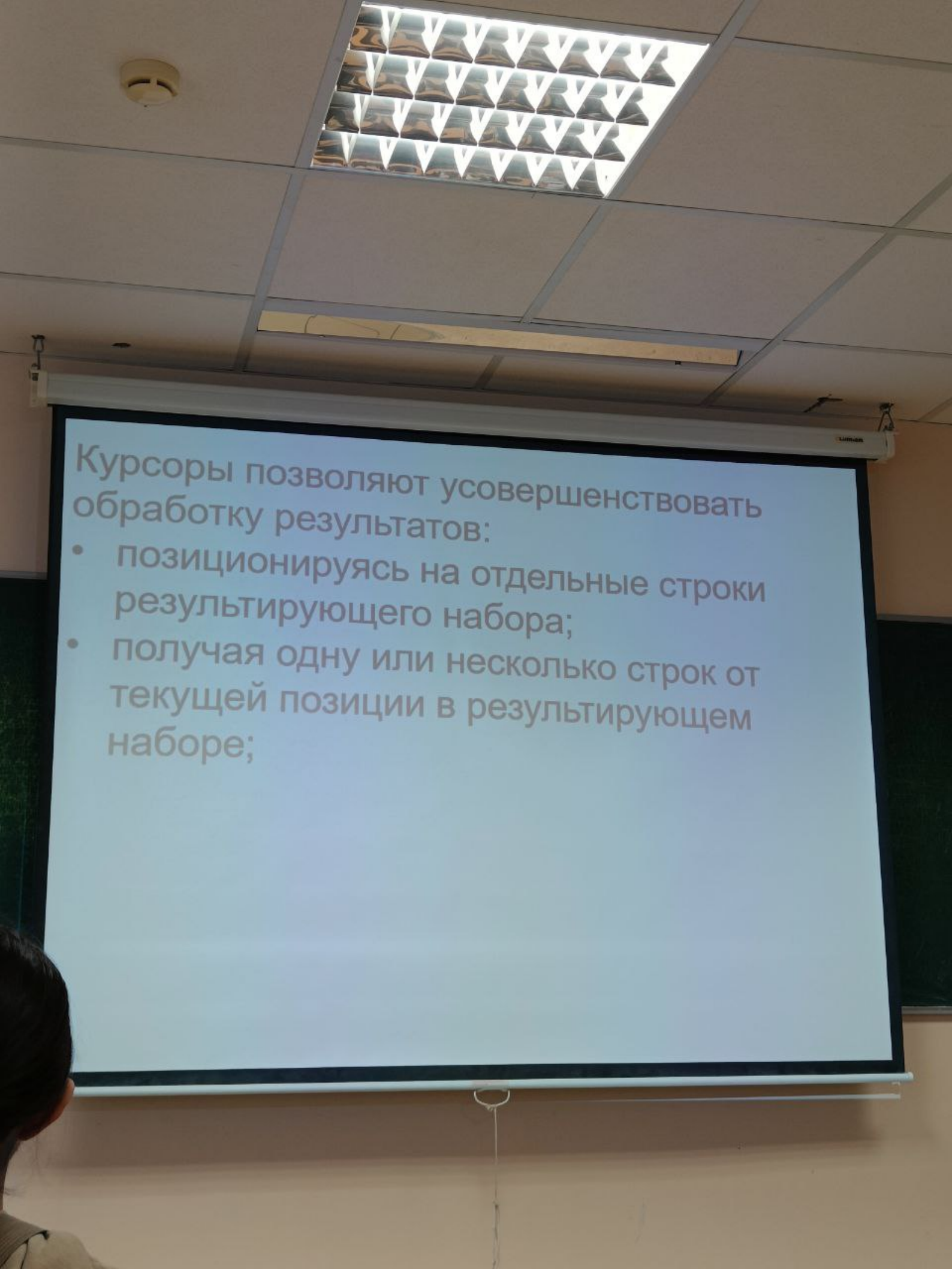
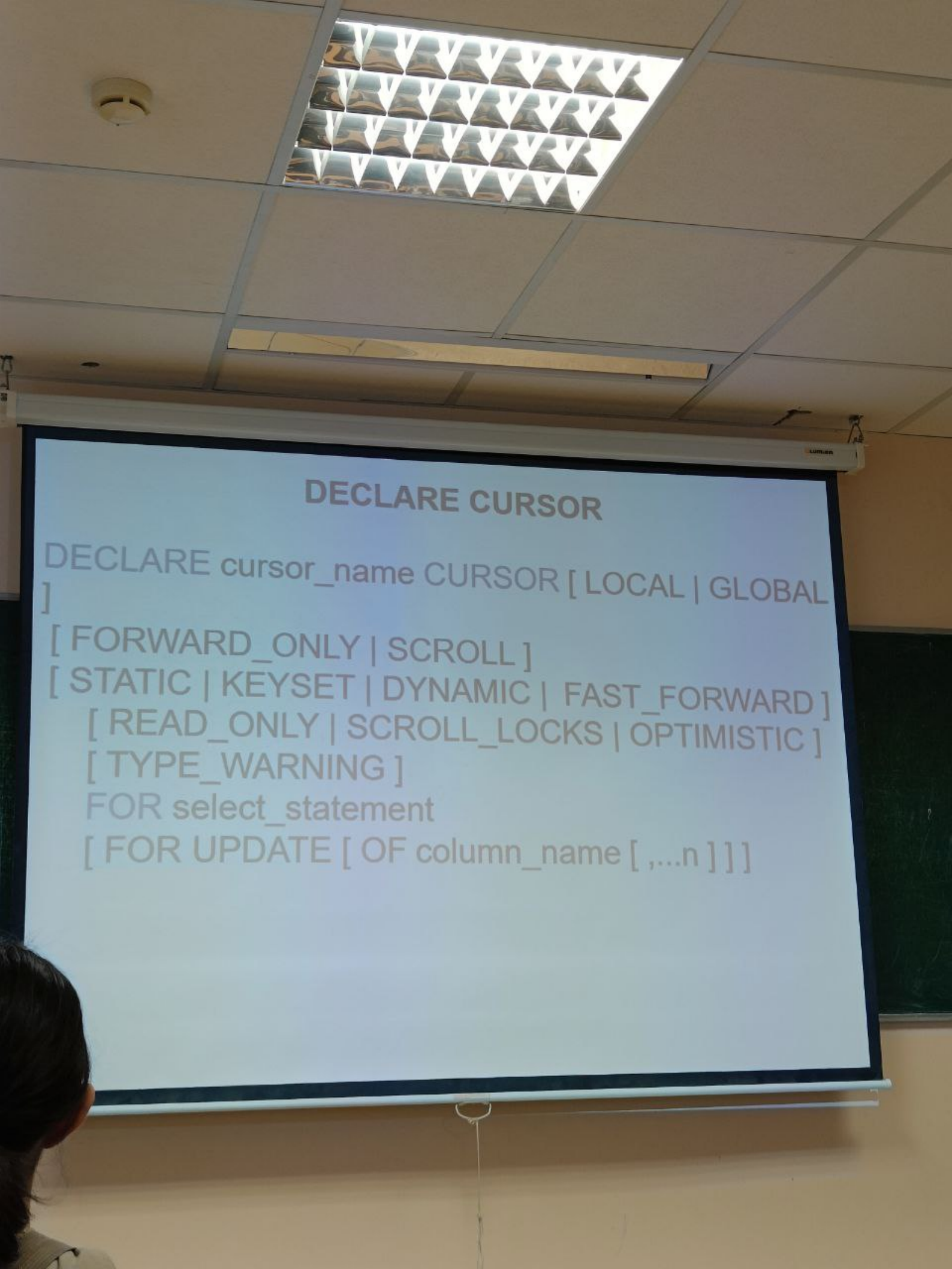
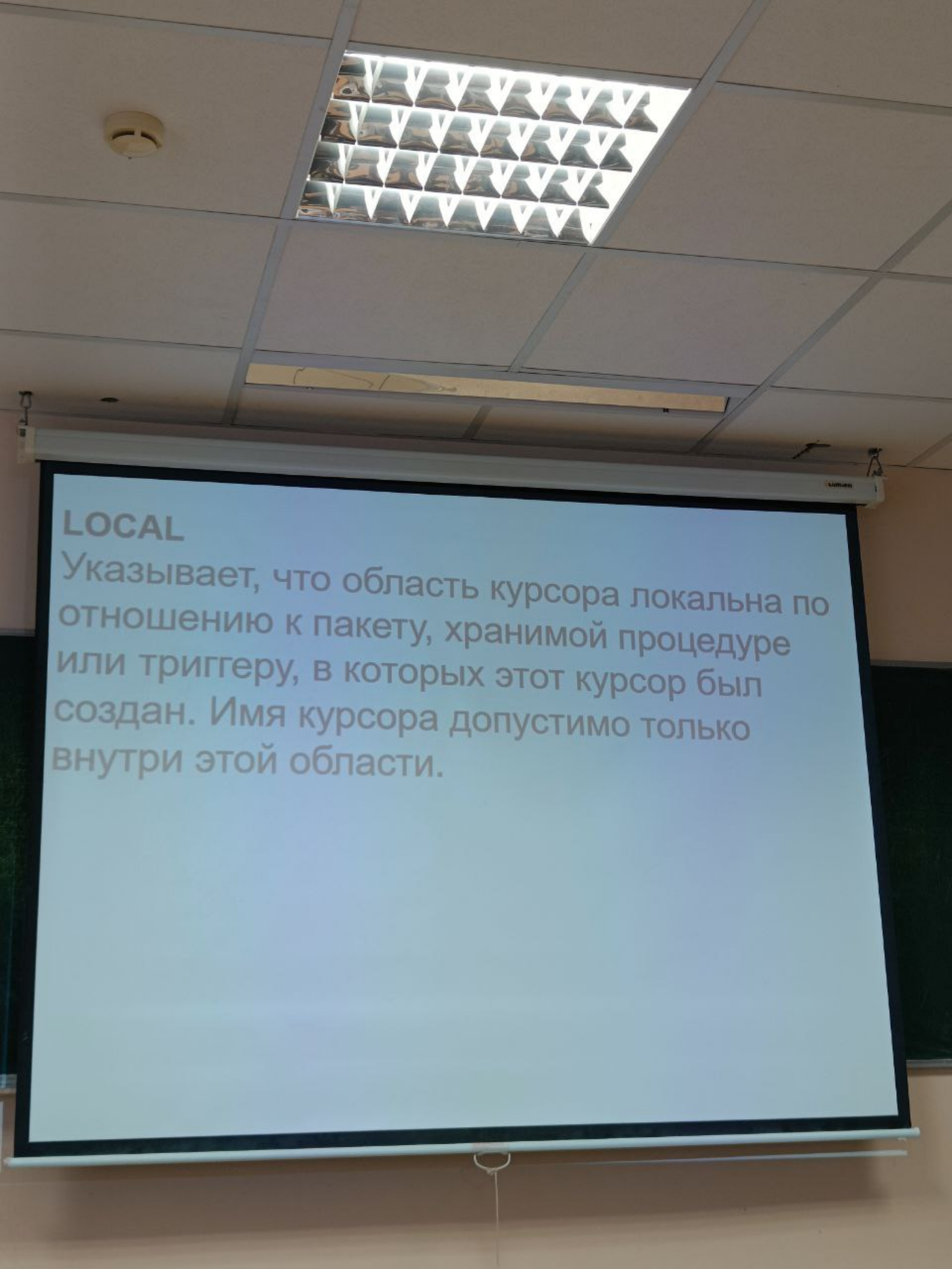
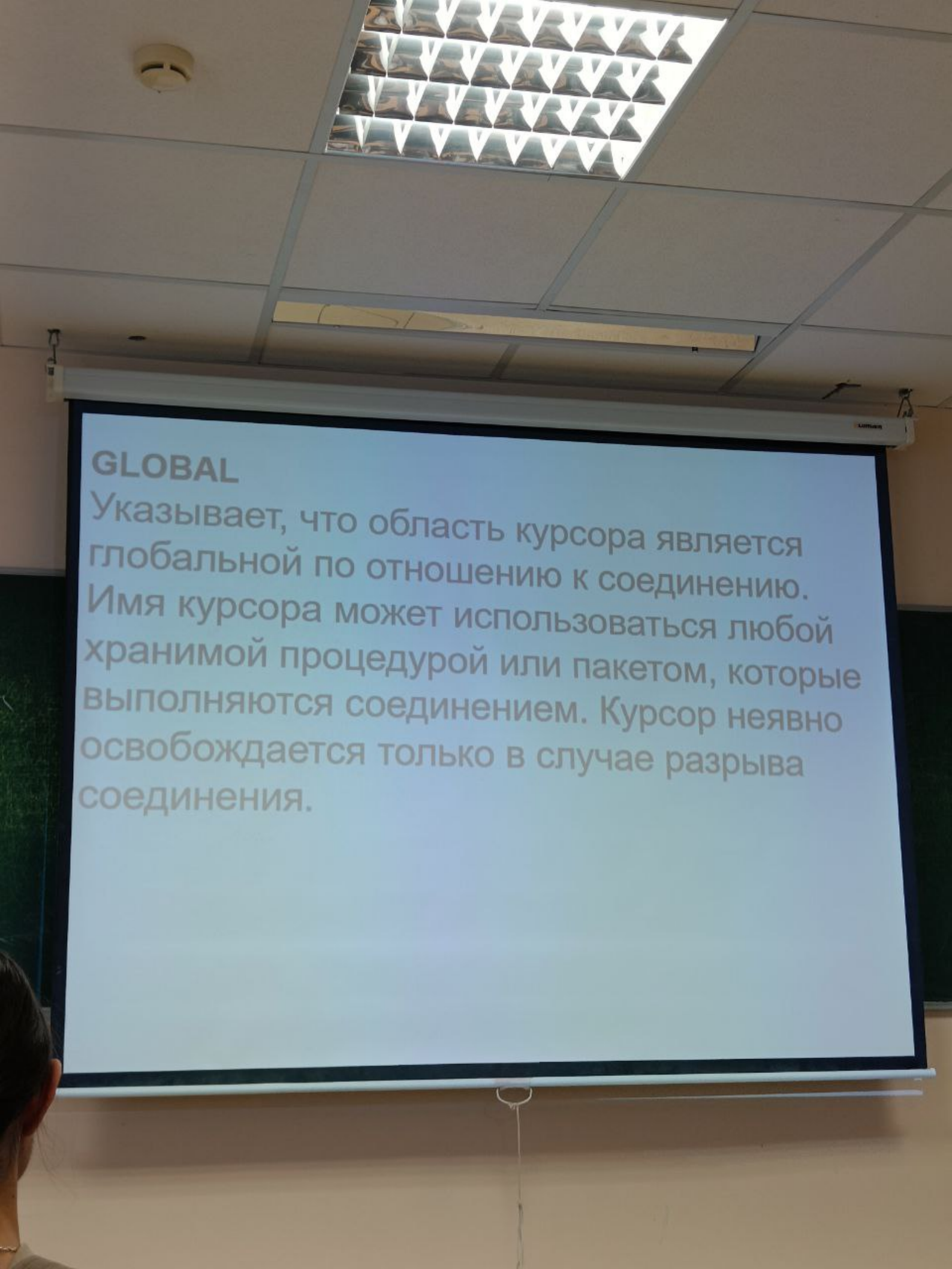
Курсоры позволяют усовершенствовать обработку результатов:

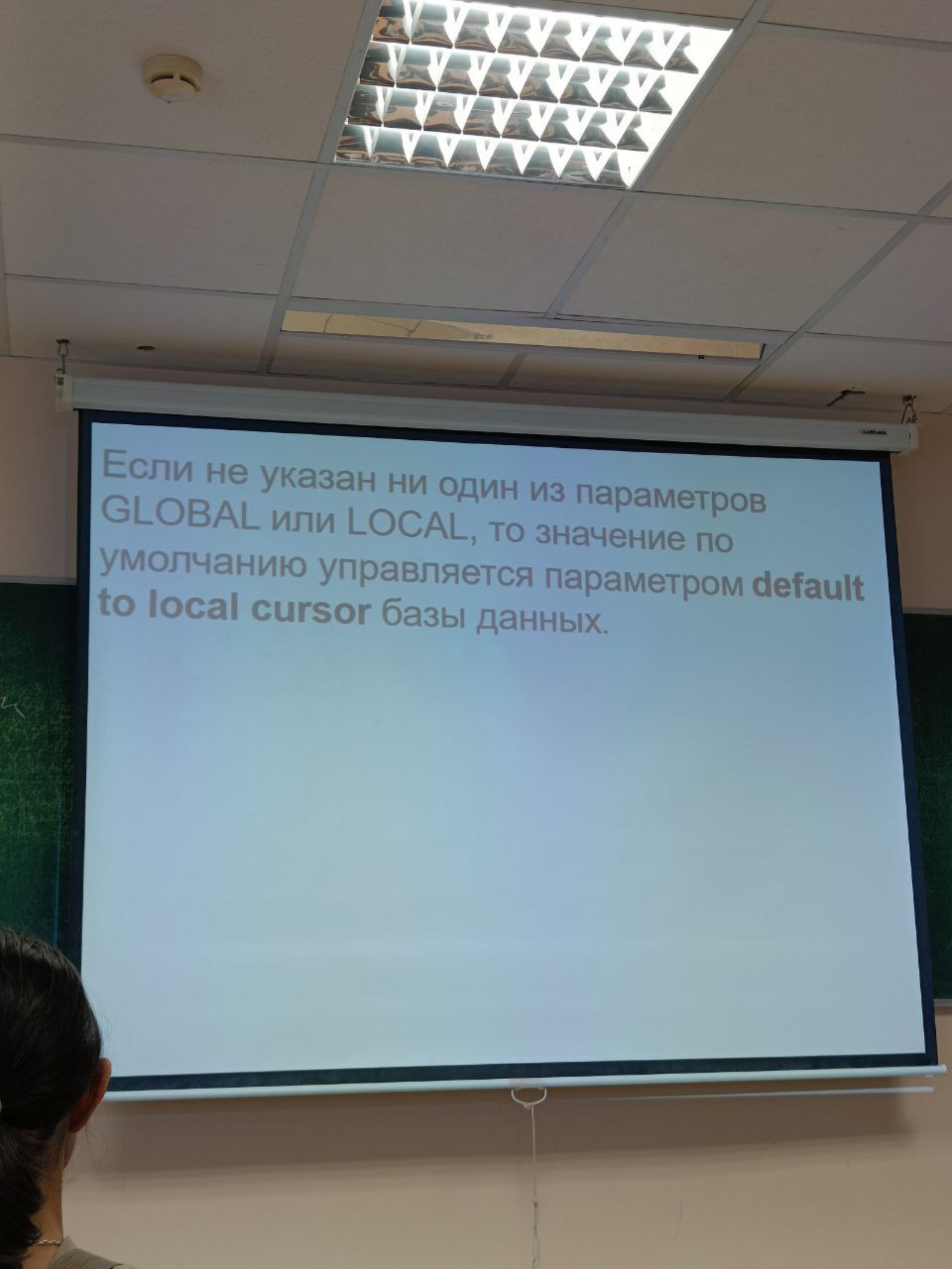
- позиционируясь на отдельные строки результирующего набора;
- получая одну или несколько строк от текущей позиции в результирующем наборе;

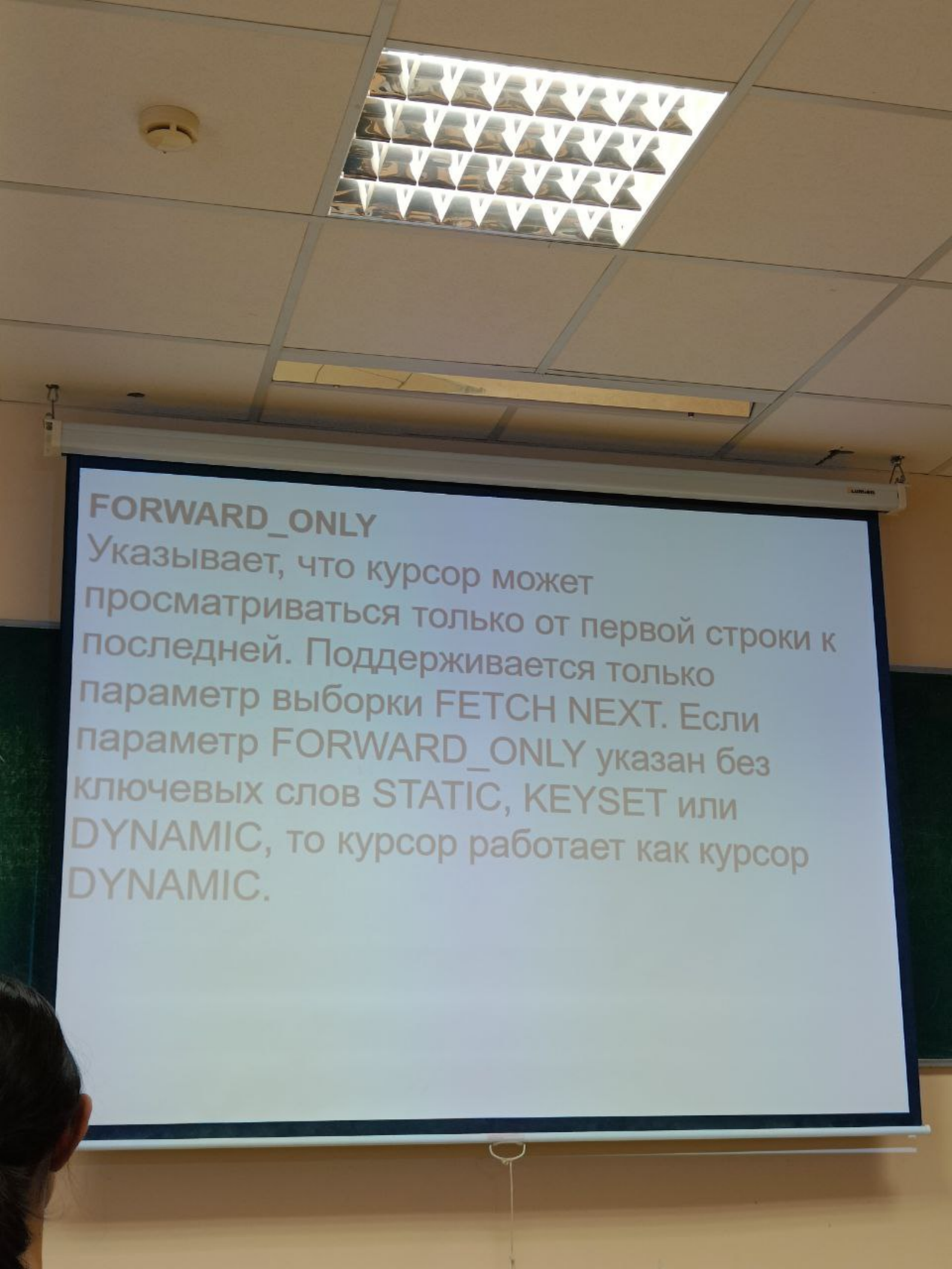

Курсоры позволяют усовершенствовать обработку результатов:
• позиционируясь на отдельные строки результирующего набора;
• получая одну или несколько строк от текущей позиции в результирующем наборе;

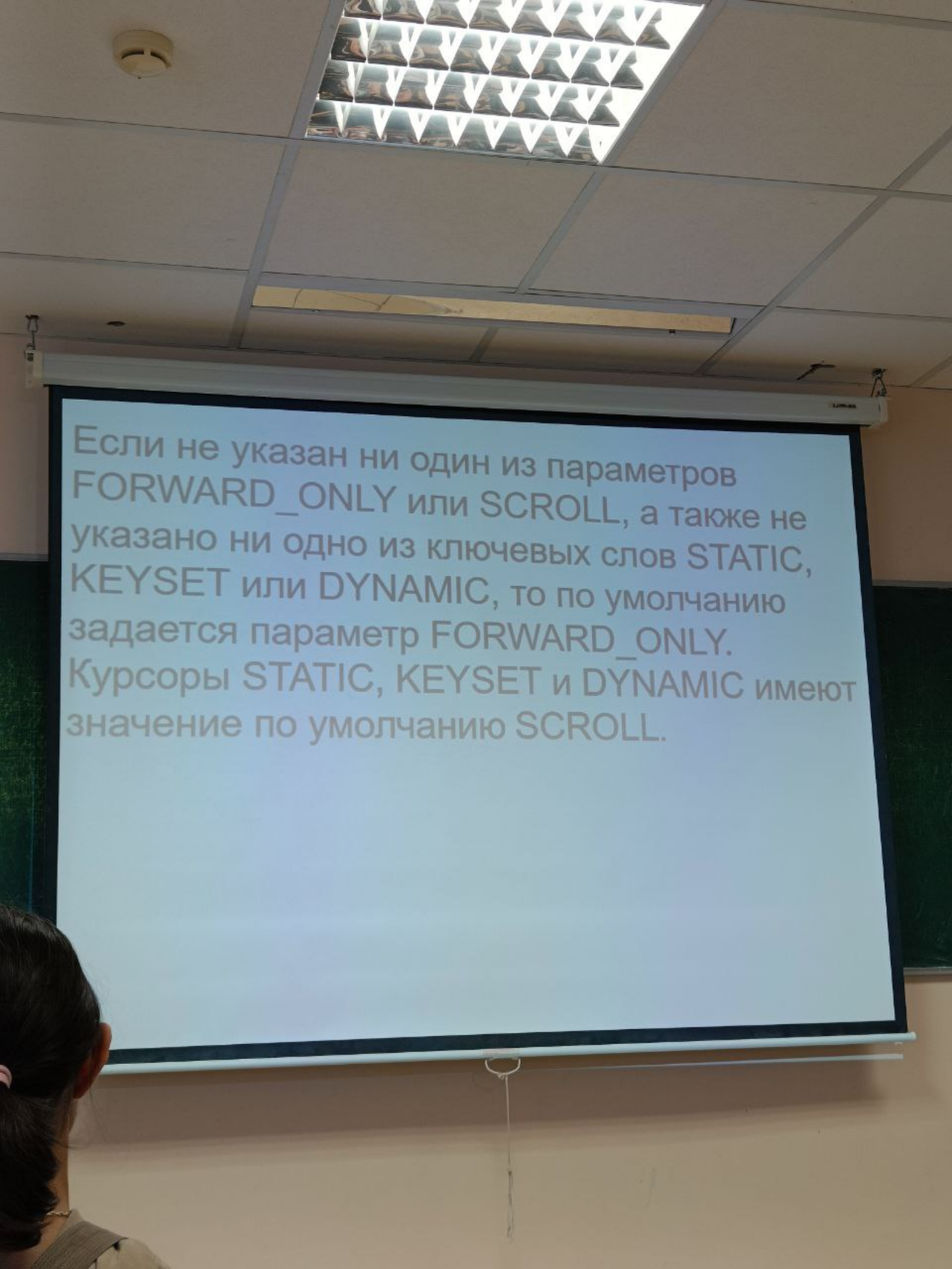

DECLARE CURSOR
DECLARE cursor_name CURSOR [LOCAL | GLOBAL]
[FORWARD_ONLY | SCROLL]
[STATIC | KEYSET | DYNAMIC | FAST_FORWARD]
[READ_ONLY | SCROLL_LOCKS | OPTIMISTIC]
[TYPE_WARNING]
FOR select_statement
[FOR UPDATE [OF column_name [,...n]]]

