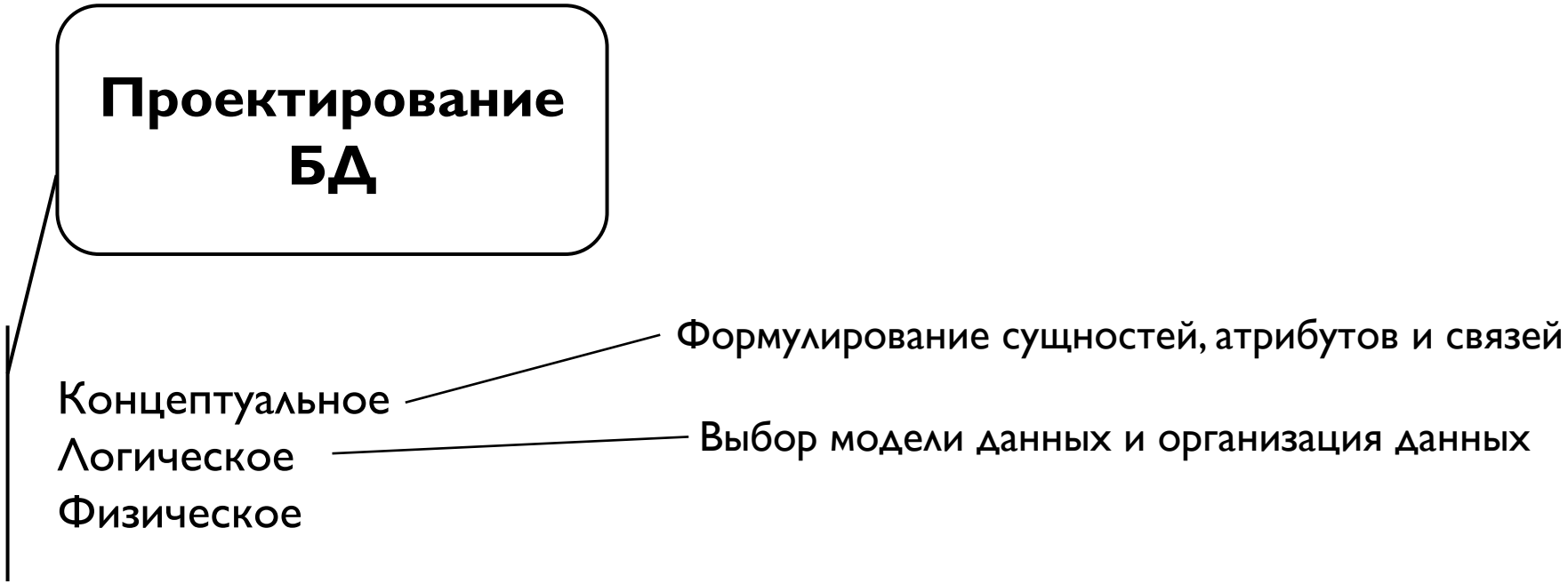
Лекция 8

Логическое проектирование

Логическое проектирование



Логическое проектирование в реляционных БД

В реляционных БД даталогическое или логическое проектирование приводит к разработке схемы БД, то есть совокупности схем отношений, которые адекватно моделируют абстрактные объекты предметной области и семантические связи между этими объектами.

Получение реляционной схемы из ER-схемы

Приведем методику получения из ER-схемы реляционной схемы. Процесс получения требуемой схемы состоит из следующих шагов:

1) Каждая простая сущность превращается в таблицу. (Простой называется сущность, не являющаяся подтипом и не имеющая подтипов.) При этом имя сущности становится именем таблицы.

2) Каждый атрибут сущности становится возможным столбцом таблицы с тем же именем; при этом может выбираться более точный формат значений атрибута. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения, обязательным – не могут.

Получение реляционной схемы из ER-схемы

3) Компоненты уникального идентификатора сущности превращаются в первичный ключ таблицы. Если в состав уникального идентификатора входят связи, к числу столбцов первичного ключа добавляется копия уникального идентификатора сущности, находящейся на дальнем конце связи (этот процесс может продолжаться рекурсивно). Для именования этих столбцов используются имена концов связей и/или имена сущностей.

4) Связи "многие-к-одному" и "один-к-одному" становятся внешними ключами, т.е. делается копия уникального идентификатора с конца связи "один" и соответствующие ему столбцы составляют внешний ключ. При этом необязательные связи соответствуют столбцам, допускающим неопределенные значения, обязательные - столбцам, не допускающим неопределенные значения.

Логическое проектирование в реляционных БД

Основой анализа корректности схемы являются так называемые функциональные зависимости между атрибутами БД. Некоторые зависимости между атрибутами отношений являются нежелательными из-за побочных эффектов и аномалий, которые они вызывают при модификации БД. При этом под процессом модификации БД мы понимаем внесение новых данных в БД или удаление некоторых данных из БД, а также обновление значений некоторых атрибутов.

Логическое проектирование в реляционных БД

- аномалии включения;
- аномалии удаления;
- аномалии модификации.

Ном_зач_кн	ФИО_студента	Код_группы	ФИО_старосты	Куратор
20-Т-201	Иванов С.И.	20-Т-11	Рябов В.С.	Доц. Фок И.И.
20-Т-215	Петров Я.Р.	20-Т-12	Сизов М.М.	Доц. Докин С.С.
20-Т-217	Рябов В.С	20-Т-11	Рябов В.С.	Доц. Фок И.И.
20-Т-211	Сенова А.Л.	20-Т-11	Рябов В.С.	Доц. Фок И.И.

