Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Алтайский государственный технический университет им. И.И. Ползунова»

Методические указания к выполнению лабораторного практикума по дисциплине “Базы данных”

Для студентов направления 09.03.04 “Программная инженерия”

Барнаул 2020

Разработчики:
доцент Ананьев П.И.
к.э.н., доцент Кайгородова М.А.

Методические указания разработаны на основании Федерального государственного образовательного стандарта высшего образования - бакалавриат по направлению подготовки 09.03.04 Программная инженерия.

Методические указания рекомендованы к печати методической комиссией кафедры ПМ АлтГТУ

Оглавление

1 Цель и задачи лабораторного практикума

Лабораторная работа является одной из форм подготовки бакалавров. Лабораторный практикум – это самостоятельная работа студентов, подготовительная ступень к написанию курсовой и дипломных работ; он способствует приобретению практического опыта и соответствующих компетенций, необходимых программисту.

Цель лабораторного практикума по дисциплине «Базы данных» — закрепление теоретических знаний и приобретение практических навыков по проектированию баз данных и их реализации.

Для достижения указанной цели студенту необходимо решить следующие **задачи**:

- описать предметную область;
- используя полученные теоретические знания по данной дисциплине, разработать концептуальную модель базы данных (БД);
- построить реляционную модель БД;
- на основании концептуальной модели с использованием CASE-средств создать логическую модель, разработать физическую модель для СУБД PostgreSQL
- сгенерировать БД на сервере;
- разработать серверную часть приложения для автоматизации предметной области.

2 Организация выполнения лабораторного практикума

Преподаватель назначает студенту номер задания из вариантов, предложенных в после теоретического материала первой лабораторной работы. Выполнение каждой лабораторной работы состоит из следующих этапов:

1. Выполнение задания.
2. Создание отчета.
3. Проверка задания преподавателем.
4. Защита работы.

3 Содержание лабораторного практикума

3.1 Лабораторная работа №1

Тема: Описание предметной области.

Задание: Выполнить описание предметной области в соответствии с выданным заданием. Составить список запросов (минимум 6), на которые должна будет отвечать разработанная в дальнейшем база данных.

Теоретический материал

Работа аналитика начинается с понимания перспектив той системы управления, в которой работает его клиент. Проблемы автоматизации возникают в первую очередь тогда, когда в существующей системе имеются проблемы или неиспользованные возможности. Они должны быть отмечены в курсовой работе.

Программист-аналитик использует в своей работе знания основ работы предприятия, а также специальные знания по соответствующей отрасли производства. Его основным инструментом является способность задавать компетентные вопросы и умение находить на них ответы, в том числе, в литературных источниках. В ходе исследования выясняются потребности в информации и возможные пути решения проблем. Целью описания предметной области является нахождение возможных путей решения поставленной проблемы. Этап начинается с анализа информационных потребностей и имеющихся для их удовлетворения ресурсов, затем выясняются возможные подходы к решению поставленных задач и, наконец, выбирается приемлемый для дальнейшего проектирования вариант.

Для лучшего понимания рассматриваемой системы желательно составить подробное описание того как она функционирует. В этом описании должна содержаться полная информация о процессах, которые протекают в организации, с указанием людей, в них участвующих. Также в описании должны быть отмечены документы, которые фигурируют в рассматриваемых процессах. Особенно важно разделить сотрудников организации по должностям. Это помогает при построении программы определить функционал, который будет доступен тому или иному сотруднику. Кроме этого в описании должны быть сформулированы вопросы, на которые будет отвечать разработанная база данных.

Ниже приведен пример описания предметной области «Зоопарк».

Служащих зоопарка можно подразделить на несколько категорий: ветеринары, уборщики, дрессировщики, строители-ремонтники, работники администрации. Каждая из перечисленных категорий работников имеет уникальные атрибуты-характеристики, определяемые профессиональной направленностью. За каждым животным ухаживает определенный круг служащих, причем только ветеринарам, уборщикам и дрессировщикам разрешен доступ в клетки к животным.

В зоопарке обитают животные различных климатических зон, поэтому часть животных на зиму необходимо переводить в отапливаемые помещения. Животных можно подразделить на хищников и травоядных. При расселении животных по клеткам необходимо учитывать не только потребности данного вида, но и их совместимость с животными в соседних клетках (нельзя рядом селить, например, волков и их добычу - различных копытных).

Для кормления животных необходимы различные типы кормов: растительный, живой, мясо и различные комбикорма. Растительный корм это фрукты и овощи, зерно и сено. Живой корм - мыши, птицы, корм для рыб. Для каждого вида животных рассчитывается свой рацион, который в свою очередь варьируется в зависимости от возраста, физического состояния животного и сезона. Таким образом у каждого животного в зоопарке имеется меню на каждый день, в котором указывается количество и время кормлений в день, количество и вид пищи (обезьянам необходимы фрукты и овощи, мелким хищникам - хорькам, ласкам, совам, некоторым кошачьим, змеям - надо давать мышей). У зоопарка имеются поставщики кормов для животных. Каждый поставщик специализируется на каких-

то конкретных видах кормов. Часть кормов зоопарк может производить сам: запасать сено, разводить мышей и т.д.

Ветеринары должны проводить медосмотры, следить за весом, ростом, развитием животного, ставить своевременно прививки и заносить все эти данные в карточку, которая заводится на каждую особь при ее появлении в зоопарке. Больным животным назначается лечение и при необходимости их можно изолировать в стационаре.

При определенных условиях (наличие пары особей, подходящих по возрасту, физическому состоянию) можно ожидать появления потомства. Потомство от данной пары животных при достижении ими положенного возраста можно либо оставить в зоопарке, создав для них подходящие условия содержания, либо обменяться с другими зоопарками или просто раздать в другие зоопарки - по решению администрации.

Из вышеизложенного описания следуют примеры запросов к базам данных информационной системы «Зоопарк»:

1. Получить список и общее число служащих зоопарка, либо служащих данной категории полностью, по продолжительности работы в зоопарке, по половому признаку, возрасту, размеру заработной платы.
2. Получить перечень и общее число служащих зоопарка, ответственных за указанный вид животных либо за конкретную особь за все время пребывания животного в зоопарке, за указанный период времени.
3. Получить список и общее число служащих зоопарка, имеющих доступ к указанному виду животных либо к конкретной особи.
4. Получить перечень и общее число всех животных в зоопарке либо животных указанного вида, живших в указанной клетке все время пребывания в зоопарке, по половому признаку, возрасту, весу, росту.
5. Получить перечень и общее число нуждающихся в теплом помещении на зиму, полностью животных только указанного вида или указанного возраста.
6. Получить перечень и общее число животных, которым поставлена указанная прививка, либо переболевших некоторой болезнью, по длительности пребывания в зоопарке, половому признаку, возрасту, признаку наличия и количеству потомства.
7. Получить перечень всех животных, совместимых с указанным видом, либо только тех животных, которых необходимо переселить, или тех, которые нуждаются в теплом помещении.
8. Получить перечень и общее число поставщиков кормов полностью, либо поставляющих только определенный корм, поставлявших в указанный период, по количеству поставляемого корма, цене, датам поставок.
9. Получить перечень и объем кормов, производимых зоопарком полностью, либо только тех кормов, в поставках которых зоопарк не нуждается (обеспечивает себя сам).
10. Получить перечень и общее число животных полностью, либо указанного вида, которым необходим определенный тип кормов, в указанном сезоне, возрасте или круглый год.
11. Получить полную информацию (рост, вес, прививки, болезни, дата поступления в зоопарк или дата рождения, возраст, количество потомства) о всех животных, или о животных только данного вида, о конкретном животном, об особи, живущей в указанной клетке.

Варианты заданий:

1. Модель для университета. Сколько преподавателей работает на математическом факультете? Их фамилии? Какие предметы они преподают?

2. Модель для университета. Какие студенты специализируются в истории? В английском?

3. Модель для университета. Кто из преподавателей читает социологические курсы? Какие курсы они читают? Каким группам студентов?

4. Модель для университета. Сколько студентов, чьей специальностью является немецкий язык, официально зарегистрированы на усиленной программе? Кто является преподавателем каждого из них?

5. Модели для торговой фирмы. Какие товары имеют продажную цену более 200 рублей? Какие из них имеют закупочную цену менее 150 рублей? Какие товары произведены на Москве? Кто их изготовители?

6. Модели для торговой фирмы. Кто из продавцов продал товары ценой более 200 рублей? Даты этих продаж? Какова базовая зарплата этих продавцов?

7. Модели для банка. Какой процент обладателей текущих счетов банка составляют его служащие?

8. Модели для банка. Сколько кассиров имеют в банке сберегательные счета? Сколько менеджеров? Сколько кассиров не имеют таких счетов?

9. Модели для банка. Кто из менеджеров, имеющих в банке сберегательные счета, руководит служащими, имеющими в банке сберегательные счета?

10. Выведите концептуальную модель данных из следующего отчета

Консультационная служба **Отчет о специализации консультантов**				
Фамилия	**№ Страховки**	**Дата приема**	**Код специальности**	**Описание специальности**
Иванов	**539-88-4242**	**22/11/2000**	**A** **B** **O**	**Обучение пользователей** **Ввод данных** **Преобразование файлов**
Петров	**560-43-1111**	**8/11/1999**	**C** **D** **F**	**Программирование** **Преобразование файлов** **Системное проектирование**
Сидоров	**524-33-8119**	**7/3/1990**	**A** **C** **D** **F**	**Обучение пользователей** **Программирование** **Системный анализ** **Системное**

11. Сколько студентов занимается по программе Физика 201? Какие предметы изучает Иванов? Сколько раз Петров изучал Бухгалтерский Учет 201, когда и кто был его преподавателем и какие оценки он получил?

12. База данных должна давать ответы на вопросы по истории Европы. Создайте отдельную модель данных для указанной задачи.
Сколько королей Пруссии носили имя Фредерик? В какие годы они жили и в какие — правили? Управляли ли они на протяжении своей жизни какими-либо еще странами? Управлялись ли в XVII веке какие-либо европейские страны женщинами? Если да, то какие?
13. База данных должна давать ответы на вопросы по истории Европы.
Правил ли дед Марии-Антуанетты какой-либо страной? Какой и когда? Кто была ее мать? Были ли случаи, когда правители двух разных стран женились между собой? Сколько детей Генриха VIII стали королями Англии? Кто были их матери?

14. Какие станции транслируют программы «Бэтмэн»? Повторяла ли компания Brick Wall в этом году какие-либо серии Косби-шоу за 1988 год? Показывали ли они пятую серию? Когда и какая станция его транслировала?

15. Репортажи о скольких бейсбольных матчах Brick Wall показала за последний год? Когда они транслировали встречи между командами «Dodgers» и «Mets»? Матчи какой команды показывались больше всего? Как насчет футбольных матчей? Баскетбольных? Теннисных? Турниров по гольфу? Других видов спорта? Был ли показан хоть один теннисный матч с участием Штеффи Граф? Когда и какой станцией?

16. Какие коммерческие объявления Brick Wall показала более трех раз в течение одного часа на одной станции? Когда это было? В течение какого часа, какого числа и на какой станции? Какую плату Brick Wall назначила за трансляцию каждого из этих коммерческих сообщений?

17. Создайте модель базы данных, отвечающей на юридические вопросы. В каких делах высказывались мнения по Разделу 411.3c федерального кодекса? В каких судах? Были ли они отвергнуты? Какой раздел федерального кодекса был затронут в процессе Блэка против Вильямса?

18. Создайте модель базы данных, отвечающей на юридические вопросы. Какие юридические фирмы представляли General Continental в судах в течение последних десяти лет? Какие дела разбирались; какая сторона выиграла; каков был размер вознаграждения? Какие фирмы представляли противную сторону? Какие еще крупные компании представляли эти юридические фирмы в процессах в то же самое время?

19. Авиакомпания хочет получать ответы на подобные вопросы о своих самолетах: «Сколько посадочных мест в Боинге 727? Сколько у него двигателей? Какой средний возраст Боингов 727 нашего авиапарка? Кто главный механик, ответственный за обслуживание самолета номер 1388? Какая компания создала этот самолет?»

20. Администрация в большом городе должна отслеживать имеющееся у нее компьютерное оборудование. Она также хочет получать ответы на вопросы о моделях компьютеров. Создайте модель данных, отвечающую на следующие вопросы:
Какой максимальный объем памяти возможен у IBM PC? У PC-XT и PC-AT? Каков максимальный объем памяти у Macintosh II? У кого из наших служащих в кабинете есть IBM PC? У кого стоит компьютер с серийным номером 4538842? Какова его оперативная память?

21. В процессе работы над проектом для страховой компании один из аналитиков консультационной фирмы создал отчет, оценивающий производительность труда

операторов, вводящих данные. Этот отчет выдает число транзакций каждого типа, введенных каждым клерком в каждый день месяца. Выведите концептуальную модель данных, которая могла бы быть использована в качестве основы для указанного отчета.

СТРАХОВАЯ КОМПАНИЯ

МЕСЯЧНЫЙ ОТЧЕТ О ПРОИЗВОДИТЕЛЬНОСТИ ТРУДА СЛУЖАЩИХ

За месяц по 31 марта

№служащего	Фамилия	Дата	Тип транзакции	Количество
3855	Иванов	3/1	Новый полис	15
			Взнос	75
			Требование	22
		3/2	Новый полис	18
			Взнос	53
			Требование	25
3921	Петров	3/1	Взнос	45
			Изменение	83
			Требование	10
		3/2	Новый полис	8
			Взнос	63
			Изменение	35

22. Построить модель для ответов на вопросы по теме: “Издание учебной литературы для ВУЗов”, отвечающую на следующие вопросы:
Определить ВУЗы, в которых есть заданная специальность и определить план приема на указанную специальность по каждому ВУЗу. Какие учебники готовятся для студентов указанной специальности? Для каких специальностей может быть использовано в учебном процессе указанное издание? Определить размер тиража, если задается количество экземпляров издания на одного студента, когда оно включено в список основной и дополнительной литературы. Какое количество учебной литературы должно быть отправлено в указанный ВУЗ.

23. Построить модель для отображения результатов самостоятельной работы студента в течение семестра с целью организации учебного процесса, его совершенствования и выдачи справочной информации студенту и в управляющие инстанции, отвечающую на следующие вопросы:
Какой график выполнения самостоятельных работ по указанной дисциплине? Какие задания на самостоятельную работу у указанного студента? Как выполняет самостоятельные работы указанный студент? Какие студенты имеют результаты по указанной контрольной торчке?

24. Создать модель для помощи разработчику СУБД или лицу, которое использует СУБД, отвечающую на следующие вопросы:
Какие характеристики есть у указанной СУБД (вид используемой модели данных, вид техники, временные характеристики, обеспечение секретности)? Какую литературу можно использовать при изучении указанной СУБД? В каких организациях внедрена и работает данная СУБД? Какие организации занимаются разработкой ПО с использованием указанной СУБД?

25. Сбербанк. Сведения о вкладчиках филиала Сбербанка: номер Лицевого счета, паспортные данные, сумма вклада, категория вклада, дата последней операции.

26. Сессия. Сводная ведомость группы по итогам сессии: фамилия с инициалами, номер зачетки, совокупность оценок по зачетом и экзаменам.

27. Склад. Инвентарная ведомость наличия товаров на складе: наименование товара, единица измерения, цена одной единицы, количество.

28. Магазин. Учетная ведомость наличия товаров в коммерческом магазине: наименование товара, количество (штук) - сдано и осталось, цена одной штуки.

29. Ломбард. Наименование каждого хранимого товара, оценочная стоимость; сумма, выданная сдатчику; дата сдачи, оговоренный срок хранения.

30. Коммерческий вестник. Наименование товара или услуги, единица измерения, количество, цена, наименование продавца, условия оплаты, условия поставки-отгрузки.

31. Автосалон. Марка предлагаемого автомобиля, цвет, технические характеристики, фирма-изготовитель, дата выпуска, пробег, цена.

32. Справочник аудиторий. Номер аудитории, корпус, вместимость, особенности (например, нет доски), спецоборудование (например, лингафонный кабинет).