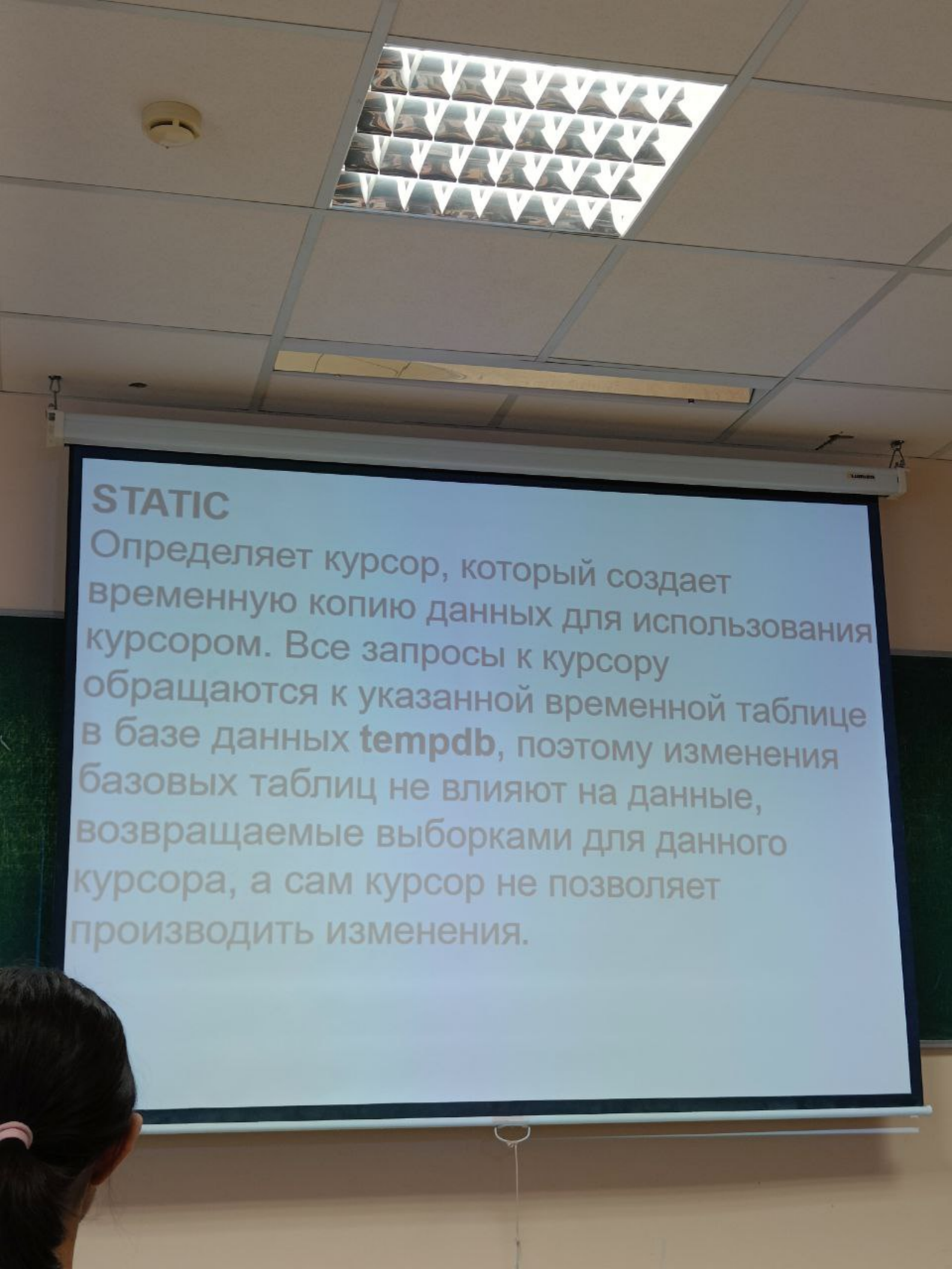

LOCAL
Указывает, что область курсора локальна по отношению к пакету, хранимой процедуре или триггеру, в которых этот курсор был создан. Имя курсора допустимо только внутри этой области.

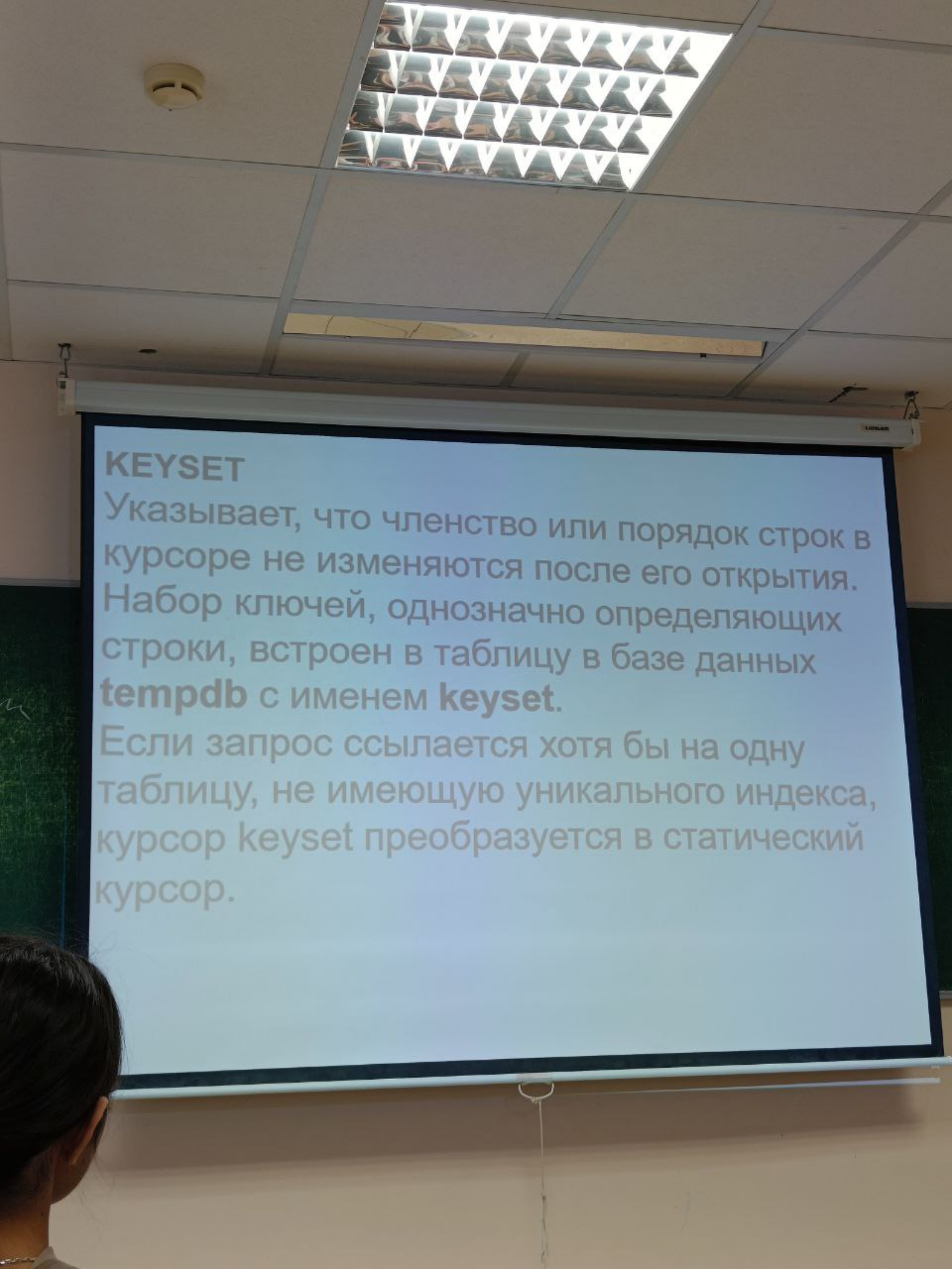

GLOBAL
Указывает, что область курсора является глобальной по отношению к соединению. Имя курсора может использоваться любой хранимой процедурой или пакетом, которые выполняются соединением. Курсор неявно освобождается только в случае разрыва соединения.

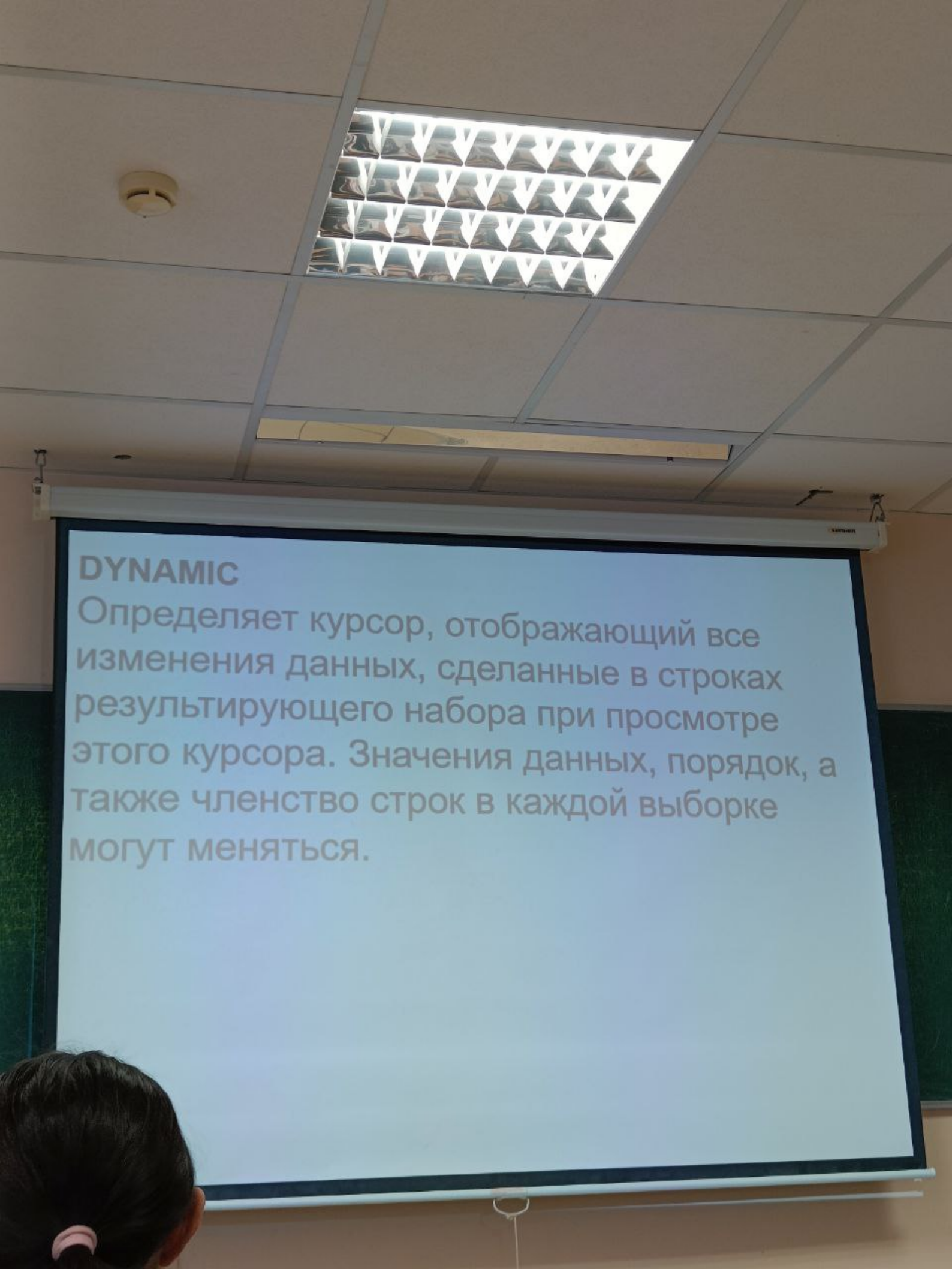

Если не указан ни один из параметров GLOBAL или LOCAL, то значение по умолчанию управляется параметром **default to local cursor** базы данных.

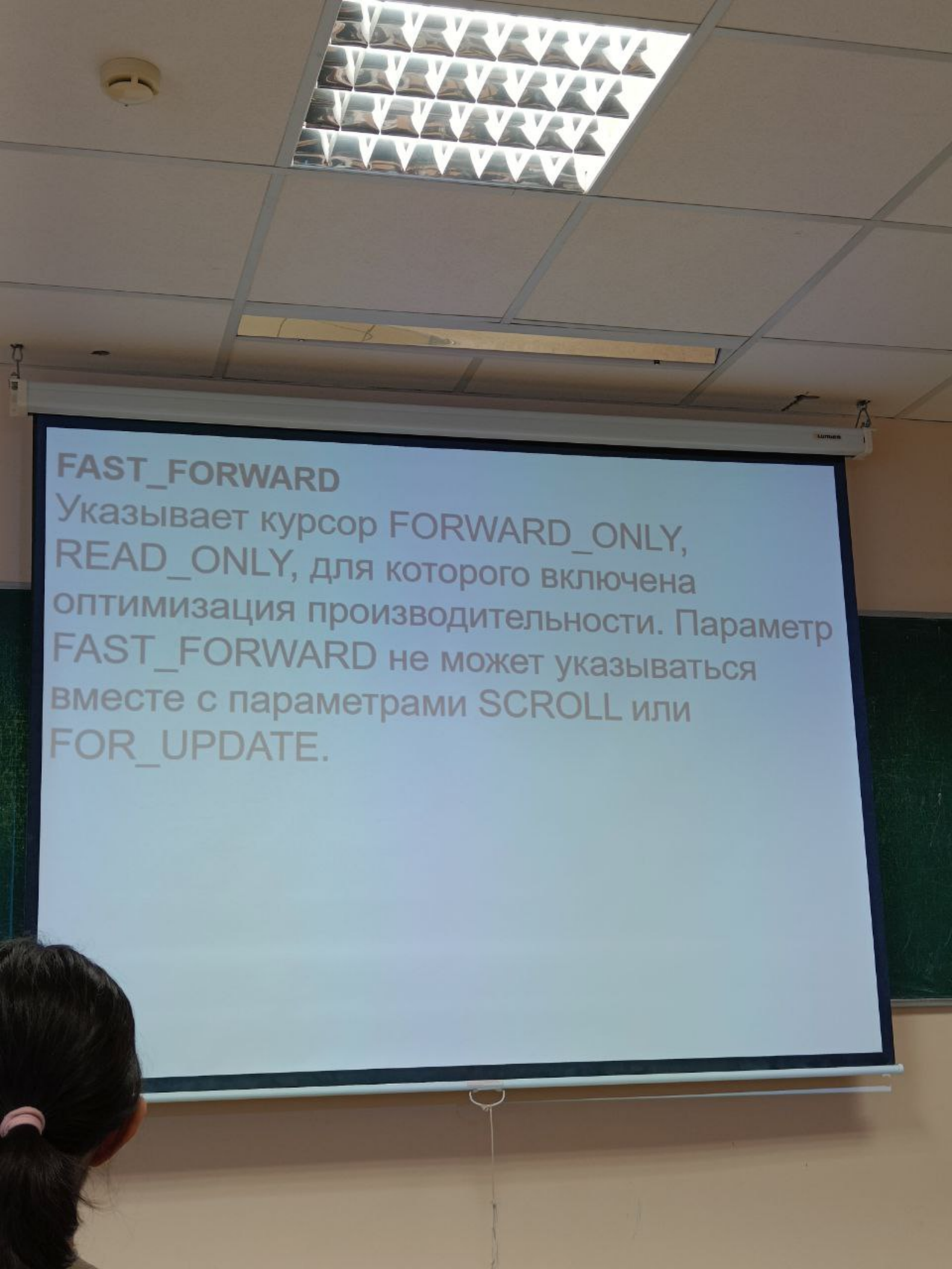

FORWARD_ONLY
Указывает, что курсор может просматриваться только от первой строки к последней. Поддерживается только параметр выборки FETCH NEXT. Если параметр FORWARD_ONLY указан без ключевых слов STATIC, KEYSET или DYNAMIC, то курсор работает как курсор DYNAMIC.