Логическое проектирование в реляционных БД

Для устранения рассмотренных выше недостатков и применяется процесс нормализация отношений.
Теория нормализации отношений основана на анализе функциональных зависимостей между атрибутами отношений.

Функциональные зависимости определяют устойчивые отношения между объектами и их свойствами в рассматриваемой предметной области. Именно поэтому процесс поддержки функциональных зависимостей, характерных для данной предметной области, является базовым для процесса проектирования.

Логическое проектирование в реляционных БД

Процесс проектирования представляет собой процесс последовательной нормализации схем отношений, при этом каждая последующая итерация соответствует нормальной форме более высокого уровня обладает лучшими свойствами по сравнению с предыдущей.

Каждой нормальной форме соответствует некоторый определенный набор ограничений, отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений.

Логическое проектирование в реляционных БД

В теории реляционных БД обычно выделяется следующая последовательность нормальных форм:

- первая нормальная форма (НФ1);
- вторая нормальная форма (НФ2);
- третья нормальная форма (НФ3);
- нормальная форма Бойса—Кодда (НФБК);
- четвертая нормальная форма (НФ4);
- пятая нормальная форма, или форма проекции-соединения (НФ5 пли НФРJ).

каждая следующая нормальная форма в некотором смысле улучшает свойства предыдущей;
при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

Логическое проектирование в реляционных БД

Схемы БД называются *эквивалентными,* если содержание исходной БД может быть получено путем естественного соединения отношений, входящих в результирующую схему, и при этом не появляется новых кортежей в исходной БД.

Преобразование БД должно сохранять *эквивалентность* схем БД при замене одной схемы на другую.

Функциональная зависимость

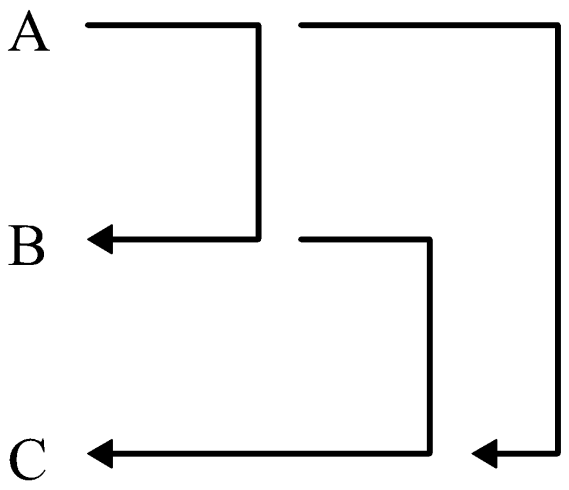
Определение 1. ***Функциональная зависимость***.
В отношении *R* атрибут *B* функционально зависит от атрибута *A* (*A* и *B* могут быть составными), если каждому значению атрибута *A* соответствует в точности одно значение атрибута *B*, т.е. имеет место отображение $R.A \rightarrow R.B$.
(Если известно значение атрибута *A*, можно получить значение атрибута *B*.)

Определение 2. ***Полная функциональная зависимость.***
Атрибут (набор атрибутов) *B* полностью зависит от другого набора атрибутов *A* отношения *R*, если *B* функционально зависит от всего множества *A*, но не зависит ни от какого подмножества *A*.

т.е. для любого A1, являющегося подмножеством A, R.B функционально не зависит от R.A, в противном случае зависимость R.A -> R.B называется неполной.

Функциональная зависимость

Определение 3. ***Транзитивная функциональная зависимость***.
Функциональная зависимость $R.A \rightarrow R.C$ называется транзитивной, если в отношении R существует такой атрибут B, что имеют место зависимости $R.A \rightarrow R.B$ и $R.B \rightarrow R.C$.