33. Кадры сотрудников. Паспортные данные сотрудника, образование, степень, звание, специальность, подразделение, должность, оклад.

34. Кадры студентов. Паспортные данные, группа, адрес родителей, изучаемый язык, размер стипендии, наличие места в общежитии (иди потребность в нем).

35. Расписание занятий. Номер группы, номер недели, день недели, номер пары и все данные о занятии на этой паре.

36. Каталог библиотеки. Инвентарный номер, выходные данные книги, цена, факт наличия или кому выдана.

37. Расписание автобусов. Номер маршрута, направление, время отправления, расстояние, марка автобуса, цена билета, количество проданных билетов.

38. Расписание самолетов. Номер рейса, время вылета и прибытия на конечный пункт, тип самолета, периодичность полетов, цена билета, пункты промежуточной посадки.

39. Домоуправление. Адрес, тип квартиры, площадь, степень благоустройства, форма собственности, отношение к капремонту и сносу, паспортные данные проживающих.

40. Горжилуправление. Адрес дома, количество квартир, общая и жилая площадь, год постройки, состояние, год последнего ремонта.

41. Справочник селекционера. Наименование сорта какой-либо культуры, автор, родительские сорта, урожайность, характеристики плодов, морозоустойчивость, устойчивость к вредителям и болезням, наличие в селекционном фонде.

42. Справочник ГАИ. Марка, цвет, заводской и бортовой номера, дата выпуска, дата последнего техосмотра транспортного средства, паспортные данные владельца.

43. Справочник автолюбителя. Марка, фирма, год начала серийного выпуска и технические характеристики различных автомобилей.

44. Справочник покупателя. Наименование и номер магазина, адрес, телефоны, характер специализации, форма собственности, средний объем товарооборота.

45. Справочник предприятий. Наименование, адрес, руководитель предприятия, выпускаемая продукция, потребляемое сырье, статус (форма собственности), численность работающих.

46. Справочник абитуриента. Наименование вуза, адрес, список факультетов, конкурс прошлого года по каждому факультету.

47. Справочник службы быта. Наименование фирмы, адрес, профиль (вид оказываемых услуг), статус (форма собственности), разряд (категория цен).

48. Справочник кутилы (рестораны и кафе). Название, адрес, наценочная категория, особенности кухни, часы работы, вид музыкально-эстрадной обслуживания.

49. Биржа труда-1. Справочник безработных: паспортные данные, профессия, образование, место и должность последней работы, причина увольнения, семейное положение, жилищные условия, адрес.

50. Биржа труда-2. Справочник вакансий: предприятие, должность, зарплата, жилищные условия, требования к специалисту.

51. Справочник знахаря. Наименование болезни, симптомы, возможные последствия, рекомендуемые лекарства, снадобья и процедуры.

52. Справочник видеомана. База видеофильмов: название, студия, жанр, год выпуска, режиссер, исполнители главных ролей, краткое содержание, субъективная оценка фильма.

53. Бюро знакомств. База клиентов: пол, возраст, рост, вес, знак Зодиака, материально-жилищное положение, профессия, хобби, требования к будущему партнеру.

54. Домашняя библиотека. Автор, название книги, год и место издания, раздел библиотеки (специальная литература, домашнее хозяйство, хобби, беллетристика и так далее), год и место приобретения.

55. Справочник фаната. База спортсменов: паспортные и антропологические данные, гражданство, происхождение, вид спорта, клуб или команда, данные о последнем рекорде или победах и так далее.

56. Справочник радиолюбителя. База паспортных данных транзисторов дли других деталей: марка, характеристики, предельно допустимые условия эксплуатации, цена и так далее.

57. Записная книжка. Фамилии, адреса, телефоны знакомых и родственников, характер знакомства, дата рождения и так далее.
58. Справочник коммерческих банков. Наименование, адрес, статус (форма собственности), условия хранения средств на лицевом счете (годовые проценты на разных типах вклада).

59. Справочник начальника тюрьмы. Паспортные данные заключенных, статья, срок, дата заключения под стражу, место в тюремной иерархии, сведения о родственниках, особенности характера.

60. Справочник командира. Список подчиненных военнослужащих: паспортные данные, адрес родителей. гражданская профессия, звание и дата его получения, должность, подразделение, форма службы (срочная, кадровая, контрактная и так далее), период службы (для срочнослужащих), особенности характера и отношение к службе.

61. Картотека Интерпола. Данные по каждому зарегистрированному преступнику: фамилия, имя, кличка, рост, цвет волос и глаз, особые приметы, гражданство, место и дата рождения, последнее место жительства, знание языков, преступная профессия, последнее дело и так далее. Преступные и мафиозные группировки (данные о подельниках). Выборка по любому подмножеству признаков. Перенос "завязавших" в архив; удаление - только после смерти.

62. Справочник покупателя. База торговых точек города: название, адрес и телефоны, специализация, форма собственности, время работы. Выбор магазинов произвольному шаблону.

63. Генеалогическое дерево. Паспортные данных членов некоторого родового клана; ссылки на детей (или на родителей). Поиск всех потомков или всех предков для указанного лица.

64. Администратор гостиницы. База номеров: класс, число мест. База гостей: паспортные, данные, даты приезда и отъезда, номер. Поселение гостей: выбор подходящего номера (при наличии свободных мест), регистрация, оформление квитанции. Отъезд: выбор всех постояльцев, отъезжающих сегодня, освобождение места или оформление задержки с выпиской дополнительной квитанции.

65. Справочник меломана. База групп и исполнителей; база песен; база дисков с перечнем песен (в виде ссылок). Выбор всех песен заданной группы; всех дисков, где встречается заданная песня.

66. Ежедневник. База намечаемых мероприятий - дата, время, место проведения. Автоматическое напоминание ближайшего дела: по текущей дате и времени; удаление вчерашних дел либо перенос на будущее. Просмотр дел на завтра, послезавтра и так далее.

67. Терминология. База определений какой-либо науки: вводимый термин, его толкование (определение), ссылки на используемые термины. Возможность просмотра всей цепочки от заданного термина до первичных понятий.

68. Шеф-повар. База рецептур блюд: раскладка, рецепт приготовления. База продуктов на складе: наименование, цена, количество. Формирование меню на день (на заданное число персон). Проверка достаточности запасов; формирование расходной накладной на склад, корректировка запасов.

69. Справочник лекаря. База болезней: название, симптомы, процедуры, перечень рекомендуемых лекарств с указанием требуемого количества. База медикаментов на складе: название, количество, взаимозаменяемость. Формирование рецепта после осмотра больного, проверка наличия лекарств, корректировка запасов.

70. Справочник фирм. Название, адрес и телефоны, первое лицо, статус (форма собственности), сырье, продукция. Выбор по произвольному шаблону.

71. Обмен жилья. База предложений по обмену: район, площадь, планировка и так далее; требования к вариантам обмена. Регистрация клиентов, выбор подходящих вариантов, удаление при состоявшемся обмене или отказе.
Развитие задачи - возможность съезда или разъезда, в том числе "несколько на несколько"; "возможны варианты".

72. Справочник почтовой индексации. Республика, область (край), район, населенный пункт, почтовый индекс. Поиск по любой совокупности полей (кроме последнего); иерархическая связь между полями (обратите внимание, что Новомосковск есть и в Тульской, и в Днепропетровской областях).

73. Купи-продай. База продавцов: наименование товара, объем партии при оптовой продаже, цена, условия продажи-отгрузки, форма оплаты, контактный адрес или телефон, примечание (например, "посредников прошу не беспокоиться"). База покупателей: наименование товара, объем покупки, приемлемая цена и форма оплаты, контактный адрес или телефон, примечание. Поиск и регистрация вариантов с той и другой стороны; формирование объявлений для печати, удаление в архив после купли-продажи (возможно, один из клиентов остается. неудовлетворенным), полное удаление при отказе от услуг.

74. Успеваемость. База студентов: фамилия, имя, отчество, группа. База предметов: название, форма контроля (совокупность зачетов и экзаменов по семестрам). Ввод результатов очередной сессии в сводную ведомость, пополнение/исправление после пересдач. Формирование списка задолжников.

75. Классификация до Дарвину- База растений и/или животных с указанием царства, класса, типа, семейства, рода, вида. Иерархическая организация классификации (например, просмотр только тех семейств, которые входят в данный тип). Отличительные признаки, по которым ведется классификация. Просмотр произвольной ветви дерева.

76. Рынок Компьютеров. База фирм-продавцов: название, адрес, телефоны. База компьютеров с их характеристиками; база комплектующих и расходных материалов. Регистрация наличия на фирмах разных моделей, расходных и комплектующих с указанием цены (в рублях или СКВ). Корректировка данных по рекламным объявлениям. Поиск подходящих вариантов.

3.2 Лабораторная работа №2

Тема: Построение концептуальной модели.

Задание: Построить концептуальную модель, описывающую предметную область из первой лабораторной работы. Для этого необходимо в предметной области выделить классы, определить их названия и атрибуты, связать классы между собой, определить мощности полученных связей.

Теоретический материал

Ниже приведены основные понятия, используемые при проектировании концептуальной модели.

Главными элементами концептуальной модели данных являются *объекты* и *отношения.* Объекты часто представляют в виде *существительных,* а отношения — в виде *глаголов.*

Объект можно неформально определить как осязаемую реальность, проявляющую четко выделяемое поведение. С точки зрения восприятия человеком объектом может быть:

- осязаемый и (или) видимый предмет;
- нечто, воспринимаемое мышлением;
- нечто, на что направлена мысль или действие.

Таким образом, мы расширили неформальное определение объекта новой идеей: **объект моделирует часть окружающей действительности и таким образом существует во времени и пространстве.**

Объектами реального мира не исчерпываются типы объектов, являющиеся существенными при проектировании информационного обеспечения систем. Другие важные типы объектов вводятся на этапе проектирования, и их взаимодействие друг с другом служит механизмом отображения поведения более высокого уровня. Из сказанного следует более четкое понятие: «Объект представляет собой конкретный опознаваемый предмет, единицу или сущность (реальную или абстрактную), имеющую четко определенное функциональное назначение в данной предметной области».

Существуют такие объекты, для которых определены явные концептуальные границы, но сами объекты представляют собой неосязаемые события или процессы. Таким образом, объект — это нечто, имеющее четко определенные границы. Но этого недостаточно, чтобы отделить один объект от другого или дать оценку качества абстракции. Поэтому объект можно определить следующим образом.

Объект обладает состоянием, поведением и идентичностью; структура и поведение схожих объектов определяет общий для них класс; термины "экземпляр класса" и "объект" взаимозаменяемы.

Состояние объекта характеризуется перечнем (обычно статическим) всех свойств данного объекта и текущими (обычно динамическими) значениями каждого из этих свойств.

К числу свойств объекта относятся присущие ему или приобретаемые им характеристики, черты, качества или способности, делающие данный объект самим собой. Все свойства имеют некоторые значения. Эти значения могут быть простыми количественными характеристиками, а могут ссылаться на другой объект.

Таким образом, можно установить различие между объектами и простыми величинами: простые количественные характеристики (например, число 3) являются «постоянными, неизменными и непреходящими», тогда как объекты существуют во времени, изменяются, имеют внутреннее состояние, преходящи и могут создаваться, уничтожаться и разделяться.

Тот факт, что всякий объект имеет состояние, означает, что всякий объект занимает определенное пространство (физически или в памяти компьютера).

Что такое поведение. Объекты не существуют изолированно, а подвергаются воздействию или сами воздействуют на другие объекты. ***Поведение - это то, как объект действует и реагирует; поведение выражается в терминах состояния объекта и передачи сообщений.***

Иными словами, поведение объекта - это его наблюдаемая и проверяемая извне деятельность. А операцией называется определенное воздействие одного объекта на другой с целью вызвать соответствующую реакцию.

Класс (объектное множество). Понятия класса и объекта настолько тесно связаны, что невозможно говорить об объекте безотносительно к его классу. Однако существует важное различие этих двух понятий. В то время как объект обозначает конкретную сущность, определенную во времени и в пространстве, класс определяет лишь абстракцию существенного в объекте.

В общепонятных терминах можно дать следующее определение класса: группа, множество или вид с общими свойствами или общим свойством, разновидностями, отличиями по качеству, возможностями или условиями.

Класс (объектное множество) — это некое множество объектов, имеющих общую структуру и общее поведение.

Любой конкретный объект является просто экземпляром класса.

Объекты представляют вещи, которые пользователи считают важными в моделируемой нами части реальности. Примерами объектов могут быть люди, автомобили, деревья, дома, молотки и книги. Это конкретные объекты. Концептуальными объектами будут компании, навыки, организации, проекты товаров, деловые операции, штатное расписание и др.

Объектные множества бывают **лексическими и абстрактными.**

Лексическое объектное множество. Объектное множество, состоящее из элементов, которые можно напечатать.

Абстрактное объектное множество. Объектное множество, состоящее из элементов, которые нельзя напечатать.

В компьютерной реализации концептуальной модели элементы лексических объектов будут представлены в виде строк символов. Элементы абстрактных объектов будут представлены внутренними номерами, не имеющими смысла вне системы. Внутренний номер иногда называют *«идентификатор объекта»* или **суррогатным ключом.**

Суррогатный ключ. *«Идентификатор»* абстрактного объекта-элемента в компьютерной системе; вне системы смысла не имеет.

Отношение. Связь между элементами двух объектных множеств.

Отношение связывает два объектных множества. Графически мы представляем отношение между двумя объектными множествами в виде соединяющего их отрезка.

Мощность - максимальное количество элементов одного объектного множества, связанных с одним элементом другого объектного множества.

Максимальная мощность является значительно более важным понятием, чем минимальная.

Максимальная мощность в одном из направлений, равная одному, соответствует математическому понятию функции, поэтому отношение, имеющее максимальную мощность в одном из направлений, равную одному, называется **функциональным** в этом направлении.

Если максимальная мощность отношения в обоих направлениях равна одному, то она называется отношением **один-к-одному.** Если максимальная мощность в одном направлении равна одному, а в другом — многим, то отношение называется отношением **один-ко-многим.** И, наконец, если максимальная мощность в обоих направлениях равна многим, то

отношение называется отношением **много-ко-многим.** В таблице 1 приведены характеристики трех основных мощностей отношений.

Атрибут. Функциональное отношение объектного множества с другим объектным множеством. На самом деле, атрибут объекта — просто функциональное отношение объектного множества этого объекта к другому объектному множеству. Удобно представлять атрибуты следующим образом (рис.1).

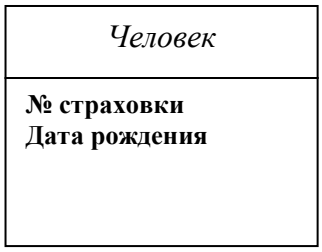


Рисунок 1. Представление объектных множеств с указанием атрибутов

Атрибуты подразделяются на *простые* и *составные, однозначные* и *многозначные,* а также *производные.*

Простой атрибут - атрибут, состоящий из одного компонента с независимым существованием.

Простые атрибуты не могут быть разделены на более мелкие компоненты. Примером простых атрибутов является атрибут Должность сущности Сотрудник. Простые атрибуты иногда называют *элементарными.*

Составной атрибут – атрибут, состоящий из нескольких компонентов, каждый из которых характеризуется независимым существованием.

Некоторые атрибуты могут быть разделены на более мелкие компоненты, которые характеризуются независимым существованием. Например, атрибут (Адрес) сущности Отделение компании, со значением ‘656011, г.Барнаул, пр.Ленина, 106' может быть разбит на отдельные атрибуты Улица, Номер дома, Город, Индекс.

Однозначный атрибут – атрибут, который содержит одно значение для каждого экземпляра сущности определенного типа.

Большинство атрибутов являются однозначными.

Многозначный атрибут – атрибут, который содержит несколько значений для каждого экземпляра сущности определенного типа.

Некоторые атрибуты могут иметь несколько значений для каждого экземпляра сущности. Например, сущность Отделение компании может иметь несколько значений для атрибута Номер телефона: '095-339-2178' и '095-339-4439'. Следовательно, атрибут Номер телефона в этом случае будет многозначным.