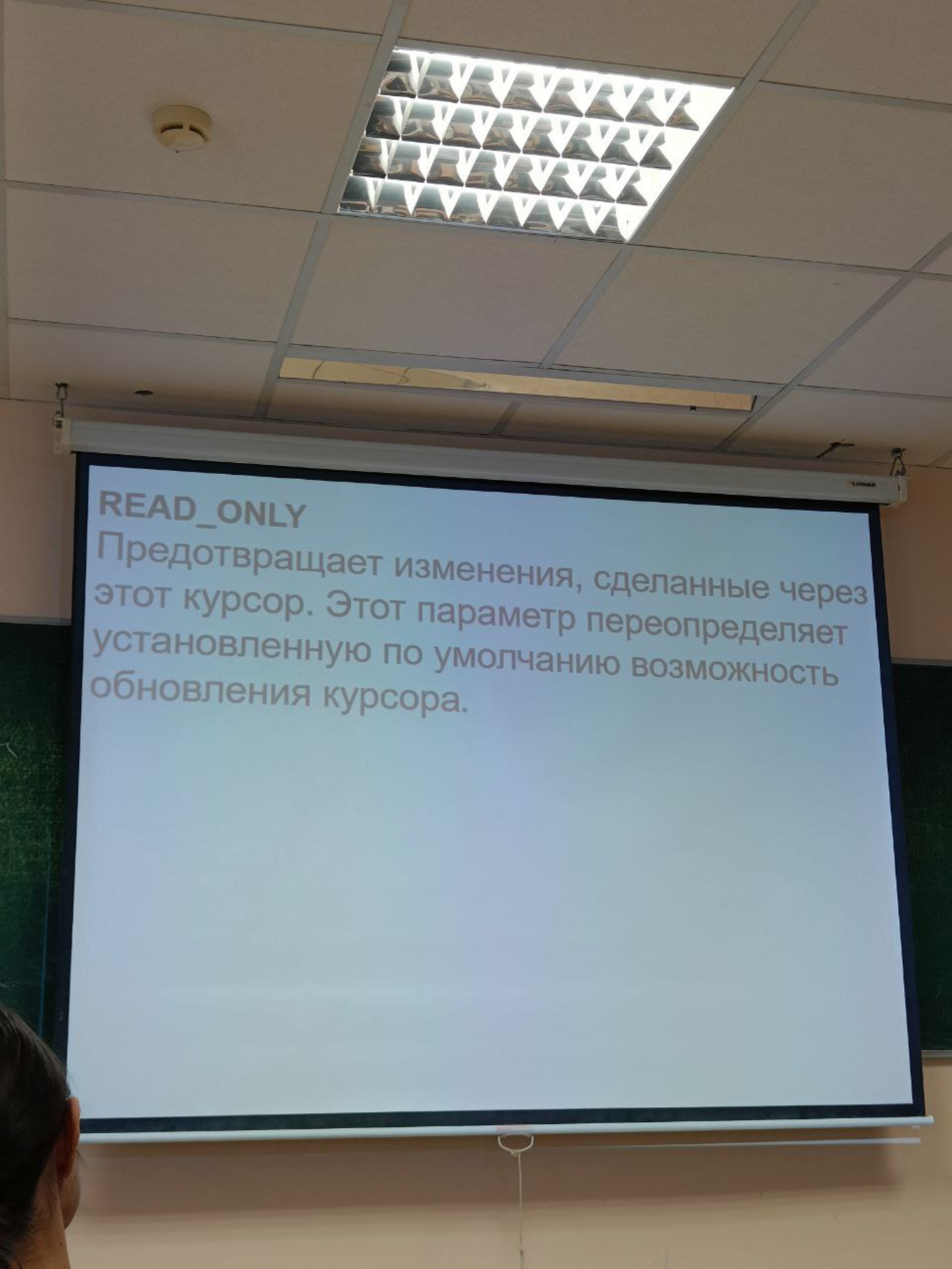

Если не указан ни один из параметров FORWARD_ONLY или SCROLL, а также не указано ни одно из ключевых слов STATIC, KEYSET или DYNAMIC, то по умолчанию задается параметр FORWARD_ONLY. Курсоры STATIC, KEYSET и DYNAMIC имеют значение по умолчанию SCROLL.

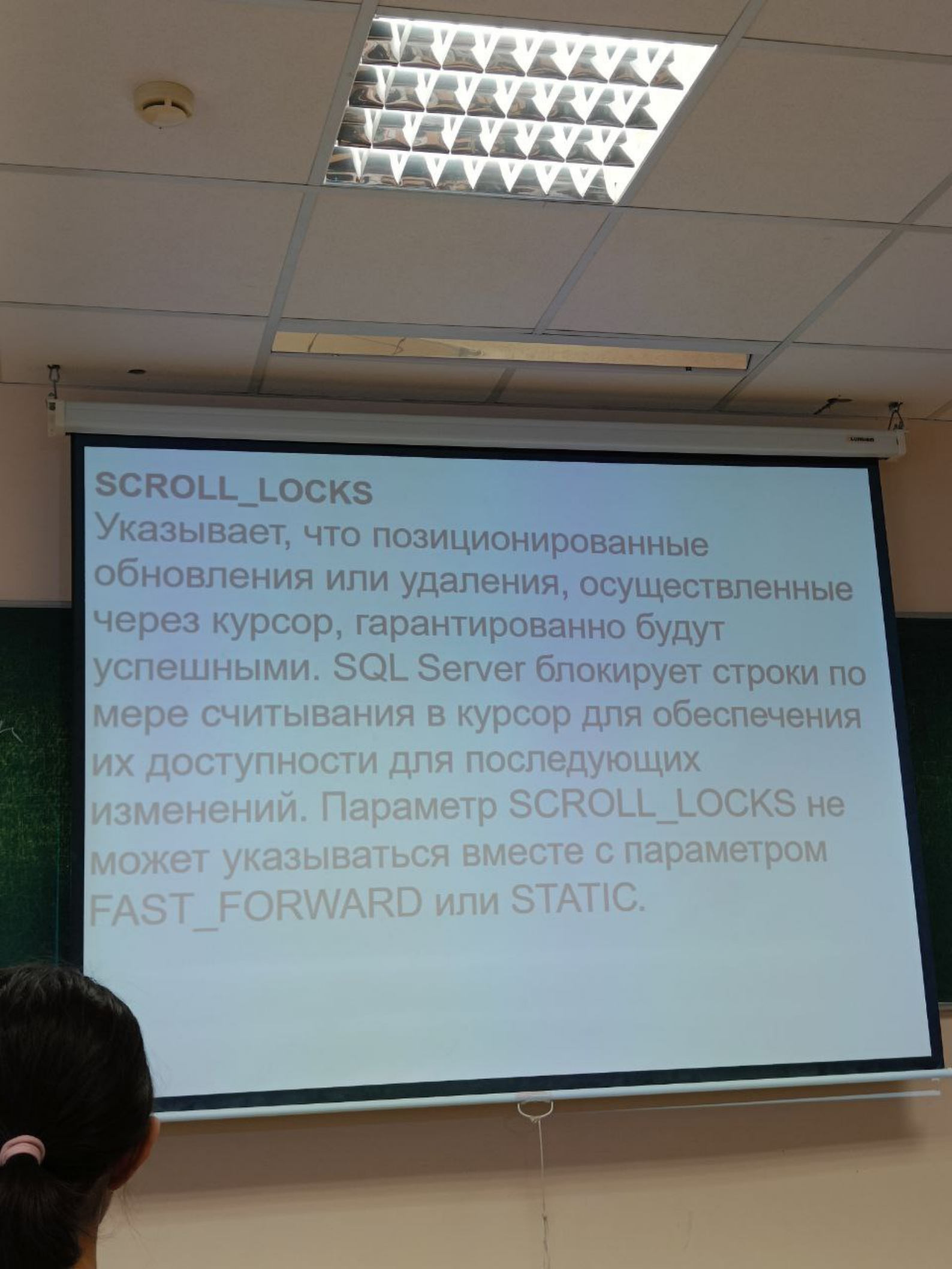

STATIC
Определяет курсор, который создает временную копию данных для использования курсором. Все запросы к курсору обращаются к указанной временной таблице в базе данных **tempdb**, поэтому изменения базовых таблиц не влияют на данные, возвращаемые выборками для данного курсора, а сам курсор не позволяет производить изменения.

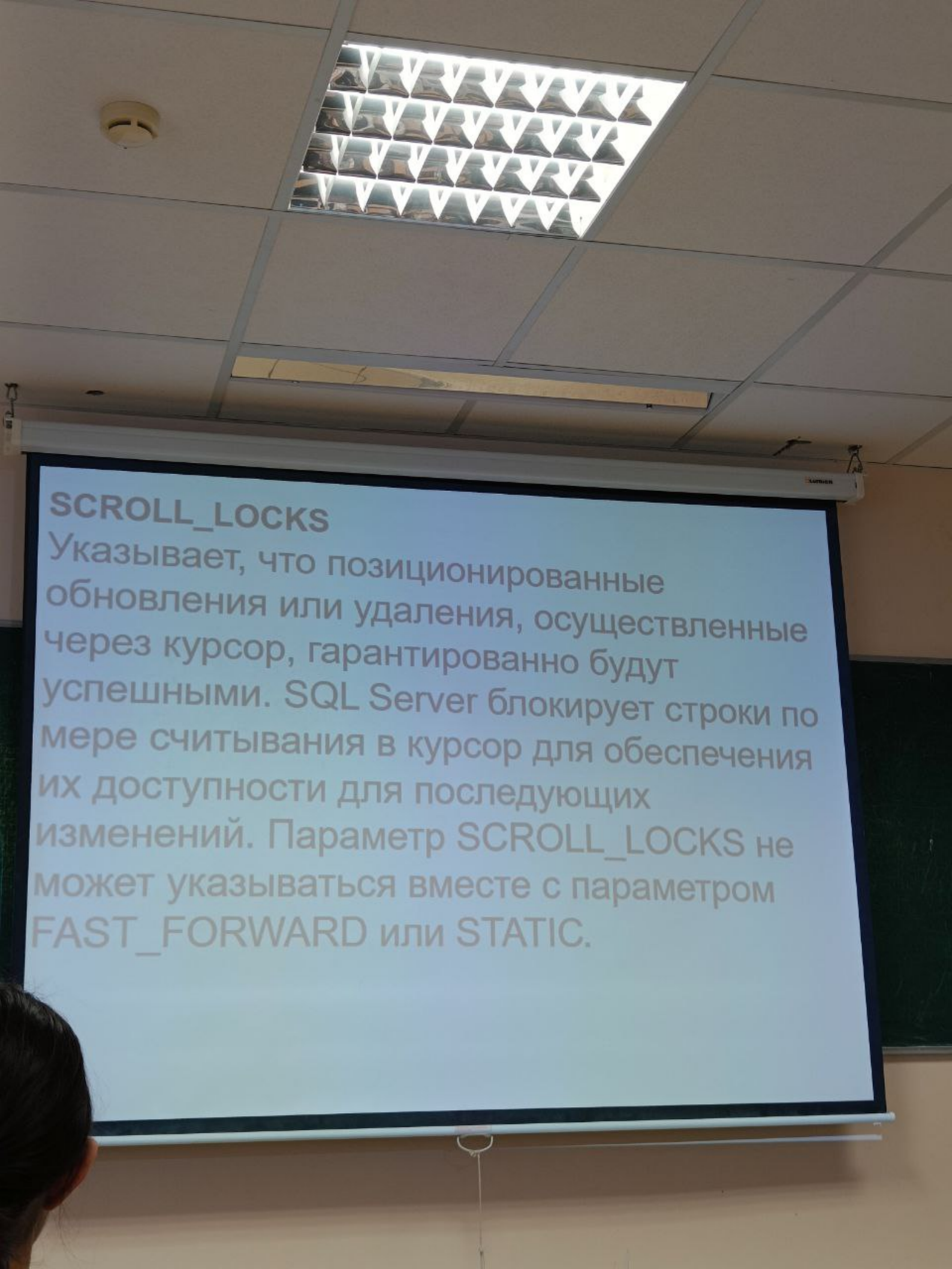

KEYSET
Указывает, что членство или порядок строк в курсоре не изменяются после его открытия. Набор ключей, однозначно определяющих строки, встроен в таблицу в базе данных **tempdb** с именем **keyset**.
Если запрос ссылается хотя бы на одну таблицу, не имеющую уникального индекса, курсор keyset преобразуется в статический курсор.

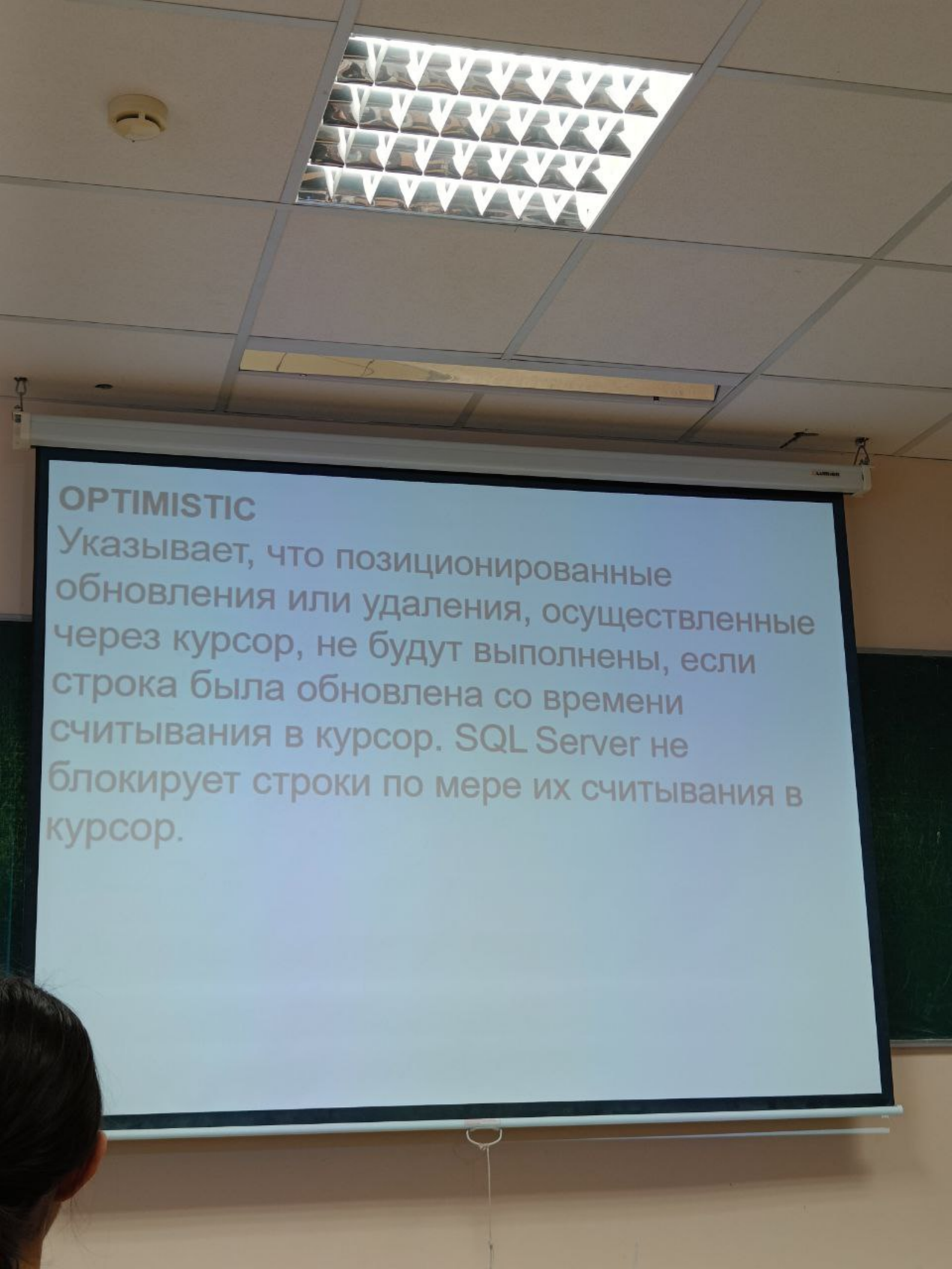

DYNAMIC
Определяет курсор, отображающий все изменения данных, сделанные в строках результирующего набора при просмотре этого курсора. Значения данных, порядок, а также членство строк в каждой выборке могут меняться.