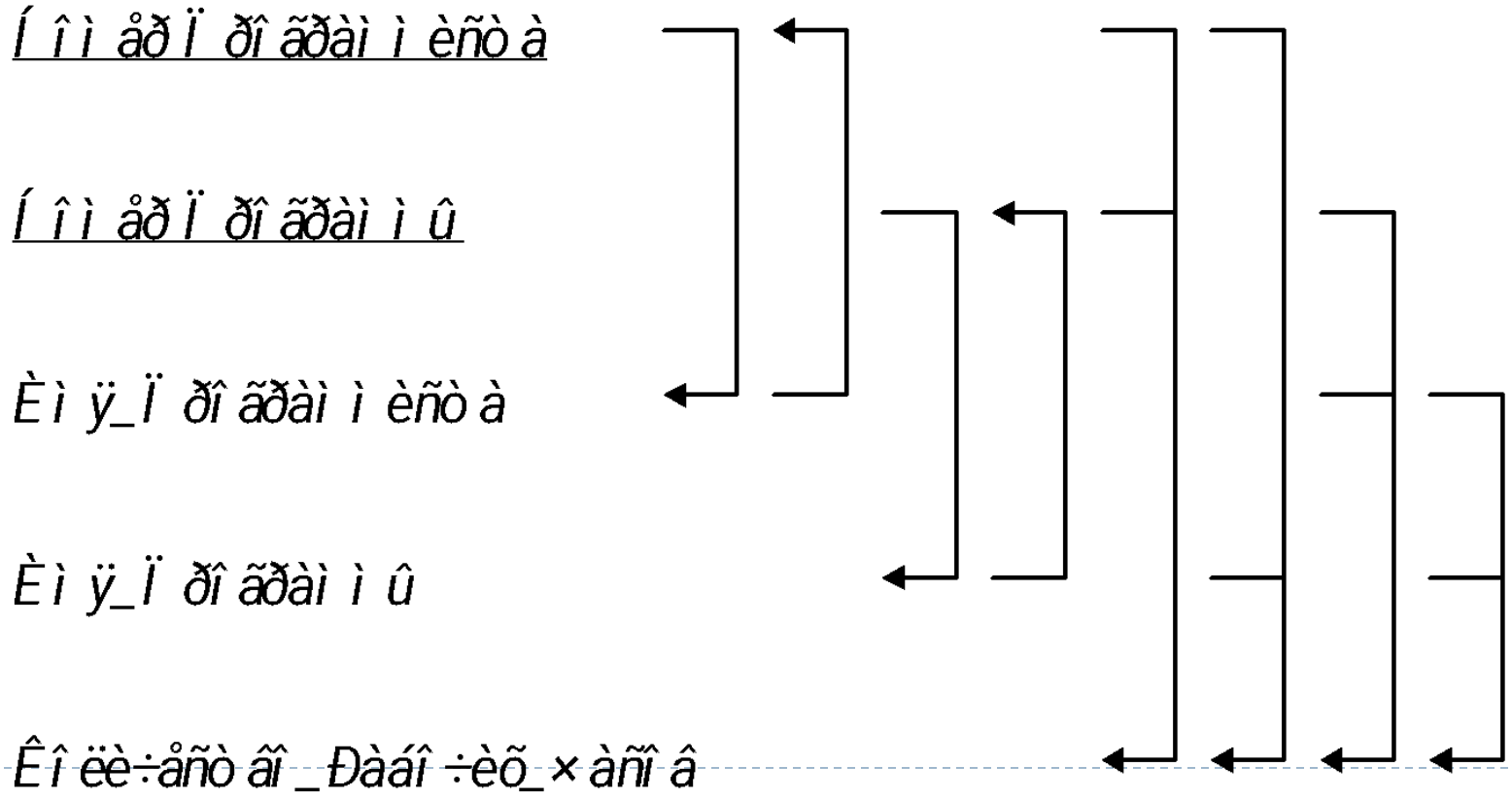
Определение 4. Неключевой атрибут.
Неключевым (неосновным) называется атрибут, не входящий в состав первичного ключа.

Определение 5. ***Возможным ключом*** отношения называется набор атрибутов отношения, который полностью и однозначно (функционально полно) определяет значения всех остальных атрибутов отношения, то есть возможный ключ — это набор атрибутов, однозначно определяющий кортеж отношения, и при этом при удалении любого атрибута из этого набора его свойство однозначной идентификации кортежа теряется.

Определение 6. Если в отношении существует несколько функциональных зависимостей, то каждый атрибут или набор атрибутов, от которого зависит другой атрибут, называтся ***детерминантом отношения.***

ДЕЯТЕЛЬНОСТЬ_ПРОГРАММИСТА

(Номер Программиста, Номер Программы, Имя_Программиста, Имя_Программы, Количество_Рабочих_Часов).



Первая нормальная форма

Отношение находится в первой нормальной форме тогда и только тогда, когда па пересечении каждого столбца и каждой строки находятся только элементарные значения атрибутов.

В некотором смысле это определение избыточно, потому что собственно оно определяет само отношение в теории реляционных баз данных. Отношения, находящиеся в первой нормальной форме, часто называют просто нормализованными отношениями. Соответственно, ненормализованные отношения могут интерпретироваться как таблицы с неравномерным заполнением, например

Первая нормальная форма

Преподаватель	День недели	Номер пары	Название дисциплины	Тип занятий	Группа
Петров	Пн Вт Вт	1 1 2	ТЕОР.ВЫЧ.ПРОЦ. КОМП.ГРАФИКА КОМП. ГРАФИКА	Лекция Лаб. Лаб.	4906 4907 4906
Киров	Пн Вт Вт	2 3 4	Теор. Информ Пр-е на С++ Пр-е на С++	Лекция Лаб. Лаб.	4906 4907 4906
Серов	Пн Ср Чт	3 4 3	Защита инф. VB VB	Лекция Лаб. Лаб.	4944 4942 4922

Для приведения отношения «Расписание» к первой нормальной форме необходимо дополнить каждую строку фамилией преподавателя.

Вторая нормальная форма

Отношение находится во второй нормальной форме тогда и только тогда, когда оно находится в первой нормальной форме и не содержит неполных функциональных зависимостей непервичных атрибутов от атрибутов первичного ключа.

Т.е. каждый неключевой атрибут полностью зависит от первичного ключа.

Если в отношении R ключ содержит один атрибут, то это отношение уже задано в НФ-2.

