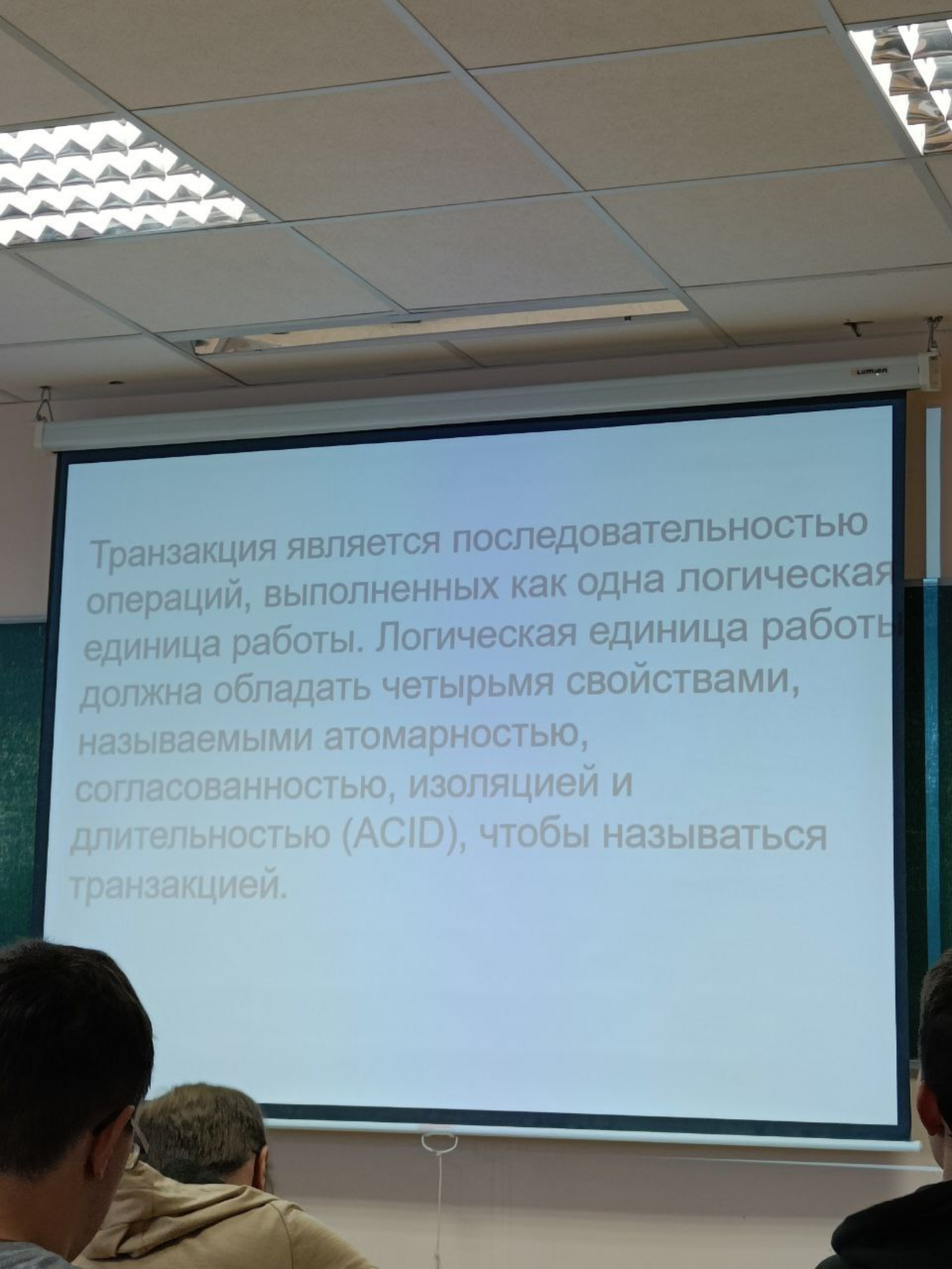
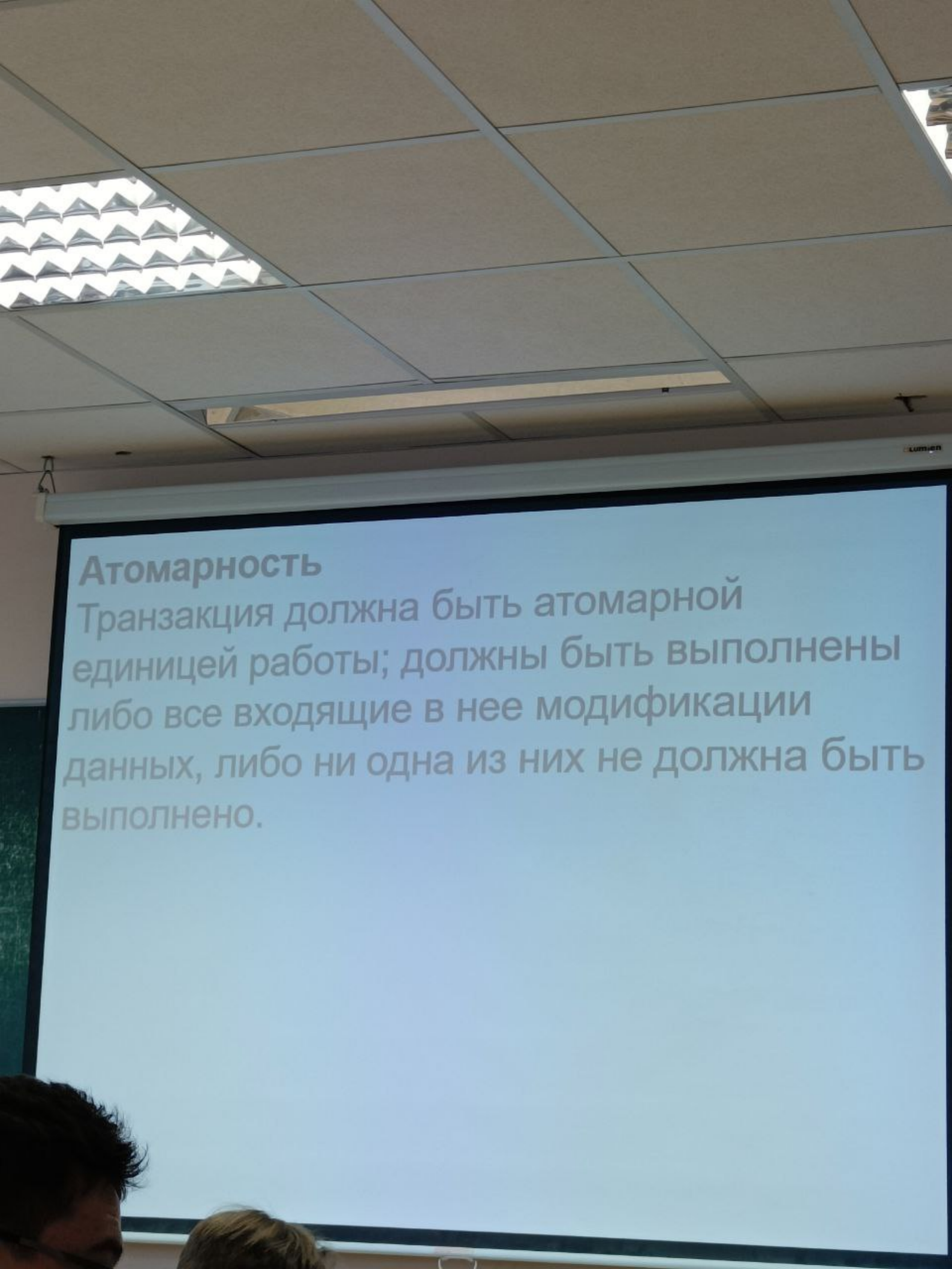
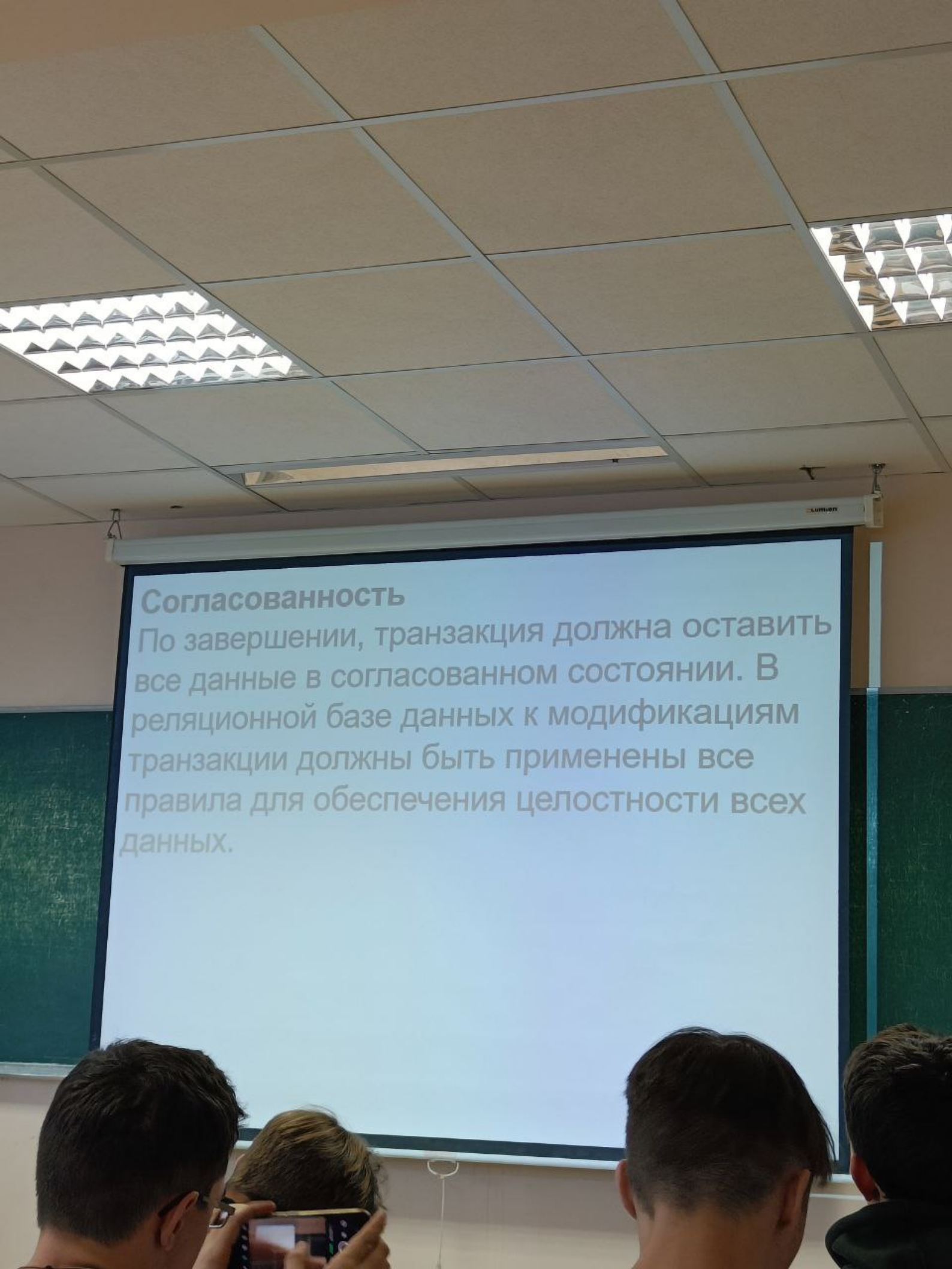
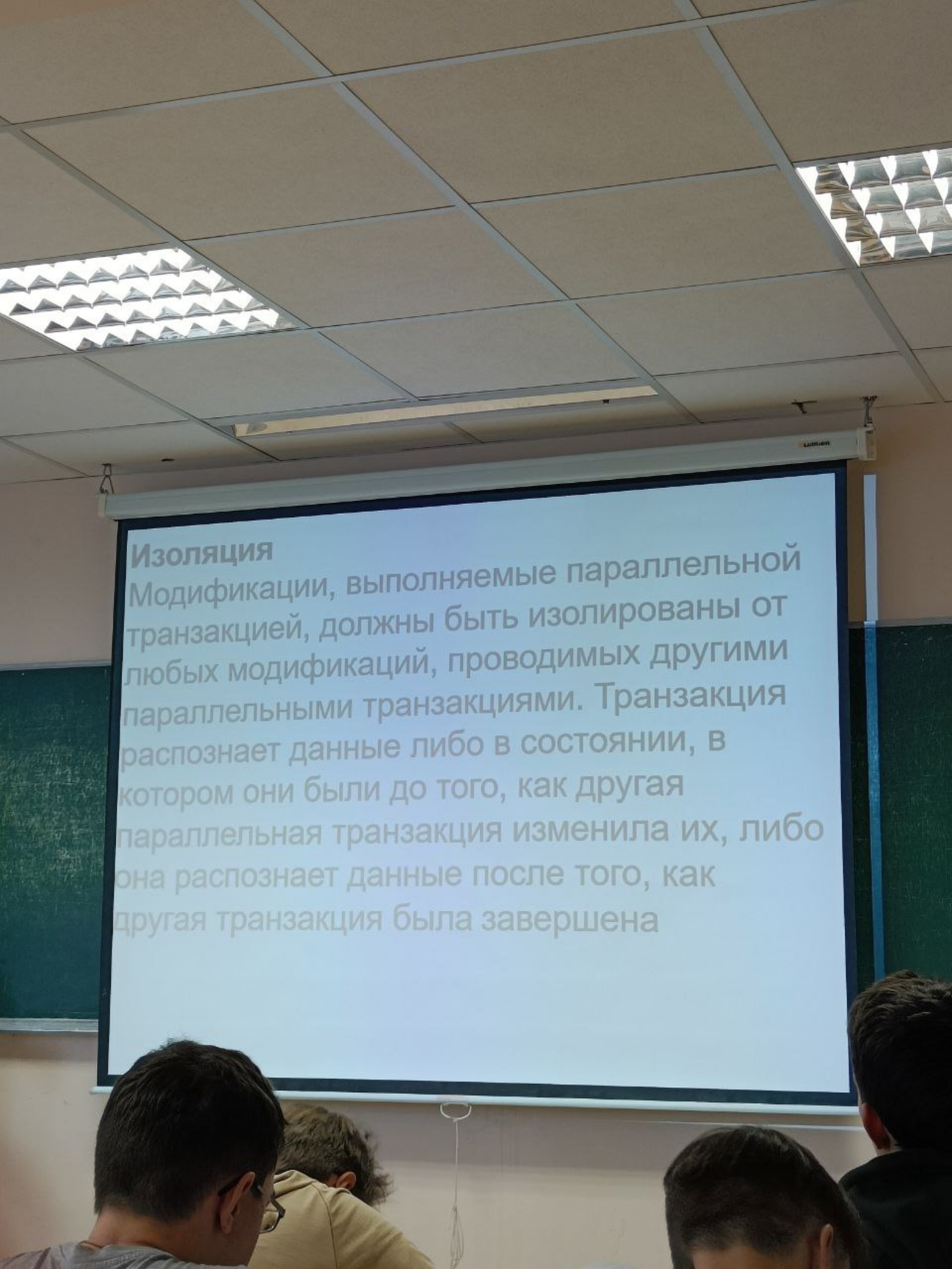
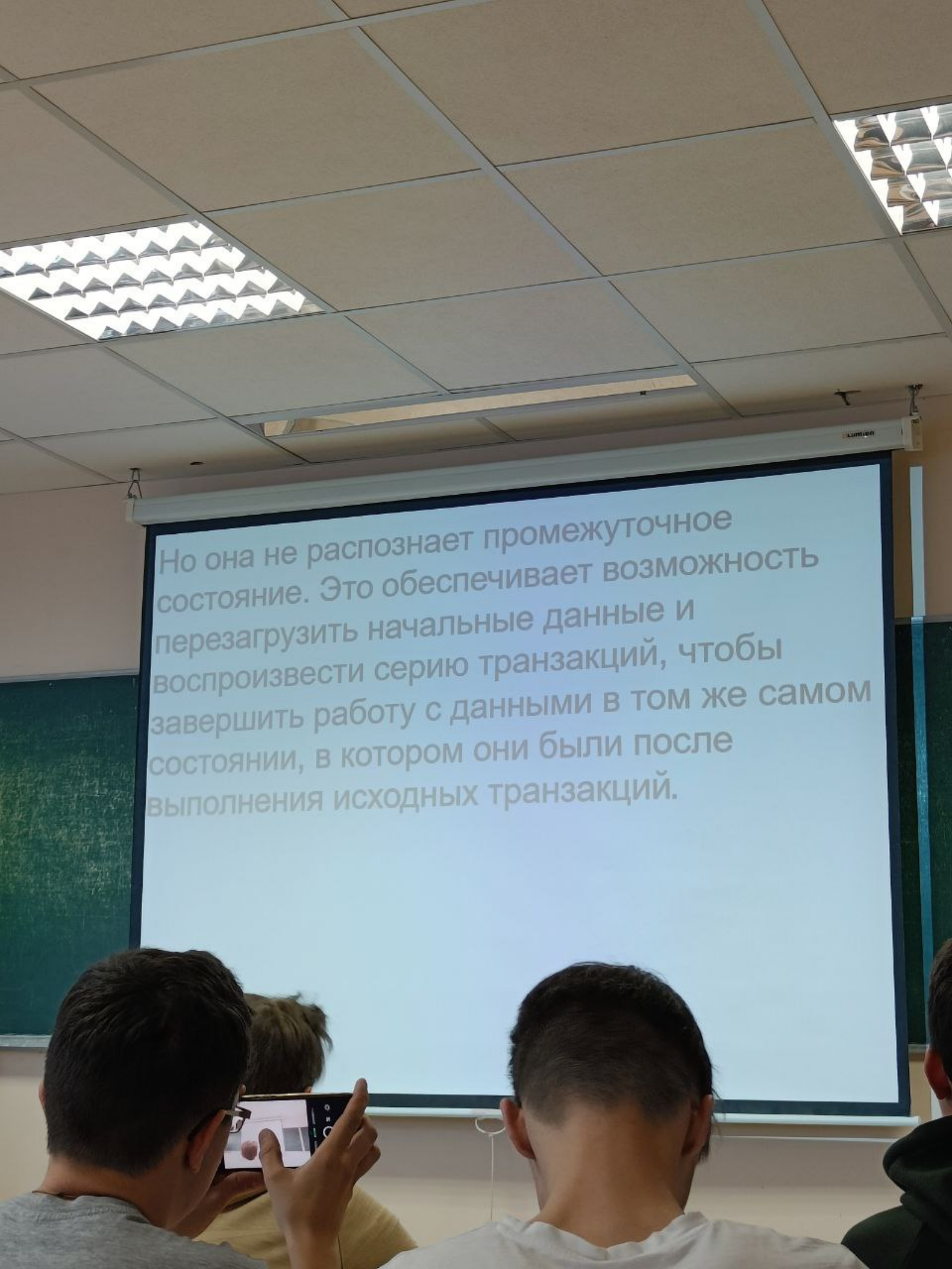
Транзакции

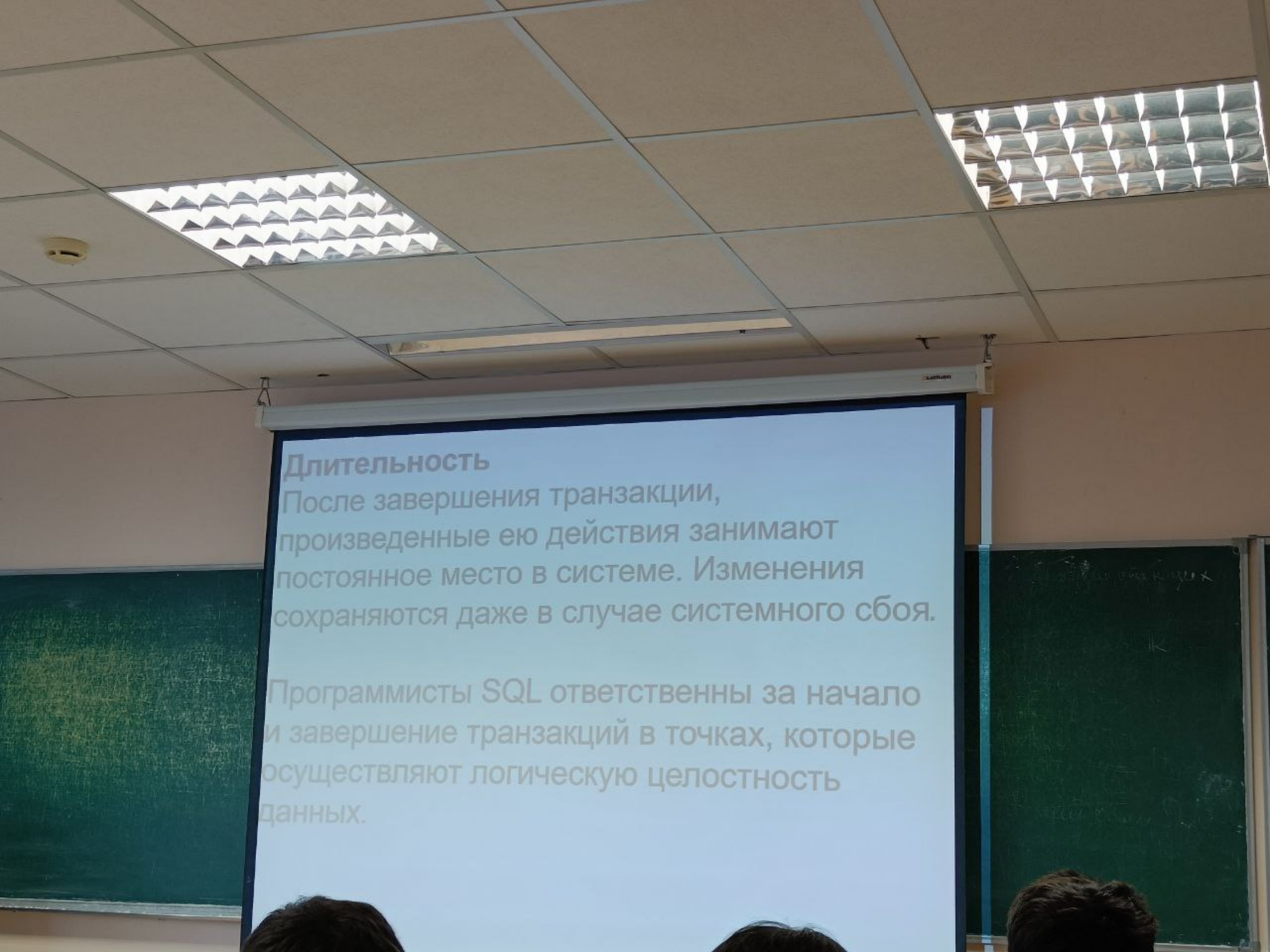

Транзакция является последовательностью операций, выполненных как одна логическая единица работы. Логическая единица работы должна обладать четырьмя свойствами, называемыми атомарностью, согласованностью, изоляцией и длительностью (ACID), чтобы называться транзакцией.

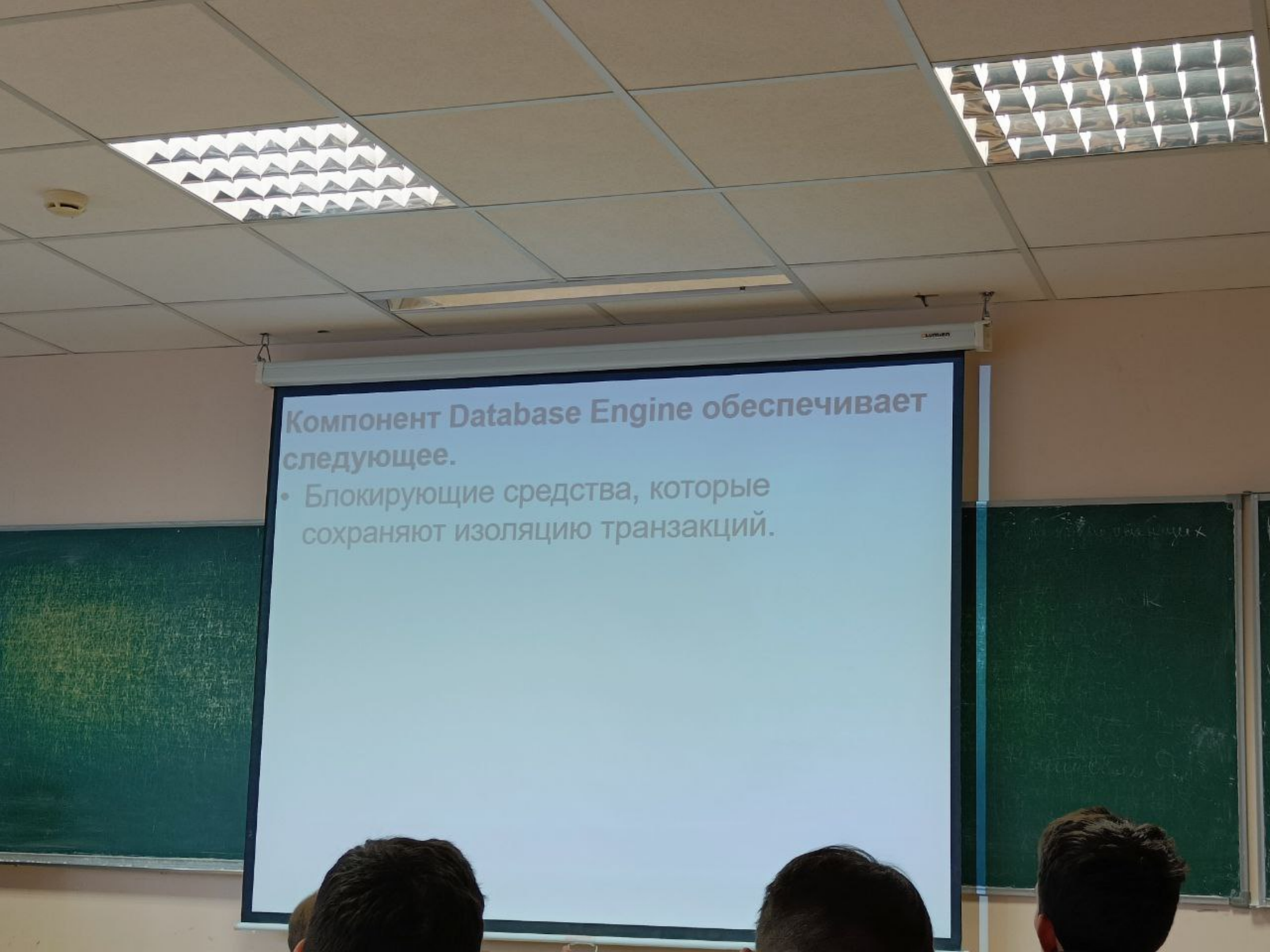

Атомарность
Транзакция должна быть атомарной единицей работы; должны быть выполнены либо все входящие в нее модификации данных, либо ни одна из них не должна быть выполнено.


Согласованность
По завершении, транзакция должна оставить все данные в согласованном состоянии. В реляционной базе данных к модификациям транзакции должны быть применены все правила для обеспечения целостности всех данных.

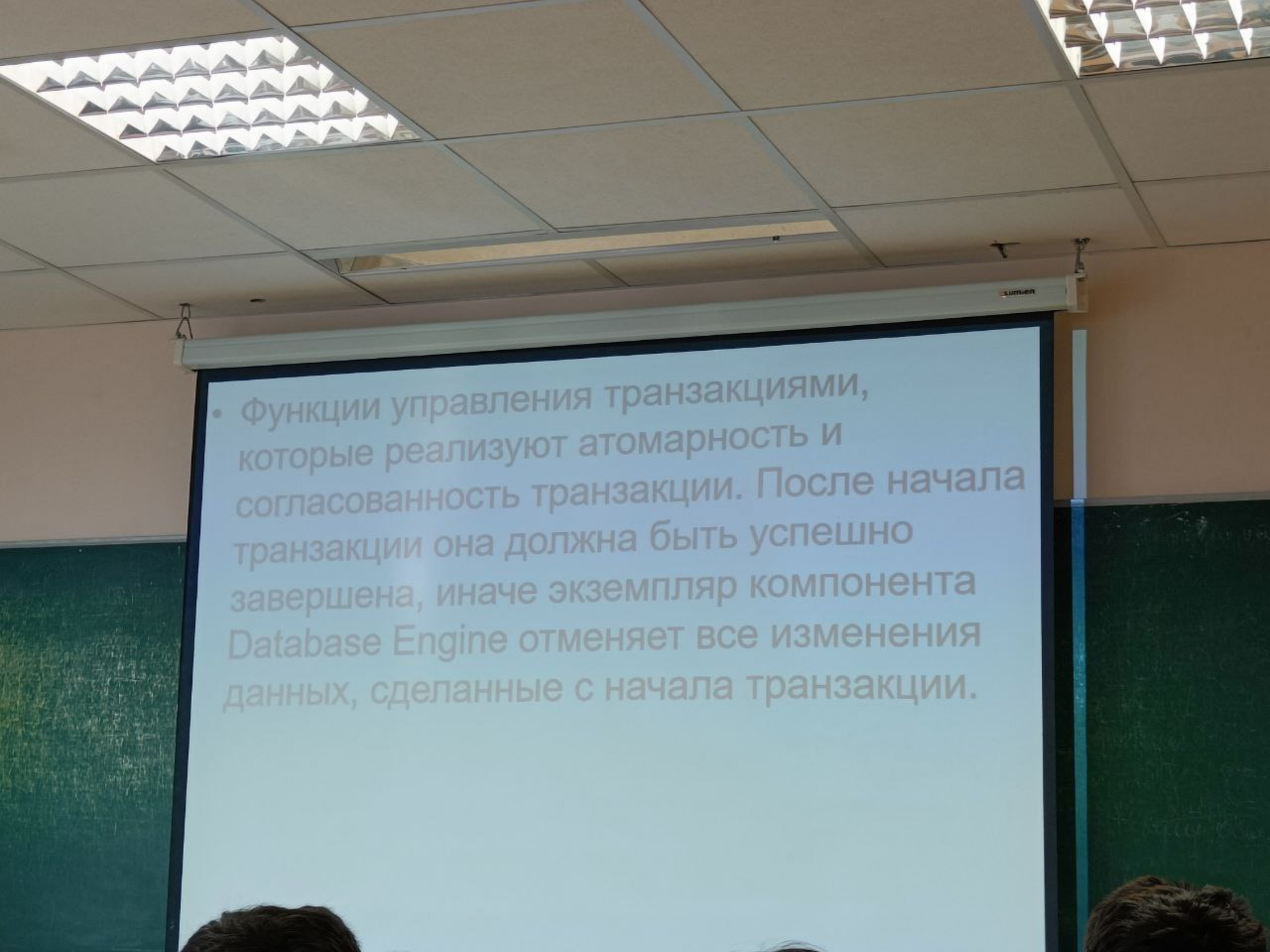

Изоляция
Модификации, выполняемые параллельной транзакцией, должны быть изолированы от любых модификаций, проводимых другими параллельными транзакциями. Транзакция распознает данные либо в состоянии, в котором они были до того, как другая параллельная транзакция изменила их, либо она распознает данные после того, как другая транзакция была завершена

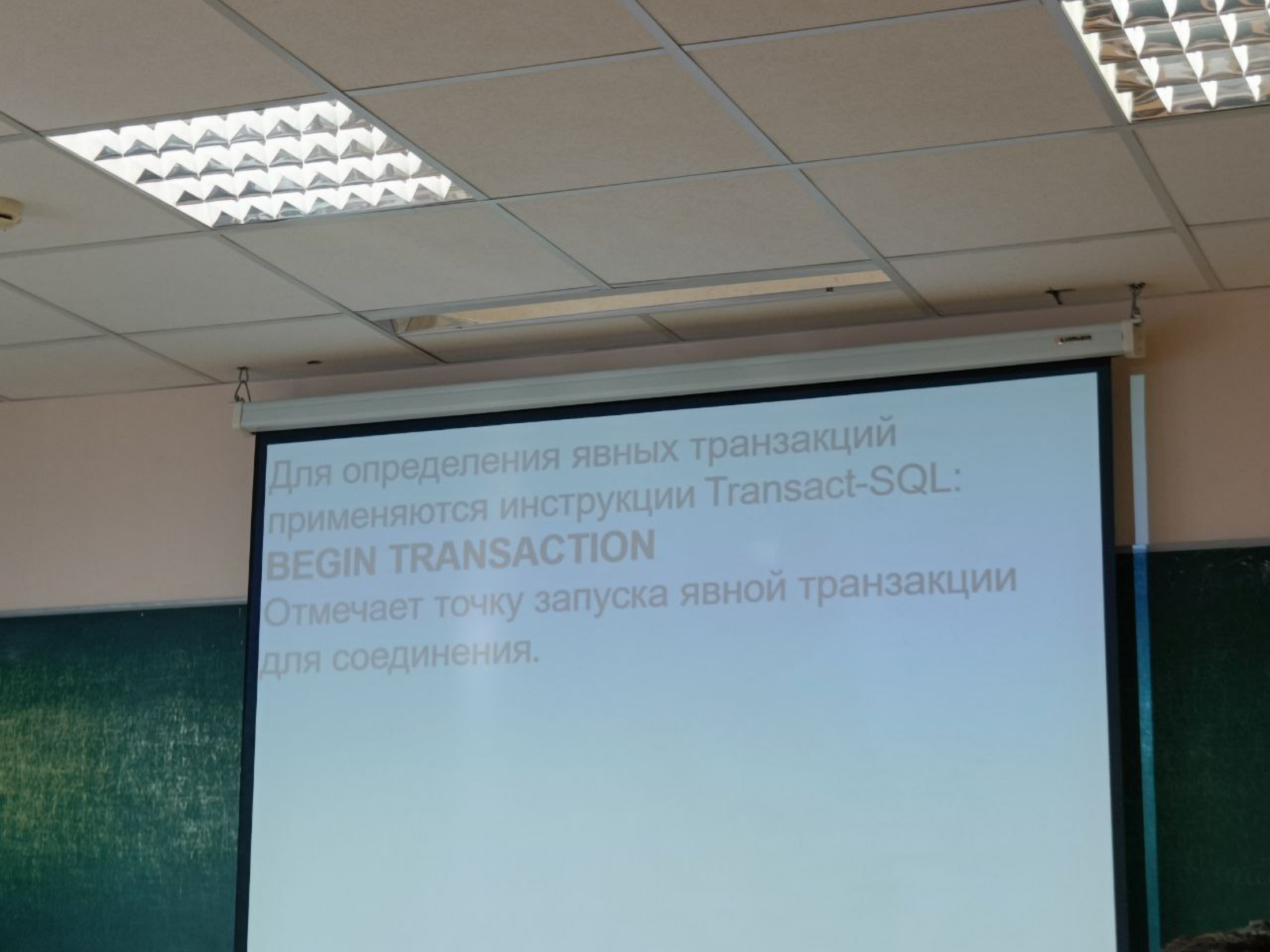

Но она не распознает промежуточное состояние. Это обеспечивает возможность перезагрузить начальные данные и воспроизвести серию транзакций, чтобы завершить работу с данными в том же самом состоянии, в котором они были после выполнения исходных транзакций.