Производный атрибут – атрибут, который представляет значение, производное от значения связанного с ним атрибута или некоторого множества атрибутов, принадлежащих некоторому (не обязательно данному) типу сущности.

Некоторые атрибуты могут быть связаны с определенной сущностью. Например, значение атрибута Срок действия сущности Договор аренды вычисляется на основе атрибутов Начало срока аренды и Конец срока аренды, которые также относятся к типу сущности Договор аренды.

Ключ — это значение, которое однозначно определяет элемент объектного множества.

Пример концептуальной модели для предметной области «Зоопарк».

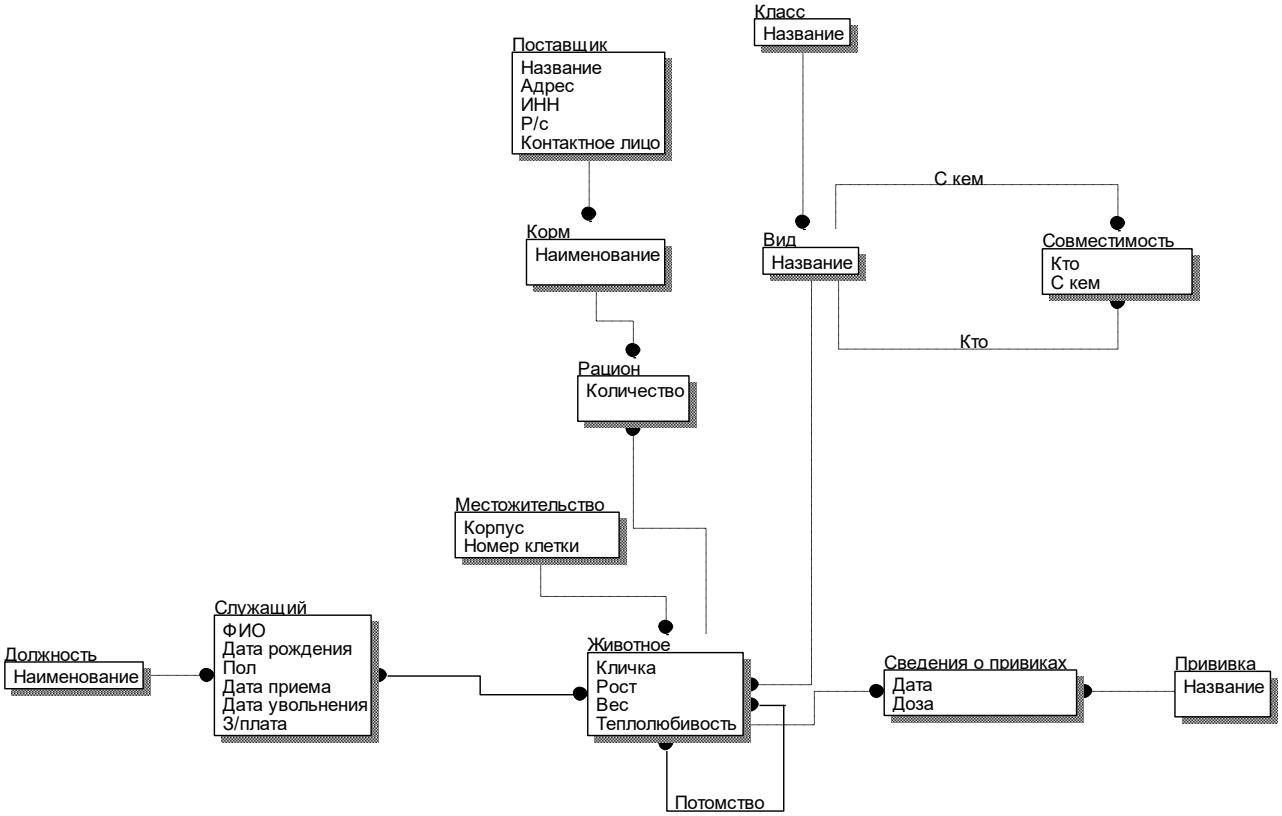


Рисунок 2. Концептуальная модель, описывающая зоопарк.

3.3 Лабораторная работа №3

Тема: Построение реляционной модели.

Задание: На основании концептуальной модели, разработанной во второй лабораторной работе построить реляционную модель.

Теоретический материал

Реляционная модель основана на математическом понятии *отношения*, физическим представлением которого является таблица. Реляционная модель предусматривает организацию данных исключительно в виде таблиц. Для потребителя, проектировщика БД, администратора БД и конечного пользователя данные представляются совершенно одинаково — в виде таблиц. Следовательно, таблица — базовый языковый объект реляционной модели.

Таблица представляет собой множество именованных *атрибутов,* или столбцов, и множество *записей (кортежей),* или строк. Очень часто столбец называется *полем* таблицы *(field).* Пересечение строки и столбца образуют *ячейку* таблицы (*cell*). Набор допустимых значений столбца — домен (domain) — характеризуется определенным типом данных, например *символьным* или *целым.* Строки же и представляют собой данные.

Реляционная модель предъявляет к таблице определенные требования.

- Данные в ячейках таблицы должны быть структурно неделимыми. Каждая ячейка может содержать *только одну* порцию данных. Это свойство часто определяется как *принцип информационной неделимости.* Недопустимо, чтобы в ячейке таблицы реляционной модели содержалось более одной порции данных, что иногда именуется *информационным кодированием (information coding).* Примером может служить идентификационный номер транспортного средства (Vehicle Identification Number — VIN). Если записать его в одну ячейку, то будет нарушен принцип неделимости информации, поскольку в ячейке окажутся такие *разделяемые* данные, как наименование производителя, модели, сведения о местонахождении предприятия-изготовителя и т.д. Хотя на практике решение о том, следовать ли жестко требованиям теории, почти всегда принимает проектировщик, нужно учитывать, что нарушение этих требований может затем отрицательно сказаться на обеспечении целостности данных.
- Данные в одном столбце должны быть одного типа.
- Каждый столбец должен быть уникальным (недопустимо дублирование столбцов).
- Столбцы размещаются в произвольном порядке.
- Строки (записи) размещаются в таблице также в произвольном порядке.
- Столбцы имеют уникальные наименования.

Помимо собственно таблиц и их свойств, реляционная модель имеет еще и собственные операции. Эти операции позволяют выполнять некоторые действия над подмножествами столбцов и/или строк, объединять (сливать) таблицы и выполнять другие операции над множествами.

Ключи

В таблице не должно быть повторяющихся строк. Поэтому необходимо иметь возможность уникальной идентификации каждой отдельной строки таблицы по значениям одного или нескольких столбцов (называемых реляционными ключами). Рассмотрим терминологию, используемую для обозначения реляционных ключей.

Суперключ – столбец или множество столбцов, которое единственным образом идентифицирует строку данной таблицы.

Суперключ однозначно обозначает каждую строку в таблице. Но суперключ может содержать дополнительные столбцы, которые необязательны для уникальной идентификации строки, поэтому рассмотрим суперключи, состоящие только из тех столбцов, которые действительно необходимы для уникальной идентификации строк.

Потенциальный ключ – суперключ, который не содержит подмножества, также

являющегося суперключем данного отношения.

Таблица может содержать несколько потенциальных ключей. Если ключ состоит из нескольких столбцов, то он называется составным ключом. Для идентификации потенциального ключа требуется знать смысл используемых столбцов в «реальном мире»; только это позволит обоснованно принять решение о возможности существования значений-дубликатов.

Первичный ключ – потенциальный ключ, который выбран для уникальной идентификации сток внутри таблицы.

Поскольку таблица не содержит строк-дубликатов, всегда можно уникальным образом идентифицировать каждую ее строку. Это значит, что таблица всегда имеет первичный ключ. В худшем случае все множество столбцов может использоваться как первичный ключ, но обычно, чтобы различить строки, достаточно использовать несколько меньшее подмножество столбцов.

Чаще всего в реальных системах в качестве первичного ключа используют суррогатный ключ. Это позволяет избежать проблем, связанных с изменением значения первичного ключа.

Пустое значение – указывает, что значение столбца в настоящий момент неизвестно или неприемлемо для этой строки.

Пустое значение (которое условно обозначается как NULL) следует рассматривать как логическую величину «неизвестно». NULL не следует понимать как нулевое численное значение или заполненную пробелами текстовую строку. Нули и пробелы представляют собой некоторые значения, тогда как ключевое слово NULL обозначает отсутствие какого-либо значения.

Внешний ключ — это столбец или подмножество столбцов одной таблицы, который может служить в качестве первичного ключа для другой таблицы. Говорят также, что внешний ключ одной таблицы является *ссылкой* на первичный ключ другой таблицы.

И последнее, на чем необходимо остановиться в этом кратком обзоре свойств реляционной модели, — два фундаментальных правила целостности: правило ***целостности объектов*** *(entity integrity rule)* и правило ***ссылочной целостности*** *(referential integrity rule).*

Правило целостности объектов утверждает, что первичный ключ не может быть полностью или частично пустым, т.е. иметь значение NULL.

Правило ссылочной целостности гласит, что внешний ключ может быть либо пустым (иметь значение NULL), либо соответствовать значению первичного ключа, на который он ссылается.

Нормализация полученной реляционной модели.

Нормализация представляет собой процесс дальнейшего совершенствования реляционной модели. Она выполняется после создания приближенной модели и предназначена для повышения уровня ее структурной организации. В основе нормализации лежит определенный математический аппарат, базирующийся на концепции *функциональной зависимости.*

Говорят, что один или множество столбцов функционально зависят от одного или множества столбцов ***X,*** если данное множество значений для ***X*** определяет единственное множество значений для ***Y***. Утверждение "***Y* функционально зависит от *X*"** равносильно утверждению "***X*** определяет ***Y***", которое записывается в форме $X \Rightarrow Y$.

Функциональная зависимость. Поле B таблицы функционально зависит от поля A той же таблицы в том и только в том случае, когда в любой заданный момент времени для каждого из различных значений поля A обязательно существует только одно из различных значений поля B. Отметим, что здесь допускается, что поля A и B могут быть составными.

Полная функциональная зависимость. Поле B находится в полной функциональной зависимости от составного поля A, если оно функционально зависит от A и не зависит функционально от любого подмножества поля A.

Наиболее очевидным примером может быть первичный ключ таблицы реляционной модели, который однозначно определяет строку этой таблицы. Однако могут существовать и другие зависимости, в которые не входят первичные ключи. Главная цель нормализации — избавить реляционную таблицу от зависимостей, не связанных с первичными ключами.

Рассмотрим нормальные формы.

Таблица представлена в первой нормальной форме (1НФ) тогда и только тогда, когда все ее столбцы содержат только неделимые (в том смысле, что их дальнейшее разложение невозможно) значения и в ней отсутствуют повторяющиеся группы (столбцов) в пределах одной строки.

Таблица представлена во второй нормальной форме (2НФ) тогда и только тогда, когда она представлена в 1НФ и каждый неключевой атрибут полностью определяется первичным ключом. Атрибут полностью определяется первичным ключом, если он находится в правой части выражения, описывающего функциональную зависимость, а левую часть этого выражения представляет первичный ключ или какое-либо выражение, которое может быть вычислено на его основе с использованием транзитивных свойств функциональной зависимости.

Таблица представлена в третьей нормальной форме (3НФ), если она удовлетворяет определению 2НФ и не одно из неключевых полей не зависит функционально от любого другого неключевого поля.

На практике, при создании логической модели данных, как правило, не проводят нормализацию. Опытные разработчики обычно сразу строят отношения в 3НФ. Кроме того, основным средством разработки логических моделей данных являются различные варианты ER-диаграмм. Особенность этих диаграмм в том, что они сразу позволяют создавать отношения в 3НФ.

Перевод концептуальной модели в реляционную

Переход от концептуальной к реляционной модели не делается прямо "в лоб". Класс может быть преобразован в группу таблиц, а группа классов — в одну таблицу, и т.д. Более точно было бы говорить о классах как о группах таблиц.

Каждая таблица должна иметь первичный ключ, гарантирующий, что каждая строка таблицы может быть однозначно определена. Первичный ключ может состоять из одного или более столбцов таблицы. База данных моделирует реальный мир, таблица является аналогом класса, а запись таблицы представляет отдельный объект класса. База данных должна иметь средства для идентификации объектов. Если она не позволяет отличить два разных объекта, значит, она неточно соответствует реальности. Аналогичным образом, каждый отдельный объект может быть представлен только одним способом. Если разрешить повторяющиеся записи, то модель будет представлять несколько копий одного и того же, что противоречит теории.

Реляционная модель в качестве ключевых значений позволяет использовать реальные атрибуты объектов, например, имя человека. На практике программисты часто испытывают недостаток таких признаков. Так, для таблицы людей нельзя использовать имя в качестве ключа, поскольку оно не является уникальным. Не подходит и номер паспорта: существует масса дубликатов и подделок, у детей паспортов нет, люди могут отказаться сообщать номера, и паспорта имеют смысл только внутри одной страны. Поэтому в качестве идентификатора человека обычно используют выработанное компьютером значение.

В требования к системе должны входить значения верхних пределов. Лучше всего выбирать такой размер полей, чтобы можно было записывать значения, от 10 до 100 раз превышающие используемые в настоящий момент — для возможности роста и развития.

После определения первичных ключей для каждой таблицы они могут быть использованы для связывания таблиц в базу данных.

В объектно-ориентированной модели отношения между классами выражаются через иерархии, контейнеры и ассоциации. В реляционной модели они выражаются только через совместно используемые ключи, поскольку это исключительно модель данных

Отношения много-ко-многим - это — классическая проблема для реляционного моделирования. Трудность заключается в том, что реляционная модель не имеет непосредственной поддержки отношений много-ко-многим.

Решение состоит в создании промежуточной таблицы. Первичный ключ этой таблицы должен состоять из двух внешних ключей — ключа таблицы 1 и ключа таблицы 2.

Пример такого преобразования показан на рисунке 3.

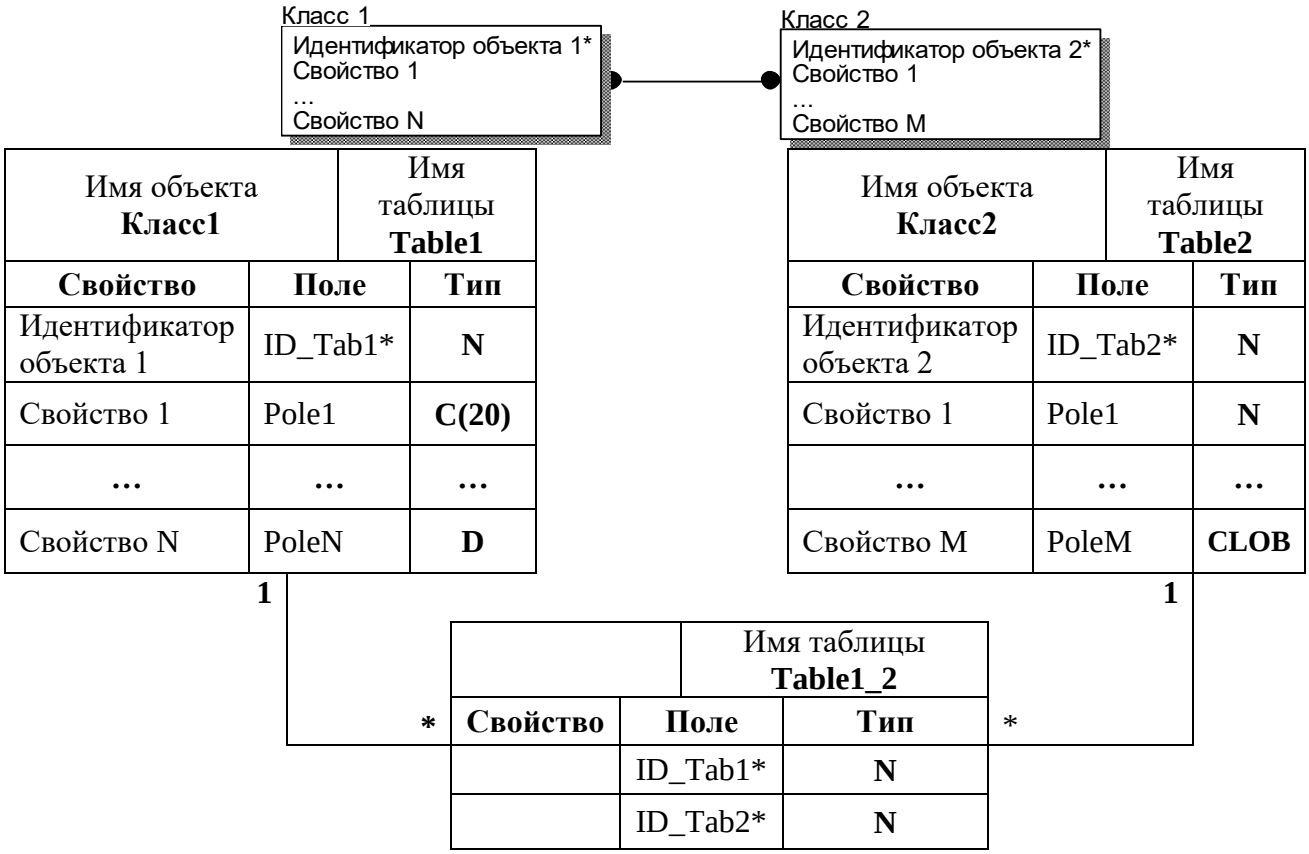


Рисунок 3. Преобразование связи много-ко-многим

Следующий этап работы над проектом предполагает переход от свойств классов к столбцам таблиц. При этом требуется определить типы и размеры всех столбцов, а также добавить специальные столбцы.

