

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное автономное образовательное учрежде-  
ние высшего образования**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**  
**АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ**

---

**СИСТЕМЫ БАЗ ДАННЫХ**  
**(Часть 2)**

**Учебно-методическое пособие**

**ГУАП**  
**Санкт-Петербург**

**2024**

Рецензенты:

Доцент, кандидат технических наук, доцент кафедры компьютерных технологий и программной инженерии ГУАП *С. В. Щекин*;

Доцент, кандидат физ.-мат. наук, доцент кафедры бизнес-информатики и операционного менеджмента НИУ ВШЭ СПб *М. В. Фаттахова*

**Богословская Н. В., Бржезовский А. В.**

Системы баз данных (Часть 2): учебно-методическое пособие / Н. В. Богословская, А. В. Бржезовский. – СПб: ГУАП 2024. – 35 с.

Во второй части учебно-методического пособия рассмотрены вопросы повышения производительности запросов за счет индексации данных в БД, анализа использования индексов в планах выполнения запросов, а также использования указаний запросов (hints). Приведены основные сведения для организации транзакций в БД, описаны проблемы, возникающие при многопользовательском доступе к данным и механизмы их устранения на основе применения блокировок. Описаны уровни изоляции транзакций, которые могут выбираться для выполнения запросов в зависимости от требований, предъявляемых задачами, решаемыми информационной системой.

Материалы пособия необходимы для изучения дисциплины «Методы и средства проектирования информационных систем и технологий» во втором семестре, а также при курсовом проектировании по данной дисциплине.

Примеры синтаксических конструкций, представленные в пособии, реализованы на диалекте Transact-SQL для Microsoft SQL Server 2012..2022, они могут иметь незначительные отличия от других диалектов языка и для других версий Microsoft SQL Server.

Учебно-методическое пособие предназначено для студентов, обучающихся по направлению 09.03.02 — Информационные системы и технологии.

## 9. ИНДЕКСАЦИЯ ДАННЫХ

### 9.1 Принципы индексации данных

В современных БД количество записей в таблицах может исчисляться миллионами (например, БД, связанные с населением, БД, используемые в службах техподдержки и контакт-центрах крупных корпораций), как следствие, сложные запросы в таких БД могут выполняться неприемлемо долго. Одним из способов решения этой проблемы является индексация данных.

В основе принципов индексации лежит тот факт, что в отсортированных массивах данных возможен двоичный поиск, который работает существенно быстрее, чем линейный. Предположим, что в таблице *Студент* 1024 записи, тогда линейный поиск заданного студента потребует в среднем выполнения 512 операций сравнения (сложность линейного поиска  $N/2$ ). Если данные отсортированы, возможен двоичный поиск: берется средний элемент, сравнивается с ключом поиска, если ключ меньше — процедура поиска продолжается в верхней половине массива, иначе — в нижней. В случае двоичного поиска для нахождения записи о заданном студенте по уникальному ключу потребуется всего 10 (при  $N=1024$ ) операций сравнения (сложность двоичного поиска  $\log_2(N)$ ).

На практике в СУБД, чаще всего, реализуют индексы в форме сбалансированных деревьев (*B-Tree*), а основной эффект достигается за счет минимизации числа страниц, считываемых из внешней памяти:

(i) СУБД хранят данные таблиц в виде цепочек страниц, типовые размеры которых соответствуют 2К, 4К, 8К или 16К, размер страницы может устанавливаться как свойство сервера, в MS SQL размер страницы фиксированный — 8К [<https://learn.microsoft.com/ru-ru/sql/relational-databases/pages-and-extends-architecture-guide?view=sql-server-ver16>];

(ii) размер БД, как правило, не позволяет разместить ее в полном объеме в оперативной памяти, для хранения БД используется внешняя память;

(iii) обмен с внешней памятью (чаще всего БД располагается на жестком диске) является гораздо более медленной операцией, чем чтение/запись оперативной;

следовательно, чем меньше страниц будет считываться из внешней памяти в оперативную в ходе выполнения запроса, тем быстрее он будет выполнен.

Решение об использовании индексов принимает оптимизатор запросов СУБД, таким образом:

(i) задача разработчика БД — предложить систему индексов исходя из потенциального множества запросов, которые к ней будут выполняться;

(ii) задача оптимизатора — построить как можно больше возможных планов выполнения запросов и выбрать план с минимальной стоимостью выполнения.

Создание индекса в простейшем случае обеспечивает оператор:

```
create index <имя индекса>
on <имя таблицы> ( <имя столбца 1> [ , <имя столбца 2> [ , ... ] ] )
```

Например, если в таблице **Студент** используется поиск по столбцу **ФИО**, ускорить его поможет индекс:

```
create index Студент_ФИО
on Студент (ФИО)
go
```

## 9.2 Рекомендации по выбору индексов

Индексы ускоряют выборку данных, но приводят к расходу дополнительной памяти (индекс дублирует часть данных основной таблицы) и дополнительным затратам времени при модификации данных (вставка/удаление/модификация записей таблиц требует модификации и сортировки индекса). Выбор системы индексации в базе данных является нетривиальной задачей, вместе с тем можно дать ряд рекомендаций.

Индексы следует создавать:

(i) для полей, по которым происходит отбор данных (указанных в разделе **where** запросов);

(ii) для полей, используемых при сортировке выбираемых запросами наборов данных (указанных в разделе *order by* запросов);

(iii) для полей, по которым происходит отбор диапазонов (указанных в конструкции *between ... and* или эквивалентных логических выражениях, заданных в разделе *where* запросов);

(iv) для полей, по которым осуществляется соединение таблиц (указанных в конструкции *join ... on* или эквивалентных логических выражениях, заданных в разделе *where* запросов).

Индексы не следует создавать:

(i) для полей, по которым не происходит отбора записей и сортировки (не используемых в разделе *where* и других перечисленных выше разделах запросов);

(ii) для полей, содержащих мало различных значений: пол, группа крови и др., чем больше различных значений содержит столбец таблицы, тем более эффективно будет работать созданный по нему индекс (в массивах, содержащих много повторяющихся значений двоичный поиск деградирует до линейного);

(iii) для таблиц, содержащих мало записей.

Относительно п. (iii) можно заметить следующее. Часто в БД используются небольшие справочные таблицы, которые могут занимать всего одну страницу (2..16К) (в БД университета примером такой таблицы может быть **Факультет, Номер** — 1 байт, **Название** — *char(150)* — 150 байт, длина записи — 151 байт + служебная информация, 10 факультетов, размер таблицы ~1510 байт). Как было отмечено выше, основной эффект от использования индексов состоит в минимизации количества страниц, читаемых из внешней памяти. Если таблица умещается на одной или небольшом количестве страниц, создание индекса для нее никак не повлияет на скорость выполнения запросов.

Кроме индексов на основе сбалансированных деревьев, в последних версиях MS SQL появилась поддержка специализированных индексов для таблиц, содержащих разреженную информацию, и таблиц, оптимизированных для памяти

[<https://learn.microsoft.com/ru-ru/sql/relational-databases/sql-server-index-design-guide?view=sql-server-ver16>].

Как правило, индексы нельзя создавать для столбцов типов: *bit*, *text*, *image*. Поля типа *bit* часто упаковываются в байты — 8 подряд идущих битовых полей в записи таблицы займут один байт. Поля типов *text*, *image* и эквивалентные им, предназначенные для хранения текстовых и двоичных данных большого размера, не хранятся на страницах данных таблиц. Они располагаются на отдельных страницах памяти, указатели на которые содержатся в записях таблиц БД.