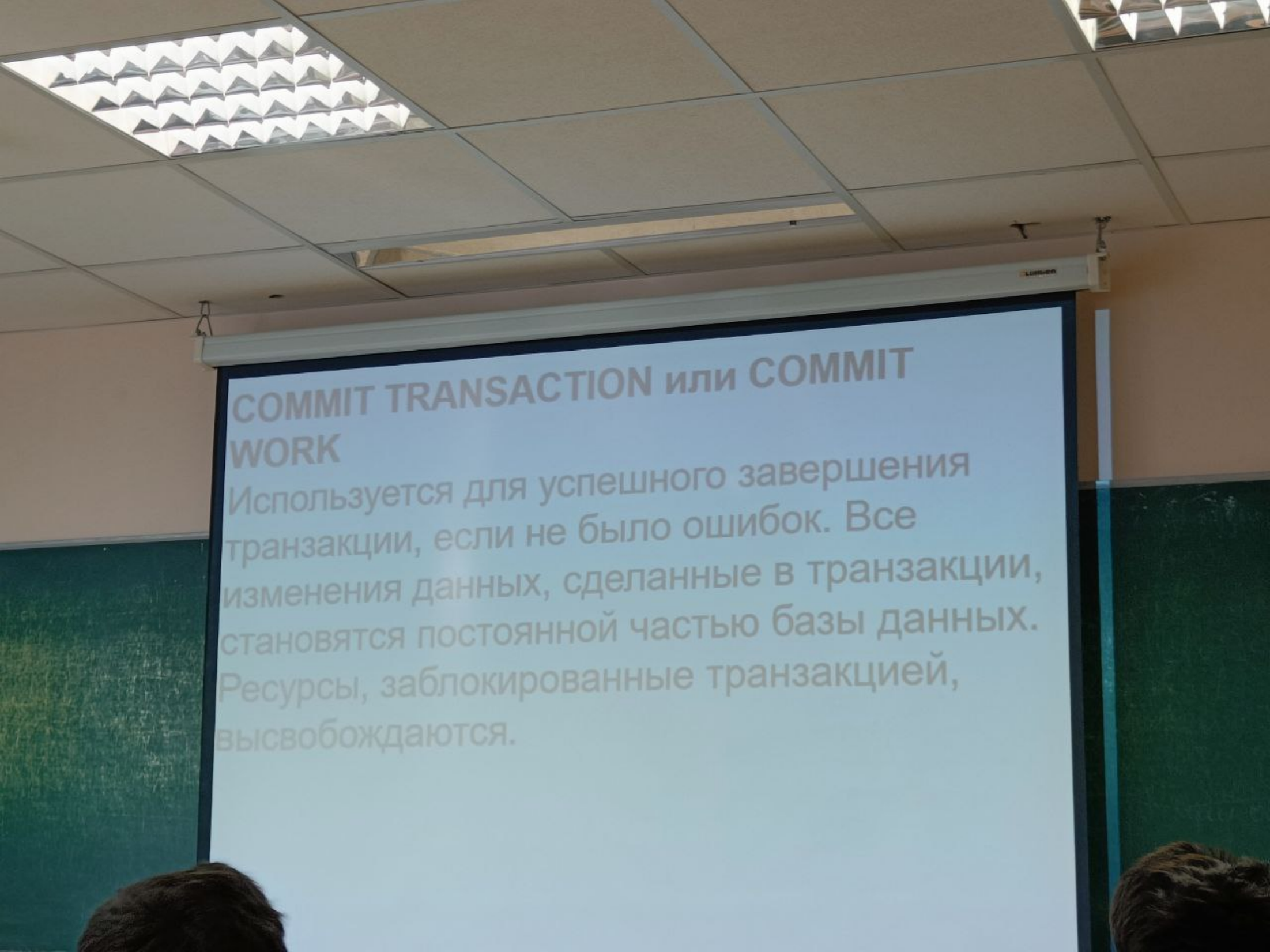

Длительность
После завершения транзакции, произведенные ею действия занимают постоянное место в системе. Изменения сохраняются даже в случае системного сбоя.
Программисты SQL ответственны за начало и завершение транзакций в точках, которые осуществляют логическую целостность данных.

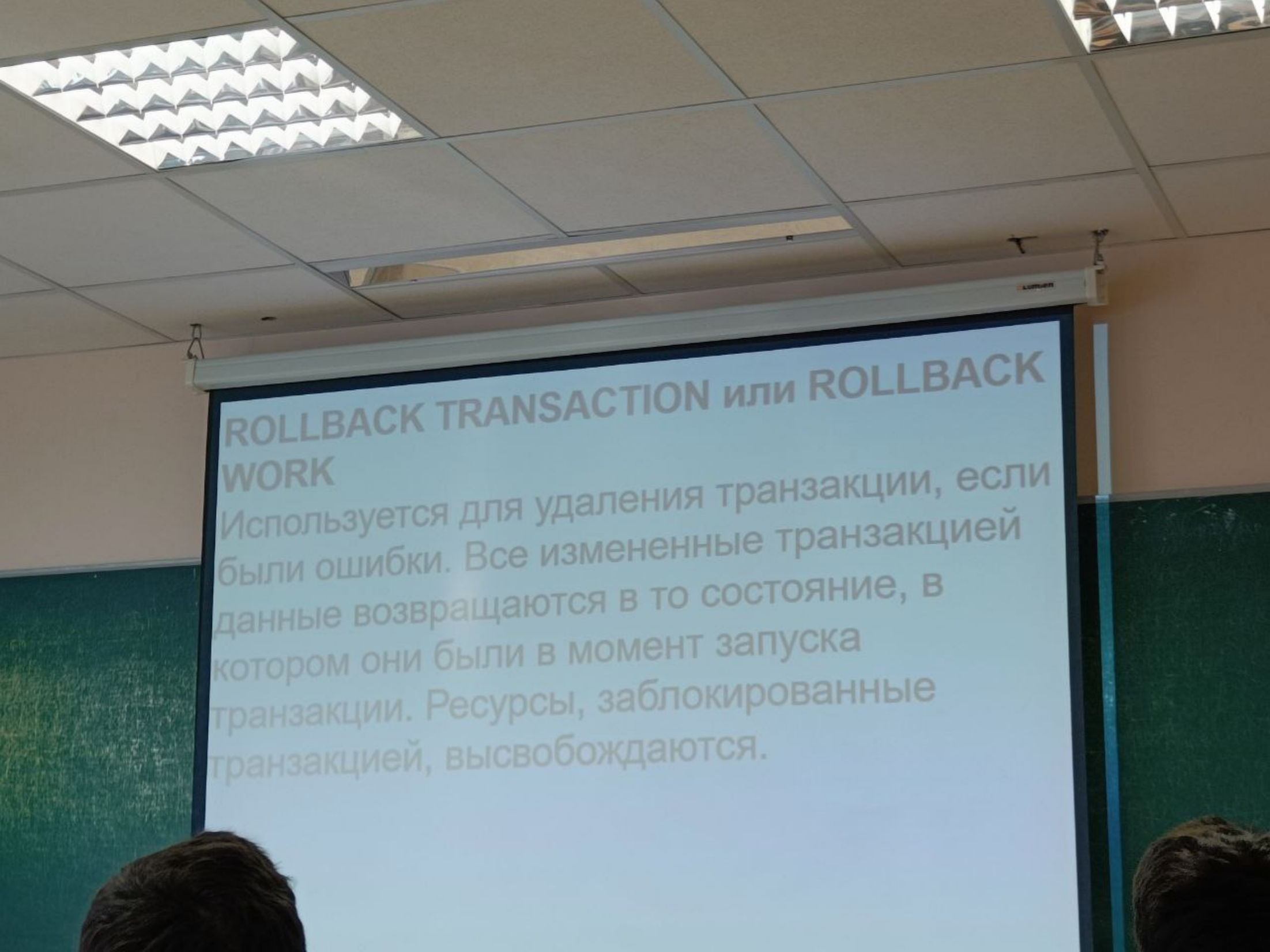

Компонент Database Engine обеспечивает следующее.
• Блокирующие средства, которые сохраняют изоляцию транзакций.

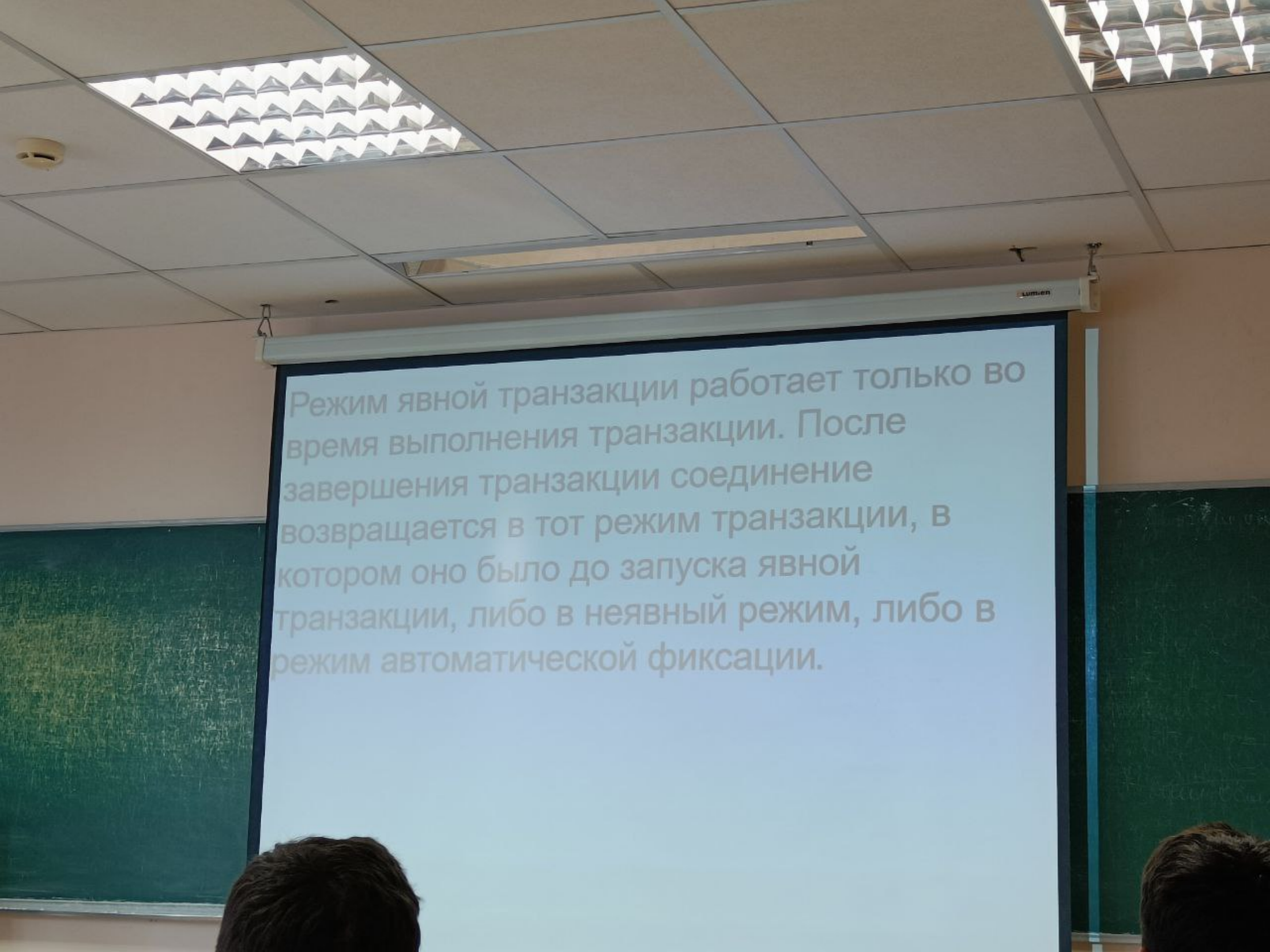
- Регистрирующие средства, которые гарантируют длительность транзакции. Даже если в оборудовании сервера, операционной системе или экземпляре Database Engine произойдет сбой, после перезапуска экземпляр использует журналы транзакций для автоматического отката любых незавершенных транзакций до момента сбоя системы.

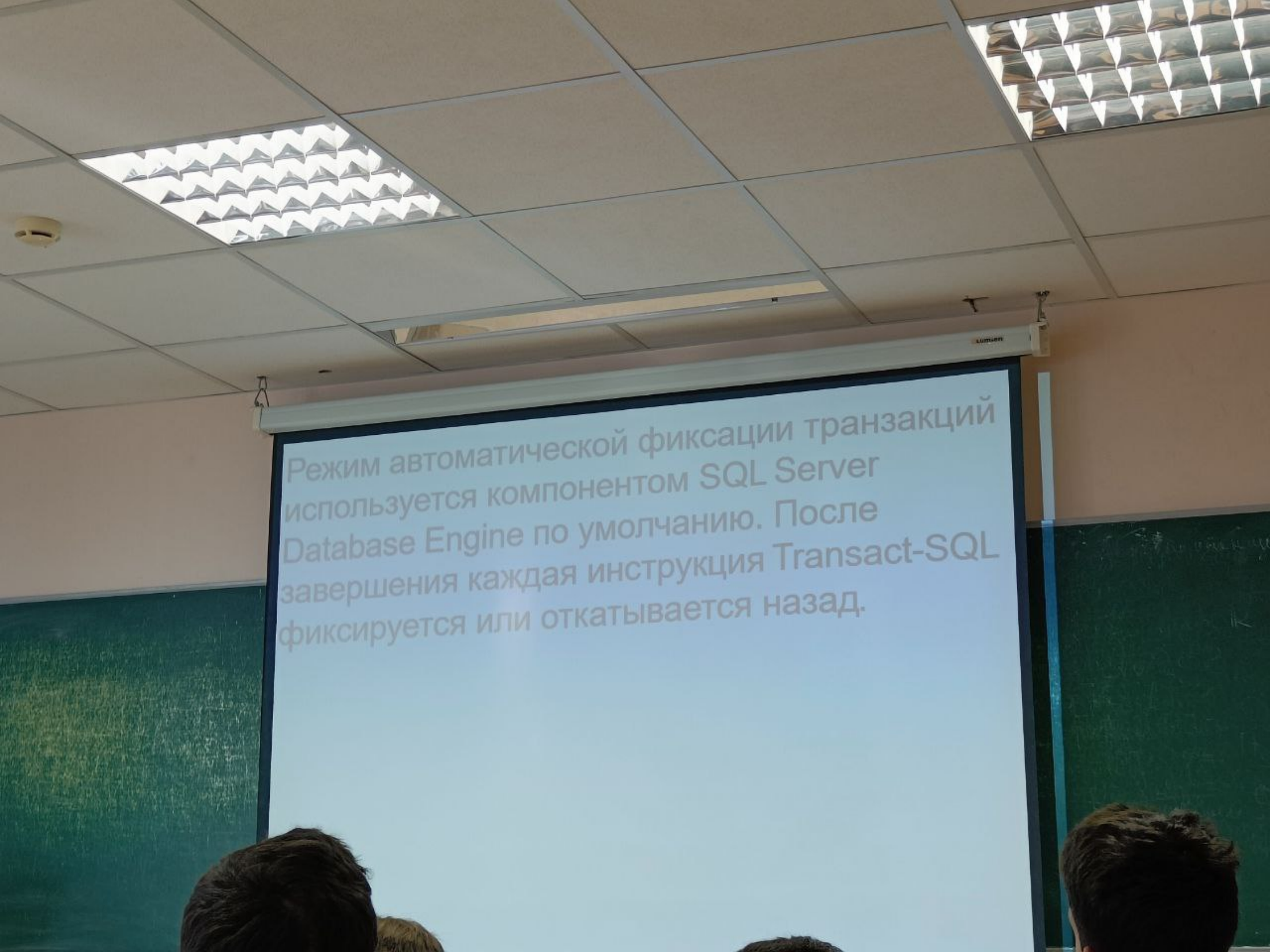

• Функции управления транзакциями, которые реализуют атомарность и согласованность транзакции. После начала транзакции она должна быть успешно завершена, иначе экземпляр компонента Database Engine отменяет все изменения данных, сделанные с начала транзакции.

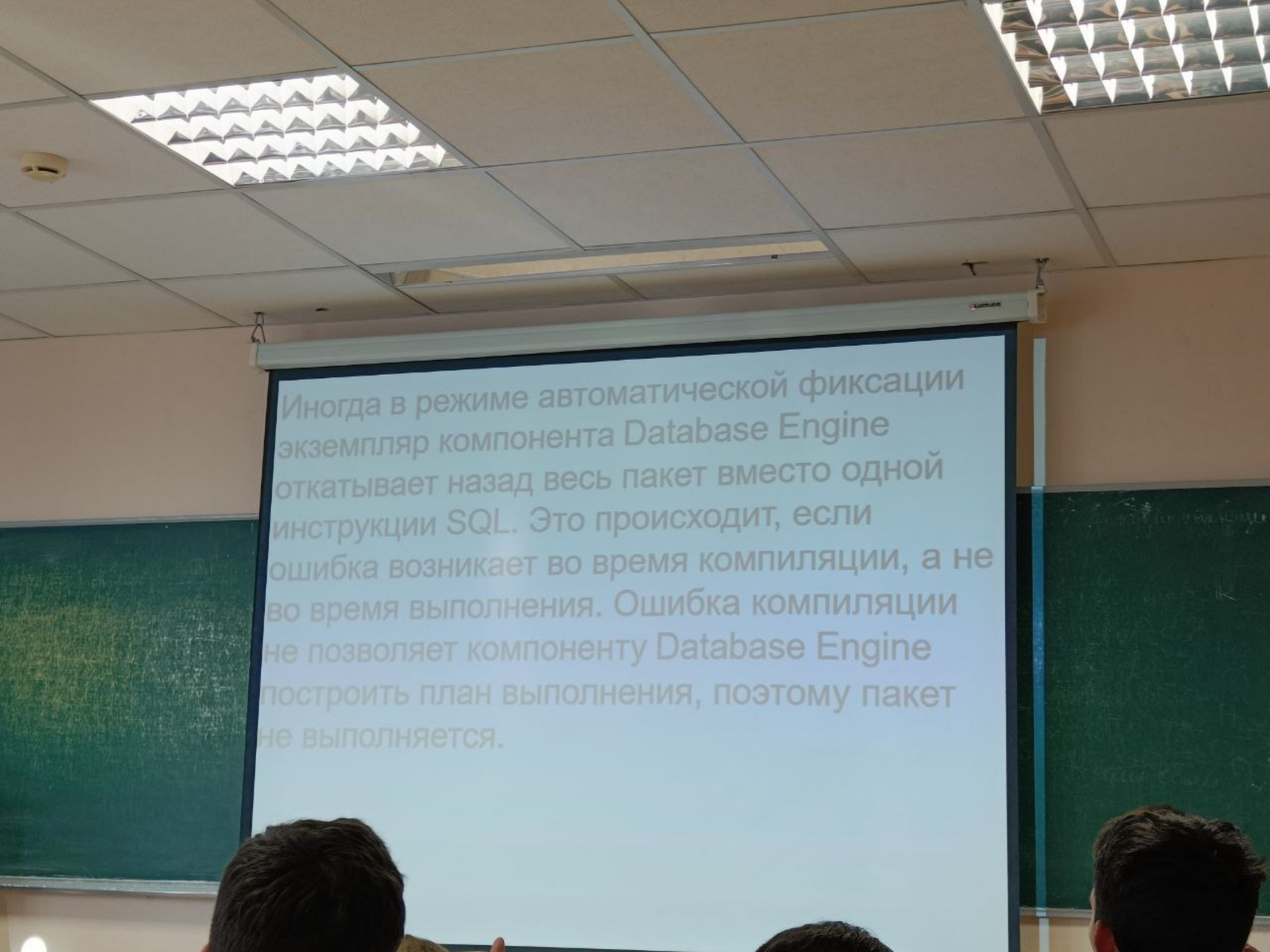

Для определения явных транзакций
применяются инструкции Transact-SQL:
BEGIN TRANSACTION
Отмечает точку запуска явной транзакции
для соединения.


COMMIT TRANSACTION или COMMIT WORK
Используется для успешного завершения транзакции, если не было ошибок. Все изменения данных, сделанные в транзакции, становятся постоянной частью базы данных. Ресурсы, заблокированные транзакцией, высвобождаются.


ROLLBACK TRANSACTION или ROLLBACK WORK
Используется для удаления транзакции, если были ошибки. Все измененные транзакцией данные возвращаются в то состояние, в котором они были в момент запуска транзакции. Ресурсы, заблокированные транзакцией, высвобождаются.

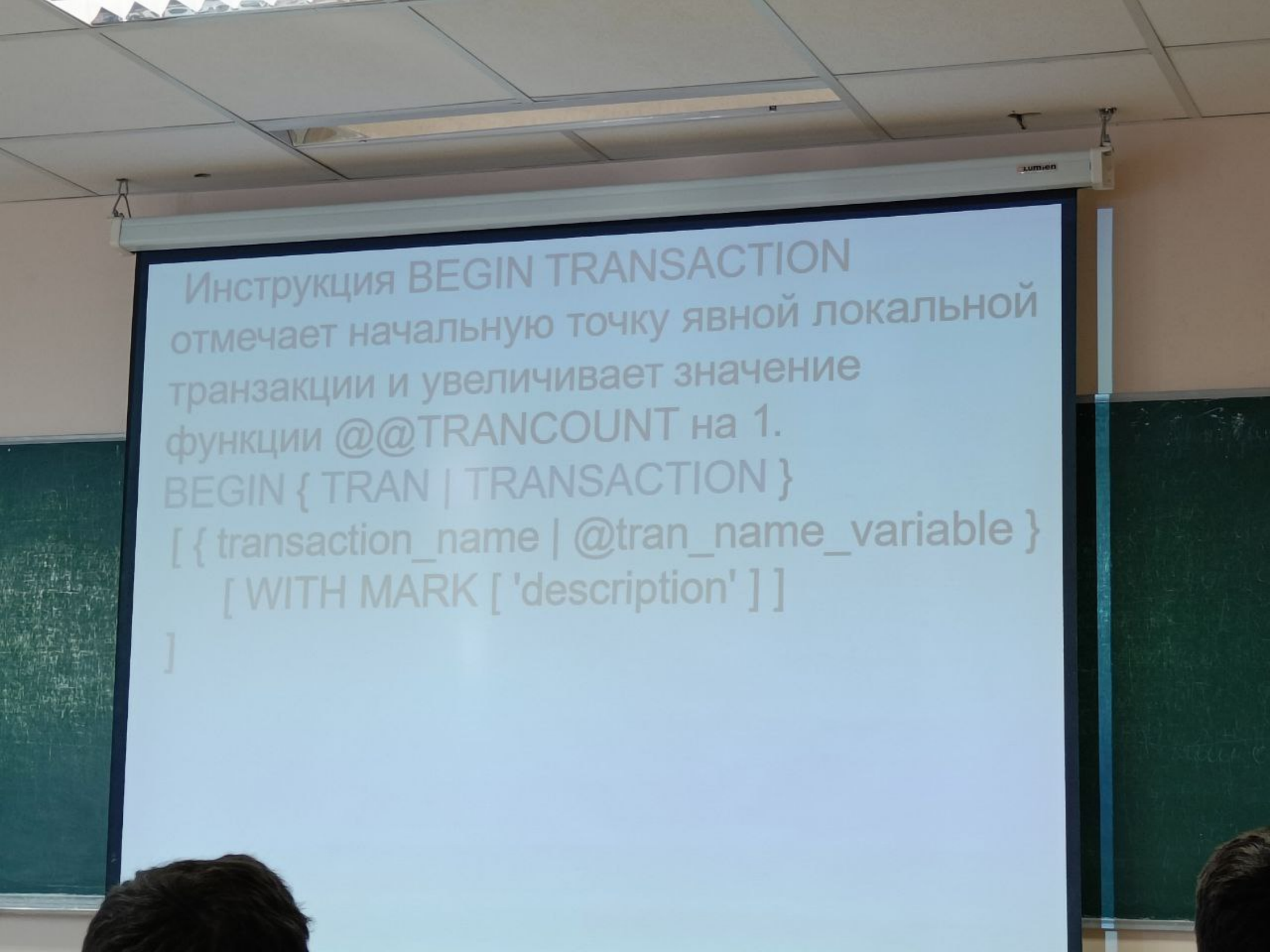

Режим явной транзакции работает только во время выполнения транзакции. После завершения транзакции соединение возвращается в тот режим транзакции, в котором оно было до запуска явной транзакции, либо в неявный режим, либо в режим автоматической фиксации.

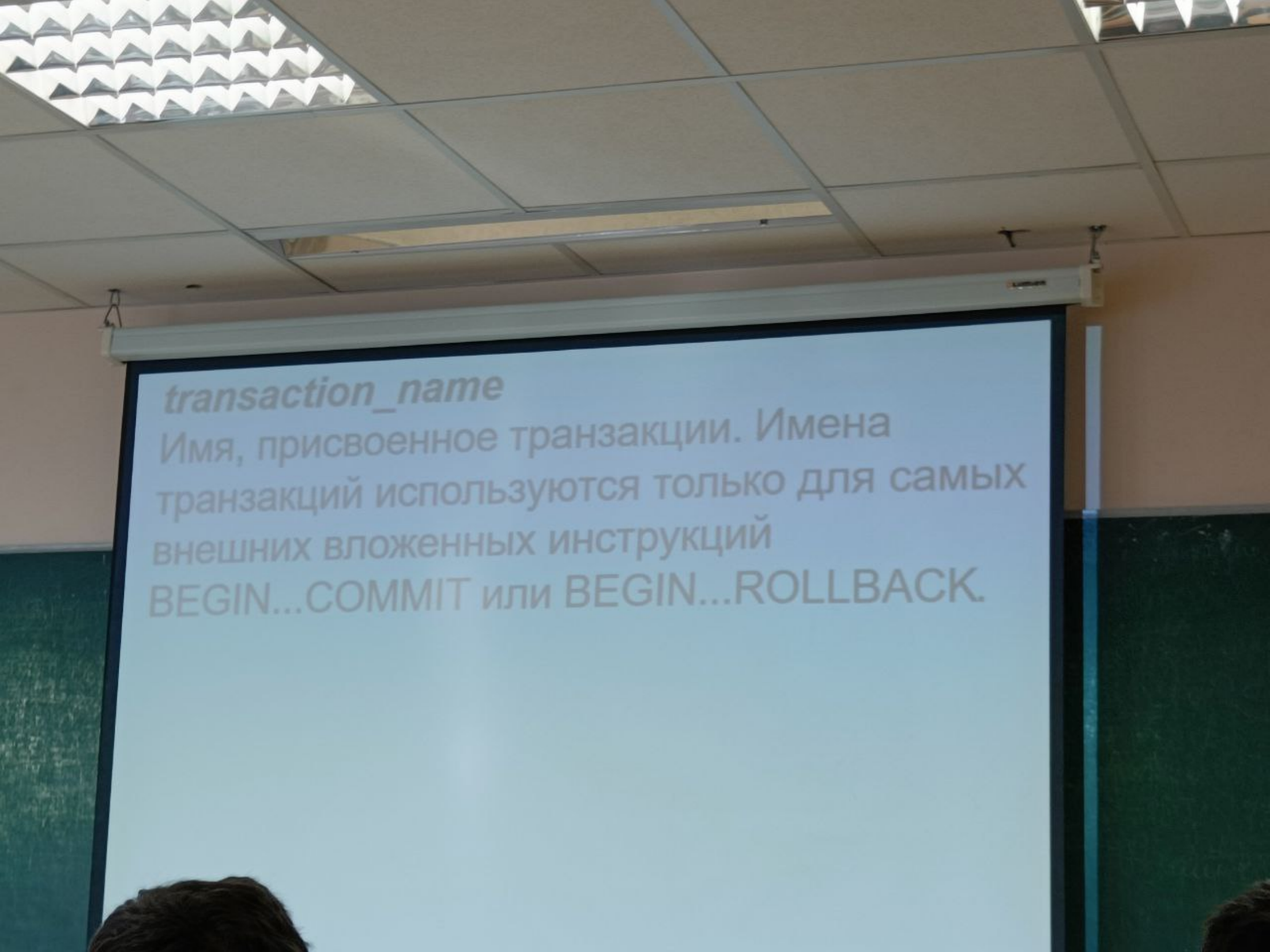

Режим автоматической фиксации транзакций
используется компонентом SQL Server
Database Engine по умолчанию. После
завершения каждая инструкция Transact-SQL
фиксируется или откатывается назад.

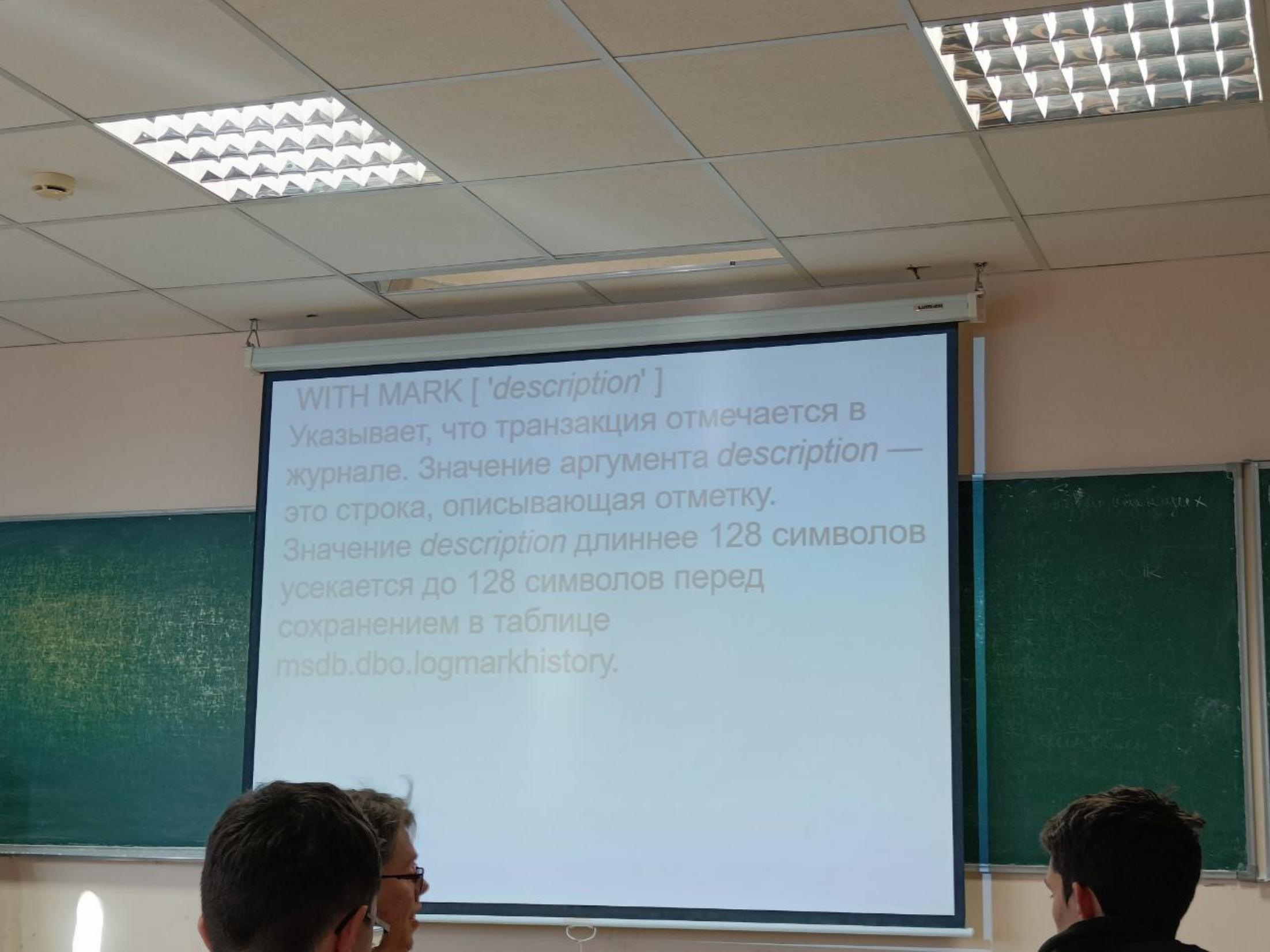

Иногда в режиме автоматической фиксации
экземпляр компонента Database Engine
откатывает назад весь пакет вместо одной
инструкции SQL. Это происходит, если
ошибка возникает во время компиляции, а не
во время выполнения. Ошибка компиляции
не позволяет компоненту Database Engine
построить план выполнения, поэтому пакет
не выполняется.

