EmStDate	EmEnDate
CHAR(6) NOT NULL	CHAR(6) NOT NULL	SMALLINT NOT NULL	DEC(5,2)	DATE	DATE
Номер сотрудника	Номер проекта	Номер задания	Доля времени, затрачиваемая сотрудником на выполнение задания	Дата начала работы над заданием	Дата завершения задания

CREATE TABLE MA_EMP_ACT (
EmpNo **CHAR(6) NOT NULL,**
ProjNo **CHAR(6) NOT NULL,**
ActNo **SMALLINT NOT NULL,**
EmpTime **DEC**(5,2),
EmStDate **DATE,**
EmEnDate **DATE**)

INSERT INTO MA_EMP_ACT
SELECT * FROM EMP_ACT
WHERE ProjNo LIKE ’MA%’

Предложения языка манипулирования данными

Предложение DELETE

DELETE FROM *имя_таблицы* [**WHERE** *условие_поиска*]

имя_таблицы – должно указывать существующую таблицу базы данных.
условие_поиска – определяет условие отбора удаляемых строк

Свойства:

*В **условии поиска** в конструкции WHERE могут быть использованы только колонки таблицы, указанной в предложении DELETE;*
***Условие поиска** применяется к каждой строке таблицы, и удаляются те строки, для которых результатом условия поиска является значение true;*
При удалении строк из родительской таблицы проверяются ограничения ссылочной целостности;
Если при выполнении DELETE, возникла ошибка, никакие изменения в базе данных не происходят

Предложения языка манипулирования данными

Предложение DELETE

Примеры

Пример 1. Удалить из таблицы DEPARTMENT отдел с номером (DeptNo) ‘D11’

```
DELETE FROM DEPARTMENT
WHERE DeptNo = ’D11’
```

Пример 2. Удалить из таблицы DEPARTMENT все строки.

```
DELETE FROM DEPARTMENT
```

Пример 3. Удалить из таблицы EMPLOYEE сотрудников, выполняющих операции SALESREP или FIELDREP, кто не выполнял продажи в 1995 г.

```
DELETE FROM EMPLOYEE
WHERE Job IN (’SALESREP’,’FIELDREP’)
 AND
 LastName NOT IN (SELECT Sales_Person
                  FROM SALES
                  WHERE YEAR(Sales_Date)=1995)
```

Предложения языка манипулирования данными

Предложение UPDATE

UPDATE *имя_таблицы* **SET** *имя_колонки* = *значение*, … [**WHERE** *условие_поиска*]

имя_таблицы – должно указывать имя существующей таблицы базы данных

Конструкция *SET* определяет, какие колонки таблицы и каким образом изменяют свои значения.

имя_колонки – указывает колонку, значение которой должно быть модифицировано.

значение – указывает новое значение колонки.

условие_поиска – условие, определяющее, какие строки должны быть модифицированы

Предложения языка манипулирования данными

Предложение UPDATE

Свойства:

*В **условии поиска** в конструкции WHERE могут быть использованы только колонки таблицы, указанной в предложении UPDATE;*
***Условие поиска** применяется к каждой строке таблицы, и модифицируются те строки, для которых результатом условия поиска является значение true;*
*Все ограничения проверяются, и если хотя бы одно из них будет нарушено, соответствующая таблица изменена не будет, а предложение **UPDATE** завершится с ошибкой.*

Предложения языка манипулирования данными

Предложение UPDATE

Примеры

Пример 1.

UPDATE EMPLOYEE
SET JOB = ’LABORER’
WHERE EMPNO = ’000290’

Пример 2.

UPDATE PROJECT
SET PRSTAFF = PRSTAFF + 1.5
WHERE DEPTNO = ’D21’

Пример 3. Для всех сотрудников, кроме менеджера отдела (WORKDEPT) ‘E21’, были сделаны временные переназначения. Установить в таблице EMPLOYEE должность (JOB) соответствующих сотрудников в NULL и их оплату (SALARY, BONUS, COMM) в 0.

UPDATE EMPLOYEE
SET JOB=**NULL**, SALARY=0, BONUS=0, COMM=0
WHERE WORKDEPT = ’E21’ **AND** JOB <> ’MANAGER’