Таблица 1. Числовые типы PostgreSQL.

Имя	Размер	Описание	Диапазон
smallint	2 байта	целое в небольшом диапазоне	-32768 .. +32767
integer	4 байта	типичный выбор для целых чисел	-2147483648 .. +2147483647
bigint	8 байт	целое в большом диапазоне	-9223372036854775808 .. 9223372036854775807
decimal	переменный	вещественное число с указанной точностью	до 131072 цифр до десятичной точки и до 16383 — после
Numeric	переменный	вещественное число с указанной точностью	до 131072 цифр до десятичной точки и до 16383 — после
Real	4 байта	вещественное число с переменной точностью	точность в пределах 6 десятичных цифр
double precision	8 байт	вещественное число с переменной точностью	точность в пределах 15 десятичных цифр
smallserial	2 байта	небольшое целое с автоувеличением	1 .. 32767
Serial	4 байта	целое с автоувеличением	1 .. 2147483647
bigserial	8 байт	большое целое с автоувеличением	1 .. 9223372036854775807

Таблица 2. Денежные типы PostgreSQL.

Имя	Размер	Описание	Диапазон
money	8 байт	денежная сумма	-92233720368547758.08 .. +92233720368547758.07

Таблица 3. Символьные типы PostgreSQL.

Имя	Описание
character varying(*n*), varchar(*n*)	строка ограниченной переменной длины
character(*n*), char(*n*)	строка фиксированной длины, дополненная пробелами
text	строка неограниченной переменной длины

Таблица 4. Типы даты/времени PostgreSQL.

Имя	Размер	Описание	Наименьшее значение	Наибольшее значение	Точность
timestamp [(*p*)] [without time zone]	8 байт	дата и время (без часового пояса)	4713 до н. э.	294276 н. э.	1 микросекунда
timestamp [(*p*)] with time zone	8 байт	дата и время (с часовым поясом)	4713 до н. э.	294276 н. э.	1 микросекунда
date	4 байта	дата (без времени суток)	4713 до н. э.	5874897 н. э.	1 день
time [(*p*)] [	8 байт	время суток (без	00:00:00	24:00:00	1

Имя	Размер	Описание	Наименьшее значение	Наибольшее значение	Точность
`without time zone ]`		даты)			микросекунда
`time [ (`*p*`) ] with time zone`	12 байт	время дня (без даты), с часовым поясом	00:00:00+1559	24:00:00-1559	1 микросекунда
`interval [ `*поля*` ] [ (`*p*`) ]`	16 байт	временной интервал	-178000000 лет	178000000 лет	1 микросекунда

Пример. Перевод концептуальной модели, описывающей зоопарк, в реляционную.

Имя объекта: Поставщик	**Имя таблицы:Post**	
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Post	Numeric
Название	Naz	Varchar(25)
Адрес	Adres	Varchar(150)
ИНН	INN	Varchar(12)
Р/с	RS	Varchar(100)
Контактное лицо	FIO	Varchar(45)

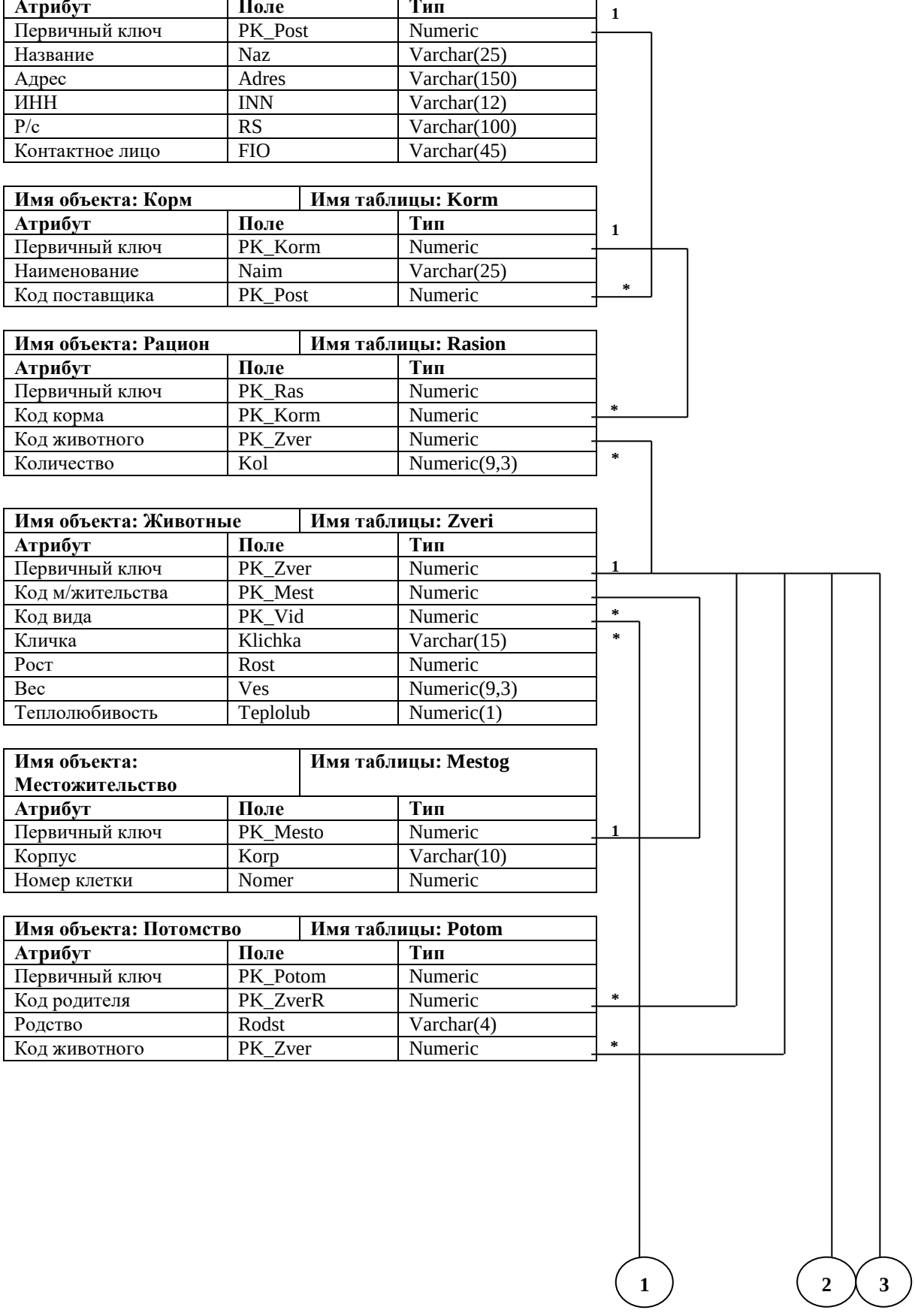
Имя объекта: Корм	**Имя таблицы: Korm**	
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Korm	Numeric
Наименование	Naim	Varchar(25)
Код поставщика	PK_Post	Numeric

Имя объекта: Рацион	**Имя таблицы: Rasion**	
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Ras	Numeric
Код корма	PK_Korm	Numeric
Код животного	PK_Zver	Numeric
Количество	Kol	Numeric(9,3)

Имя объекта: Животные	**Имя таблицы: Zveri**	
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Zver	Numeric
Код м/жительства	PK_Mest	Numeric
Код вида	PK_Vid	Numeric
Кличка	Klichka	Varchar(15)
Рост	Rost	Numeric
Вес	Ves	Numeric(9,3)
Теплолюбивость	Teplolub	Numeric(1)

Имя объекта: Местожительство	**Имя таблицы: Mestog**	
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Mesto	Numeric
Корпус	Korp	Varchar(10)
Номер клетки	Nomer	Numeric

Имя объекта: Потомство	**Имя таблицы: Potom**	
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Potom	Numeric
Код родителя	PK_ZverR	Numeric
Родство	Rodst	Varchar(4)
Код животного	PK_Zver	Numeric



Имя объекта: Сведения о прививках		Имя таблицы: Sved
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Sved	Numeric
Код животного	PK_Zver	Numeric
Код прививки	PK_Priv	Numeric
Дата	Data	Date
Доза	Doza	Numeric(9,3)

Имя объекта: Прививки		Имя таблицы: Privivki
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Potom	Numeric
Название	Nazv	Varchar(20)

Имя объекта: Подопечные		Имя таблицы: Podopech
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Podop	Numeric
Код животного	PK_Zver	Numeric
Код служащего	PK_Slug	Numeric

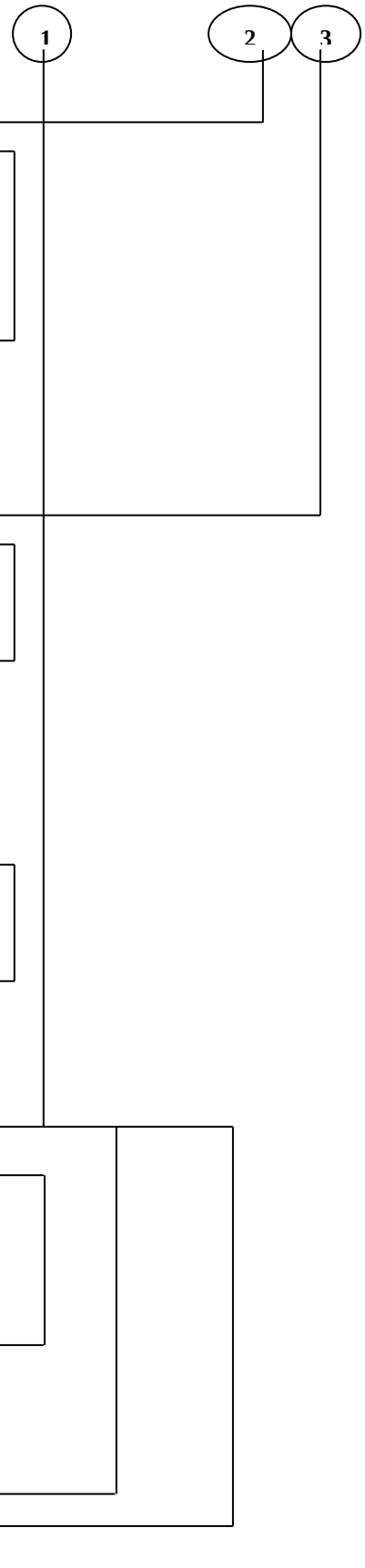
Имя объекта: Служащие		Имя таблицы: Slug
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Slug	Numeric
ФИО	FIO	Varchar(45)
Дата рождения	Drogd	Date
Пол	Pol	Char(1)
Дата приема	Priem	Date
Дата увольнения	Uvoln	Date
З/плата	Zpl	Numeric(9,2)
Код должности	PK_Dolg	Numeric

Имя объекта: Должность		Имя таблицы: Dolg
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Dolg	Numeric
Наименование	Naim	Varchar(15)

Имя объекта: Вид		Имя таблицы: Vid
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Vid	Numeric
Название	Naz	Varchar(15)
Код класса	PK_Klass	Numeric

Имя объекта: Класс		Имя таблицы: Klass
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Klass	Numeric
Название	Naz	Varchar(15)

Имя объекта: Совместимость		Имя таблицы: Sovm
Атрибут	**Поле**	**Тип**
Первичный ключ	PK_Sovm	Numeric
Код_Кто	PK_Vid	Numeric
Код_С_кем	PK_Vid	Numeric



3.4 Лабораторная работа №4

Тема: Разработка логической и физической модели в технологии IDEF1X.

Задание: Используя концептуальную модель из второй лабораторной работы, построить логическую модель в технологии IDEF1X. Предусмотреть поддержание ссылочной целостности. На основании логической модели разработать физическую модель для СУБД PostgreSQL.

Теоретический материал

3.4.1 Модель «сущность-связь»

Одна из наиболее сложных проблем проектирования базы данных связана с тем, что проектировщики, программисты и конечные пользователи, как правило, рассматривают данные и их назначение по-разному. Разработанный проект позволит удовлетворить все требования пользователей только при том условии, что и проектировщики, и пользователи придут к единому пониманию того, как работает данная конкретная организация. Чтобы добиться полного понимания характера данных и способов их использования в организации, необходимо применять в процессе обмена информацией между специалистами общую модель, которая не усложнена техническими подробностями и не допускает двойных толкований. Одним из примеров модели такого типа является модель "сущность-связь" (Entity-Relationship model, или ER-модель). ER-моделирование представляет собой нисходящий подход к проектированию базы данных, который начинается с выявления наиболее важных данных, называемых сущностями (entities), и связей (relationships) между данными, которые должны быть представлены в модели. Затем в модель вносятся дополнительные сведения, например, указывается информация о сущностях и связях, называемая *атрибутами* (attributes), а также все ограничения, относящиеся к сущностям, связям и атрибутам. ER-модель может быть представлена различными способами. Одним из способов представления является методология IDEF1X.

3.4.2 Методология IDEF1X

Метод IDEF1, разработанный Т.Рэмей (T.Ramey), также основан на подходе П.Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. В настоящее время на основе совершенствования методологии IDEF1 создана ее новая версия - методология IDEF1X. IDEF1X разработана с учетом таких требований, как простота изучения и возможность автоматизации. IDEF1X-диаграммы используются рядом распространенных CASE-средств (в частности, ERwin, Design/IDEF, Toad Data Modeler Freeware).

Объекты модели называются сущностями (Entity). Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. Атрибут выражает определенное свойство объекта. С точки зрения БД сущности соответствует таблица, экземпляру сущности - строка в таблице, а атрибуту - колонка таблицы.

Построение модели предполагает определение сущностей и атрибутов,

Сущность в методологии IDEF1X является независимой от идентификаторов или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется зависимой от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности.

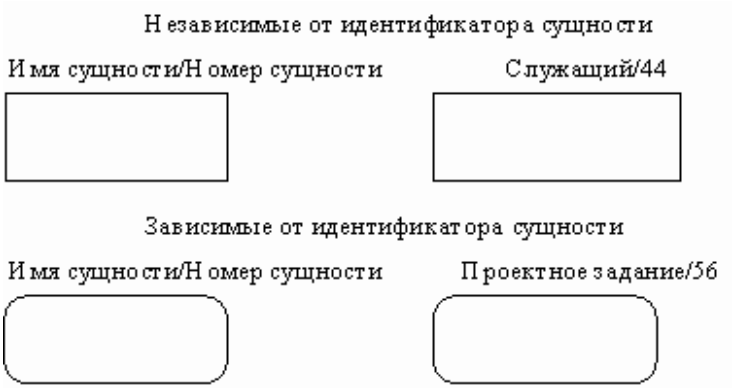


Рисунок 4 Изображение независимых и зависимых сущностей

Каждой сущности присваивается уникальное имя и номер, разделяемые косой чертой "/" и помещаемые над блоком. Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой.

