II. РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Впервые термин "реляционная модель данных" появился в статье сотрудника фирмы IBM д-ра Кодда (Codd E.F., A Relational Model of Data for Large Shared Data Banks. CACM 13: 6, June 1970). Будучи математиком по образованию, Кодд предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как ***отношение*** **– relation** (англ.).

2.1. Определение реляционной модели

Реляционная модель данных (РМД) некоторой предметной области представляет собой набор отношений, изменяющихся во времени. При создании информационной системы совокупность отношений позволяет хранить данные об объектах предметной области и моделировать связи между ними. Элементы РМД и формы их представления приведены в табл. 2.1.

Таблица 2.1

Элементы реляционной модели

Реляционный термин	**Соответствующий «табличный» термин**
База данных	Набор таблиц
Схема базы данных	Набор заголовков таблиц
Отношение	Таблица
Заголовок отношения	Заголовок таблицы
Тело отношения	Тело таблицы
Атрибут отношения	Наименование столбца таблицы
Схема отношения	Строка заголовков столбцов таблицы (заголовок таблицы)
Кортеж	Строка таблицы
Сущность	Описание свойств объекта
Атрибут	Заголовок столбца таблицы
Домен	Множество допустимых значений атрибута
Домены и типы данных	Типы данных в ячейках таблицы
Значение атрибута	Значение поля в записи
Первичный ключ	Один или несколько атрибутов
Тип данных	Тип значений элементов таблицы
Степень (-арность) отношения	Количество столбцов таблицы
Мощность отношения	Количество строк таблицы

Основными понятиями, с помощью которых определяется реляционная модель, являются следующие:

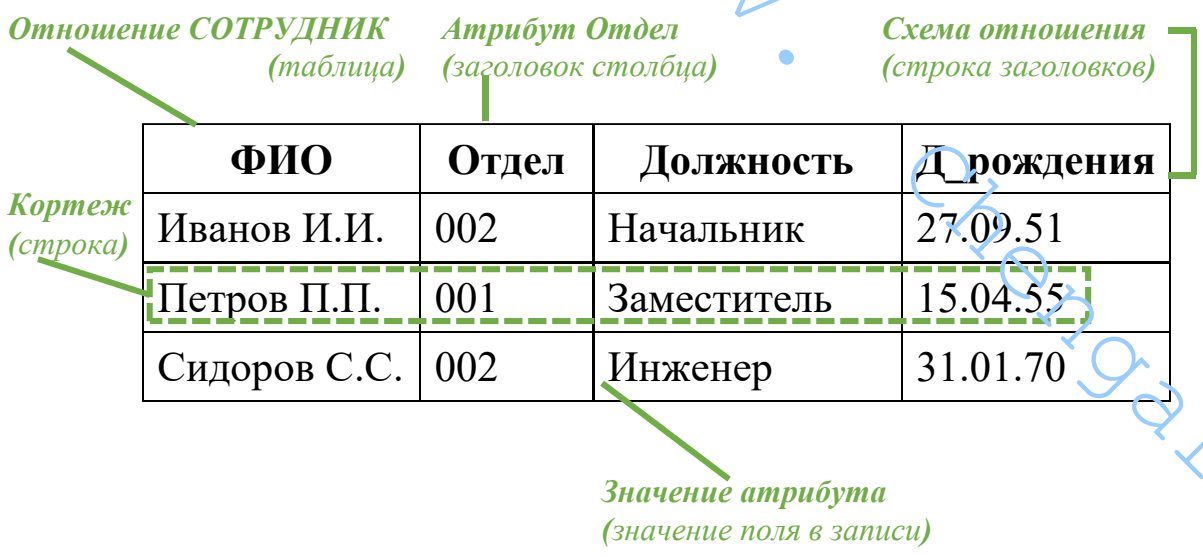
1. ***Реляционная БД*** – набор нормализованных отношений;
2. ***Отношение*** – файл, плоская таблица, состоящая из столбцов и строк; таблица, в которой каждое поле является атомарным; это множество кортежей, соответствующих одной схеме отношения.
3. ***Сущность*** есть объект любой природы, данные о котором хранятся в базе данных. Данные о сущности хранятся в отношении.
4. ***Кортеж*** – это множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения, запись, строка таблицы;
5. ***Атрибуты*** – поименованные поля, столбцы таблицы

Атрибуты представляют собой свойства, характеризующие сущность. В структуре таблицы каждый атрибут именуется и ему соответствует заголовок некоторого столбца таблицы.

Математически отношение можно описать следующим образом. Пусть даны n множеств D1, D2, D3,..., Dn, тогда отношение R есть множество упорядоченных ***кортежей*** <dl, d2, d3 ,..., dn>, где dk ∈ Dk, dk — ***атрибут***, а Dk - ***домен*** отношения R.

На рис. 2.1 приведен пример представления отношения СОТРУДНИК.