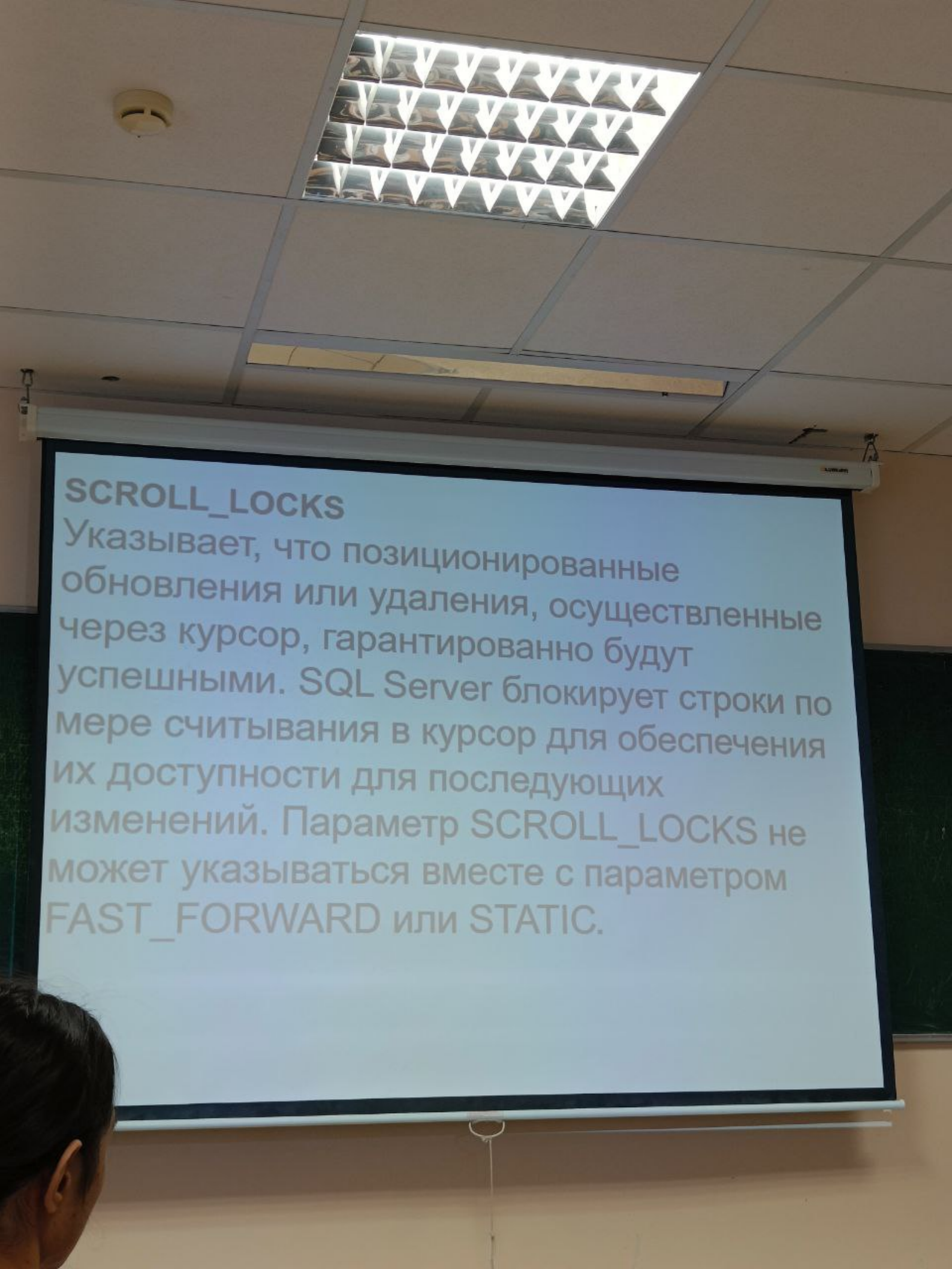

FAST_FORWARD
Указывает курсор FORWARD_ONLY, READ_ONLY, для которого включена оптимизация производительности. Параметр FAST_FORWARD не может указываться вместе с параметрами SCROLL или FOR_UPDATE.

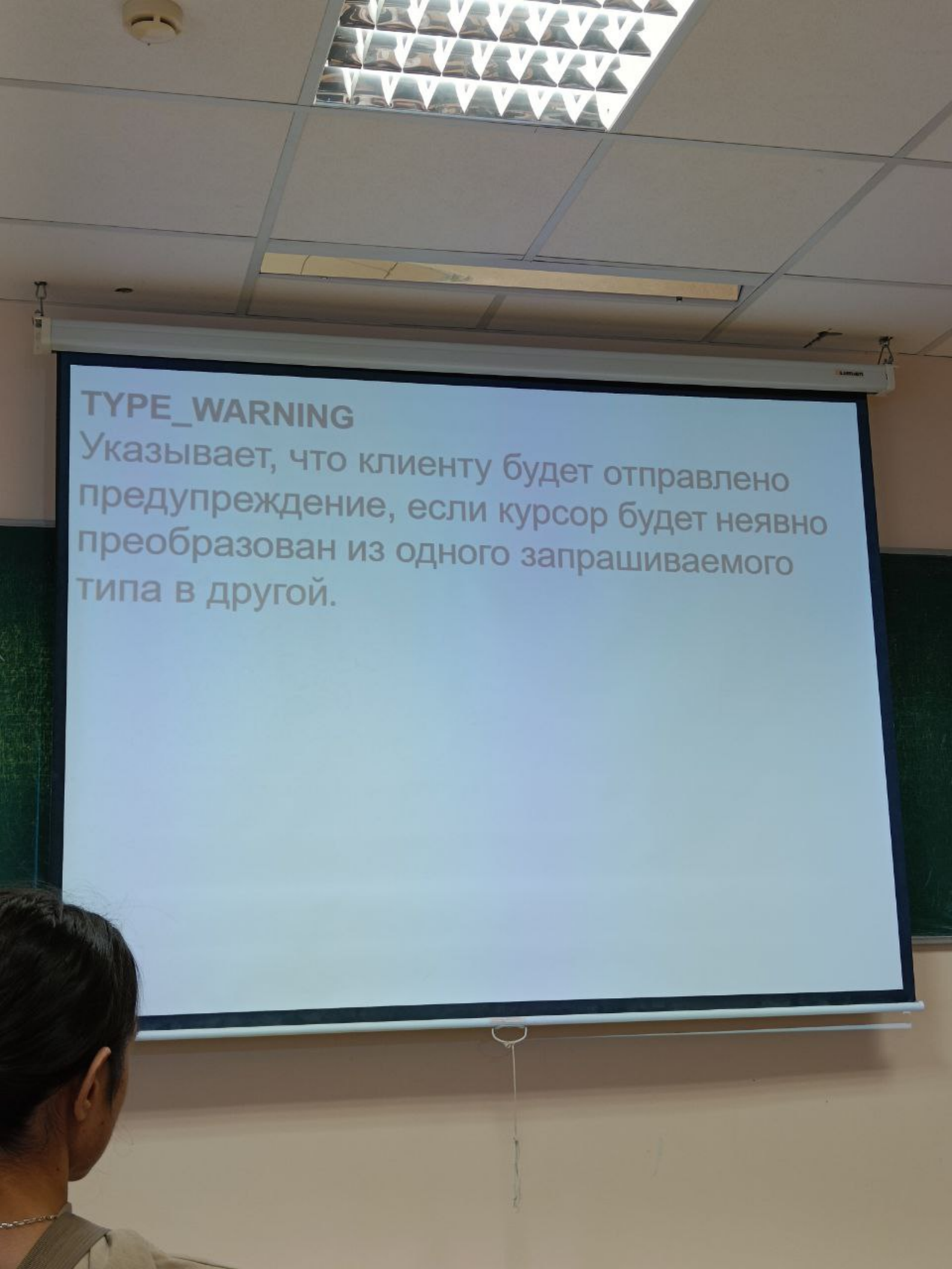

READ_ONLY
Предотвращает изменения, сделанные через этот курсор. Этот параметр переопределяет установленную по умолчанию возможность обновления курсора.

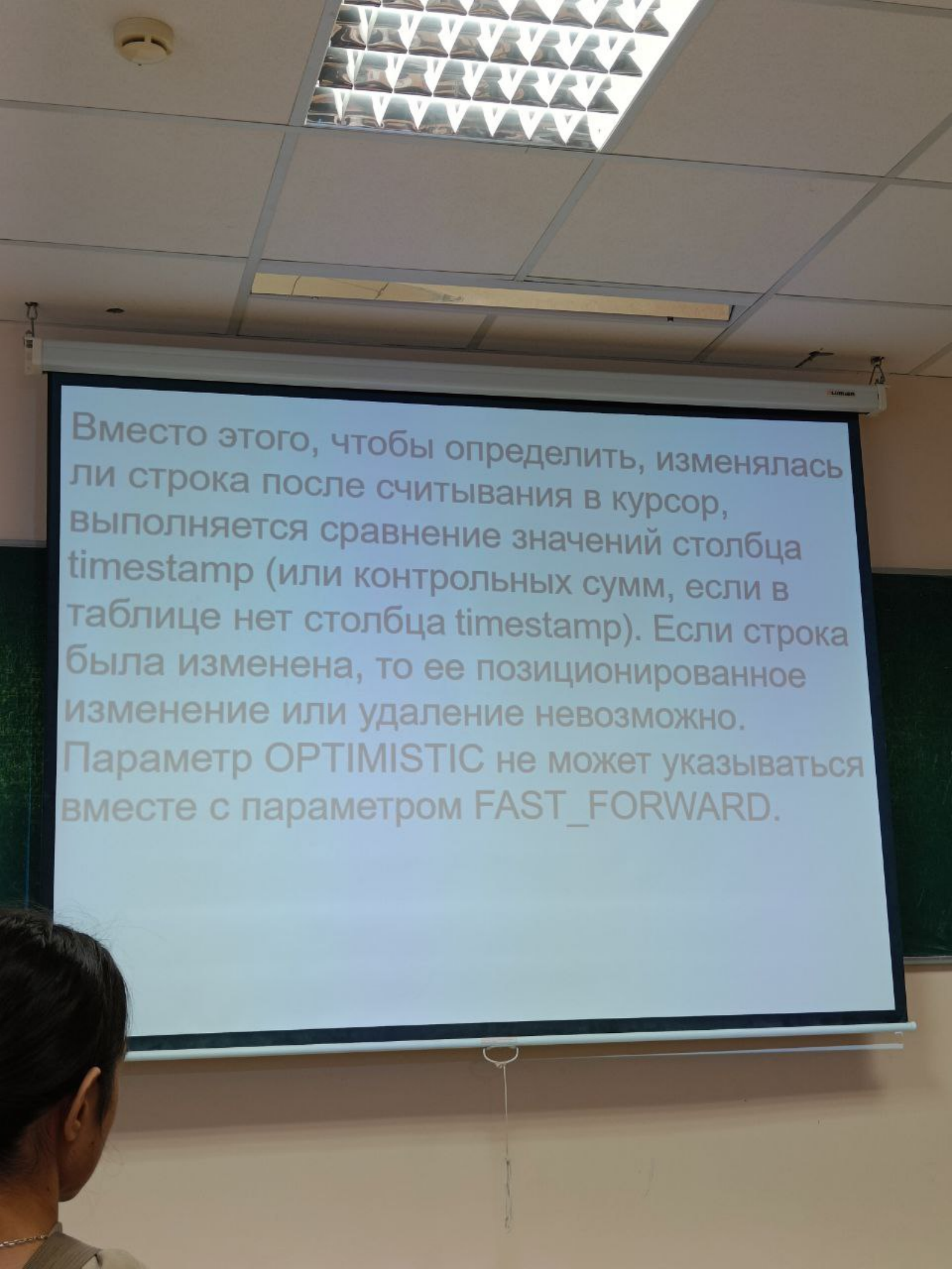

SCROLL_LOCKS
Указывает, что позиционированные обновления или удаления, осуществленные через курсор, гарантированно будут успешными. SQL Server блокирует строки по мере считывания в курсор для обеспечения их доступности для последующих изменений. Параметр SCROLL_LOCKS не может указываться вместе с параметром FAST_FORWARD или STATIC.