### 9.3 Операторы языка SQL для создания и удаления индексов

В более общем случае синтаксис оператора для создания индекса выглядит следующим образом:

```
create [ unique ] [ clustered | nonclustered ] index <имя индекса>
on <имя таблицы> ( <имя столбца 1> [ asc | desc ] [ , ... ] )
[ with fillfactor = n ]
go
```

Полный синтаксис для MS SQL и других SQL-серверов позволяет указать значительно больше параметров и свойств [<https://learn.microsoft.com/ru-ru/sql/t-sql/statements/create-index-transact-sql?view=sql-server-ver16>], здесь рассмотрим только основные из них.

Индекс может быть:

(i) Уникальным или неуникальным (директива *unique*). Уникальный индекс не допускает дублирования значений, как следствие, попытка добавить в таблицу строки, содержащие значения уже имеющиеся в столбцах индекса будет отклонена. По умолчанию (директива *unique* опущена), создается неуникальный индекс. Часто СУБД реализуют ограничение *unique*, задаваемое в { *alter* / *create* } *table* путем создания уникального некластерного индекса. Создание уникального индекса не будет выполнено, если столбцы содержат повторяющиеся значения.

(ii) Кластерным или некластерным (директивы *clustered* или *nonclustered*). Создание кластерного индекса приводит к тому, что записи таблицы физически упорядочиваются в соответствии с порядком, определенным индексом. Как следствие, у таблицы может быть только один кластерный индекс. Количество некластерных индексов зависит от СУБД, типовым значением является 249 (в последних версиях MS SQL увеличено до 999). Часто СУБД реализуют ограничение *primary key*, задаваемое в *{ alter / create } table* путем создания уникального кластерного индекса. В дереве индекса выделяют корневую страницу – *root* (в индексе может быть только одна корневая страница), страницы промежуточных уровней – *intermediate* (промежуточных уровней может быть несколько) и листовые страницы – *leaf*. В кластерном индексе листовые страницы совпадают со страницами данных таблицы (*data pages*), в некластерном – содержат указатели на них. Таким образом в случае кластерного индекса данные таблицы оказываются встроены в индекс.

(iii) Простым (состоит из одного столбца) или составным (включает несколько столбцов таблицы). СУБД могут ограничивать число столбцов индекса и длину записи индекса (сумма длин столбцов, образующих индекс), в последних версиях MS SQL это до 32 столбцов, 900 байт для кластерных и 1700 для некластерных индексов. Записи в индексе могут упорядочиваться по возрастанию (*asc*) или по убыванию (*desc*), порядок сортировки задается отдельно для каждого столбца таблицы.

Имя индекса должно быть уникально в рамках таблицы.

Значение *n* для параметра *fillfactor* задает % заполнения страниц индекса при его создании. Неполное заполнение страниц позволяет избежать при вставке данных большого количества случаев переполнения страниц, приводящих к их расщеплению. Так как расщепление страницы достаточно затратная операция, для таблиц, в которые производится интенсивная вставка данных, рекомендуется создавать индексы с неполным заполнением страниц. Параметр *fillfactor* может относиться к листовым страницам индекса и страницам промежуточного уровня,

или только листовым страницам, в MS SQL это регулируется параметром *pad\_index = { on / off }*.

Созданный индекс, как и другие объекты БД, существует, пока не будет явно удален директивой (сохранена для обеспечения совместимости с предыдущими версиями):

```
drop index <имя таблицы>.<имя индекса> [ , ... ]
go
```

или директивой:

```
drop index <имя индекса> on <имя таблицы> [ , ... ]
go
```

Перестройку индекса в целях устранения фрагментации [<https://learn.microsoft.com/ru-ru/sql/relational-databases/indexes/reorganize-and-rebuild-indexes?view=sql-server-ver16>] осуществляет директива:

```
alter index <имя индекса> on <имя таблицы> rebuild
go
```

Информацию о существующих индексах таблицы возвращает СХП:

```
sp_helpindex <имя таблицы>
go
```

С учетом изложенного можно уточнить рекомендации по выбору индексов, кластерные индексы целесообразно создавать для столбцов:

- (i) являющихся первичными ключами;
- (ii) по которым осуществляется сортировка;
- (iii) по которым осуществляется отбор диапазонов;
- (iv) по которым осуществляется соединение таблиц;
- (v) которые нечасто изменяются.

Для столбцов, не включенных в кластерный индекс, создаются некластерные, при этом следует стремиться что бы:



(i) индекс «покрывал» запрос — содержал все используемые в запросе ключи поиска (столбцы, используемые в разделе *where* и др.) и все выбираемые значения (столбцы, используемые в разделе *select*);

(ii) осуществлялось префиксное сканирование индекса (обход по дереву).

Для достижения компромисса между тем, чтобы строки индекса были по возможности короткими, и тем, чтобы индекс покрывал запрос, в последних версиях MS SQL в оператор *create index* была введена директива *include*.

Короткие строки индекса позволяют разместить на одной странице индекса большее количество записей, соответственно, при считывании одной страницы индекса из внешней памяти, СУБД получает информацию о большем количестве строк таблицы.

Директива *include* (*<имя столбца I> [ , ... ]*) позволяет дополнить индекс столбцами, не входящими в ключи поиска. Значения таких столбцов не хранятся на корневой странице индекса и страницах промежуточного уровня, ими дополняются только листовые страницы индекса. MS SQL также автоматически дополняет листовые страницы некластерных индексов ключами таблицы. Включенные столбцы не учитываются при проверке ограничения на максимальное число столбцов и максимальную длину строки индекса. Максимальное количество включенных столбцов — 1023, это на 1 меньше, чем максимальное количество столбцов таблицы.

Префиксное сканирование индекса возможно, если в условиях запроса содержится значение для первого столбца, входящего в индекс. В противном случае осуществляется сплошное сканирование листовых страниц индекса. Тем не менее, сплошное сканирование индекса как правило эффективнее сканирования страниц данных таблицы, так как строки в индексе короче и считывая одну страницу индекса СУБД получает информацию о большем количестве строк чем при чтении страницы данных таблицы.

## 9.4 Генерация тестовых данных

Для тестирования выполнения запросов и оценки эффективности выбранных индексов необходимо, чтобы в таблицах БД было количество записей, сопоставимое с тем, которое будет достигнуто (возможно, в течение нескольких лет) в ходе реальной эксплуатации информационной системы. Один из способов решения этой задачи – использование генератора тестовых данных.

Существует достаточно много продуктов, осуществляющих генерацию данных для тестирования запросов к БД в ходе ее разработки, здесь можно упомянуть CASE-системы (например, SAP Sybase PowerDesigner [<http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38058.1600/doc/html/rad1232021421990.html>]), отдельные продукты (например, EMS Data Generator for SQL Server [<https://www.sqlmanager.net/ru/products/mssql/datagenerator>], SQL Data Generator [<https://www.red-gate.com/products/sql-development/sql-data-generator>], dbForge Data Generator for SQL Server [<https://www.qbssoftware.com/dbforge-data-generator.html>], ApexSQL Generate [[https://www.apexsql.com/sql\\_tools\\_generate.aspx](https://www.apexsql.com/sql_tools_generate.aspx)]).

## 9.5 Анализ использования индексов

Решение об использовании индексов принимает оптимизатор запросов СУБД, проанализировать используется ли индекс и каким образом происходит его сканирование можно с помощью планов выполнения запросов. Один из способов просмотра планов выполнения запросов — использование директивы [<https://learn.microsoft.com/ru-ru/sql/t-sql/statements/set-showplan-text-transact-sql?view=sql-server-ver16>]:

```
set showplan_text { on | off }
go
```

или более подробно с помощью директивы [<https://learn.microsoft.com/ru-ru/sql/t-sql/statements/set-showplan-all-transact-sql?view=sql-server-ver16>]:

```
set showplan_all { on | off }
go
```

В MS SQL директивы отображения планов отключают выполнение запросов, для выполнения запросов необходимо установить значения параметров в *off*.

