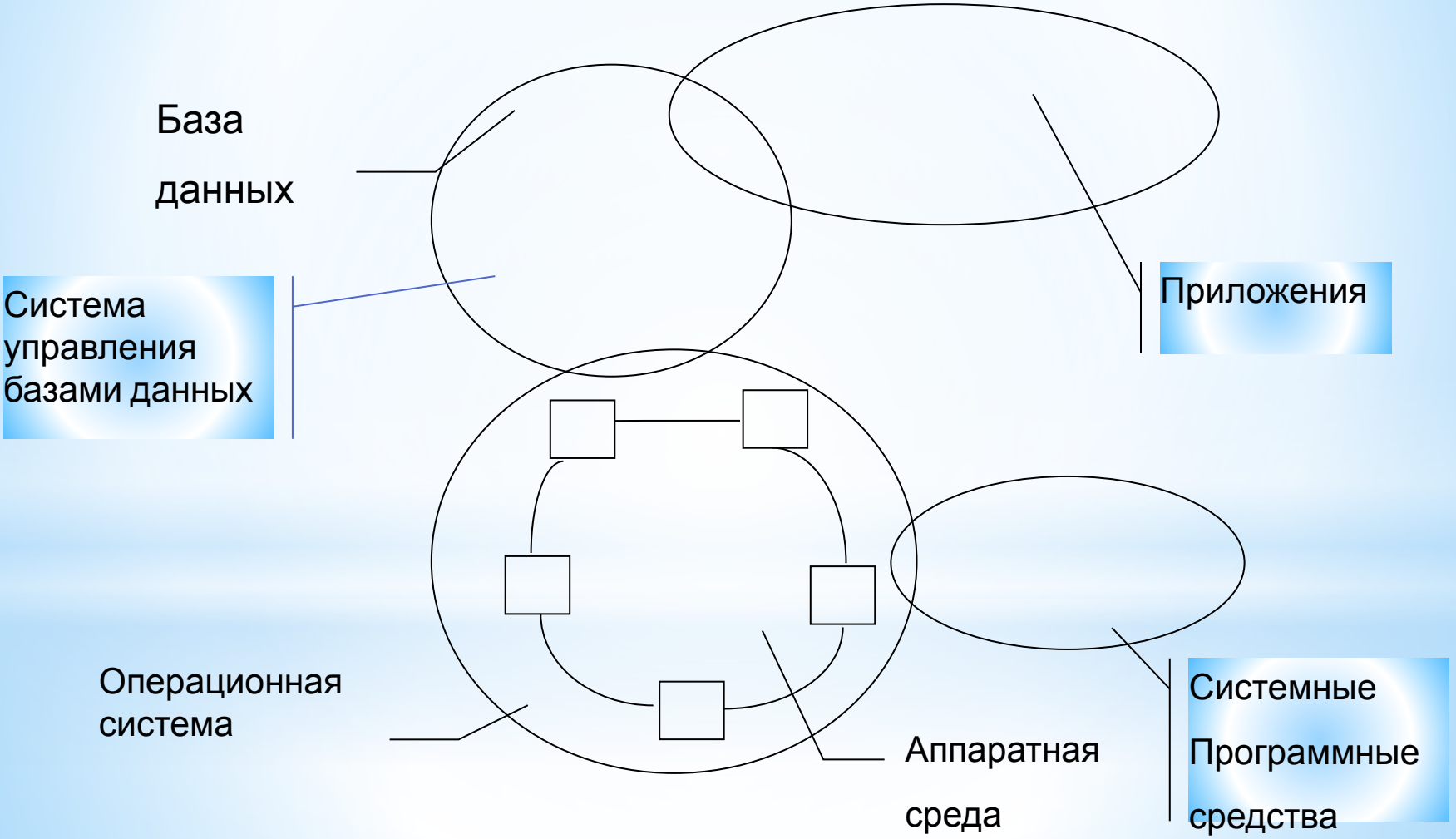
Базы принятия решений

Место БД в вычислительной среде можно отразить следующей схемой:



* Понятия и терминология баз данных

Обществом была осознана необходимость централизованного управления данными и появилось понятие банка или базы данных (БД). БД можно определить как взаимосвязанную совокупность данных, хранящуюся в электронном виде и предназначенную для коллективного использования. Появление БД привело к возникновению новых следующих понятий:

1. Системы управления базами данных (СУБД);
2. Администратор данных (АД) и администратор базы данных (АБД);

СУБД представляет собой совокупность программных средств, предназначенных для организации хранения данных в электронном виде и доступа к ним.

Администратор данных – это человек, отвечающий за стратегию и политику принятия решений, связанных с данными объекта управления.

Администратор базы данных – это человек или группа людей, обеспечивающих проектирование структуры БД, управление созданием базы и поддержанием ее работоспособности, обучение и консультации пользователей.

С точки зрения структуры база данных представляет собой совокупность элементов данных, объединенных в логические записи, и связей между ними.

Связи между элементами данных или логическими записями отражают обязательные соответствия между ними.

Для отображения состава логических записей базы данных и связей между ними используют схемы различных видов, которые принято называть моделями данных.

*Уровни представления данных

Концептуальный уровень. Концептуальный уровень предполагает изображение модели в виде поименованных объектов и связей между ними.

Логический уровень. Логический уровень состоит из логических записей, составляющих их атрибутов и связей между ними.

Физический уровень или физическое представление так же характеризуется записями и связями между ними. Однако записи организованы в соответствии с физическими особенностями носителей, на которых они хранятся

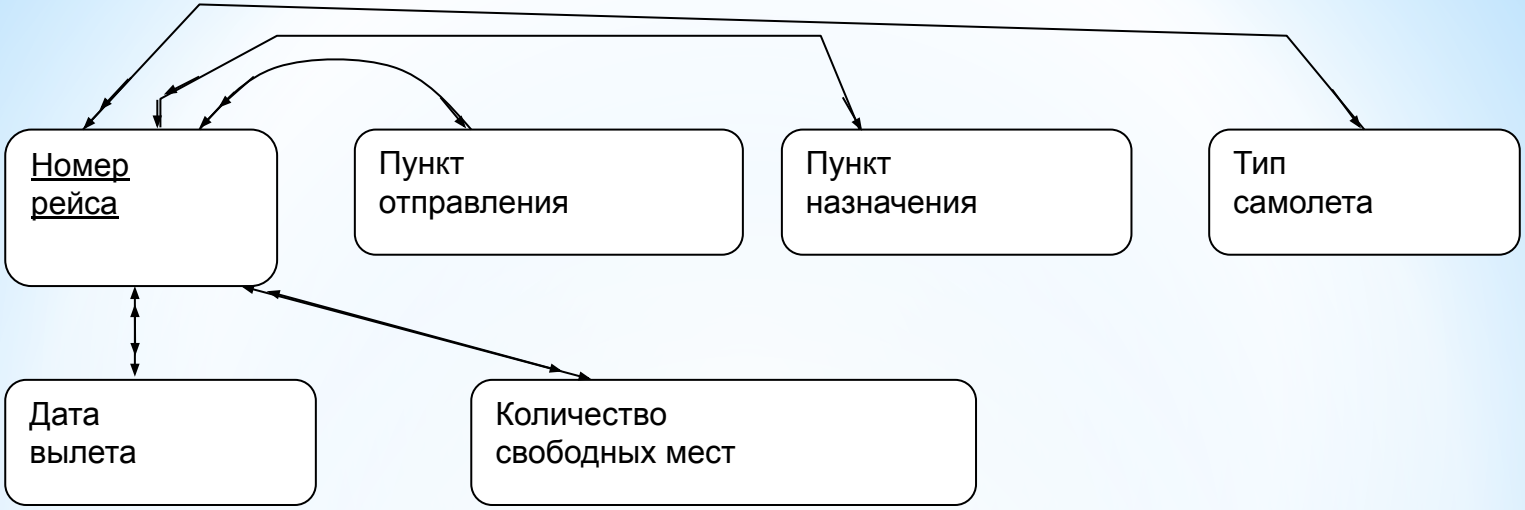
Между элементами A и B определена связь один к одному, если в каждый момент времени каждому элементу A соответствует только один ассоциированный с ним элемент B.
Между элементами A и B определена связь один ко многим, если в каждый момент времени каждому элементу A соответствует ноль, один или несколько ассоциированных с ним элементов B.
Связи между объектами (атрибутами) могут существовать в обоих направлениях, т.е. возможны четыре варианта связей: 1:1, 1:M, M:1, M:M.