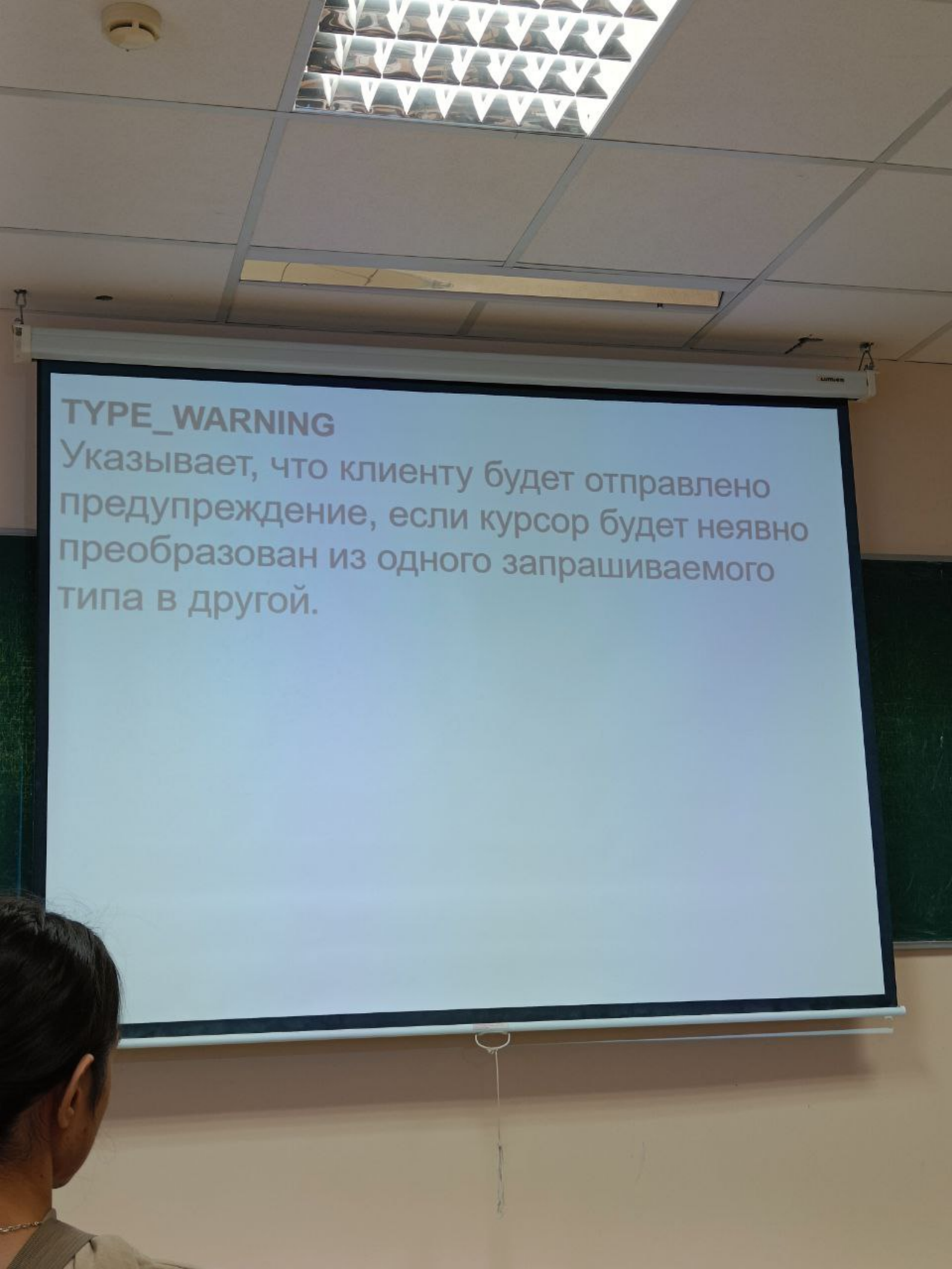

SCROLL_LOCKS
Указывает, что позиционированные обновления или удаления, осуществленные через курсор, гарантированно будут успешными. SQL Server блокирует строки по мере считывания в курсор для обеспечения их доступности для последующих изменений. Параметр SCROLL_LOCKS не может указываться вместе с параметром FAST_FORWARD или STATIC.

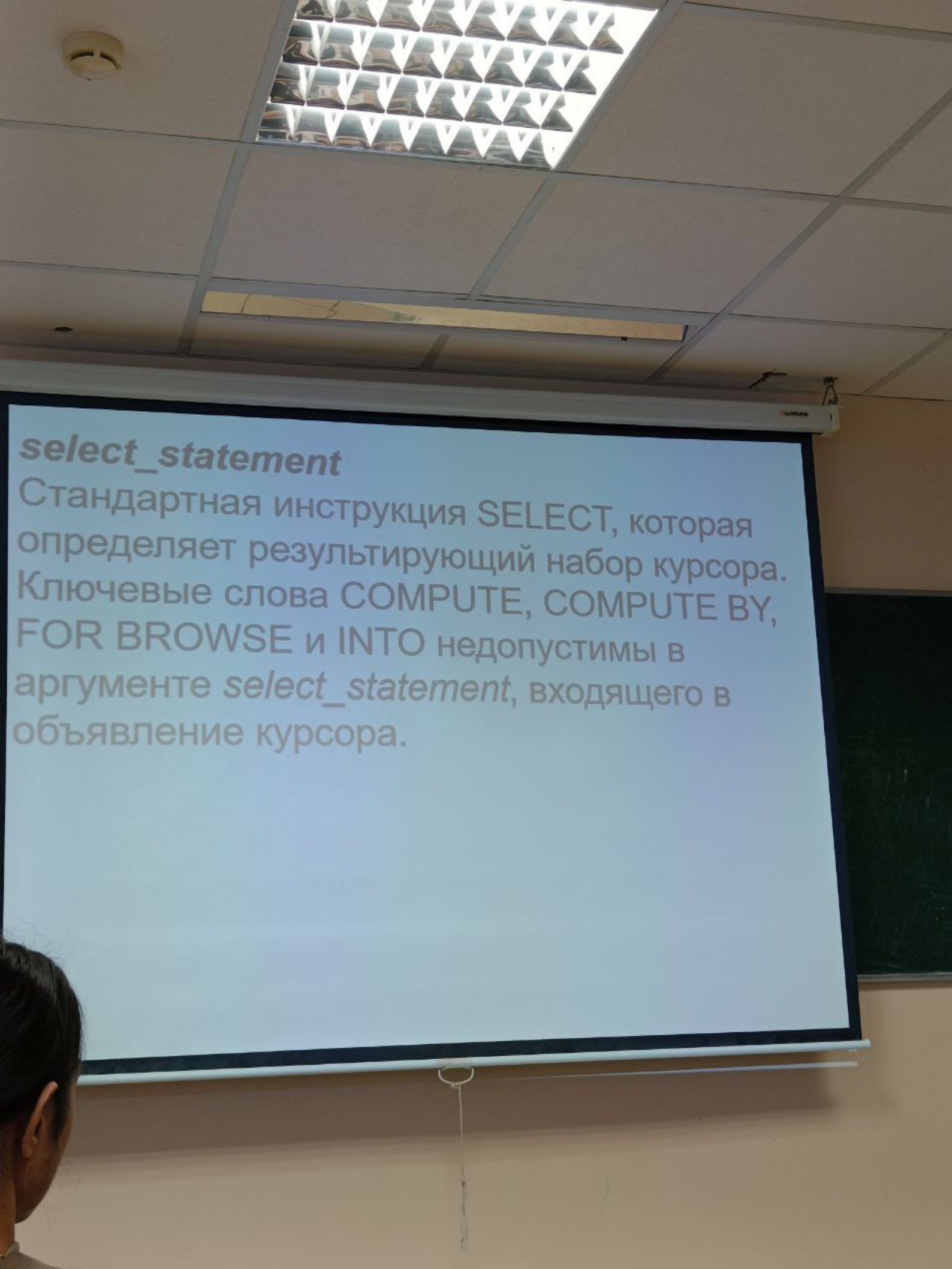

OPTIMISTIC
Указывает, что позиционированные обновления или удаления, осуществленные через курсор, не будут выполнены, если строка была обновлена со времени считывания в курсор. SQL Server не блокирует строки по мере их считывания в курсор.

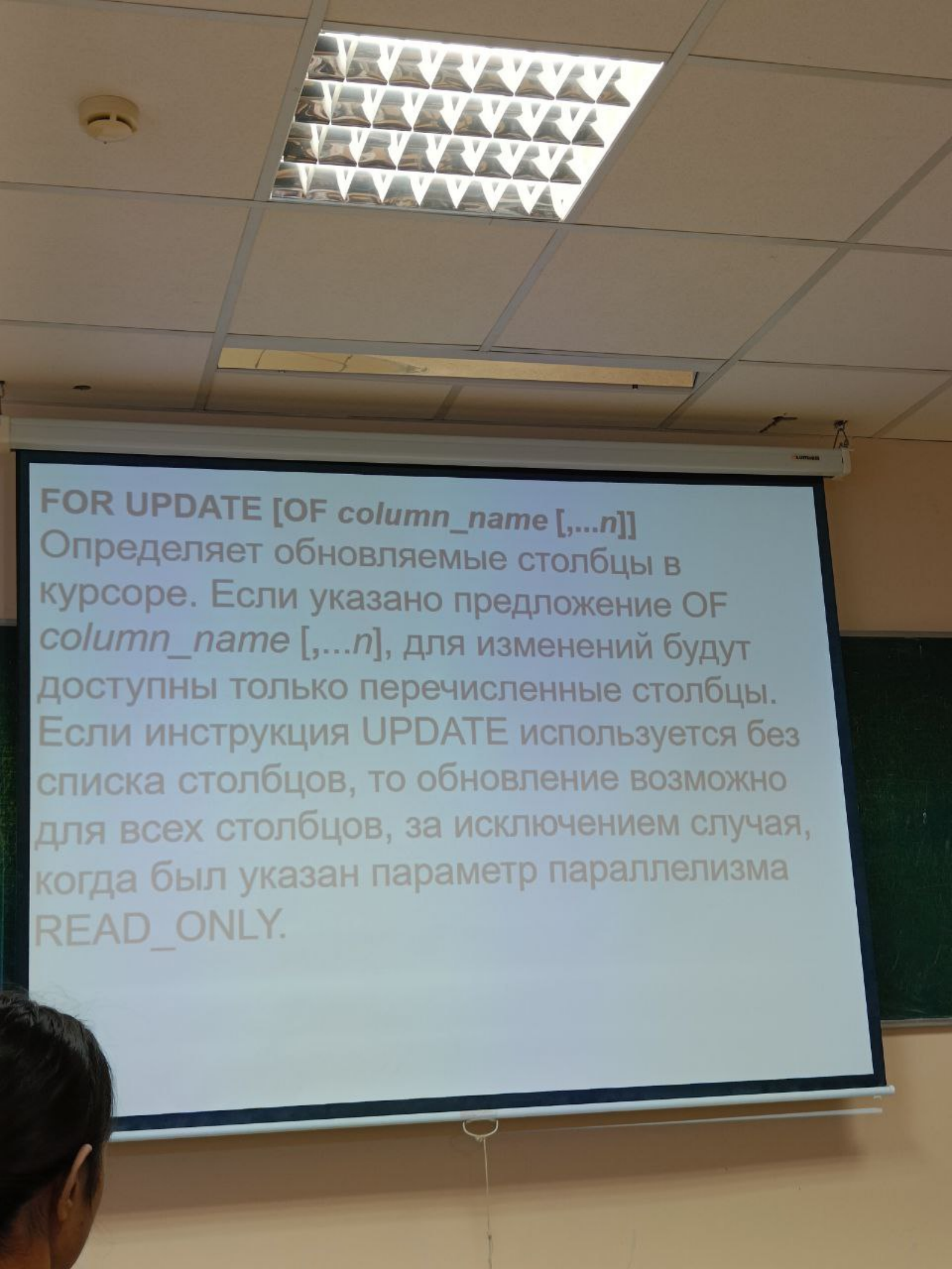

SCROLL_LOCKS
Указывает, что позиционированные обновления или удаления, осуществленные через курсор, гарантированно будут успешными. SQL Server блокирует строки по мере считывания в курсор для обеспечения их доступности для последующих изменений. Параметр SCROLL_LOCKS не может указываться вместе с параметром FAST_FORWARD или STATIC.

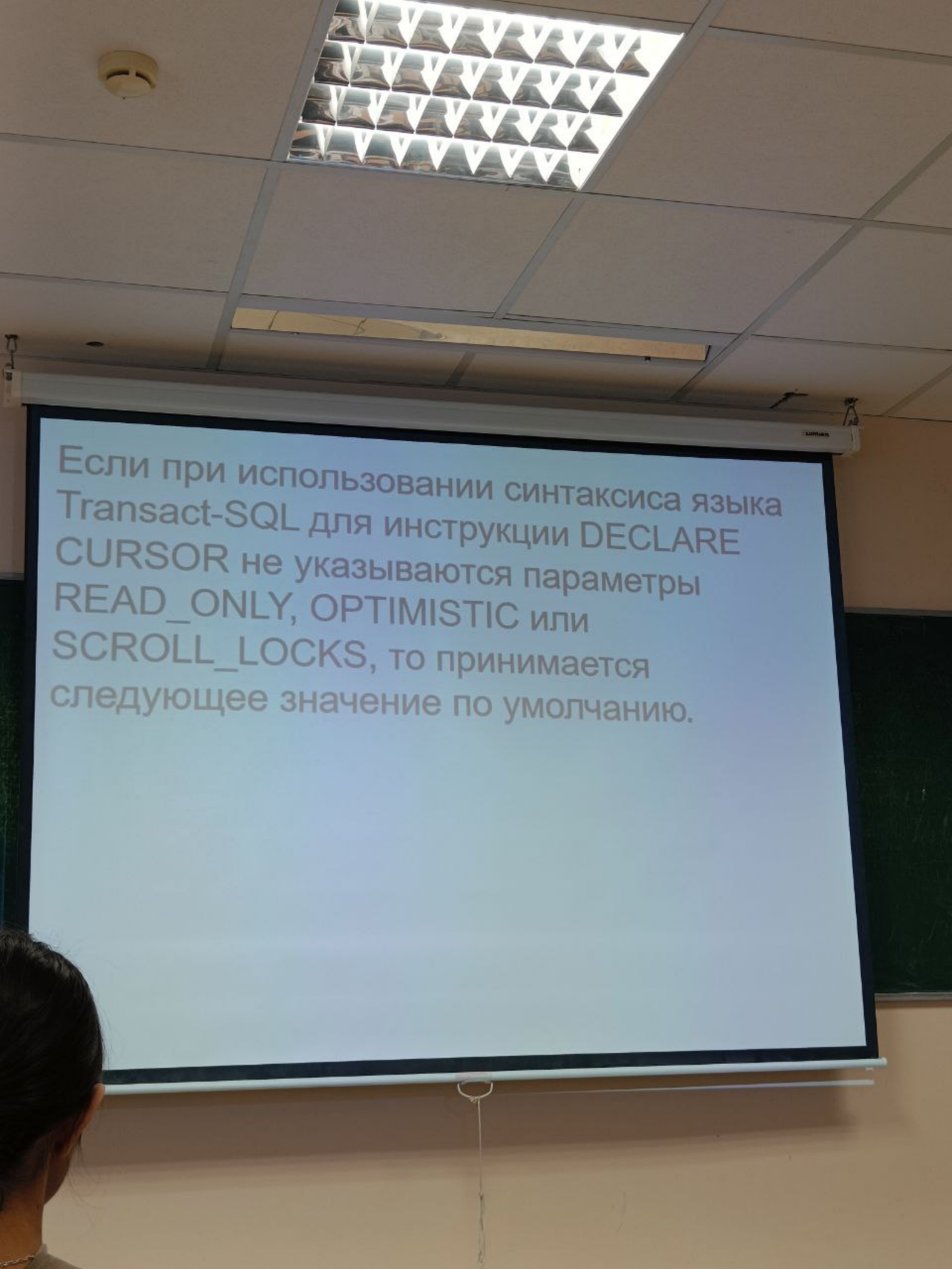

TYPE_WARNING
Указывает, что клиенту будет отправлено предупреждение, если курсор будет неявно преобразован из одного запрашиваемого типа в другой.

