Министерство образования и науки Российской Федерации
Ярославский государственный университет им. П.Г. Демидова
Кафедра теоретической информатики

Проектирование реляционной базы данных и интерфейса

Методические указания лабораторного практикума по дисциплинам специализации «Технология программирования», «СУБД Oracle» и «Визуальные системы программирования»

Ярославль 2007

Составитель В.С. Рублев

ББК

Проектирование реляционной базы данных и интерфейса: Метод. указания лабораторного практикума по дисциплинам специализации «Метрология и качество программного обеспечения» (для специальности «Математическое обеспечение и администрирование информационных систем») и «СУБД Oracle» и «Визуальные системы программирования» (для специальности «Прикладная математика и информатика»)

Рецензент – кафедра Вычислительных и программных систем Ярославского государственного университета им. П.Г. Демидова

1. Цель работы

1. Обучение методам проектирования схемы реляционной базы данных (БД) и программного обеспечения БД.
2. Обучение методам разработки интерфейса работы с БД в визуальных системах программирования.

2. Общее задание

1. Разработать схему реляционной базы данных (в 4-й нормальной форме – каждое данное кроме идентифицирующих присутствует только в одной таблице) заданной динамической ситуации и пакет процедур и функций работы с этой базой данных, при помощи которого можно динамически изменять и показывать информацию по любому объекту и списку. Предусмотреть реакцию на недопустимые данные.
2. Разработать интерфейс «связанных таблиц» работы с реляционной БД в визуальной системе программирования, включающий отображение основных таблиц БД, меню выбора главной таблицы и кнопки или меню изменения данных таблиц. При выборе записи в главной таблице программа должна отображать в связанных с ней таблицах данные, относящиеся к выбранной записи.
3. Разработать «разворачивающийся» интерфейс работы с реляционной БД в визуальной системе программирования, который отображает выбранную главную таблицу вместе со столбцами, связывающими ее с вспомогательными таблицами. При двойном щелчке левой кнопкой мыши на таком столбце должна появляться всплывающая таблица, содержащая данные, относящиеся к выбранной записи главной таблицы.
4. Разработать полный набор тестов для каждой процедуры и функции и полный набор тестов для интерфейса в каждом из видов («связанных таблиц» или «разворачивающийся»).
5. Составить отчет, содержащий:
    1) постановку задачи (общего и индивидуального задания);
    2) схему реляционной БД с обоснованием выбора схемы;
    3) пакет функций и процедур с подробными комментариями к каждой процедуры и функции в спецификации пакета (смысл процедуры или функции, назначение параметров, дополнительно возвращаемые функцией значения в случае ошибочной ситуации с аргументами функции).
    4) программы интерфейса «связанных таблиц» в визуальной системе программирования;
    5) программы «разворачивающегося» интерфейса в визуальной системе программирования;
    6) набор тестов (входные данные и ожидаемые результаты) и программы тестирования интерфейса того или иного вида.

3. Варианты индивидуальных заданий

1. *Сессия* определяется изменяющимся списком сдаваемых *студентами предметов*, зависящих от *курса* студентов и *оценками*, полученными каждым студентом по этим предметам, а также состоянием студента:
    1) *подготовка* – сессия не закончена (не все предметы сдавались студентом);
    2) *перевод на* следующий *курс* – сессия сдана успешно, а курс студента не последний;
    3) *отчисление* – сессия закончена и более чем по 2 предметам оценки неудовлетворительные;
    4) *окончание* – сессия сдана успешно и курс студента последний.
2. *Назначение научного руководителя* определяется для *студентов*, специализирующихся по кафедре работы преподавателя, имеющих курс не ниже 2-го:
    1) *без руководителя* – курс ниже 3-го;
    2) *перевод на следующий курс* (отчисление после последнего курса);
    3) *назначение руководителя* – 2-й курс;
    4) *изменение руководителя* – курс выше 3-го.
3. *Предприятие* определяется *отделами* и списками *работников* отделов с их стажами. Во главе каждого отдела стоит один из работников отдела – *руководитель*. Состояния работника:

1) прием – стаж равен 0;
2) выслуга – стаж увеличивается на 1;
3) проводы на пенсию – достижение пенсионного стажа;
4) назначение руководителем – если отдел не имеет руководителя;
5) перевод в другой отдел.

4. *Автобусные маршруты*, каждый из которых определяется номером и кольцевым списком *остановок,* а каждый *автобус* определяется номером машины и номером маршрута. Состояния автобуса:
   1) снят с маршрута – новый или в ремонте;
   2) назначение на маршрут;
   3) изменение маршрута;
   4) списание.

5. *Вступительные экзамены* на *факультеты,* каждый из которых имеет свой список экзаменов и проходной балл, а *абитуриент* может подать на несколько факультетов и определяется состоянием:
   1) *подача заявления* – не сформирован список факультетов для абитуриента;
   2) *экзамены* – список факультетов абитуриента сформирован, но не сданы все экзамены;
   3) *выбор факультета* – все экзамены сданы;
   4) *зачисление* – если на выбранный факультет набран проходной балл.

   По окончанию зачисления всех абитуриентов формирование списков зачисленных.

6. *Прием на работу* со списками *рабочих мест,* определяемых профессией и ставкой, и *претендентов,* определяемых профессиями и рейтингом (квалификацией). Состояния претендента:
   1) заявление – определяются профессии претендента и рейтинг;
   2) выбор места – все профессии и рейтинг определены;
   3) зачисление – рейтинг претендента больше рейтингов других на это место.
   Формирование списка зачисленных претендентов с местами.

7. *Библиотечный абонемент* со списками *книг* и *читателей,* для каждого из которых определяется состояние:
   1) *заявка* – формирование списка заявленных книг;
   2) *чтение* – формирование списка прочитанных книг;
   3) *обмен* – сдача прочитанных книг, получение заявленных книг, если они свободны.
   Библиотека имеет 3 состояния:
   1) поступление – книга поступает новая или из ремонта;
   2) ремонт – книга поступает в ремонт;
   3) прием-выдача книг от читателей.

8. *Рынок* со списками *товаров* (с количеством и ценой) и очередью *покупателей* (с кошельком – количество денег). Состояние покупателя:
   1) занять очередь – стать последним; при этом список покупок пустой;
   2) стоять в очереди – определить список необходимых покупок, если не первый и не последний в очереди;
   3) купить – если первый в очереди; если денег не хватило, снова занять очередь.

9. *Пассажирские перевозки* со списками *поездов* (номер, список остановок, остановка местонахождения) и *пассажиров* (откуда, куда). Состояние поезда:
   1) формирование – пополнение списка остановок;
   2) *остановка* – посадка-высадка пассажиров;
   3) движение – список остановок, список пассажиров поезда не изменяются.

10. *Обед в столовой* со списками *блюд* (с ценой) и очередью *студентов* (с кошельком – количество денег). Состояние студента:
   1) занять очередь – стать последним; при этом список блюд пустой;
   2) стоять в очереди – определить список блюд на обед, если не первый и не последний в очереди;
   3) заплатить – если первый в очереди; если денег не хватило, снова занять очередь.

11. *Авиаперевозки* со списками *аэропортов*, *рейсов* (аэропорт вылета, аэропорт назначения, список аэропортов следования, вместимость пассажиров, аэропорт местоположения) и пассажиров (аэропорт вылета, аэропорт назначения). Состояние рейса:
    1) *посадка-высадка* – выходят пассажиры, чей аэропорт назначения совпадает с аэропортом местоположения самолета, и садятся пассажиры, чей аэропорт назначения совпадает с одним из последующих в списке аэропортов следования;
    2) *полет* – список пассажиров не меняется; местоположения самолета меняется на следующий аэропорт следования.

    Состояния пассажира:
    1) *без билета* – определяются аэропорты вылета и назначения;
    2) *с билетом на посадку* – определяется подходящий рейс;
    3) *полет*;
    4) *высадка* – когда самолет достиг аэропорта назначения пассажира.

12. *Хоккей* со списками команд (список игроков, список на игру), хоккеистов (команда, разряд, забитые шайбы, количество сыгранных игр, количество добытых очков, начальный разряд = 5) и игр (команда-хозяин, команда-гость, упорядоченный список игроков, забивших шайбы). Состояния игры:
    1) *определить игру* – определить команды и списки игроков на игру;
    2) *начать игру* – опустошить список игроков, забивших шайбы;
    3) *закончить игру* – определить список игроков, забивших шайбы; повысить разряд (уменьшить значение) для хоккеистов, для которых отношение добытых очков к сыгранным играм >2 и в этом случае сбрасывает число игр на 0, а число добытых очков делает равным (разряд-4)*10.