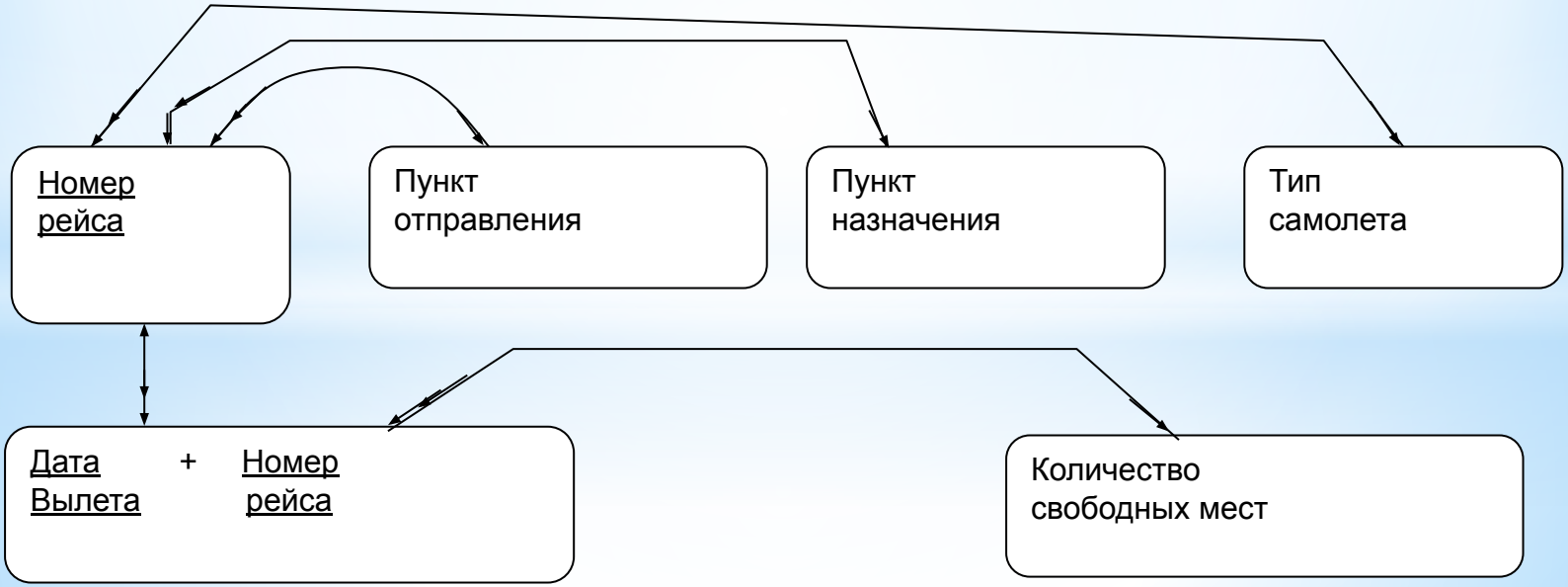
Связи в моделях

*Построение логических записей

Логическая запись описывает объект и его свойства и состоит из совокупности взаимосвязанных атрибутов. Причем один или несколько атрибутов отражают суть объекта, отличающую один экземпляр объекта от другого. Эти атрибуты называются *ключом*. Значения ключа являются уникальными для каждого типа записей. Все остальные атрибуты логической записи связаны с ключом. Причем допускаются связи с любой стороны ключа. Данные принципы создают формальную основу для образования логических записей.