Связь может дополнительно определяться с помощью указания степени или мощности (количества экземпляров сущности-потомка, которое может существовать для каждого экземпляра сущности-родителя). В IDEF1X могут быть выражены следующие мощности связей:

- каждый экземпляр сущности-родителя может иметь ноль, один или более связанных с ним экземпляров сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется ***идентифицирующей,*** в противном случае - ***неидентифицирующей.***

Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком с точкой на конце линии у сущности-потомка.

Идентифицирующая связь между сущностью-родителем и сущностью-потомком изображается сплошной линией. Сущность-потомок в идентифицирующей связи является зависимой от идентификатора сущностью. Сущность-родитель в идентифицирующей связи может быть как независимой, так и зависимой от идентификатора сущностью (это определяется ее связями с другими сущностями).

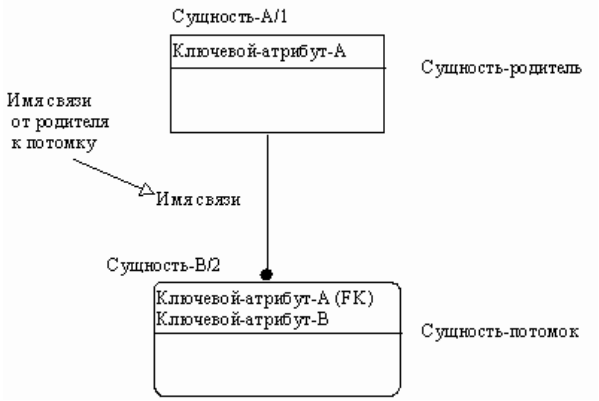


Рисунок 5 Пример идентифицирующей связи

Пунктирная линия изображает неидентифицирующую связь. Сущность-потомок в неидентифицирующей связи будет независимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей связи.

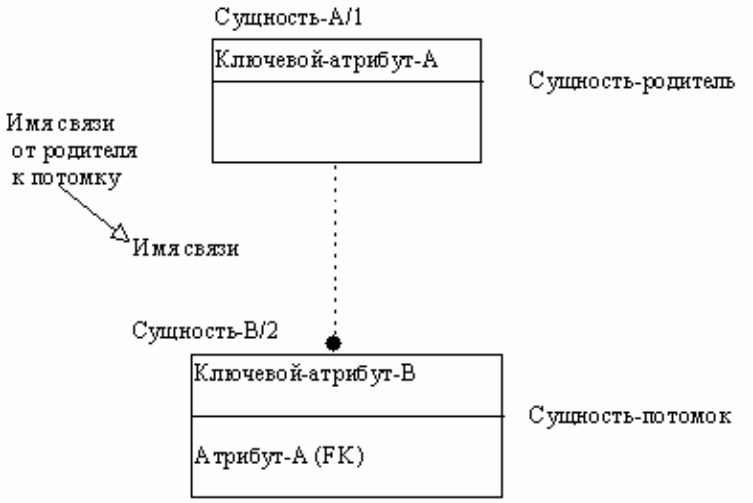


Рисунок 6 Пример неидентифицирующей связи

Отдельные свойства сущностей называются *атрибутами.* Например, сущность Staff (Персонал) может быть описана с помощью атрибутов staffNo (Табельный номер работника), name (Имя), position (Должность) и salary (Зарплата). Атрибуты содержат значения, которые описывают каждый экземпляр сущности и составляют основную часть информации, сохраняемой в базе данных.

Атрибуты изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой.

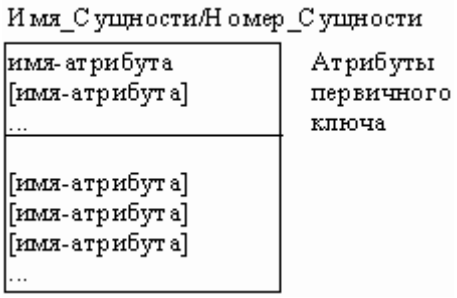


Рисунок 7 Пример определения атрибутов.

Сущности могут иметь также внешние ключи (Foreign Key), которые могут использоваться в качестве части или целого первичного ключа или неключевого атрибута. Внешний ключ изображается с помощью помещения внутрь блока сущности имен атрибутов, после которых следуют буквы FK в скобках.

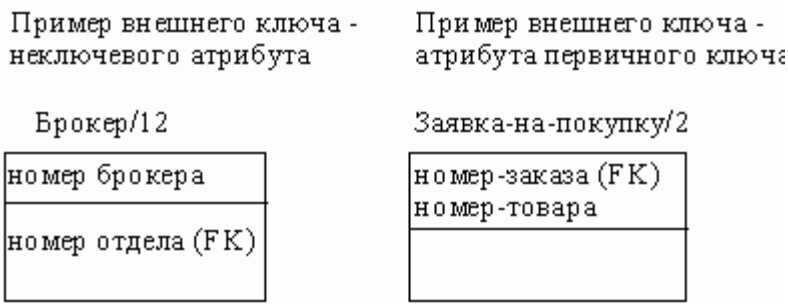


Рисунок 8 Перенос атрибутов

3.4.3 Создание модели данных с помощью Toad Data Modeler Freeware

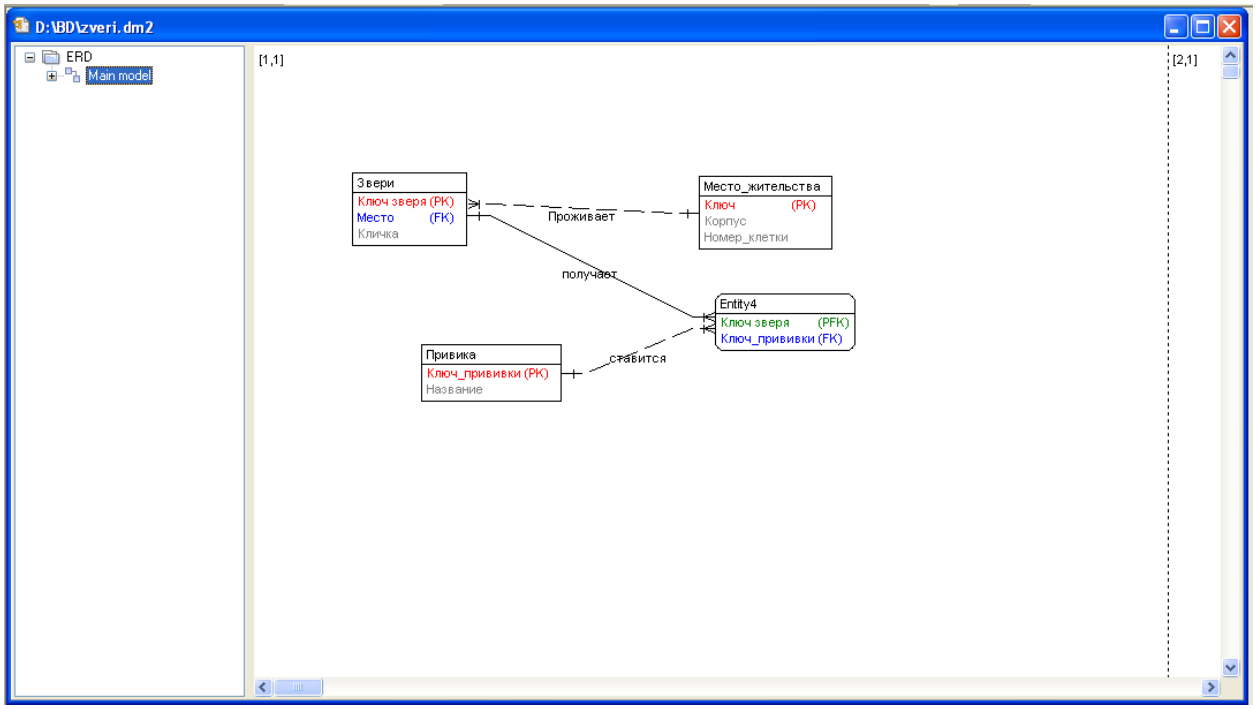


Рисунок 9. Логическая модель в Toad Data Modeler

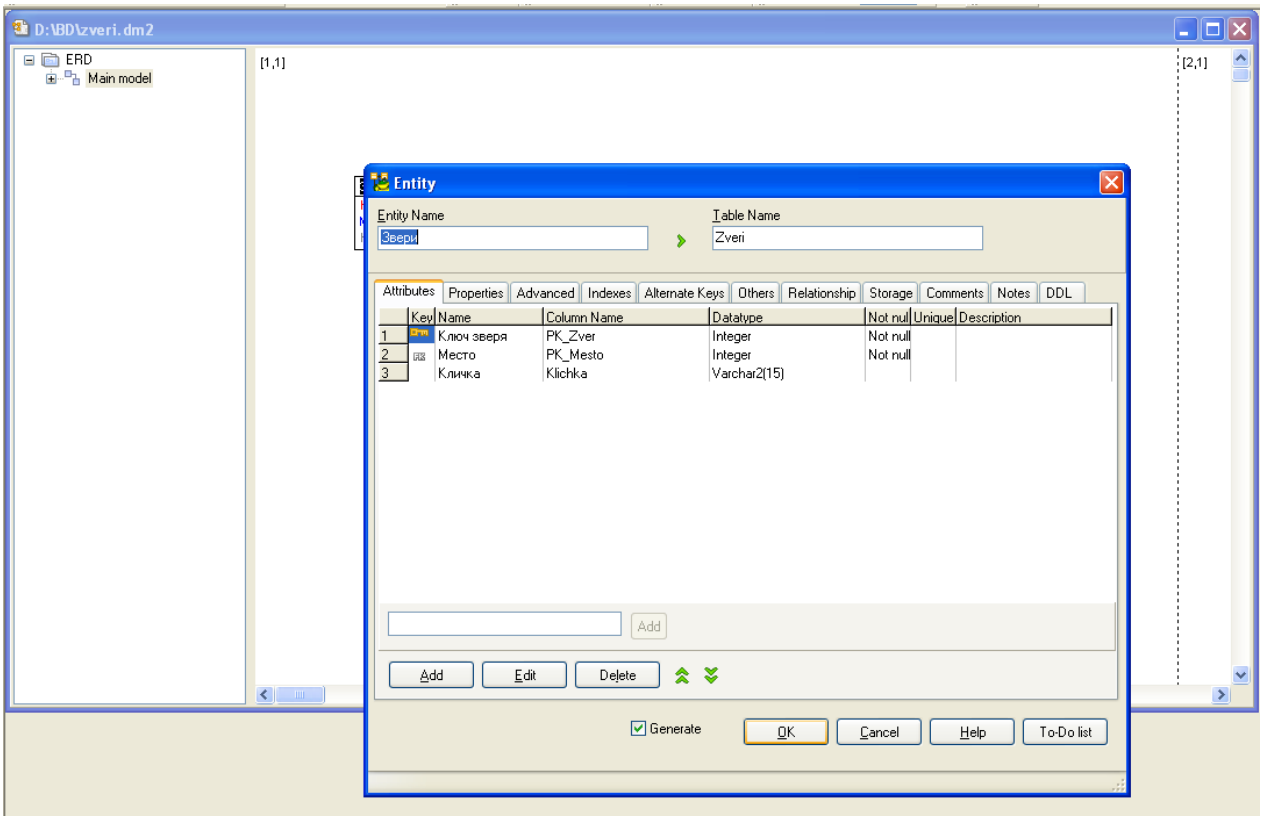


Рисунок 10. Окно добавления сущности в модель.

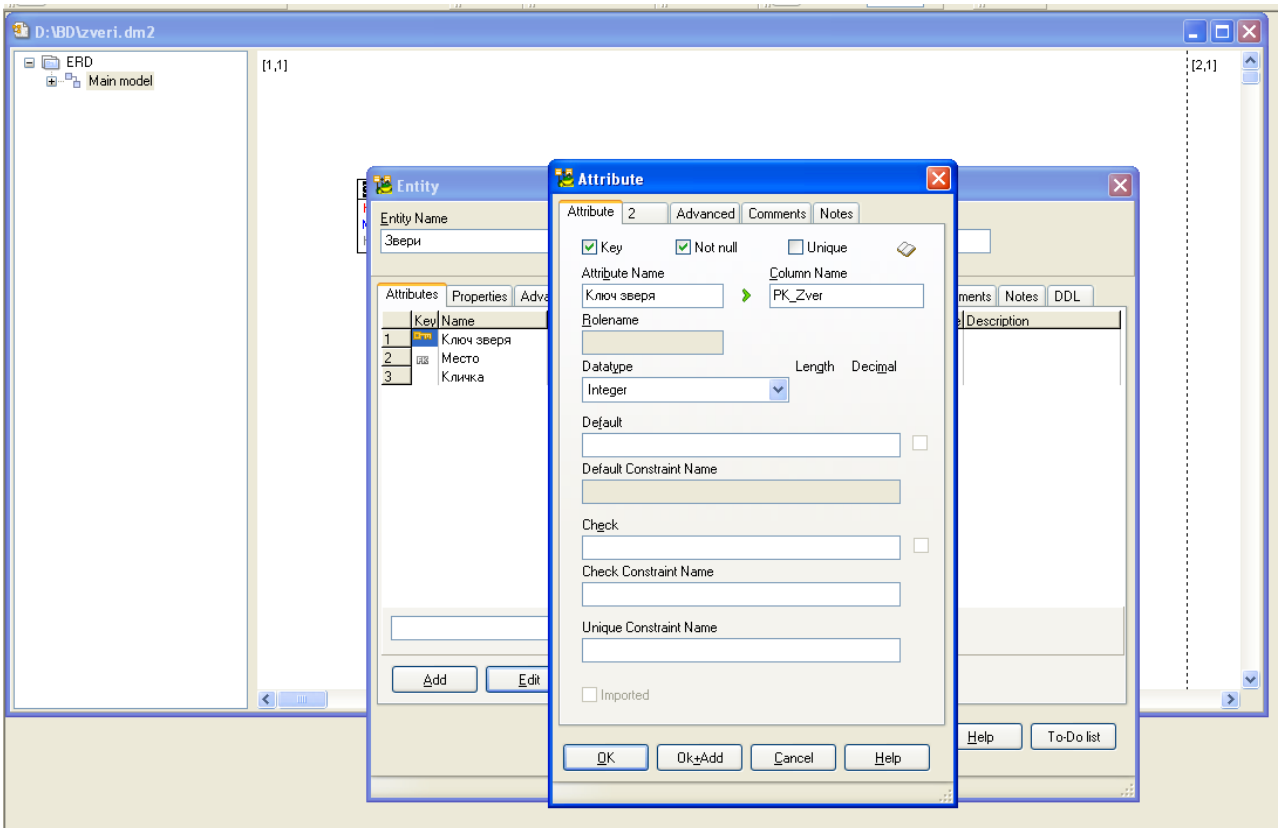


Рисунок 11. Добавление первичного ключа.