В общем случае порядок кортежей в отношении, как и в любом множестве, не определен. Однако в реляционных СУБД для удобства кортежи все же упорядочивают. Чаще всего для этого выбирают некоторый атрибут, по которому система автоматически сортирует кортежи по возрастанию или убыванию. Если пользователь не назначает атрибута упорядочения, система автоматически присваивает номер кортежам в порядке их ввода.



ФИО	Отдел	Должность	Д_рождения
Иванов И.И.	002	Начальник	27.09.51
Петров П.П.	001	Заместитель	15.04.55
Сидоров С.С.	002	Инженер	31.01.70

Рисунок 2.1 – Представление отношения СОТРУДНИК

Формально, если переставить атрибуты в отношении, то получается новое отношение. Однако в реляционных БД перестановка атрибутов не приводит к образованию нового отношения.

6. ***Домен*** – совокупность допустимых значений, из которой берется значение соответствующего атрибута определенного отношения.

Наиболее правильной интуитивной трактовкой понятия домена является понимание его как допустимого потенциального множества значений данного типа.

Следует отметить семантическую нагрузку понятия домена: данные считаются сравнимыми (тета-сравнимыми) только в том случае, когда они относятся к одному домену, а не к одному базовому типу.

Отношение СОТРУДНИК включает 4 домена. *Домен 1* содержит фамилии всех сотрудников, *домен 2* — номера всех отделов фирмы, *домен 3* — названия всех должностей, *домен 4* — даты рождения всех сотрудников. Каждый домен образует значения одного типа данных, например, числовые или символьные.

Отношение СОТРУДНИК содержит 3 кортежа. Кортеж рассматриваемого отношения состоит из 4 элементов, каждый из которых выбирается из соответствующего домена. Каждому кортежу соответствует строка таблицы (рис. 2.1).

7. ***Универсум*** – совокупность значений всех полей или совокупность доменов.

8. ***Кардинальность*** - количество строк в таблице.

9. ***Схема отношения*** – это именованное множество пар ***{имя атрибута, имя домена}***, т.е. представляет собой список имен атрибутов.

Например, для приведенного примера схема отношения имеет вид СОТРУДНИК(ФИО, Отдел, Должность, Д_Рождения). Множество собственно кортежей отношения часто называют ***содержимым {телом) отношения***.

10. ***Степень схемы отношения*** - количество атрибутов; мощность этого множества (множество пар ***{имя атрибута, имя домена}***).

11. ***Схема реляционной БД*** – совокупность схем отношений.

12. ***Первичным ключом {ключом отношения, ключевым атрибутом)*** называется атрибут отношения, однозначно идентифицирующий каждый из его кортежей.

Правило целостности объектов утверждает, что первичный ключ не может быть полностью или частично пустым.

Например, в отношении СОТРУДНИК (ФИО, Отдел, Должность, Д_Рождения) ключевым является атрибут «ФИО».

Ключ может быть ***составным {сложным)***, то есть состоять из нескольких атрибутов.

Каждое отношение обязательно имеет комбинацию атрибутов, которая может служить ключом. Ее существование гарантируется тем, что отношение – это множество, которое не содержит одинаковых элементов – кортежей. То

есть в отношении нет повторяющихся кортежей, а это значит, что по крайней мере вся совокупность атрибутов обладает свойством однозначной идентификации кортежей отношения. Во многих СУБД допускается создавать отношения, не определяя ключи.

Возможны случаи, когда отношение имеет несколько комбинаций атрибутов, каждая из которых однозначно определяет все кортежи отношения. Все эти комбинации атрибутов являются ***возможными ключами отношения***. Любой из возможных ключей может быть выбран как *первичный*.

Если выбранный первичный ключ состоит из минимально необходимого набора атрибутов, говорят, что он является ***не избыточным***.

Ключи обычно используют для достижения следующих целей:

1) исключения дублирования значений в ключевых атрибутах (остальные атрибуты в расчет не принимаются);

2) упорядочения кортежей. Возможно упорядочение по возрастанию или убыванию значений всех ключевых атрибутов, а также смешанное упорядочение (по одним — возрастание, а по другим — убывание);

3) ускорения работы к кортежами отношения (подраздел 2.2);

4) организации связывания таблиц (подраздел 2.3).

Пусть в отношении R1 имеется *не ключевой* атрибут A, значения которого являются значениями ключевого атрибута B другого отношения R2. Тогда говорят, что атрибут A отношения R1 есть ***внешний ключ***.

С помощью внешних ключей устанавливаются связи между отношениями. Например, имеются два отношения СТУДЕНТ(ФИО, Группа, Специальность) и ПРЕДМЕТ (Назв.Пр, Часы), которые связаны отношением СТУДЕНТ_ПРЕДМЕТ(ФИО, Назв.Пр, Оценка) (рис. 2.2). В связующем отношении атрибуты ФИО и Назв.Пр образуют составной ключ. Эти атрибуты представляют собой внешние ключи, являющиеся первичными ключами других отношений.