Для графического отображения плана выполнения запроса используются команды меню Management Studio *Запрос\Показать предполагаемый план выполнения* (*Query\Display Estimated Execution Plan*) и *Запрос\Включить действительный план выполнения* (*Query\Include Actual Execution Plan*). В планах выполнения запросов отображается довольно много пиктограмм и видов сообщений [\[https://learn.microsoft.com/ru-ru/sql/relational-databases/showplan-logical-and-physical-operators-reference?view=sql-server-ver16\]](https://learn.microsoft.com/ru-ru/sql/relational-databases/showplan-logical-and-physical-operators-reference?view=sql-server-ver16), об использовании индексов говорят сообщения *[ Clustered ] Index { Seek / Scan }*, *Seek* соответствует префиксному сканированию индекса, а *Scan* — сплошному.

Повлиять на решения, принимаемые планировщиком запросов относительно порядка соединения таблиц, использования индексов, алгоритмов выполнения соединений и др. операций можно с помощью подсказок (указаний) — *hints* [\[https://learn.microsoft.com/ru-ru/sql/t-sql/queries/hints-transact-sql?view=sql-server-ver16\]](https://learn.microsoft.com/ru-ru/sql/t-sql/queries/hints-transact-sql?view=sql-server-ver16).

## 9.6 Лабораторная работа 9

Произвести генерацию и вставку тестовых данных в БД; выполнить запросы из ЛР 3..5 [1] или аналогичные им, зафиксировать планы и время выполнения запросов; создать систему индексов для ускорения выполнения запросов, исходя из рекомендаций в п. 9.2; повторно выполнить запросы, зафиксировать планы и время выполнения.

Для исключения влияния кэширования при оценке времени выполнения запросов можно воспользоваться директивами:

```

checkpoint
go
dbcc freeproccache
go
dbcc freesystemcache ('all')
go
dbcc dropcleanbuffers
go

```

Альтернативой является перезапуск службы MS SQL.

С помощью указаний (*hints*) в операторе *select* задать оптимизатору решения относительно использования индексов, алгоритмов соединения таблиц и др. операций, сравнить планы выполнения с созданными оптимизатором. Варианты заданий приведены в ПРИЛОЖЕНИИ.

Содержание отчета:

- операторы для создания индексов;
- планы и время выполнения запросов до, после индексации и с использованием директив *hints*;
- сопоставление, анализ и описание изменений, произошедших в планах.

## 10. ТРАНЗАКЦИИ И БЛОКИРОВКИ

### 10.1 Транзакции

Транзакция — совокупность действий в БД, которая должна быть или полностью успешно выполнена или полностью отклонена. Типовым примером транзакции является перевод средств в банковской системе:

- (i) сумма списывается со счета *A*;
- (ii) средства зачисляются на счет *B*.

Если между (i) и (ii) происходит программно-аппаратный сбой, БД оказывается в несогласованном состоянии. Механизм транзакций гарантирует, что либо оба действия будут успешно выполнены, и БД окажется в согласованном состоянии, либо не будет выполнено ни одно из них, и БД окажется в согласованном состоянии, в котором она находилась до начала транзакции. Говоря о

транзакциях, обычно отмечают ряд их свойств сокращенно обозначаемых как **ACID** [<https://learn.microsoft.com/ru-ru/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=sql-server-ver16>].

По умолчанию SQL-сервера рассматривают каждый оператор *insert*, *update*, *delete*, *select* и др. как отдельную транзакцию. Для оформления в виде транзакции группы операторов используются директивы [<https://learn.microsoft.com/ru-ru/sql/t-sql/language-elements/transactions-transact-sql?view=sql-server-ver16>]:

- (i) *begin { tran / transaction } [ <имя транзакции> ]* — начинает новую транзакцию;
- (ii) *save { tran / transaction } <имя точки сохранения>* — сохраняет текущее состояние транзакции;
- (iii) *commit [ { tran / transaction } [ <имя транзакции> ] ]* — фиксирует в БД изменения, произведенные транзакцией, транзакция завершается успешно;
- (iv) *rollback [ { tran / transaction } [ { <имя транзакции> | <имя точки сохранения> } ] ]* — отменяет изменения, произведенные транзакцией, либо до первого *begin transaction* (транзакция завершается с откатом), либо до указанной точки сохранения.

Пусть в БД существует таблица **Счет**:

```
create table Счет (
    Номер int primary key,
    ФИО varchar(50) not null,
    Сумма money not null)
go
```

оформим с помощью транзакций перевод денег с одного счета на другой:

```

begin tran
    update Счет set Сумма = Сумма + 100 where Номер = 33
    if (select Сумма from Счет where Номер = 44) >= 100
    begin
        update Счет set Сумма = Сумма - 100 where Номер = 44
        commit tran
    end
    else
        rollback tran
go

```

Таким образом, если на счете с номером 44 недостаточно средств, транзакция будет отклонена.

Помимо пакетных заданий транзакции могут использоваться в теле хранимой процедуры или триггера [<http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc32300.1550/html/sqlug/X16153.htm>].

Транзакция может находиться в одном из четырех возможных состояний:

- 0 — транзакция выполняется, последний оператор выполнен успешно;
- 1 — транзакция успешно завершена (зафиксированы изменения);
- 2 — транзакция выполняется, последний оператор вызвал ошибку;
- 3 — транзакция завершена с откатом (все изменения отменены);

текущее состояние хранится в глобальной системной переменной

***@@transtate***

[<http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc32300.1550/html/sqlug/X27470.htm>], в MS SQL не поддерживается, для обработки ошибок, в том числе ошибок транзакций, служит конструкция ***try ... catch*** [<https://learn.microsoft.com/ru-ru/sql/t-sql/language-elements/try-catch-transact-sql?view=sql-server-ver16>].

Транзакции могут быть вложенными, текущий уровень вложенности содержится в глобальной системной переменной ***@@trancount*** [<https://learn.microsoft.com/ru-ru/sql/t-sql/functions/trancount-transact-sql?view=sql-server-ver16>]

[<https://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc32300.1550/html/sqlug/X34712.htm>].

SQL-сервера могут поддерживать два режима начала транзакций — неявные транзакции (транзакция начинается автоматически при выполнении очередного оператора DML) и явные (транзакция начинается после *begin transaction*), переключение режимов осуществляет директива *set implicit\_transactions { on / off }* [<https://learn.microsoft.com/ru-ru/sql/t-sql/statements/set-implicit-transactions-transact-sql?view=sql-server-ver16>]<sup>1</sup>.

Так как транзакции устанавливают и удерживают блокировки, при их разработке следует придерживаться ряда рекомендаций:

- (i) в теле транзакции не следует запрашивать ввод от пользователя, организовывать просмотр данных;
- (ii) транзакция должна быть по возможности короткой и затрагивать как можно меньше данных;
- (iii) для избежания тупиков транзакциям следует обращаться к данным в одном и том же порядке и использовать по возможности низкие уровни изоляции (см. далее).