13. *Самодеятельность* со списком *кружков* (руководитель, список учеников кружка), списком *учеников* (список кружков ученика), списком *руководителей* (список кружков руководителя, список учеников руководителя). Состояния ученика:
    1) *желание* – определение списка кружков;
    2) *поступление* – включение в списки кружков;
    3) *уход* – исключение из кружка.

14. *Школа* со списком *классов* (список учеников класса), списком *учеников*, списком *учителей* (список классов учителя, стаж). Состояния учителя:
    1) *поступление* – стаж равен 0, определяются классы;
    2) *преподавание* – стаж по событию увеличивается на 1;
    3) *выход на пенсию* – при достижении пенсионного стажа.

15. *Метро* со списком *линий* (список станций линии, список поездов линии), список *поездов* (номер, линия), списком *станций.*

16. *Ресторан* со списком *столиков*, списком *официантов* (список столиков официанта, число собранных «штанов» за неуплату), списком *блюд* (цена), списком *посетителей* (кошелек, список желаемых блюд). Состояния столика: занят, свободен. Состояние посетителя:
    1) *ожидание* – ждет свободного столика;
    2) *заказ* – определяет список блюд;
    3) *обед*;
    4) *расчет* – оплачивает обед; если денег в кошельке мало, расплата «штанами»; столик освобождается.

17. *Командный шахматный турнир* со списком *команд* (список шахматистов команды) и списком *шахматистов* (разряд, команда, число сыгранных партий, число добытых очков, список соперников, с которыми сыграны партии). Состояния шахматиста:
    1) *зачисление в команду* – начальные значения параметров (разряд=5);
    2) *назначение на игру* – выбор соперника, с которым не играл;
    3) *игра* – изменение числа добытых очков в зависимости от введенного результата партии;
    4) *квалификация* – повышение разряда (меньше на 1), если отношение выигранных очков к числу сыгранных партий больше ½ и в этом случае сбрасывает число сыгранных партий на 0, а число очков на (разряд-5)*0,05);
    5) *отчисление* – если за турнир все партии проиграл.

18. *Читальный зал* со списком *книг* (список претендентов, занятость) и списком *читателей* (список взятых книг, список прочитанных книг, список заказа). Состояния читателя:
   1) запись – формирование заказа;
   2) сдача-выдача – сдача прочитанных книг, выдача тех книг заказа, которые свободны.

19. *Перевозки маршрутками* со списком *маршруток* (номер, список пунктов прохождения, пункт местонахождения, список пассажиров маршрутки) и списком *пассажиров* (пункты посадки и высадки, список подходящих маршруток). Состояния пассажира:
   1) появление пассажира – определяются пункты посадки, высадки и список подходящих маршруток;
   2) посадка – в маршрутку из списка подходящих, если ее пункт местонахождения совпадает с пунктом посадки;
   3) поездка – учет в списке пассажиров маршрутки;
   4) высадка – пункт местонахождения маршрутки совпадает с пунктом высадки.
   Состояния маршрутки:
   1) посадка – пункт местонахождения маршрутки совпадает с пунктами посадки пассажиров;
   2) движение – к следующему в списке пункту прохождения.

20. *Торговля* со списком *товаров* (цена, количество – пополняемое) и очередью *покупателей* (список товаров для покупки с количествами, кошелек - пополняемый). Состояния покупателя:
   1) занять очередь – определить список товаров для покупки;
   2) *касса* – рассчитаться за товары; если денег не хватило, повторно занять очередь с предварительным пополнением кошелька;
   3) *пополнение кошелька* – добавление денег.
   Состояния торговли:
   1) пополнение товарами – если нет покупателей или не хватило товара;
   2) *продажа* – расчет покупателя, если товаров хватает.

21. *Биржа труда* со списками *рабочих мест,* определяемых профессией и ставкой, и *безработных,* определяемых профессиями и рейтингом (квалификацией). Состояния безработного:
   1) постановка на учет – определяются профессии безработного и рейтинг;
   2) появление места – все профессии и рейтинг определены;
   3) зачисление – рейтинг безработного больше рейтингов других на это место.
   Формирование списка безработных, получивших работу с местами.

22. *Приемные экзамены* со списком *факультетов* (число мест, список экзаменов на факультет, список абитуриентов на факультет), списком *экзаменов* (список абитуриентов с оценками по экзамену) и списком *абитуриентов* (список заявленных абитуриентом факультетов, список экзаменов с оценками). Состояние абитуриента:
   1) подача заявления – определение ранжированного списка факультетов и списка экзаменов;
   2) экзамен – определение оценки; при неудовлетворительной оценке забирает документы;
   3) *зачисление* – при сдаче всех экзаменов после конкурса.
   Состояния приема:
   1) прием заявлений – определение списков факультетов и экзаменов;
   2) экзамен – определение оценок для всех сдающих этот экзамен абитуриентов;
   3) *конкурс* – после сдачи всех экзаменов по следующему алгоритму:
      1. ранжирование по набранным баллам абитуриентов на каждый факультет;
      2. зачисление абитуриентов на первый факультет в их списке, если они в списке факультета входят в число мест факультета;
      3. коррекция для каждого факультета числа оставшихся мест и оставшегося ранжированного списка абитуриентов; остановка алгоритма, если оставшихся мест ни на один факультет нет;
      4. коррекция для оставшихся абитуриентов их ранжированных списков факультетов вычеркиванием первого факультета; переход на п.2.

23. *Руководство студенческой научной работой* определяется списками *студентов* и преподавателей по кафедре специализации студента. Состояния студента:
   1) *без руководителя* – курс ниже 3-го;

2) *перевод на следующий курс* (отчисление после последнего курса);
3) *назначение руководителя* – 3-й курс;
4) *изменение руководителя* – курс выше 3-го.

24. *Зачетная сессия* определяется списками *студентов* (список зачетных предметов с отметкой зачет/незачет) и списком предметов (список сдающих зачет в зависимости от номера курса). Состояния студента:
1) *подготовка* – нет сданных зачетов;
2) *зачеты* – все зачеты определены;
3) *допуск в экзаменационную сессию* – срок прошел.

25. *Самодеятельность* со списками *руководителей* (список кружков руководителя и список учеников руководителя), *кружков* (руководитель и список учеников) и *учеников* (список кружков). Состояния ученика:
1) *запись в кружки;*
2) *работа в кружках*
3) *уход из кружка*.

26. *Работа* со списками *отделов* (руководитель, список работников отдела) и *работников* (отдел, стаж). Состояние работника:
прием – стаж равен 0;
повышение – увеличение стажа на 1, если стаж не пенсионный;
проводы на пенсию – достигнут пенсионный стаж.

27. *Трамвайные маршруты* со списками *маршрутов* (список трамваев маршрута, упорядоченный список остановок маршрута), *остановок* (список маршрутов), *трамваев* (список маршрутов). Состояние остановки:
1) *подготовка* – маршрутам не присвоена;
2) *определение маршрутов;*
3) *удаление остановки.*

28. *Библиотека* со списками *книг* (количество экземпляров), *читателей* (списки взятых, потребных и сдаваемых книг). Состояния читателя:
1) *заявка* – формирование списка потребных книг;
2) *чтение* – формирование списка сдаваемых книг;
3) *обмен* – сдача прочитанных книг, получение заявленных книг, если есть свободные экземпляры.

29. *Торговля* со списком *товаров* (цена, количество – пополняемое) и очередью *покупателей* (список товаров для покупки с количествами, кошелек - пополняемый). Состояния покупателя:
1) *занять очередь* – определить список товаров для покупки;
2) *пополнение кошелька* – добавление денег;
3) *покупка* – рассчитаться за товары; если товара не хватило, повторно занять очередь.
Состояния торговли:
1) *подвоз товаров* – если нет покупателей или не хватило товара;
2) *продажа* – расчет покупателя, если товаров хватает и денег у покупателя.

30. *Чемпионат по футболу* со списками *команд* (список футболистов, список игр), *футболистами* (разряд, число добытых очков, занятость в игре), *игр* (упорядоченный список игроков, забивших голы). Состояние футболиста:
1) *зачисление в команду* – начальные значения параметров (разряд=4);
2) *назначение на игру* – формирование списка на игру;
3) *игра* – формирование списка игроков, забивших голы;
4) *квалификация* – повышение разряда (меньше на 1), если отношение добытых очков к сыгранным играм >1,5 и в этом случае сбрасывает число игр на 0, а число добытых очков делает равным (разряд-5)*10.

31. *Кафе* со списками *блюд* (с ценой) и очередью *клиентов* (деньги, список заказа). Состояние клиента:
1) *занять очередь* – стать последним; при этом список заказанных блюд пустой;
2) *стоять в очереди* – определить список заказа, если не первый и не последний в очереди;

3) *заплатить* – если первый в очереди; если денег не хватило, отказаться от самого дорогого заказанного блюда.