Вторая нормальная форма

Рассмотрим отношение
ФИО, НомерЗач, Группа, Дисциплина, Оценка)

первичным ключом отношения может быть (НомерЗач , Дисциплина)
С другой стороны, атрибуты ФИО и Группа зависят только от части первичного ключа — от значения атрибута НомерЗач, поэтому есть неполные функциональные зависимости.
Для приведения ко второй нормальной форме – разбить на проекции

(ФИО, НомерЗач, Группа)

(НомерЗач, Дисциплина, Оценка)

Этот набор отношений не содержит неполных функциональных зависимостей, и поэтому эти отношения находятся во второй нормальной форме. '

Вторая нормальная форма

почему надо приводить отношения ко второй нормальной форме?
ФИО, НомерЗач, Группа, Дисциплина, Оценка)

Ситуация - студент переведен из одной группы в другую. Мы должны найти все записи сданным студентом и в них изменить значение атрибута Группа на новое. Есть опасность нарушения корректности (непротиворечивости содержания)

Кроме того, если у нас есть студенты, которые еще не сдавали экзамены, то в исходном отношении мы вообще не можем хранить о них информацию

Третья нормальная форма

Отношение *R* находится в третьей нормальной форме, если оно находится во второй нормальной форме и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Определение 3. ***Транзитивная функциональная зависимость***.
Функциональная зависимость $R.A \rightarrow R.C$ называется транзитивной, если в отношении R существует такой атрибут B, что имеют место зависимости $R.A \rightarrow R.B$ и $R.B \rightarrow R.C$.

КОНСУЛЬТАЦИИ (Таб_Ном_преп, Ном_зач_кн, Дата, Время, Аудитория, Вместимость)

отношение содержит транзитивную зависимость:
(Таб_Ном_преп, Ном_зач_кн, Дата) → Аудитория → Вместимость.

Третья нормальная форма

КОНСУЛЬТАЦИИ (Таб_Ном_преп, Ном_зач_кн, Дата, Время, Аудитория, Вместимость)

отношение содержит транзитивную зависимость:
(Таб_Ном_преп, Ном_зач_кн, Дата) → Аудитория → Вместимость.

Следовательно, это отношение не находится в НФ-3 со всеми вытекающими из этого последствиями :

- если аудитория исключается из расписания консультаций, то о ней вообще теряются сведения;
- если аудитория перестроена и в результате изменилась ее вместимость, то придется просмотреть все кортежи и провести модификацию значений атрибута.

Третья нормальная форма

КОНСУЛЬТАЦИИ (Таб_Ном_преп, Ном_зач_кн, Дата, Время, Аудитория, Вместимость)

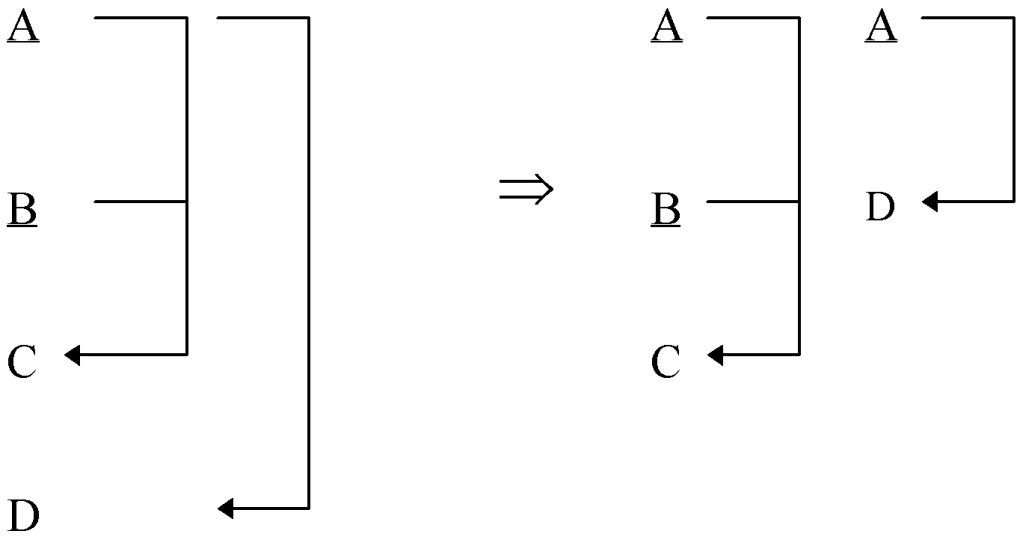
отношение содержит транзитивную зависимость:
(Таб_Ном_преп, Ном_зач_кн, Дата) → Аудитория → Вместимость.

Для устранения транзитивной зависимости необходимо провести декомпозицию последнего отношения, удалив из него транзитивно-зависимый атрибут и поместив его в новое отношение вместе с копией того атрибута, от которого он зависит

КОНСУЛЬТАЦИИ (Таб_Ном_преп, Ном_зач_кн, Дата, Время, Аудитория)
АУДИТОРИЯ (Аудитория, Вместимость)

Схема нормализации

Приведение к НФ-2:



Приведение к НФ-3:

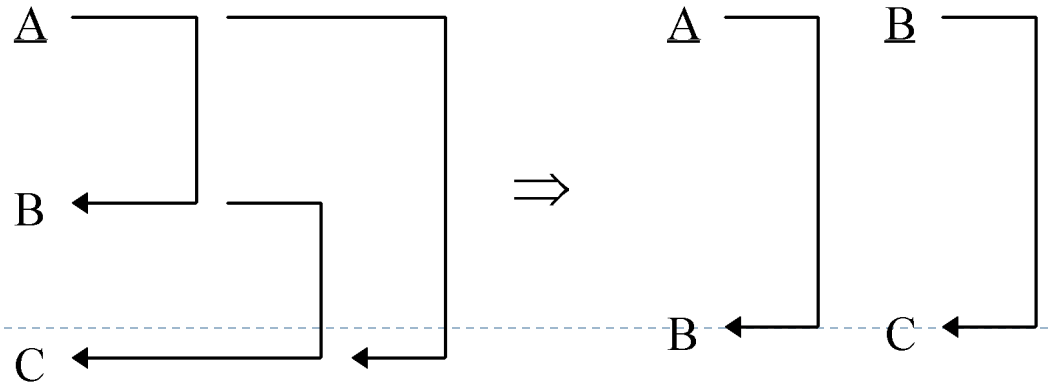


Схема нормализации

При проектировании структуры реляционной базы данных считается корректной установка, что любая БД должна находиться как минимум в НФ-3. На практике третья нормальная форма схем отношений достаточна в большинстве случаев, и приведением к третьей нормальной форме процесс проектирования реляционной базы данных обычно заканчивается

Нормальная форма Бойса-Кодда

Отношение находится в нормальной форме Бойса—Кодда, если оно находится в третьей нормальной форме и каждый Детерминант отношения является возможным ключом отношения,

Определение 6. Если в отношении существует несколько функциональных зависимостей, то каждый атрибут или набор атрибутов, от которого зависит другой атрибут, называется ***детерминантом отношения.***

Определение для НФ-3 было дано Коддом для ситуаций с упрощающим картину допущением того, что отношение имеет только один потенциальный ключ, который, естественно, и является первичным ключом. Естественно, что не все отношения могут быть уложены в данные довольно жесткие рамки. Более обобщающими являются случаи, когда в наличии имеются следующие условия:

- □ отношение имеет два (или более) потенциальных ключа;
- □ два потенциальных ключа являются составными;
- □ два потенциальных ключа перекрываются, т. е. имеют, по крайней мере, один общий атрибут.

Нормальная форма Бойса-Кодда

Отношение находится в нормальной форме Бойса—Кодда, если оно находится в третьей нормальной форме и каждый Детерминант отношения является возможным ключом отношения,

ПОСТАВКА (Индекс_поставщ, Имя_поставщ, Индекс_товара, Колич_товара).

В такой ситуации можно выделить два составных потенциальных ключа:
(Индекс поставщ, Индекс_товара);
(Имя_поставщ, Иидекс_товара).

В рассматриваемом отношении есть два атрибута **Индекс_поставщ** и **Имя_поставщ**, которые идентифицируют один и тот же экземпляр
В таком случае отношение содержит два детерминанта, но эти детерминанты не являются потенциальными ключами отношения

Нормальная форма Бойса-Кодда

ПОСТАВКА (Индекс_поставщ, Имя_поставщ, Индекс_товара, Колич_товара).

Для схемы отношения, не находящейся в НФБК, можно провести декомпозицию в схему БД в НФБК. Из исходного отношения убирается и переносится в новое отношение зависимая часть вместе с копией детерминанта.

Первый вариант: если учитывается зависимость
Индекс_поставщ → Имя_поставщ,
в результате чего имеем следующих два отношения:
ПОСТАВКА (Иидекс_поставщ, Индекс_товара, Колич_товара);
ПОСТАВЩИК (Индекс_поставщ, Имя_поставщ),

Второй вариант исходит из зависимости
Имя _поставщ → Индекс_поставщ
ПОСТАВКА (Имя_поставщ, Индекс_товара, Колич_товара);
ПОСТАВЩИК (Индекс_поставщ, Имя_поставщ).

Четвертая нормальная форма

В отношении R (A B, C) существует *многозначная зависимость (multi valid dependence, MVD)* R.A → R,B в том и только в том случае» если множество значений B, соответствующее паре значений A и C, зависит только от A и не зависит от C.
Когда мы рассматривали функциональные зависимости, то каждому значению детерминанта соответствовало только одно значение зависимого от него атрибута. При рассмотрении многозначных зависимостей мы выделяем случаи, когда одному значению некоторого атрибута соответствует устойчиво постоянное множество значений другого атрибута.

Пусть дано отношение, которое .моделирует предстоящую сдачу экзаменов на сессии. Допустим, оно имеет вид:

(НомерЗач , Группа, Дисциплина)

Перечень дисциплин, которые должен сдавать студент, однозначно определяется не его фамилией, а номером группы (то есть специальностью, на которой он учится).

Четвертая нормальная форма

(НомерЗач , Группа, Дисциплина)

В данном отношении существуют следующие две многозначные зависимости:
Группа -» Дисциплина
Группа -» НомерЗач,

В теории реляционных баз данных доказывается, что в общем случае в отношении R (A, B, C) существует многозначная зависимость R.A -» R.B в том и только в том случае, когда существует многозначная зависимость R.A -» R.C. Дальнейшая нормализация отношений, подобных нашему, основывается на теореме Фейджина.