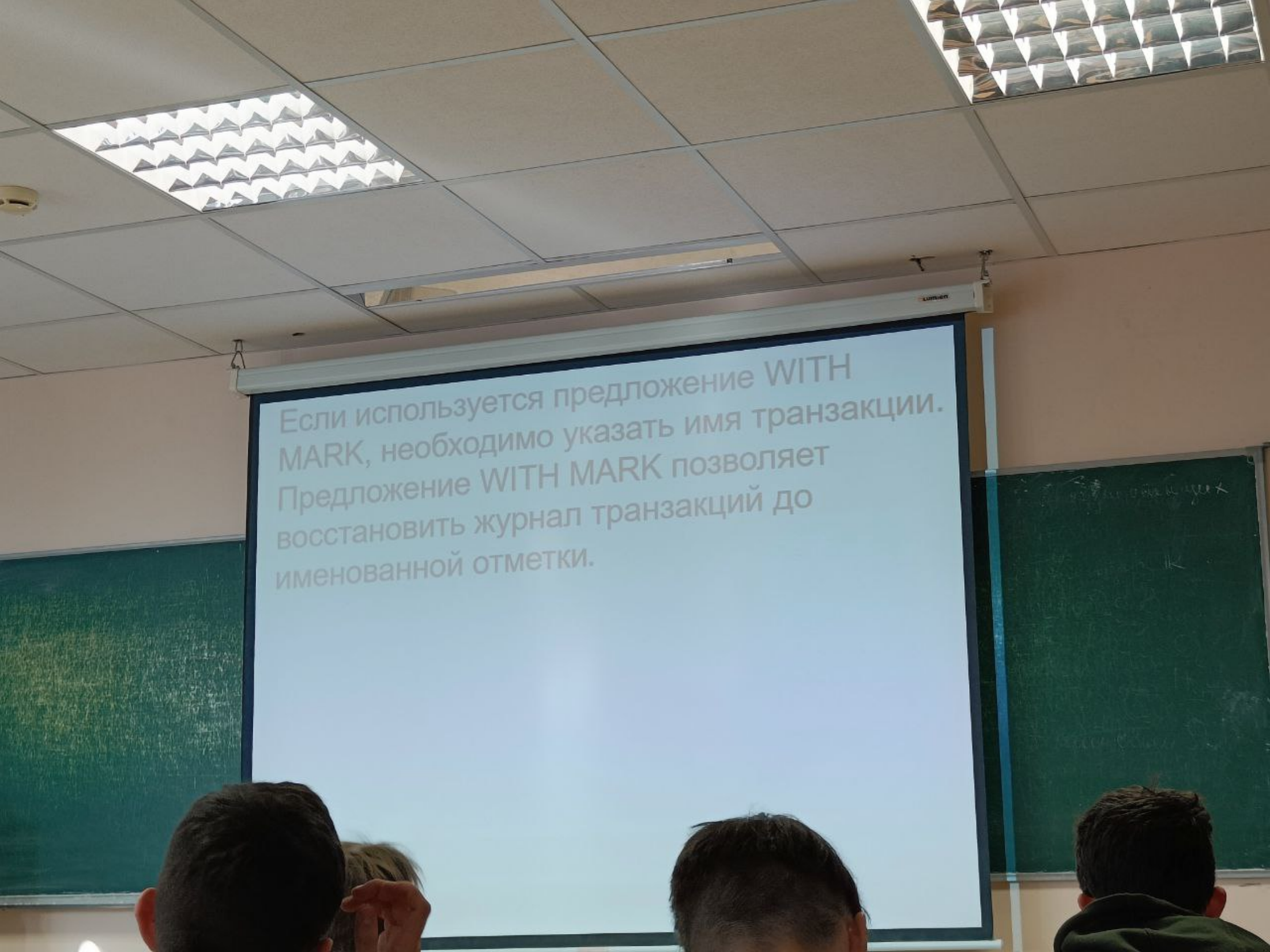
После установления на соединении режима неявных транзакций экземпляр компонента Database Engine автоматически запускает транзакцию, если вначале выполняет любую из следующих инструкций:

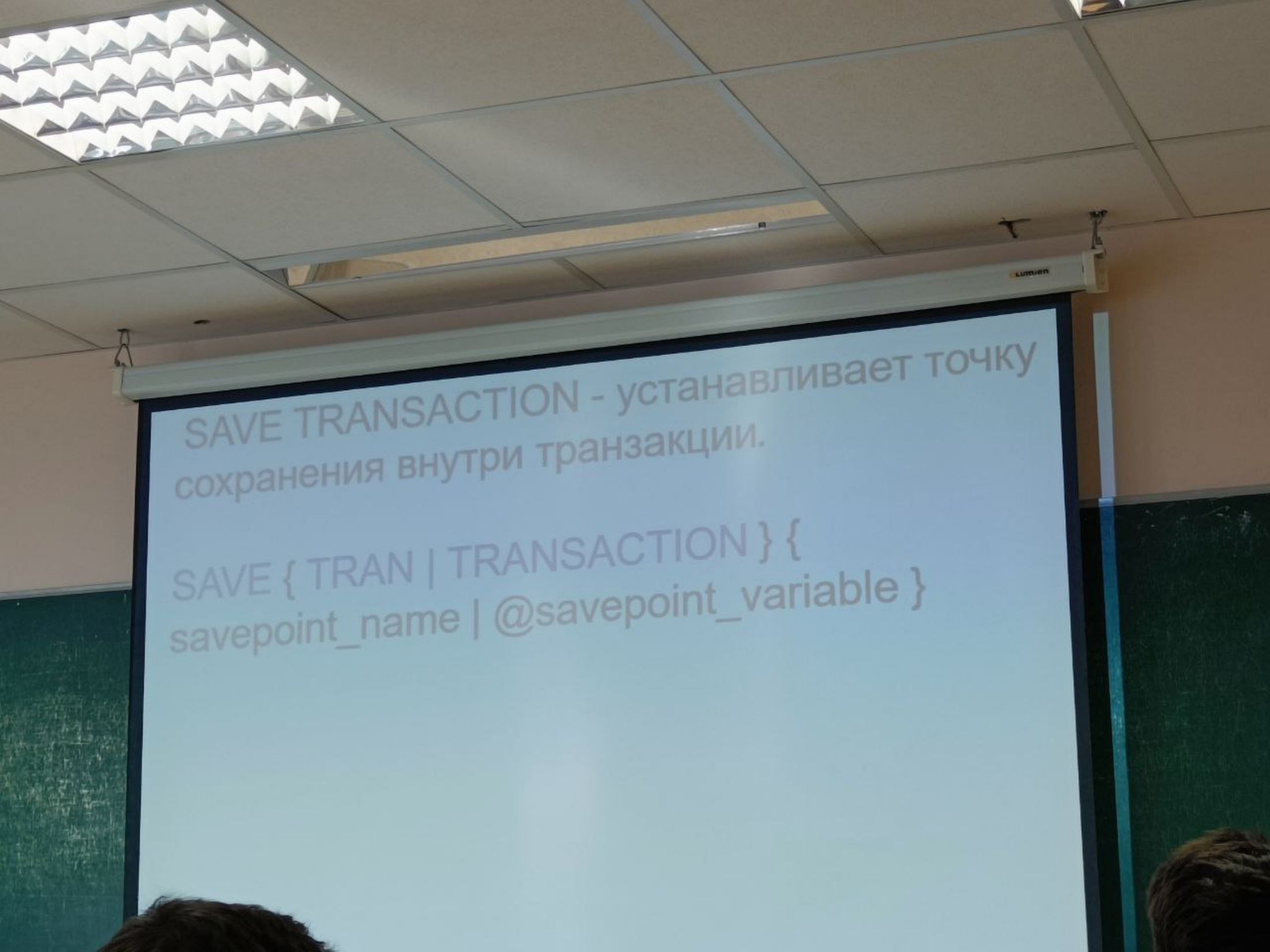
ALTER TABLE	INSERT
CREATE	OPEN
DELETE	REVOKE
DROP	SELECT
FETCH	TRUNCATE

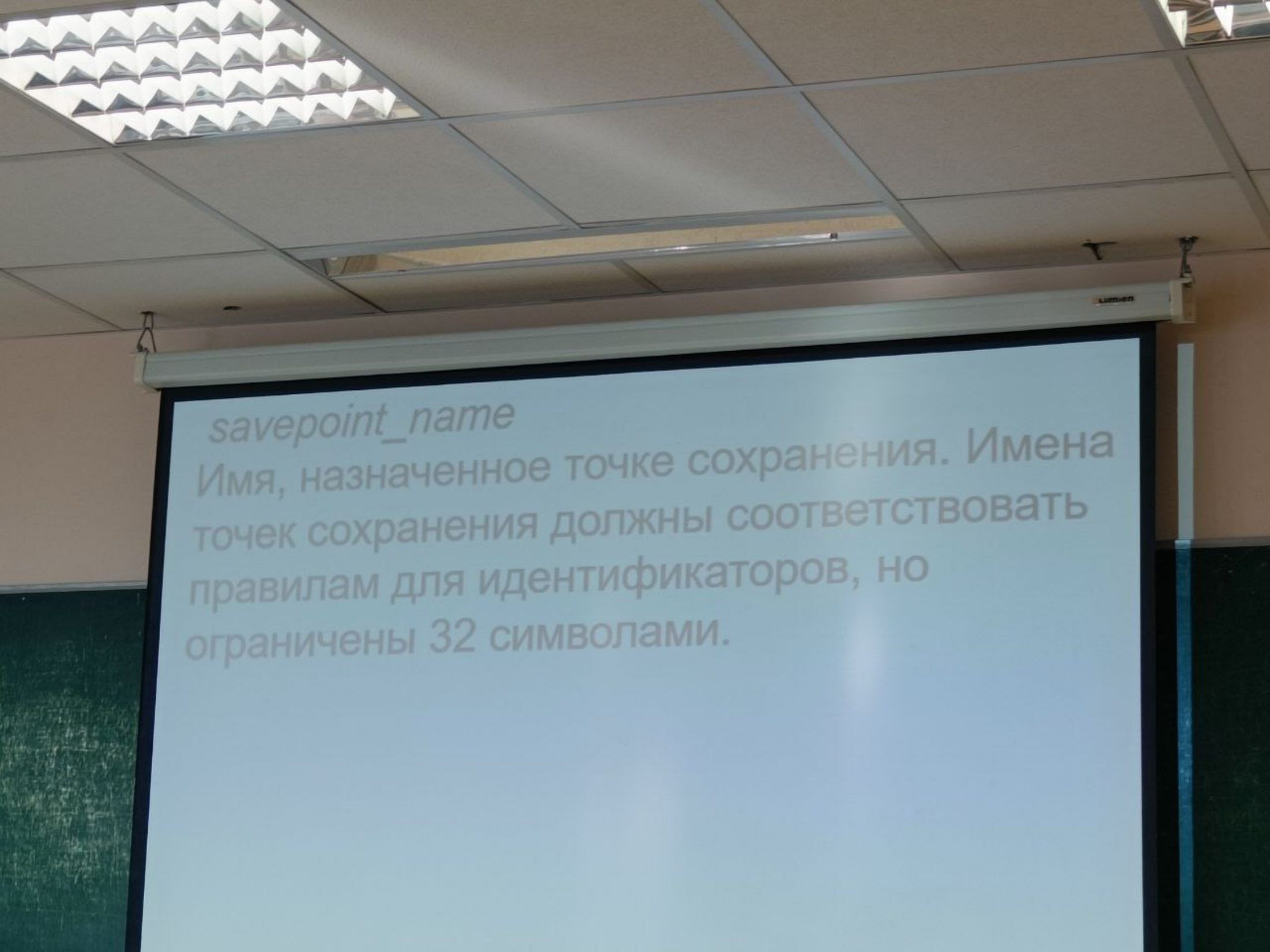

Инструкция BEGIN TRANSACTION
отмечает начальную точку явной локальной
транзакции и увеличивает значение
функции @@TRANCOUNT на 1.
BEGIN { TRAN | TRANSACTION }
[{ transaction_name | @tran_name_variable }
[WITH MARK ['description']]
]


transaction_name
Имя, присвоенное транзакции. Имена транзакций используются только для самых внешних вложенных инструкций BEGIN...COMMIT или BEGIN...ROLLBACK.


WITH MARK ['description']
Указывает, что транзакция отмечается в журнале. Значение аргумента description — это строка, описывающая отметку. Значение description длиннее 128 символов усекается до 128 символов перед сохранением в таблице msdb.dbo.logmarkhistory.

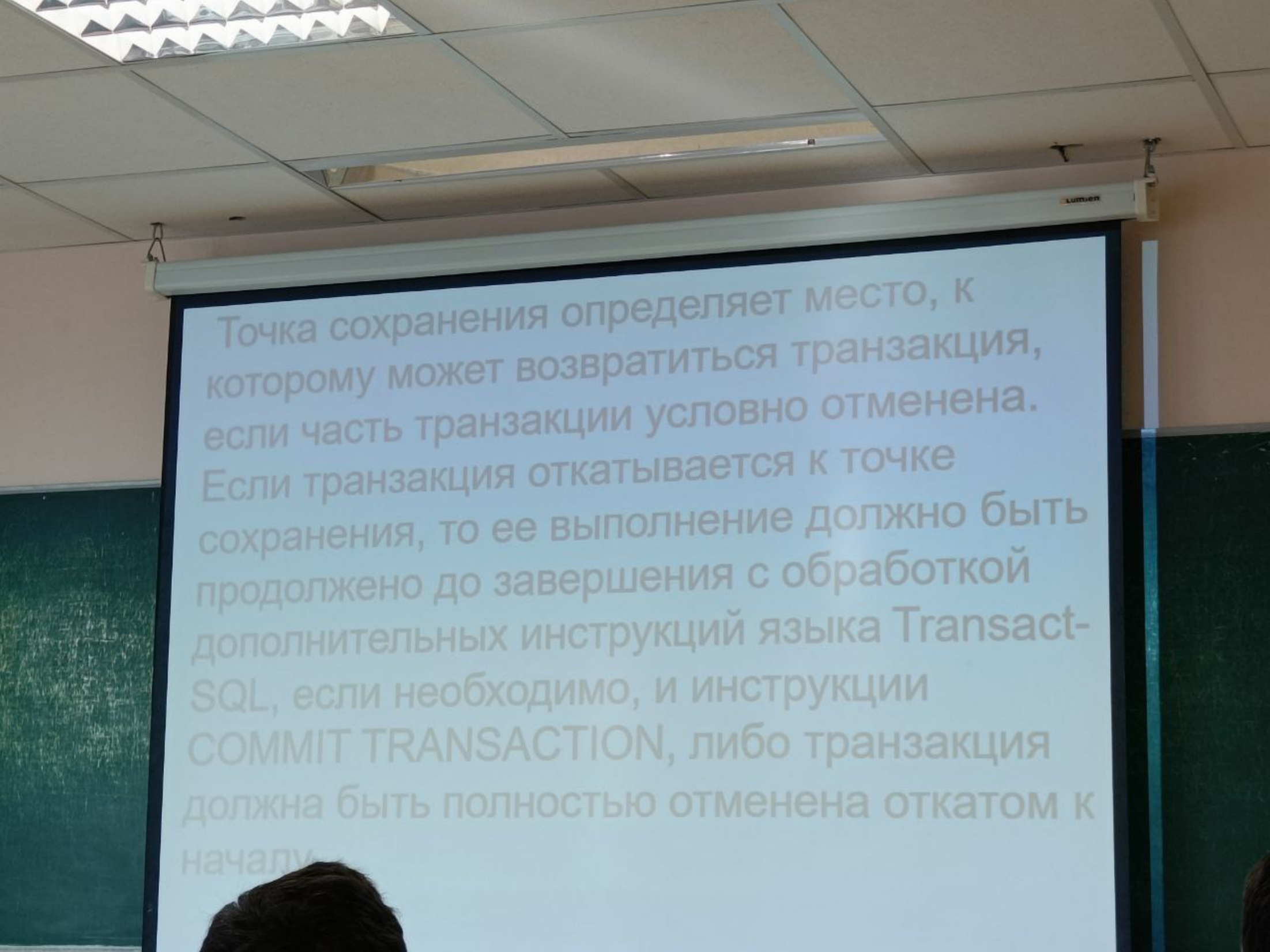

Если используется предложение WITH MARK, необходимо указать имя транзакции. Предложение WITH MARK позволяет восстановить журнал транзакций до именованной отметки.

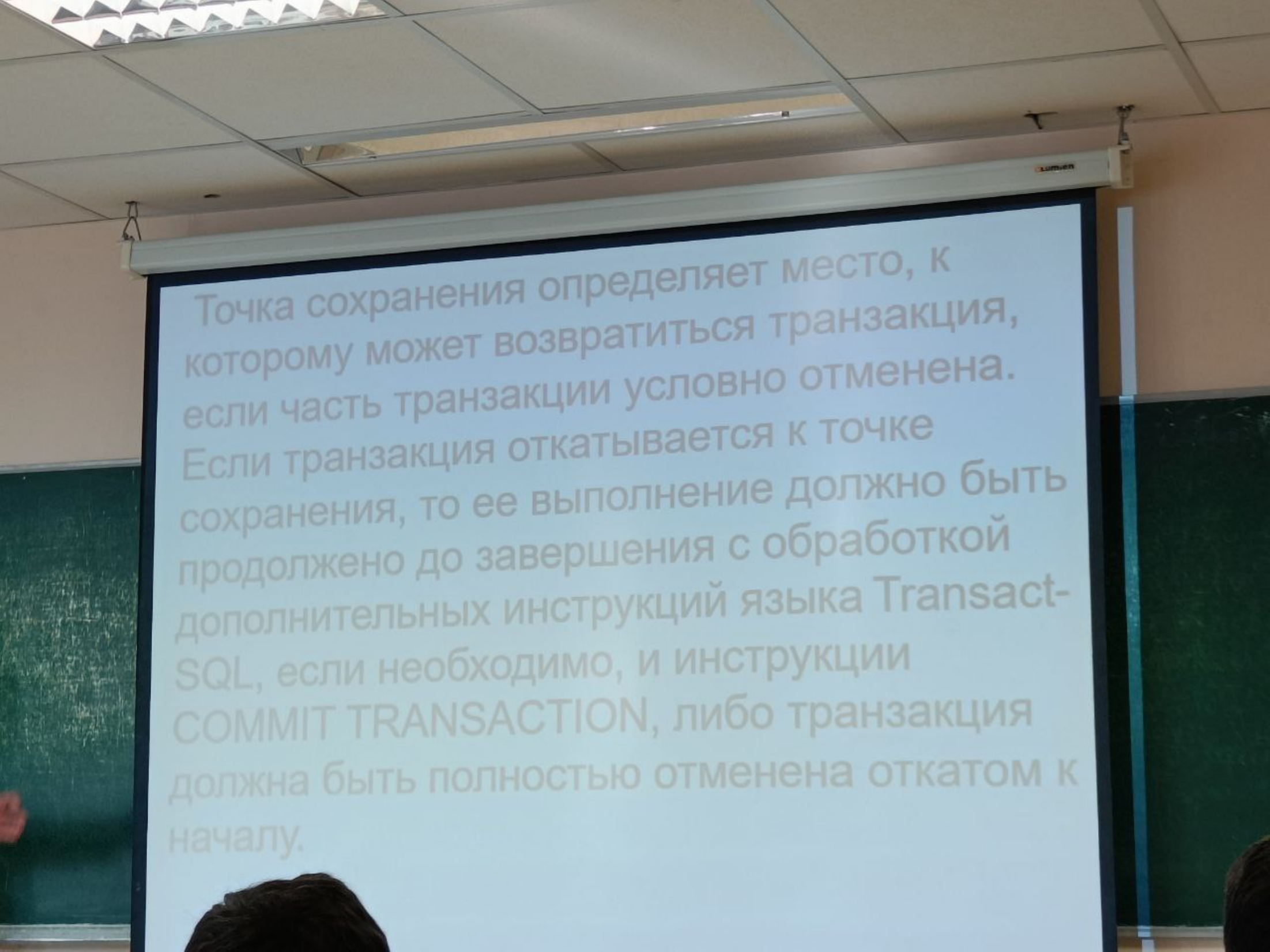

SAVE TRANSACTION - устанавливает точку сохранения внутри транзакции.
SAVE { TRAN | TRANSACTION } {
savepoint_name | @savepoint_variable }

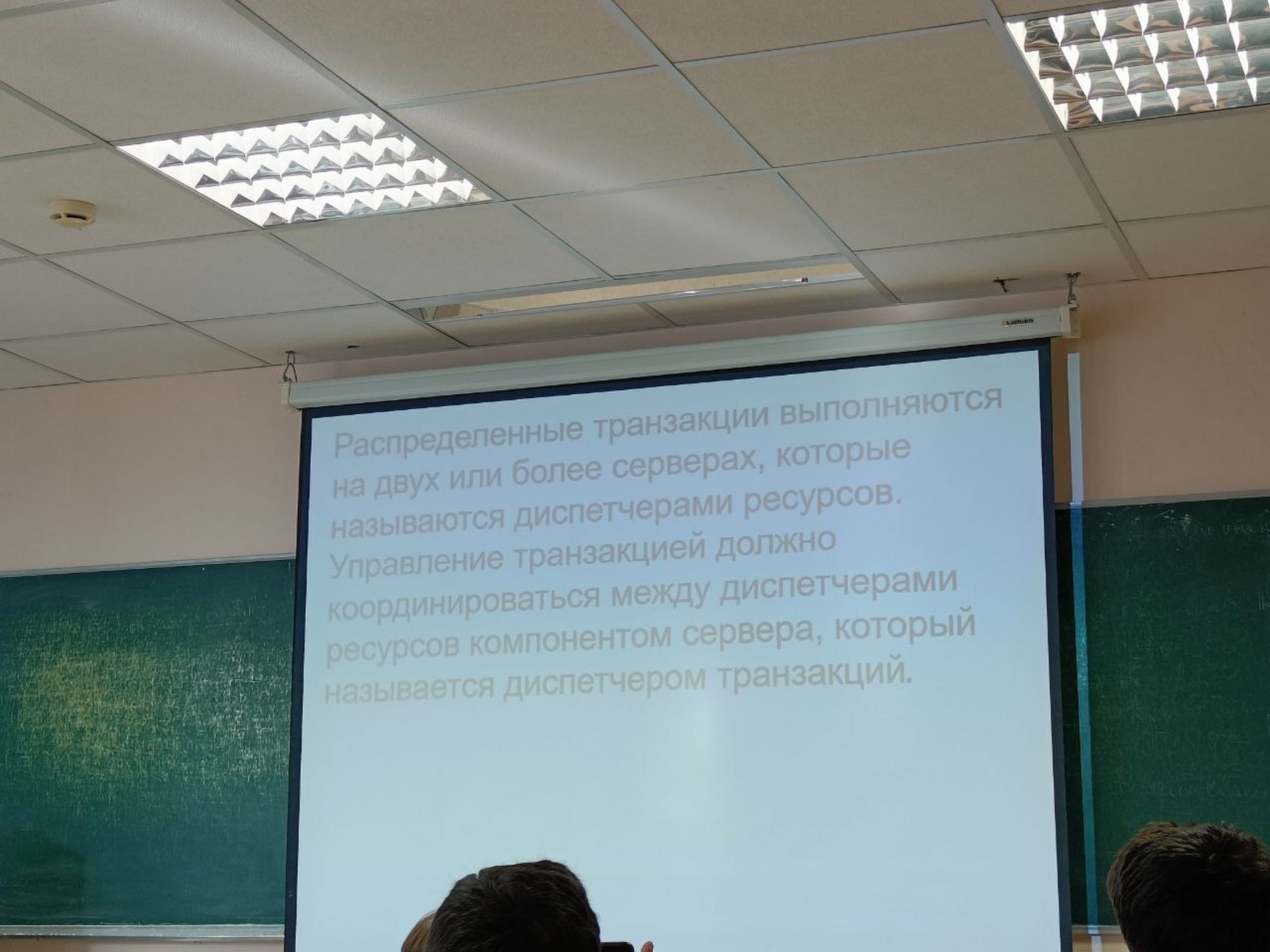

savepoint_name
Имя, назначенное точке сохранения. Имена точек сохранения должны соответствовать правилам для идентификаторов, но ограничены 32 символами.

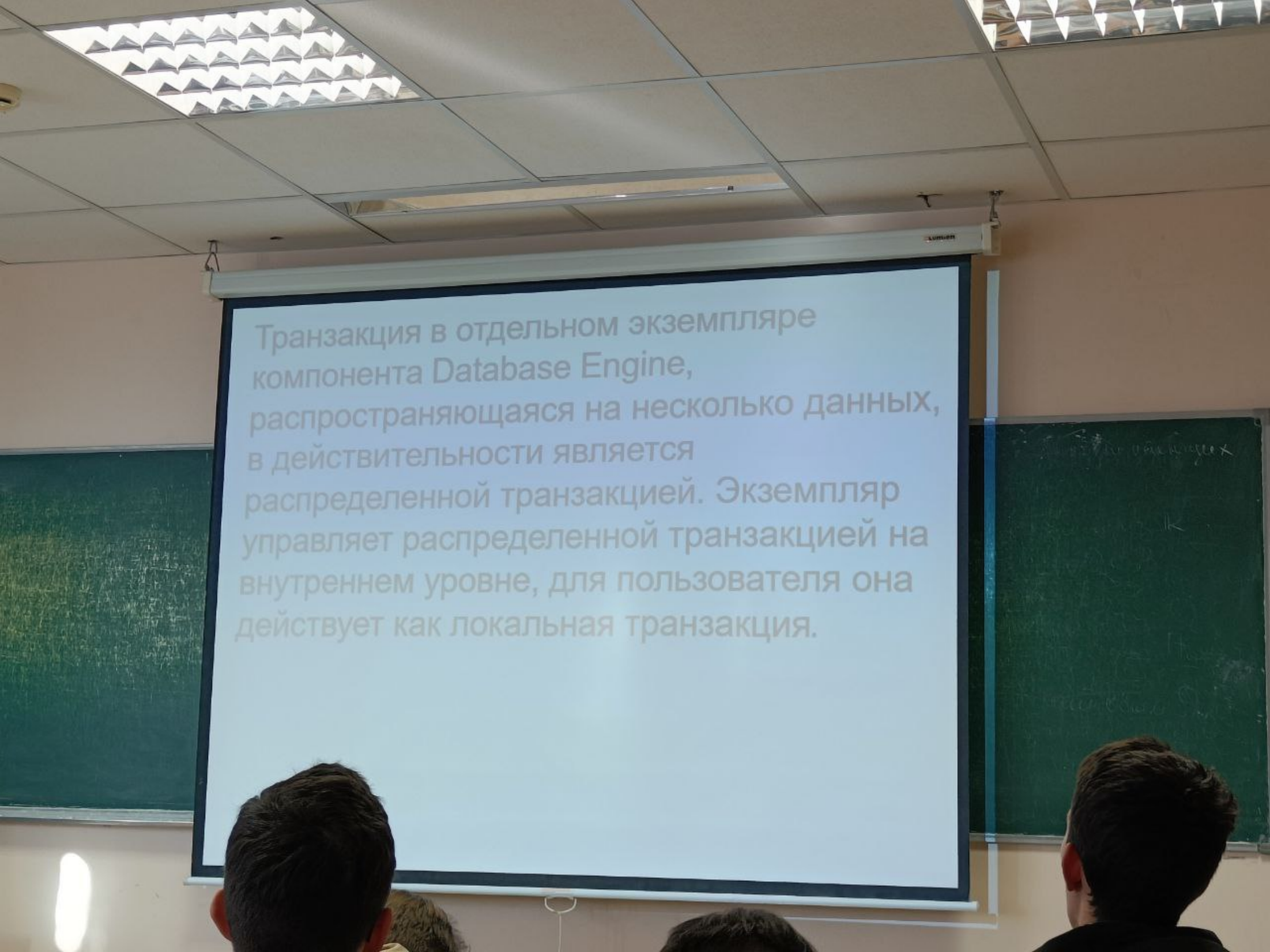
@savepoint_variable

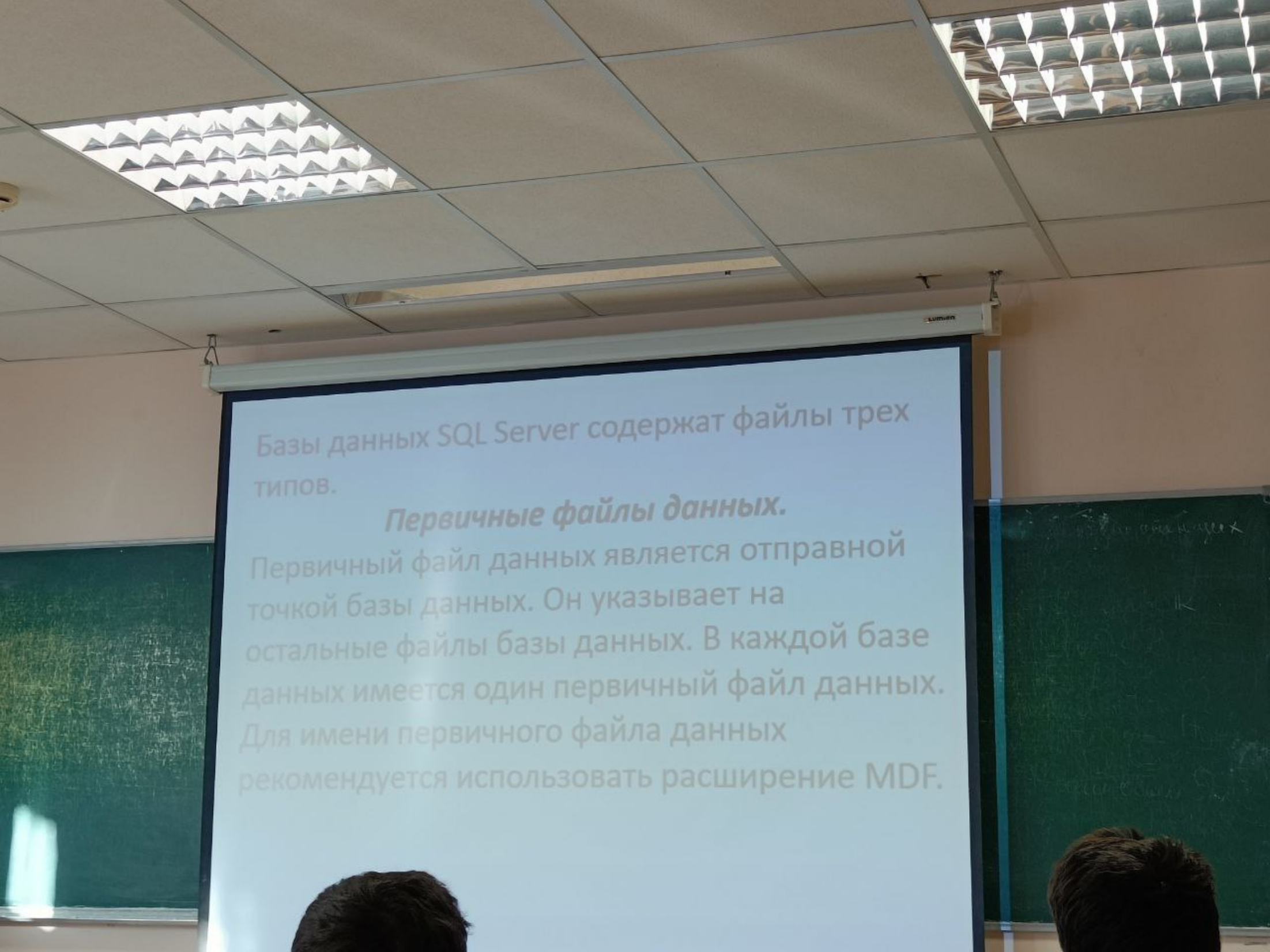
Имя пользовательской переменной, содержащей допустимое имя точки сохранения. Переменная должна иметь тип данных char, varchar, nchar или nvarchar. В переменную может быть передано более 32 символов, но будут использованы только 32 первых символа.

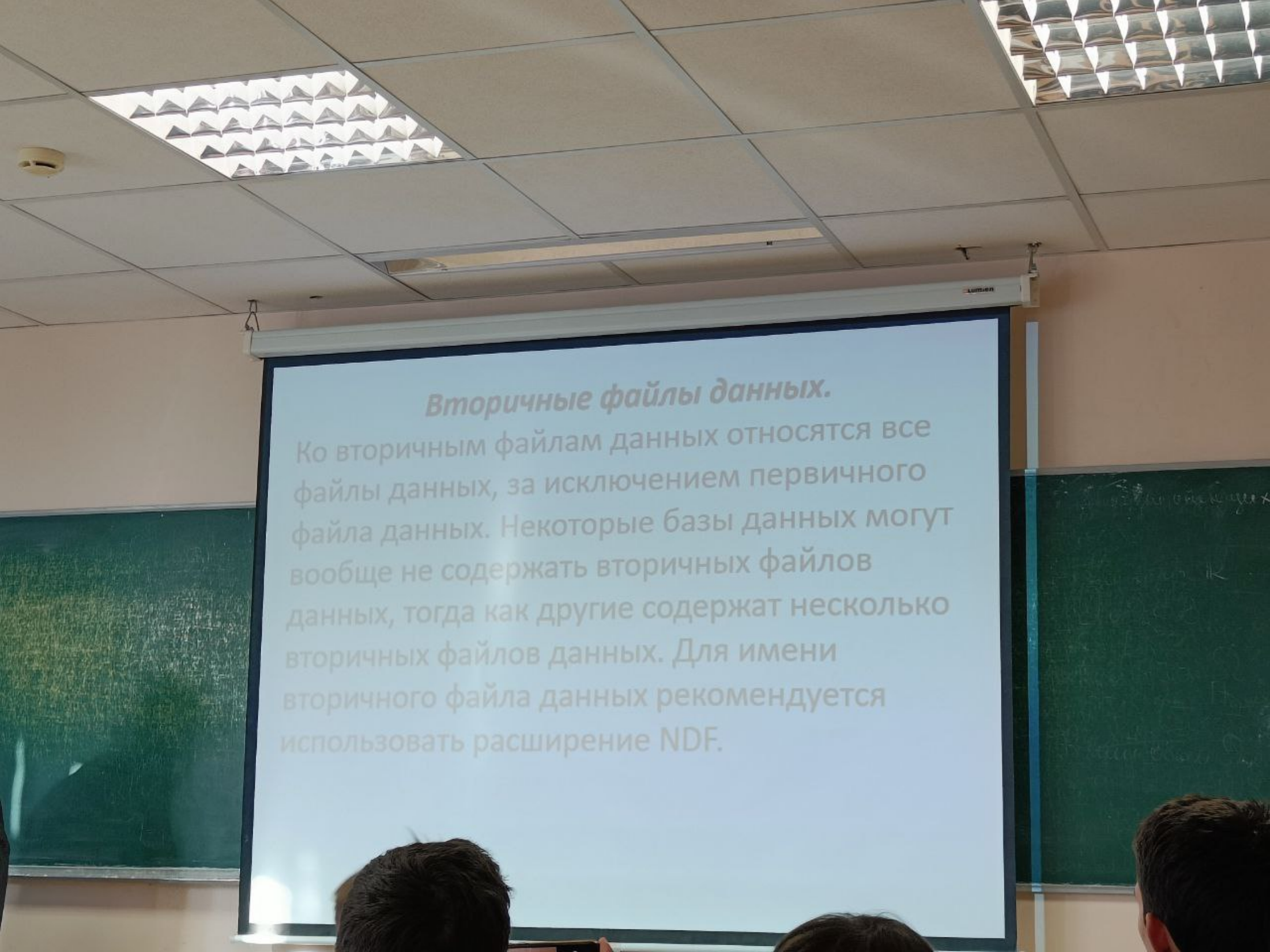

Точка сохранения определяет место, к
которому может возвратиться транзакция,
если часть транзакции условно отменена.
Если транзакция откатывается к точке
сохранения, то ее выполнение должно быть
продолжено до завершения с обработкой
дополнительных инструкций языка Transact-
SQL, если необходимо, и инструкции
COMMIT TRANSACTION, либо транзакция
должна быть полностью отменена откатом к
началу.