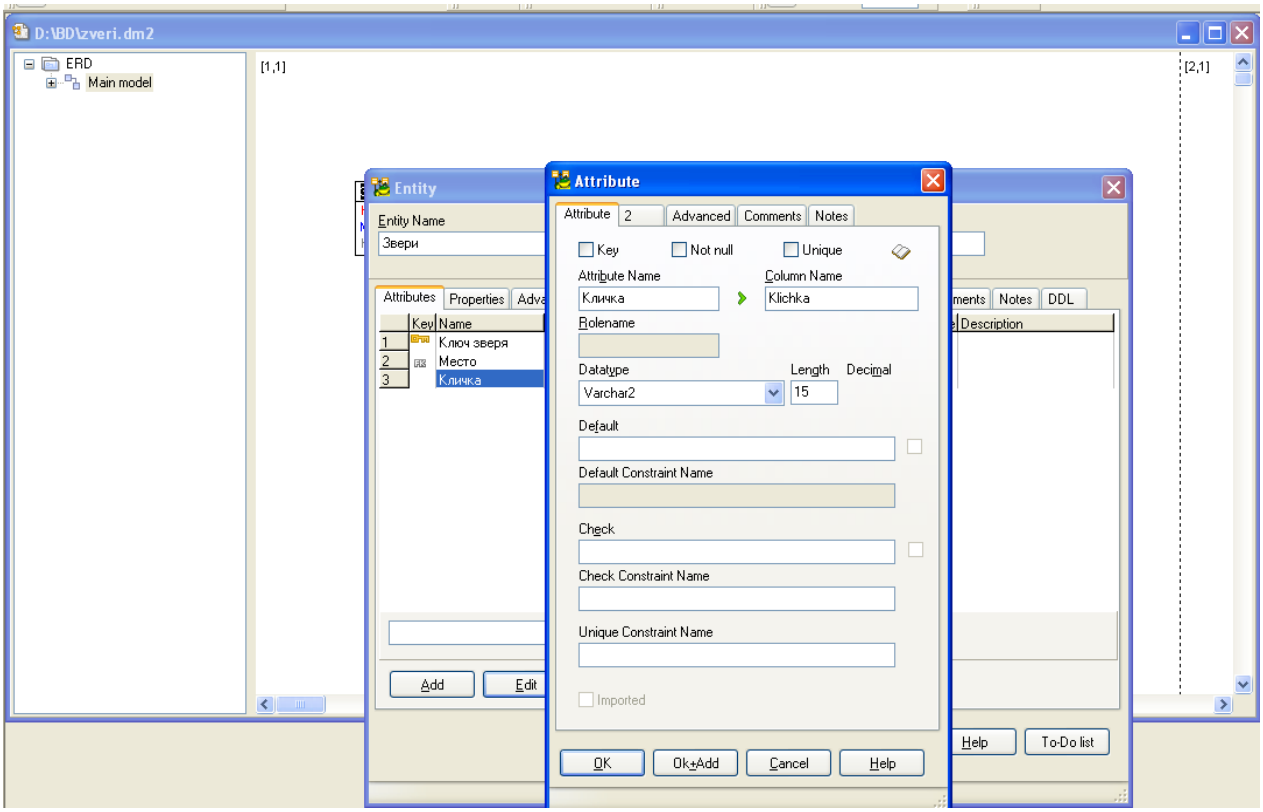


Рисунок 12. Добавление обычного атрибута.

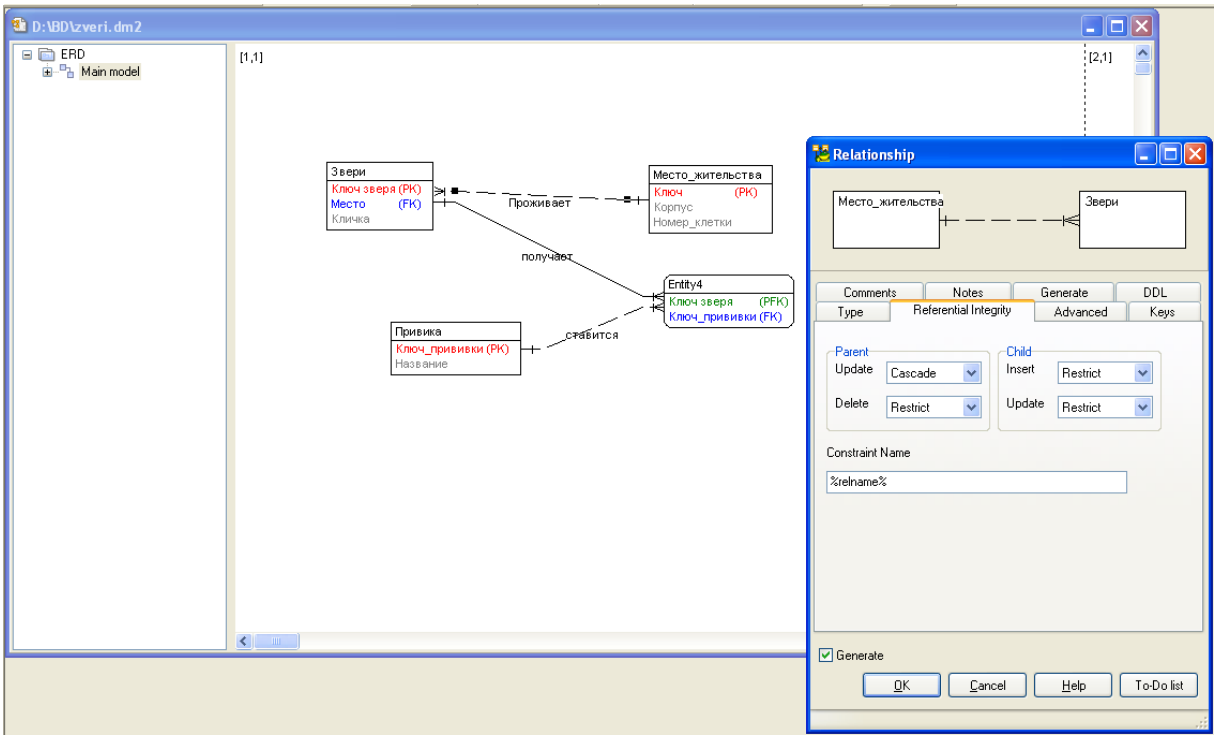


Рисунок 13. Установление ссылочной целостности в окне свойств связи.

3.4.4 Проблемы ER-моделирования

Существуют проблемы, которые принято называть *дефектами соединения* (connection trap). Они обычно возникают «вследствие неправильной интерпретации смысла некоторых связей». Рассмотрим два основных типа дефектов соединения: *дефект типа "разветвление"* (fan trap) и *дефект типа "разрыв"* (chasm trap), а также способы их выявления и устранения в создаваемых ER-моделях. В общем случае для выявления дефектов соединения необходимо убедиться в том, что смысл каждой связи определен четко и ясно. При недостаточном понимании сути установленных связей может быть создана модель, которая не будет являться истинным представлением реального мира.

Дефекты типа "разветвление"

Дефект типа "разветвление" **имеет место в том случае, когда модель отображает связь между типами сущностей, но путь между отдельными сущностями этого типа определен неоднозначно.**

Дефект типа "разветвление" возникает в том случае, когда две или несколько связей типа 1:* исходят из одной сущности. Потенциальный дефект типа "разветвление" показан на рис. 9, на котором две связи типа 1:* (Has и Operates) исходят из одной и той же сущности Division.

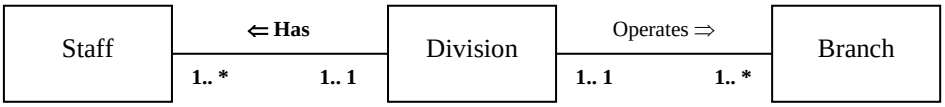


Рисунок 14 Пример дефекта типа "разветвление"

На основании этой модели можно сделать вывод, что один отдел (Division) может состоять из нескольких отделений компании (Branch) и в нем может работать многочисленный штат сотрудников. Проблемы начинаются при попытках выяснить, в каком отделении компании работает каждый из сотрудников отдела.

Неспособность дать точный ответ на поставленный вопрос является результатом дефекта типа «разветвление», связанного с неправильной интерпретацией связей между сущностями Staff, Division и Branch. Устранить эту проблему можно путем перестройки ER-модели для представления правильного взаимодействия этих сущностей таким образом, как показано на рис. 10.

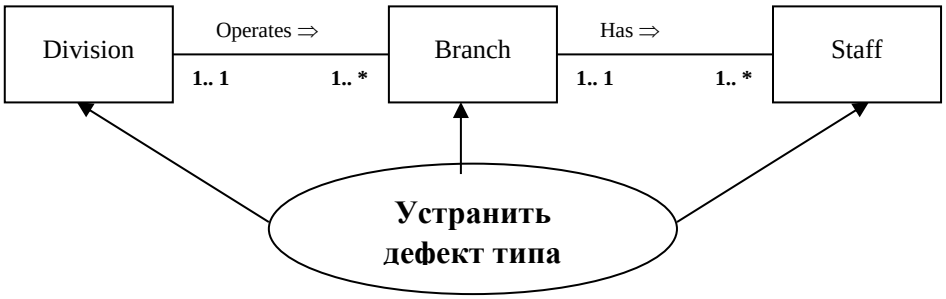


Рисунок 15 Пример переработки ER-модели (рис.14) с целью устранения дефекта типа "разветвление"

Если проверить эту структуру на уровне отдельных связей *Operates к Has*, можно убедиться, что теперь легко дать однозначный ответ на поставленный выше вопрос.

Дефекты типа "разрыв"

Дефект типа «разрыв» появляется в том случае, когда в модели предполагается наличие связи между типами сущностей, но не существует пути между отдельными сущностями этих типов.

Дефект типа "разрыв" может возникать, если существует одна или несколько связей с минимальной кратностью, равной нулю (которая обозначает необязательное участие), и эти связи составляют часть пути между взаимосвязанными сущностями. На рис. 11 потенциальный дефект типа "разрыв" показан на примере связей между сущностями Branch, Staff и PropertyForRent.

Рассмотрев эту модель, можно сделать вывод, что одно отделение компании имеет много сотрудников, которые работают со сдаваемыми в аренду объектами. Однако не все сотрудники непосредственно работают с объектами и не все сдаваемые в аренду объекты недвижимости в каждый конкретный момент находятся в ведении кого-либо из сотрудников компании. В данном случае проблема возникает, когда необходимо выяснить, какие объекты недвижимости приписаны к тому или иному отделению компании.

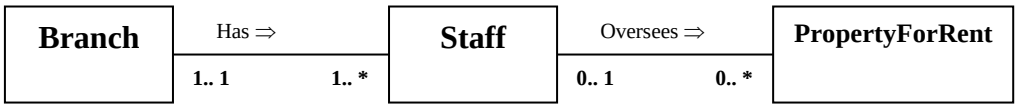


Рисунок 16 Пример дефекта типа "разрыв"

Попробуем ответить на следующий вопрос: "Какое отделение компании отвечает за работу с объектом под номером 'PA14'? К сожалению, на данный вопрос нельзя дать ответ, если этот объект в текущий момент не связан ни с одним из сотрудников, работающих в каком-либо из отделений компании. Неспособность дать ответ на заданный вопрос

рассматривается как утрата информации (поскольку известно, что любой объект недвижимости *должен* быть приписан к какому-то отделению компании) в результате которой и возникает дефект типа "разрыв". Кратность сущности Staff и PropertyForRent в связи Oversees имеет минимальное значение, равное нулю, а это означает, что некоторые объекты недвижимости не могут быть связаны с отделением компании с помощью информации о сотрудниках. Поэтому для разрешения этой проблемы следует ввести недостающую связь *Offers* между сущностями Branch и PropertyForRent. ER-модель, показанная на рис. 12, отображает истинные связи между этими сущностями. Такая структура гарантирует, что всегда будут известны объекты недвижимости, связанные с каждым отделением компании, включая объекты недвижимости которые в данный момент не поручены никому из сотрудников этой компаний.

Рисунок 17 ER-модель, представленная на рис.16, после переработки с целью устранения дефекта типа "разрыв"

3.5 Лабораторная работа №5

Тема: Построение SQL-запросов.

Задание: Сгенерировать базу данных на сервере, используя созданную в четвертой лабораторной работе модель. Заполнить базу данных. Продемонстрировать умение составлять различные запросы к базе данных.

Теоретический материал

SQL (Structured Query Language, язык структурированных запросов) - это специальный язык, используемый для определения данных, доступа к данным и их обработки. SQL относится к *непроцедурным (nonprocedural)* языкам - он лишь описывает нужные компоненты (например, таблицы) и желаемые результаты, не указывая, как именно эти результаты должны быть получены. Каждая реализация SQL является надстройкой над *процессором базы данных (database engine),* который интерпретирует операторы SQL и определяет порядок обращения к структурам БД для корректного и эффективного формирования желаемого результата.

SQL является «подъязыком данных», который предназначен только для использования в качестве языка взаимодействия с базой данных. Сам по себе SQL не содержит тех средств, которые необходимы для разработки законченных программ, и может использоваться в виде одной из трех прикладных реализаций:

1. **Интерактивный или автономный SQL** дает возможность пользователям непосредственно извлекать информацию из базы данных или записывать ее в базу.
2. **Статический SQL** – фиксированный (исполняемый), записанный заранее, а не генерируемый во время выполнения программы код SQL, который обычно используется в приложениях. Существуют две версии статического SQL. Встроенный SQL – это код SQL, включенный в код исходного текста программы, написанной на другом языке. Другое использование статического SQL – модульный язык. В этом случае модули SQL скомпонованы с с модулями кода других языков.
3. **Динамический SQL** – код SQL, сгенерированный приложением во время исполнения. Он заменяет статистический SQL в тех случаях, когда необходимый код SQL еще не может быть определен во время написания приложения, так как сам код зависит от того, какой выбор сделает пользователь.

SQL отличается от языков программирования высокого уровня несколькими признаками. Во-первых, он относиться к непроцедурным языкам. На языке типа С можно записать для компьютера шаг за шагом все инструкции, необходимые для исполнения задания. SQL просто декларирует, что нужно делать, а исполнение возлагает на СУБД. Такой подход лежит в русле философии реляционных баз данных. СУБД в данном случае рассматривается как «черный ящик»: что делается внутри него – пользователя не касается. Его интересует только получение правильного ответа из базы данных и внесение в нее необходимых изменений. Другим отличием SQL является трехзначная логика..

Данные содержаться в таблицах, таблицы сгруппированы в схемы, а схемы – в каталоги. Каталоги могут быть в дальнейшем сгруппированы в кластеры. В некоторых приложениях баз данных эти термины несколько отличаются от определений стандарта.

С точки зрения конкретного сеанса SQL кластер содержит все таблицы, к которым имеется доступ в данном сеансе.

Схемой называется именованный набор объектов базы данных, управляемых одним пользователем и в определенных случаях рассматриваемых как единое целое.

Рекомендуется использовать домены, позволяющие построить более точную классификацию данных по типам, чем та, которая достигается с помощью стандартного

набора типов данных. Например, телефонные номера относятся не к тому типу данных, к которому принадлежат номера социальных страховок, даже если и те и другие выражаются числами, в то же время, информацию одного и того же типа данных иногда не имеет смысла сравнивать, так как она может принадлежать разным доменам.

Определение домена содержит тип данных, но может, кроме того, включать предложения, которые определяют значения по умолчанию, ограничения и последовательность сортировки для упорядочения наборов символов для домена (под ограничениями понимаются правила, ограничивающие значения данных, которые разрешено размещать в определенном столбце).

Язык SQL состоит из двух специальных наборов команд. DDL (Data Definition Language, язык определения данных) - это подмножество SQL, используемое для определения и модификации различных структур данных, а DML (Data Manipulation Language, язык манипулирования данными) - это подмножество SQL, применяемое для получения и обработки данных, хранящихся в структурах, определенных ранее с помощью DDL. DDL состоит из большого количества команд, необходимых для создания таблиц, индексов, представлений и ограничений, а в DML входит всего четыре оператора:

INSERT

Добавляет данные в базу данных.

UPDATE

Изменяет данные в базе данных.

DELETE

Удаляет данные из базы данных.

SELECT

Извлекает данные из базы данных.

Некоторым кажется, что применение DDL является прерогативой администраторов базы данных, а операторы DML должны писать разработчики, но эти два языка не так-то просто разделить. Сложно организовать эффективный доступ к данным и их обработку, не понимая, какие структуры доступны и как они связаны. Также сложно проектировать соответствующие структуры, не зная, как они будут обрабатываться. Сказав это, сосредоточимся на DML. DDL в книге будет встречаться лишь тогда, когда это необходимо для иллюстрации применения DML. Причины особого внимания к DML таковы:

- DDL хорошо описан во многих книгах по проектированию и администрированию баз данных, а также в справочниках по SQL.
- Проблемы производительности обычно бывают вызваны неэффективными операторами DML.
- Хотя операторов всего четыре, DML - большая тема.