32. *Поликлиника* со списками врачей (список больных, специализация: терапевт, хирург, невропатолог) и *больных* (лечащий врач, диагноз). Состояние больного:
1) *диагностика* – у терапевта определяется диагноз;
2) *лечение* – определяется лечащий врач;
3) *выписка* – удаление из списка.

33. *Сады* со списками *садовников* (список деревьев, возможность – максимальное число деревьев, за которыми может ухаживать) и *деревьев* (садовник, возраст, число плодов). Состояние дерева:
1) *посадка* – начальные данные;
2) *цветение* – увеличение возраста на 1 год;
3) *созревание* – определение числа плодов;
4) *урожай* – сбор урожая.

34. *Маркетинг* со списками *магазинов* (списки имеющихся и дефицитных товаров), *товаров* (количество, список магазинов с товаром) и *продаж* (магазин, товар, количество). Состояние товара:
1) *завоз товара в магазин* – увеличение количества;
2) *продажа* – уменьшение количества; если нехватка, переход в дефицитные.

35. *Авиамаршруты* со списками *авиарейсов* (пункт вылета, пункт посадки, список пассажиров, вместимость пассажиров) и *пассажиров* (пункт вылета, пункт назначения, маршрут). Состояние авиарейса:
посадка – садятся пассажиры, чей пункт вылета совпадает с пунктом вылета авиарейса и авиарейс входит в его маршрут;
полет – список пассажиров не меняется;
прилет – пассажиры выходят.
Состояния пассажира:
1) *покупка билетов* – определяются пункты вылета и назначения, а также маршрут с наименьшим числом пересадок;
2) *полет*;
3) *высадка* – изменение пункта вылета на следующий пункт маршрута.

36. *Комплектация изделий* со списками *изделий* разного типа:
деталь – не содержит других изделий;
агрегат – содержит другие изделия.
Состояние комплектации:
1) *сборка* – включение в изделие других комплектующих изделий;
2) *приемка* – составление списка всех включенных деталей (как непосредственно включенных, так и включенных в агрегаты) с их суммарным количеством по каждой детали.

37. *Троллейбусные движение* со списками *маршрутов* (циклический упорядоченный список остановок маршрута и список троллейбусов маршрута), *троллейбусов* (маршрут, остановка следования) и *остановок* (тип: *обычная, по требованию, конечная*). Состояние троллейбуса:
1) *выход на линию* – остановка следования – одна из конечных;
2) *движение* – к следующей остановке по маршруту, которая становится остановкой следования.

38. *Первенство факультетов* со списками *соревнований* (список факультетов, списки игр проведенных и предстоящих с командами других факультетов), *факультетов* (список соревнований), *игр* (2 факультета, очки каждому: 2 – выигрыш, 1 – ничья, 0 – проигрыш). Состояние игры:
1) *не проводилась* – очки не определены;
2) *закончилась* – определяется победитель.

39. *Зрелище* со списками *театров* (репертуар: список пар дата-пьеса), *пьес* (список театров, ставящих пьесу), *зрителей* (список желаемого просмотра, план: список пар посещения дата-пьеса). Состояние зрителя:
    1) *желание* – определение желаемого пьес просмотра;
    2) *планирование* – максимальный план, при котором зритель посещает максимальное число постановок, но в 1 день не может смотреть 2 пьесы;
    3) *замена* – изменение плана, когда пьеса снята из репертуара на запланированный день.

40. *Дешевые обеды* со списками *столовых* (список клиентов столовой, меню: список пар блюдо-цена), *клиентов* (список желаемых блюд; столовая) и *блюд* (список столовых, имеющих блюдо). Состояние клиента:
    1) *заказ* – определение списка желаемых блюд;
    2) *выбор* – столовой наиболее дешевого обеда;
    3) *обед.*

41. *Экономичные покупки* со списками *магазинов* (прейскурант: список пар товар-цена, список покупок: пары покупатель-товар), *товаров* (список магазинов, имеющих товар) и покупателей (список товаров для покупки и список покупок: магазин-товар). Состояние покупателя:
    1) *заказ* – определение списка товаров;
    2) *выбор магазинов* – с наиболее низкими ценами;
    3) *покупки* – поход по магазинам.

42. *Телевидение* со списками *ТВ-каналов* (список зрителей канала, список передач: пары время – передача), *передач* (список ТВ-каналов, ведущих передачу), *зрителей* (список желаемых передач, план просмотра: пары передача-ТВ-канал). Состояние зрителя:
    1) *заявка* – составление списка желаемых передач);
    2) *планирование* – составление максимального плана;
    3) *просмотр.*

43. *Семестр* со списками *предметов* (список студентов, слушающих предмет; план лекций: список пар преподаватель-номер семестра), *преподавателей* (список проводимых преподавателем предметов: пары предмет-номер семестра) и *студентов* (семестр, список слушаемых студентом предметов, список слушаемых студентом преподавателей). Состояние студента:
    1) *начало семестра* – составление списка предметов для слушания;
    2) *занятия* – составление списка слушаемых преподавателей.

44. *Труд* со списками *предприятий* (директор, список отделов), *отделов* (менеджер, список работников) и *работников* (стаж, статус: обычный, менеджер отдела, директор). Состояние отделов:
    1) *слияние двух отделов предприятия* – менеджер первого становится менеджером нового, а менеджер второго увольняется;
    2) *разделение отдела* – (выделение из отдела нового отдела с менеджером: обычным работником отдела с наибольшим стажем, и списком работников, идущих в списке отдела после менеджера).

    Состояние работника:
    1) *прием на работу* – обычный работник со стажем 0;
    2) *стаж* – увеличение на 1;
    3) *повышение* – статуса;
    4) *увольнение.*

45. *Поездки в автобусе* со списками *остановок, автобусов* (упорядоченный список остановок автобуса, остановка следования, список пассажиров) и *пассажиров* (*остановки* посадки, высадки, список подходящих автобусов). Состояния пассажира:
    1) *готовится к поездке* – определяются остановки посадки, высадки и список подходящих автобусов;
    2) *ждет* – на остановке;
    3) *посадка* – в автобус;
    4) *поездка;*
    5) *высадка* – на остановке высадки.

46. *Конкурсный прием* со списками *экзаменов, факультетов* (количество свободных мест, занятых мест, список абитуриентов факультета, список экзаменов факультета) и абитуриентов (список заявленных факультетов, список экзаменов: пары экзамен-оценка, список факультетов, куда прошел, выбранный факультет). Состояния абитуриента:
    1) *подача заявления* – определение заявленных факультетов;
    2) *экзамены* – изменение оценок;
    3) *конкурс* – определение списка абитуриентов на каждый факультет, набравших наибольшую сумму балов по числу свободных мест:
    4) *выбор факультета* – коррекция числа свободных и занятых мест на выбранном факультете и списков для других факультетов.

47. *Трудовой набор* со списками *профессий, предприятий* (список свободных профессий на предприятии, список приглашенных претендентов на предприятие, список принятых работников) и *претендентов* (список профессий претендента, список заявленных предприятий, список пригласивших предприятий, выбранное предприятие). Состояние предприятия:
    1) *потребность* – определение списка потребных профессий;
    2) *приглашение* – определение списка приглашенных претендентов;
    3) *зачисление* – определение списка принятых работников.

    Состояния претендента:
    1) *поиск работы* – определение списка профессий;
    2) *заявление* – определение списка предприятий;
    3) *выбор* – из списка пригласивших предприятий.

48. *Выбор руководителя* со списками *руководитель* (кафедра, список руководимых студентов, список студентов руководителя, список студентов, заявивших руководителя, количество свободных мест для руководителя) и *студентов* (курс, упорядоченный список заявленных руководителей, выбранный руководитель). Состояния руководителя:
    1) *объявление набора* – определение количества свободных мест;
    2) *выбор студентов* – из списка каждого студента, заявившего руководителя первым;
    3) *конец набора* – удаление руководителя из списков студентов, заявивших руководителя, но не выбранных им.

    Состояния студента:
    1) *выбор не нужен* – курс не равен 3;
    2) *заявление руководителя* – составление упорядоченного списка заявленных руководителей;
    3) *выбор руководителем.*

49. *Маршруты трамваев* со списками *маршрутов* (список трамваев маршрута, упорядоченный циклический список остановок маршрута, количество трамваев маршрута), *остановок* (тип: обычная, по требованию), *трамваев* (список маршрутов). Состояние маршрута:
    1) *остановки* – определение упорядоченного циклического списка остановок;
    2) *трамваи* – определение трамваев маршрута.