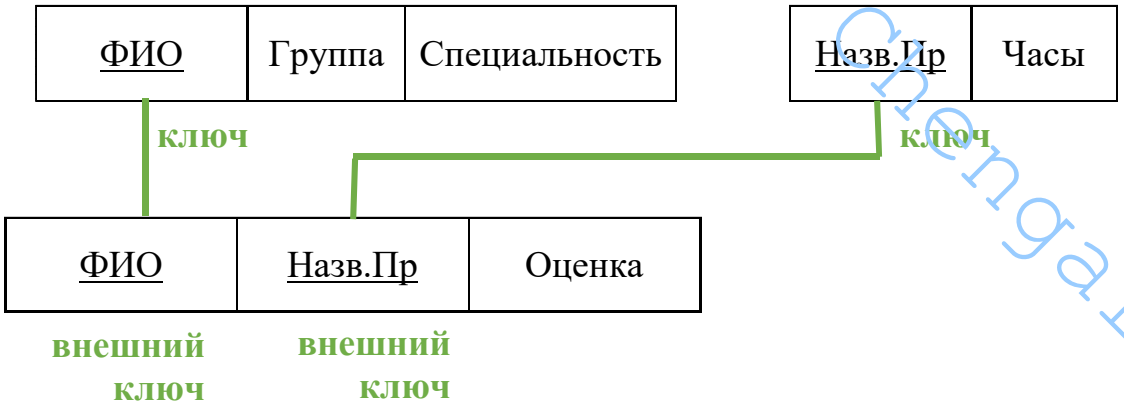


Рисунок 2.2 – Связь отношений

Реляционная модель накладывает на внешние ключи ограничение для обеспечения целостности данных, называемое ссылочной целостностью. Это

означает, что каждому значению внешнего ключа должны соответствовать строки в связываемых отношениях.

Поскольку не всякой таблице можно поставить в соответствие отношение, приведем условия, выполнение которых позволяет таблицу считать отношением.

1. Все строки таблицы должны быть уникальны, то есть не может быть строк с одинаковыми первичными ключами.
2. Имена столбцов таблицы должны быть различны, а значения их простыми, то есть недопустима группа значений в одном столбце одной строки.
3. Все строки одной таблицы должны иметь одну структуру, соответствующую именам и типам столбцов.
4. Порядок размещения строк в таблице может быть произвольным.

Наиболее часто таблица с отношением размещается в отдельном файле. В некоторых СУБД одна отдельная таблица (отношение) считается базой данных. В других СУБД база данных может содержать несколько таблиц.

В общем случае можно считать, что БД включает одну или несколько таблиц, объединенных смысловым содержанием, а также процедурами контроля целостности и обработки информации в интересах решения некоторой прикладной задачи. Например, при использовании СУБД Microsoft Access в файле БД наряду с таблицами хранятся и другие объекты базы: запросы, отчеты, формы, макросы и модули.

Таблица данных обычно хранится на магнитном диске в отдельном файле операционной системы, поэтому по ее именованию могут существовать ограничения. Имена полей хранятся внутри таблиц. Правила их формирования определяются СУБД, которые, как правило, на длину полей и используемый алфавит серьезных ограничений не накладывают.

Если задаваемое таблицей отношение имеет ключ, то считается, что таблица тоже имеет ключ, и ее называют ***ключевой*** или ***таблицей с ключевыми полями***.

У большинства СУБД файл таблицы включает управляющую часть (описание типов полей, имена полей и другая информация) и область размещения записей.

К отношениям можно применять систему операций, позволяющую получать одни отношения из других. Например, результатом запроса к реляционной БД может быть новое отношение, вычисленное на основе имеющихся отношений. Поэтому можно разделить обрабатываемые данные на хранимую и вычисляемую части.

Основной единицей обработки данных в реляционных БД является отношение, а не отдельные его кортежи (записи).

2.2. Индексирование

Как отмечалось выше, определение ключа для таблицы означает автоматическую сортировку записей, контроль отсутствия повторений

значений в ключевых полях записей и повышение скорости выполнения операций поиска в таблице. Для реализации этих функций в СУБД применяют *индексирование*.

Термин «индекс» тесно связан с понятием «ключ», хотя между ними есть и некоторое отличие.

Под ***индексом*** понимают средство ускорения операции поиска записей в таблице, а следовательно, и других операций, использующих поиск: извлечение, модификация, сортировка и т. д. Таблицу, для которой используется индекс, называют ***индексированной***.

Индекс выполняет роль *оглавления* таблицы, просмотр которого предшествует обращению к записям таблицы. В некоторых системах, например, Paradox, индексы хранятся в индексных файлах, хранимых отдельно от табличных файлов.

Варианты решения проблемы организации физического доступа к информации зависят в основном от следующих факторов:

– вида содержимого в поле ключа записей индексного файла;
– типа используемых ссылок (указателей) на запись основной таблицы;
– метода поиска нужных записей.