ТЕОРЕМА ФЕЙДЖИНА
Отношение R (A, B, C) можно спроецировать без потерь в отношения R1 (A, B) и R2 (A, C) в том и только в том случае, когда существует MVD A -» B | C
(что равнозначно наличию двух зависимостей A -» B и A -» C),

Проецирование без потерь, значит исходное отношение полностью восстанавливается и без избыточности

Четвертая нормальная форма

Отношение R находится о четвертой нормальной форме в том и только и том случае, если в случае существования многозначной зависимости A *-» B* все остальные атрибуты R функционально зависят от A.

(НомерЗач , Группа, Дисциплина)

В нашем примере можно произнести декомпозицию исходного отношения в два отношения:
(НомерЗач , Группа)
(Группа, Дисциплина)

Пятая нормальная форма

переменная отношение R (X, У, ,,., Z) удовлетворяет зависимости соединения (X,Y Z) в том и только в том случае, когда R восстанавливается без потерь путем соединения своих проекций на X,Y,..., Z. Здесь X,Y,..., Z — наборы атрибутов отношения R,

Иными словами, переменная отношения *R* удовлетворяет зависимости соединения ***{**X, У, **, . . . , Z}** тогда и только тогда, когда любое допустимое значение переменной отношения *R* эквивалентно соединению её проекций по подмножествам X, У, , …, **Z** множества атрибутов.

Зависимость соединения является предельным обобщением понятий многозначной и функциональной зависимости, то есть это наиболее общая форма зависимости между атрибутами отношения.. Наличие PJ-зависимости в отношении делает его в некотором роде избыточным и затрудняет операции модификации.

Важно понимать, что зависимость соединения определяется не для конкретного значения переменной отношения в тот или иной момент времени, а по всем возможным значениям. Поэтому понятие зависимости соединения определено не для *отношения* (конкретного значения), а для *переменной отношения*.

Пятая нормальная форма

Отношение R находится *в пятой нормальной форме,* в проекционно-соединительной нормальной форме, в том и только в том случае, когда любая зависимость соединения в R следует из существования некоторого возможного ключа в R.

Пятая нормальная форма

Отношение R находится *в пятой нормальной форме* в том и только в том случае, когда любая зависимость соединения в R следует из существования некоторого возможного ключа в R.

Рассмотрим отношение R1:

R1 (Преподаватель, Кафедра, Дисциплина)

Предположим, что каждый преподаватель может работать на нескольких кафедрах и на каждой кафедре может вести несколько дисциплин. В этом случае ключом отношения является полный набор из трех атрибутов. В отношении отсутствуют многозначные зависимости, и поэтому отношение находится в НФ4.

Введем следующие обозначения наборов атрибутов:
ПК (Преподаватель, Кафедра)
ПД (Преподаватель. Дисциплина)
КД (Кафедра, Дисциплина)

Пятая нормальная форма

Допустим, что отношение R1 удовлетворяет зависимости проекции соединения (ПК, ПД, КД). Тогда отношение R1 не находится в НФ5, потому что единственным ключом его является полный набор атрибутов, а наличие зависимости PJ связано с наборами атрибутов, которые не составляют возможные ключи отношения R1. Для того чтобы привести это отношение к НФ5, его надо представить в виде трех отношений:

R2 (Преподаватель. Кафедра)
R3 (Преподаватель. Дисциплина)
R4 (Кафедра, Дисциплина)

Пятая нормальная форма

Пятая нормальная форма редко используется на практике. В большей степени она является теоретическим исследованием. Очень тяжело определить само наличие зависимостей «проекции—соединения», потому что утверждение о наличии такой зависимости делается для всех возможных состояний БД, а не только для текущего экземпляра отношения R1. Однако знание о возможном наличии подобных зависимостей, даже теоретическое, нам все же необходимо.

Лекция 8

SQL запросы

Манипулирование данными в SQL

В операции манипулирования данными входят три операции;

- операция удаления записей — ей соответствует оператор DELETE
- операция добавления или ввода новых записей — ей соответствует оператор INSERT
- и операция изменения (обновления записей) — ей соответствует оператор UPDATE,

Все операторы манипулирования данными позволяют изменить данные только в одной таблице.

INSERT

Оператор ввода данных INSERT имеет следующий синтаксис:

INSERT INTO имя_таблицы [(<список столбцов>)] VALUES (<список значений>)

Подобный синтаксис позволяет ввести только одну строку в таблицу. Задание списка столбцов необязательно тогда, когда мы вводим строку с заданием значений всех столбцов. Например, введем новую книгу в таблицу BOOKS

INSERT INTO BOOKS (ISBN,TITL,AUTOR,CQAUTOR,YEARIZD,PAGES) VALUES ("5-88782-290-2","Аппаратные средства IBM PC. Энциклопедия “Гук М“,””,2000,816)

Так как мы вводим полную строку, то мы можем не задавать список столбцов, ограничиться только заданием перечня значений

INSERT

INSERT INTO BOOKS (ISBN,TITL,AUTOR,CQAUTOR,YEARIZD,PAGES)
VALUES ("5-88782-290-2","Аппаратные средства IBM PC. Энциклопедия
“Гук М“,””,2000,816)

INSERT INTO BOOKS
VALUES ("5-88782-290-2","Аппаратные средства IBM PC. Энциклопедия
“Гук М“,””,2000,816)

неполный перечень значений

INSERT INTO BOOKS (ISBN,TITL,AUTOR,YEARIZD,PAGES)
VALUES ("5-88782-290-2","Аппаратные средства IBM PC. Энциклопедия",
“Гук М.",2000,816)

Столбцу COAUTOR будет присвоено в этом случае значение NULL.