## 10.2 Проблемы многопользовательского доступа к данным, их решение с помощью блокировок

SQL-сервер обычно выполняет смесь транзакций, поступающих от параллельно работающих пользовательских приложений. Количество пользователей в современных корпоративных информационных системах (КИС) может исчисляться сотнями и тысячами. В литературе [2] [3] [4] [5] обычно рассматривается три основных проблемы параллелизма, проиллюстрируем их на примере приведенной выше таблицы *Счет*:

- (i) Потеря результатов обновления:

---

<sup>1</sup> В других диалектах SQL встречается *set chained { on / off }*  
 [<https://help.sap.com/viewer/b65d6a040c4a4709afd93068071b2a76/16.0.3.1/en-US/aaa4c423bc2b1014b68ee2b35060ae28.html?q=set%20chained>]

<u>Момент времени</u>	<u>Транзакция А</u>	<u>Транзакция В</u>
t1 —————	Чтение кортежа: Номер = 33, Сумма = 200	—————
t2 —————	—————	Чтение кортежа: Номер = 33, Сумма = 200
t3 —————	Запись кортежа: Номер = 33, Сумма = 220	—————
t4 —————	—————	Запись кортежа: Номер = 33, Сумма = 190
t5 —————	Фиксация транзакции	Фиксация транзакции

Рис. 10.1

Таким образом, параллельное выполнение транзакций **А** (увеличивает сумму на 20) и **В** (списывает со счета 10) приводит к появлению некорректной суммы (190), вместо 210, как было бы при последовательном выполнении транзакций.

(ii) Зависимость от незафиксированных результатов:

<u>Момент времени</u>	<u>Транзакция А</u>	<u>Транзакция В</u>
t1 —————	—————	Чтение кортежа: Номер = 33, Сумма = 200
t2 —————	—————	Запись кортежа: Номер = 33, Сумма = 190
t3 —————	Чтение кортежа: Номер = 33, Сумма = 190	—————
t4 —————	Запись кортежа: Номер = 33, Сумма = 210	—————
t5 —————	—————	Откат транзакции



t6 ————— Фиксация транзакции —————

*Рис. 10.2*

Транзакция **A** воспользовалась незафиксированным изменениям, выполненным транзакцией **B**, в результате сумма оказалась равной 210, а не 220, как при последовательном выполнении транзакций.

(iii) Несовместный анализ:

<u>Момент времени</u>	<u>Транзакция A</u>	<u>Транзакция B</u>
t1 —————	Чтение кортежа: Номер = 33, Сумма = 200; $\Sigma = 200$	—————
t2 —————	Чтение кортежа: Номер = 34, Сумма = 300; $\Sigma = 500$	—————
t3 —————	—————	Чтение кортежа: Номер = 35, Сумма = 100
t4 —————	—————	Запись кортежа: Номер = 35, Сумма = 120
t5 —————	—————	Чтение кортежа: Номер = 33, Сумма = 200
t6 —————	—————	Запись кортежа: Номер = 33, Сумма = 150
t7 —————	Чтение кортежа: Номер = 35, Сумма = 120; $\Sigma = 620$	—————
t8 —————	Фиксация транзакции	Фиксация транзакции

*Рис. 10.3*

Таким образом, транзакция **A**, суммируя средства на счетах, возвращает  $\Sigma = 620$ , в то время как на момент ее завершения на данных счетах  $\Sigma = 570$ .

Чаще всего для решения проблем многопользовательского доступа к данным СУБД используют блокировки:

- (i) блокировку записи — ***X-Lock*** (от eXclusive Lock);
- (ii) блокировку чтения — ***S-Lock*** (от Shared Lock).

Если транзакция планирует осуществить чтение кортежа, она предварительно устанавливает блокировку чтения, если запись — блокировку записи, при этом действуют следующие правила совместимости блокировок:

- (i) если на кортеж установлена ***X-Lock***, то запросы на блокировку от других транзакций отклоняются (ставятся в очередь);
- (ii) если на кортеж установлена ***S-Lock***, то запросы на ***S-Lock*** от других транзакций удовлетворяются, а запросы на ***X-Lock*** отклоняются (ставятся в очередь).

Совместимость блокировок иллюстрирует таблица:

Таблица 10.1

Совместимость блокировок

Транзакция А / В	X	S	—
X	Нет	Нет	Да
S	Нет	Да	Да
—	Да	Да	Да

Таким образом, при отсутствии блокировок удовлетворяется любой запрос на блокировку, ***S-Lock*** совместимы с другими ***S-Lock***, ***S-Lock*** и ***X-Lock***, а также две ***X-Lock*** не совместимы.

Блокировки используются в сочетании с алгоритмом двухфазного блокирования:

- (i) прежде чем начать работу с кортежем, транзакция должна установить блокировку;
- (ii) после разблокирования какого-либо кортежа транзакция не должна устанавливать других блокировок.

Иначе говоря, все операции блокирования должны предшествовать первой операции разблокирования, что приводит к удержанию блокировок до конца транзакции.

Блокировки позволяют решить проблемы многопользовательского доступа к данным:

(i) Решение проблемы потери результатов обновления:

<u>Момент времени</u>	<u>Транзакция А</u>	<u>Транзакция В</u>
t1 —————	X-Lock кортежа: Номер = 33	—————
t2 —————	Чтение кортежа: Номер = 33, Сумма = 200	X-Lock кортежа: Номер = 33
t3 —————	Запись кортежа: Номер = 33, Сумма = 220	Ожидание
t4 —————	Фиксация транзакции, снятие блокировок	Ожидание
t5 —————	—————	Чтение кортежа: Номер = 33, Сумма = 220
t6 —————	—————	Запись кортежа: Номер = 33, Сумма = 210
t7 —————	—————	Фиксация транзакции, снятие блокировок

*Рис. 10.4*

(ii) Решение проблемы зависимости от незафиксированных результатов:

<u>Момент времени</u>	<u>Транзакция А</u>	<u>Транзакция В</u>
t1 —————	—————	X-Lock кортежа: Номер = 33
t2 —————	—————	Чтение кортежа:

		Номер = 33, Сумма = 200
t3	X-Lock кортежа: Номер = 33	Запись кортежа: Номер = 33, Сумма = 190
t4	Ожидание	_____
t5	_____	Откат транзакции, сня- тие блокировок
t6	Чтение кортежа: Номер = 33, Сумма = 200	_____
t7	Запись кортежа: Номер = 33, Сумма = 220	_____
t8	Фиксация транзакции, снятие блокировок	_____

Рис. 10.5

(iii) Решение проблемы несовместного анализа:

<u>Момент времени</u>	<u>Транзакция А</u>	<u>Транзакция В</u>
t1	S-Lock кортежа: Номер = 33	_____
t2	Чтение кортежа: Номер = 33, Сумма = 200; $\Sigma = 200$	_____
t3	S-Lock кортежа: Номер = 34	X-Lock кортежа: Номер = 35
t4	Чтение кортежа: Номер = 34, Сумма = 300; $\Sigma = 500$	Чтение кортежа: Номер = 35, Сумма = 100
t5	S-Lock кортежа: Номер = 35	Запись кортежа: Номер = 35, Сумма = 120
t6	Ожидание	X-Lock кортежа:



Рис. 10.6

Таким образом, блокировки решают проблемы многопользовательского доступа, но создают проблему тупиков (взаимной блокировки — *deadlock*). Поэтому СУБД ведет граф зависимости транзакций, показывающий, какая транзакция ожидает снятия блокировок другой транзакцией. Если в графе обнаруживается цикл, одна из транзакций приносится в жертву — принудительно откатывается и перезапускается через некоторый интервал времени или инициируется исключительная ситуация. В качестве жертвы, как правило, выбирается транзакция, сделавшая наименьший объем изменений в БД.

Разработчик БД может повлиять на механизм обработки тупиков, задавая:

(i) приоритет (относительную важность) текущего сеанса по отношению к другим сеансам, работающим с БД, с помощью директивы *set deadlock\_priority* [<https://learn.microsoft.com/ru-ru/sql/t-sql/statements/set-deadlock-priority-transaction?view=sql-server-ver16>].