50. *Лесничества* со списками *лесников* (список участков лесника, ограничение возрастов вырубки и подсадки), *участков леса* (лесник, список деревьев: пары возраст-количество, ограничения количества деревьев), *деревьев* (пары возраст-количество). Состояния лесника:
    1) *план* – составление для каждого участка списка деревьев для вырубки и списка деревьев для подсадки с проверкой ограничений;
    2) *вырубка* – изменение деревьев на участке;
    3) *посадка* – изменение деревьев на участке.

51. *Лабораторный практикум* со списками *лабораторий* (список студентов лаборатории, план работ: пары преподаватель – номер семестра), *преподавателей* (список лабораторных работ: пары лаборатория – номер семестра) и *студентов* (семестр, список заданных студенту лабораторных работ, список выполненных лабораторных работ). Состояния преподавателя:
    1) *планирование* – определение плана работ;
    2) занятия – определение списков студентов для каждой работы.

    Состояния студента:
    1) *выдача работ* – определение списка работ для студента;

2) *выполнение* – определение списка выполненных работ.

52. *Программистский проект* со списками *программ* (разряд программы, программист) и *программистов* (категория программиста, резюме: список выполненных программ как пар разряд программ - количество программ). Состояния программы:
    1) *задание* – определение программиста;
    2) *выполнение;*
    3) *сдача* – корректировка резюме программиста.

53. *Студенческий обед* со списками блюд (объем, цена) и студентов (кошелек: деньги на обед, аппетит: желаемый суммарный объем блюд, список выбранных студентом блюд). Состояния студента:
    1) *перевод* – пополнение кошелька;
    2) *желание* – определение аппетита;
    3) *выбор* – определение списка блюд наиболее близких по суммарному объему;
    4) *касса* – расчет, если денег в кошельке хватает;
    5) *отказ* – замена наиболее дорогого блюда на более дешевое с близким объемом.

54. *Конкурсы в Вузы* со списками *Вузов* (список факультетов Вуза, список абитуриентов Вуза), *факультетов* (Вуз, список экзаменов факультета, проходной балл, список абитуриентов факультета, список зачисленных), *абитуриентов* (список заявленных Вузов-факультетов, список сданных экзаменов с оценками) и список *экзаменов* (Вуз, факультет). Состояния абитуриента:
    1) *заявления* – определение списка Вузов-факультетов;
    2) *экзамены* – определение оценок по экзаменам;
    3) *конкурс* – определение Вузов-факультетов, на которые набран проходной балл;
       зачисление – выбор Вуза-факультета.

55. *Очереди в магазинах* со списками *товаров, магазинов* (список товаров магазина: товар-количество-цена, очередь покупателей: упорядоченный список) и *покупателей* (список необходимых товаров: товар-количество, кошелек: количество денег). Состояние покупателя:
    1) *желание* – определение необходимых товаров;
    2) *планирование* – определение списка товар-магазин;
    3) *очередь* – занять очереди в магазинах;
    4) *покупка* – в том магазине, где покупатель первый.

56. *Отдых* со списками *туристов* (список желаемых мест с примерными сроками, путевка), *агентств* (список путевок: пар путевка – количество, список туристов агентства) и *путевок* (агентство, место отдыха, сроки, количество). Состояния туриста:
    1) *желание* – определение срока и списка желаемых мест;
    2) *поиск* – агентства с путевкой наиболее подходящей (сроки наименее отличаются по сумме отклонений начала и конца);
    3) *покупка* – изменение списка туристов и путевок агентства.

4. Методика проектирования схемы базы данных

Проектирование таблиц реляционной БД начинается с анализа данных и выделения сущностей динамической ситуации. Для каждой сущности выделяются данные, характеризующие ее (атрибуты), и связи с другими сущностями динамической ситуации. Все сущности необходимо разбить на группы так, чтобы:

1) все сущности одной группы обладали одинаковым набором атрибутов;
2) каждый атрибут кроме атрибутов, идентифицирующих сущность, относился только к одной группе.

В этом случае на первом этапе разработки схемы для каждой группы мы спроектируем таблицу реляционной БД со столбцами, отвечающими атрибутам этой группы.

Рассмотрим в качестве примера следующую динамическую ситуацию: *Порт перегрузки определяется динамически меняющимися списками грузов (на складах порта и на кораблях груз определяется именем, портом назначения, количеством) и кораблей* (*имя*, *список грузов*, *свободная грузовместимость: сколько можно еще погрузить, состояние погрузки или разгрузки*). *В состоянии разгрузки корабля разгружаются грузы* (*из списка корабля в список складов порта*),

порт назначения которых не соответствует порту назначения корабля. В состоянии погрузки – погружаются грузы (*из списка складов порта в список корабля*)*, порт назначения которых соответствует порту назначения корабля.*

Анализ данных показывает, что атрибут *имя* идентифицирует имя сущности *груз* или *корабль*, а потому может относиться как к группе сущностей *груз*, та и к группе сущностей *корабль*. Далее, значение атрибута *количество* груза зависит не только от имени груза, но и от *порта назначения*, а также списка грузов: того или иного корабля или складов порта. Поэтому на первом этапе проектирования схемы БД выделяются 2 группы сущностей: *грузы* (с атрибутом *имя*) и *корабли* (с атрибутами *имя*, *свободная грузовместимость* и *состояние*). Кроме того, выделяется сущность *порт* (порт назначения корабля, порт назначения груза, порт перегрузки) с атрибутом *имя* и атрибутом *перегрузка*, отмечающим среди прочих портов порт перегрузки. Для этих 3 выделенных сущностей и требуется спроектировать таблицы БД *Порты, Грузы, Корабли.* Атрибуты *количество* груза и *порт назначения* груза характеризует связь таблицы *Грузы* с таблицей *Корабли*, а также являются дополнительными характеристиками грузов на складах порта перегрузки.

На втором этапе проектирования схемы БД необходимо реализовать связи таблиц. При этом для каждой связи мы можем выделить дополнительные атрибуты таблицы: так в примере для связи таблиц Корабли и Грузы мы можем в таблице Грузы дополнительно выделить атрибуты *груз* и *количество*. Но в этом случае вместо одной записи для корабля мы должны будем ввести столько записей, сколько грузов везет корабль и повторять данные для остальных атрибутов корабля. Это не рационально. Другой эффективный путь состоит в том, чтобы создать дополнительную таблицу для связи, определив ее атрибутами, идентифицирующими связываемые записи в обеих таблицах. Принято для основных таблиц сущностей вводить числовой объектный идентификатор (атрибут *Id*), уникально характеризующий каждую запись-сущность, а в таблице связи употреблять 2 атрибута, являющихся объектными идентификаторами для обеих связываемых таблиц. Помимо этого, в связующую таблицу мы должны добавить атрибуты характеристик связи, если они имеют место. Так в примере для таблицы *Грузы* мы добавим атрибут *Id_груза*, уникально характеризующий каждую пару (*имя, порт назначения*)*,* для таблицы *Корабли* мы добавим атрибут *Id_корабля*, уникально характеризующий каждый корабль, а для связи этих таблиц мы образуем таблицу *Грузы кораблей*, содержащую атрибуты *Id_груза*, *Id_корабля* и *количество.* Таким образом, следует поступать, если отношение связываемых таблиц типа «многие ко многим». При этом для каждого корабля определяется список грузов, который везет этот корабль, а для каждого груза определяется список кораблей, которые везут этот груз. В случае дополнительных характеристик грузов порта перегрузки у нас всего-навсего 1 порт перегрузки и поэтому не надо создавать связь «многие ко многим», а достаточно ввести таблицу связи *Грузы складов порта*, содержащую атрибуты *Id_груза* и *количество*, связав ее с таблицей *Грузы* отношением «один к одному». При этом для каждого груза, определяемого объектным идентификатором *Id_груза*, по такому же идентификатору в связанной с ней таблицей *Грузы* определяется имена грузов, а их порты назначения и количество определяются по первой таблице.

Анализ возможных состояний корабля показывает, что 2 состояний (разгрузка и погрузка), указанных в задании мало: перед разгрузкой необходимо состояние, когда грузы определяются на корабле, плывущем в порт перегрузки, а также состояние, когда после погрузки корабль уходит из порта перегрузки. Поэтому введем 4 состояния корабля:

1) *приплытие* (в порт перегрузки) – определяется список грузов корабля и порт назначения корабля (в этом состоянии корабль вводится);
2) *разгрузка* – грузы, имеющие порт назначения либо порт перегрузки, либо другой, чем порт назначения корабля, разгружаются на склады порта перегрузки;
3) *погрузка* – грузы на складах порта перегрузки, имеющие порт назначения, отличный от порта перегрузки и одинаковый с портом назначения корабля, погружаются на корабль;
4) *отплытие* (из порта перегрузки) – корабль удаляется.