INSERT

Оператор ввода данных позволяет ввести сразу множество строк, если их можно выбрать из некоторой другой таблицы.
Допустим, что у нас есть таблица со студентами и в ней указаны основные данные о студентах: их фамилии, адреса, домашние телефоны и даты рождения. Тогда мы можем сделать всех студентов читателями нашей библиотеки одним оператором;

```
INSERT INTO READER (NAME_READER,ADRESS, PHONE.,BIRTH_DAY)
SELECT (NAME_STUDENT,  ADRESS, PHONE, BIRTH_DAY)
FROM STUDENT
```

DELETE

Оператор удаления данных позволяет удалить одну или несколько строк из таблицы в соответствии с условиями, которые задаются для удаляемых строк.

Синтаксис оператора DELETE следующий:
DELETE FROM имя_таблицы [WHERE условия_отбора]

Если условия отбора не задаются то из таблицы удаляются все строки, но таблица остается

Например, если нам надо удалить результаты прошедшей сессии, то мы можем удалить все строки из отношения R1 командой
DELETE FROM R1

DELETE

Условия отбора в части WHERE имеют тот же вид, что и условия фильтрации в операторе SELECT. Эти условия определяют, какие строки из исходного отношения будут удалены. Например, если мы исключим студента Миронова Л. В., то мы должны написать следующую команду;

DELETE FROM R2
WHERE ФИО = ‘Миронов А. В.‘

DELETE

В части WHERE может находиться встроенный запрос.

Например, если нам надо исключить неуспевающих студентов, в условиях отбора надо найти студентов, имеющих либо две или более двоек, либо два и более несданных экзамена из числа тех, которые студент сдавал.
Надо выбрать из отношения R1 все строки с оценкой 2 или с неопределенным значением, потом надо сгруппировать полученный результат по атрибуту ФИО и, подсчитав количество строк в каждой группе, которое соответствует количеству несданных экзаменов каждым студент
количество строк не менее двух.

при выполнении операции DELETE,
включающей сложный подзапрос,
в подзапросе нельзя упоминать таблицу, из
которой удаляются строки,

```
DELETE FROM R2
WHERE R2.ФИО IN
        (SELECT R1.ФИО
           FROM RI
          WHERE Оценка = 2 OR Оценка IS NULL
          GROUP BY RI.ФИО
            HAVING COUNT(*) >= 2)
```

UPDATE

Операция обновления данных UPDATE требуется тогда, когда происходят изменения во внешнем мире и их надо адекватно отразить в базе данных

Например, студент Степанова К. Е. пересдала экзамен по дисциплине «Базы данных» с двойки на четверку. В этом случае нам надо выполнить соответствующую корректировку таблицы R1. Операция обновления имеет следующий формат;

```
UPDATE имя_таблицы
SET имя_столбца = новое_значение
[WHERE условие_отбора]
```

Если условие отбора не задается, то операция модификации будет применена ко всем строкам таблицы.

UPDATE

Например, студент Степанова К. Е. пересдала экзамен по дисциплине «Базы данных» с двойки на четверку. В этом случае нам надо выполнить соответствующую корректировку таблицы R1. Операция обновления имеет следующий формат;

```
UPDATE R1
SET R1.Оценка = 4
WHERE R1.ФИО  = "Степанова К.Е" AND R1.Дисциплина = "Базы данных"
```

UPDATE

изменение в нескольких строках

Например, если мы расширим нашу учебною базу данных еще одним отношением, которое содержит перечень курсов, на которых учатся наши студенты, то можно с помощью операции обновления промоделировать операцию перевода *групп* на следующий курс. Пусть новое отношение R4 имеет следующую схему;

R4= <Группа, Курс>

R4	
Группа	Курс
4906	3
4807	4

В этом случае перевод па следующий курс можно выполнить следующей операцией обновления:

UPDATE R4
SET R4.Курс = R4.Курс + 1

UPDATE

Операция модификации, так же как и операция удаления, может использовать сложные подзапросы. Расширим нашу базу еще одним отношением, которое будет содержать перечень студентов, получающих стипендию с указанием надбавки, которую они получают за отличную учебу. Исходно там могут находиться все студенты с указанием неопределенного размера стипендии. По мере анализа отношения R1 мы можем постепенно заменять неопределенные значения на конкретные размеры стипендии

R5		
ФИО	Группа	Стипендия
Петров	4906	NULL
Сидоров	4906	NULL
Миронов	4906	NULL
Крылова	4906	NULL
Владимиров	4906	NULL
Трофимов	4807	NULL
Иванова	4807	NULL
Уткина	4807	NULL

UPDATE

Будем считать наличие трех пятерок по сессии признаком повышенной стипендии, + 50% к основной, наличие двух пятерок из сданных экзаменов и отсутствие двоек и троек на сданных экзаменах - признаком повышения стипендии на25%, наличие хотя бы одной тройки среди сданных экзаменов - признаком снятия или отсутствия стипендии вообще, то есть -100% надбавки. При отсутствии троек на сданных экзаменах назначим обычную стипендию с надбавкой 0%

R5		
ФИО	Группа	Стипендия
Петров	4906	NULL
Сидоров	4906	NULL
Миронов	4906	NULL
Крылова	4906	NULL
Владимиров	4906	NULL
Трофимов	4807	NULL
Иванова	4807	NULL
Уткина	4807	NULL

UPDATE

Для сессии в которой 3 экзамена.