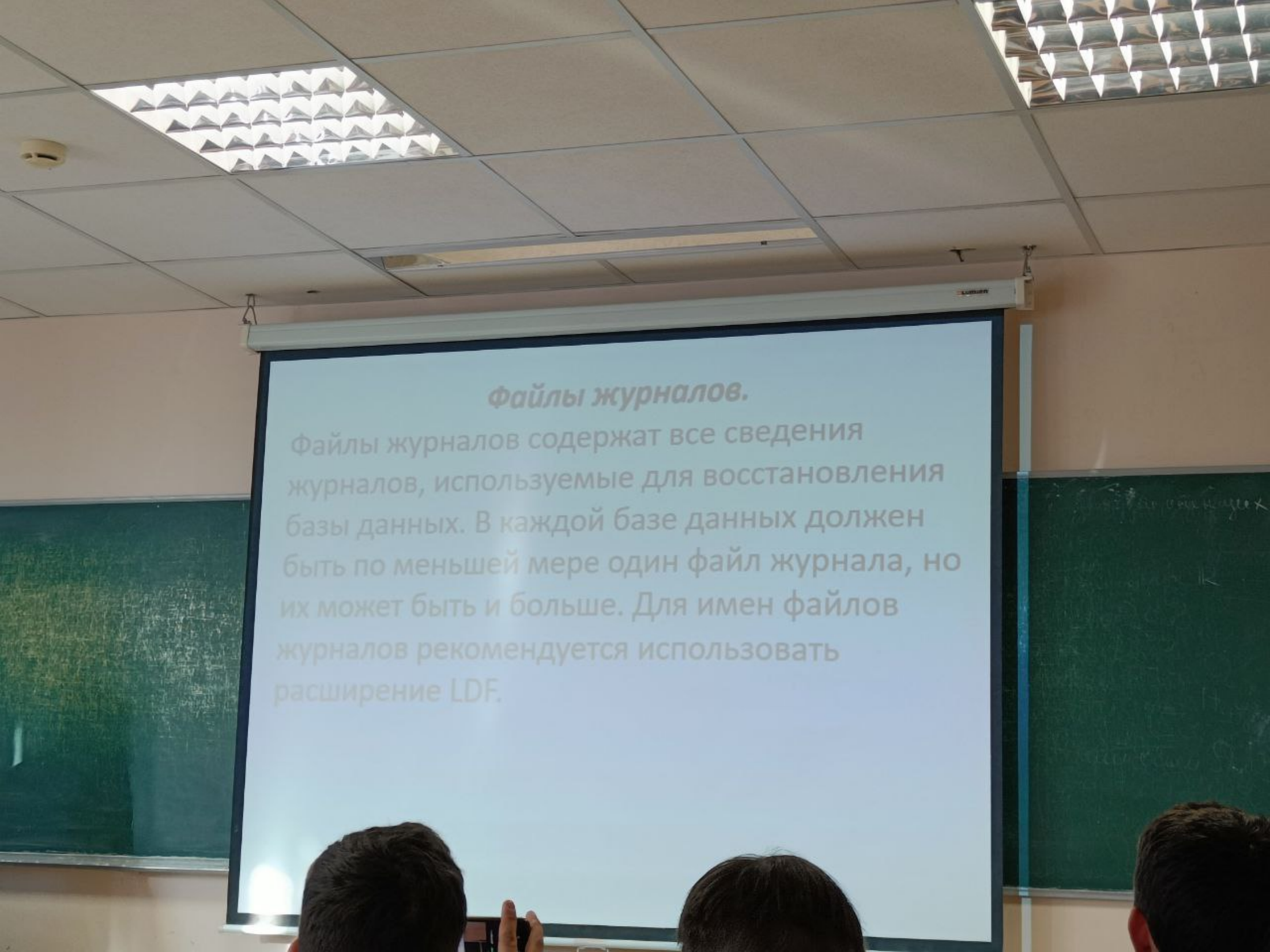

Точка сохранения определяет место, к которому может возвратиться транзакция, если часть транзакции условно отменена. Если транзакция откатывается к точке сохранения, то ее выполнение должно быть продолжено до завершения с обработкой дополнительных инструкций языка Transact-SQL, если необходимо, и инструкции COMMIT TRANSACTION, либо транзакция должна быть полностью отменена откатом к началу.


Распределенные транзакции выполняются
на двух или более серверах, которые
называются диспетчерами ресурсов.
Управление транзакцией должно
координироваться между диспетчерами
ресурсов компонентом сервера, который
называется диспетчером транзакций.

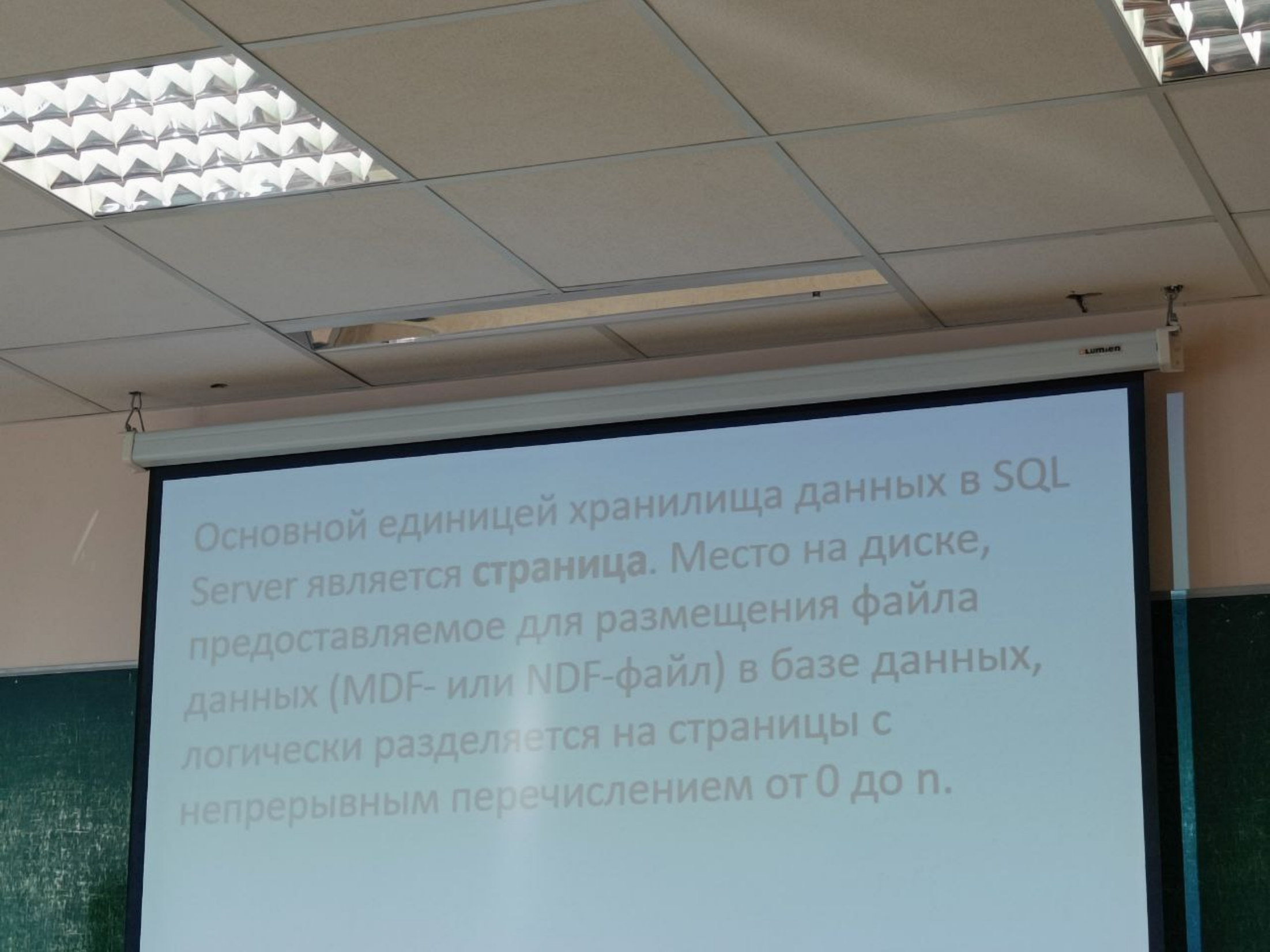

Транзакция в отдельном экземпляре компонента Database Engine, распространяющаяся на несколько данных, в действительности является распределенной транзакцией. Экземпляр управляет распределенной транзакцией на внутреннем уровне, для пользователя она действует как локальная транзакция.

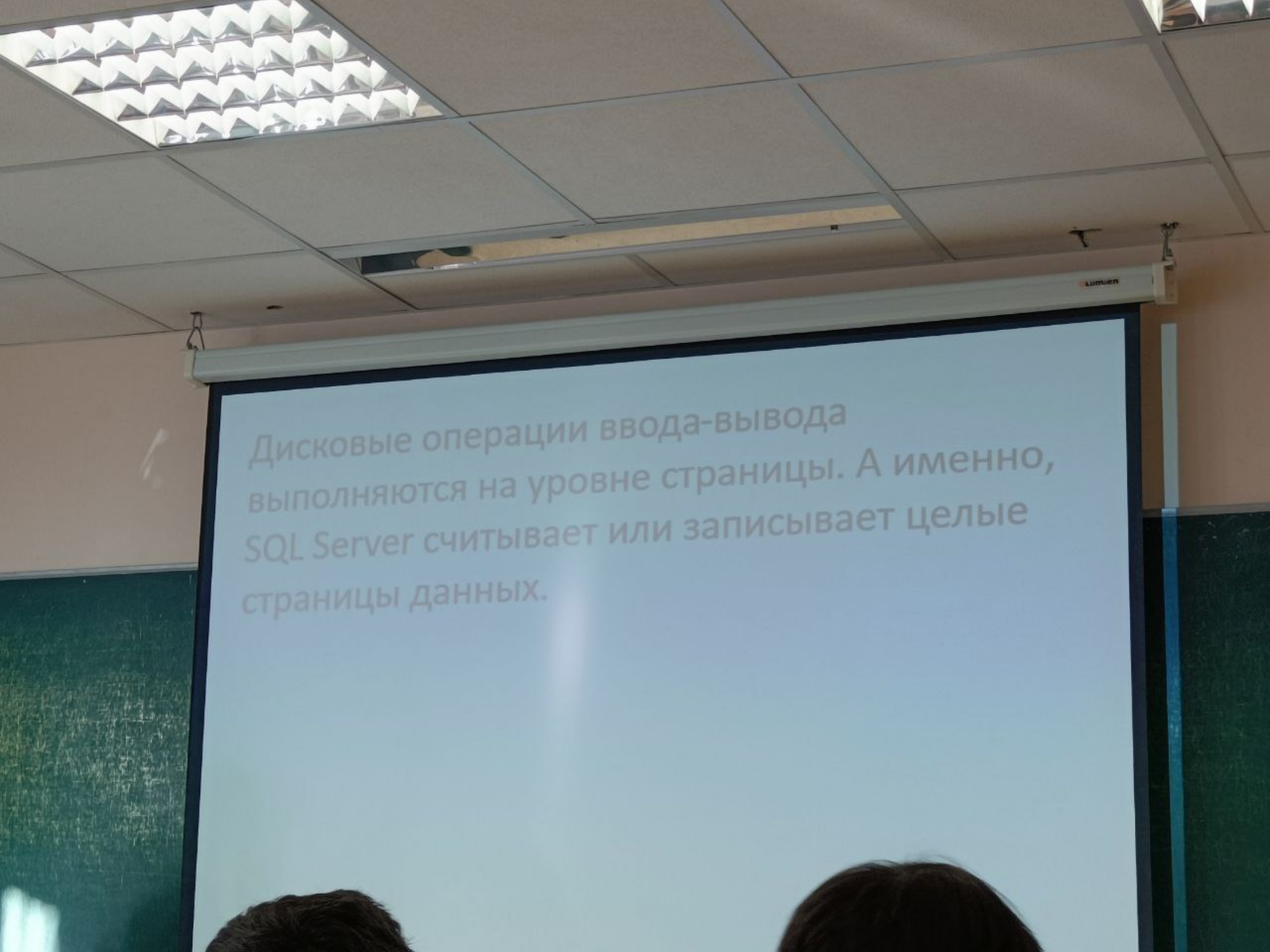

Базы данных SQL Server содержат файлы трех типов.
Первичные файлы данных.
Первичный файл данных является отправной точкой базы данных. Он указывает на остальные файлы базы данных. В каждой базе данных имеется один первичный файл данных. Для имени первичного файла данных рекомендуется использовать расширение MDF.

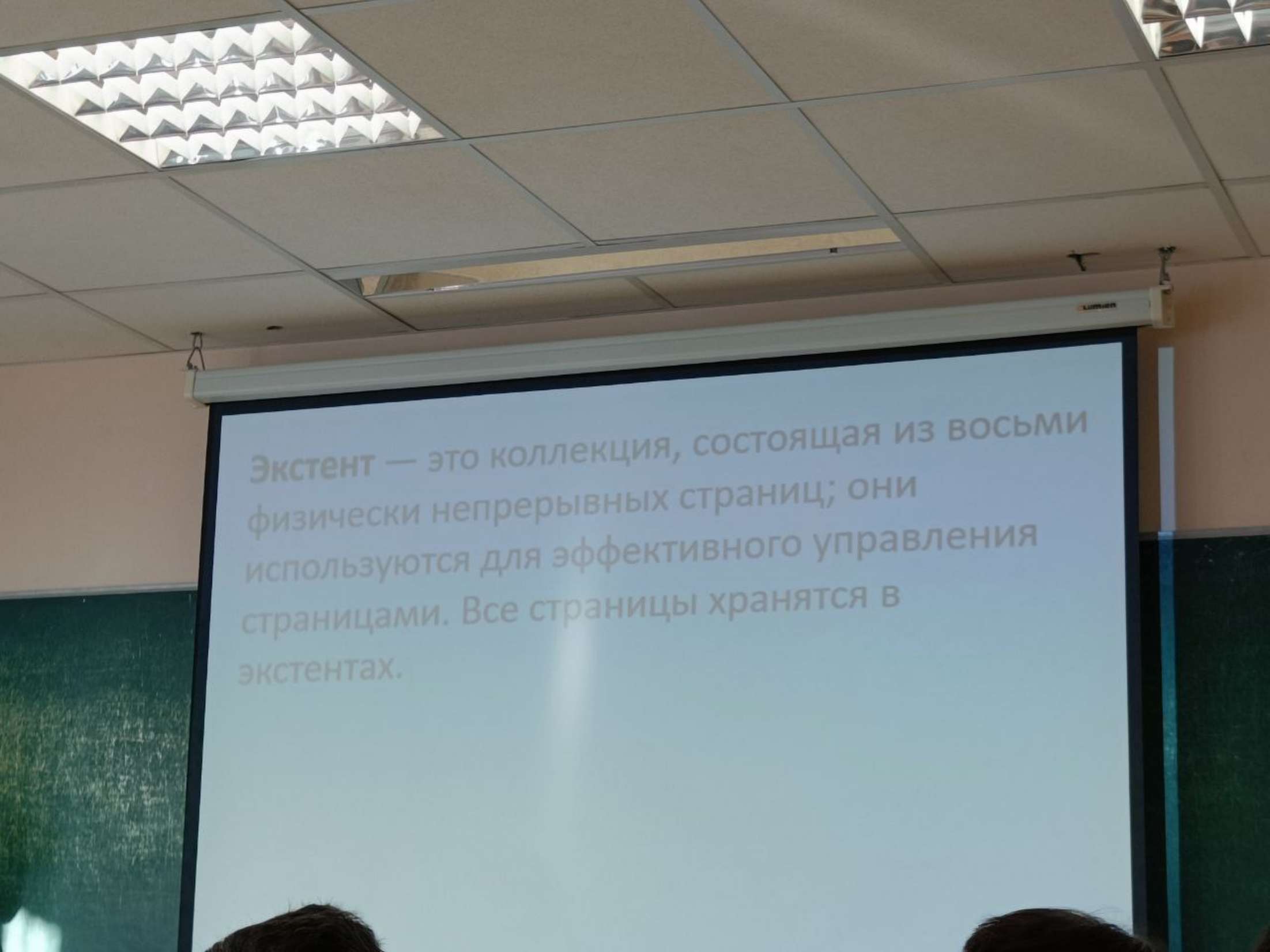

Вторичные файлы данных.
Ко вторичным файлам данных относятся все файлы данных, за исключением первичного файла данных. Некоторые базы данных могут вообще не содержать вторичных файлов данных, тогда как другие содержат несколько вторичных файлов данных. Для имени вторичного файла данных рекомендуется использовать расширение NDF.

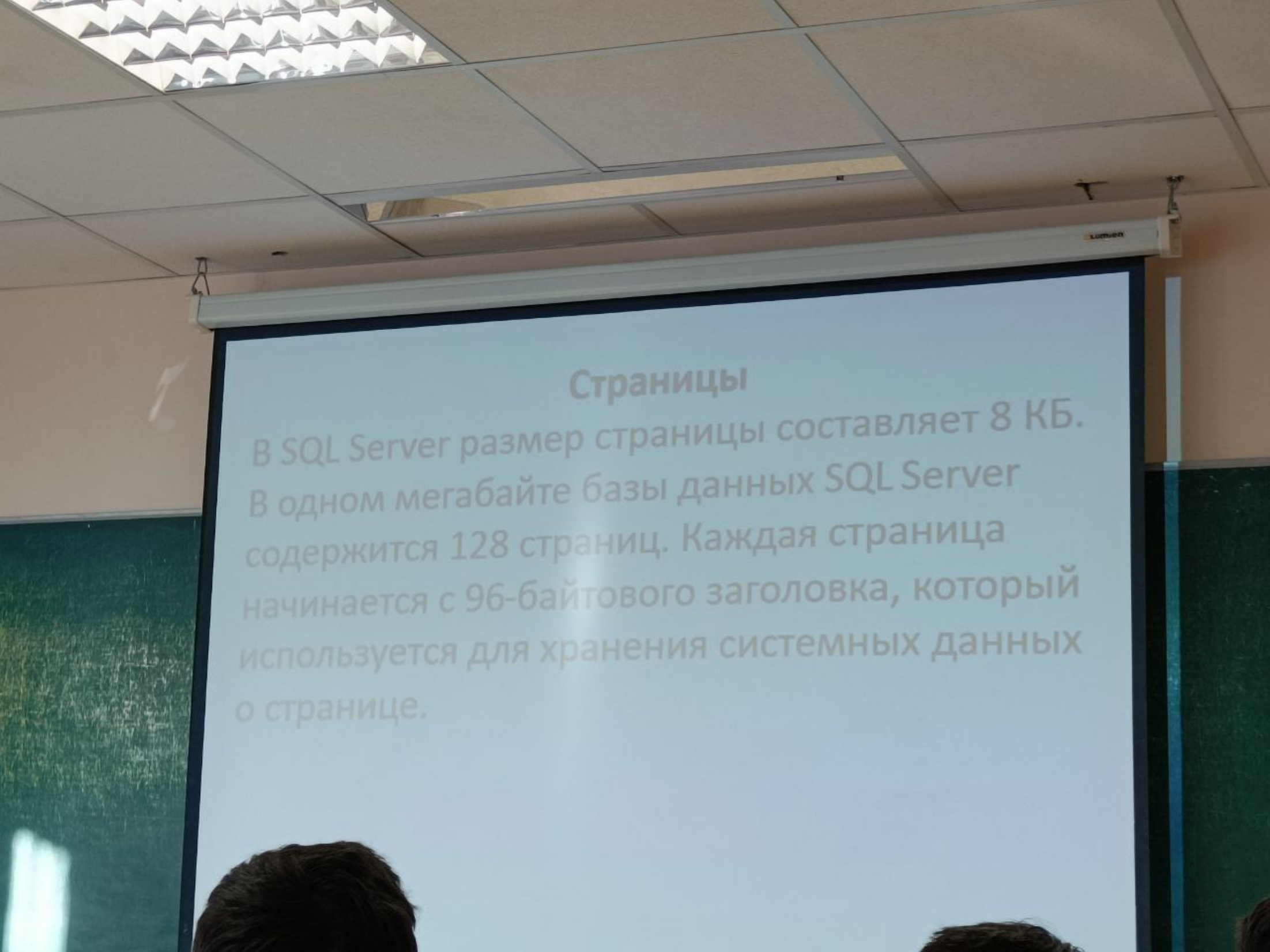

Файлы журналов.
Файлы журналов содержат все сведения журналов, используемые для восстановления базы данных. В каждой базе данных должен быть по меньшей мере один файл журнала, но их может быть и больше. Для имен файлов журналов рекомендуется использовать расширение LDF.

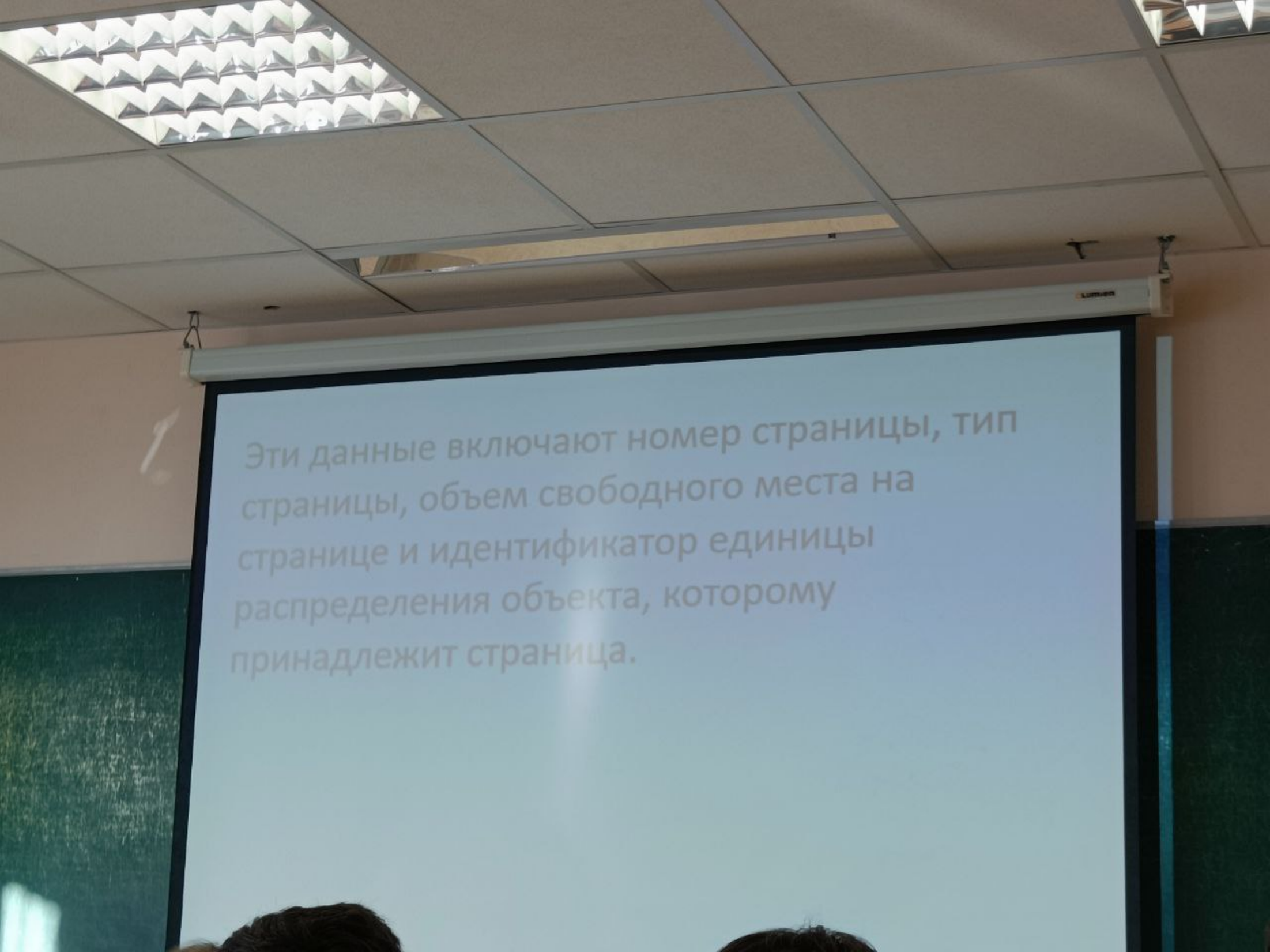
SQL Server не требует обязательного использования расширений файлов MDF, NDF и LDF. Однако эти расширения помогают пользователю идентифицировать различные виды файлов и правильно их использовать.

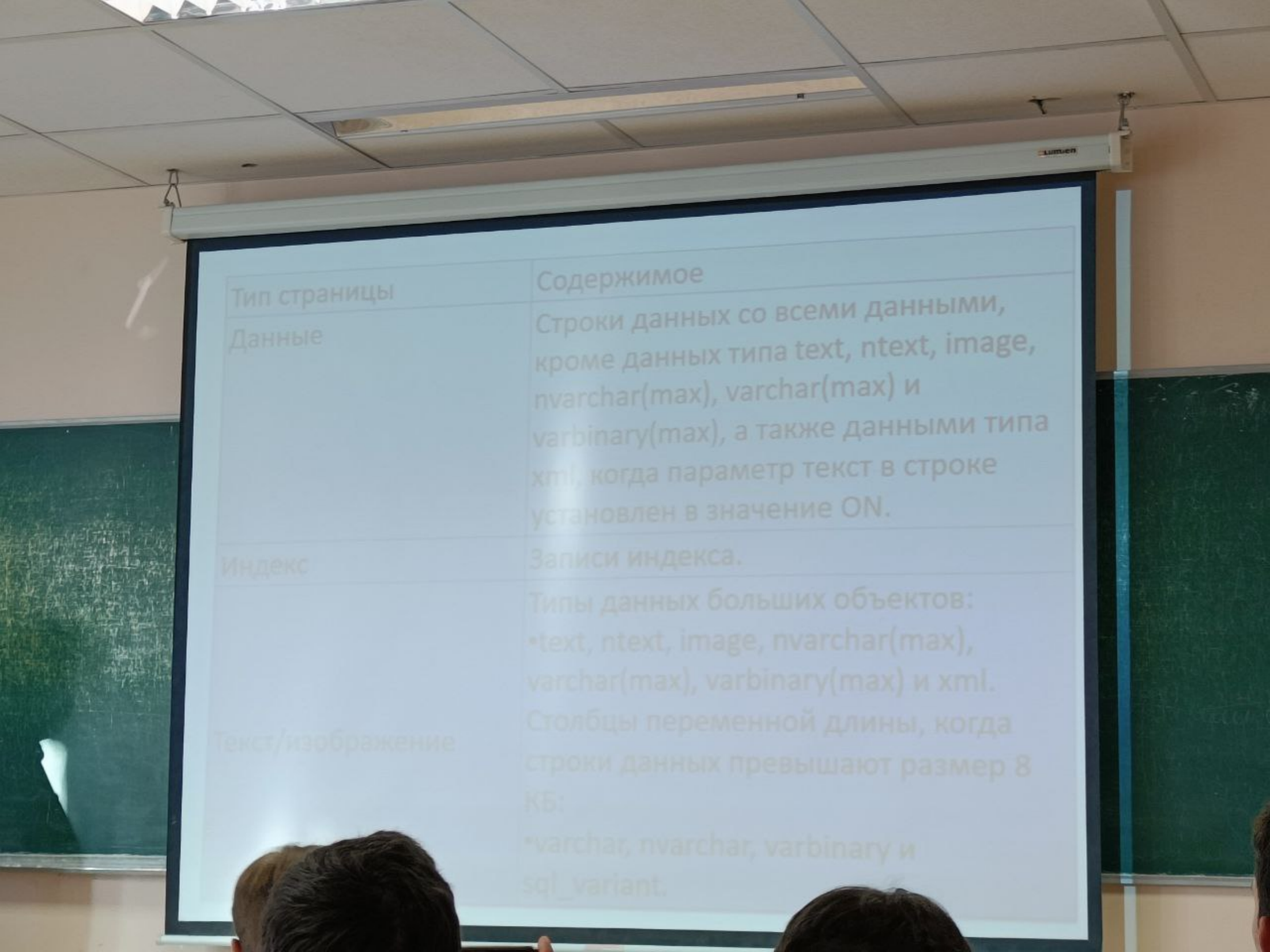

Основной единицей хранилища данных в SQL Server является **страница**. Место на диске, предоставляемое для размещения файла данных (MDF- или NDF-файл) в базе данных, логически разделяется на страницы с непрерывным перечислением от 0 до n.


Дисковые операции ввода-вывода
выполняются на уровне страницы. А именно,
SQL Server считывает или записывает целые
страницы данных.

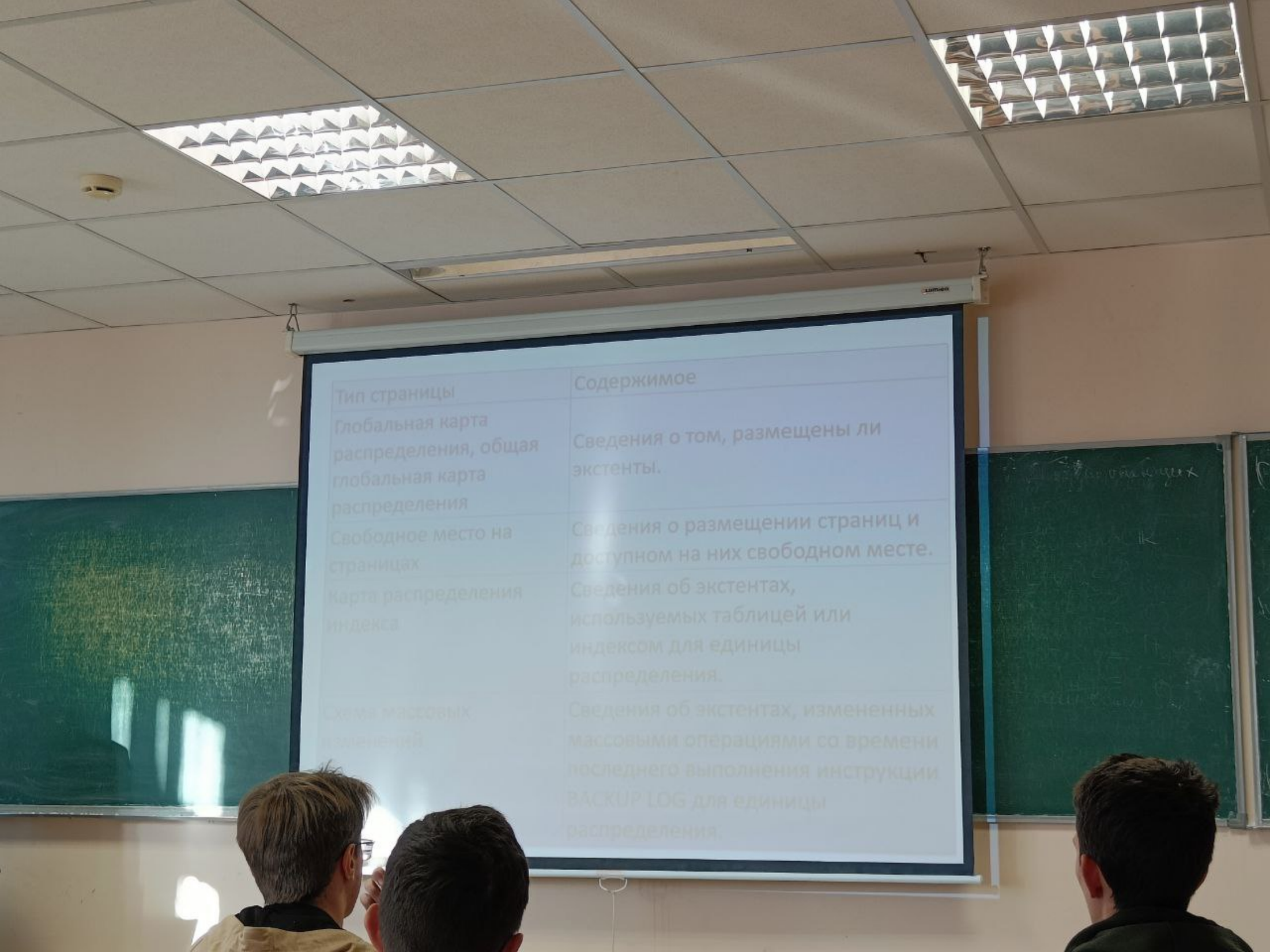

Экстент — это коллекция, состоящая из восьми физически непрерывных страниц; они используются для эффективного управления страницами. Все страницы хранятся в экстентах.