Теперь для быстрой выборки данных необходимо ввести индексы. С этой целью объектные идентификаторы каждой из таблиц следует объявить первичными ключами (*primary key*) этих таблиц, а каждый объектный идентификатор каждой связующей таблицы следует объявить внешним ключом (foreign key).

Окончательный вид спроектированных реляционных таблиц и спроектированная схема данных приведены ниже.

Ports – порты

Имя поля	Тип	Ключ	Комментарий
Id	Number(6)	Primary key	Идентификатор порта – первичный ключ
Name	Varchar2(20)	Unique key	Имя порта – уникальный ключ
Overload_port	Number(6)	Check (Sum(Overload_port)= 1)	1 – для порта перегрузки 0 – для остальных портов

Cargoes – грузы

Имя поля	Тип	Ключ	Комментарий
Id	Number(6)	Primary key	Идентификатор груза – первичный ключ
Name	Varchar2(20)	Unique key	Имя груза – уникальный ключ

Ships – корабли

Имя поля	Тип	Ключ	Комментарий
Id	Number(6)	Primary key	Идентификатор корабля – первичный ключ
Name	Varchar2(20)	Unique key	Имя корабля – уникальный ключ
Id_port_dest	Number(6)	Foreign key	Порт назначения – внешний ключ (связь с таблицей Ports)
Free_tonnage	Number(5)		Свободная грузовместимость
Situation	Number(1)		Состояние: 1 – приплытие; 2 – разгрузка; 3 – погрузка; 4 – отплытие

Ships_ Cargoes – грузы кораблей

Имя поля	Тип	Ключ	Комментарий
Id_ship	Number(6)	Foreign key	Идентификатор корабля – внешний ключ (связь с таблицей Ships)
Id_cargo	Number(6)	Foreign key	Идентификатор груза – внешний ключ (связь с таблицей Cargoes)
Quantity	Number(4)	Check(Quntity≥0)	Количество груза
Id_port_dest	Number(6)	Foreign key	Порт назначения – внешний ключ (связь с таблицей Ports)
Id_ship, Id_port_dest, Id_cargo		Primary key	Первичный ключ

Cargoes_Port – грузы на складах порта перегрузки

Имя поля	Тип	Ключ	Комментарий
Id	Number(6)	Primary key Foreign key	Идентификатор груза – первичный ключ, внешний ключ (связь с таблицей Cargoes)
Quantity	Number(6)		Количество груза
Id_port_dest	Number(6)	Foreign key	Порт назначения – внешний ключ (связь с таблицей Ports)

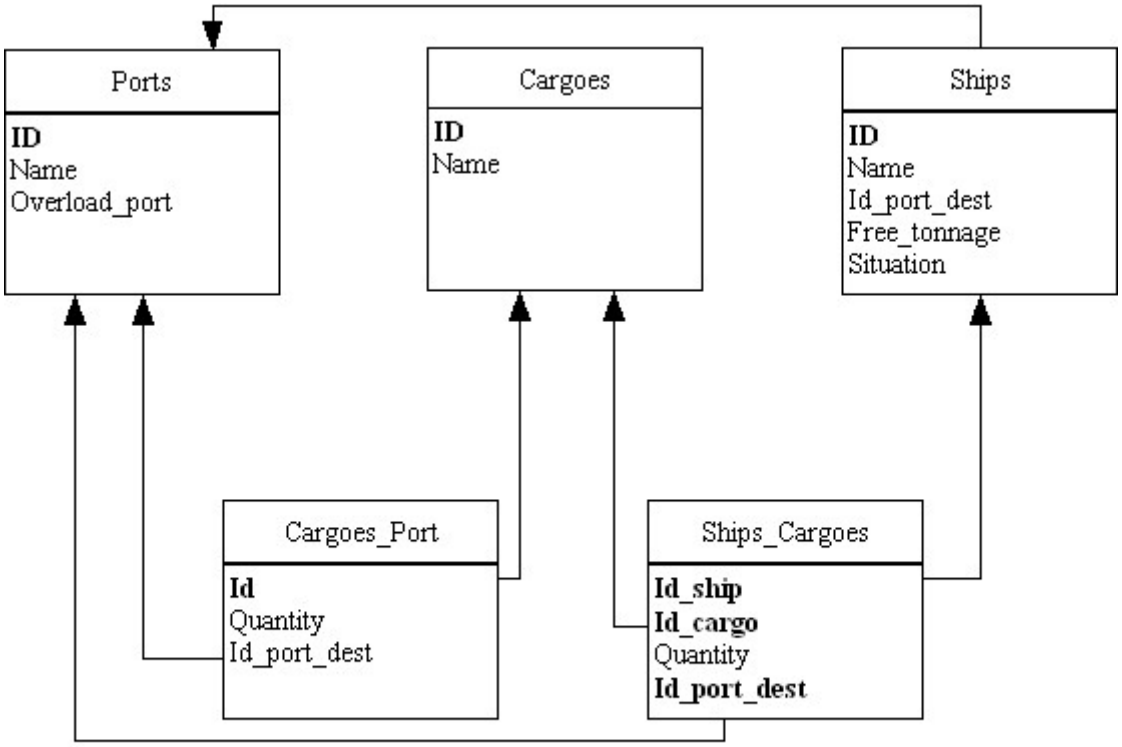

Схема данных

Теперь можно приступить к созданию таблиц и ключей. Рационально это делать при помощи программы PL/SQL Developer. Отметим, что создание ограничения Check в таблице Ports связано с тем, что лишь один порт должен быть портом перегрузки. Поэтому при добавлении первой записи в эту таблицу следует отнести ее к порту перегрузки (значение поля Overload_port установить в 1), а если порт перегрузки нужно изменить, то надо сбросить значение Overload_port для прежнего порта и установить в 1 для нового.

5. Методика разработки пакета процедур и функций работы с БД

Пакет состоит из интерфейсной части и тела пакета. В интерфейсе пакета прописываются спецификации всех функций и процедур, которые должны быть видны для пользователя, а также типы, глобальные переменные и константы. Поэтому в этой части не должно быть процедур и функций, связанных с объектными идентификаторами, так как для пользователя они не должны быть видны. Кроме того, следует учитывать, что должна быть реакция (например, сообщение) на некорректные данные процедур и функций. В случае дальнейшего использования пакета в других системах программирования (а мы предполагаем именно такое использование при создании интерфейса) необходимо, чтобы реакцию на некорректные данные осуществляла такая другая система. Поэтому функция или процедура должна передавать такую информацию. Проще всего это организовать при помощи возвращения функцией кода ошибки (скажем, некоторое отрицательное число). В спецификации каждой процедуры или функции должен быть определен комментарий, указывающий смысл функции (процедуры), смысл ее параметров и возвращаемого функцией значения.

Разработку интерфейсной части пакета начнем с функций, определяющих создание, удаление и коррекцию данных таблиц Ports, Cargoes, Ships:

1. Function AddPort (NamePort Ports.Name%Type)) return number – добавить в таблицу Ports новый порт с именем NamePort. При добавлении первого порта значение Overload_port устанавливает в 1. Возвращает:
   0 – благополучное добавление порта;
   -1 – такой порт уже существует.

2. Function DelPort (NamePort Ports.Name%Type) return number – удалить из таблицы Ports порт с именем NamePort. При удалении порта перегрузки делает портом перегрузки первый порт таблицы, если она не пустая. Возвращает:
   0 – благополучное удаление порта;
   1 – изменился порт перегрузки;
   -2 – нет порта с указанным именем.

3. Function ChangePortName (OldName Ports.Name%Type, NewName Ports.Name%Type) return number – изменить имя порта OldName на имя NewName. Возвращает:
   0 – благополучное изменение имени;
   -1 – порт с именем NewName уже существует;
   -2 – нет порта с указанным именем OldName.

4. Function ChangeOverload (NewPort Ports.Name%Type) return number – изменяет порт перегрузки на порт с именем NewPort. Возвращает:
   0 – благополучное изменение порта перегрузки;
   -2 – нет порта с указанным именем newPort.

5. Function AddCargo (NameCargo Cargoes.Name%Type) return number – добавляет в таблицу Cargoes новый груз с именем NameCargo. Возвращает:
   0 – благополучное добавление груза;
   -3 – такой груз уже существует.

6. Function DelCargo (NameCargo Cargoes.Name%Type) return number – удалить из таблицы Cargoes груз с именем NameCargo. Возвращает:
   0 – благополучное удаление груза;
   -4 – нет груза с указанным именем.