(ii) периодичность проверки возникновения тупика с помощью вызова СХП *sp\_configure 'deadlock checking period', <количество миллисекунд>* [[https://help.sap.com/docs/SAP\\_ASE/379424e5820941d0b14683dd3a992d5c/a759f5f7bc2b101494dff57ac2419b6c.html?locale=en-US](https://help.sap.com/docs/SAP_ASE/379424e5820941d0b14683dd3a992d5c/a759f5f7bc2b101494dff57ac2419b6c.html?locale=en-US)] (в MS SQL не поддерживается, период проверки автоматически выбирается сервером).

Блокировки снижают степень параллелизма при выполнении транзакций и создают проблему тупиков, поэтому современные СУБД используют не две, а до двух десятков видов блокировок для достижения большей производительности и минимизации вероятности появления тупика. Кроме того, блокировки могут применяться к объектам на различных уровнях иерархии: строке, странице, таблице [<https://learn.microsoft.com/ru-ru/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=sql-server-ver16#lock-granularity-and-hierarchies>].

### 10.3 Уровни изоляции транзакций

Повысить производительность и снизить вероятность тупика разработчик БД может, выбирая уровень изоляции транзакций. В стандарте языка SQL предложены 4 уровня изоляции транзакций, изменение уровня влияет на логику работы блокировок. В зависимости от выбранного уровня допускается или исключается появление в БД следующих ситуаций:

(i) ***dirty read*** — «грязное» чтение, данная ситуация возникает, когда транзакция читает незафиксированные данные, измененные другой транзакцией:

<u>Момент времени</u>	<u>Транзакция А</u>	<u>Транзакция В</u>
t1 —————	begin transaction	—————
t2 —————	—————	begin transaction
t3 —————	update Счет set Сумма = Сумма - 100 where Номер = 777	—————
t4 —————	—————	select sum(Сумма) from Счет where Номер < 1000
t5 —————	—————	commit transaction
t6 —————	rollback transaction	—————

Рис. 10.7

Транзакция **В**, получила значение суммы, не соответствующее состоянию БД из-за отката изменений, сделанных транзакцией **А**.

(ii) ***non-repeatable read*** — неповторяемое чтение, данная ситуация возникает, когда транзакция дважды читает данные и получает различные значения из-за изменений, сделанных другой транзакцией:

<u>Момент времени</u>	<u>Транзакция А</u>	<u>Транзакция В</u>
t1 —————	begin transaction	—————
t2 —————	—————	begin transaction

t3	select Сумма from Счет where Номер = 777	
t4		update Счет set Сумма = Сумма - 100 where Номер = 777
t5		commit transaction
t6	select Сумма from Счет where Номер = 777	
t6	commit transaction	

Рис. 10.8

Транзакция **A** дважды выполняя один и тот же запрос в моменты времени **t3** и **t6**, получает различные результаты.

(iii) *phantom rows* — фантомные строки, данная ситуация возникает, когда одна транзакция отбирает набор данных запросом по некоторому условию, вторая транзакция добавляет, удаляет или изменяет данные так, что строки перестают или начинают удовлетворять условию запроса, после чего первая транзакция повторяет запрос и получает другой набор данных:

<u>Момент времени</u>	<u>Транзакция A</u>	<u>Транзакция B</u>
t1	begin transaction	
t2		begin transaction
t3	select * from Счет where Номер < 1000	
t4		insert into Счет values (999, 'Сыроежкин', 200)
t5		commit transaction
t6	select * from Счет where Номер < 1000	
t6	commit transaction	

Рис. 10.9

Транзакция **A** дважды выполняя один и тот же запрос в моменты времени **t3** и **t6**, получит различные наборы данных. Случаи (ii) и (iii) могут показаться

похожими, основное отличие в том, что в (ii) речь идет о стабильности отдельных строк, а в (iii) — наборов строк (диапазонов).

Исключить появление рассмотренных ситуаций в БД можно, изменяя уровни изоляции транзакций:

Таблица 10.2

Уровни изоляции транзакций

Но- мер	Наименование	dirty read	non-repeatable read	phantom rows
0	read uncommitted	да	да	да
1	read committed	нет	да	да
2	repeatable read	нет	нет	да
3	serializable	нет	нет	нет

Как следует из таблицы 2, на самом низком уровне изоляции (*read committed*) проявляются все рассмотренные выше ситуации, повышение уровня изоляции последовательно приводит к их исключению. На самом высоком уровне изоляции (*serializable*) транзакции выполняются так, как если бы работа с БД происходила последовательно. Низкие уровни изоляции транзакций снижают вероятность возникновения тупиков и повышают параллелизм, так как на этих уровнях СУБД изменяет логику работы блокировок, отступая от рассмотренной выше.

Установку текущего уровня изоляции транзакций осуществляет директива *set transaction isolation level { read committed / read uncommitted / repeatable read / serializable <sup>1</sup> }*. Установленный уровень изоляции транзакций действует до конца сеанса или до тех пор, пока не будет изменен другой директивой *set transaction isolation level*. Получить значение текущего уровня изоляции транзакций можно,

---

<sup>1</sup> Некоторые SQL сервера допускают указание как названия, так и номера уровня, в MS SQL не поддерживается.



или выполнив директиву *dbcc useroptions* (MS SQL) или опросив глобальную системную переменную *@@isolation* (в MS SQL не поддерживается).

Задать уровень изоляции для отдельного запроса или таблицы, не изменяя уровень изоляции, установленный для сеанса, можно с помощью табличных подсказок (указаний) — *hints* [<https://learn.microsoft.com/ru-ru/sql/t-sql/queries/hints-transact-sql-table?view=sql-server-ver16>]<sup>1</sup>.

## 10.4 Конфигурирование блокировок, отчеты о блокировках

Для конфигурирования блокировок используются директивы, позволяющие задать:

(i) тайм-аут установления блокировки — *set lock\_timeout* *<количество миллисекунд>*, значение параметра находится в глобальной системной переменной *@@lock\_timeout*;

(ii) максимальное количество блокировок — *sp\_configure "locks", <количество>*, если в качестве количества указан 0 — память распределяется автоматически; параметр, в том числе, оказывает влияние на решение о повышении уровня блокировки [<https://learn.microsoft.com/ru-ru/sql/database-engine/configure-windows/configure-the-locks-server-configuration-option?view=sql-server-ver16>].

Для получения отчетов о блокировках используются директивы:

(i) *sp\_lock* [*<процесс I>* [, ...]] — возвращает сведения о всех блокировках или блокировках для указанных процессов (сеансов работы) [<https://learn.microsoft.com/ru-ru/sql/relational-databases/system-stored-procedures/sp-lock-transact-sql?view=sql-server-ver16>];

(ii) *sp\_who* [*<имя пользователя>*] — возвращает сведения о всех процессах или процессах указанного пользователя [<https://learn.microsoft.com/ru-ru/sql/relational-databases/system-stored-procedures/sp-who-transact-sql?view=sql-server-ver16>].

---

<sup>1</sup> В других диалектах SQL для этих целей может использоваться директива *at isolation* [<http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00938.1502/html/locking/locking105.htm>]

Альтернативными способами получения сведений о блокировках являются системные представления *sys.dm\_tran\_locks* [<https://learn.microsoft.com/ru-ru/sql/relational-databases/system-dynamic-management-views/sys-dm-tran-locks-transact-sql?view=sql-server-ver16>], *sys.syslockinfo* [<https://learn.microsoft.com/ru-ru/sql/relational-databases/system-compatibility-views/sys-syslockinfo-transact-sql?view=sql-server-ver16>].

Идентификатор сеанса для текущего пользовательского процесса содержится в глобальной переменной @@spid.