В поле *ключа индексного файла* можно хранить значения ключевых нолей индексируемой таблицы либо свертку ключа (так называемый хеш-код). Преимущество хранения хеш-кода вместо значения состоит в том, что длина свертки независимо от длины исходного значения ключевого поля всегда имеет некоторую постоянную и достаточно малую величину (например, 4 байта), что существенно снижает время поисковых операций. Недостатком хеширования является необходимость выполнения операции свертки (требует определенного времени), а также борьба с возникновением коллизий (свертка различных значений может дать одинаковый хеш-код).

Для организации *ссылки* на запись таблицы могут использоваться три типа адресов: абсолютный (действительный), относительный и символический (идентификатор).

На практике чаще всего используются *два метода поиска*: последовательный и бинарный (основан на делении интервала поиска пополам).

Проиллюстрируем организацию индексирования таблиц двумя схемами: одноуровневой и двухуровневой. При этом примем ряд предположений, обычно выполняемых в современных вычислительных системах. Пусть ОС поддерживает прямую организацию данных на магнитных дисках, основные таблицы и индексные файлы хранятся в отдельных файлах. Информация файлов хранится в виде совокупности блоков фиксированного размера, например целого числа кластеров.

При ***одноуровневой схеме*** в индексном файле хранятся короткие записи, имеющие два поля: поле содержимого старшего ключа (хеш-кода ключа) адресуемого блока и поле адреса начала этого блока (рис. 2.3).

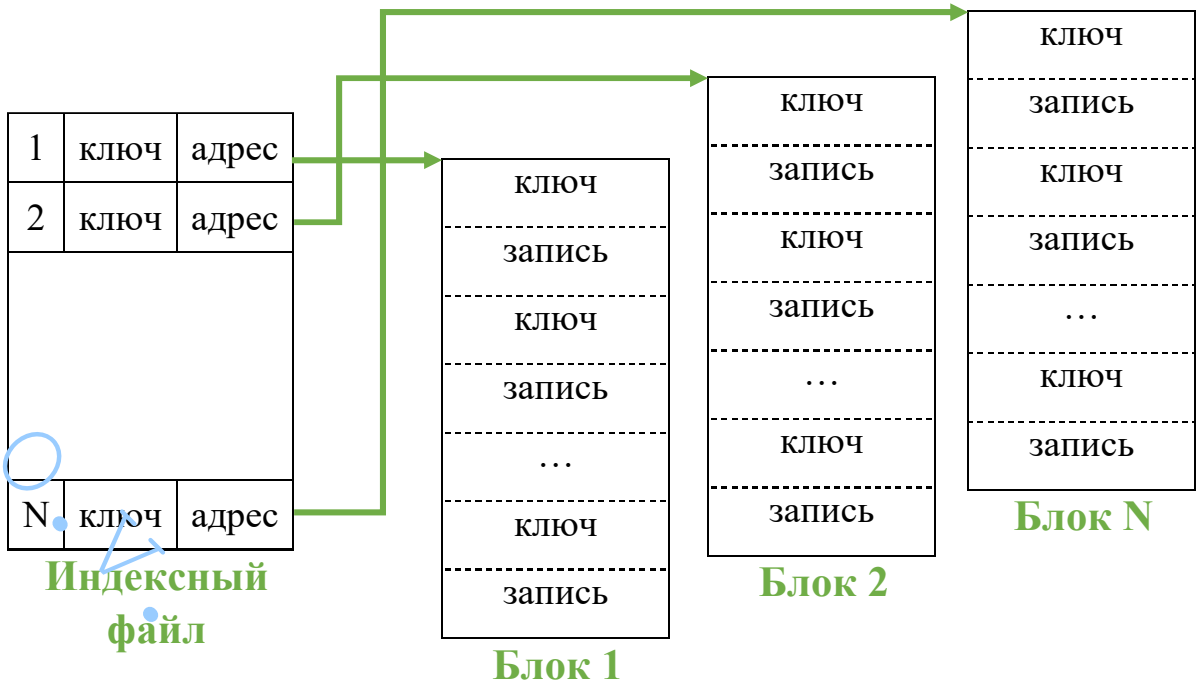


Рисунок 2.3 – Одноуровневая схема индексации

В каждом блоке записи располагаются в порядке возрастания значения ключа или свертки. Старшим ключом каждого блока является ключ его последней записи.

Если в индексном файле хранятся хеш-коды ключевых полей индексированной таблицы, то алгоритм поиска нужной записи (с указанным ключом) в таблице включает в себя следующие три этапа.

1. Образование свертки значения ключевого поля искомой записи.
2. Поиск в индексном файле записи о блоке, значение первого поля которого больше полученной свертки (это гарантирует нахождение искомой свертки в этом блоке).
3. Последовательный просмотр записей блока до совпадения сверток искомой записи и записи блока файла. В случае коллизий сверток ищется запись, значение ключа которой совпадает со значением ключа искомой записи.

Основным *недостатком* одноуровневой схемы является то, что ключи (свертки) записей хранятся вместе с записями. Это приводит к увеличению времени поиска записей из-за большой длины просмотра (значения данных в записях приходится пропускать).