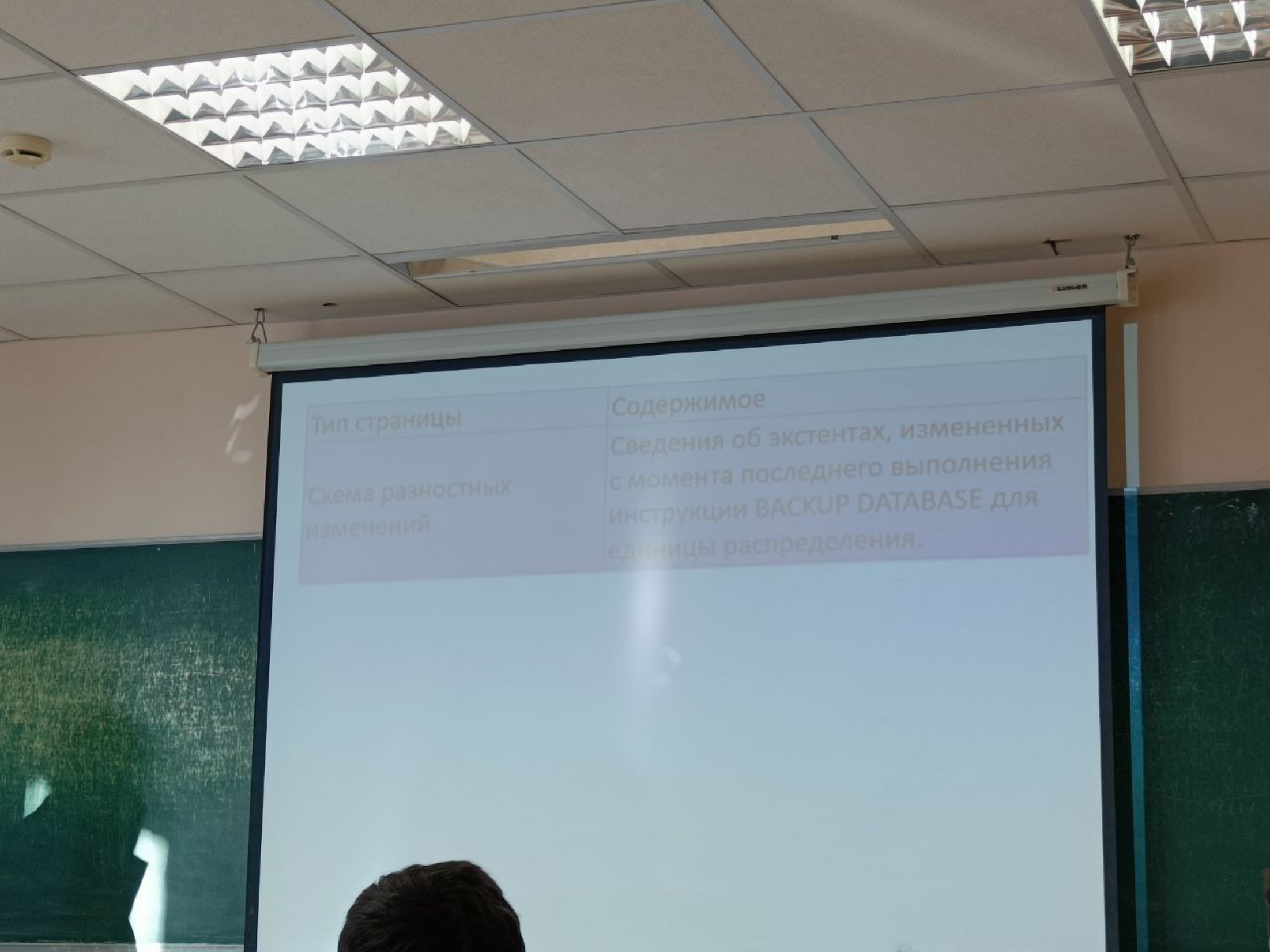

Страницы
В SQL Server размер страницы составляет 8 КБ. В одном мегабайте базы данных SQL Server содержится 128 страниц. Каждая страница начинается с 96-байтового заголовка, который используется для хранения системных данных о странице.

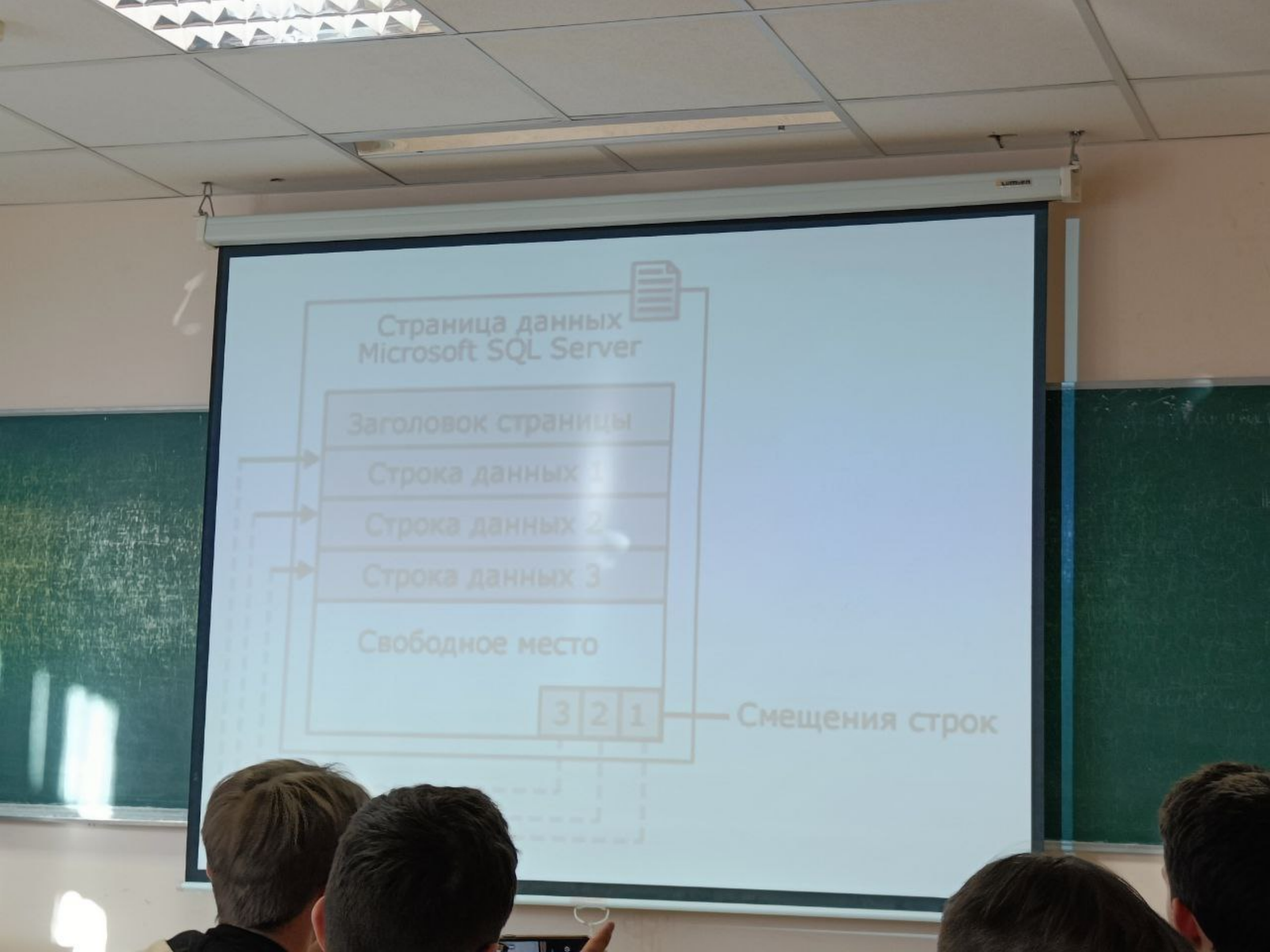

Эти данные включают номер страницы, тип страницы, объем свободного места на странице и идентификатор единицы распределения объекта, которому принадлежит страница.

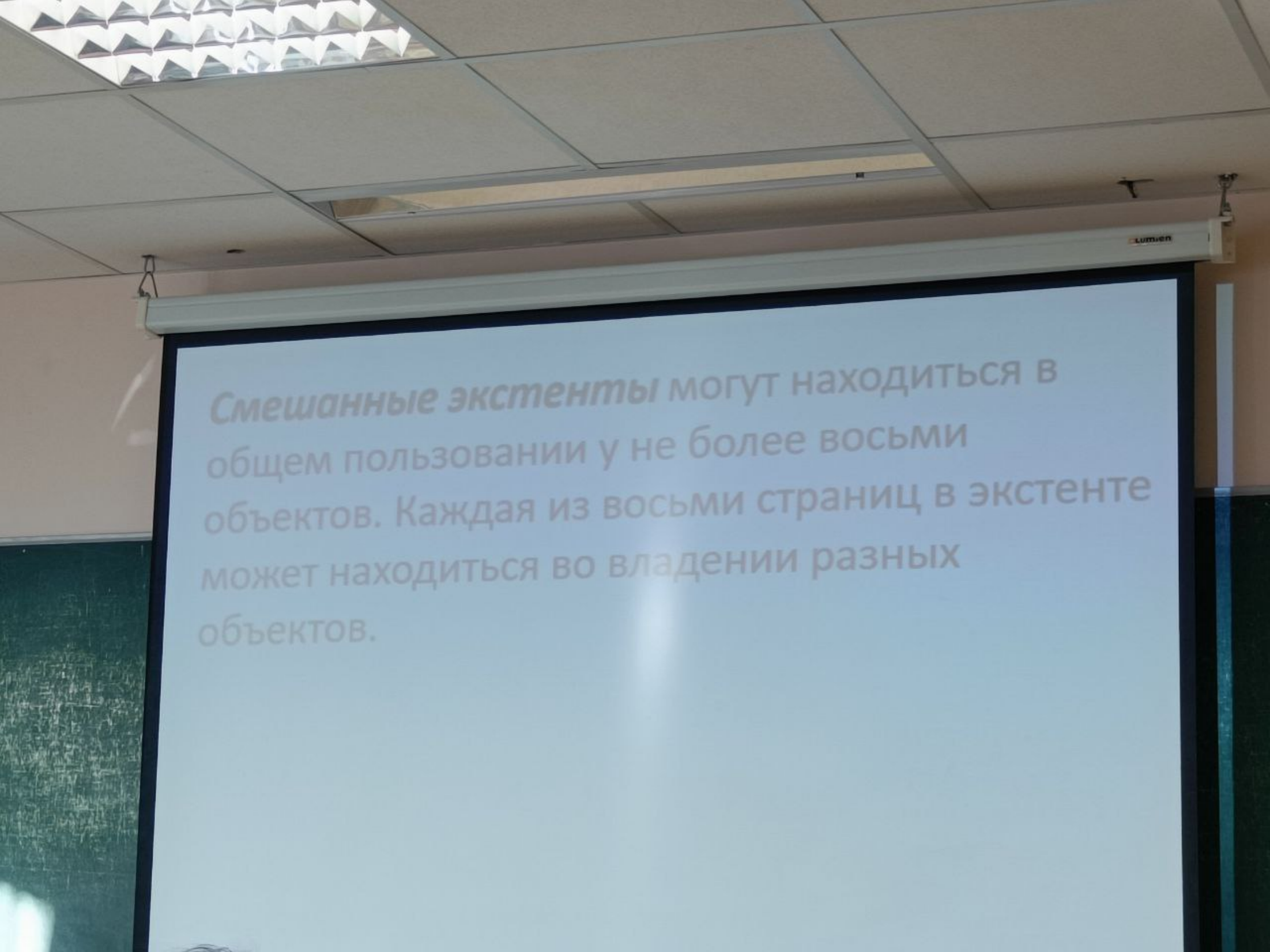


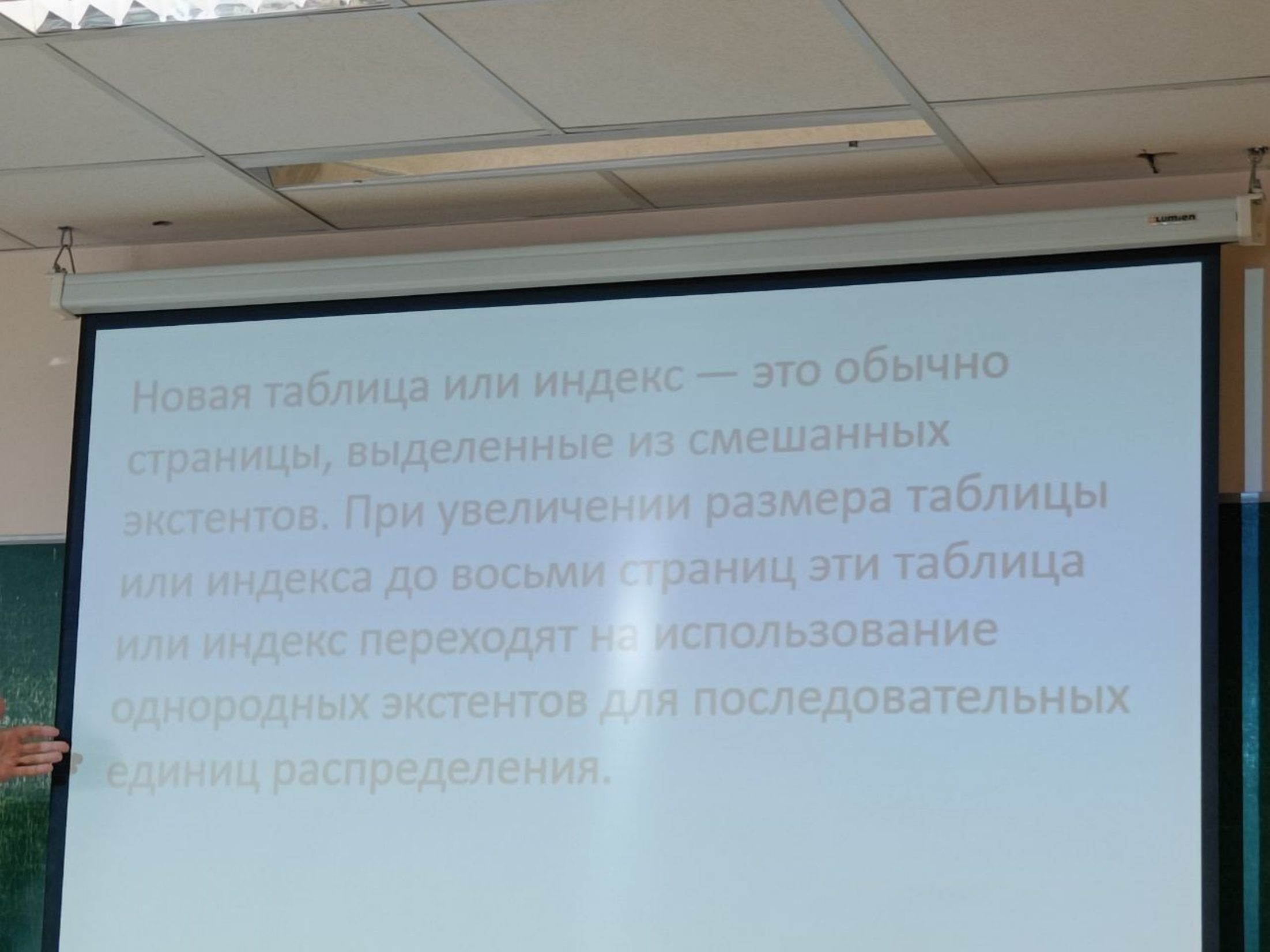
Тип страницы	Содержимое
Данные	Строки данных со всеми данными, кроме данных типа text, ntext, image, nvarchar(max), varchar(max) и varbinary(max), а также данными типа xml, когда параметр текст в строке установлен в значение ON.
Индекс	Записи индекса.
Текст/изображение	Типы данных больших объектов: •text, ntext, image, nvarchar(max), varchar(max), varbinary(max) и xml. Столбцы переменной длины, когда строки данных превышают размер 8 КБ: •varchar, nvarchar, varbinary и sql_variant.

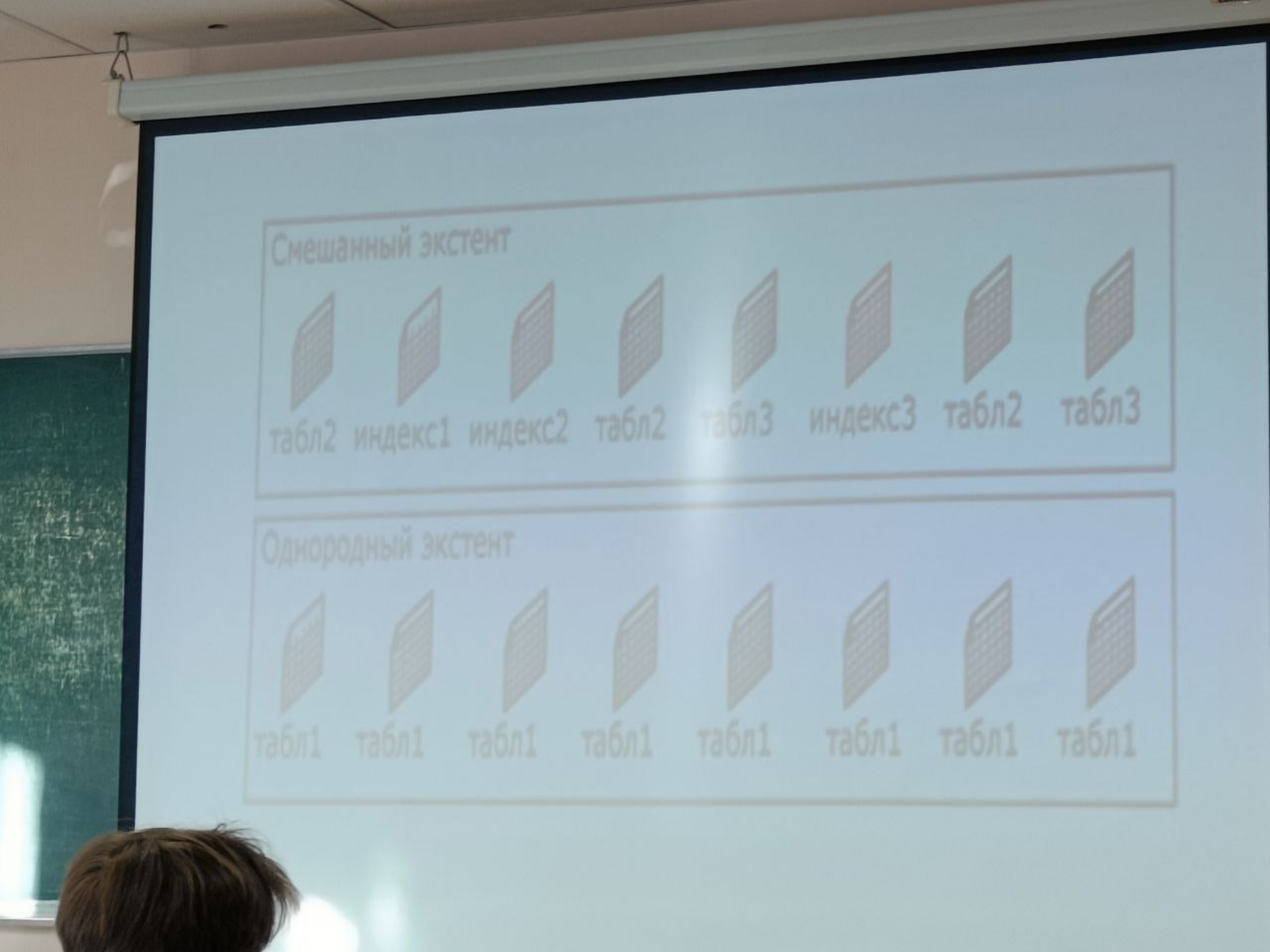

Тип страницы
Содержимое
Глобальная карта распределения, общая глобальная карта распределения
Сведения о том, размещены ли экстенты.
Свободное место на страницах
Сведения о размещении страниц и доступном на них свободном месте.
Карта распределения индекса
Сведения об экстентах, используемых таблицей или индексом для единицы распределения.
Схема массовых изменений
Сведения об экстентах, измененных массовыми операциями со времени последнего выполнения инструкции BACKUP LOG для единицы распределения.


Тип страницы
Содержимое
Схема разностных изменений
Сведения об экстентах, измененных с момента последнего выполнения инструкции BACKUP DATABASE для единицы распределения.


Страница данных
Microsoft SQL Server
Заголовок страницы
Строка данных 1
Строка данных 2
Строка данных 3
Свободное место
3
2
1
Смещения строк


Lumien
Смешанные экстенты могут находиться в общем пользовании у не более восьми объектов. Каждая из восьми страниц в экстенте может находиться во владении разных объектов.


Новая таблица или индекс — это обычно страницы, выделенные из смешанных экстентов. При увеличении размера таблицы или индекса до восьми страниц эти таблица или индекс переходят на использование однородных экстентов для последовательных единиц распределения.


Смешанный экстент
табл2
индекс1
индекс2
табл2
табл3
индекс3
табл2
табл3
Однородный экстент
табл1
табл1
табл1
табл1
табл1
табл1
табл1
табл1

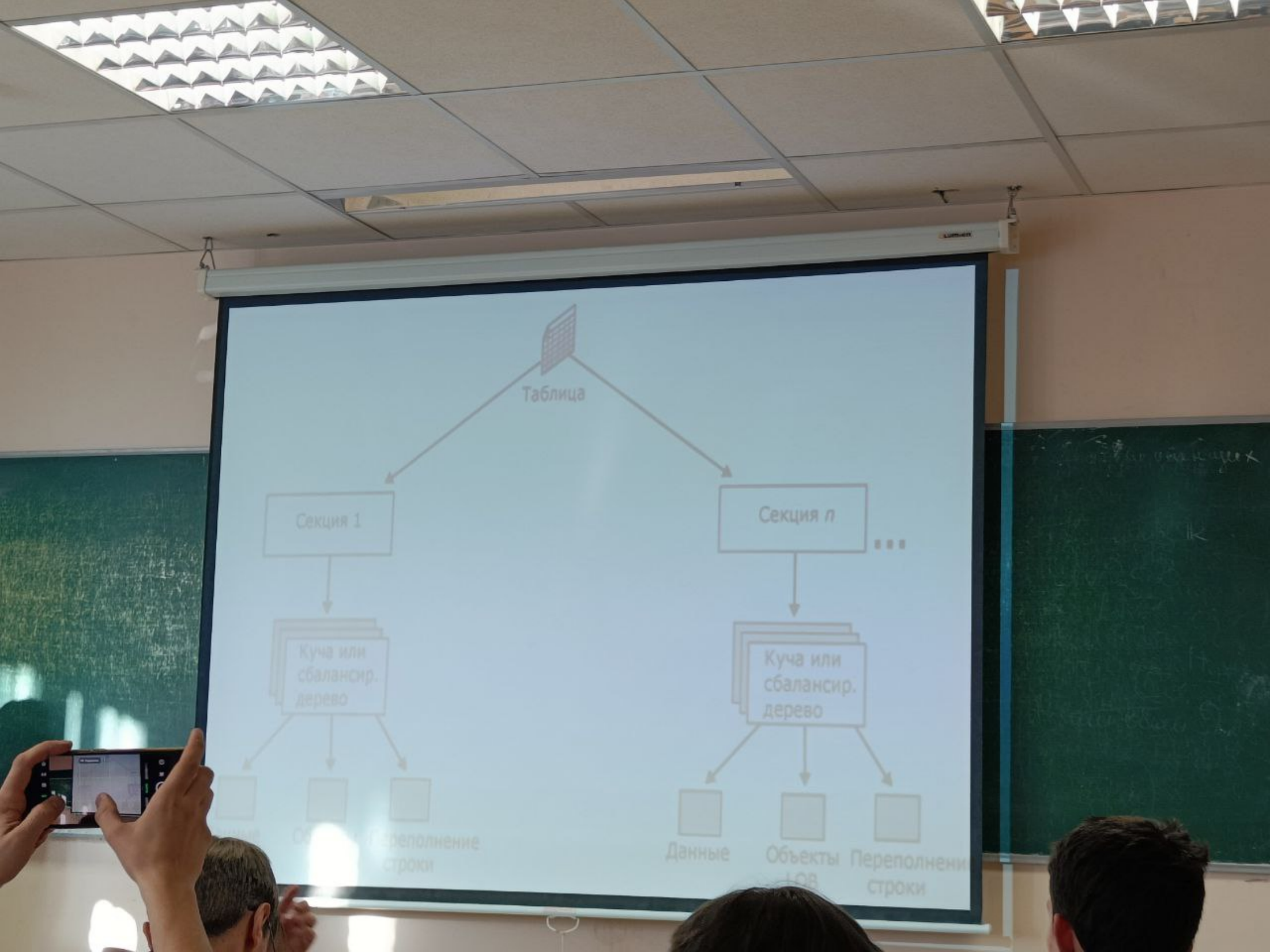
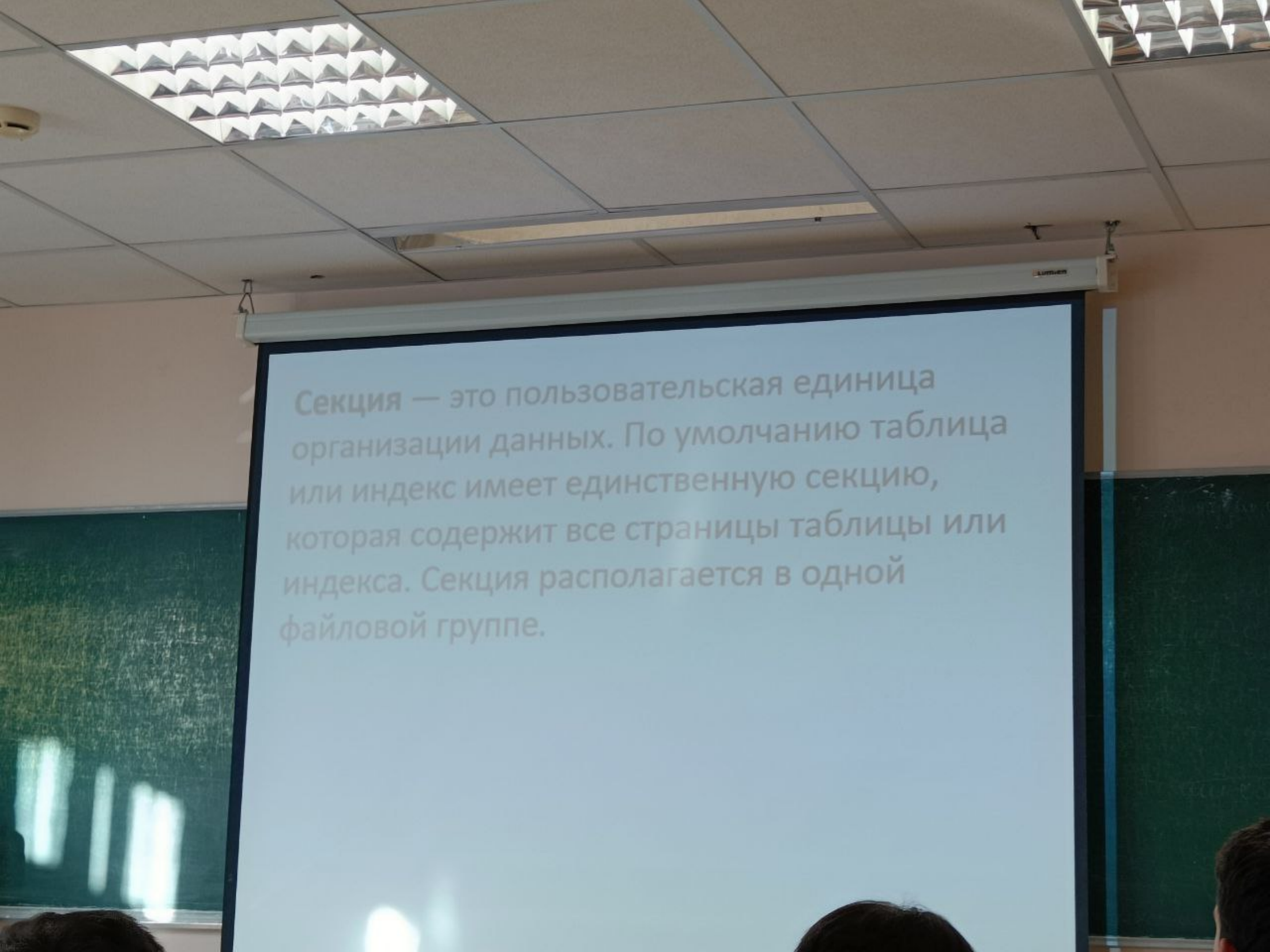
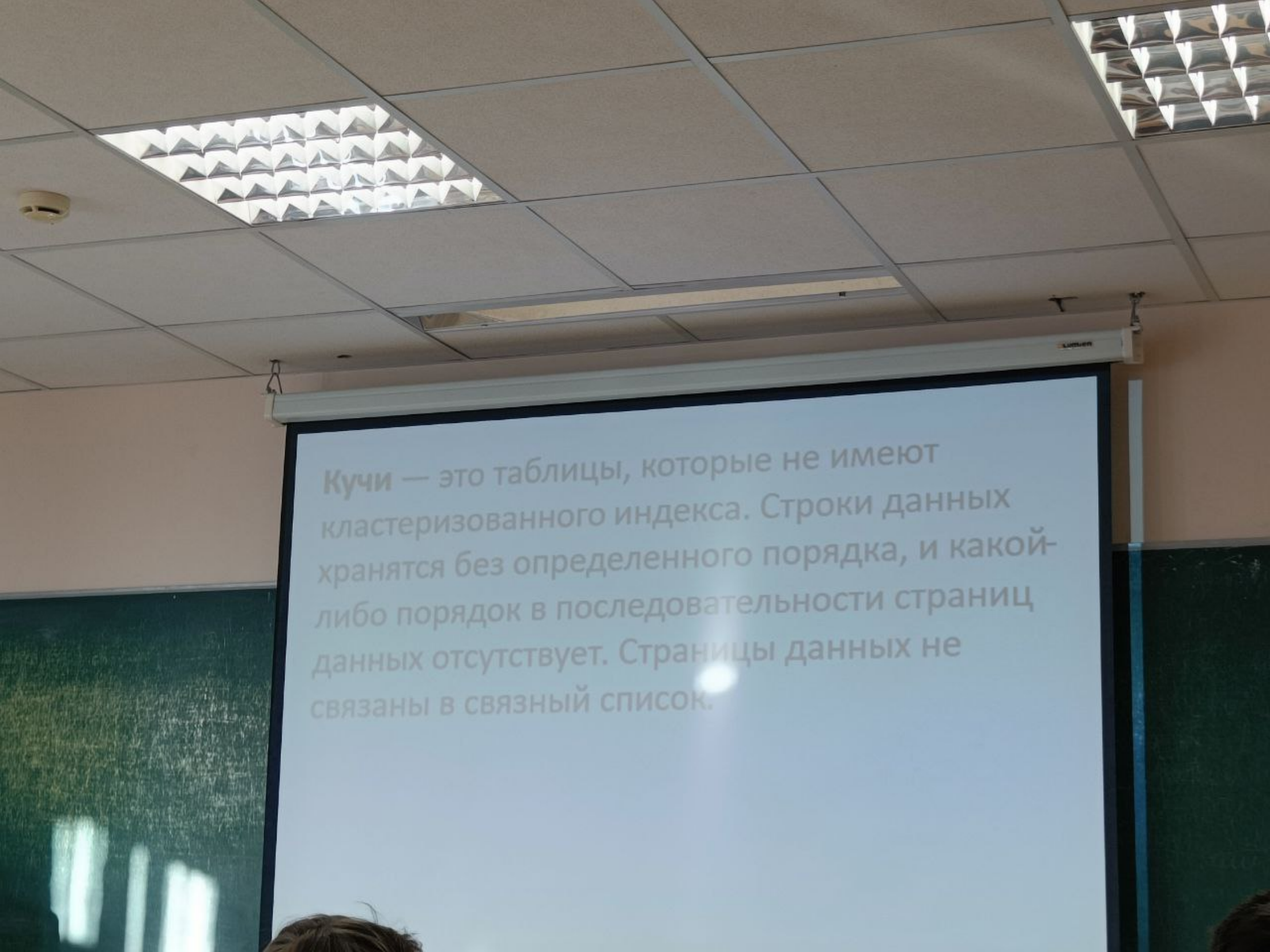
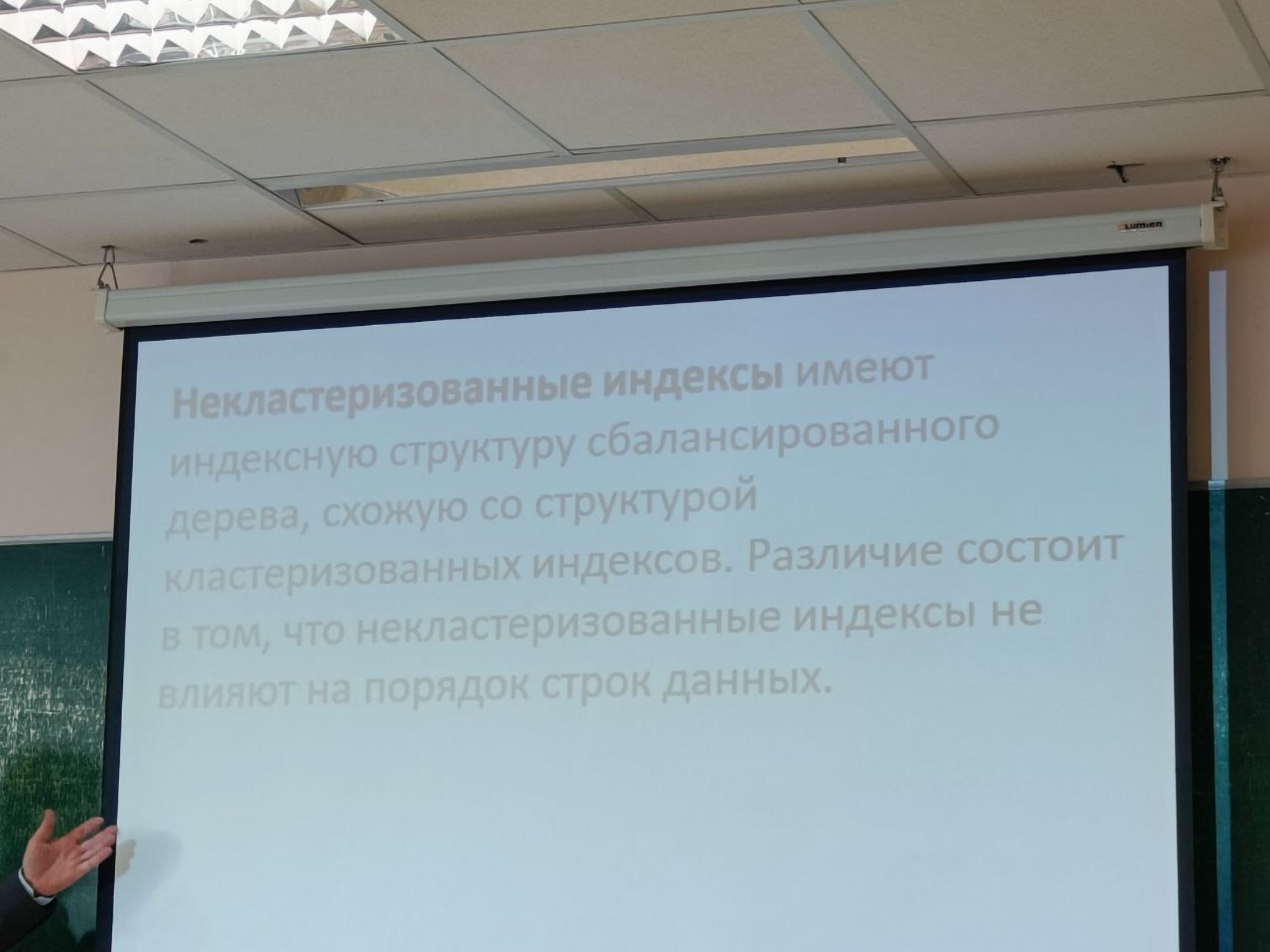
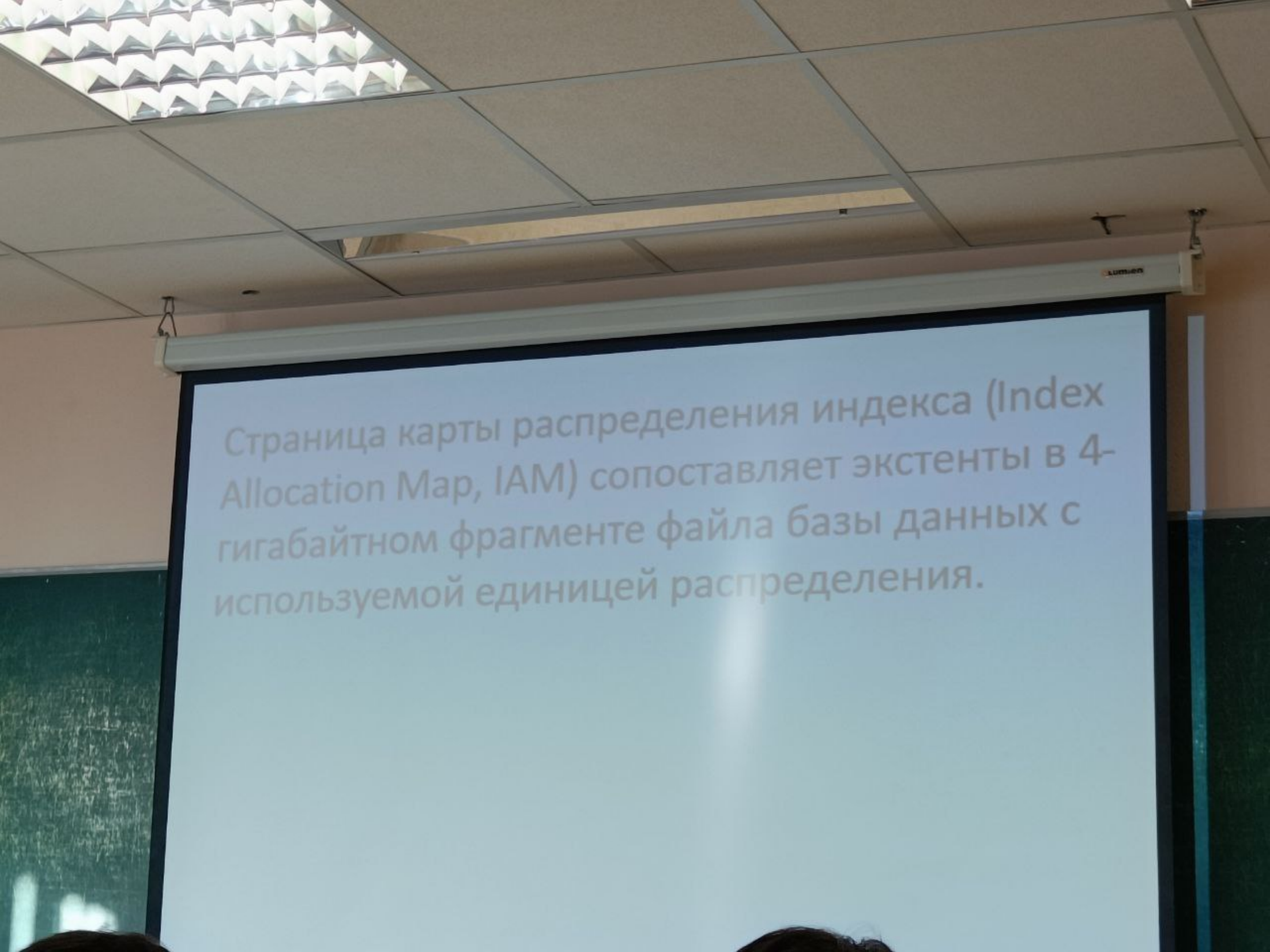