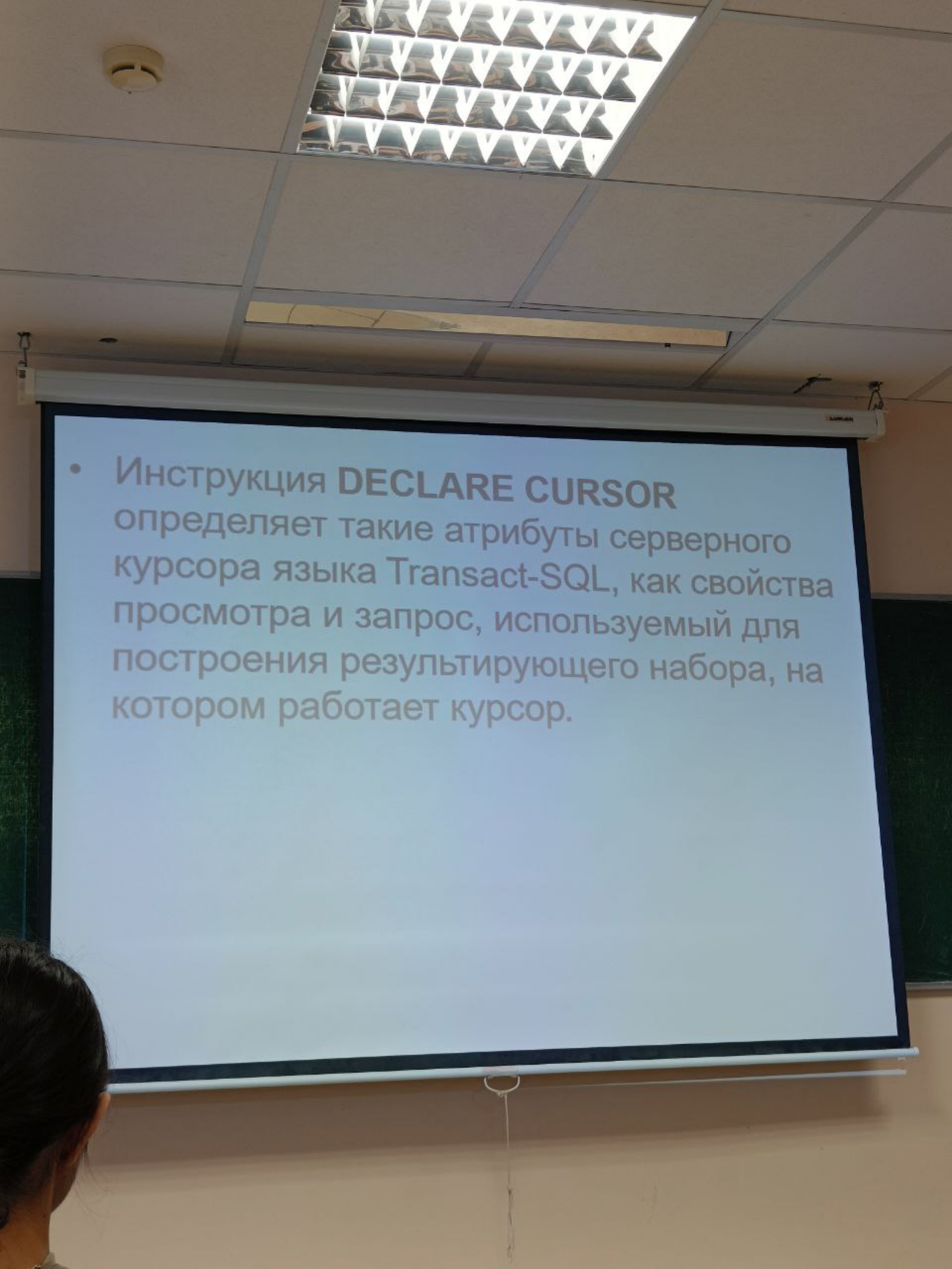

Вместо этого, чтобы определить, изменялась ли строка после считывания в курсор, выполняется сравнение значений столбца timestamp (или контрольных сумм, если в таблице нет столбца timestamp). Если строка была изменена, то ее позиционированное изменение или удаление невозможно. Параметр OPTIMISTIC не может указываться вместе с параметром FAST_FORWARD.

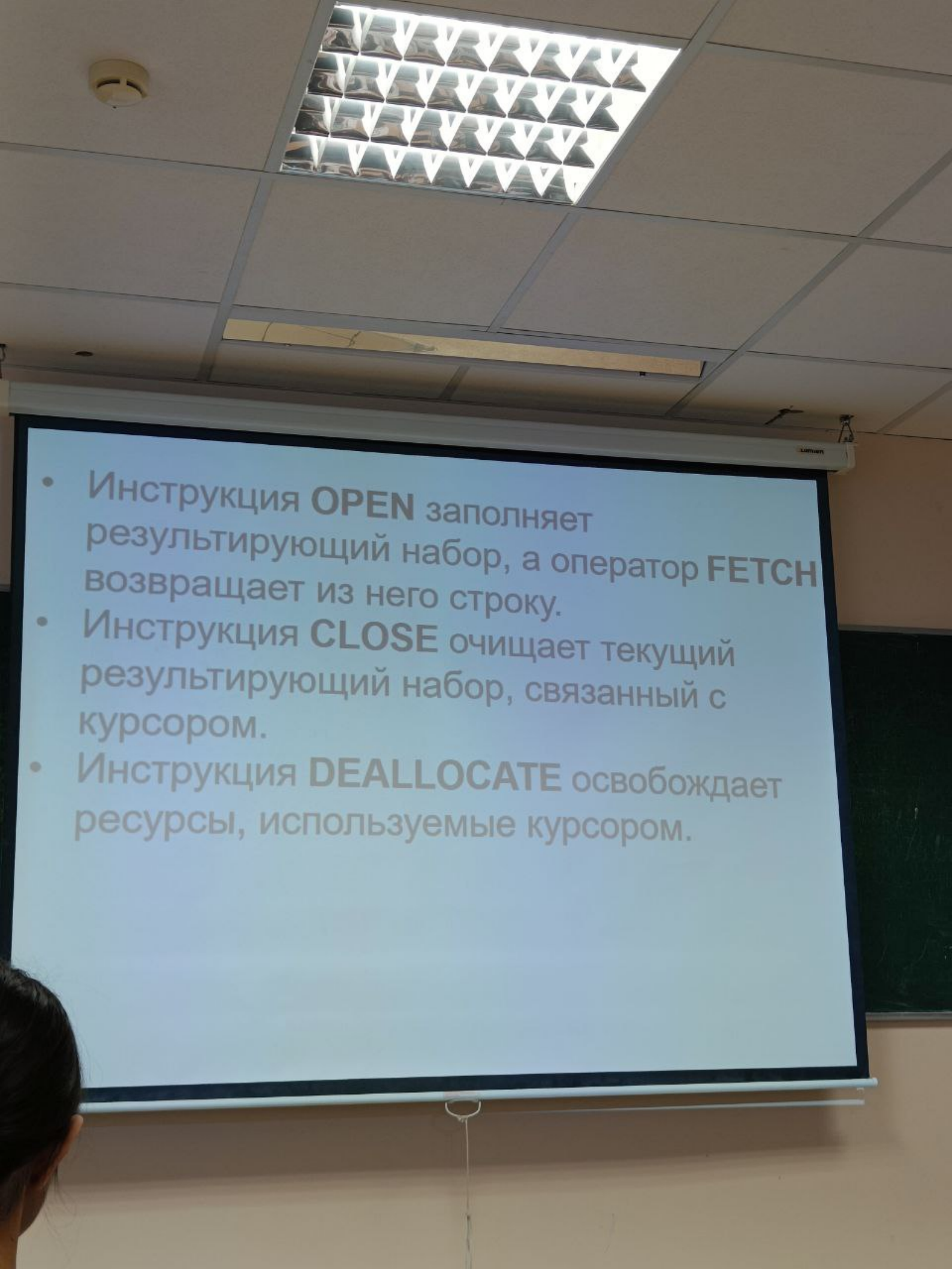

TYPE_WARNING
Указывает, что клиенту будет отправлено предупреждение, если курсор будет неявно преобразован из одного запрашиваемого типа в другой.

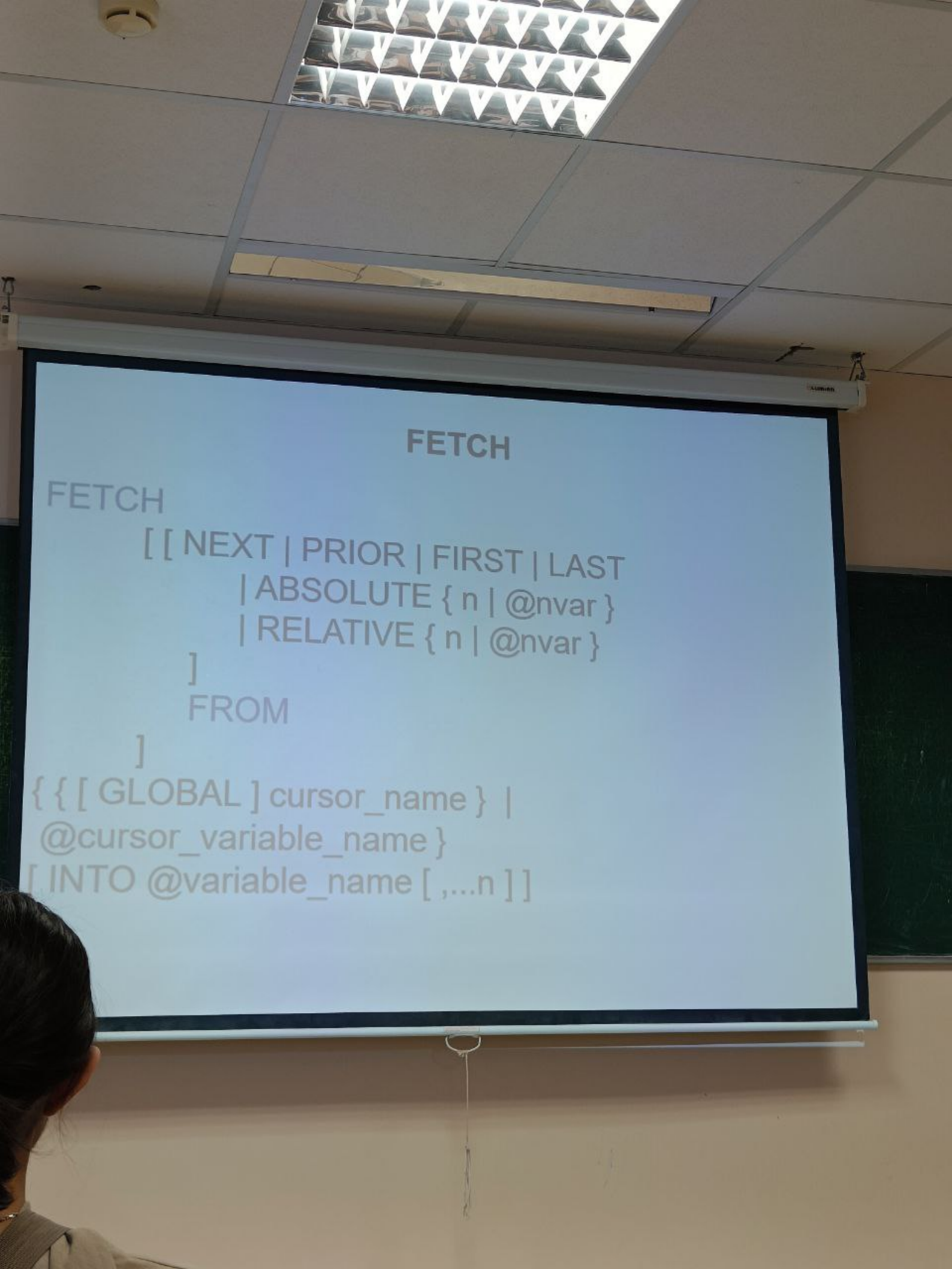

select_statement
Стандартная инструкция SELECT, которая определяет результирующий набор курсора. Ключевые слова COMPUTE, COMPUTE BY, FOR BROWSE и INTO недопустимы в аргументе *select_statement*, входящего в объявление курсора.