Двухуровневая схема в ряде случаев оказывается более рациональной, в ней ключи (свертки) записей отделены от содержимого записей (рис. 2.4).

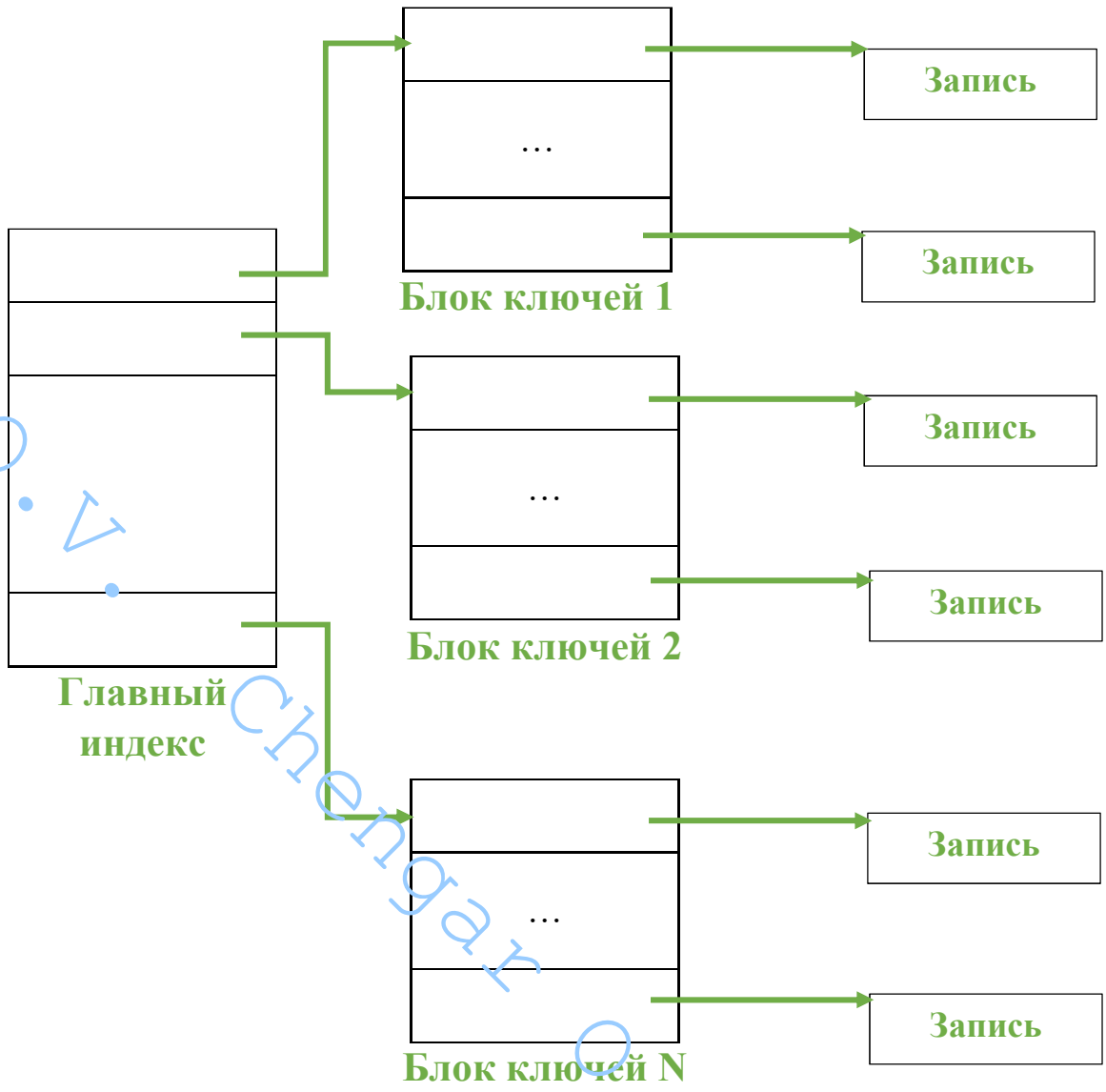


Рисунок2.4 – Двухуровневая схема индексации

В этой схеме индекс основной таблицы распределен по совокупности файлов: одному файлу главного индекса и множеству файлов с блоками ключей.

На практике для создания индекса для некоторой таблицы БД пользователь указывает поле таблицы, которое требует индексации. Ключевые поля таблицы во многих СУБД как правило индексируются автоматически. Индексные файлы, создаваемые по ключевым полям таблицы, часто называются *файлами первичных индексов*.

Индексы, создаваемые пользователем для не ключевых полей, иногда называют ***вторичными (пользовательскими) индексами***. Введение таких индексов не изменяет физического расположения записей таблицы, но влияет на последовательность просмотра записей. Индексные файлы, создаваемые для поддержания вторичных индексов таблицы, обычно называются *файлами вторичных индексов*.