7. Function ChangeCargoName (OldName Cargoes.Name%Type, NewName Cargoes.Name%Type) return number – изменить имя груза OldName на имя NewName. Возвращает:
   0 – благополучное изменение имени;
   -3 – груз с именем NewName уже существует;
   -4 – нет груза с указанным именем OldName.

8. Function AddShip (NameShip Ships.Name%Type, DestPort Ships.Name%Type) return number – добавляет в таблицу Ships новый корабль с именем NameShip и портом назначения DestPort. Состояние корабля (Situation) устанавливает в 1. Возвращает:
   0 – благополучное добавление корабля;
   -5 – такой корабль уже существует;
   -2 – нет порта назначения с указанным именем.

9. Function DelShip (NameShip Ships.Name%Type) return number – удалить из таблицы Ships корабль с именем NameShip. Возвращает:
   0 – благополучное удаление корабля;
   -6 – нет корабля с указанным именем.

10. Function ChangeShipName (OldName Ships.Name%Type, NewName Ships.Name%Type) return number – изменить имя корабля OldName на имя NewName. Возвращает:
   0 – благополучное изменение имени;
   -5 – корабль с именем NewName уже существует;
   -6 – нет корабля с указанным именем OldName.

11. Function ChangeDestPort (NameShip Ships.Name%Type, DestPort Ports.Name%Type) return number – изменяет порт назначения корабля с именем NameShip на DestPort. Возвращает:
   0 – благополучное изменение порта назначения;
   -6 – такой корабль не существует;
   -2 – нет порта назначения с указанным именем.

12. Function MoveShip (NameShip Ships.Name%Type) return number – изменяет состояние корабля на следующее (Situation увеличивает на 1, изменение состояния *отплытие* удаляет корабль). Возвращает:
   0 – благополучное изменение состояния;
   -6 – нет корабля с указанным именем;
   -7 – недопустимое значение состояния (не находится в интервале от 1 до 4).

   Следующие функции изменяют значения связующих таблиц Ships_Cargoes и Cargoes_Port:

13. Function AddShipCargo (NameShip Ships.Name%Type, NameCargo Cargoes.Name%Type, Quant Ships_Cargoes.Quantity%Type, DestCargo Ports.Name%Type) return number – добавление груза NameCargo в количестве Quant с портом назначения DestCargo в список корабля NameShip. Возвращает:
   0 – благополучное добавление груза;
   -4 – нет груза с указанным именем;
   -6 – нет корабля с указанным именем;
   -2 – нет порта назначения с указанным именем.

14. Function DelShipCargo (NameShip Ships.Name%Type, NameCargo Cargoes.Name%Type) return number – удаляет груз NameCargo из списка корабля NameShip. Возвращает:
   0 – благополучное удаление груза;
   -4 – нет груза с указанным именем;
   -6 – нет корабля с указанным именем.

15. Function ChangeShipCargo (NameShip Ships.Name%Type, NameCargo Cargoes.Name%Type, Quant Ships_Cargoes.Name%Type, DestCargo Ports.Name%Type) return number – изменяет количество груза NameCargo из списка корабля NameShip на Quant и порт назначения груза на DestPort. Возвращает:
   0 – благополучное изменение количества груза;
   -2 – нет порта назначения с указанным именем;
   -4 – нет груза с указанным именем;
   -6 – нет корабля с указанным именем.

16. Function UnloadingShip (NameShip Ships.Name%Type) return number – разгрузка на корабле NameShip грузов, порт назначения которых не соответствует порту назначения корабля. Возвращает:
   0 – благополучная разгрузка;

-6 – нет корабля с указанным именем.

17. Function LoadingShip (NameShip Ships.Name%Type) return number – погрузка корабля NameShip в пределах его свободной грузовместимости грузами порта перегрузки, порт назначения которых соответствует порту назначения корабля. Возвращает:
    0 – благополучная погрузка;
    -6 – нет корабля с указанным именем.

Перейдем теперь к проектированию спецификаций дополнительных процедур и функций для тела пакета. Во-первых, необходимо написать ряд функций поиска, которые по имени порта, груза или корабля находили объектные идентификаторы этих записей в таблицах Ports, Cargoes и Ships:

18. Function IdPort (NamePort Ports.Name%Type) return number – возвращает идентификатор порта по его имени NamePort. Если такого имени нет, то возвращает -2.

19. Function IdCargo (NameCargo Cargoes.Name%Type) return number – возвращает идентификатор груза по его имени NameCargo. Если такого имени нет, то возвращает -4.

20. Function IdShip (NameShip Ships.Name%Type) return number – возвращает идентификатор корабля по его имени NameShip. Если такого имени нет, то возвращает -6.

Во-вторых, все

21. Function DelIdPort (IdPort Ships.Id%Type) return number – удалить из таблицы Ports порт с именем IdPort. При удалении порта перегрузки делает портом перегрузки первый порт таблицы, если она не пустая. Возвращает:
    0 – благополучное удаление порта;
    1 – изменился порт перегрузки;
    -12 – нет порта с указанным идентификатором.

22. Function ChangeIdOverload (NewIdPort Ports.Id%Type) return number – изменяет порт перегрузки на порт с идентификатором NewIdPort. Возвращает:
    0 – благополучное изменение порта перегрузки;
    -12 – нет порта с указанным идентификатором.

23. Function DelIdCargo (IdCargo Cargoes.Id%Type) return number – удалить из таблицы Cargoes груз с идентификатором IdCargo. Возвращает:
    0 – благополучное удаление груза;
    -14 – нет груза с указанным идентификатором.

24. Function ChangeIdCargoName (IdCargo Cargoes.Id%Type, NewName Cargoes.Name%Type) return number – для груза с идентификатором IdCargo изменить имя на NewName. Возвращает:
    0 – благополучное изменение имени;
    -3 – груз с именем NewName уже существует;
    -14 – нет груза с указанным идентификатором IdCargo.

25. Function DelIdShip (IdShip Ships.Id%Type) return number – удалить из таблицы Ships корабль с идентификатором IdShip. Возвращает:
    0 – благополучное удаление корабля;
    -16 – нет корабля с указанным идентификатором.

26. Function ChangeIdShipName (IdShip Ships.Id%Type, NewName Ships.Name%Type) return number – изменить имя корабля OldName на имя NewName. Возвращает:
    0 – благополучное изменение имени;
    -5 – корабль с именем NewName уже существует;
    -16 – нет корабля с указанным идентификатором IdShip.

27. Function ChangeDestIdPort (IdShip Ships.Id%Type, DestIdPort Ports.Id%Type) return number – изменяет порт назначения корабля с идентификатором IdShip на DestIdPort. Возвращает:
    0 – благополучное изменение порта назначения;
    -16 – такой корабль не существует;
    -12 – нет порта назначения с указанным идентификатором.

28. Function MoveIdShip (IdShip Ships.Id%Type) return number – изменяет состояние корабля на следующее (Situation увеличивает на 1, изменение состояния *отплытие* удаляет корабль). Возвращает:
    0 – благополучное изменение состояния;
    -16 – нет корабля с указанным идентификатором;
    -7 – недопустимое значение состояния (не находится в интервале от 1 до 4).

29. Function AddIdShipCargo (IdShip Ships.Id%Type, IdCargo Cargoes.Id%Type, Quant Ships_Cargoes.Quantity%Type, DestIdCargo Ports.Id%Type) return number – добавление груза с идентификатором IdCargo в количестве Quant с портом назначения определяемым идентификатором DestIdCargo в список корабля с идентификатором IdShip. Возвращает:
    0 – благополучное добавление груза;
    -14 – нет груза с указанным идентификатором;
    -16 – нет корабля с указанным идентификатором;
    -12 – нет порта назначения с указанным идентификатором.

30. Function DelIdShipCargo (IdShip Ships.Id%Type, IdCargo Cargoes.Id%Type) return number – удаляет груз с идентификатором IdCargo из списка корабля с идентификатором IdShip. Возвращает:
    0 – благополучное удаление груза;
    -14 – нет груза с указанным идентификатором;
    -16 – нет корабля с указанным идентификатором.

31. Function ChangeIdShipCargo (IdShip Ships.Id%Type, IdCargo Cargoes.Id%Type, Quant Ships_Cargoes.Quantity%Type, DestIdCargo Ports.Id%Type) return number – изменяет количество груза с идентификатором IdCargo из списка корабля с идентификатором IdShip на Quant и идентификатор порта назначения груза на DestIdPort. Возвращает:
    0 – благополучное изменение количества груза;
    -12 – нет порта назначения с указанным идентификатором;
    -14 – нет груза с указанным идентификатором;
    -16 – нет корабля с указанным идентификатором.

32. Function UnloadingIdShip (IdShip Ships.Id%Type) return number – разгрузка на корабле с идентификатором IdShip грузов, порт назначения которых не соответствует порту назначения корабля. Возвращает:
    0 – благополучная разгрузка;
    -16 – нет корабля с указанным идентификатором.