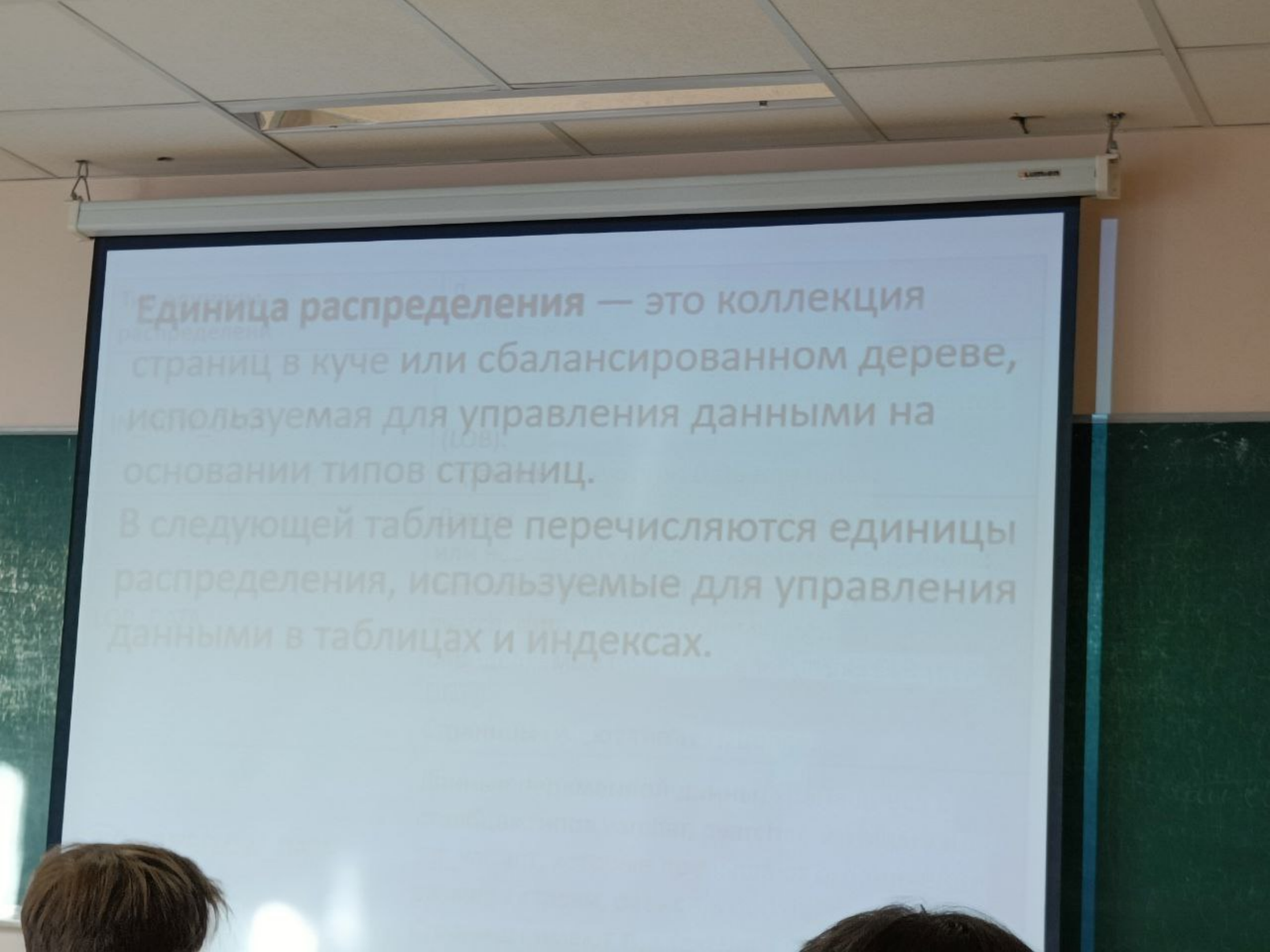
Таблица
Секция 1
Секция n
Куча или сбалансир. дерево
Куча или сбалансир. дерево
Данные
Объекты
Переполнени строки

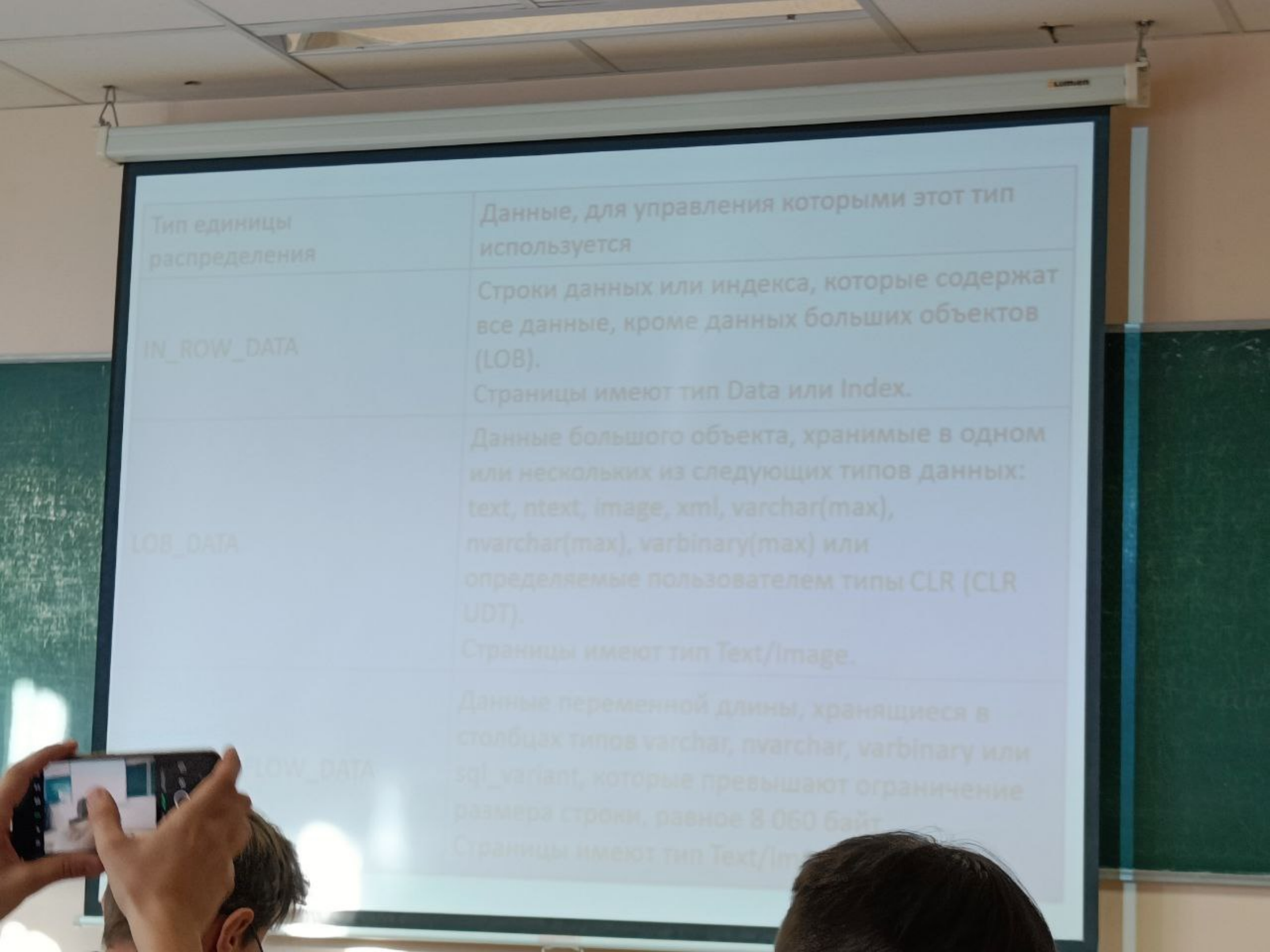

Секция — это пользовательская единица организации данных. По умолчанию таблица или индекс имеет единственную секцию, которая содержит все страницы таблицы или индекса. Секция располагается в одной файловой группе.


Кучи — это таблицы, которые не имеют кластеризованного индекса. Строки данных хранятся без определенного порядка, и какой-либо порядок в последовательности страниц данных отсутствует. Страницы данных не связаны в связный список.

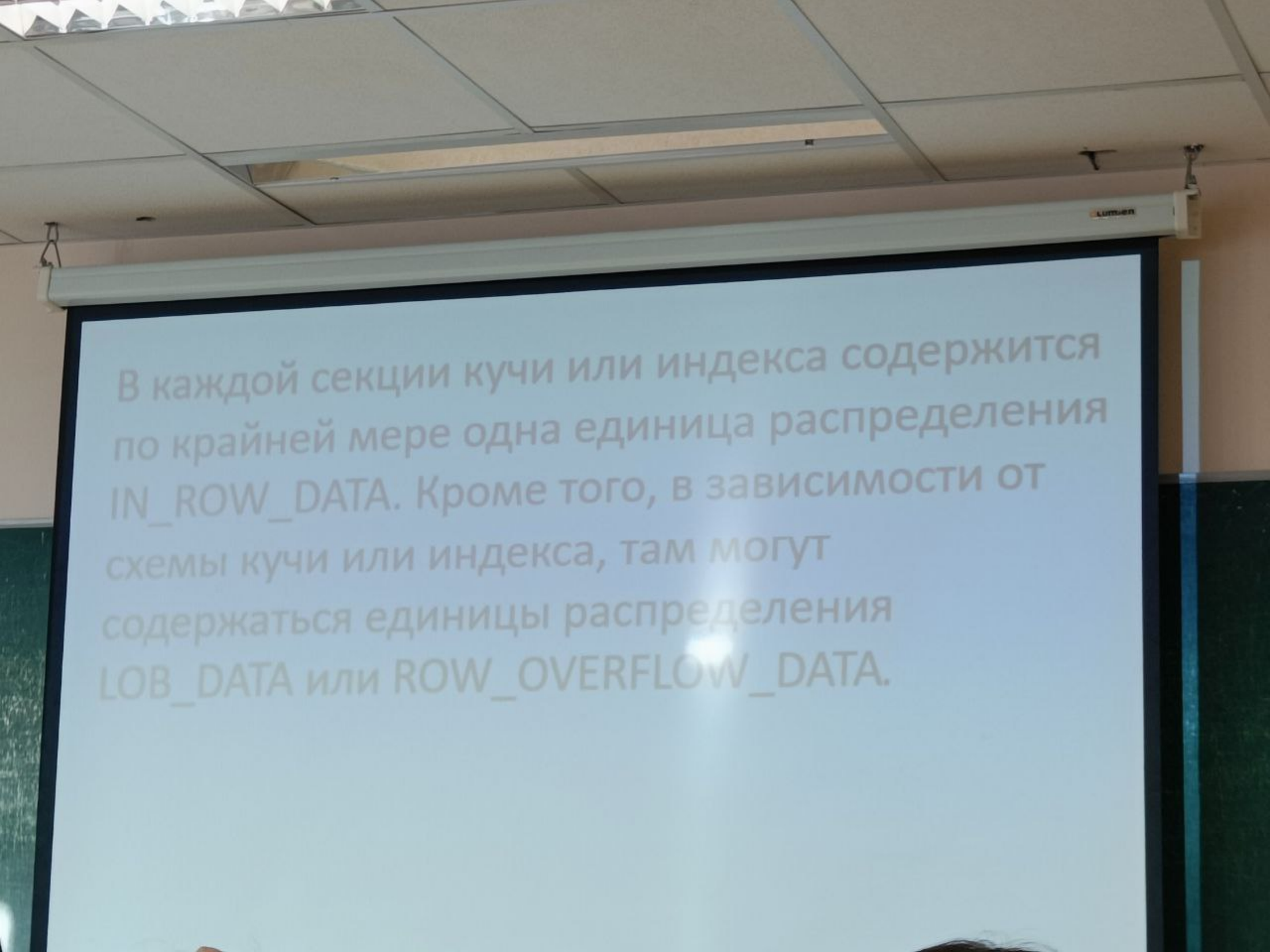

Некластеризованные индексы имеют индексную структуру сбалансированного дерева, схожую со структурой кластеризованных индексов. Различие состоит в том, что некластеризованные индексы не влияют на порядок строк данных.


Страница карты распределения индекса (Index Allocation Map, IAM) сопоставляет экстенты в 4-гигабайтном фрагменте файла базы данных с используемой единицей распределения.

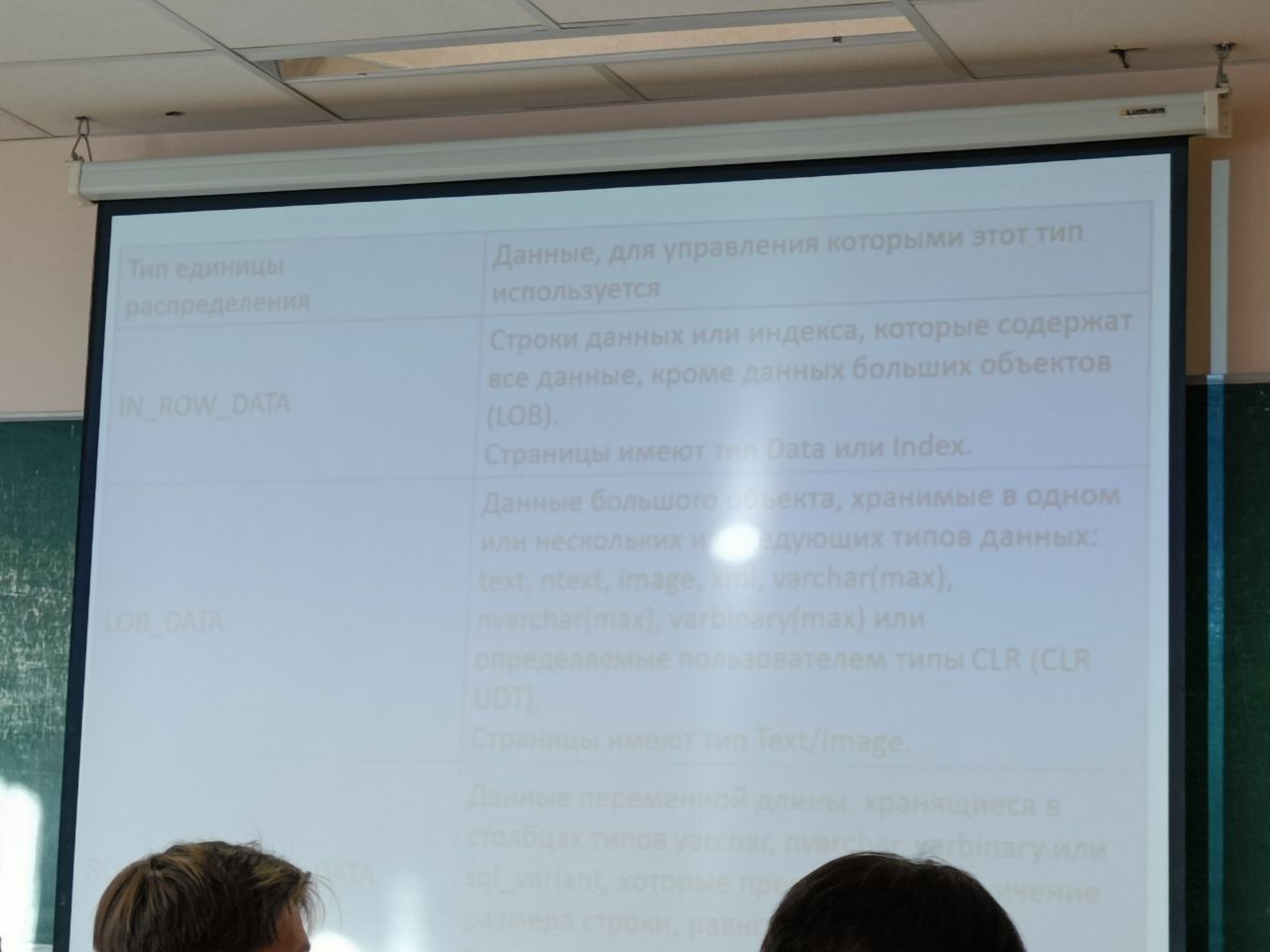

Единица распределения — это коллекция страниц в куче или сбалансированном дереве, используемая для управления данными на основании типов страниц.
В следующей таблице перечисляются единицы распределения, используемые для управления данными в таблицах и индексах.

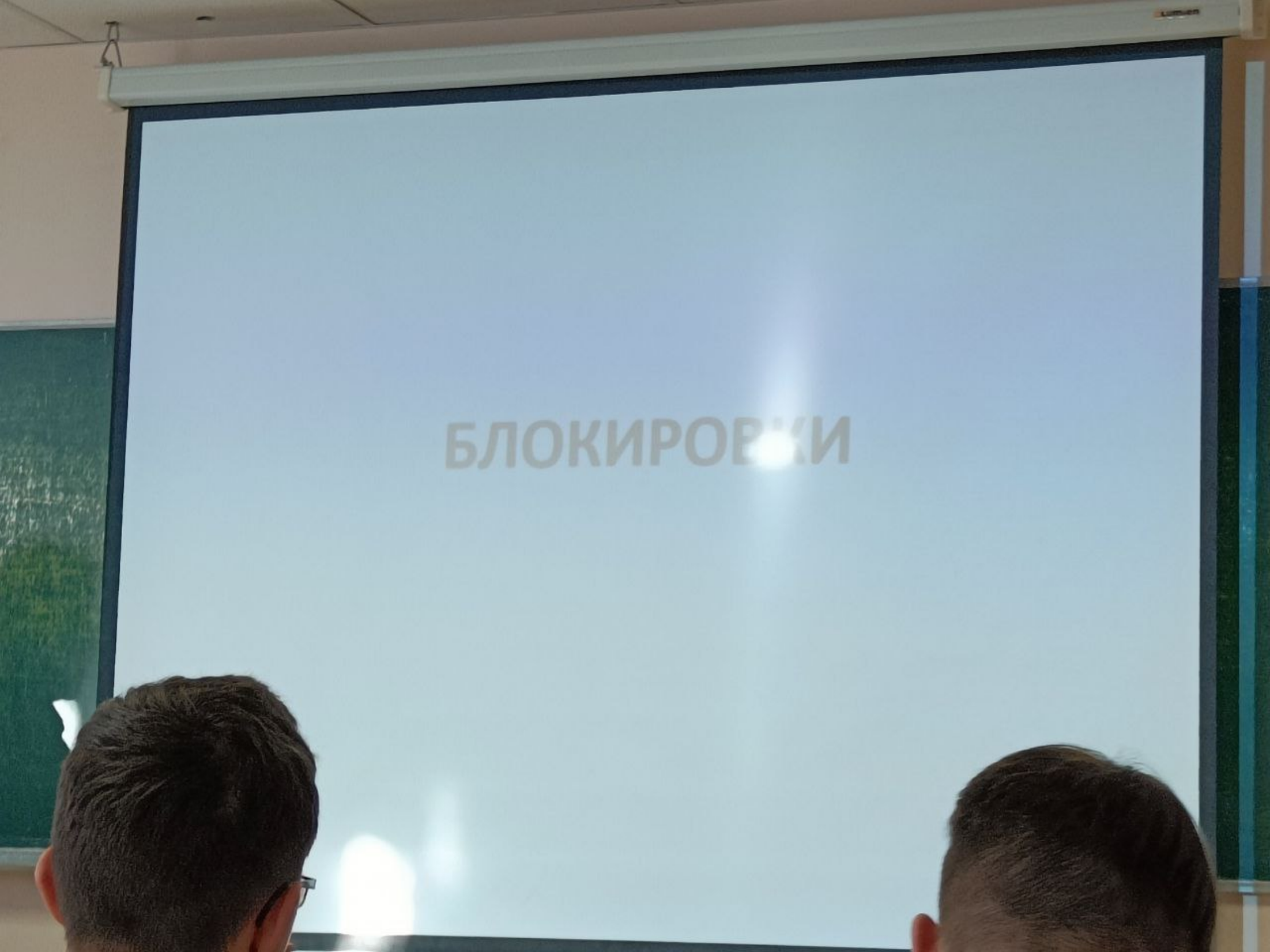


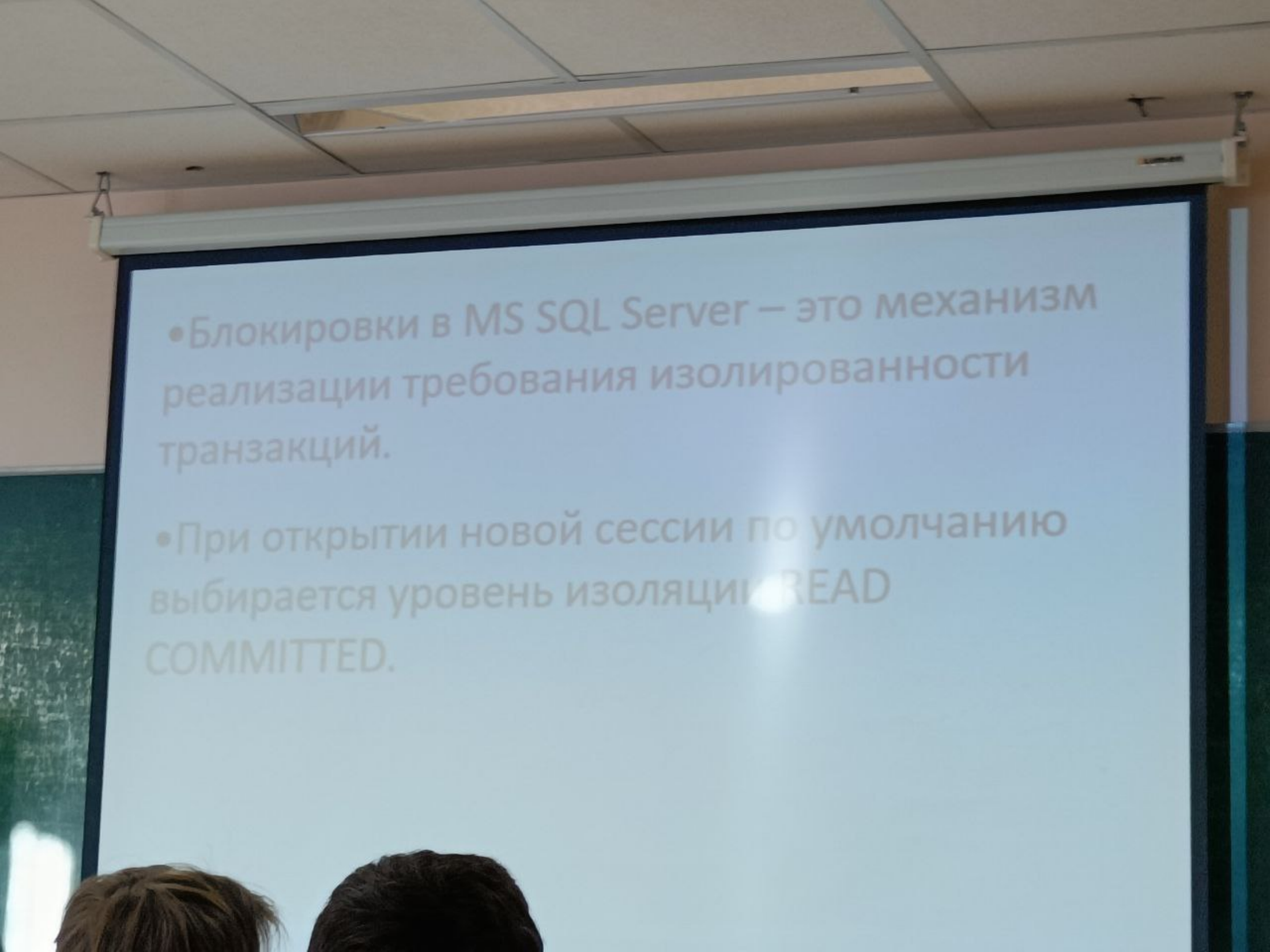
Тип единицы распределения	Данные, для управления которыми этот тип используется
IN_ROW_DATA	Строки данных или индекса, которые содержат все данные, кроме данных больших объектов (LOB). Страницы имеют тип Data или Index.
LOB_DATA	Данные большого объекта, хранимые в одном или нескольких из следующих типов данных: text, ntext, image, xml, varchar(max), nvarchar(max), varbinary(max) или определяемые пользователем типы CLR (CLR UDT). Страницы имеют тип Text/Image.
ROW_DATA	Данные переменной длины, хранящиеся в столбцах типов varchar, nvarchar, varbinary или sql_variant, которые превышают ограничение размера строки, равное 8 060 байт. Страницы имеют тип Text/Im

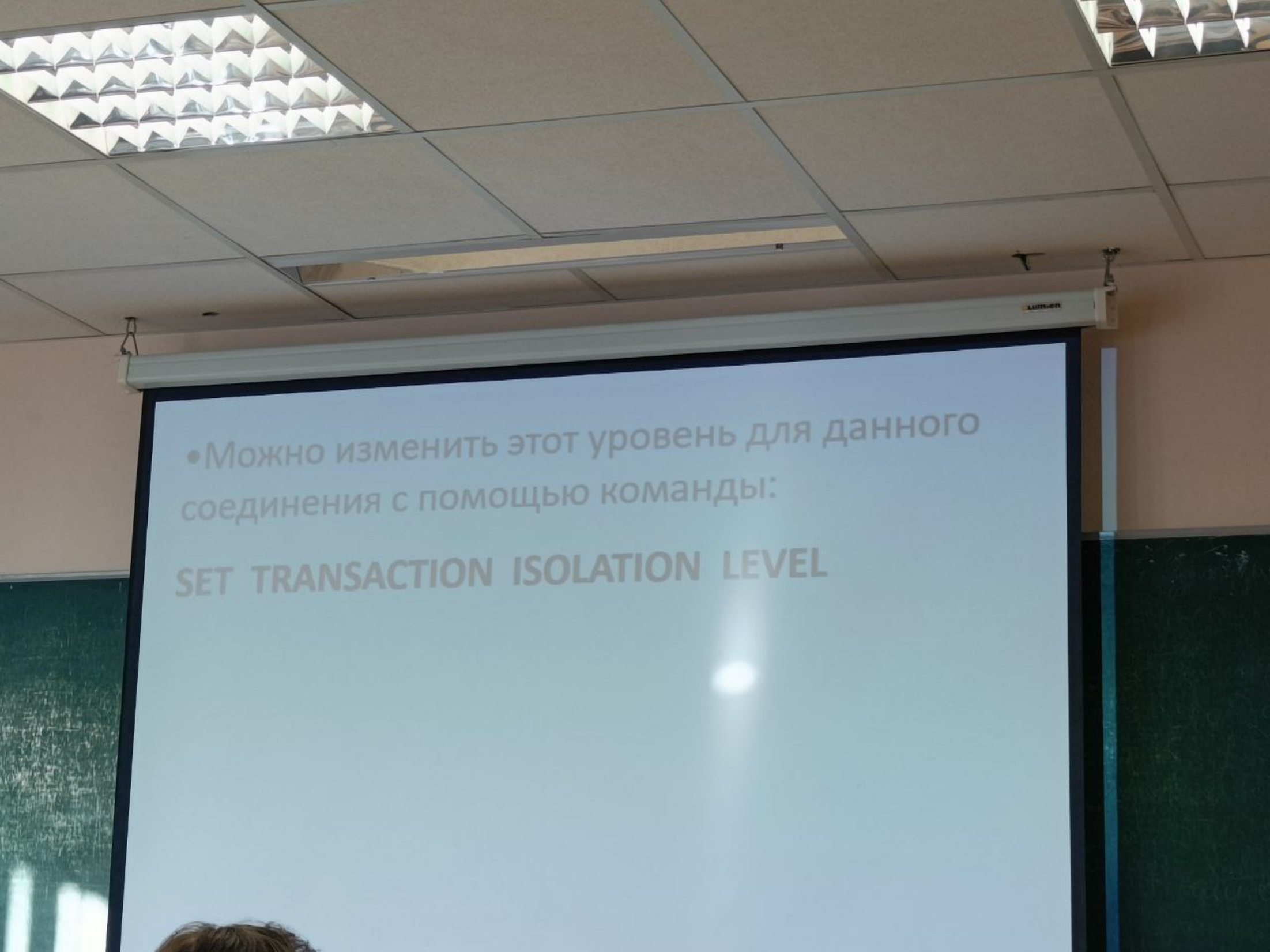

В каждой секции кучи или индекса содержится по крайней мере одна единица распределения IN_ROW_DATA. Кроме того, в зависимости от схемы кучи или индекса, там могут содержаться единицы распределения LOB_DATA или ROW_OVERFLOW_DATA.