Связь вторичного индекса с элементами данных базы может быть установлена различными способами. Один из них — использование

вторичного индекса как входа для получения первичного ключа, по которому затем с использованием первичного индекса производится поиск необходимых записей (рис. 2.5).

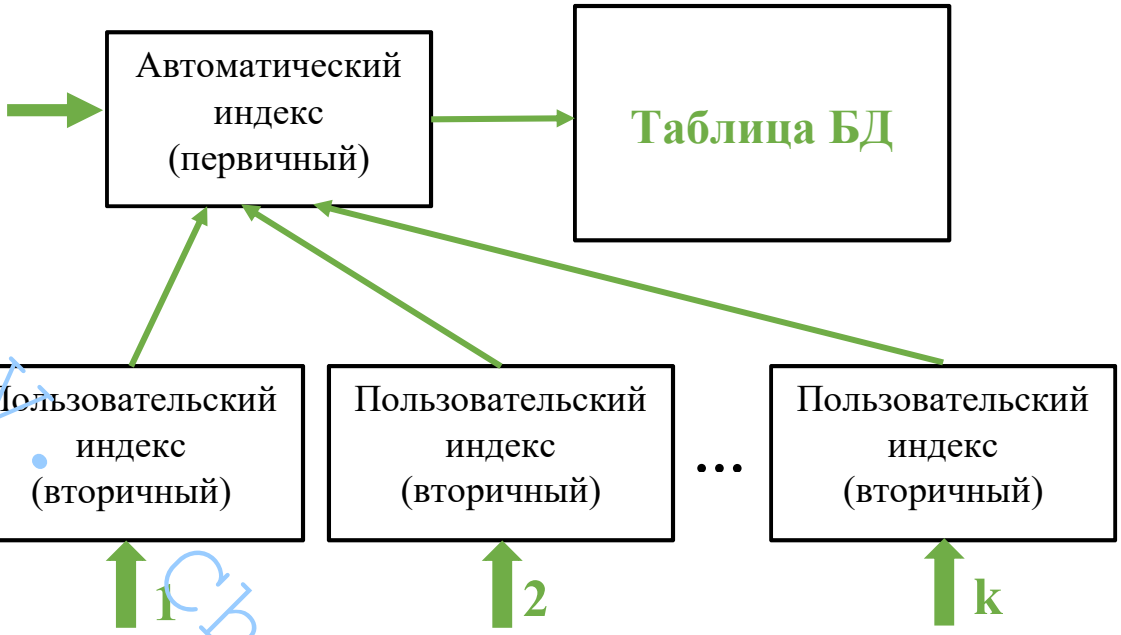


Рисунок 2.5 – Способ использования вторичных индексов

Некоторыми СУБД, например Access, деление индексов на первичные и вторичные не производится. В этом случае используются автоматически создаваемые индексы и индексы, определяемые пользователем по любому из не ключевых полей.

Главная причина повышения скорости выполнения различных операций в индексированных таблицах состоит в том, что основная часть работы производится с небольшими индексными файлами, а не с самими таблицами.

Наибольший эффект повышения производительности работы с индексированными таблицами достигается для значительных но объему таблиц. Индексирование требует небольшого дополнительного места на диске п незначительных затрат процессора па изменение индексов в процессе работы.

Индексы в общем случае могут изменяться перед выполнением запросов к БД, после выполнения запросов к БД, по специальным командам пользователя или программным вызовам приложений.

2.3. Связывание таблиц

При проектировании реальных БД информацию обычно размещают в нескольких таблицах. Таблицы при этом связаны семантикой информации. В реляционных СУБД для указания связей таблиц производят операцию их *связывания*.

Укажем выигрыш, обеспечиваемый в результате связывания таблиц. Многие СУБД при связывании таблиц автоматически выполняют контроль

целостности вводимых в базу данных в соответствии с установленными связями. В конечном итоге это *повышает достоверность* хранимой в БД информации.

Кроме того, установление связи между таблицами *облегчает доступ* к данным. Связывание таблиц при выполнении таких операций, как поиск, просмотр, редактирование, выборка и подготовка отчетов, обычно обеспечивает возможность обращения к произвольным полям связанных записей. Это уменьшает количество явных обращений к таблицам данных и число манипуляций в каждой из них.

Основные ВИДЫ связи таблиц

Между таблицами могут устанавливаться бинарные (между двумя таблицами), тернарные (между тремя таблицами) и, в общем случае, n-арные связи. Рассмотрим наиболее часто встречающиеся *бинарные связи*.

При связывании двух таблиц выделяют основную и дополнительную (подчиненную) таблицы. Логическое связывание таблиц производится с помощью **ключа связи**.

Ключ связи, по аналогии с обычным ключом таблицы, состоит из одного или нескольких полей, которые в данном случае называют ***полями связи*** (ПС).

Суть связывания состоит в установлении соответствия полей связи основной и дополнительной таблиц. Поля связи основной таблицы могут быть обычными и ключевыми. В качестве полей связи подчиненной таблицы чаще всего используют ключевые поля.