* Построение логических записей

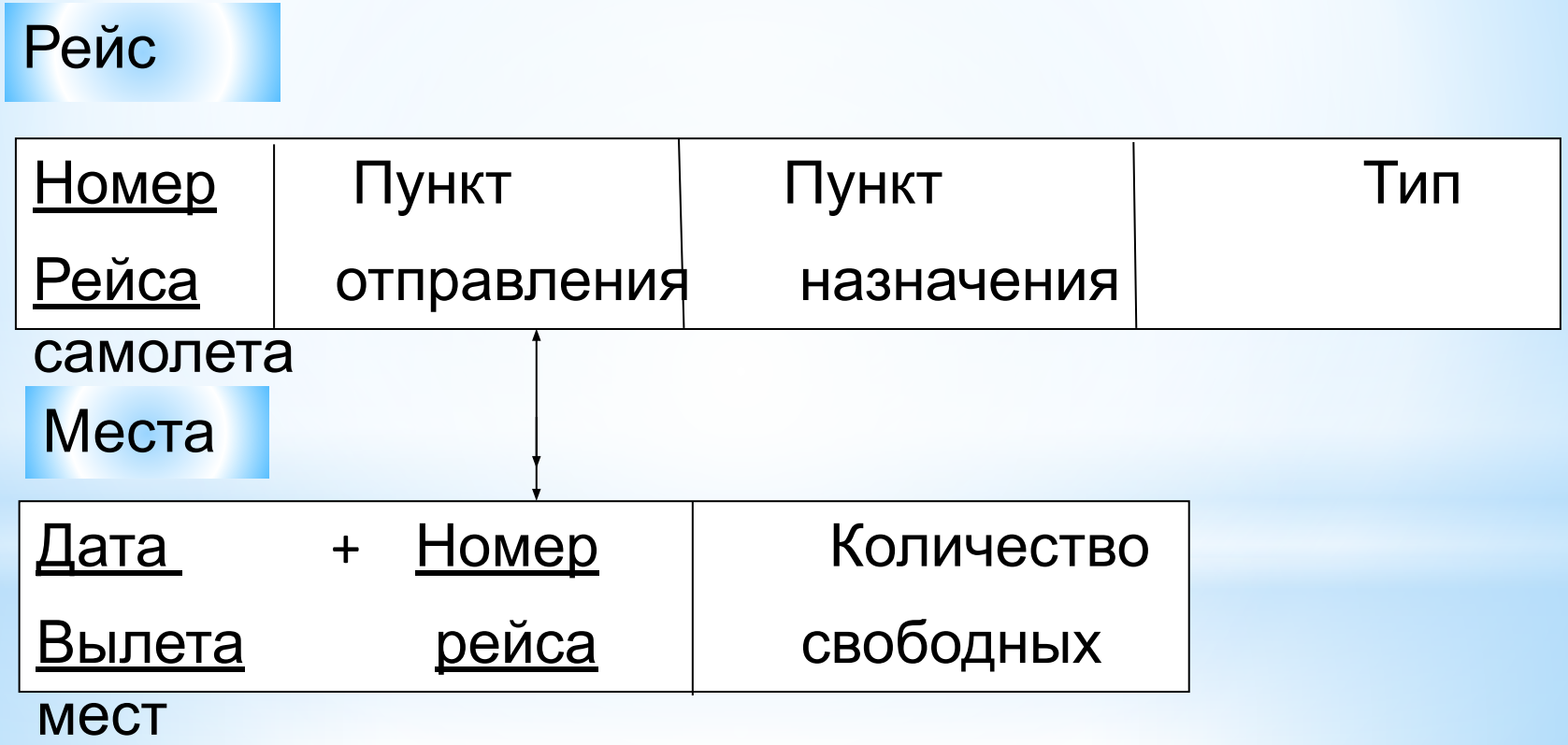




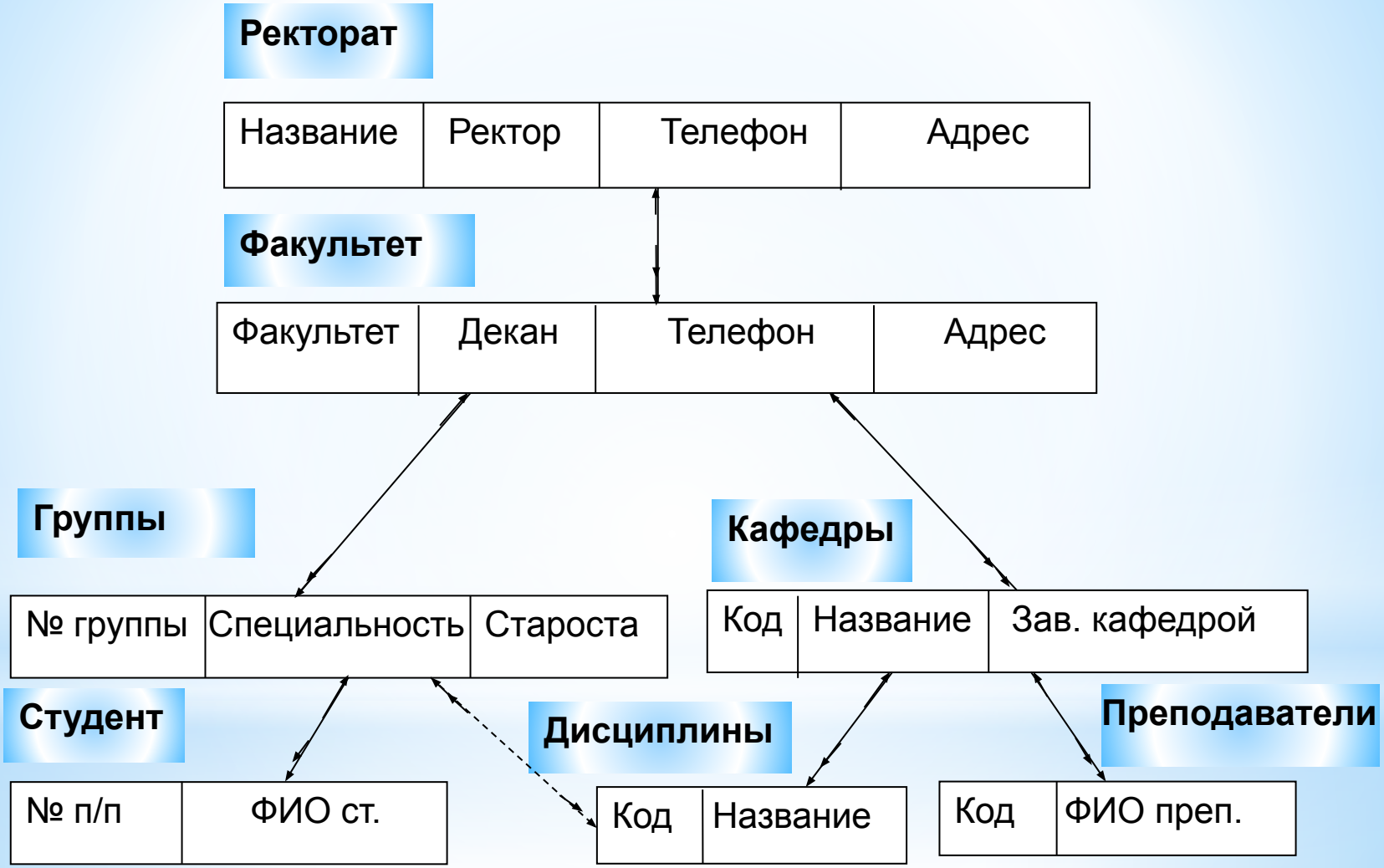
Пояснения к примеру

Поскольку все элементы данных в примере относятся к рейсам самолетов, то ключом логической записи логично было бы выбрать номер рейса. Действительно этот атрибут обладает основным свойством ключа – каждый рейс имеет свой уникальный номер. Связи между элементами определяются ролью каждого из них по отношению к ключу. Так, из пункта отправления могут отправляться много рейсов, но каждый рейс имеет только один пункт отправления, поэтому связь между ними со стороны ключа будет M:1. Следовательно, эти атрибуты можно объединить в логическую запись. То же касается атрибутов «Пункт назначения», «Тип самолета». Между ключом и атрибутом «Дата вылета» существует связь M:M, так как один и тот рейс может вылетать в разные даты, а в одну и ту же дату могут вылетать разные рейсы, то есть эти атрибуты нельзя объединить в логическую запись. То же касается и атрибута «Количество свободных мест». Однако эти два атрибута несут существенную информационную нагрузку и должны быть включены в модель. Чтобы решить эту проблему, можно преобразовать схему так

Итоговая модель



Иерархические модели данных



Основное назначение модели данных это описание структуры базы данных. При этом БД используется для получения ответов на запросы. Критерием качества модели может служить ее возможность получать выборки данных в ответ на различные запросы. Под запросом понимается описание требований к выбираемым из базы данным. Рассмотрим реализацию трех запросов с использованием модели данных приведенного примера.

Найти группу, где учится студент А.

Найти факультет, где учится студент А.

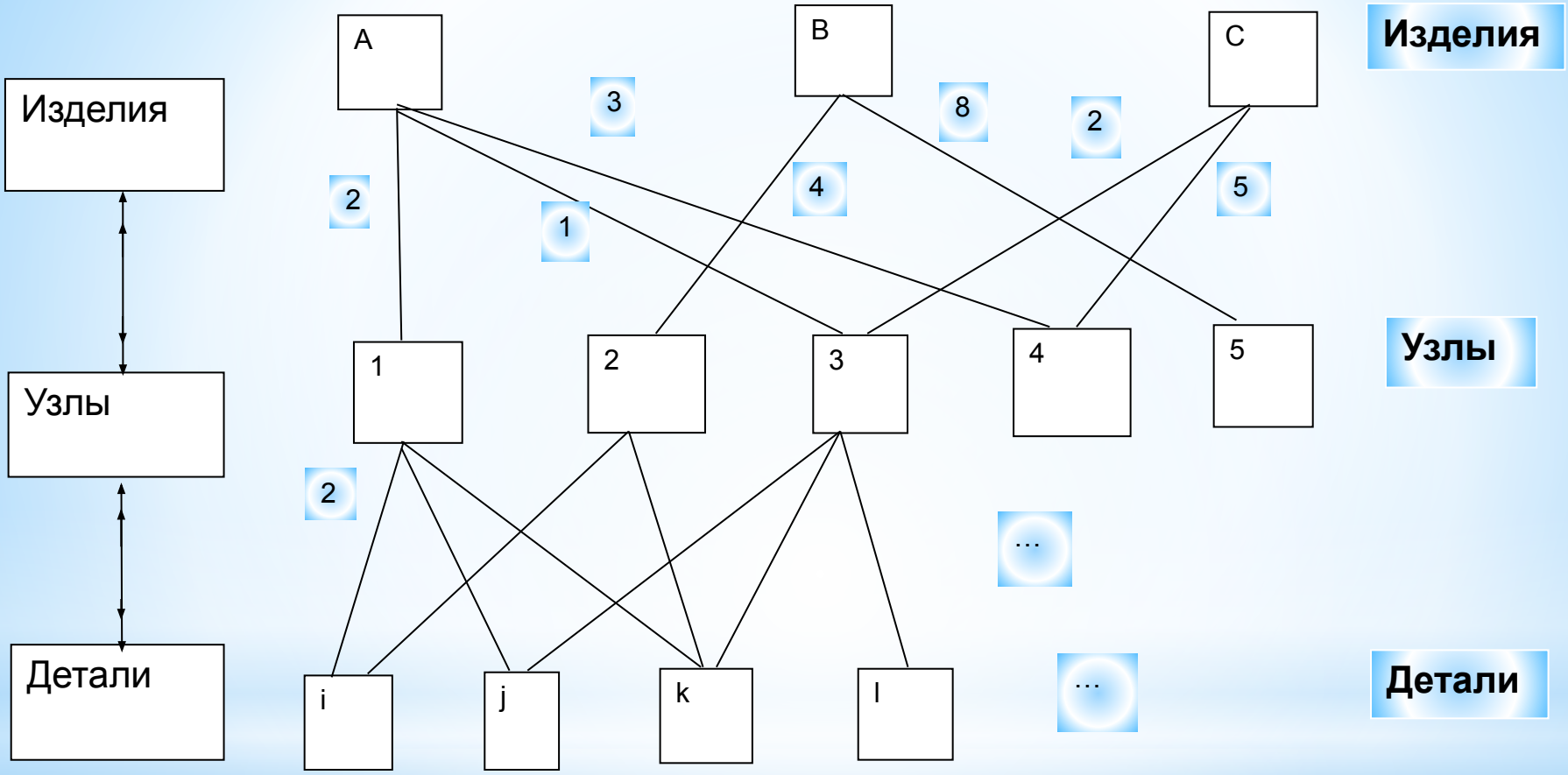
Найти дисциплины, которые изучает студент А.