Эффективное хранение и извлечение информации сейчас важно как никогда ранее:

- **Все** больше компаний предлагают свои услуги через Интернет. В часы пик они вынуждены обслуживать тысячи параллельных запросов, и задержки означают прямую потерю прибыли. Для таких систем каждый оператор SQL должен быть тщательно продуман, чтобы обеспечивать требуемую производительность при увеличении объема данных.
- Сегодня есть возможность хранить гораздо больше данных, чем пять лет назад. Один дисковый массив вмещает десятки терабайт данных, и уже не за горами хранение сотен терабайт. Программное обеспечение, применяемое для загрузки и анализа данных в этих средах, должно использовать весь потенциал SQL, чтобы обрабатывать неизменно увеличивающийся объем данных за постоянные (или сокращающиеся) промежутки времени.

Оператор SELECT

Оператор SELECT используется для извлечения данных из базы. Множество данных, извлекаемое оператором SELECT, называется *результирующим множеством (result set).* Как и таблица, результирующее множество состоит из строк и столбцов, что позволяет заполнить таблицу данными результирующего множества. Общий вид оператора SELECT таков:
SELECT <один или несколько обьектов>
FROM <одно или несколько мест>
WHERE <ни одного, одно или несколько условий>

Инструкции SELECT и FROM необходимы, а вот инструкция WHERE необязательна (хотя и она почти всегда используется). Начнем с простого примера, извлекающего три столбца из каждой строки таблицы CUSTOMER (заказчики):

SELECT cust_nbr, name, region.id FROM customer;

Инструкция WHERE не была использована, и никаких ограничений на данные мы не наложили, поэтому запрос возвращает все строки таблицы заказчиков. Если необходимо ограничить возвращаемый набор данных, можно добавить в оператор инструкцию WHERE с одним условием:

SELECT cust_nbr, name, region_id FROM customer
WHERE region_id = 8;

Теперь результирующее множество содержит только заказчиков, проживающих в области с идентификатором region_id, равным 8. Но что если нужно ссылаться на область по имени, а не по номеру? Можно выбрать нужное имя из таблицы REGION, а затем, зная region_id, обратиться к таблице CUSTOMER. Чтобы не писать два разных запроса, можно получить тот же результат с помощью одного запроса, использующего *объединение (join):*

SELECT customer.cust_nbr, customer.name, region.name FROM customer, region WHERE region.name = 'New England' AND region.region_id = customer.region_ld;

Теперь в инструкции FROM не одна таблица, а две, и инструкция WHERE содержит *условие объединения (join condition),* которое указывает, что таблицы заказчиков и областей должны быть объединены по столбцу region_id, имеющемуся в каждой таблице.

Так как обе таблицы содержат столбец с названием *name,* необходимо как-то определить, какой именно столбец вас интересует. В предыдущем примере это делается с помощью точечной нотации - добавления через точку имени таблицы перед именем столбца. Если же на написание полных имен таблиц у вас уходит слишком много времени, назначьте для каждого названия таблицы в инструкции FROM *псевдоним (alias)* и используйте его вместо имени в инструкциях SELECT и WHERE:

SELECT c.cust_nbr, c.name, r.name FROM customer c, region r WHERE r.name = 'New England' AND r.region_id = c.region_id;

В этом примере псевдоним «с» был присвоен таблице заказчиков, а псевдоним «г» - таблице областей. Теперь можно в инструкциях SELECT и WHERE писать «с» вместо «customer» и «г» вместо «region».

Элементы инструкции SELECT

Рассмотренные ранее результирующие множества, порожденные нашими запросами, содержали столбцы одной или нескольких таблиц. Как правило, элементами инструкции SELECT действительно являются ссылки на столбцы; среди них также могут встречаться:

• Константы, такие как числа (1) или строки ('abc')

• Выражения, например shape, diameter * 3.1415927

• Функции, такие как TO_DATE(01-JAN-2002', 'DD-MON-YYYY')

• Псевдостолбцы, например ROWID, ROWNUM или LEVEL

Если первые три пункта в списке довольно просты, то последний нуждается в пояснении. В СУБД могут быть столбцы-призраки, называемые *псевдостолбцами (pseudocolumns),* которые не присутствуют ни в одной таблице. Их значения появляются во время выполнения запроса, и в некоторых ситуациях они бывают полезны.

Упорядочение результатов

В общем случае нет гарантии, что результирующее множество будет сформировано в каком-либо определенном порядке. Если нужно отсортировать результаты по одному или нескольким столбцам, следует добавить инструкцию ORDER BY сразу же после WHERE. Следующий пример сортирует заказчиков из Новой Англии по фамилиям:

```
SELECT c.cust_nbr, c.name, r.name FROM customer c,
region r WHERE r.name = 'New England'
AND r.reglon_id = c.region_id ORDER BY c.name;
```

Сортируемые столбцы можно определять по их положению в инструкции SELECT. Отсортируем предыдущий запрос по столбцу CUST_NBR (номер заказчика), который в инструкции SELECT указан первым:

```
SELECT c.cust_nbr, c.name, r.name FROM customer c,
region r WHERE r.name = 'New England' AND r.region_id
= c.reglon_id ORDER BY 1;
```

Указание ключей сортировки по позиции сэкономит вам немного времени, но может привести к ошибкам, если в дальнейшем порядок следования столбцов в инструкции SELECT будет изменен.

Удаление дубликатов

В некоторых случаях результирующее множество может содержать одинаковые данные. Например, при формировании списка проданных за последний месяц деталей те детали, которые присутствовали в нескольких заказах, встретятся в результирующем множестве несколько раз. Чтобы исключить повторы, вставьте в инструкцию SELECT ключевое слово DISTINCT:

```
SELECT DISTINCT li.part_nbr
FROM cust_order co, line_item li
WHERE co.order_dt >= TO_DATE('01-JUL-2001•,'DD-MON-YYYY')
  AND co.order_dt < TO_DATE('01-AUG-2001','DD-MON-YYYY•)
  AND co.order_nbr = li.order_nbr;
```

Запрос возвращает множество отличающихся друг от друга записей о деталях, заказанных в течение июля 2001 года. Без ключевого слова DISTINCT вывод содержал бы по одной строке для каждой строки каждого заказа, и одна деталь могла бы встречаться несколько раз, если бы она содержалась в нескольких заказах. Принимая решение о включении DISTINCT в инструкцию SELECT, следует иметь в виду, что поиск и удаление дубликатов требует сортировки, которая будет служить дополнительной нагрузкой при выполнении запроса.

Оператор INSERT

Оператор INSERT - это механизм загрузки данных в базу данных. За один раз данные можно вставить только в одну таблицу, зато брать вставляемые данные можно из нескольких дополнительных таблиц. Вставляя данные в таблицу, не нужно указывать значения для каждого столбца, однако следует обратить внимание на то, допускают ли столбцы использование значений NULL[1] или же нет.

В операторе INSERT необходимо указать значения как минимум для тех столбцов, которые содержат пометку NOT NULL. Например, как показано ниже:

```
INSERT INTO employee (emp_id, lname, dept_id)
    VALUES (101, 'Smith', 2);
```

Количество элементов в инструкции VALUES должно совпадать с количеством элементов в списке столбцов, а их типы данных должны соответствовать определениям столбцов. В нашем примере emp_id и dept_id хранят численные значения, а lname - строковое, поэтому оператор INSERT выполнится без ошибок.

Иногда вставляемые данные нужно предварительно извлечь из одной или нескольких таблиц. Так как оператор SELECT формирует результирующее множество, состоящее из строк и столбцов, то можно непосредственно передать его оператору INSERT:

```
INSERT INTO employee (emp_id, fname, lname, dept_id, hire_date)
  SELECT 101, 'Dave', 'Smith', d.dept_ld, SYSDATE
      FROM department d WHERE d.name = 'Accounting';
```

В данном примере оператор SELECT извлекает идентификатор отдела для бухгалтерии (Accounting). Остальные четыре столбца в операторе SELECT представлены константами.

Оператор DELETE

Оператор DELETE обеспечивает удаление данных из базы. Как и SELECT, оператор DELETE содержит инструкцию WHERE с условиями для идентификации удаляемых строк. Забыв указать инструкцию WHERE в операторе DELETE, вы удалите все строки из указанной таблицы. Следующий оператор удаляет всех сотрудников с фамилией Hooper из таблицы EMPLOYEE:

```
DELETE FROM employee WHERE lname = 'Hooper';
```

Иногда значения, необходимые для построения условия в инструкции WHERE, располагаются в других таблицах. Например, решение компании вынести вовне функции бухучета потребует удаления всего бухгалтерского персонала из таблицы EMPLOYEE:

```
DELETE FROM employee WHERE dept_id = (SELECT dept.id FROM department WHERE name
= 'Accounting');
```

Подобное использование оператора SELECT носит название *подзапроса (subquery)*.

Оператор UPDATE

С помощью оператора UPDATE вносятся изменения в существующие данные. Как и DELETE, оператор UPDATE включает в себя инструкцию WHERE для указания тех строк, которые будут изменены. Посмотрим, как можно предоставить 10-процентное повышение зарплаты тем, у кого годовой доход меньше 40 000 долларов:

```
    UPDATE employee
```

```
SET salary = salary • 1.1
WHERE salary < 40000;
```

Если необходимо изменить несколько столбцов, вы можете выбрать один из двух вариантов: задать набор пар столбец-значение, разделенных запятыми, или указать набор столбцов и подзапрос. Два следующих оператора UPDATE изменяют столбцы inactive_dt и inactive_ind в таблице CUSTOMER для клиентов, не сделавших ни одного заказа за последний год:

```
UPDATE customer
SET lnactive_dt = SYSDATE, inactive_ind = 'Y'
WHERE last_order_dt < SYSDATE - 365;

UPDATE customer SET (inactive_dt, inactive_ind) =
  (SELECT SYSDATE, Y FROM dual) WHERE last_order_dt < SYSDATE - 365;
```

Подзапрос во втором примере выглядит немного неестественно, так как он обращается к таблице dual для построения результирующего множества, состоящего из двух констант; он приведен для иллюстрации использования подзапросов в операторе UPDATE.

CREATE TABLE

Создает таблицу.

Пример простейшей команды по созданию таблицы.

```
CREATE TABLE dept
  (deptno  NUMBER (2) PRIMARY KEY,
   dname   VARCHAR2(10),
   loc     VARCHAR2(9) )
```

CREATE SEQUENCE

Создает sequence. Sequence - это объект базы данных необходимый для того, чтобы несколько пользователей могли генерировать уникальное целое значение. Обычно sequences используется для автоматической генерации значения первичного ключа.

Когда sequence генерирует число, его значение увеличивается. Если два пользователя пытаются одновременно получить значение одного и того же sequence, то сначала генерируется значение для первого одного пользователя, а затем для другого. Пользователь не может получить значение сгенерированное для другого пользователя

Когда sequence создан, вы можете получить доступ к его значениям в SQL - выражениях с помощью псевдоколонок CURRVAL (возвращает текущее значение sequence) или NEXTVAL (увеличивает значение sequence и возвращает это новое значение).

```
CREATE SEQUENCE eseq   INCREMENT BY 10;
```

При первом обращении к ESEQ.NEXTVAL возвратит 1.При втором возвратит 11. И т.д.

```
CREATE SEQUENCE ADM.GURSEQ INCREMENT BY 1 START WITH 10 CYCLE;
```

Объединения

Часто бывает необходима информация из нескольких таблиц. Конструкция языка SQL, комбинирующая данные двух и более таблиц, называется *объединением (join).* В данной главе будут рассмотрены объединения, их типы и способы использования.

Объединение - это SQL-запрос, который извлекает информацию из двух или более таблиц или представлений. При указании в инструкции FROM нескольких таблиц или представлений СУБД выполняет объединение, связывая вместе строки различных таблиц. Существует несколько типов объединений:

Внутренние объединения (inner joins)

Внутренние объединения - это стандартный вариант объединения, который возвращает строки, удовлетворяющие условию объединения. Каждая строка, возвращенная внутренним объединением, содержит данные всех таблиц, включенных в объединение.

Внешние объединения (outer join)

Внешние объединения - это расширение внутренних. Внешнее объединение возвращает строки, удовлетворяющие условию объединения, а также те строки одной таблицы, для которых не найдено строк другой таблицы, отвечающих условию объединения.

Внутренние объединения

Внутреннее объединение возвращает строки, удовлетворяющие условию объединения. Давайте рассмотрим понятие «объединение» на примере. Пусть необходимо вывести фамилию и название подразделения для каждого сотрудника. Используем следующий оператор SQL:

```
SELECT E.LNAME, D.NAME
FROM EMPLOYEE E, DEPARTMENT D
WHERE E.DEPT_ID = D.DEPT_ID;
```

В этом примере запрос обращается к двум таблицам, так как фамилия служащего хранится в таблице EMPLOYEE, а название подразделения - в таблице DEPARTMENT. Обратите внимание на то, что в инструкции FROM названия двух таблиц, EMPLOYEE и DEPARTMENT, перечислены через запятую. Если нужно объединить три и более таблиц, укажите все таблицы в инструкции FROM, перечислив их через запятую. В списке оператора SELECT могут упоминаться столбцы из любой таблицы, указанной в инструкции FROM.

Условие объединения

Обычно при выполнении объединения в инструкцию WHERE включается условие, которое устанавливает соответствие таблиц, указанных в инструкции FROM. Такое условие называется условием объединения. Условие объединения определяет, как следует объединять строки одной таблицы со строками другой. Как правило, условие объединения применяется к столбцам, которые являются внешними ключами таблиц.

В первом примере предыдущего раздела в инструкции WHERE было задано условие объединения, в котором указывалось равенство столбцов DEPT_ID таблицы EMPLOYEE и таблицы DEPARTMENT:

```
WHERE E.DEPTJD = D.DEPT.ID
```

Для выполнения объединения СУБД берет одну комбинацию строк из двух таблиц и проверяет истинность условия объединения. Если условие объединения истинно, СУБД включает данную комбинацию строк в результирующее множество. Процесс повторяется для всех сочетаний строк двух таблиц. Приведем несколько важных фактов, касающихся условия объединения.

• Нет необходимости включать столбцы, входящие в условие объединения, в список SELECT. В следующем примере условие объединения содержит столбец DEPT_ID таблицы EMPLOYEE и таблицы DEPARTMENT, но при этом столбец DEPT_ID не участвует в выборке:

```
SELECT E.LNAME, D.NAME
FROM EMPLOYEE E, DEPARTMENT D
WHERE E.DEPT_ID = D.DEPT.ID;
```

• Обычно условие объединения указывается для столбцов, являющихся внешним ключом одной таблицы и первичным или уникальным ключом другой таблицы. Однако можно использовать и другие столбцы. Каждое условие объединения затрагивает столбцы, которые устанавливают связь между двумя таблицами.

• Условие объединения может включать в себя несколько столбцов. Так обычно бывает, если внешний ключ состоит из нескольких столбцов.

• Общее количество условий объединения всегда на единицу меньше общего количества таблиц.

• Условия объединения должны содержать столбцы с совместимыми типами данных. Обратите внимание на то, что типы данных должны быть *совместимыми,* но не обязаны *совпадать.*

• Оператор равенства (=) не обязательно должен входить в условие объединения. Возможно использование других операторов. В объединениях могут участвовать операторы, о которых будет рассказано далее в этом разделе.