В зависимости от того, как определены поля связи основной и дополнительной таблиц (как соотносятся ключевые поля с полями связи), между двумя таблицами в общем случае могут устанавливаться следующие четыре основных вида связи (табл. 2.2):

- один — один (1:1);
- один — много (1:M);
- много — один (M: 1);
- много — много (M:M или M:N).

Таблица 3.2

Характеристика видов связей таблиц

Характеристика полей связи по видам	1:1	1 :M	M:1	M:M
Поля связи основной таблицы	являются ключом	являются ключом	не являются ключом	не являются ключом
Поля связи дополнительной таблицы	являются ключом	не являются ключом	являются ключом	не являются ключом

Дадим характеристику названным видам связи между двумя таблицами и приведем примеры их использования.

Пример. Дана совокупность информационных объектов, отражающих учебный процесс в вузе:

- СТУДЕНТ (Номер, Фамилия, Имя, Отчество, Пол, Дата рождения, Группа)
- СЕССИЯ (Номер, Оценка1, Оценка2, Оценка3, Оценка4, Результат)
- СТИПЕНДИЯ (Результат, Процент)
- ПРЕПОДАВАТЕЛЬ (Код преподавателя, Фамилия, Имя, Отчество).

Связь вида 1:1

Связь вида 1:1 образуется в случае, когда все поля связи основной и дополнительной таблиц являются ключевыми. Поскольку значения в ключевых полях обеих таблиц не повторяются, обеспечивается взаимно-однозначное соответствие записей из этих таблиц. Сами таблицы, по сути, здесь становятся равноправными.

Связь один к одному (1:1) предполагает, что в каждый момент времени одному экземпляру информационного объекта А соответствует не более одного экземпляра информационного объекта В и наоборот (рис. 2.6).

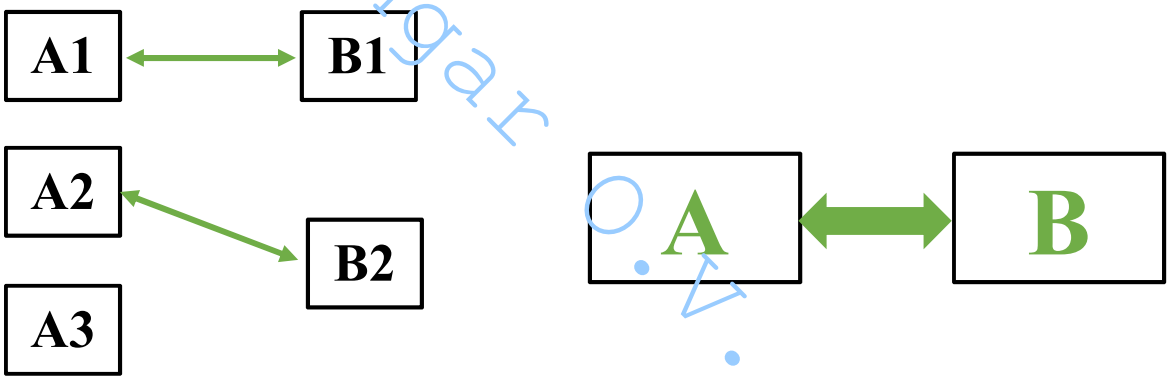


Рисунок 2.6 – Графическое изображение реального отношения (1:1)

Примером связи 1:1 может служить связь между информационными объектами СТУДЕНТ и СЕССИЯ: СТУДЕНТ ↔ СЕССИЯ. Каждый студент имеет определенный набор экзаменационных оценок в сессию.

Связь вида 1:М (М:1)

Наиболее широко используется связь 1:М, при которой одна запись главной таблицы оказывается связанной с несколькими записями дополнительной, или подчиненной таблицы.

Причем, вид связи (1:М или М:1) зависит от того, какая таблица является главной, а какая дополнительной (рис.2.7).

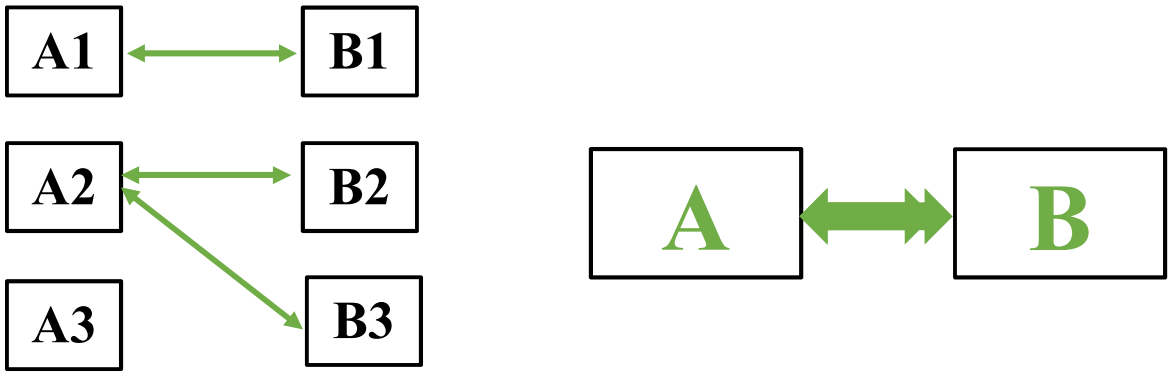


Рисунок 2.7 – Графическое изображение реального отношения (1:M)

Примером связи 1:M служит связь между информационными объектами СТИПЕНДИЯ и СЕССИЯ: СТИПЕНДИЯ ↔ СЕССИЯ. Установленный размер стипендии по результатам сдачи сессии может повторяться многократно для различных студентов.