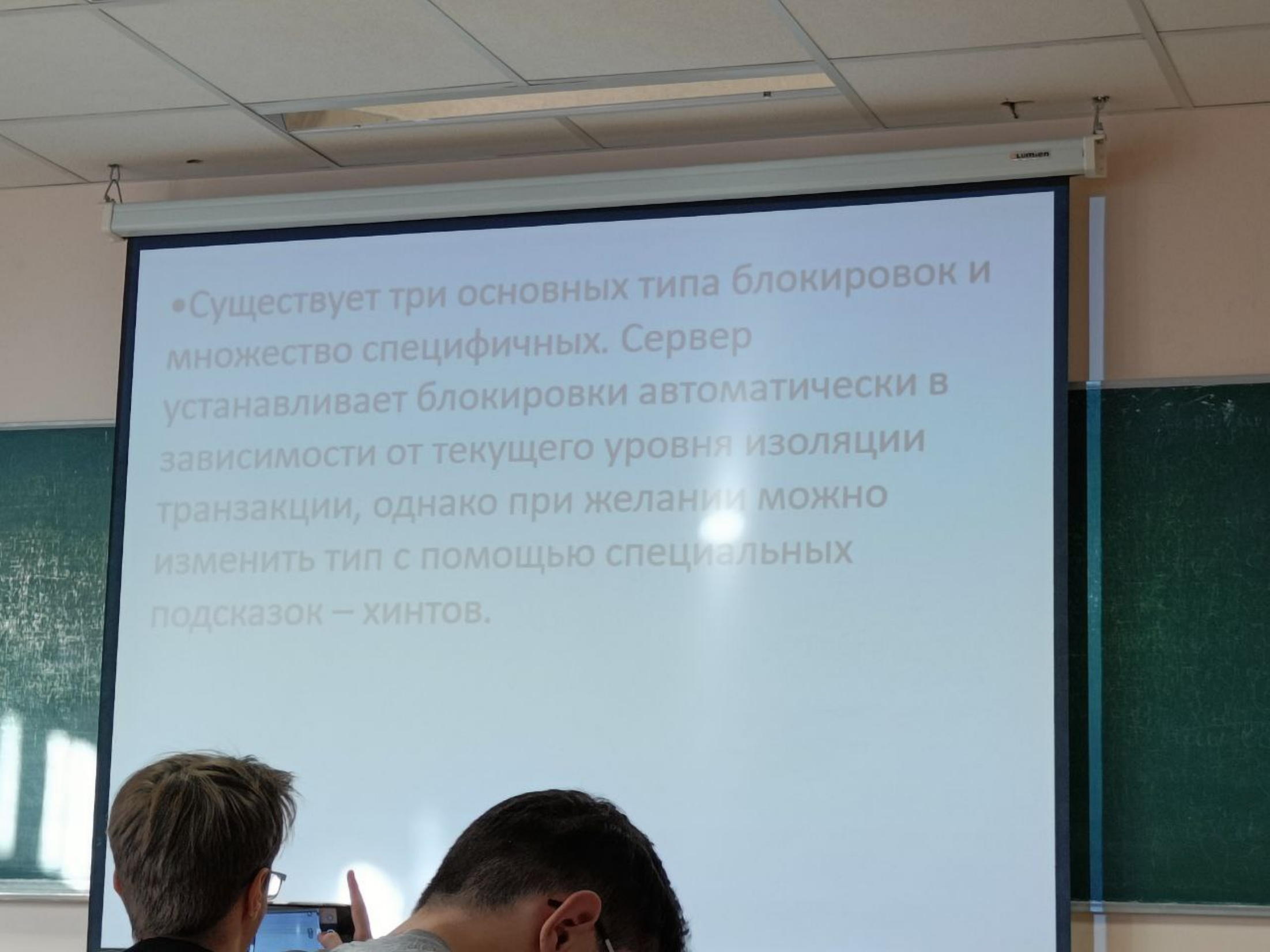
Тип единицы распределения	Данные, для управления которыми этот тип используется
IN_ROW_DATA	Строки данных или индекса, которые содержат все данные, кроме данных больших объектов (LOB). Страницы имеют тип Data или Index.
LOB_DATA	Данные большого объекта, хранимые в одном или нескольких из следующих типов данных: text, ntext, image, xml, varchar(max), nvarchar(max), varbinary(max) или определяемые пользователем типы CLR (CLR UDT). Страницы имеют тип Text/Image.
ROW_OVERFLOW_DATA	Данные переменной длины, хранящиеся в столбцах типов varchar, nvarchar, varbinary или sql_variant, которые превышают ограничение размера строки, равное 8060 байтам.

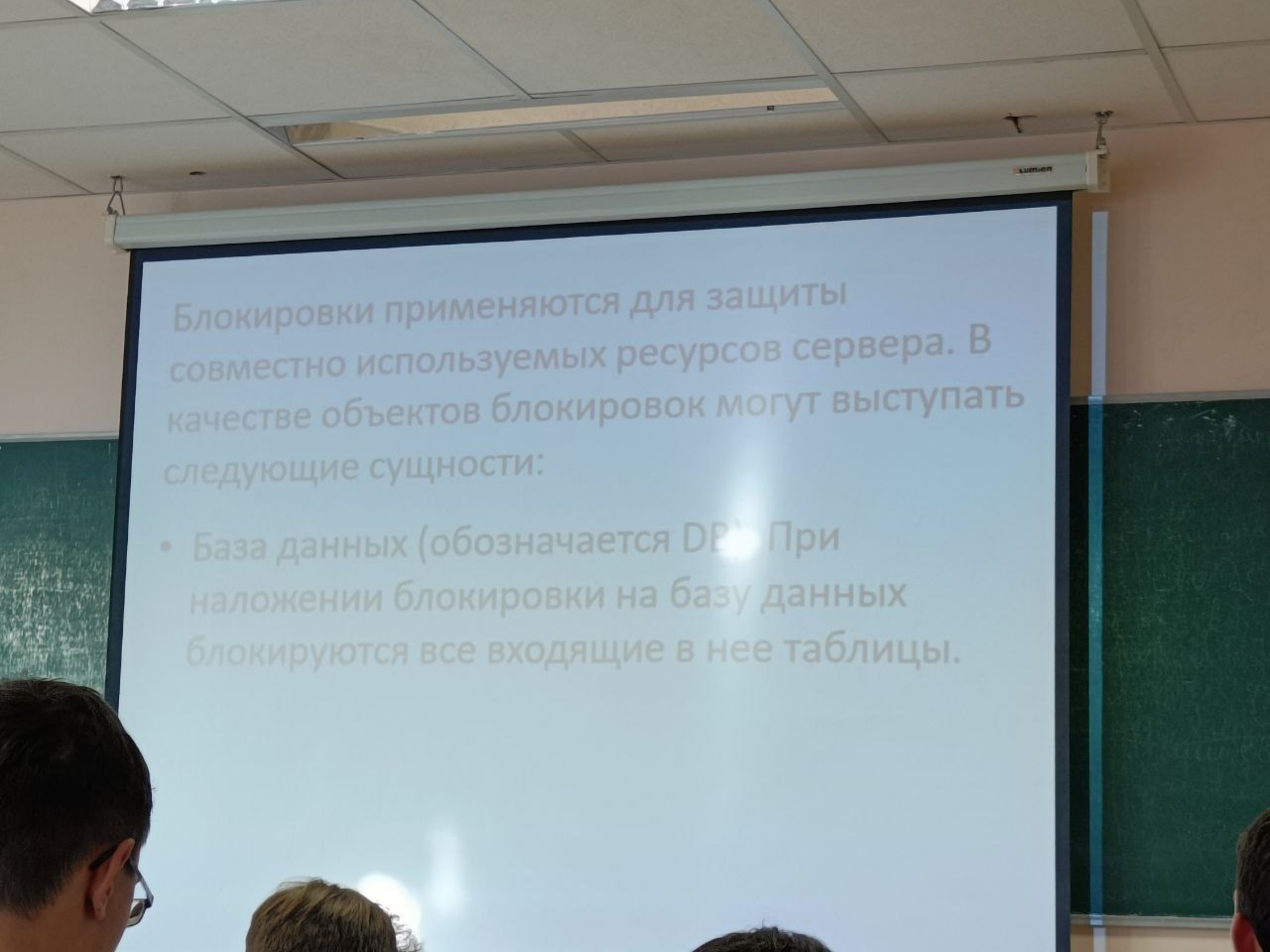


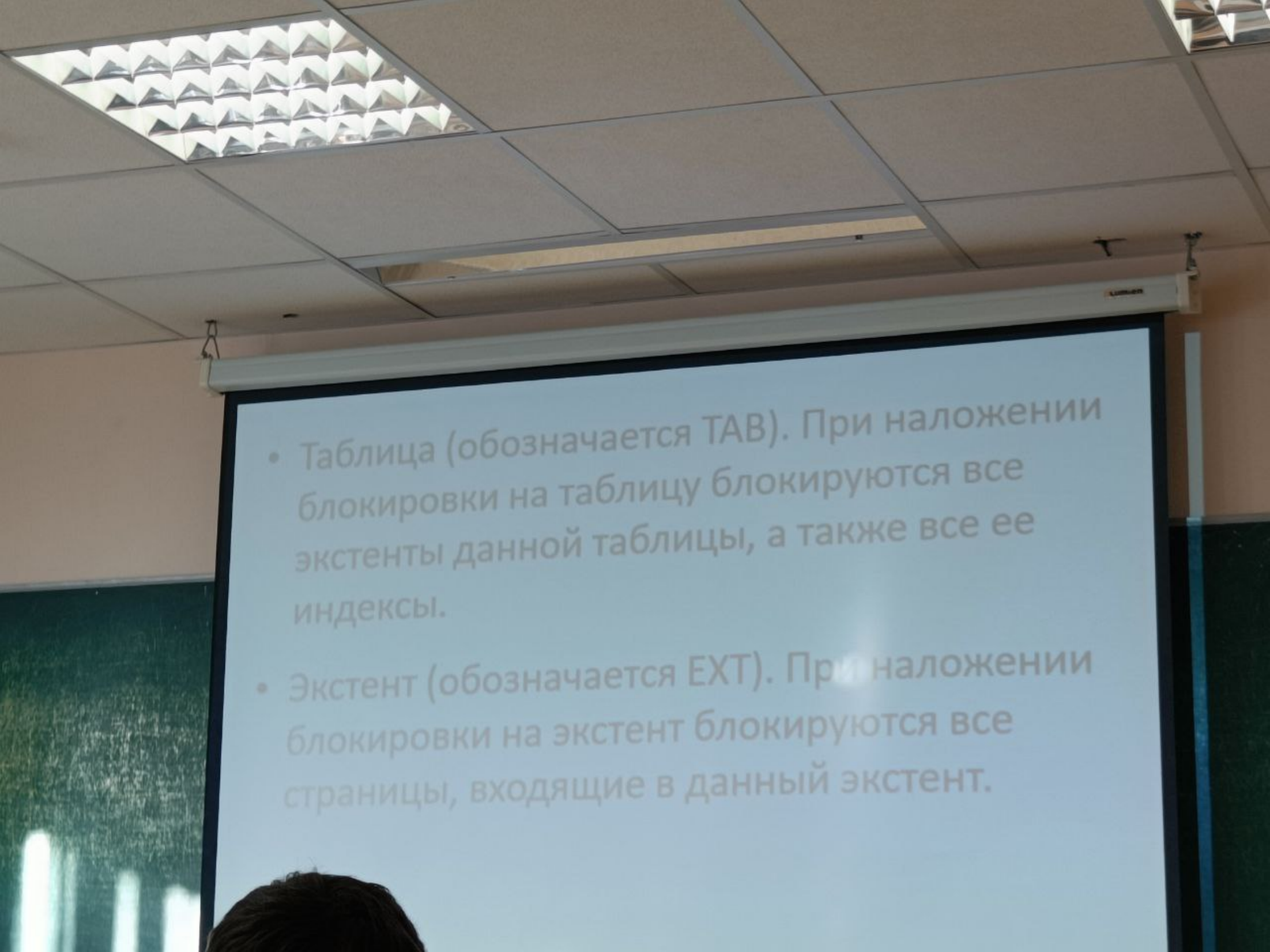

БЛОКИРОВКИ

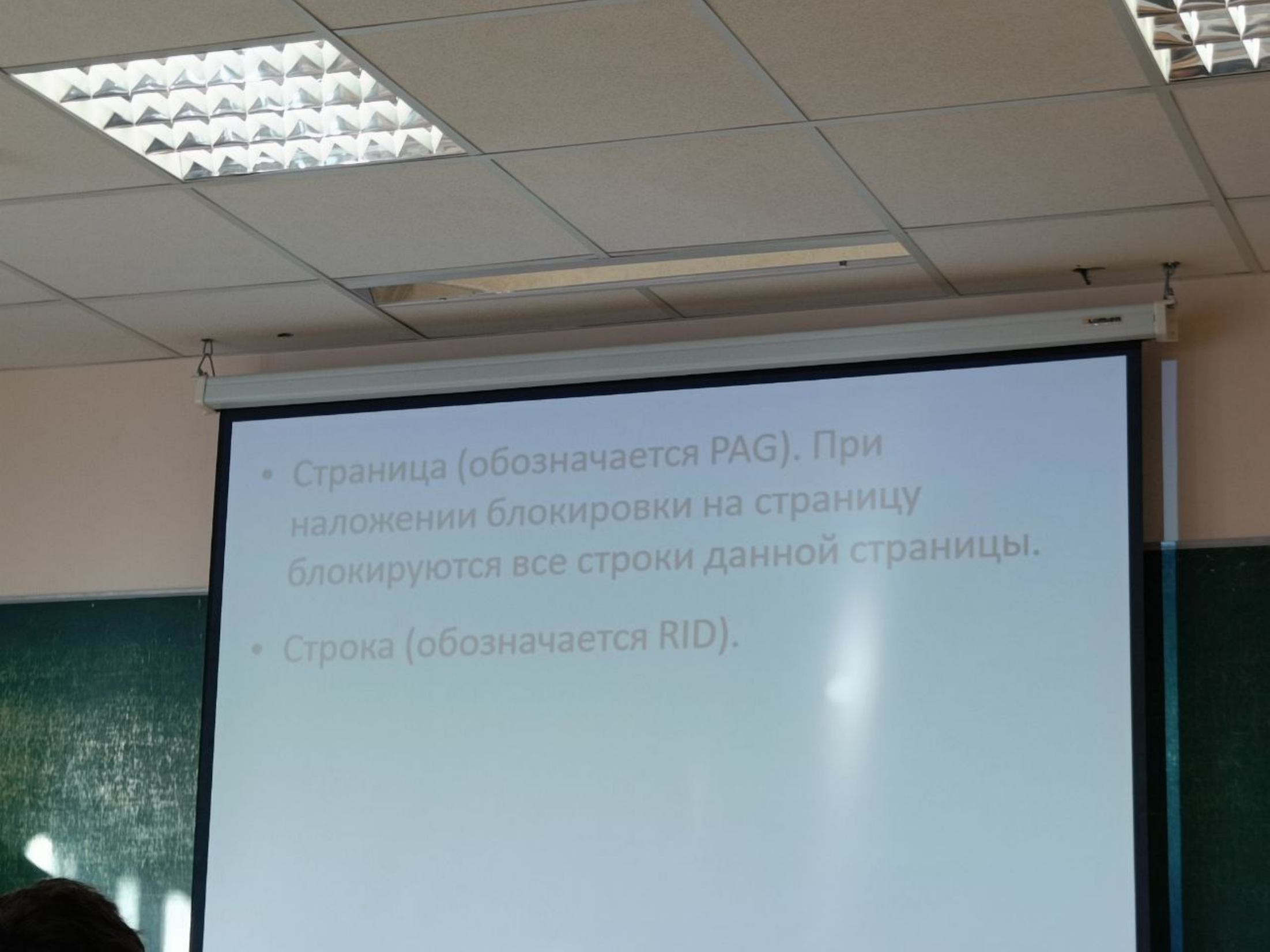

•Блокировки в MS SQL Server – это механизм реализации требования изолированности транзакций.
•При открытии новой сессии по умолчанию выбирается уровень изоляции READ COMMITTED.

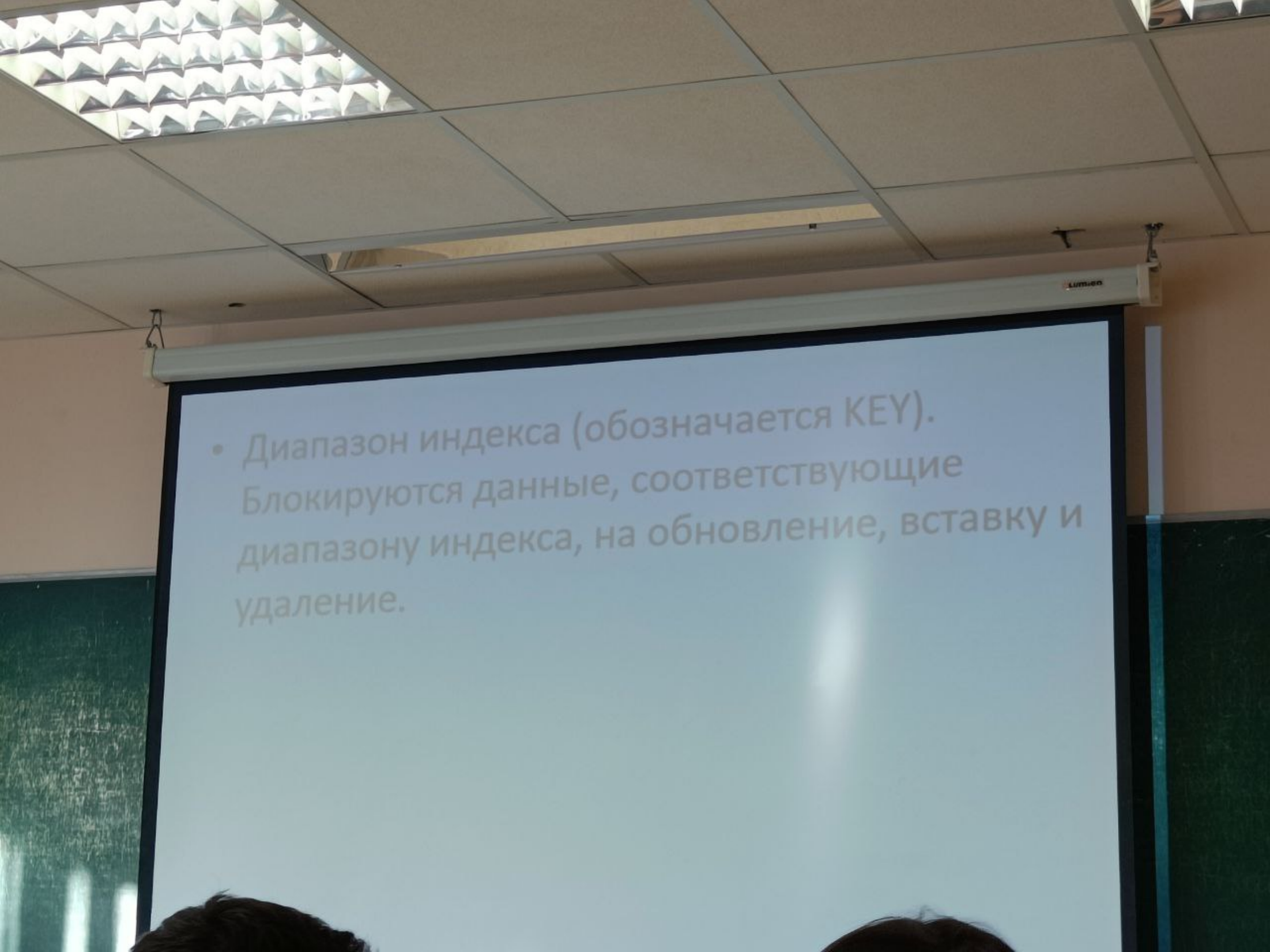

•Можно изменить этот уровень для данного соединения с помощью команды:
SET TRANSACTION ISOLATION LEVEL

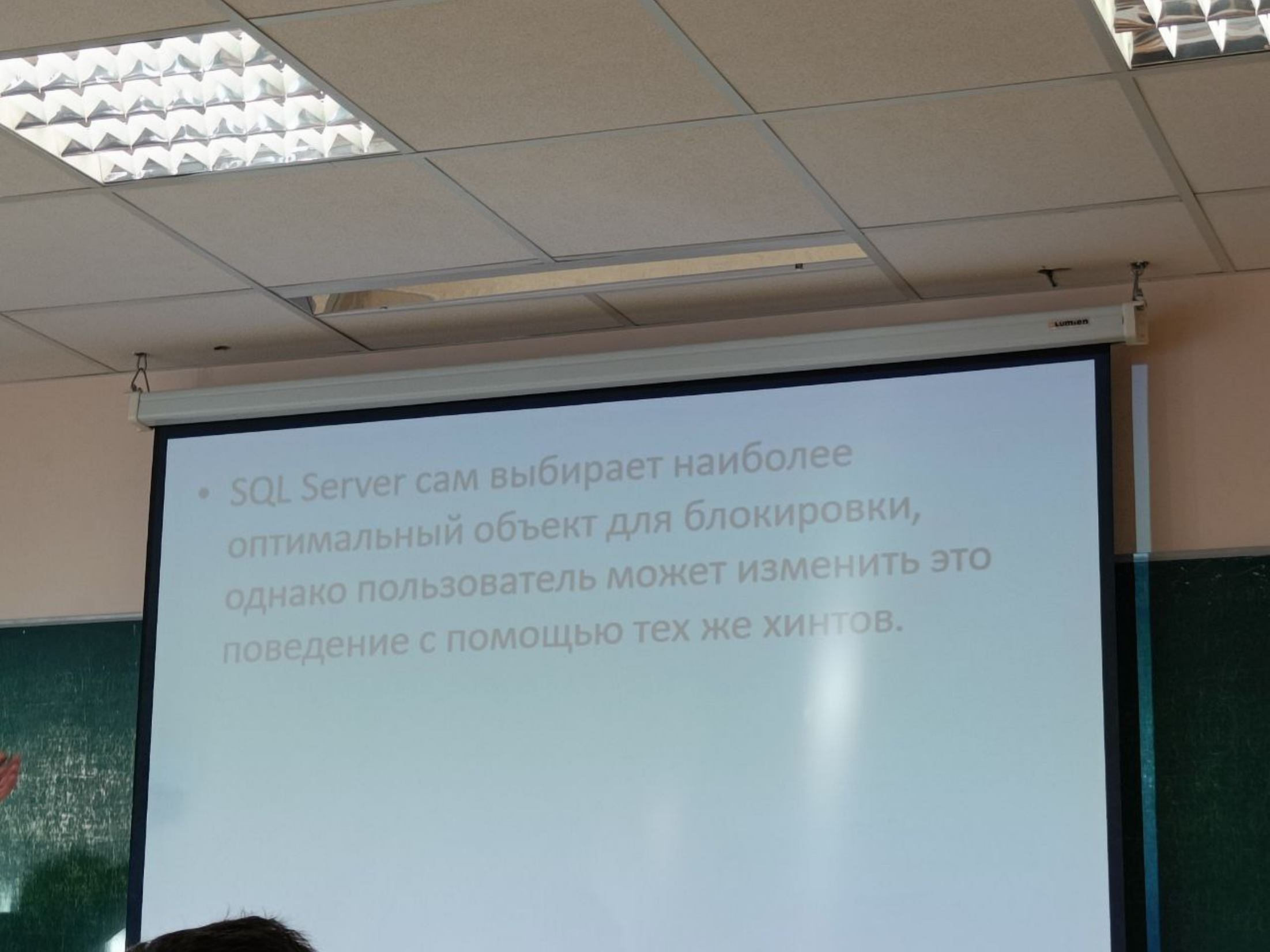

•Существует три основных типа блокировок и множество специфичных. Сервер устанавливает блокировки автоматически в зависимости от текущего уровня изоляции транзакции, однако при желании можно изменить тип с помощью специальных подсказок – хинтов.

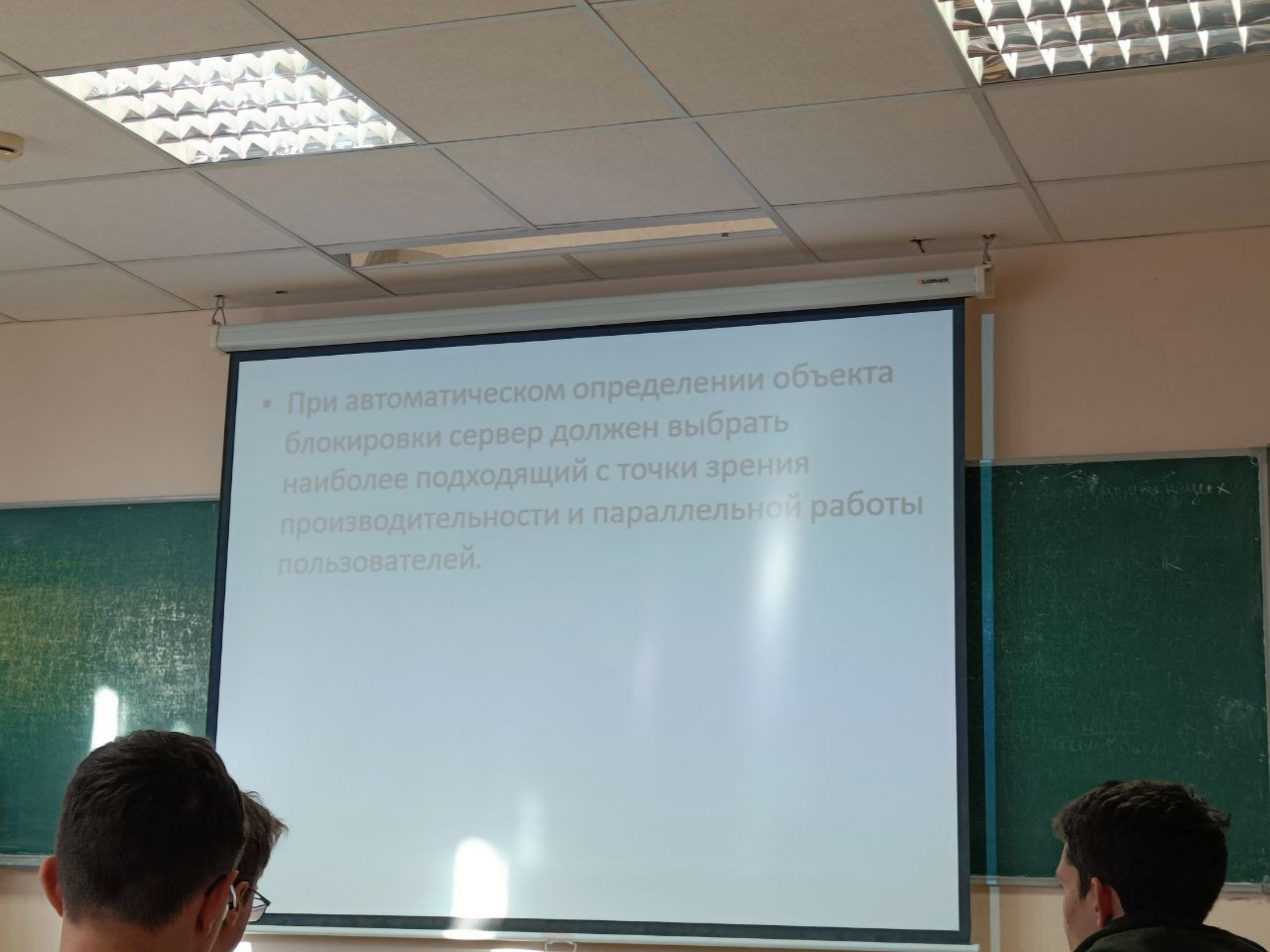

Блокировки применяются для защиты совместно используемых ресурсов сервера. В качестве объектов блокировок могут выступать следующие сущности:
• База данных (обозначается DB). При наложении блокировки на базу данных блокируются все входящие в нее таблицы.

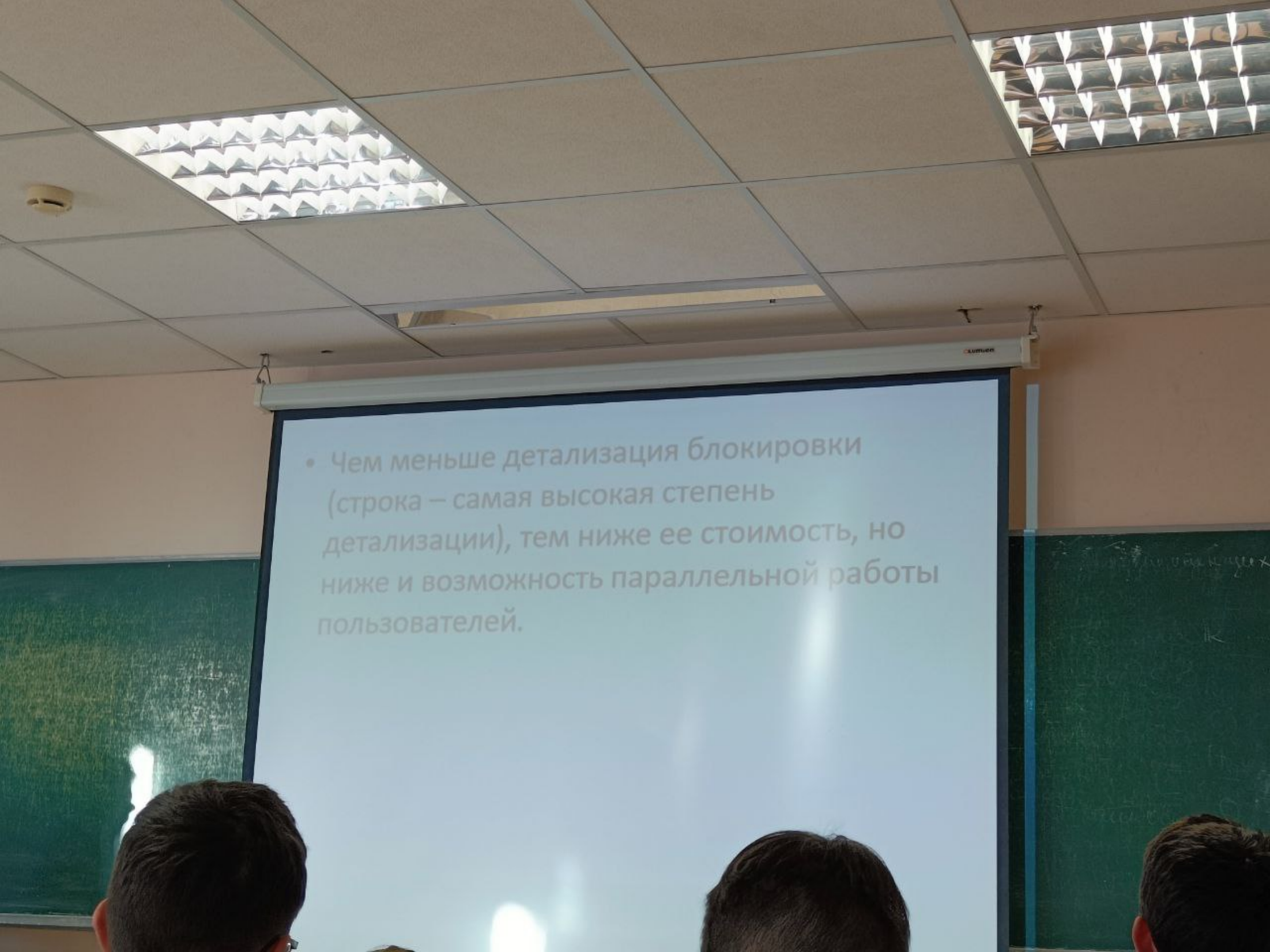

• Таблица (обозначается TAB). При наложении блокировки на таблицу блокируются все экстенты данной таблицы, а также все ее индексы.
• Экстент (обозначается EXT). При наложении блокировки на экстент блокируются все страницы, входящие в данный экстент.

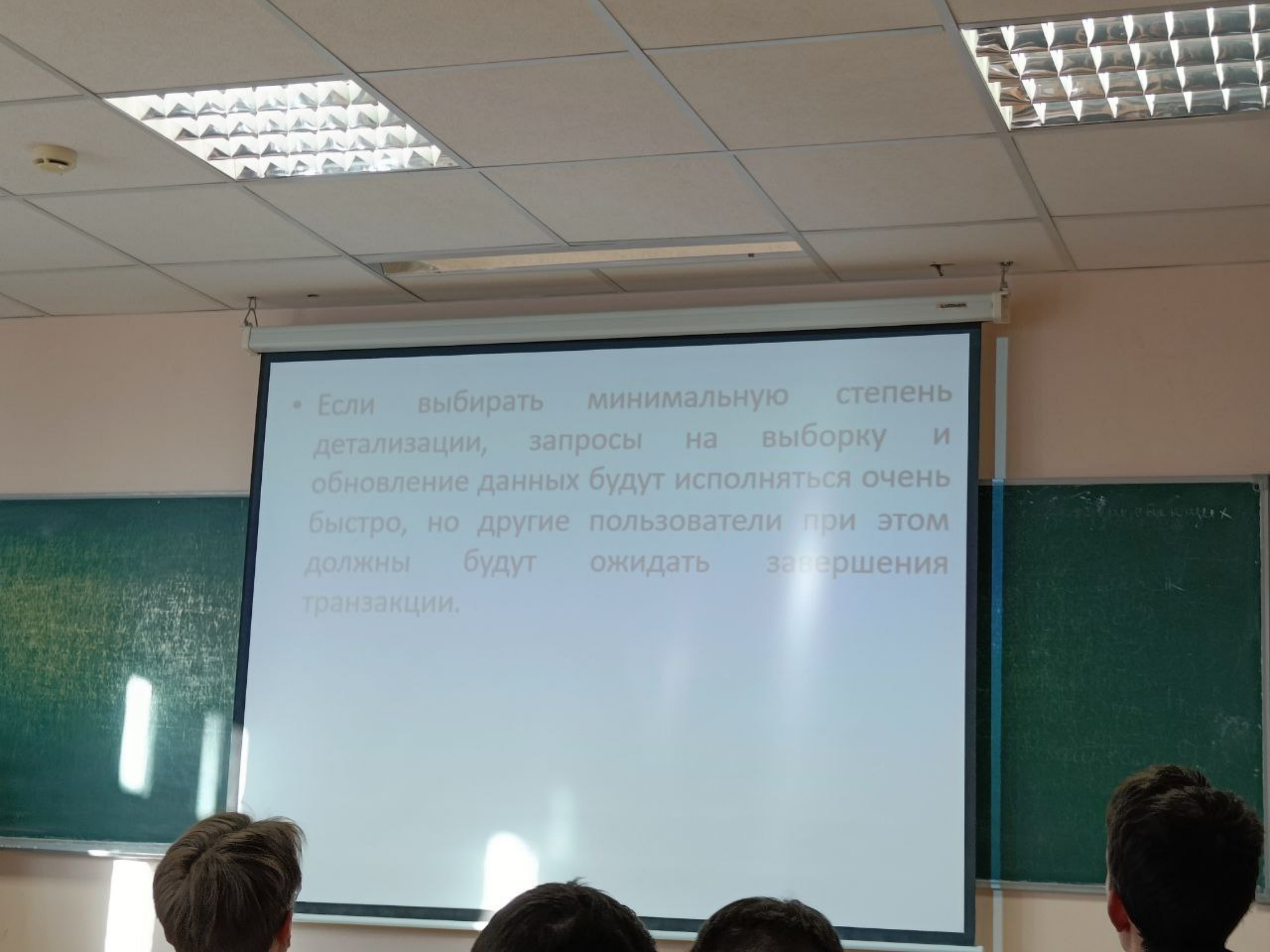

• Страница (обозначается PAG). При наложении блокировки на страницу блокируются все строки данной страницы.
• Строка (обозначается RID).