33. Function LoadingIdShip (IdShip Ships.Id%Type) return number – погрузка корабля NameShip в пределах его свободной грузовместимости грузами порта перегрузки, порт назначения которых соответствует порту назначения корабля. Возвращает:
    0 – благополучная погрузка;
    -16 – нет корабля с указанным идентификатором.

Для отображения данных в системах программирования построения интерфейса Builder C++ потребуются курсоры для каждого табличного отображения. С этой целью мы вводим ряд функций, которые создают и открывают курсор. В интерфейсной части пакета для этого нужно объявить ссылочный тип курсора, например

Type refCursor **is ref Cursor**;

а в теле пакета написать соответствующие функции. Так в рассматриваемом примере потребуются следующие описания функций:

34. **Function** Status (Situat Ships.Situation**%Type**) **returns Varchar2(10) is**

```
    ------------------------------------------------------------------------
    -- возвращает по номеру состояния корабля строку состояния --
    ------------------------------------------------------------------------
    Begin
        If    Situat=1 then return "Приплытие";
        Elsif Situat=2 then return "Разгрузка";
```

```
        Elsif Situat=3 then return "Погрузка";
        Elsif Situat=4 then return "Отплытие";
End Status
```

35.
```
Function Ships (limSituation Ships.Situation%Type, limDestPort Ports.Id%Type)
        returns refCursor is
-------------------------------------------------------------------------------------------------------------
-- список кораблей с их состоянием  и портом назначения;                                                   --
-- limSituation – ограничение кораблей по состоянию (0 – без ограничений);                                 --
-- limDestPort – ограничение кораблей по порту назначения (0 – без ограничений)                            --
-------------------------------------------------------------------------------------------------------------
        rShips  refCursor:
Begin
        Open rShips for
                Select s.Id, s.Name, Status (s.Situation), p.Name
                        From Ships s, Ports p
                        Where s.Id_port_dest = p.Id and
                                s.Situation=0 or s.Situation=limSituation and
                                 Id_port_dest=0 or Id_port_dest=limDestPort;
        Return rShips;
End Ships;
```

36.
```
Function Ports (limPort Number(1), limShip Ships.Id%Type, limCargo Number(1))
        returns refCursor is
-------------------------------------------------------------------------------------------------------------
-- список портов;                                                                                          --
-- limPort – ограничение на порт (0 – без ограничений; 1 – кроме порта перегрузки);                        --
-- limShip – ограничение на корабль (0 – без огр.; >0 – порты назн. грузов корабля);                       --
-- limCargo – огранич.на порты назнач. грузов порта перегрузки  (0 – без огранич.)                         --
-------------------------------------------------------------------------------------------------------------
        rPorts refCursor;
Begin
        Open rPorts for
                Select p.Id, p.Name From Ports p, Ships s, Ships_Cargoes sc, Cargoes_Port cp
                Where (limPort=0 or Overload_port <> 1) and
  (limShip=0 or (limShip=s.IdShip and sc.Id_ship=s.IdShip and p.Id=sc.Id_port_dest)) and
                        (limCargo=0 or p.Id=cp.Id_port_dest);
        Return rPorts;
End Ports;
```

37.
```
Function Cargoes returns refCursor is
-----------------------
-- список грузов  --
-----------------------
        rCargoes refCursor;
Begin
        Open rCargoes for Select Id, Name From Cargoes;

        Return rCargoes;
End Cargoes;
```

38.
```
Function CargoesShips (IdShip Ships.Id%Type, limPort Ports.Id%Type)
        returns refCursor is
-------------------------------------------------------------------------------------------------------------
-- список грузов корабля (IdShip=0 – для всех кораблей);                                                   --
-- limPort – ограничение на порт (0 – без ограничений; >0 – порт назначений грузов);                       --
-------------------------------------------------------------------------------------------------------------
        rCargoes refCursor;
Begin
```

```
        Open rCargoes for
            Select sc.Id_ship, sc.Id_cargo, c.Name, sc.Quantity, p.Name
            From Ships_Cargoes sc, Cargoes c, Ports p
            Where (IdShip=0 or sc.Id_ship=limShip) and c.Id=sc.Id_cargo and
                (limPort=0 or p.Id=sc.Id_port_dest );
        Return rCargoes;
    End CargoesShips;
```

39.
```
    Function ShipsCargoes (IdCargo Cargoes.Id%Type)
        returns refCursor is
    ------------------------------------------------------------------------------
    -- список кораблей везущих груз (IdCargo=0 – для всех грузов)  --
    ------------------------------------------------------------------------------
        rCargoes refCursor;
    Begin
        Open rCargoes for
            Select sc.Id_ship, sc.Id_cargo, s.Name, sc.Quantity, p.Name
            From Ships_Cargoes sc, Ships s, Ports p
            Where (IdCargo=0 or sc.Id_cargo=IdCargo) and
                s.Id=sc.Id_ship and p.Id=sc.Id_port_dest ;
        Return rCargoes;
    End ShipsCargoes;
```

40.
```
    Function CargoesPort (limCargo Cargoes.Id%Type, limPort Ports.Id%Type)
        returns refCursor is
    ------------------------------------------------------------------------------------------------------------
    -- список грузов на складах порта перегрузки;                                                            --
    -- limCargo – ограничение на груз порта перегрузки (0 – без огран.; 1 – ограничен.);    --
    -- limPort – ограничение на порт (0 – без ограничений; >0 – порт назначений грузов);  --
    ------------------------------------------------------------------------------------------------------------
        rCargoes refCursor;
    Begin
        Open rCargoes for
            Select cp.Id, c.Name, cp.Quantity, p.Name
            From Cargoes_Port cp, Cargoes c, Ports p
            Where (limCargo=0 or cp.Id=limCargo) and c.Id=cp.Id and
                (limPort=0 or p.Id=cp.Id_port_dest);
        Return rCargoes;
    End CargoesPort;
```

6. Методика построения интерфейса «связанных таблиц»

Построение интерфейса в системе Builder C++ основано на отображении специальных запросов в таблицах формы. При этом интерфейс предполагает размещение на форме табличных отображений формы, связанных с основными таблицами проекта в Oracle. Форма должна быть снабжена главным меню для выбора главной таблицы и меню изменения этой таблицы (*добавление, коррекция, удаление*) или соответствующие кнопки. Трехзвенная архитектура связи с БД и отображения таблиц предполагает использование следующих компонент:

1) невизуальная компонента TDataBase – для связи с БД;
2) невизуальная компонента TStoredProc – для связи с хранимыми функциями пакета;
3) невизуальная компонента TDataSource – для связи с источником данных (может быть кроме TStoredProc также TTable и TQuery);
4) невизуальная компонента TDataSet – для связи источника данных с отображением данных;
5) визуальная компонента TDBGrid – для отображения данных в табличном виде (кроме этой компоненты можно использовать компоненты TDBEdit для редактирования, TDBComboBox для отображения выбора из списка).

Эти компоненты следует расположить и настроить именно в этом порядке. Настройку компонент можно выполнять различным образом, но мы рекомендуем следующий путь.

Прежде всего, после размещения на форме компоненты TDataBase необходимо сделать двойной щелчок (левой кнопкой мыши) на этой компоненте. Откроется форма определения основных параметров связи с БД. Нужно указать в поле Name имя компоненты, а в поле DataBaseName – имя, которое нужно будет употреблять в других компонентах для связи. Чаще всего компонента TDataBase одна, и потому для обоих имен можно выбрать одно и тоже имя, совпадающее с именем БД в Oracle. В поле DriverName следует выбрать ORACLE. Затем нужно открыть кнопкой Defaults список значений параметров и указать следующие значения:

SERVER NAME = <имя реальной БД в Oracle> – например, UNIVERSE;

USER NAME = <имя пользователя БД в Oracle> – например, SCOTT;

NET PROTOCOL = TCP – сменить на это имя;

PASSWORD = <пароль пользователя БД в Oracle > – например, TIGER,

а остальные параметры с их значениями удалить. Далее необходимо выключить переключатель Login prompt, иначе при каждом обращении к БД будет идти запрос на вход в БД.

После размещения компоненты TStoredProc необходимо определить в качестве значения ее поля StoredProcName <имя БД>.<имя пакета>.<имя хранимой функции>. Затем следует выбрать свойство Params (щелчком на многоточии) и в открывшемся окне параметров прописать их имена в том же порядке, что и в хранимой функции (и лучше с теми же именами и совместимыми типами C++), добавив к ним параметр с именем Result и типом возвращаемого функцией значения (для хранимых процедур не требуется этот параметр). Затем необходимо определить имя DataBaseName, которое мы выше указали для компоненты TDataBase.