Если начать две транзакции, 1-й сеанс (1-е окно запроса в SSMS):

```
set transaction isolation level serializable

begin transaction

update Счет
set Сумма = Сумма - 100
where Номер = 33
```

2-й сеанс (2-е окно запроса в SSMS):

```
set transaction isolation level serializable

begin transaction

select sum(Сумма)
from Счет
where Номер < 1000
```

можно получить отчет о блокировках, 3-й сеанс (3-е окно запроса в SSMS):

```
select * from sys.dm_tran_locks
where resource_database_id = DB_ID('Университет')
go
```

который будет выглядеть следующим образом (часть столбцов, возвращаемых представлением, опущена):

resource_type	...	request_mode	request_type	request_status
DATABASE	...	S	LOCK	GRANT
DATABASE		S	LOCK	GRANT
PAGE		IS	LOCK	GRANT

PAGE	IX	LOCK	GRANT
KEY	X	LOCK	GRANT
KEY	RangeS-S	LOCK	WAIT
OBJECT	IS	LOCK	GRANT
OBJECT	IX	LOCK	GRANT

Транзакция во втором сеансе выставила запрос на блокировку диапазона ключей, используемую для предотвращения фантомов, и находится в режиме ожидания снятия монопольной блокировки, установленной транзакцией в первом сеансе.

SQL-сервера поддерживают средства для предоставления отчетов о тупиках, это могут быть СХП [<http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc36273.1572/html/sprocs/X25628.htm>], или отдельные приложения, например, SQL Server Profiler, входящий в состав MS SQL [<https://learn.microsoft.com/ru-ru/sql/tools/sql-server-profiler/sql-server-profiler?view=sql-server-ver16>].

SQL Server Profiler позволяет отображать данные о тупиках, в том числе, в графическом виде. Запуск SQL Server Profiler — *Пуск\Все приложения\Microsoft SQL Server Tools 18\SQL Server Profiler 18*. Для выполнения трассировки — *Файл\Создать трассировку...\Соединить*, далее на вкладке *Выбор событий* установить флаг *Показать все события*, после чего выбрать раздел *Locks*, установить флаг *Deadlock graph* (другие установленные флаги можно сбросить) и нажать кнопку *Запустить* [<https://learn.microsoft.com/ru-ru/sql/relational-databases/performance/save-deadlock-graphs-sql-server-profiler?view=sql-server-ver16>].

## 10.5 Лабораторная работа 10

Смоделировать в БД<sup>1</sup> грязное чтение, неповторяемое чтение, фантомы<sup>2</sup>, изменяя уровень изоляции транзакций продемонстрировать их исключение, сформировать отчеты о блокировках, пояснить их содержание.

Смоделировать в БД тупик (взаимную блокировку), получить с помощью приложения SQL Server Profiler или его аналога отчет о тупике, пояснить его содержание.

Содержание отчета:

- скрипты транзакций;
- отчеты о блокировках и пояснения к ним;
- отчет о тупике, пояснения к нему.

Варианты заданий приведены в ПРИЛОЖЕНИИ.

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Богословская Н. В., Бржезовский А. В. Системы баз данных : учебно-методическое пособие. ч. 1 — С.-Петербург. гос. ун-т аэрокосм. приборостроения. — Санкт-Петербург : Изд-во ГУАП, 2021. — 75 с.
2. Г. Гарсиа-Молина Г., Ульман Дж. Д., Д. Уидом Д. Системы баз данных. Полный курс./Пер. с англ. — М.: Вильямс, 2003. — 1088 с.
3. Коннолли Т, Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика./Пер. с англ. — 3-е изд. — М.: Вильямс, 2003. — 1436 с.
4. Дейт К. Дж. Введение в системы баз данных./Пер. с англ. — 8-е изд. — М.: Вильямс, 2005. — 1138 с.

---

<sup>1</sup> В SQL Server Management Studio (SSMS) каждое новое окно запроса является отдельным сеансом работы с БД.

<sup>2</sup> Для моделирования параллельного выполнения транзакций можно использовать пошаговое выполнение операторов в разных сеансах или оператор *waitfor delay* для приостановки выполнения транзакций.

5. Роб П. Системы баз данных: проектирование, реализация и управление.  
— 5-е изд. — СПб.: БХВ - Петербург, 2004. — 1040 с.

## СОДЕРЖАНИЕ

<b>9. ИНДЕКСАЦИЯ ДАННЫХ</b>	<b>3</b>
9.1 Принципы индексации данных	3
9.2 Рекомендации по выбору индексов	4
9.3 Операторы языка SQL для создания и удаления индексов	6
9.4 Генерация тестовых данных	10
9.5 Анализ использования индексов	10
9.6 Лабораторная работа 9	11
<b>10. ТРАНЗАКЦИИ И БЛОКИРОВКИ</b>	<b>12</b>
10.1 Транзакции	12
10.2 Проблемы многопользовательского доступа к данным, их решение с помощью блокировок	15
10.3 Уровни изоляции транзакций	22
10.4 Конфигурирование блокировок, отчеты о блокировках	25
10.5 Лабораторная работа 10	28
<b>БИБЛИОГРАФИЧЕСКИЙ СПИСОК</b>	<b>28</b>
<b>СОДЕРЖАНИЕ</b>	<b>29</b>
<b>ПРИЛОЖЕНИЕ Варианты заданий</b>	<b>30</b>

## ПРИЛОЖЕНИЕ

### Варианты заданий

1. Создайте базу данных для хранения следующих сведений: алфавитный и тематический каталоги книг в библиотеке, читатель, читательский билет, формуляр читателя (выданные и возвращенные книги), напоминание о необходимости возврата книги. Составьте запросы, позволяющие выбрать:
  - а) читателей, которые брали книги на прошлой неделе;
  - б) читателей, которые брали книги Ахо и Ульмана;
  - в) читателей, у которых на руках две или более книги одного автора;
  - г) количество книг, находящихся на руках у каждого из читателей;
  - д) читателей, прочитавших более ста книг;
  - е) читателей, у которых на руках максимальное количество книг;
  - ж) читателей, взявших книги, которые больше никому не выдавались;
  - з) читателей, читающих книги всех жанров;
  - и) читателей, которые читают только книги жанра «приключения».
2. Создайте базу данных для хранения следующих сведений: специальность, учебный план, кафедра, преподаватель, дисциплина, группа, курс, вид занятия, пара. Составьте запросы, позволяющие выбрать:
  - а) преподавателей, ведущих Базы данных на различных факультетах;
  - б) преподавателей, ведущих как Базы данных, так и Логическое программирование;
  - в) преподавателей, которые ведут более двух видов занятий по одной дисциплине;
  - г) количество дисциплин для каждого преподавателя;
  - д) группы, у которых в среднем менее 3-х пар в день;
  - е) преподавателей, ведущих более трех различных дисциплин;
  - ж) преподавателей, которые ведут занятия только на старших курсах;
  - з) преподавателей, ведущих все виды занятий;
  - и) преподавателей, ведущих занятия в максимальном количестве групп.
3. Создайте базу данных для хранения следующих сведений: турфирма, тур, страна, турист, путевка, загранпаспорт, виза, ваучер отеля, билет. Составьте запросы, позволяющие выбрать:
  - а) туристов, посещавших в прошлом году Италию и Францию;
  - б) турфирмы, продающие туры в Египет и Турцию;
  - в) туристов, пользовавшихся услугами двух и более турфирм;
  - г) количество путевок, проданных каждой из турфирм за прошлый год;
  - д) среднюю цену путевки в Тунис;
  - е) туристов, побывавших во Франции более пяти раз;
  - ж) туристов, побывавших только в одной стране;
  - з) туристов, побывавших во всех странах, в которые предлагаются туры;
  - и) пары туристов, которые всегда путешествуют вместе.
4. Создайте базу данных для хранения следующих сведений: студент, группа, дисциплина, преподаватель, лабораторная работа, рейтинг за сданную лабораторную работу. Составьте запросы, позволяющие выбрать:
  - а) максимальный рейтинг, который может получить студент за работу №8 по БД;
  - б) работы и рейтинги, сданные и полученные конкретным студентом;
  - в) дисциплины, у которых есть лабораторные работы с одинаковыми названиями;
  - г) количество работ, сданных каждым студентом по БД;
  - д) студентов, у которых средний рейтинг за сданные лабораторные работы по БД превышает 4;