* Функциональные возможности иерархической модели

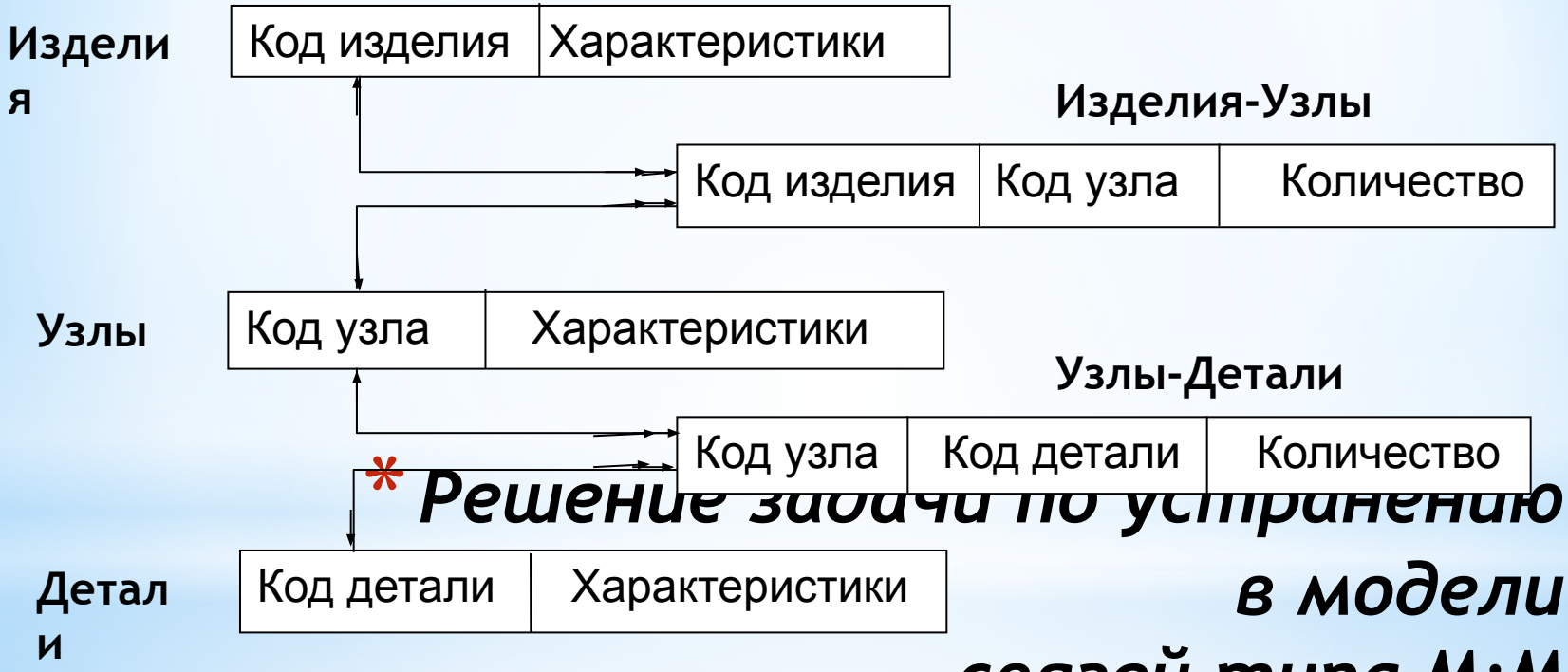
*Сетевые модели данных

В сетевых моделях связи могут устанавливаться произвольным образом. Кроме того в них допускаются связи M:M. Однако этот тип связей несет неопределенность соответствия записей. Этот тип связей показывает только характер соответствия записей, но не может быть использован для получения ответов на запросы. Проблемы, связанные с применением этого типа связей рассмотрим на следующем примере.

* Пример сетевой модели



Рядом со связями указаны данные пересечения



* Решение задачи по устранению в модели связей типа М:М

*Реляционные модели данных

Основу реляционной модели составляют таблицы или *отношения*. Под отношением понимают совокупность логически связанных между собой данных структурированных по строкам и столбцам. Строки отношения принято называть *кортежами*, а столбцы *доменами*. Под связью между таблицами (отношениями) понимается соответствие между значениями доменов отношений. В реляционных моделях допускаются связи 1:М, М:1 и 1:1.

Особенности связей в реляционных моделях

№ заказа	Поставщик	Дата	Товар	Характеристики	Цена
121	А…	17.04.03	П…	О……	256.00
121	А…	17.04.03	Р…	Л…	4598.00
121	А…	17.04.03	Е…	Н…..	785.00
122	В…	18.04.03	П…	О……	256.00

№ заказа	Поставщик	Дата
121	А…	17.04.03
122	В…	18.04.03

№ заказа	Товар	Характеристики	Цена
121	П…	О……	256.00
121	Р…	Л…	4598.00
121	Е…	Н…..	785.00
122	П…	О……	256.00

ЗАКАЗ(№ заказа, Поставщик, Дата)
ТОВАРЫ(№ заказа, Товар,
Характеристики, Цена)

*Преобразование сетевых моделей в реляционные (се

*Преобразование сетевых моделей в реляционные(реляционная модель)

Отделения

Код отделения	Наименование	Руководитель	Город

Здания

Номер здания	Адрес

Отделы

Код отдела	Код отделения	Наименование	Начальник	Телефон

Размещение

Номер Здания	Код отдела	Количество персонала

Проект

Номер проекта	Код отдела	Содержание	Дата окончания

Персонал

Табельный Номер	Номер здания	Код отдела	Номер проекта	Имя	Должность	Дата рождения

ОТДЕЛЕНИЯ(Код отделения, Наименование, Руководитель, Город)
ЗДАНИЯ(Номер здания, Адрес)
ОТДЕЛЫ(Код отдела, Код отделения, Наименование, Начальник, Телефон)
РАЗМЕЩЕНИЕ(Номер здания, Код отдела, Количество персонала)
ПРОЕКТ(Номер проекта, Код отдела, Содержание, Дата окончания)
ПЕРСОНАЛ(Табельный номер, Код отдела, Номер проекта, Имя, Должность, Дата рождения)

* *Стандарт описания реляционной модели*

B *функционально зависит* от атрибута A, если в каждый момент времени каждому значению атрибута A соответствует только одно, связанное с ним значение атрибута B и обозначают A $\longrightarrow$ B. Это же выражение можно прочитать как A *функционально определяет* B.

Например, пусть имеется множество атрибутов A, B, C и их значения, собранные в отношении R.

A	B	C
M	2	4
N	5	8
M	2	6
L	6	9
N	5	8
L	6	7

Попытаемся установить следующие зависимости A B и A C . Для этого отсортируем отношение по домену A:

A	B	C
M	2	4
M	2	6
N	5	8
N	5	8
L	6	9
L	6	7

Из таблицы видно, что каждому значению атрибута A соответствует только одно значение атрибута B и разные значения атрибута C. Следовательно A функционально определяет B, а C не зависит от A.

*Функциональная зависимость атрибутов

*Вторая нормальная форма

Отношения , в которых каждый атрибут не являющийся ключом функционально зависит только от одного возможного ключа представлены во *второй нормальной форме*. Пример.

ЗАКАЗ(Код поставщика, Код товара, Наименование поставщика, Адрес, Наименование товара, Характеристики товара, Цена)

В этом отношении ключ состоит из пары атрибутов Код поставщика, Код товара . При этом **Наименование поставщика, Адрес** функционально зависят от атрибута **Код поставщика** , **Наименование товара, Характеристики товара** зависят от **Код товара** , а **Цена** от ключа отношения. Такое разнообразие функциональных зависимостей приводит к следующим проблемам.

- При появлении нового поставщика необходимо добавлять строчку в отношение. Если он еще не начал поставлять товар, то все остальные атрибуты остаются не заполненными.
- Если из отношения удаляются сведения о поставщике, то удалятся и сведения о товарах.
- Для изменения адреса поставщика, наименование товара нужно проделывать это в нескольких строках отношения.
- Для изменения адреса поставщика, наименование товара нужно проделывать это в нескольких строках отношения.

Модель во второй нормальной форме

ПОСТАВЩИКИ(Код поставщика,
Наименование поставщика, Адрес)
ТОВАРЫ(Код товара, Наименование товара,
Характеристики товара)
ЗАКАЗ(Код поставщика, Код товара, Цена)

*Третья нормальная форма

Отношение задано в *третьей нормальной форме*, если оно представлено во второй нормальной форме и каждый атрибут не являющийся ключом не транзитивно зависит от ключа. Пример

ПЕРСОНАЛ(Табельный номер, ФИО, Должность, Номер проекта, Дата окончания)

Наличие транзитивной зависимости в примере приводит к следующим проблемам:

- при известных номере проекта и дате окончания их негде разместить пока не появятся сведения хотя бы об одном исполнителе;
- Если изменилась дата окончания проекта, ее надо менять в стольких кортежах, сколько людей работает над данным проектом.

Устранение этих проблем можно сделать, преобразовав исходное отношение к третьей нормальной форме:

ПЕРСОНАЛ(Табельный номер, ФИО, Должность, Номер проекта)
ПРОЕКТЫ(Номер проекта, Дата окончания)

*Схема проектирования реляционной модели данных (эмпирический подход)

Для создания реляционной модели данных при разработке базы данных для заданного объекта – предприятия необходимо выполнить следующие действия:

- Обследование информационной деятельности предприятия;
- Анализ информационных потоков и интеграция требований;
- Проектирование сетевой модели, отражающей структуру и информационные связи предприятия;
- Преобразование сетевой модели к реляционной;
- Нормализация отношений реляционной модели

*Основы реляционной алгебры

Реляционной алгеброй или алгеброй отношений называют систему операций манипулирования отношениями, каждый оператор которой в качестве операнда (операндов) имеет одно или несколько отношений, образуя новое отношение по заранее обусловленному правилу. Основными операциями реляционной алгебры являются:

- Операция проекции;
- Операция объединения;
- Операция разности;
- Операция декартова произведения;
- Операция селекции.

Операция проекции

Обозначение $\pi_{R(A)}$.
Представляет собой выборку кортежей отношения с неповторяющимися значениями домена А. Значения остальных доменов не играет роли.

Пример.

Сессия

Студент	Предмет	Семестр	Оценка
А..	Математика	1	4
А…	Информатика	1	3
Б..	Математика	1	5
Б…	Информатика	1	4
Б…	История	1	3
В...	Математика	1	4
В...	Информатика	1	3
В...	История	1	3

Проекция отношения $\pi_{\text{Сессия}}$

(Студент)

Студент	Предмет	Семестр	Оценка
А..	Математика	1	4
Б..	Математика	1	5
В...	Математика	1	4

Операция объединения

Обозначение операции R **U** S .

Объединение отношений R и S представляет собой множество кортежей, которые принадлежат отношениям либо R, либо S , либо им обоим.

Сессия 1

Студент	Предмет	Семестр	Оценка
А..	Математика	1	4
Б..	Математика	1	5
В...	Математика	1	3

Сессия 2

Студент	Предмет	Семестр	Оценка
А..	Математика	2	5
Б..	Математика	2	4
В...	Информатика	2	3
В...	История	2	4

Сессия1 U Сессия2

Студент	Предмет	Семестр	Оценка
А..	Математика	1	4
Б..	Математика	1	5
В...	Математика	1	4
А..	Математика	2	5
Б..	Математика	2	4
В...	Информатика	2	3
В...	История	2	4

Операция разности

Математическое обозначение R – S .
Разностью отношений называется множество кортежей входящих в R , но не входящих в S. Замечание по совместимости отношений справедливо и для разности.