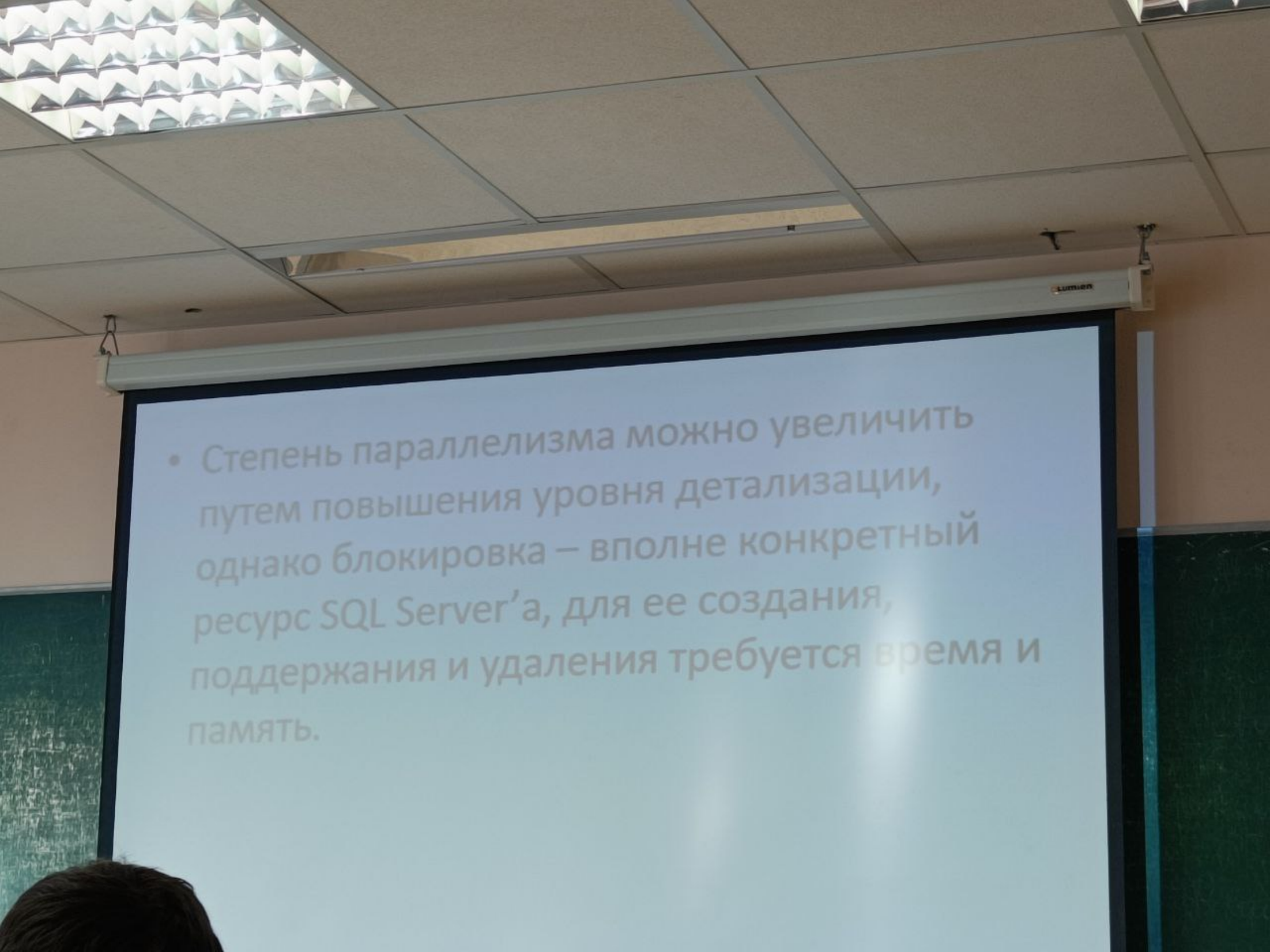

• Диапазон индекса (обозначается KEY). Блокируются данные, соответствующие диапазону индекса, на обновление, вставку и удаление.

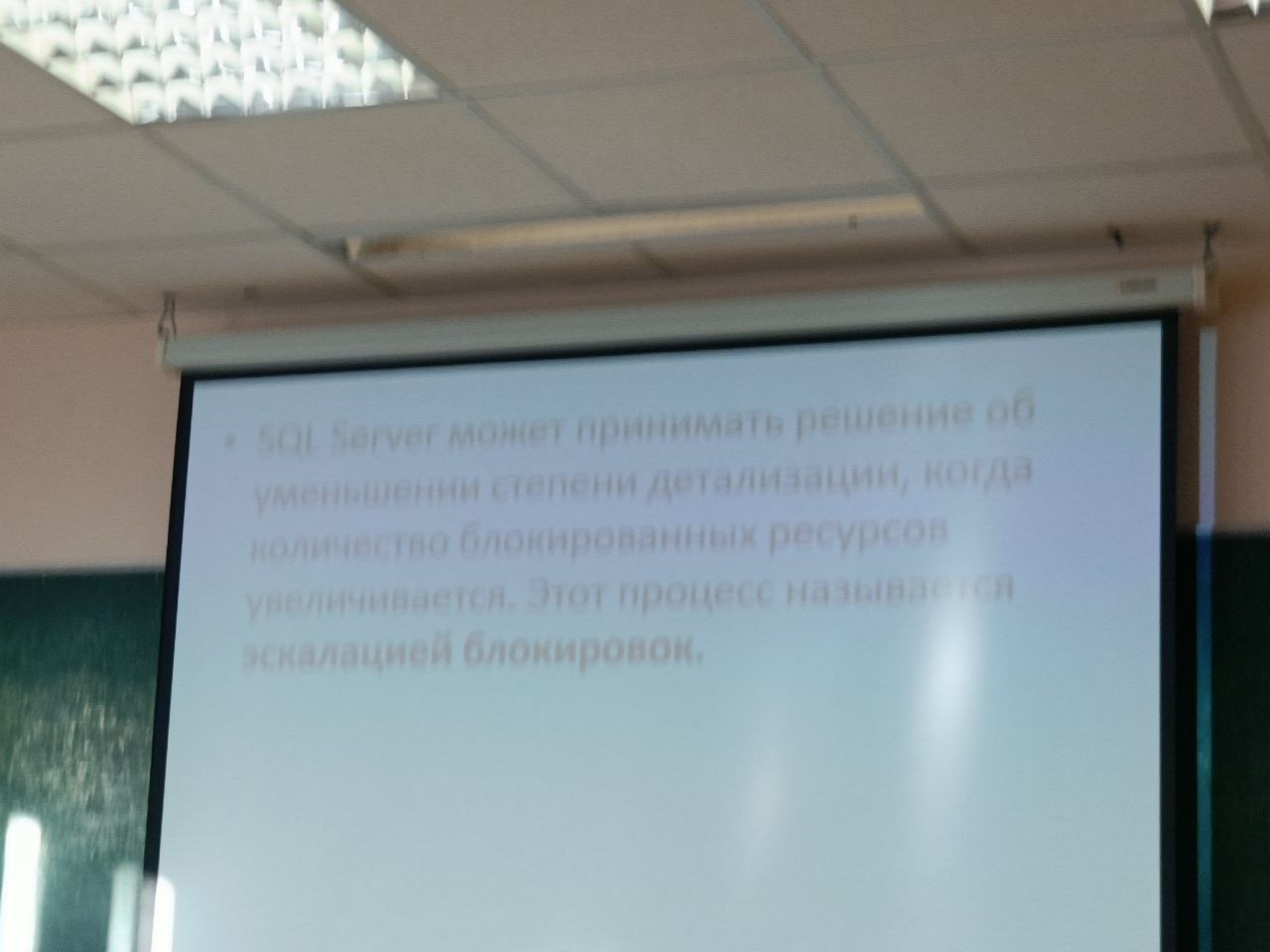

• SQL Server сам выбирает наиболее оптимальный объект для блокировки, однако пользователь может изменить это поведение с помощью тех же хинтов.

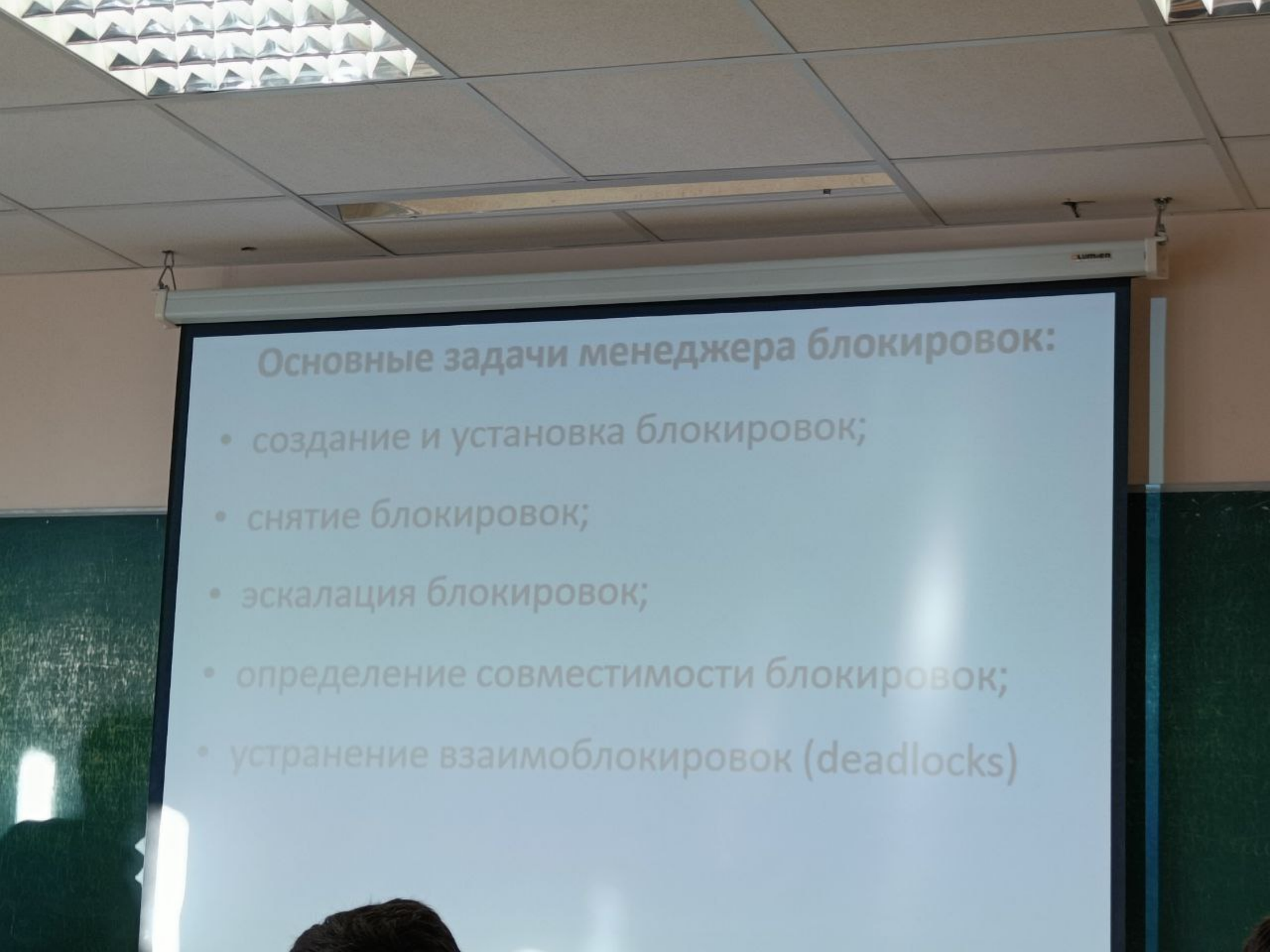

• При автоматическом определении объекта блокировки сервер должен выбрать наиболее подходящий с точки зрения производительности и параллельной работы пользователей.

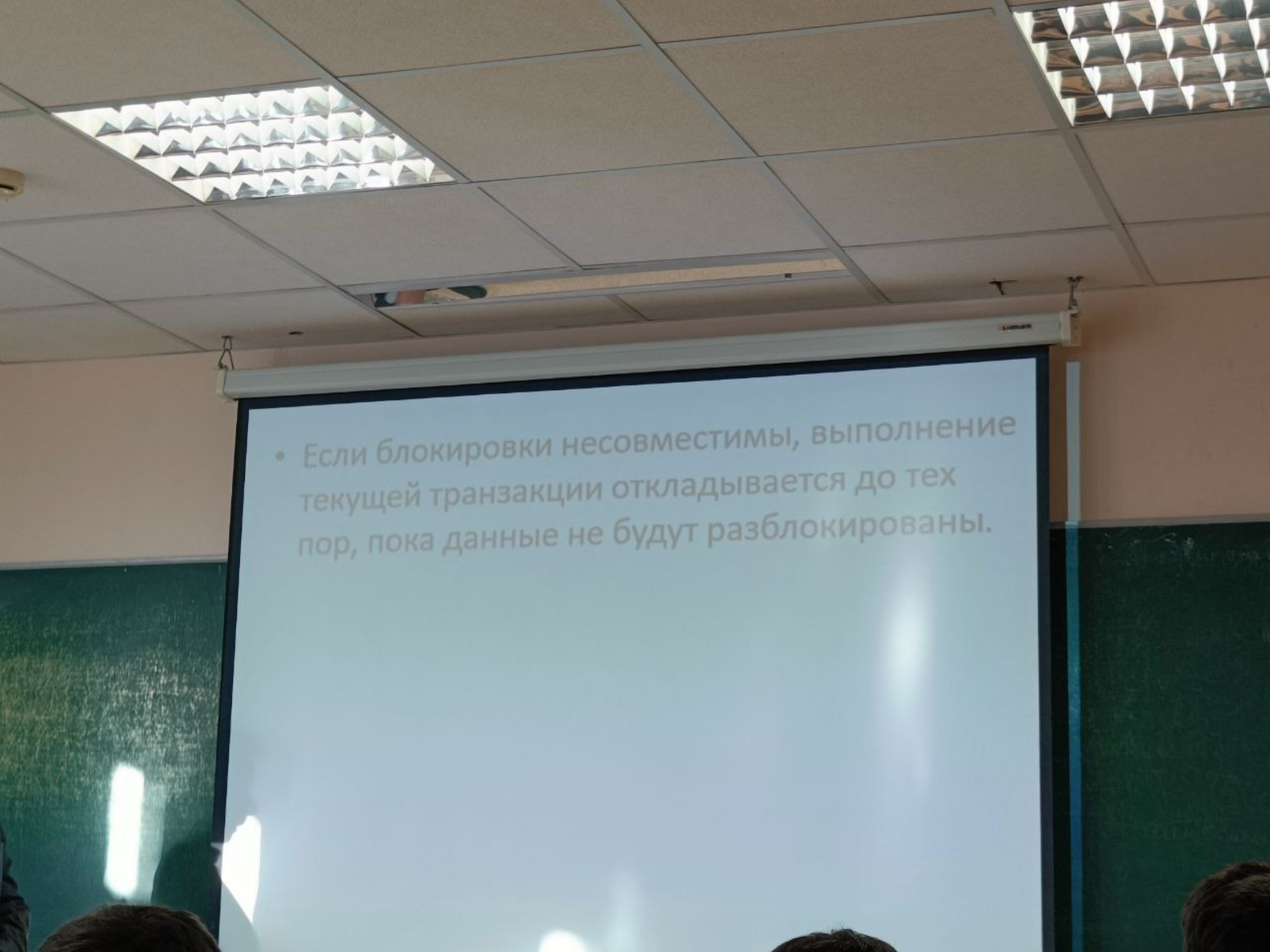

• Чем меньше детализация блокировки (строка – самая высокая степень детализации), тем ниже ее стоимость, но ниже и возможность параллельной работы пользователей.

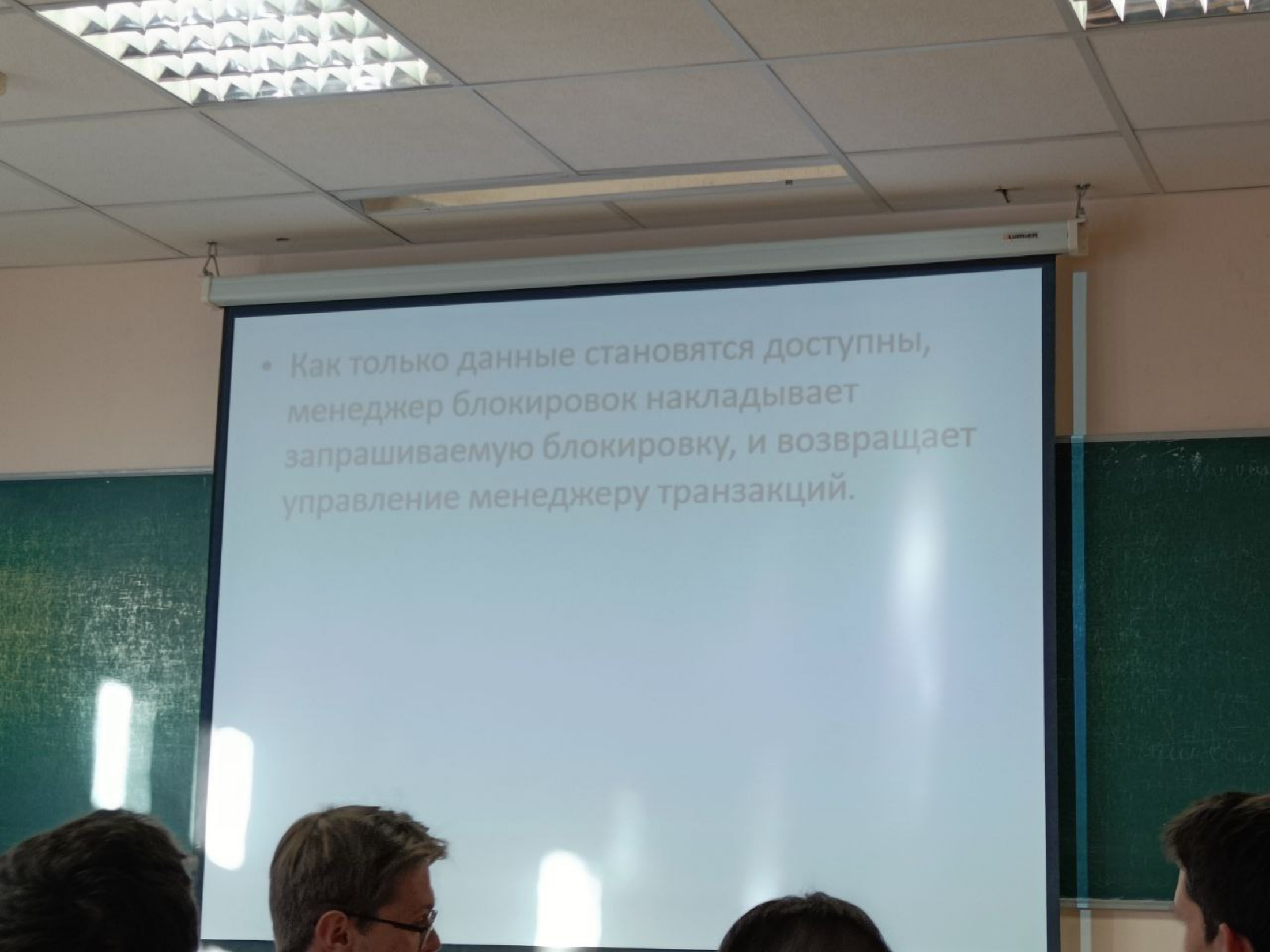

• Если выбирать минимальную степень детализации, запросы на выборку и обновление данных будут исполняться очень быстро, но другие пользователи при этом должны будут ожидать завершения транзакции.

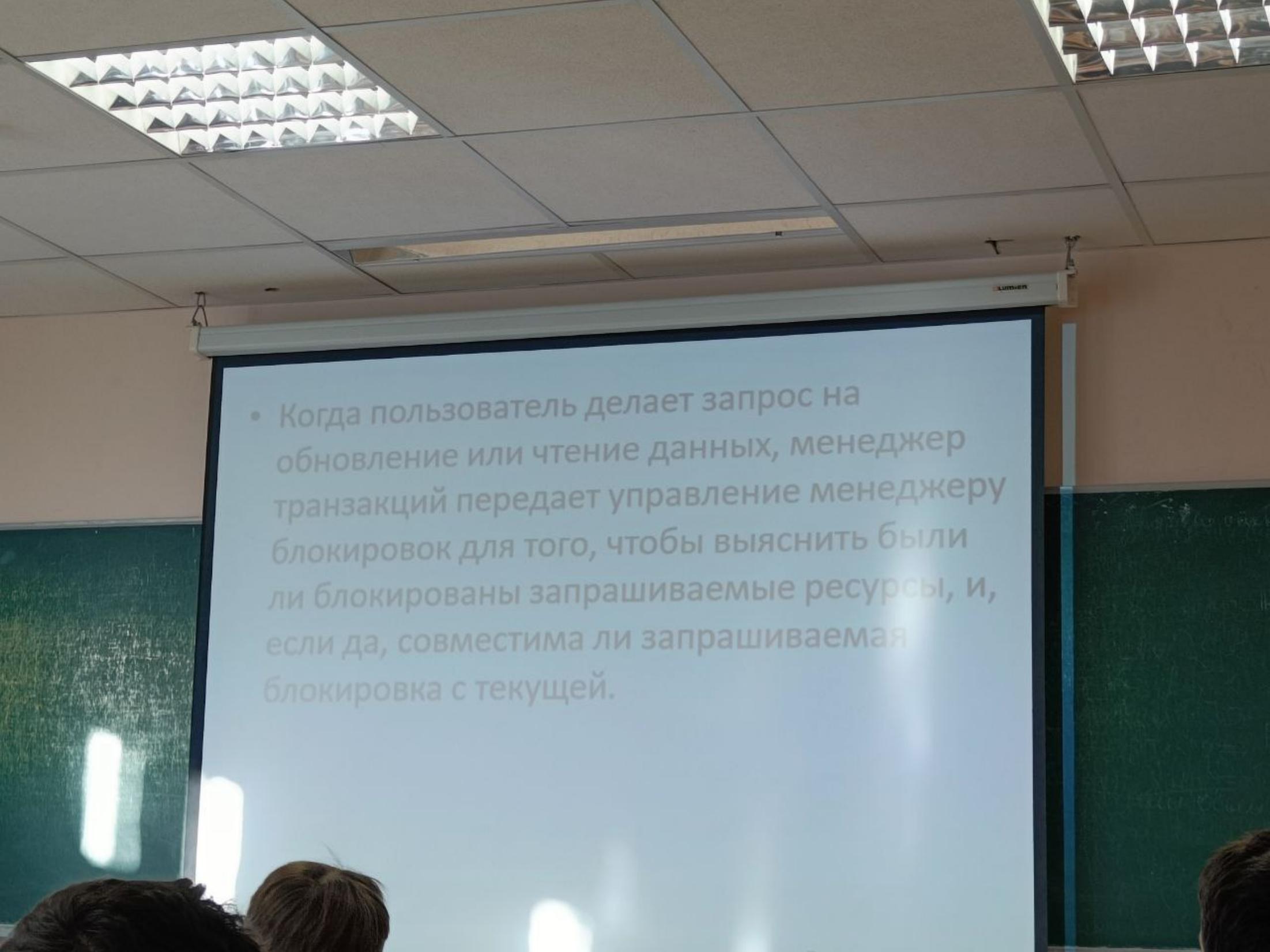

• Степень параллелизма можно увеличить путем повышения уровня детализации, однако блокировка – вполне конкретный ресурс SQL Server’а, для ее создания, поддержания и удаления требуется время и память.

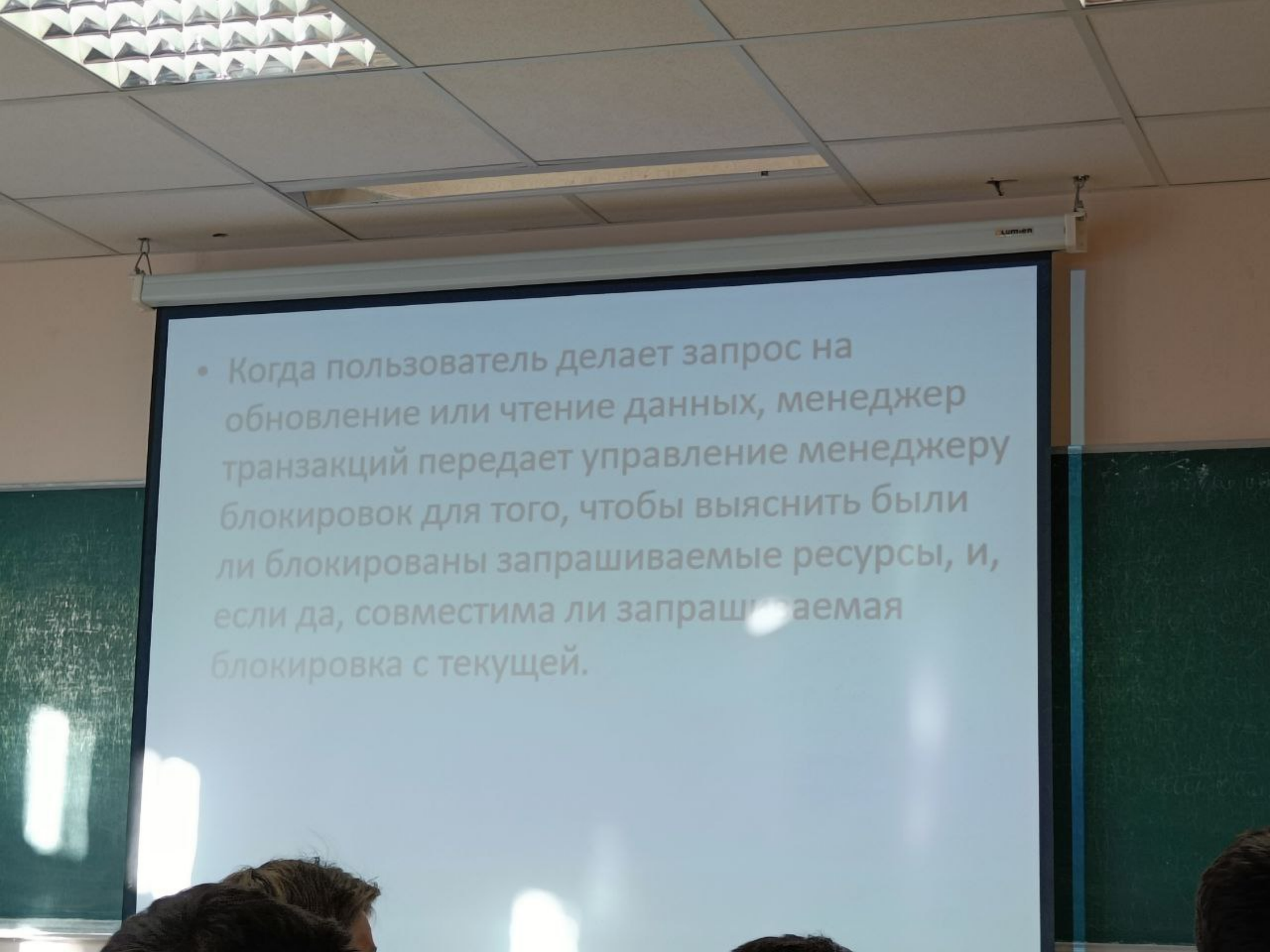

• SQL Server может принимать решение об
уменьшении степени детализации, когда
количество блокированных ресурсов
увеличивается. Этот процесс называется
эскалацией блокировок.

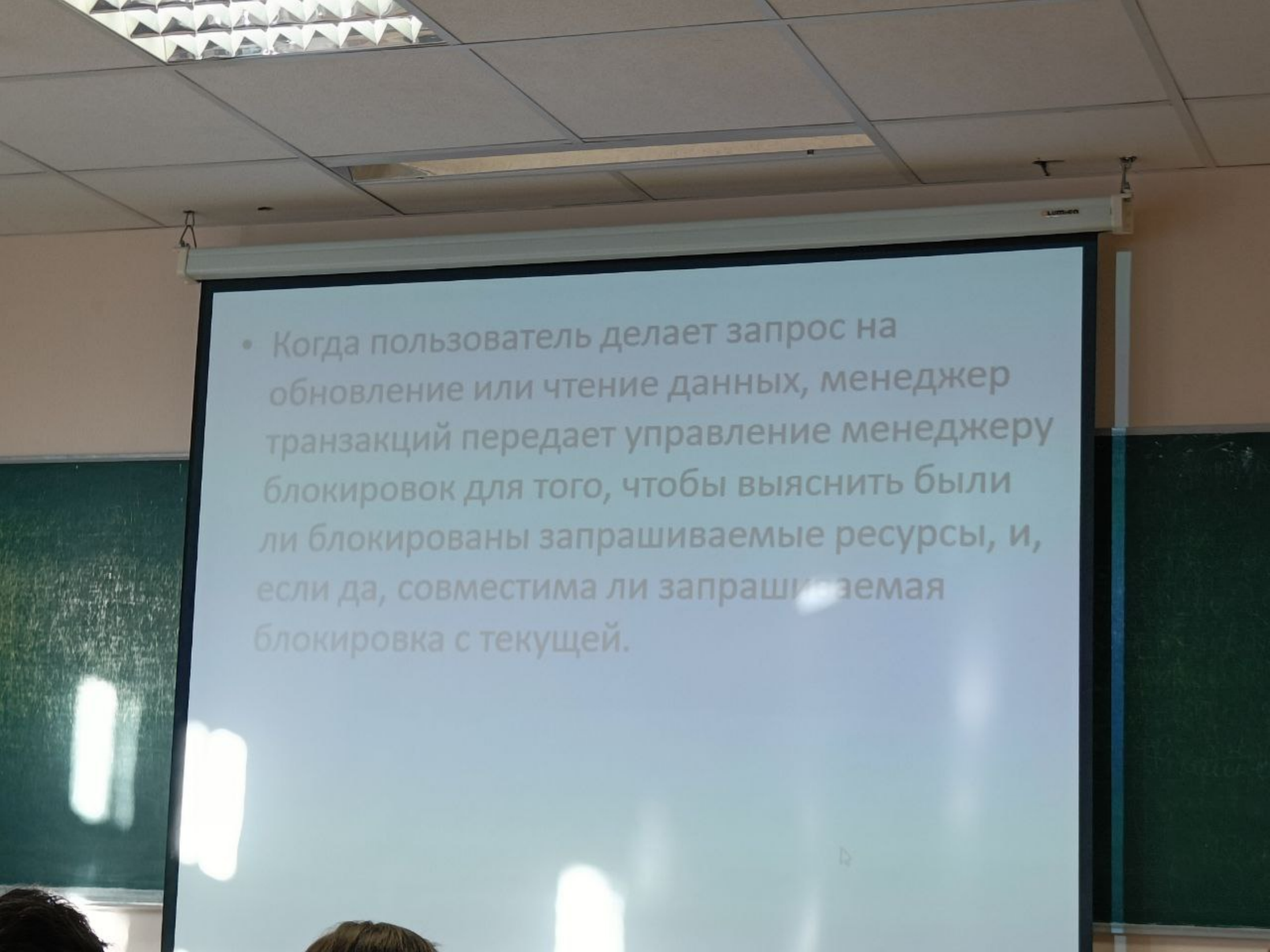

Основные задачи менеджера блокировок:
• создание и установка блокировок;
• снятие блокировок;
• эскалация блокировок;
• определение совместимости блокировок;
• устранение взаимоблокировок (deadlocks)


• Если блокировки несовместимы, выполнение текущей транзакции откладывается до тех пор, пока данные не будут разблокированы.


• Как только данные становятся доступны, менеджер блокировок накладывает запрашиваемую блокировку, и возвращает управление менеджеру транзакций.


• Когда пользователь делает запрос на обновление или чтение данных, менеджер транзакций передает управление менеджеру блокировок для того, чтобы выяснить были ли блокированы запрашиваемые ресурсы, и, если да, совместима ли запрашиваемая блокировка с текущей.


• Когда пользователь делает запрос на обновление или чтение данных, менеджер транзакций передает управление менеджеру блокировок для того, чтобы выяснить были ли блокированы запрашиваемые ресурсы, и, если да, совместима ли запрашиваемая блокировка с текущей.


• Когда пользователь делает запрос на обновление или чтение данных, менеджер транзакций передает управление менеджеру блокировок для того, чтобы выяснить были ли блокированы запрашиваемые ресурсы, и, если да, совместима ли запрашиваемая блокировка с текущей.