- е) студентов, не сдавших ни одной работы по БД;
  - ж) лабораторные по БД, которые нужно досдать Сыроежкину из группы 4000;
  - з) студентов, получивших одинаковый рейтинг за все работы по БД;
  - и) студентов, сдавших все работы по БД.
5. Создайте базу данных для хранения следующих сведений: ВУЗ, студент, группа, факультет, конференция, тема доклада, программа конференции. Составьте запросы, позволяющие выбрать:
- а) студентов первого факультета, выступавших на конференции Информатика;
  - б) темы докладов студентов для заданной группы;
  - в) выступления, подготовленные двумя студентами различных факультетов;
  - г) количество докладов для каждой конференции;
  - д) среднее количество докладов, сделанных студентами третьего факультета на конференциях;
  - е) студентов, выступивших на трех или большем числе конференций;
  - ж) студентов четвертого факультета, не выступавших на конференциях;
  - з) студентов, выступивших на всех конференциях;
  - и) пары студентов, всегда выступающие вместе.
6. Создайте базу данных для хранения следующих сведений: компьютерный магазин, модель компьютера, комплектующие, производители, поставщики, поставка, заказ, сборка. Составьте запросы, позволяющие выбрать:
- а) модели компьютеров, в которых используются винчестеры Samsung;
  - б) модели компьютеров, в которых используются как накопители SSD, так и HDD;
  - в) модели компьютеров, имеющие одинаковый размер оперативной и внешней памяти;
  - г) количество моделей, продаваемых в каждом из магазинов;
  - д) магазины, в которых средняя цена компьютера ниже, чем в других;
  - е) магазины, в которых продается наибольшее количество моделей;
  - ж) модели компьютеров, не имеющие накопителей DVD;
  - з) магазины, в которых не продаются модели, укомплектованные одновременно оборудованием Intel и Samsung;
  - и) модели компьютеров, укомплектованные всеми типами периферийных устройств.
7. Создайте базу данных для хранения следующих сведений: номер маршрута автобуса, остановка, транспортная компания, лицензия, график движения, автобус, водитель. Составьте запросы, позволяющие выбрать:
- а) маршруты, выполняемые заданной компанией;
  - б) маршруты, которыми можно доехать до Дворцовой площади;
  - в) маршруты, имеющие общие остановки;
  - г) количество маршрутов, обслуживаемых каждой компанией;
  - д) компании, обслуживающие наибольшее число маршрутов;
  - е) компании, средняя продолжительность маршрутов которых ниже чем у других;
  - ж) компании, маршруты которых не останавливаются на Дворцовой площади;
  - з) компании, у которых нет маршрутов короче, чем 10 остановок;
  - и) маршруты, которые включают все остановки заданного маршрута.
8. Создайте базу данных для хранения следующих сведений: театр, спектакль, жанр, автор, режиссер, актер, репертуар. Составьте запросы, позволяющие выбрать:
- а) спектакли жанра комедия;
  - б) спектакли, в которых занят заданный актер;
  - в) спектакли, идущие более чем в одном театре;
  - г) количество спектаклей для каждого из театров;

- д) театры, в которых количество драм превышает число комедий;
  - е) спектакли, в которых занято наибольшее число актеров;
  - ж) спектакли одного актера;
  - з) театры, в которых идут спектакли всех жанров;
  - и) актеров, занятых только в одном театре.
9. Создайте базу данных для хранения следующих сведений: аптека, медикамент: международное непатентованное и торговое наименование, состав, дозировка, упаковка, форма выпуска, цена, производитель. Составьте запросы, позволяющие выбрать:
- а) аптеки, в которых есть лекарства заданного производителя;
  - б) аптеки, в которых продается одно и то же лекарство различных производителей;
  - в) цена аспирина в различных аптеках;
  - г) количество наименований лекарств, продающихся в каждой из аптек;
  - д) аптеки, в которых цена аспирина минимальна;
  - е) средняя стоимость аспирина компании АБВ в аптеках;
  - ж) аптеки, в которых нет медикаментов, заданного производителя;
  - з) пары производителей, у которых нет ни одного одинакового медикамента;
  - и) аптеки, в которых есть все лекарства.
10. Создайте базу данных для хранения следующих сведений: фильм, сериал, литературная основа, студия, жанр, актер, режиссер, сценарист, продюсер. Составьте запросы, позволяющие выбрать:
- а) список фильмов, снятых заданной студией за заданный период;
  - б) перечень студий, в фильмах которых играл заданный актер;
  - в) актеров, снимавшихся как в комедиях, так и в мелодрамах;
  - г) студии, на которых количество мелодрам превышает число комедий;
  - д) актеров, снявшихся в десяти фильмах;
  - е) среднее количество фильмов каждого из жанров, снимающееся на студии за год;
  - ж) студии, на которых снимаются фильмы только одного жанра;
  - з) студии, на которых снимаются фильмы всех жанров;
  - и) студии, никогда не выпускавшие ремейков.
11. Создайте базу данных для хранения следующих сведений: автомобиль, модель, производитель, дилер, город, цена, продажа, владелец, техническое обслуживание. Составьте запросы, позволяющие выбрать:
- а) перечень моделей для заданного дилера;
  - б) дилеров, представляющих одновременно Toyota и УАЗ;
  - в) дилеров, продающих автомобили иностранного производства;
  - г) количество автомобилей, проданных каждым из дилеров в прошлом году;
  - д) среднюю цену Toyota Camry у дилеров в Москве;
  - е) производителей, у которых наибольшее количество дилеров в Санкт-Петербурге;
  - ж) дилеров, не продающих одновременно Ford и Renault;
  - з) дилеров, предлагающих модели всех производителей;
  - и) дилеров, у которых нет моделей дороже 300000.
12. Создайте базу данных для хранения следующих сведений: подразделение, штатное расписание, должность, сотрудник, дети, прием, перевод, увольнение. Составьте запросы, позволяющие выбрать:
- а) список сотрудников заданного подразделения;
  - б) подразделения, входящие в состав заданного;
  - в) сотрудников, у которых есть дети различного пола;
  - г) среднюю численность подразделений;