Зачет – Экзамен

ФИО
Аверянов
Баранов
Вольский
Грачев
Григорьев
Дмитриев

Экзамен

ФИО
Баранов
Вольский
Григорьев
Дмитриев
Петров
Семенов

Зачет

ФИО
Аверьянов
Грачев

Операция декартово произведение

Математическое обозначение R x S.
Декартово произведение на двух отношениях определяет новое отношение, у которого число столбцов равно сумме числа столбцов исходных отношений, а число кортежей равно произведению числа кортежей операндов. При этом каждому кортежу первого отношения ставятся в соответствие все кортежи второго.

R

A	B
x	G
y	H
z	L

S

C	D
1	5
2	6

R x S

A	B	C	D
x	G	1	5
x	G	2	6
y	H	1	5
y	H	2	6
z	L	1	5
z	L	2	6

Операция селекции

Математическое обозначение $\sigma_{(A\ \theta\ B)}$ или $\sigma_{(A\ \theta\ V)}$.
Здесь A и B обозначения доменов, V – числовая или символьная константа, θ – знак логической операции (<,>,<>,<=,>=).
Операция селекции, это выборка кортежей со значениями доменов, удовлетворяющих заданному условию.

Студент	Предмет	Семестр	Оценка
А..	Математика	2	5
Б..	Математика	2	4
В...	Информатика	2	3
В...	История	2	4

селекция $\sigma_{(Оценка > 3)}$

Студент	Предмет	Семестр	Оценка
А..	Математика	2	5
Б..	Математика	2	4
В...	История	2	4

Операция пересечения

Операция обозначается R ∩ S и может быть выражена через операцию вычитания следующим образом: R – (R – S). По смыслу операция образует из двух отношений новое, которое включает совпадающие кортежи исходных отношений. Для примера рассмотрим исходные отношения операции вычитания. Если необходимо выяснить какие студенты сдали и зачет и экзамен, то результат будет получен при выполнении операции

Зачет -(Зачет -Экзамен)

Операция соединения

Математическое обозначение $R\,[\sigma_{(A\ \theta\ B)}]S$

Операция соединения представляет собой селекцию из декартова произведения.

Различают θ – соединение и естественное соединения.

Наряд

ФИО	Код	Объем
А…	01	10
Б…	02	15
В…	01	12
Г…	03	14

Нормы

Код	Наименование	Норма
01	Сварка	11
02	Расточка	14
03	Резка	15
04	Укладка	20

Тэта соединение

Наряд [Код = Код And Объем < Норма] Нормы

	ФИО	Код	Объем	Код	Наименование	Норма
√	А…	01	10	01	Сварка	11
	А…	01	10	02	Расточка	14
	А…	01	10	03	Резка	15
	А…	01	10	04	Укладка	20
	Б…	02	15	01	Сварка	11
	Б…	02	15	02	Расточка	14
	Б…	02	15	03	Резка	15
	Б…	02	15	04	Укладка	20
	В…	01	12	01	Сварка	11
	В…	01	12	02	Расточка	14
	В…	01	12	03	Резка	15
	В…	01	12	04	Укладка	20
	Г…	03	14	01	Сварка	11
	Г…	03	14	02	Расточка	14
√	Г…	03	14	03	Резка	15
	Г…	03	14	04	Укладка	20

В результирующее отношение попадут помеченные галочкой два кортежа

Естественное соединение

Если для указанного примера необходимо получить отношение, в котором определены объемы и нормы по каждому работнику, то выражение операции и результат должны выглядеть так: Наряд [Код = Код] Нормы

	ФИО	Код	Объем	Наименование	Норма
√	А…	01	10	Сварка	11
	А…	01	10	Расточка	14
	А…	01	10	Резка	15
	А…	01	10	Укладка	20
	Б…	02	15	Сварка	11
√	Б…	02	15	Расточка	14
	Б…	02	15	Резка	15
	Б…	02	15	Укладка	20
√	В…	01	12	Сварка	11
	В…	01	12	Расточка	14
	В…	01	12	Резка	15
	В…	01	12	Укладка	20
	Г…	03	14	Сварка	11
	Г…	03	14	Расточка	14
√	Г…	03	14	Резка	15
	Г…	03	14	Укладка	20

Галочкой помечены кортежи составляющие результат

*Реляционное исчисление

Наряду с реляционной алгеброй является способом получения результирующего отношения в реляционной модели данных. В реляционном исчислении различают:

- Исчисление кортежей
- Исчисление доменов

Исчисление кортежей - направление реляционного исчисления, где областями определения переменных (операндов) являются отношения базы данных, то есть допустимым значением каждой переменной является кортеж некоторого отношения. В исчислении кортежей, как и в процедурных языках программирования, сначала нужно описать используемые переменные, а затем записать выражения запроса к данным.

Описательную часть исчисления можно представить в виде: RANGE OF <переменная> IS <список>. Конструкция RANGE указывает идентификатор переменной кортежа <переменная> и область ее допустимых значений - <список> - последовательность одного или более элементов: x_1, ..., x_n, каждый из которых является либо отношением, либо выражением над отношением При этом в любой момент <переменная> принимает в качестве значения только один из кортежей <списка> отношений.

Схемы отношений списка должны быть эквивалентными. Область допустимых значений <переменной> образуется путем объединения значений всех элементов списка.

Пример:

RANGE OF Студент IS Очный_студент,Заочный_студент

Область определения переменной Студент включает в себя все значения из отношения, которое является объединением отношений Очный_студент и Заочный_студент.

Выражением реляционного исчисления кортежей называется конструкция вида <целевой_список > WHERE <WFF>
Значением выражения является отношение, тело (множество кортежей) которого должно удовлетворять WFF (well formulated formula — правильно построенная формула), а схема (набор атрибутов и их имена) определяется целевым списком. Целевой список по существу определяет операцию проекции, а формула WFF - селекцию кортежей.

В паре <переменная>.<атрибут> первая составляющая служит для указания переменной кортежа (определенной конструкцией RANGE), а вторая — для определения атрибута отношения, на котором изменяется переменная кортежа. Необязательная часть «AS <атрибут>» используется для переименования атрибута целевого отношения. Если она отсутствует, то имя атрибута целевого отношения наследуется от соответствующего имени атрибута исходного отношения.

WFF служат для выражения условий, накладываемых на кортежные переменные. Основой WFF являются простые сравнения, представляющие собой операции сравнения скалярных значений (значений атрибутов переменных или констант). Например, конструкция "СТУДЕНТ. НОМЕР_ЗАЧЕТНОЙ_КНИЖКИ = 625432" является простым сравнением. По определению, простое сравнение является WFF, а WFF, заключенная в круглые скобки, является простым

Более сложные варианты WFF строятся с помощью логических связок NOT, AND, OR и IF ... THEN. Так, если <формула> - WFF, а <сравнение> - простое сравнение, то

NOT <формула>

<сравнение> AND <формула>

<сравнение> OR <формула>

IF <сравнение> THEN <формула>

являются WFF.

Допускается построение WFF с помощью кванторов. Если <формула> - это WFF, в которой участвует <переменная>, то конструкции
EXISTS <переменная> (<формула>)
FORALL <переменная> (<формула>)
являются WFF.
В первом случае WFF означает: "Существует по крайней мере одно такое значение <переменной>, что вычисление <формулы> дает значение ИСТИНА".
Во втором случае WFF означает: "Для всех значений переменной <переменной> вычисление <формулы> дает значение ИСТИНА".

Пример. Пусть СОТР1 и СОТР2 - две кортежные переменные, определенные на отношении СОТРУДНИКИ.

СОТРУДНИКИ

ФИО	ЗАРПЛАТА
А…	14000
П…	9000
В…	6500
Л…	12000
Д…	8300

	СОТР2
(FORALL)СОТР1	true
(EXISTS)СОТР1	true

Тогда, WFF - EXISTS СОТР2 (СОТР1. ЗАРПЛАТА > СОТР2. ЗАРПЛАТА) для текущего кортежа переменной СОТР1 принимает значение true в том и только в том случае, если во всем отношении СОТРУДНИКИ найдется кортеж (связанный с переменной СОТР2) такой, что значение его атрибута . ЗАРПЛАТА удовлетворяет внутреннему условию сравнения.
WFF FORALL СОТР2 (СОТР1. ЗАРПЛАТА > СОТР2. ЗАРПЛАТА) для текущего кортежа переменной СОТР1 принимает значение true в том и только в том случае, если для всех кортежей отношения СОТРУДНИКИ (связанных с переменной СОТР2) значения атрибута ЗАРПЛАТА удовлетворяют условию сравнения.

Описанное исчисление не обладает вычислительной полнотой, так как не позволяет выполнять вычисления. Добавление вычислительных функций в исчисление можно реализовать путем расширения определения операндов сравнения и элементов целевого списка таким образом, чтобы они допускали использование скалярных выражений с литералами, ссылками на атрибуты и итоговыми функциями.
качестве итоговых могут выступать следующие функции: COUNT (количество), SUMM (сумма), AVG (среднее), MAX (максимальное), MIN (минимальное).