Внешние объединения

При объединении двух таблиц может возникнуть необходимость вывести все строки одной из таблиц, даже если для них не существует соответствующих строк во второй таблице. Рассмотрим две таблицы: поставщиков (SUPPLIER) и деталей (PART):

SELECT * FROM SUPPLIER;

SUPPLIER_ID	NAME
101	Pacific Disks, Inc.
102	Silicon Valley Microchips
103	Blue River Electronics

SELECT * FROM PART;

PART_NBR	NAME	SUPPLIER_ID	STATUS	INVENTORY_QTY	UNIT_COST	RESUPPLY_DATE
HD211	20 GB Hard Disk	101	ACTIVE	5	2000	12-DEC-OO
P3000	3000 MHz Processor	102	ACTIVE	12	600	03-NOV-OO

Если нужно вывести всех поставщиков и поставляемые ими детали, естественно использовать следующий запрос:

SELECTS.SUPPLIER.ID, S. NAME SUPPLIER.NAME, P.PARTNBR, P.NAME PART_NAME
FROM SUPPLIER S, PART P
WHERE S.SUPPLIER_ID = P.SUPPLIER_ID;

SUPPLIERJD	SUPPLIER_NAME	PART_NBR	PART_NAME
101	Pacific Disks, Inc.	HD211	20 GB Hard Disk
102	Silicon Valley Microchips	P3000	3000 MHz Processor

Обратите внимание на то, что, хотя поставщиков трое, запрос выводит только двоих, потому что третий поставщик (Blue River Electronics) в данный момент ничего не поставляет. Когда СУБД выполняет объединение между таблицами SUPPLIER и PART, сопоставляются столбцы SUPPLIER_ID этих двух таблиц (как указано в условии объединения). Так как для SUPPLIER_ID = 103 не существует соответствующих записей в таблице PART, этот

поставщик не включается в результирующее множество. Такой тип объединения является наиболее естественным и называется *внутренним объединением.*

Понятие внутреннего объединения легче пояснить в терминах декартова произведения. При выполнении объединения таблиц SUPPLIER и PART сначала формируется декартово произведение (физически оно не материализуется), а затем условия инструкции WHERE ограничивают результат только теми строками, в которых совпадают значения SUPPLIER_ID.

Но хотелось бы получить полный список поставщиков, включающий и тех, кто в данный момент ничего не поставляет. СУБД предоставляет специальный тип объединения, который позволяет включать в результирующее множество строки одной таблицы, для которых не найдены соответствующие строки в другой таблице. Такое объединение называется *внешним (outer).* Внешнее объединение позволит вывести строки для всех поставщиков, а если поставщик в настоящий момент поставляет какие-то детали, то и соответствующие строки деталей. Если в настоящий момент поставщик не поставляет детали, в результирующем множестве в столбцах таблицы PART будут возвращены значения NULL.

Синтаксис внешнего объединения несколько отличается от синтаксиса внутреннего объединения. Применяется специальный оператор, называемый *оператором внешнего объединения,* который выглядит как знак «плюс», заключенный в круглые скобки, то есть (+). Этот оператор используется в условии объединения инструкции WHERE вслед за именем поля той таблицы, которую вы хотите рассматривать как необязательную. В рассматриваемом примере про детали и поставщиков таблица PART не содержит информацию об одном поставщике. Просто добавляем оператор (+) к условию объединения со стороны таблицы PART. Запрос и результирующее множество будут выглядеть следующим образом:

```
SELECT S.SUPPLIER_ID, S.NAME SUPPLIER_NAME, P.PART_NBR, P.NAME
PART_NAME
    FROM SUPPLIER S, PART P
    WHERE S.SUPPLIER_ID = P.SUPPLIER_ID (+);
```

SUPPLIER_ID	SUPPLIER_NAME	PART_NBR	PART_NAME
101	Pacific Disks, Inc.	HD211	20 GB Hard Disk
102	Silicon Valley Microchips	P3000	3000 MHz Processor
103	Blue River Electronics		

Заметьте, что оператор (+) следует за P.SUPPLIER_ID, что делает таблицу PART необязательной (в данном объединении). Если поставщик ничего не поставляет в настоящий момент, СУБД создаст для данного поставщика запись в таблице PART со значениями NULL во всех ячейках. Результирующее множество теперь содержит всех поставщиков независимо от состояния их текущих поставок. Как видите, столбцы PART для поставщика с идентификатором 103 содержат NULL.

Оператор внешнего объединения (+) может появляться как в левой, так и в правой части условия объединения. Вы только должны быть уверены в том, что применяете оператор к соответствующей таблице (в контексте данного запроса). Например, если поменять местами части оператора равенства из предыдущего примера, это никак не повлияет на результат:

```
SELECT S.SUPPLIER_ID, S.NAME SUPPLIER.NAME, P.PART_NBR, P.NAME
PART.NAME
    FROM SUPPLIER S, PART P
    WHERE P.SUPPLIER_ID (+) = S.SUPPLIER_ID;
```

SUPPLIER_ID	SUPPLIERJAME	PART_NBR	PART_NAME
101	Pacific Disks, Inc.	HD211	20 GB Hard Disk
102	Silicon Valley Microchips	P3000	3000 MHz Processor
103	Blue River Electronics		

Ограничения, налагаемые на внешние объединения

Существует ряд правил и ограничений, относящихся к использованию внешних объединений в запросах. Если в запросе выполняется внешнее объединение, СУБД не разрешает использовать в этом же запросе некоторые другие операции. Далее мы поговорим о таких ограничениях и о некоторых способах их обхода.

• Оператор внешнего объединения может присутствовать только в одной части условия объединения. При попытке использовать его в обеих частях возникает ошибка. Например:

SELECT S.SUPPLIER.ID, S.NAME SUPPLIER.NAME, P.PART.NBR, P.NAME PART. NAME
FROM SUPPLIER S, PART P
WHERE S.SUPPLIER.ID (+) = P,SUPPLIER.ID (+);

WHERE S.SUPPLIER.ID (+) = P.SUPPLIER_ID (+)
*
ERROR at line 3:
a predicate may reference only one outer-joined table

• Если в объединении участвует более двух таблиц, то каждая из таблиц в запросе не может участвовать во внешнем объединении с более чем одной другой таблицей.

• В условии внешнего объединения, содержащем оператор (+), запрещено использование оператора IN. Например:

SELECT E.LNAME, J.FUNCTION
FROM EMPLOYEE E, JOB J
WHERE E.JOB.ID (+) IN (66B, 670, 667);

WHERE E.JOB.ID (+) IN (668, 670, 667)
ERROR at line 3:
outer join operator (+) not allowed in operand of OR or IN

• Условие внешнего объединения, содержащее оператор (+), нельзя комбинировать с другими условиями при помощи оператора OR. Например:

SELECT E.LNAME, D.NAME
FROM EMPLOYEE E, DEPARTMENT D
WHERE E.DEPT.ID = D.DEPT.ID (+) OR D.DEPT.ID =10;

WHERE E.DEPT_ID = D.DEPT_ID (+)
ERROR at line 3:
outer join operator (+) not allowed in operand of OR or IN

• Условие внешнего объединения, содержащее оператор (+), не может содержать подзапрос. Например:

SELECT E.LNAME
FROM EMPLOYEE E
WHERE E.DEPT_ID (+) =
(SELECT DEPT_ID FROM DEPARTMENT WHERE NAME = 'ACCOUNTING');

(SELECT DEPT_ID FROM DEPARTMENT WHERE NAME = 'ACCOUNTING')
ERROR at line 4:

a column may not be outer-joined to a subquery

Чтобы достичь желаемого эффекта и избежать ошибки, можно использовать встроенное представление:

```
SELECT E.LNAME
FROM EMPLOYEE E,
(SELECT DEPT_ID FROM DEPARTMENT WHERE NAME = 'ACCOUNTING') V
WHERE E.DEPTJD (+) = V.DEPT_ID;
```

Групповые операции

В повседневной работе SQL-программист часто имеет дело с групповыми операциями. Используя SQL для доступа к базе данных, часто задаются вопросы, подобные перечисленным ниже:

• Какова максимальная заработная плата в данном подразделении?

• Сколько в каждом подразделении менеджеров?

• Сколько заказчиков существует для каждого продукта?

• Можно ли вывести среднемесячное значение продаж для каждого региона?

Для ответа на такие вопросы необходимы групповые операции. СУБД предоставляют широкий спектр возможностей по обработке групповых операций, в том числе обобщающие функции, инструкции GROUP BY и HAVING.

Обобщающие функции

Если говорить по существу, *обобщающая функция (aggregate function)* суммирует результаты выражения для некоторого количества строк, возвращая одно значение. Синтаксис большинства обобщающих функций таков:

обобщающая_функция([DISTINCT | ALL] *выражение*)

Приведем перечень элементов конструкции:

обобщающая функция

Указывает имя функции, например SUM, COUNT, AVG, MAX, MIN и др.

DISTINCT

Указывает, что обобщающая функция должна учитывать только неповторяющиеся значения *выражения.*

ALL

Указывает, что обобщающая функция должна учитывать все значения *выражения,* в том числе и все дублирующиеся. По умолчанию считается, что использовано ALL.

выражение

Указывает столбец или любое другое выражение, по которому необходимо выполнить обобщение.

Давайте рассмотрим простой пример. Для нахождения максимальной зарплаты сотрудников SQL-оператор использует функцию MAX:

```
SELECT MAX(SALARY) FROM EMPLOYEE;
MAX(SALARY)
5000
```

Инструкция GROUP BY

Инструкция GROUP BY, используемая совместно с обобщающими функциями, разбивает результирующее множество на несколько групп, а затем для каждой группы выдается одна строка сводной информации. Например, если нужно вычислить общее количество заказов каждого клиента, выполним следующий запрос:

```
SELECT CUST.NBR, COUNT(ORDER_NBR)
FROM CUST.ORDER
```

GROUP BY CUST.NBR;

Запрос выдает одну сводную строку для каждого клиента. В этом заключается суть запроса GROUP BY. Мы просим СУБД сгруппировать (GROUP) результаты по номеру клиента (BY CUST_NBR), поэтому для каждого уникального значения CUST_NBR порождается одна строка вывода. Каждое значение для определенного клиента представляет собой сводную информацию по всем строкам данного клиента.

Необобщенное выражение CUST_NBR из списка SELECT присутствует и в инструкции GROUP BY. Если в списке SELECT присутствует смесь обобщенных и необобщенных значений, SQL считает, что вы собираетесь выполнить операцию GROUP BY, поэтому все необобщенные выражения должны быть указаны и в инструкции GROUP BY. Если этого не сделать, SQL выдаст сообщение об ошибке.

Аналогично, если не включить все необобщенные выражения списка SELECT в инструкцию GROUP BY, то SQL выдаст такую ошибку:

```
SELECT CUST_NBR, SALES_EMP_ID, COUNT(ORDER_NBR)
FROM CUST_ORDER
GROUP BY CUST_NBR;
SELECT CUST_NBR, SALES_EMP_ID, COUNT(ORDER_NBR)
ERROR at line 1:
ORA-00979: not a GROUP BY expression
```

Наконец, не разрешено использование групповой (обобщающей) функции в инструкции GROUP BY. При попытке такого использования, как в приведенном ниже примере, вы получите следующее сообщение об ошибке:

```
SELECT CUST_NBR, COUNT(ORDER_NBR)
FROM CUST_ORDER
GROUP BY CUST_NBR, COUNT(ORDER_NBR);
GROUP BY CUST_NBR, COUNT(ORDER_NBR)
*
ERROR at line 3:
ORA-00934: group function is not allowed here
```

Инструкция HAVING

Инструкция HAVING тесно связана с инструкцией GROUP BY. Инструкция HAVING используется для наложения фильтра на группы, созданные инструкцией GROUP BY. Если запрос содержит инструкцию HAVING и инструкцию GROUP BY, результирующее множество будет содержать только те группы, которые удовлетворяют условию, указанному в инструкции HAVING. Давайте рассмотрим несколько примеров, иллюстрирующих вышесказанное. Приведенный ниже запрос возвращает количество заказов каждого клиента:

```
SELECT CUST_NBR, COUNT(ORDER_NBR)
FROM CUST_ORDER
GROUP BY CUST_NBR
HAVING CUST_NBR < 260;
```

CUST_NBR	COUNT(ORDERJIBR)
201	2
231	6
244	2
255	6

Заметьте, что в выводе присутствуют только клиенты с номерами, меньшими, чем 260. Это объясняется тем, что в инструкции HAVING указано условие CUST_NBR < 260. Количество заказов подсчитывается для всех клиентов, но выводятся только те группы, для которых выполнено условие инструкции HAVING.

Этот пример является не очень удачной иллюстрацией возможностей инструкции HAVING; в данном случае она просто указывает данные, которые не должны включаться в результирующее множество. Было бы эффективнее использовать WHERE CUST.NBR < 260, а не HAVING CUST.NBR < 260, так как **инструкция WHERE исключает строки из рассмотрения до проведения группировки, а HAVING устраняет уже созданные группы**. Правильнее было бы записать предыдущий запрос так:

```
SELECT CUST_NBR, COUNT(OROER_NBR)
FROM CUST_ORDER
WHERE CUST_NBR < 260;
```

Следующий пример демонстрирует более удачное применение инструкции HAVING:

```
SELECT CUST_NBR, COUNT(OROER_NBR)
FROM CUST_ORDER
GROUP BY CUST.NBR
HAVING COUNT(ORDER_NBR) > 2;
```

Обратите внимание на использование в инструкции HAVING обобщающей функции. Здесь инструкция HAVING применена надлежащим образом, так как результат выполнения обобщающей функции доступен только после проведения группировки.

Синтаксис инструкции HAVING подобен синтаксису инструкции WHERE. Но для условия инструкции HAVING существует одно ограничение. Это условие может относиться только к выражениям списка SELECT или инструкции GROUP BY. Если указать в инструкции HAVING выражение, не содержащееся ни в списке SELECT, ни в инструкции GROUP BY, то в ответ будет выдано сообщение об ошибке.

3.6 Лабораторная работа №6

Тема: Разработка серверной части программы.

Задание: Разработать серверную часть программы с использованием пакетов, процедур, функций и триггеров, реализованных на языке PL/pgSQL.

Литература

1. Ананьев П.И., Кайгородова М.А. Основы баз данных. [Электронный ресурс]: Учебное пособие/Алт. госуд. технич. ун-т им. И.И. Ползунова.-Барнаул: 2010.-189.-ил. - Режим доступа: http//elib.altstu.ru, свободный.
2. Архипенков С и др. Хранилища данных. От концепции до внедрения. – М: Диалог-МИФИ, 2002, 424 с.
3. Г. Буч. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд./Пер. с англ. - М.: “ Вильямс ”, 2008г. - 721 с., ил.
4. Гайдамакин Н.А. Автоматизация информационных систем, базы и банки данных. – М.: Гелиос АРВ, 2002. – 368с.
5. Гарсиа-Молина, Гектор, Ульман, Джеффри Д., Уидом, Дженнифер. Системы баз данных. Полный курс, Вильямс, 2003.
6. Дейт К. Введение в системы баз данных. М.:Вильямс,2008.-1328с. с ил.
7. Кайт, Том. Oracle для профессионалов: архитектура, методики программирования и основные особенности версий 9i и 10g – М.: Вильямс, 2007.- 848с.
8. Карпова И. П. Базы данных: курс лекций и материалы для практических занятий /И. П. Карпова.-Санкт-Петербург: Питер, 2013.-240 с.: ил.-(Учебное пособие) ISBN 978-5-496-00546-3р.388.70.-1000
9. Коннолли, Томас, Бегг, Каролин. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 1440 с.: ил. – Парал. тит. англ.
10. Кузин А.В. Базы данных : [учеб. пособие для вузов по направлению подгот. 654600 "Информатика и вычисл. техника"]. - 3-е изд., стер. - М. : Академия, 2008. - 314, [1] с. : ил. - (Высшее профессиональное образование. Информатика и вычислительная техника). - Библиогр.: с. 313. - 2000 экз. - **ISBN** 978-5-7695-5775-0 (15 экз).
11. Маклаков С.В.. BPwin и Erwin. CASE-средства разработки информационных систем. - М.: ДИАЛОГ-МИФИ, 1999 - 256 с.
12. Малыхина М.П. Базы данных: основы, проектирование, использование, 2-е изд. перераб. и доп. – СПб.: БХВ-Петербург, 2007.- 512с.
13. Мишра С., Бьюли А. Секреты Oracle SQL. – Пер. с англ. – СПб: Символ-Плюс, 2005. – 368 с., ил.
14. Ролланд, Фред, Основные концепции баз данных, Вильямс, 2002.