- наличие трех пятерок по сессии признаком повышенной стипендии, + 50% к основной
- наличие двух пятерок из сданных экзаменов и отсутствие двоек и троек на сданных экзаменах - признаком повышения стипендии на25%,
- наличие хотя бы одной тройки среди сданных экзаменов - признаком снятия или отсутствия стипендии вообще, то есть -100% надбавки.
- При отсутствии троек назначим обычную стипендию с надбавкой 0%

Назначение повышенной стипендии:

```
UPDATE R5
SET R5.Стипендия = 50%
WHERE R5.ФИО IN
   (SELECT R1.ФИО
   FROM R1
  WHERE R1.Оценка = 5
   GROUP BY  R1.ФИО
   HAVING COUNT(*) =3 )
```

Назначение стипендии с надбавкой 25%:

```
UPDATE R5
SET R5.Стипендия = 25%
WHERE R5.ФИО IN
   (SELECT R1.ФИО
    FROM R1
     WHERE R1.Оценка=5 AND R1.ФИО NOT
IN
          (SELECT A.ФИО
           FROM R1  A
             WHERE A.Оценка <= 3
           OR A.Оценка IS NULL)
       GROUP BY  R1.ФИО
       HAVING COUNT(*) >=2 )
```

UPDATE

Для сессии в которой 3 экзамена.

- наличие трех пятерок по сессии признаком повышенной стипендии, + 50% к основной
- наличие двух пятерок из сданных экзаменов и отсутствие двоек и троек на сданных экзаменах - признаком повышения стипендии на25%,
- наличие хотя бы одной тройки среди сданных экзаменов - признаком снятия или отсутствия стипендии вообще, то есть -100% надбавки.
- При отсутствии троек назначим обычную стипендию с надбавкой 0%

Назначение обычной стипендии:

```
UPDATE R5
SET R5.Стипендия = 0%
WHERE R5.ФИО IN
   (SELECT R1.ФИО
   FROM R1
  WHERE R1.Оценка >=4 AND R1.ФИО NOT
IN
        (SELECT A.ФИО
        FROM R1  A
          WHERE A.Оценка <= 3
         OR A.Оценка IS NULL)
```

Снятие стипендии:

```
UPDATE R5
SET R5.Стипендия =  -100%
WHERE R5.ФИО IN
     (SELECT R1.ФИО
      FROM R1
      WHERE R1.Оценка<=3
     OR R1.ОЦЕНКА IS NULL )
```

UPDATE

Назначение повышенной стипендии для сессии из трех экзаменов:

```
UPDATE R5
SET R5.Стипендия = 50%
WHERE R5.ФИО IN
   (SELECT R1.ФИО
   FROM R1
   WHERE R1.Оценка = 5
   GROUP BY  R1.ФИО
   HAVING COUNT(*) =3 )
```

Теперь составим запрос на обновление для назначения повышенной стипендии при любом количестве сданных экзаменов.

В конечном счете нам все равно надо знать, сколько экзаменов должен сдавать каждый конкретный студент, поэтому сначала сосчитаем количество экзаменов, которые должна сдавать группа, в которой учится этот студент

UPDATE

В конечном счете нам все равно надо знать, сколько экзаменов должен сдавать каждый конкретный студент, поэтому сначала сосчитаем количество экзаменов, которые должна сдавать группа, в которой учится этот студент

```
SELECT R3.Группа,  Число_экзаменов = COUNT(*)
FROM R3
GROUP BY R3.Группа
```

запрос, в котором мы определяем для каждого студента количество экзаменов. Этот запрос мы должны строить по схеме встроенного запроса;

```
SELECT COUNT(*)
FROM R3
WHERE R2.Группа = R3,Группа
GROUP BY R3.Группа
```

UPDATE

Нам надо объединить отношения R1 и R2 по атрибуту ФИО, нам надо знать группу, в которой учится каждый студент, далее надо выбрать все строки с оценкой 5 и сгруппировать их по фамилии студента, сосчитав количество строк в каждой группе, а выбирать мы будем те группы, в которых число строк в группе равно числу строк во встроенном запросе, рассмотренном ранее, при условии равенства количества строк в группе результату подзапроса, который выводит только одно число.

```
SELECT R1.ФИО
FROM R1.R2
WHERE R1. ФИО = R2.ФИО AND R1.Оценка = 5
GROUP BY R1.ФИО
HAVING COUNT(*) = (SELECT COUNT(*)
                   FROM R3
                   WHERE R2.Группа =  R3.Группа
                   GROUP BY R3.Группа)
```

UPDATE

надо заменить старый вложенный запрос, определявший отличников, получивших три пятерки на сессии, на новый универсальный запрос:

```
UPDATE R5
SET R5.Стипендия = 50%
WHERE R5.ФИО IN
    SELECT R1.ФИО
   FROM R1.R2
    WHERE R1. ФИО = R2.ФИО AND R1.Оценка = 5
    GROUP BY R1.ФИО
    HAVING COUNT(*) = (SELECT COUNT(*)
                                  FROM R3
                                 WHERE R2.Группа =  R3.Группа
                                 GROUP BY R3.Группа))
```