После этого нужно разместить компоненту TDataSource, связав ее с TStoredProc, компоненту TDataSet, связав ее с TDataSource, и, наконец, TDBGid, связав ее с TDataSet. Если пропустить включение компоненты TDataSet, связав TDBGrid непосредственно с TDataSource, то нельзя будет пользоваться полями Fields табличного отображения, необходимых для редактирования полей или выбора значения поля из списка. Обращение к полям в программе того или иного обработчика событий идет при помощи конструкции:

<имя StoredProc> —> <имя DataSet> —> <имя Fields>.

В табличном отображении TDBGrid не нужно отображать объектные идентификаторы. Однако для редактирования данных нужно знать объектные идентификаторы, чтобы не тратить время на их поиск по имени. Поэтому хранимая функция, содержащая запрос обычно включает объектный идентификатор, а для отображения он делается невидимым. Например,

<имя StoredProc> —> Open();

<имя StoredProc> —> FieldByName(“Id”) —> Visble = false;

Считывание объектного идентификатора выбранной строки идет в обработчике события Autoscroll, например, следующим образом:

<имя переменной> = <имя StoredProc> —> FieldByName(“Id”) —> As Integer;

При изменении параметров отображаемого запроса необходимо закрыть хранимую процедуру, изменить значения ее параметров и снова открыть. Например,

<имя StoredProc> —> Close();

<имя StoredProc> —> FieldByName(“имя”) = <новое значение>;

……..

<имя StoredProc> —> Open();

При работе в многопользовательском режиме с одной БД, когда разные пользователи могут обращаться к одним и тем же данным, необходимо употребление операторов транзакции. Компонента TDataSet содержит 3 таких функции, к которым нужно обращаться следующим образом:

<имя TDataSet> —> StartTransaction();

<имя TDataSet> —> RollBack();

<имя TDataSet> —> Commit();

смысл которых аналогичен конструкциям Oracle.

Интерфейс связанных таблиц предполагает, что при перемещении по главной таблице выбираемой строки в других таблицах будет отображаться информация, относящаяся именно к сущности этой строки. Так в вышеописанном примере динамической ситуации *Порт перегрузки* на главной форме необходимо расположить 3 табличных формы:

1) *Корабли* – для отображения информации, связанной с каждым кораблем (*Имя, Состояние, Порт назначения*);

2) *Порты* – для отображения информации, связанной с портами (*Имя, Признак* – порт перегрузки);
3) *Грузы* – для отображения информации связанной с грузами (*Имя, Порт назначения* грузов корабля или складов порта перегрузки, *Количество*).

При выборе в качестве главной – табличной формы *Корабли* (хранимая функция *Ships(0,0)*) в табличной форме *Грузы* должна отображаться информация, относящаяся к выбранному кораблю в главной табличной форме

(хранимая функция *CargoesShips(<идентифик. выбранного в главной таблице корабля>,0)*), и она меняется при перемещении по главной таблице. Меню или кнопки позволяют:

1) *добавить* корабль – в этом случае выбрать его порт назначения в отдельной диалоговой форме, а также его грузы в другой диалоговой форме с определением их количества (в окне редактирования) и порта назначения груза в первой диалоговой форме;
2) *корректировать* корабль (если он в состоянии *Приплытие*) – имя корабля в окне редактирования, порта назначения и грузов (как и в случае добавления корабля);
3) *Изменить состояние* – приводит к изменению состояния на следующее.

При выборе в качестве главной – табличной формы *Порты* (хранимая функция *Ports(0,0,0)*):

1) в табличной форме *Корабли* отображается информация, относящаяся к кораблям, имеющим выбранный порт назначения в главной табличной форме, если он не является портом перегрузки

(хранимая функция *Ships(0, идентификатор выбранного порта назначения)*),

2) а в табличной форме *Грузы* в этом случае отображается информация, относящаяся к грузам кораблей, имеющих тот же порт назначения

(хранимая функция *CargoesShips(0,<идентифик. выбранного в главной таблице порта)*).

Если же выбранный порт является портом перегрузки, то

1) в табличной форме *Корабли* отображается информация по всем кораблям, находящимся в состоянии погрузки (хранимая функция *Ships(3, 0)*),
2) а в табличной форме *Грузы* отображается информация по грузам на складах порта перегрузки

(хранимая функция *CargoesPort(0,0)*).

Меню или кнопки позволяют:

1) *добавить* порт;
2) *изменить* порт перегрузки – в этом случае грузы на складах порта перегрузки удаляются.

При выборе в качестве главной – табличной формы *Грузы* (хранимая функция *Cargoes*):

1) в табличной форме *Корабли* отображается информация, относящаяся к кораблям, везущим выбранный в главной таблице груз

(хранимая функция *ShipsCargoes (<идентификатор выбранного в главной таблице груза>)*),

2) а в табличной форме *Порты* отображается информация по портам, в которые везется этот груз

(хранимая функция *Ports(0,0,<идентификатор выбранного в главной таблице груза>)*).

Меню или кнопки позволяют:

1) *добавить* груз;
2) *изменить* имя груза;
3) *удалить* груз с удалением его в списках кораблей и порта перегрузки.

Следует отметить, что отображение грузов в разных ветвях программы идет при помощи 3 разных хранимых функций Cargoes, ShipsCargoes, CargoesPort. Следует для этих отображений создать разные компоненты TDBGrid, но одних и тех же размеров и расположить их в одном и том же месте главной формы.

7. Методика построения «разворачивающегося» интерфейса

При «разворачивающемся» интерфейсе в отличие от интерфейса «связанных таблиц» при выборе главной табличной формы показывается только она, но с дополнительными пустыми столбцами, характеризующими связи с другими табличными формами. При выборе (щелчком) такого пустого столбца на какой-либо строке табличной формы открывается соответствующая связанная табличная форма с информацией, относящейся к выбранной строке предыдущей

табличной формы, и расположенная левым верхним углом на пересечении выбранных строки и столбца предыдущей табличной формы. При выборе пустого столбца на какой-либо строке последней табличной формы вновь открывается связанная с ней табличная форма с теми же условиями. Выбор какой-либо строки предыдущих табличных форм ведет к закрытию всех развернутых после нее табличных форм.

В рассматриваемом примере динамической ситуации *Порт перегрузки* «разворачивающийся» интерфейс зависит от выбора главной табличной формы и должен отображаться при помощи различных компонент TDBGrid.

При выборе в качестве главной – таблицы *Корабли* интерфейс ее табличного отображения (ТО1) должен содержать поля: *Корабль* (имя), *Состояние, Порт назначения, Грузы.* Последний столбец пустой и при его выборе должно появляться табличное отображение (ТО2) *Грузы корабля* с полями: *Груз, Порт назначения, Корабли* (также везущие этот груз). Последний столбец пустой и при его выборе должно появляться табличное отображение (ТО3) *Корабли груза* с полями: *Корабль, Количество* (груза), *Порт назначения* (груза).

При выборе в качестве главной – таблицы *Грузы* интерфейс ее табличного отображения (ТО4) должен содержать поля: *Груз* (имя), *Корабль* (груза), *Порт* (назначения грузов перегрузки). Последние 2 столбца пустые и при выборе первого из них должно появляться вышеописанное табличное отображение (ТО3) *Корабли груза*, а при выборе второго – должно появляться табличное отображение (ТО5) *Груз перегрузки* с полями: *Груз* (имя), *Количество, Порт назначения.*

При выборе в качестве главной – таблицы *Порты* интерфейс ее табличного отображения (ТО6) должен содержать поля: *Порт* (имя), *Признак перегрузки* (порт перегрузки), *Грузы.* Последние 2 столбца пустые и при выборе первого из них должно появляться вышеописанное табличное отображение (ТО5) *Груз перегрузки*, а при выборе второго – должно появляться табличное отображение (ТО7) *Грузы* с полями: *Груз* (имя), *Корабли* (везущие груз). Последний столбец ТО7 – пустой и при его выборе должно появляться вышеописанное табличное отображение (ТО3) *Корабли груза.*

Отметим теперь, что все указанные табличные отображения компонент TDBGrid следует связать с уже имеющимися хранимыми функциями:

1) ТО1 с *Ships(0,0);*
2) ТО2 с *CargoesShips(<идентификатор выбранного корабля>, 0);*
3) ТО3 с *ShipsCargoes(<идентификатор выбранного груза>);*
4) ТО4 с *Cargoes;*
5) ТО5 с *CargoesPort(<идентификатор выбранного груза>, 0);*
6) ТО6 с *Ports(0,0,0);*
7) ТО7 с *Cargoes.*

Содержание