Для данных элементов целесообразно использовать спецификацию вида "AS <имя атрибута>, где можно явно задать имя результирующему атрибуту.
Пример.
Определить студента с максимальным рейтингом
Студент.ФИО, MAX(Рейтинг) AS Максимальный_Рейтинг
WHERE Студент. Номер_зачетной_книжки=Рейтинг.Номер_ зачетной_книжки _

В исчислении доменов областью определения переменных являются не отношения, а домены. Применительно к базе данных Рейтинг студентов можно говорить, например, о доменных переменных ИМЯ (значения - допустимые имена) или Номер_зачетной_книжки (значения - допустимые номера зачетных книжек студентов).

Основным формальным отличием исчисления доменов от исчисления кортежей является наличие дополнительного набора предикатов, позволяющих выражать так называемые условия членства. Если R - это n-арное отношение с атрибутами a_1, a_2, ..., a_n, то условие членства имеет вид $R(a_{1i}:v_{1i}...a_{im}:v_{im})(m <= n)$
где v_{ij} - это либо литерально задаваемая константа, либо имя кортежной переменной. Условие членства принимает значение true в том и только в том случае, если в отношении R существует кортеж, содержащий указанные значения указанных атрибутов.
Если v_{ij} - константа, то на атрибут a_{ij} задается жесткое условие, не зависящее от текущих значений доменных переменных; если же - имя доменной переменной, то условие членства может принимать разные значения при разных значениях этой переменной.

Для примера сформулируем с использованием исчисления доменов запрос "Выдать номера и имена сотрудников, не получающих минимальную заработную плату" (будем считать для простоты, что мы определили доменные переменные, имена которых совпадают с именами атрибутов отношения СОТРУДНИКИ, а в случае, когда требуется несколько доменных переменных, определенных на одном домене, мы будем добавлять в конце имени цифры):

СОТР_НОМ, СОТР_ИМЯ
WHERE EXISTS СОТР_ЗАРП1
(СОТРУДНИКИ (СОТР_ЗАРП1) AND
СОТРУДНИКИ (СОТР_НОМ, СОТР_ИМЯ, СОТР_ЗАРП) AND
СОТР_ЗАРП > СОТР_ЗАРП1)

* Реализация запросов с использованием языка SQL

Текущая версия стандарта языка SQL принята в 1992 г. (Официальное название стандарта - Международный стандарт языка баз данных SQL (1992) (International Standart Database Language SQL), неофициальное название - SQL/92, или SQL-92, или SQL2).
Язык SQL стал фактически стандартным языком доступа к базам данных. Все СУБД, претендующие на название "реляционные", реализуют тот или иной диалект SQL. Многие нереляционные системы также имеют в настоящее время средства доступа к реляционным данным. Целью стандартизации является переносимость приложений между различными СУБД.

Язык SQL оперирует терминами, несколько отличающимися от терминов реляционной теории, например, вместо "отношений" используются "таблицы", вместо "кортежей" - "строки", вместо "атрибутов" - "колонки" или "столбцы". Язык SQL является реляционно полным. Это означает, что любой оператор реляционной алгебры может быть выражен подходящим оператором SQL.

* Операторы SQL

Операторы DDL (Data Definition Language) - операторы определения объектов базы данных

CREATE SCHEMA - создать схему базы данных
DROP SHEMA - удалить схему базы данных
CREATE TABLE - создать таблицу
ALTER TABLE - изменить таблицу
DROP TABLE - удалить таблицу
CREATE DOMAIN - создать домен
ALTER DOMAIN - изменить домен
DROP DOMAIN - удалить домен
CREATE COLLATION - создать последовательность
DROP COLLATION - удалить последовательность
CREATE VIEW - создать представление
DROP VIEW - удалить представление

Операторы DML (Data Manipulation Language) - операторы манипулирования данными

SELECT - отобрать строки из таблиц
INSERT - добавить строки в таблицу
UPDATE - изменить строки в таблице
DELETE - удалить строки в таблице
COMMIT - зафиксировать внесенные изменения
ROLLBACK - откатить внесенные изменения

Операторы защиты и управления данными
CREATE ASSERTION - создать ограничение
DROP ASSERTION - удалить ограничение
GRANT - предоставить привилегии пользователю или приложению на манипулирование объектами
REVOKE - отменить привилегии пользователя или приложения
Кроме того, есть группы операторов установки параметров сеанса, получения информации о базе данных, операторы статического SQL, операторы динамического SQL.

* Реализация реляционной алгебры средствами оператора SELECT (Реляционная полнота SQL)

Оператор декартового произведения

Реляционная алгебра:

Оператор SQL:

```
SELECT A.Поле1, A.Поле2, …, B.Поле1, B.Поле2, …
    FROM A, B;
```

или

```
SELECT A.Поле1, A.Поле2, …, B.Поле1, B.Поле2, …
    FROM A CROSS JOIN B;
```

Оператор проекции

Реляционная алгебра:

Оператор SQL:

```
SELECT DISTINCT X, Y, …, Z
    FROM A;
```

Оператор выборки

Реляционная алгебра: ,

Оператор SQL:

```
SELECT *
    FROM A
    WHERE c;
```

Оператор объединения

Реляционная алгебра:

Оператор SQL:

```
SELECT *    FROM A
    UNION
    SELECT *  FROM B;
```

Оператор вычитания

Реляционная алгебра:

Оператор SQL:

```
SELECT *    FROM A
    EXCEPT
    SELECT *   FROM B
```

Реляционный оператор переименования RENAME выражается при помощи ключевого слова AS в списке отбираемых полей оператора SELECT.

Оператор соединения

Реляционная алгебра:

Оператор SQL:

```
SELECT A.Поле1, A.Поле2, …, B.Поле1, B.Поле2, …
    FROM A, B
    WHERE c;
```

или

```
SELECT A.Поле1, A.Поле2, …, B.Поле1, B.Поле2, …
    FROM A CROSS JOIN B
    WHERE c;
```

* Примеры использования операторов манипулирования данными

INSERT - вставка строк в таблицу

Пример 1. Вставка одной строки в таблицу:

```
INSERT INTO
  P (PNUM, PNAME)
  VALUES (4, "Иванов");
```

Пример 2. Вставка в таблицу нескольких строк, выбранных из другой таблицы (в таблицу TMP_TABLE вставляются данные о поставщиках из таблицы P, имеющие номера, большие 2):

```
INSERT INTO
  TMP_TABLE (PNUM, PNAME)
  SELECT PNUM, PNAME
    FROM P
    WHERE P.PNUM>2;
```

UPDATE - обновление строк в таблице

Пример 3. Обновление нескольких строк в таблице:

```
UPDATE P
  SET PNAME = "Пушников"
  WHERE P.PNUM = 1;
```

DELETE - удаление строк в таблице

Пример 4. Удаление нескольких строк в таблице:

```
DELETE FROM P
  WHERE P.PNUM = 1;
```

Пример 5. Удаление всех строк в таблице:

```
DELETE*  FROM P;
```

Примеры использования оператора SELECT

Оператор SELECT является фактически самым важным для пользователя и самым сложным оператором SQL. Он предназначен для выборки данных из таблиц, т.е. он, собственно, и реализует одно их основных назначение базы данных - предоставлять информацию пользователю.

Оператор SELECT всегда выполняется над некоторыми таблицами, входящими в базу данных.

Замечание. На самом деле в базах данных могут быть не только постоянно хранимые таблицы, а также временные таблицы и так называемые представления. Представления - это просто хранящиеся в базе данные SELECT-выражения. С точки зрения пользователей представления - это таблица, которая не хранится постоянно в базе данных, а "возникает" в момент обращения к ней.

С точки зрения оператора SELECT и постоянно хранимые таблицы, и временные таблицы и представления выглядят совершенно одинаково. Конечно, при реальном выполнении оператора SELECT системой учитываются различия между хранимыми таблицами и представлениями, но эти различия *скрыты* от пользователя.

Результатом выполнения оператора SELECT всегда является таблица. Таким образом, по результатам действий оператор SELECT похож на операторы реляционной алгебры. Любой оператор реляционной алгебры может быть выражен подходящим образом сформулированным оператором SELECT. Сложность оператора SELECT определяется тем, что он содержит в себе все возможности реляционной алгебры, а также дополнительные возможности, которых в реляционной алгебре нет.

Отбор данных из одной таблицы

Пример 6. Выбрать все данные из таблицы поставщиков (ключевые слова ***SELECT*… *FROM*…**):

```
SELECT *
  FROM P;
```

Замечание. В результате получим новую таблицу, содержащую полную копию данных из исходной таблицы P.

Пример 7. Выбрать все строки из таблицы поставщиков, удовлетворяющих некоторому условию (ключевое слово ***WHERE*…**):

```
SELECT *
  FROM P
  WHERE P.PNUM > 2;
```

Замечание. В качестве условия в разделе WHERE можно использовать сложные логические выражения, использующие поля таблиц, константы, сравнения (>, <, = и т.д.), скобки, союзы AND и OR, отрицание NOT.