- д) инженеров, у которых более пяти детей;
  - е) подразделения, в которых количество техников превышает количество инженеров;
  - ж) подразделения, в которых не работают совместители;
  - з) подразделения, в которых представлены все должности;
  - и) сотрудников, у которых все дети одного пола.
13. Создайте базу данных для хранения следующих сведений: издательство, автор, книга, жанр, план издания, тираж, реализация. Составьте запросы, позволяющие выбрать:
- а) перечень книг, выпущенных заданным издательством в прошлом году;
  - б) авторы, сотрудничающие с несколькими издательствами;
  - в) книги, написанные в соавторстве;
  - г) количество книг каждого жанра, выпущенных каждым издательством;
  - д) авторов, написавших наибольшее количество книг;
  - е) средний объем книг, выпускаемых заданным издательством;
  - ж) издательства, выпускающие только сказки и детективы;
  - з) издательства, выпускающие книги всех жанров;
  - и) издательства, не выпустившие ни одной книги в 2019 году.
14. Создайте базу данных для хранения следующих сведений: врач, специальность, пациент, прием, история болезни (медицинская карта). Составьте запросы, позволяющие выбрать:
- а) список пациентов, принятых терапевтами вчера;
  - б) врачей, совмещающих различные специальности;
  - в) пациентов, посещавших и хирурга, и кардиолога;
  - г) количество пациентов, принятых каждым из врачей за прошедший год;
  - д) врачи, принявшие меньше всего пациентов;
  - е) врачей, у которых количество принимаемых пациентов превышает среднее;
  - ж) пациентов, которые никогда не посещали хирурга;
  - з) пациентов, которые посетили всех специалистов;
  - и) врачи, не совмещающие работу по различным специальностям.
15. Создайте базу данных для хранения следующих сведений: авиакомпания, авиарейс, расписание, самолет, экипаж, аэропорт, количество и длина полос. Составьте запросы, позволяющие выбрать:
- а) список рейсов для заданной авиакомпании;
  - б) типы самолетов, используемые заданной авиакомпанией;
  - в) авиакомпании, у которых прямой и обратный рейс выполняют различные типы самолетов;
  - г) направления, на которых работает более трех авиакомпаний;
  - д) количество авиарейсов, выполняемых между каждой парой аэропортов;
  - е) авиакомпании, выполняющие максимальное количество рейсов;
  - ж) авиакомпании, не работающие в Стамбуле;
  - з) авиакомпании, использующие все типы самолетов;
  - и) авиакомпании, у которых все самолеты одного производителя.
16. Создайте базу данных для хранения следующих сведений: город, район, квартира, комната, площадь, цена, агент, продажа. Составьте запросы, позволяющие выбрать:
- а) перечень однокомнатных квартир, продаваемых в Московском районе;
  - б) квартиры, находящиеся на одной улице, но в различных районах;
  - в) двух- и трехкомнатные квартиры, имеющие одинаковую площадь;
  - г) средняя цена однокомнатной квартиры в городе;
  - д) районы, в которых продается наибольшее число объектов недвижимости;
  - е) районы, в которых минимальна стоимость квадратного метра;
  - ж) улицы, продолжительность которых ограничивается только одним районом;

- з) районы, в которых не продаются однокомнатные квартиры;
  - и) районы, в которых продаются квартиры всех строительных серий.
17. Создайте базу данных для хранения следующих сведений: олимпиада, страна, спортсмен, команда, вид спорта, место. Составьте запросы, позволяющие выбрать:
- а) страны, принявшие участие в зимней олимпиаде 2018 г.;
  - б) спортсменов, принявших участие как в летних, так и в зимних олимпиадах;
  - в) спортсменов, получивших золото по двум или более видам соревнований на одной и той же олимпиаде;
  - г) среднее число спортсменов, выставляемых каждой страной на каждый из видов олимпиад;
  - д) страны, количество побед которых на зимних олимпиадах, превышает количество побед на летних;
  - е) страны, завоевавшие наибольшее количество наград в 2018 г.;
  - ж) страны, никогда не участвовавшие в зимних олимпиадах;
  - з) страны, не участвовавшие в олимпиадах в период 2008...2018 г. г.;
  - и) страны, завоевавшие призовые места по всем видам спорта.
18. Создайте базу данных для хранения следующих сведений: город, ж/д станция, поезд, остановки, вагон, место, пассажир, билет. Составьте запросы, позволяющие выбрать:
- а) пассажиров, покупавших билеты в прошлом месяце из СПб в Москву;
  - б) пассажиров, покупавших в течение месяца и прямые и обратные билеты;
  - в) поезда, в которых есть и купейные и плацкартные и сидячие вагоны;
  - г) количество поездов из СПб в Москву;
  - д) количество билетов, проданных на каждый поезд из СПб в Москву;
  - е) средняя цена места в купейном вагоне;
  - ж) поезда из СПб, делающие остановку в Окуловке и не делающие в Бологом;
  - з) поезда, у которых соотношение цены СВ и продолжительности в пути максимально;
  - и) поезда из СПб в Москву, делающие все остановки.
19. Создайте базу данных для хранения следующих сведений: оператор сотовой связи, абонент, договор, услуги, тарифы, опции, начисления, платежи. Составьте запросы, позволяющие выбрать:
- а) абонентов, пользующихся опцией АОН по какому-либо тарифу;
  - б) абонентов, поменявших в прошлом году тариф «нормальный» на тариф «оптимальный»;
  - в) тарифы, у которых есть одинаковые опции;
  - г) количество абонентов, пользующихся каждым из тарифов;
  - д) операторов, у которых средняя цена минуты выше, чем у других;
  - е) операторов, доходность которых выше, чем у АБВ;
  - ж) абонентов, не осуществлявших платежей в прошлом квартале;
  - з) тарифы, включающие все возможные опции;
  - и) абонентов, которые всегда изменяли тарифы одновременно (в один день).
20. Создайте базу данных для хранения следующих сведений: Интернет-провайдер, подключенные дома, характеристика канала, абонент, договор, тариф, начисления, оборудование, обращение в техподдержку, результат. Составьте запросы, позволяющие выбрать:
- а) абонентов, обращавшихся в техподдержку по вопросу неисправности CM440;
  - б) абонентов, обращавшихся в техподдержку дважды в прошлом месяце;
  - в) провайдеров, которые предоставляют доступ по тарифам как на скорости 50, так 70 МБит;
  - г) провайдеров, предлагающих более семи моделей кабельных модемов;
  - д) провайдеров, предлагающих наибольшее число тарифов;

- е) количество абонентов для каждого провайдера по каждому из тарифов;
  - ж) абонентов, никогда не изменявших тариф;
  - з) абонентов, пользовавшихся всеми тарифами;
  - и) абонентов, не обращавшихся в техподдержку более трех раз в год.
21. Создайте базу данных для хранения следующих сведений: сеть ресторанов, ресторан, меню, состав блюд, бронирование столиков, чеки клиентов. Составьте запросы, позволяющие выбрать:
- а) блюда, в состав которых входит говядина;
  - б) блюда, в состав которых входят одинаковые ингредиенты;
  - в) рестораны сети, в которых одинаковые блюда имеют различную цену;
  - г) количество блюд, предлагаемых в сети АБВ;
  - д) количество блюд, в которые входит каждый из ингредиентов;
  - е) рестораны, предлагающие в точности столько же блюд, что и заданный;
  - ж) рестораны, в которых нет чека, размер которого превышает 20000;
  - з) рестораны, в которых средний размер чека минимален;
  - и) рестораны, выручка которых год от года возрастает.
22. Создайте базу данных для хранения следующих сведений: почтовое отделение, обслуживаемые адреса, письмо, заказное письмо, бандероль, отправитель, получатель, прием/выдача корреспонденции. Составьте запросы, позволяющие выбрать:
- а) людей, отправлявших заказные письма в прошлом месяце;
  - б) людей, отправлявших письма в прошлом месяце дважды по одному и тому же адресу;
  - в) людей, отправлявших письма в прошлом месяце и получавших ответы;
  - г) количество писем, пересланных из СПб в Москву в прошлом году;
  - д) количество корреспонденции каждого из видов между СПб и Москвой;
  - е) средний вес бандеролей из Москвы в СПб;
  - ж) почтовые отделения, количество корреспонденции в которых больше, чем в других;
  - з) людей, отправляющих письма всегда из одного и того же отделения;
  - и) людей, отправивших в прошлом году все виды корреспонденции.