Связь вида M:M

Самый общий вид связи M:M возникает в случаях, когда нескольким записям основной таблицы соответствует несколько записей дополнительной таблицы.

Связь многие ко многим (M:M) предполагает, что в каждый момент времени одному экземпляру информационного объекта A соответствует 0, 1 или более экземпляров объекта B и наоборот(рис.2.8).

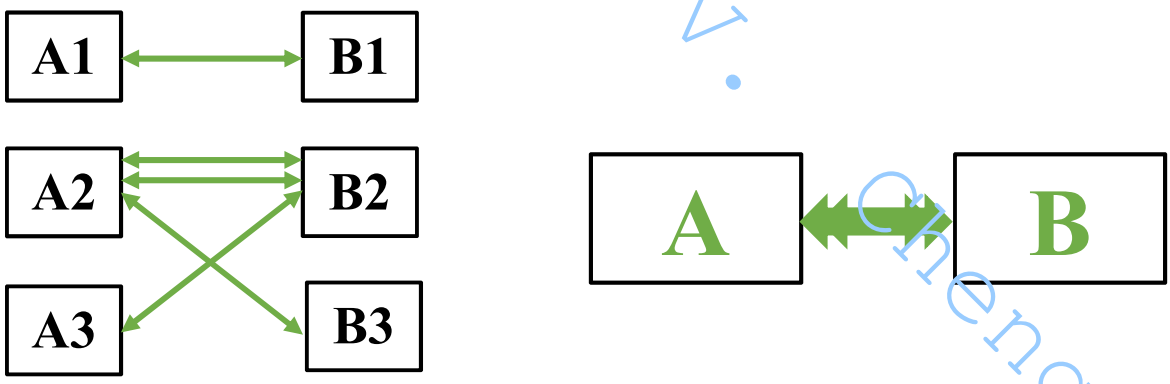


Рисунок 2.8 – Графическое изображение реального отношения (M:M)

Примером данного отношения служит связь между информационными объектами СТУДЕНТ и ПРЕПОДАВАТЕЛЬ: СТУДЕНТ ↔ ПРЕПОДАВАТЕЛЬ. Один студент обучается у многих преподавателей, один преподаватель обучает многих студентов

2.4. Контроль целостности связей

Из перечисленных видов связи чаще используется связь вида 1:M. Связь вида 1:1 можно считать частным случаем связи 1:M, когда одной записи главной таблицы соответствует одна запись вспомогательной таблицы. Связь M:1, по сути, является «зеркальным отображением» связи 1:M. Оставшийся вид связи M:M характеризуется как слабый вид связи или даже как отсутствие связи. Поэтому в дальнейшем рассматривается связь вида 1:M.

Контроль целостности связей означает анализ содержимого двух таблиц на соблюдение следующих правил:

1. Каждой записи основной таблицы соответствует NULL или более записей дополнительной таблицы.
2. В дополнительной таблице нет записей, которые не имеют родительских записей в основной таблице.
3. Каждая запись дополнительной таблицы имеет только одну родительскую запись основной таблицы.

Контроль целостности осуществляется при вводе новых записей, модификации и удалении записей.

При вводе новых записей данные сначала вводят в основную таблицу, а потом – в дополнительную.

В процессе заполнения основной таблицы контроль значений полей связи ведется как контроль обычного ключа (на совпадение со значениями полей связи основной таблицы). Ввод значения, которого нет в основной таблице, должен блокироваться.

При модификации записи используются следующие правила:

- Запись дополнительной таблицы может сменить родителя, но остаться без него не должна.
- Новое значение поля связи дополнительной таблицы должно совпадать с соответствующим значением какой-либо записи в основной таблице.

Правила редактирования поля связи основной таблицы:

- Редактируются записи, у которых нет подчиненных записей. Если есть подчиненные записи, то блокировка модификации полей связи (запрет).
- Изменение в полях связи основной таблицы можно, при мгновенной передаче во все поля связи всех записей дополнительной таблицы (каскадирование).

Удаление записи

Удаление записи в подчиненной таблице происходит бесконтрольно. Удаление записи основной таблицы необходимо подчинить следующим правилам:

1. Можно удалять запись, не имеющую подчиненных записей.
2. Запретить удаление записей при наличии подзаписей или удаление ее со всеми подчиненными записями (каскадное удаление).