Пример 8. Выбрать некоторые колонки из исходной таблицы (указание списка отбираемых колонок):

```
SELECT P.NAME
  FROM P;
```

Замечание. В результате получим таблицу с одной колонкой, содержащую все наименования поставщиков.

Замечание. Если в исходной таблице присутствовало несколько поставщиков с разными номерами, но одинаковыми наименованиями, то в результатирующей таблице *будут строки с повторениями* - дубликаты строк автоматически не отбрасываются.

Пример 9. Выбрать некоторые колонки из исходной таблицы, удалив из результата повторяющиеся строки (ключевое слово ***DISTINCT***):

```
SELECT DISTINCT P.NAME
  FROM P;
```

Замечание. Использование ключевого слова DISTINCT приводит к тому, что в результатирующей таблице будут удалены все повторяющиеся строки.

Пример 10. Использование скалярных выражений и переименований колонок в запросах (ключевое слово ***AS***…):

```
SELECT
    TOVAR.TNAME,
    TOVAR.KOL,
    TOVAR.PRICE,
    "=" AS EQU,
    TOVAR.KOL*TOVAR.PRICE AS SUMMA
  FROM TOVAR;
```

В результате получим таблицу с колонками, которых не было в исходной таблице TOVAR:

TNAME	KOL	PRICE	EQU	SUMMA
Болт	10	100	=	1000
Гайка	20	200	=	4000
Винт	30	300	=	9000

Пример 11.Упорядочение результатов запроса (ключевое слово ***ORDER BY***…):

```
SELECT
    PD.PNUM,
    PD.DNUM,
    PD.VOLUME
  FROM PD
  ORDER BY DNUM;
```

В результате получим следующую таблицу, упорядоченную по полю DNUM:

PNUM	DNUM	VOLUME
1	1	100
2	1	150
3	1	1000
1	2	200
2	2	250
1	3	300

Пример 12. Упорядочение результатов запроса по нескольким полям с возрастанием или убыванием (ключевые слова ***ASC***, ***DESC***):

```
SELECT    PD.PNUM,    PD.DNUM,    PD.VOLUME
  FROM PD
  ORDER BY     DNUM ASC,    VOLUME DESC;
```

В результате получим таблицу, в которой строки идут в порядке возрастания значения поля DNUM, а строки, с одинаковым значением DNUM идут в порядке убывания значения поля VOLUME:

PNUM	DNUM	VOLUME
3	1	1000
2	1	150
1	1	100
2	2	250
1	2	200
1	3	300

Замечание. Если явно не указаны ключевые слова ASC или DESC, то по умолчанию принимается упорядочение по возрастанию (ASC).

Отбор данных из нескольких таблиц

Пример 13. Естественное соединение таблиц (способ 1 - явное указание условий соединения):

```
SELECT P.PNUM, P.PNAME, PD.DNUM, PD.VOLUME
  FROM P, PD
  WHERE P.PNUM = PD.PNUM;
```

PNUM	PNAME	DNUM	VOLUME
1	Иванов	1	100
1	Иванов	2	200
1	Иванов	3	300
2	Петров	1	150
2	Петров	2	250
3	Сидоров	1	1000

В результате получим новую таблицу, в которой строки с данными о поставщиках соединены со строками с данными о поставках деталей:
Замечание. Соединяемые таблицы перечислены в разделе FROM оператора, условие соединения приведено в разделе WHERE. Раздел WHERE, помимо условия соединения таблиц, может также содержать и условия отбора строк.

* Использование имен корреляции (алиасов, псевдонимов)

Иногда приходится выполнять запросы, в которых таблица соединяется сама с собой, или одна таблица соединяется дважды с другой таблицей. При этом используются ***имена корреляции*** (***алиасы***, ***псевдонимы***), которые позволяют различать соединяемые копии таблиц. Имена корреляции вводятся в разделе FROM и идут через пробел после имени таблицы. Имена корреляции должны использоваться в качестве префикса перед именем столбца и отделяются от имени столбца точкой. Если в запросе указываются одни и те же поля из разных экземпляров одной таблицы, они должны быть переименованы для устранения неоднозначности в именованиях колонок результатирующей таблицы. Определение имени корреляции действует только во время выполнения запроса.

Пусть дана следующая таблица P

PNUM	PNAME	PSTATUS
1	Иванов	4
2	Петров	1
3	Сидоров	2

Отобрать все пары поставщиков таким образом, чтобы первый поставщик в паре имел статус, больший статуса второго поставщика:

```
SELECT    P1.PNAME AS PNAME1,    P1.PSTATUS AS PSTATUS1,    P2.PNAME AS
PNAME2,
    P2.PSTATUS AS PSTATUS2
  FROM    P P1, P P2
  WHERE P1.PSTATUS1 > P2.PSTATUS2;
```

В результате получим следующую таблицу:

PNAME1	PSTATUS1	PNAME2	PSTATUS2
Иванов	4	Петров	1
Иванов	4	Сидоров	2
Сидоров	2	Петров	1

* Использование агрегатных функций в запросах

Пример 21. Получить общее количество поставщиков (ключевое слово ***COUNT***):

```
SELECT COUNT(*) AS N  FROM P;
```

В результате получим таблицу с одним столбцом и одной строкой, содержащей количество строк из таблицы P:

N
3

Пример 22. Получить общее, максимальное, минимальное и среднее количества поставляемых деталей (ключевые слова ***SUM***, ***MAX***, ***MIN***, ***AVG***):

```
SELECT    SUM(PD.VOLUME) AS SM,    MAX(PD.VOLUME) AS MX,    MIN(PD.VOLUME)
AS MN,
    AVG(PD.VOLUME) AS AV
  FROM PD;
```

В результате получим следующую таблицу с одной строкой:

SM	MX	MN	AV
2000	1000	100	333.33333333

* Использование агрегатных функций с группировками

Пример 23. Для каждой детали получить суммарное поставляемое количество (ключевое слово ***GROUP BY***…):

```
SELECT    PD.DNUM,    SUM(PD.VOLUME) AS SM
  GROUP BY PD.DNUM;
```

Этот запрос будет выполняться следующим образом. Сначала строки исходной таблицы будут сгруппированы так, чтобы в каждую группу попали строки с одинаковыми значениями DNUM. Потом внутри каждой группы будет просуммировано поле VOLUME. От каждой группы в результатирующую таблицу будет включена одна строка:

DNUM	SM
1	1250
2	450
3	300

Замечание. В списке отбираемых полей оператора SELECT, содержащего раздел GROUP BY можно включать *только* агрегатные функции и поля, *которые входят в условие группировки*. Следующий запрос выдаст синтаксическую ошибку:

```
SELECT    PD.PNUM,    PD.DNUM,    SUM(PD.VOLUME) AS SM  GROUP BY PD.DNUM;
```

Причина ошибки в том, что в список отбираемых полей включено поле PNUM, которое *не входит* в раздел GROUP BY. И действительно, в каждую полученную группу строк может входить несколько строк с *различными* значениями поля PNUM. Из каждой группы строк будет сформировано по одной итоговой строке. При этом нет однозначного ответа на вопрос, какое значение выбрать для поля PNUM в итоговой строке.

* Использование подзапросов

Очень удобным средством, позволяющим формулировать запросы более понятным образом, является возможность использования подзапросов, вложенных в основной запрос.

Пример 25. Получить список поставщиков, статус которых меньше максимального статуса в таблице поставщиков (сравнение с подзапросом):

```
SELECT *
  FROM P
  WHERE P.STATYS <
    (SELECT MAX(P.STATUS)
       FROM P);
```

Замечание. Т.к. поле P.STATUS сравнивается с результатом подзапроса, то подзапрос должен быть сформулирован так, чтобы возвращать таблицу, состоящую *ровно из одной строки и одной колонки*.

Замечание. Результат выполнения запроса будет эквивалентен результату следующей последовательности действий:

1. Выполнить *один раз* вложенный подзапрос и получить максимальное значение статуса.
2. Просканировать таблицу поставщиков P, каждый раз сравнивая значение статуса поставщика с результатом подзапроса, и отобрать только те строки, в которых статус меньше максимального.

Пример 26. Использование предиката ***IN***. Получить список поставщиков, поставляющих деталь номер 2:

```
SELECT *
  FROM P
  WHERE P.PNUM IN
    (SELECT DISTINCT PD.PNUM
        FROM PD
        WHERE PD.DNUM = 2);
```

Замечание. В данном случае вложенный подзапрос может возвращать таблицу, содержащую несколько строк.

Замечание. Результат выполнения запроса будет эквивалентен результату следующей последовательности действий:

1. Выполнить *один раз* вложенный подзапрос и получить список номеров поставщиков, поставляющих деталь номер 2.
2. Просканировать таблицу поставщиков P, каждый раз проверяя, содержится ли номер поставщика в результате подзапроса.

Пример 27. Использование предиката ***EXIST***. Получить список поставщиков, поставляющих деталь номер 2:

```
SELECT *  FROM P
  WHERE EXIST
    (SELECT *    FROM PD
       WHERE
          PD.PNUM = P.PNUM AND
          PD.DNUM = 2);
```

Замечание. Результат выполнения запроса будет эквивалентен результату следующей последовательности действий:

Просканировать таблицу поставщиков P, *каждый раз выполняя подзапрос* с новым значением номера поставщика, взятым из таблицы P.

В результат запроса включить только те строки из таблицы поставщиков, для которых вложенный подзапрос вернул непустое множество строк.

Замечание. В отличие от двух предыдущих примеров, вложенный подзапрос содержит параметр (внешнюю ссылку), передаваемый из основного запроса - номер поставщика P.PNUM. Такие подзапросы называются ***коррелируемыми*** (***correlated***). Внешняя ссылка может принимать различные значения для каждой строки-кандидата, оцениваемого с помощью подзапроса, поэтому подзапрос должен выполняться заново для каждой строки, отбираемой в основном запросе. Такие подзапросы характерны для предиката EXIST, но могут быть использованы и в других подзапросах.

* Использование объединения, пересечения и разности

Пример 30. Получить имена поставщиков, имеющих статус, больший 3 или поставляющих хотя бы одну деталь номер 2 (объединение двух подзапросов - ключевое слово ***UNION***):

```
SELECT P.PNAME
  FROM P
  WHERE P.STATUS > 3
  UNION
  SELECT P.PNAME
    FROM P, PD
    WHERE P.PNUM = PD.PNUM AND
       PD.DNUM = 2;
```

Замечание. Результатирующие таблицы объединяемых запросов должны быть совместимы, т.е. иметь одинаковое количество столбцов и одинаковые типы столбцов в порядке их перечисления. *Не требуется*, чтобы объединяемые таблицы имели бы одинаковые имена колонок. Это отличает операцию объединения запросов в SQL от операции объединения в реляционной алгебре. Наименования колонок в результатирующем запросе будут автоматически взяты из результата первого запроса в объединении.

Пример 31. Получить имена поставщиков, имеющих статус, больший 3 и одновременно поставляющих хотя бы одну деталь номер 2 (пересечение двух подзапросов - ключевое слово ***INTERSECT***):

```
SELECT P.PNAME
  FROM P
  WHERE P.STATUS > 3
  INTERSECT
  SELECT P.PNAME
    FROM P, PD
    WHERE P.PNUM = PD.PNUM AND
        PD.DNUM = 2;
```

Пример 32. Получить имена поставщиков, имеющих статус, больший 3, за исключением тех, кто поставляет хотя бы одну деталь номер 2 (разность двух подзапросов - ключевое слово ***EXCEPT***):

```
SELECT P.PNAME
  FROM P
  WHERE P.STATUS > 3
EXCEPT
  SELECT P.PNAME
    FROM P, PD
    WHERE P.PNUM = PD.PNUM AND
        PD.DNUM = 2;
```

* Порядок выполнения оператора SELECT

Для того чтобы понять, как получается результат выполнения оператора SELECT, рассмотрим концептуальную схему его выполнения. Эта схема является именно концептуальной, т.к. гарантируется, что результат будет таким, как если бы он выполнялся шаг за шагом в соответствии с этой схемой. На самом деле, реально результат получается более изощренными алгоритмами, которыми "владеет" конкретная СУБД.

Стадия 1. Выполнение одиночного оператора SELECT
Если в операторе присутствуют ключевые слова UNION, EXCEPT и INTERSECT, то запрос разбивается на несколько независимых запросов, каждый из которых выполняется отдельно:
Шаг 1 (FROM). Вычисляется прямое декартовое произведение всех таблиц, указанных в обязательном разделе FROM. В результате шага 1 получаем таблицу A.
Шаг 2 (WHERE). Если в операторе SELECT присутствует раздел WHERE, то сканируется таблица A, полученная при выполнении шага 1. При этом для каждой строки из таблицы A вычисляется условное выражение, приведенное в разделе WHERE. Только те строки, для которых условное выражение возвращает значение TRUE, включаются в результат. Если раздел WHERE опущен, то сразу переходим к шагу 3. Если в условном выражении участвуют вложенные подзапросы, то они вычисляются в соответствии с данной концептуальной схемой. В результате шага 2 получаем таблицу B.

Шаг 3 (GROUP BY). Если в операторе SELECT присутствует раздел GROUP BY, то строки таблицы B, полученной на втором шаге, группируются в соответствии со списком группировки, приведенным в разделе GROUP BY. Если раздел GROUP BY опущен, то сразу переходим к шагу 4. В результате шага 3 получаем таблицу C.
Шаг 4 (HAVING). Если в операторе SELECT присутствует раздел HAVING, то группы, не удовлетворяющие условному выражению, приведенному в разделе HAVING, исключаются. Если раздел HAVING опущен, то сразу переходим к шагу 5. В результате шага 4 получаем таблицу D.
Шаг 5 (SELECT). Каждая группа, полученная на шаге 4, генерирует одну строку результата следующим образом. Вычисляются все скалярные выражения, указанные в разделе SELECT. По правилам использования раздела GROUP BY, такие скалярные выражения должны быть одинаковыми для всех строк внутри каждой группы. Для каждой группы вычисляются значения агрегатных функций, приведенных в разделе SELECT. Если раздел GROUP BY отсутствовал, но в разделе SELECT есть агрегатные функции, то считается, что имеется всего одна группа. Если нет ни раздела GROUP BY, ни агрегатных функций, то считается, что имеется столько групп, сколько строк отобрано к данному моменту. В результате шага 5 получаем таблицу E, содержащую столько колонок, сколько элементов приведено в разделе SELECT и столько строк, сколько отобрано групп.

Стадия 2. Выполнение операций UNION, EXCEPT, INTERSECT
Если в операторе SELECT присутствовали ключевые слова UNION, EXCEPT и INTERSECT, то таблицы, полученные в результате выполнения 1-й стадии, объединяются, вычитаются или пересекаются.
Стадия 3. Упорядочение результата
Если в операторе SELECT присутствует раздел ORDER BY, то строки полученной на предыдущих шагах таблицы упорядочиваются в соответствии со списком упорядочения, приведенном в разделе ORDER BY.

* Как на самом деле выполняется оператор SELECT

Если внимательно рассмотреть приведенный выше концептуальный алгоритм вычисления результата оператора SELECT, то сразу понятно, что выполнять его непосредственно в таком виде чрезвычайно накладно. Даже на самом первом шаге, когда вычисляется декартово произведение таблиц, приведенных в разделе FROM, может получиться таблица огромных размеров, причем практически большинство строк и колонок из нее будет отброшено на следующих шагах.

На самом деле в РСУБД имеется ***оптимизатор***, функцией которого является нахождение такого *оптимального алгоритма* выполнения запроса, который гарантирует получение правильного результата.

Схематично работу оптимизатора можно представить в виде последовательности нескольких шагов:

Шаг 1 (Синтаксический анализ). Поступивший запрос подвергается синтаксическому анализу. На этом шаге определяется, правильно ли вообще (с точки зрения синтаксиса SQL) сформулирован запрос. В ходе синтаксического анализа вырабатывается некоторое внутренне представление запроса, используемое на последующих шагах.

Шаг 2 (Преобразование в каноническую форму). Запрос во внутреннем представлении подвергается преобразованию в некоторую каноническую форму. При преобразовании к канонической форме используются как синтаксические, так и семантические преобразования. Синтаксические преобразования (например, приведения логических выражений к конъюнктивной или дизъюнктивной нормальной форме, замена выражений "x AND NOT x" на "FALSE", и т.п.) позволяют получить новое внутренне представление запроса, синтаксически *эквивалентное* исходному, но стандартное в некотором смысле. Семантические преобразования используют дополнительные знания, которыми владеет система, например, ограничения целостности. В результате семантических преобразований получается запрос, синтаксически *не эквивалентный* исходному, но дающий *тот же самый результат*.
Шаг 3 (Генерация планов выполнения запроса и выбор оптимального плана). На этом шаге оптимизатор генерирует множество возможных планов выполнения запроса. Каждый план строится как комбинация низкоуровневых процедур доступа к данным из таблиц, методам соединения таблиц. Из всех сгенерированных планов выбирается план, обладающий минимальной стоимостью. При этом анализируются данные о наличии индексов у таблиц, статистических данных о распределении значений в таблицах, и т.п. Стоимость плана это, как правило, сумма стоимостей выполнения отдельных низкоуровневых процедур, которые используются для его выполнения. В стоимость выполнения отдельной процедуры могут входить оценки количества обращений к дискам, степень загруженности процессора и

Шаг 4. (Выполнение плана запроса). На этом шаге план, выбранный на предыдущем шаге, передается на реальное выполнение.
Во многом качество конкретной СУБД определяется качеством ее оптимизатора. Хороший оптимизатор может повысить скорость выполнения запроса на несколько порядков. Качество оптимизатора определяется тем, какие методы преобразований он может использовать, какой статистической и иной информацией о таблицах он располагает, какие методы для оценки стоимости выполнения плана он знает.