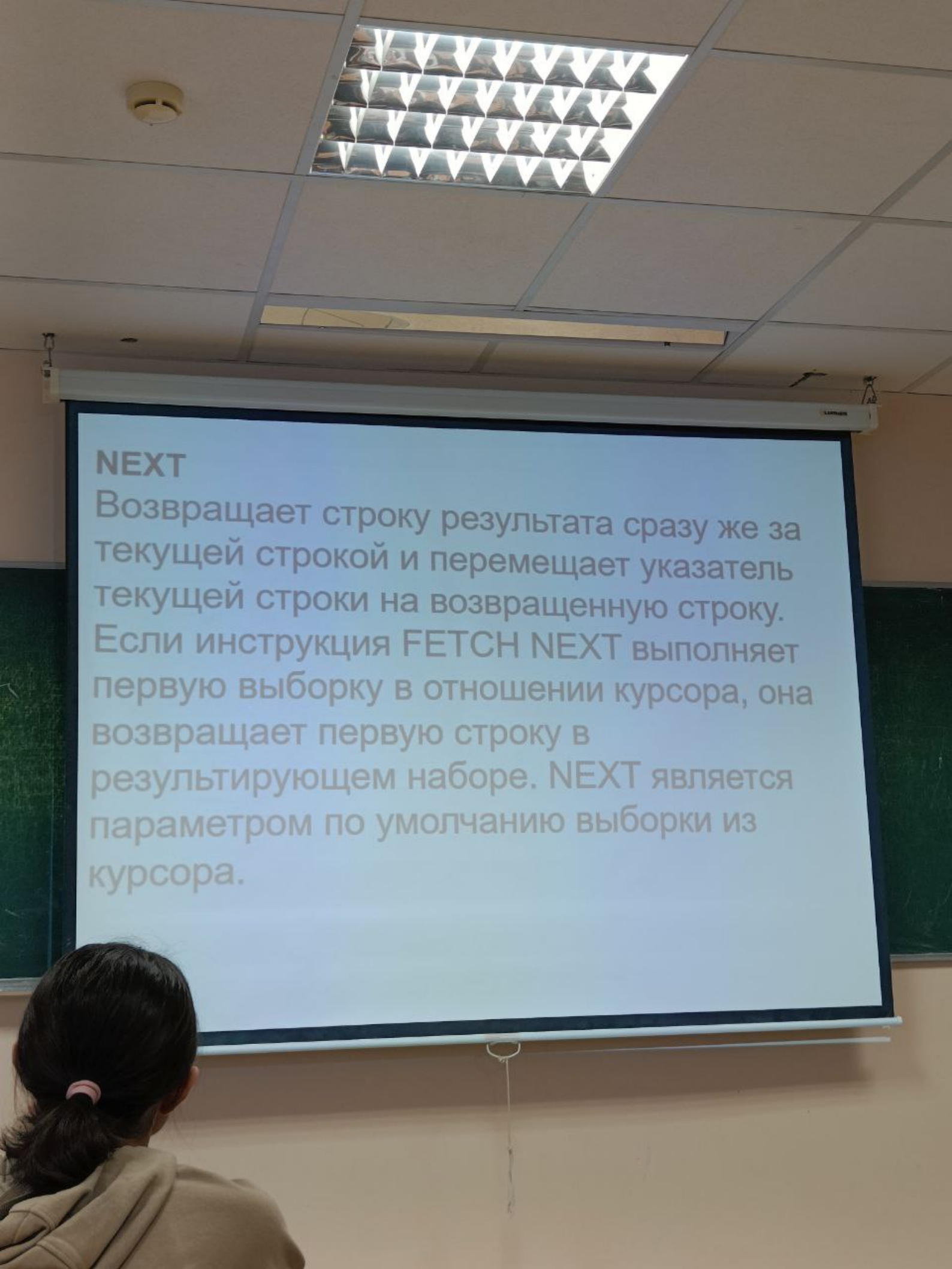

FOR UPDATE [OF *column_name* [,...*n*]]
Определяет обновляемые столбцы в курсоре. Если указано предложение OF *column_name* [,...*n*], для изменений будут доступны только перечисленные столбцы. Если инструкция UPDATE используется без списка столбцов, то обновление возможно для всех столбцов, за исключением случая, когда был указан параметр параллелизма READ_ONLY.

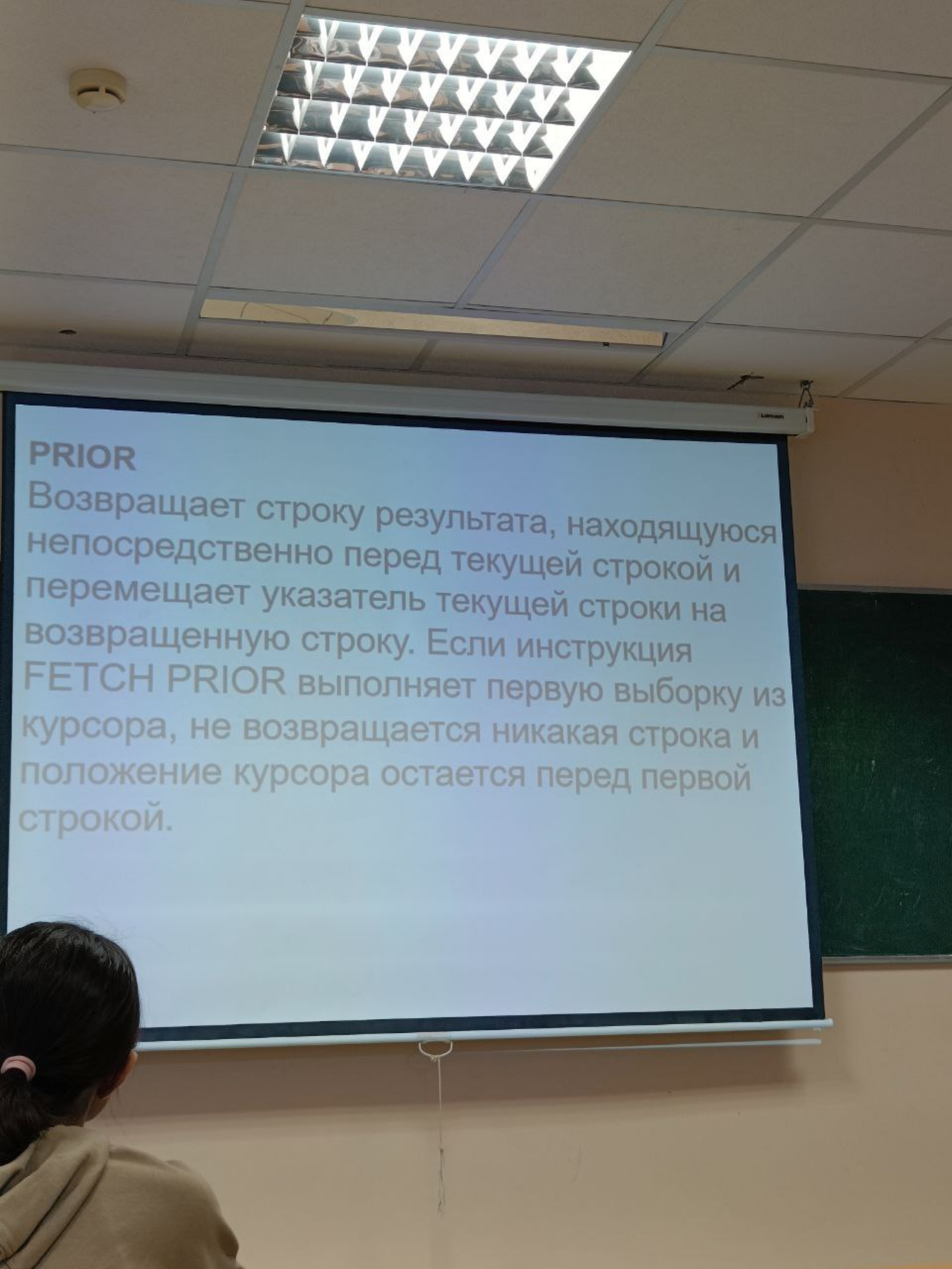

Если при использовании синтаксиса языка Transact-SQL для инструкции DECLARE CURSOR не указываются параметры READ_ONLY, OPTIMISTIC или SCROLL_LOCKS, то принимается следующее значение по умолчанию.

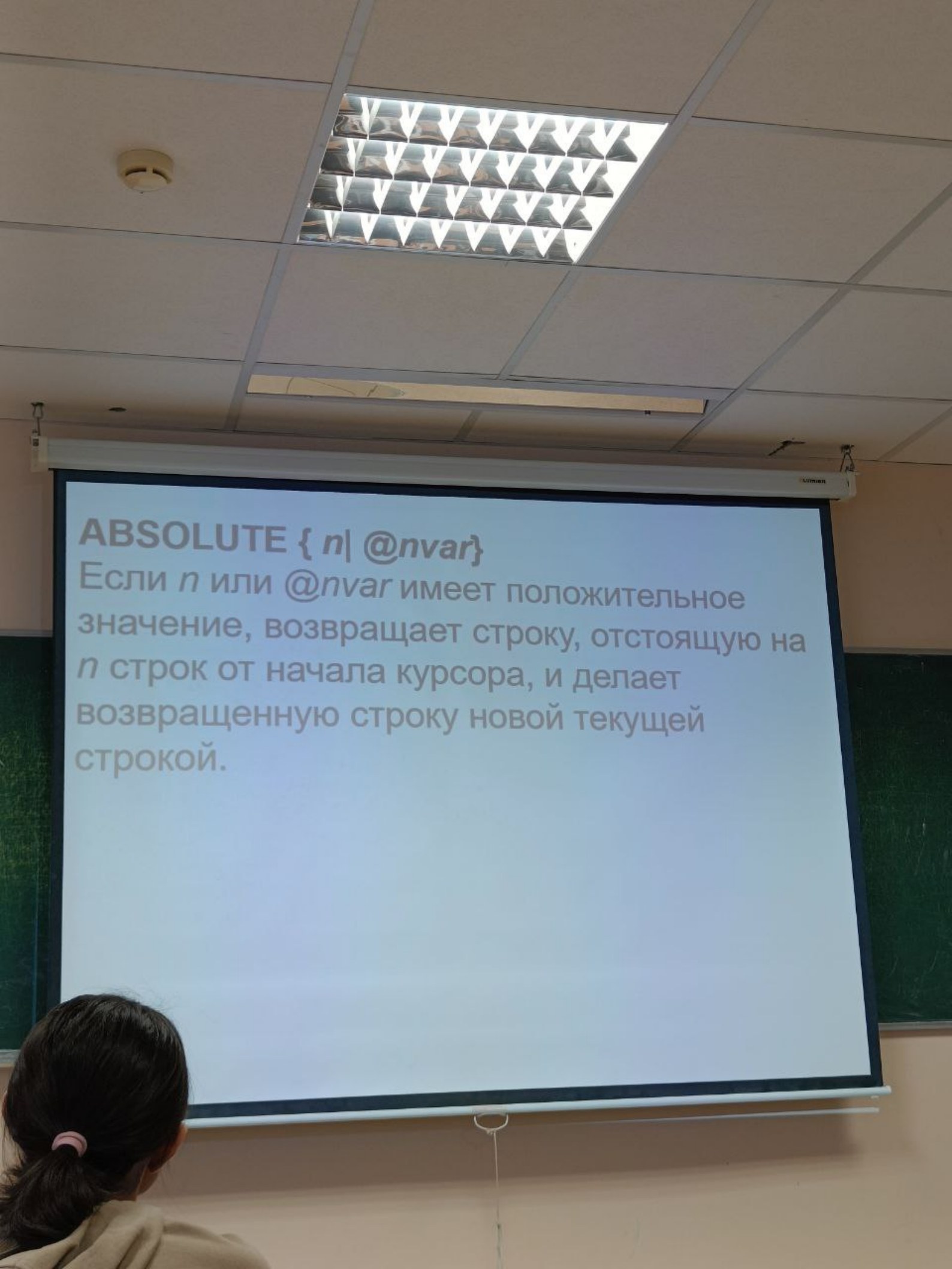

• Инструкция **DECLARE CURSOR** определяет такие атрибуты серверного курсора языка Transact-SQL, как свойства просмотра и запрос, используемый для построения результирующего набора, на котором работает курсор.

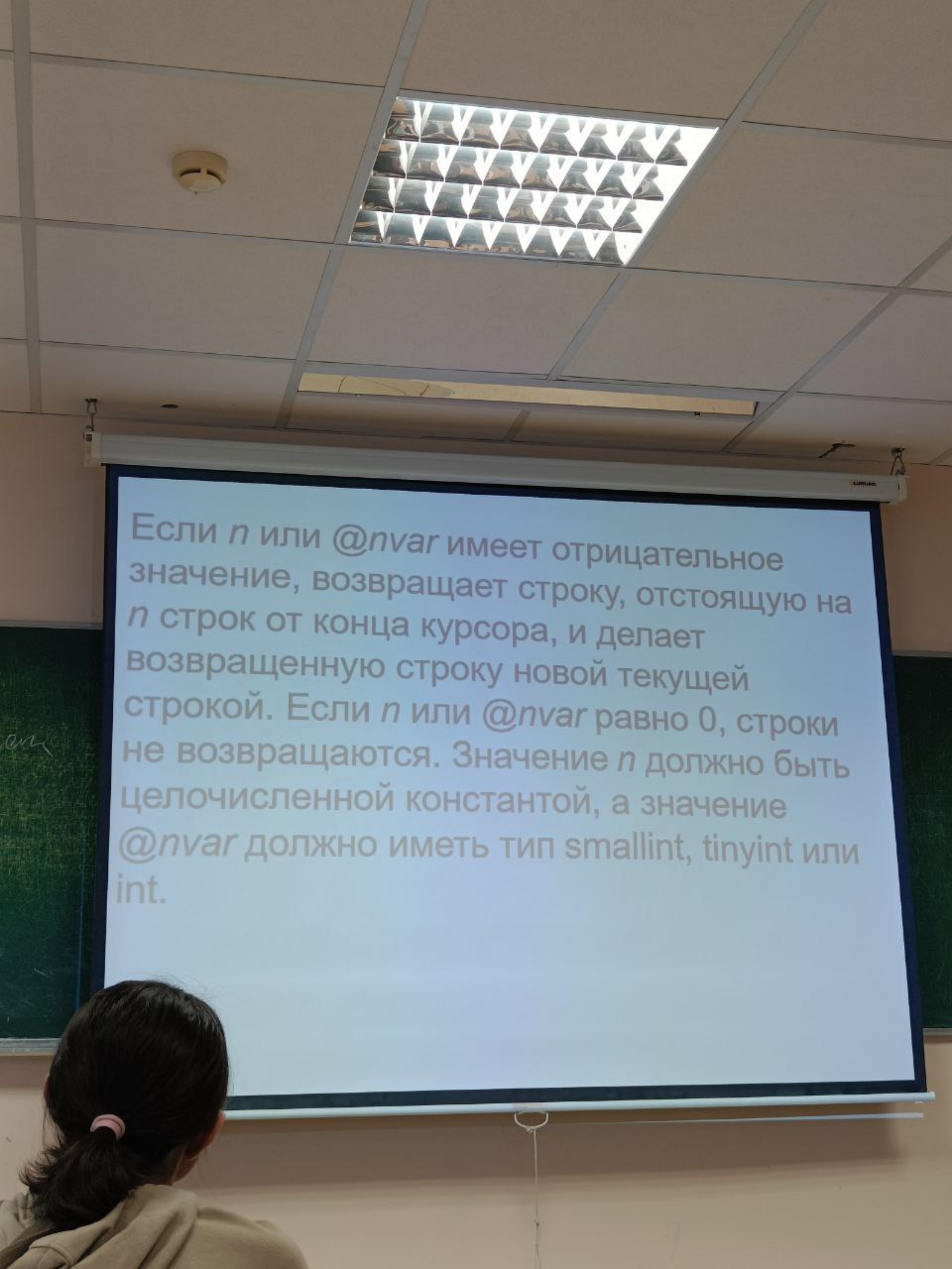

• Инструкция **OPEN** заполняет результирующий набор, а оператор **FETCH** возвращает из него строку.
• Инструкция **CLOSE** очищает текущий результирующий набор, связанный с курсором.
• Инструкция **DEALLOCATE** освобождает ресурсы, используемые курсором.

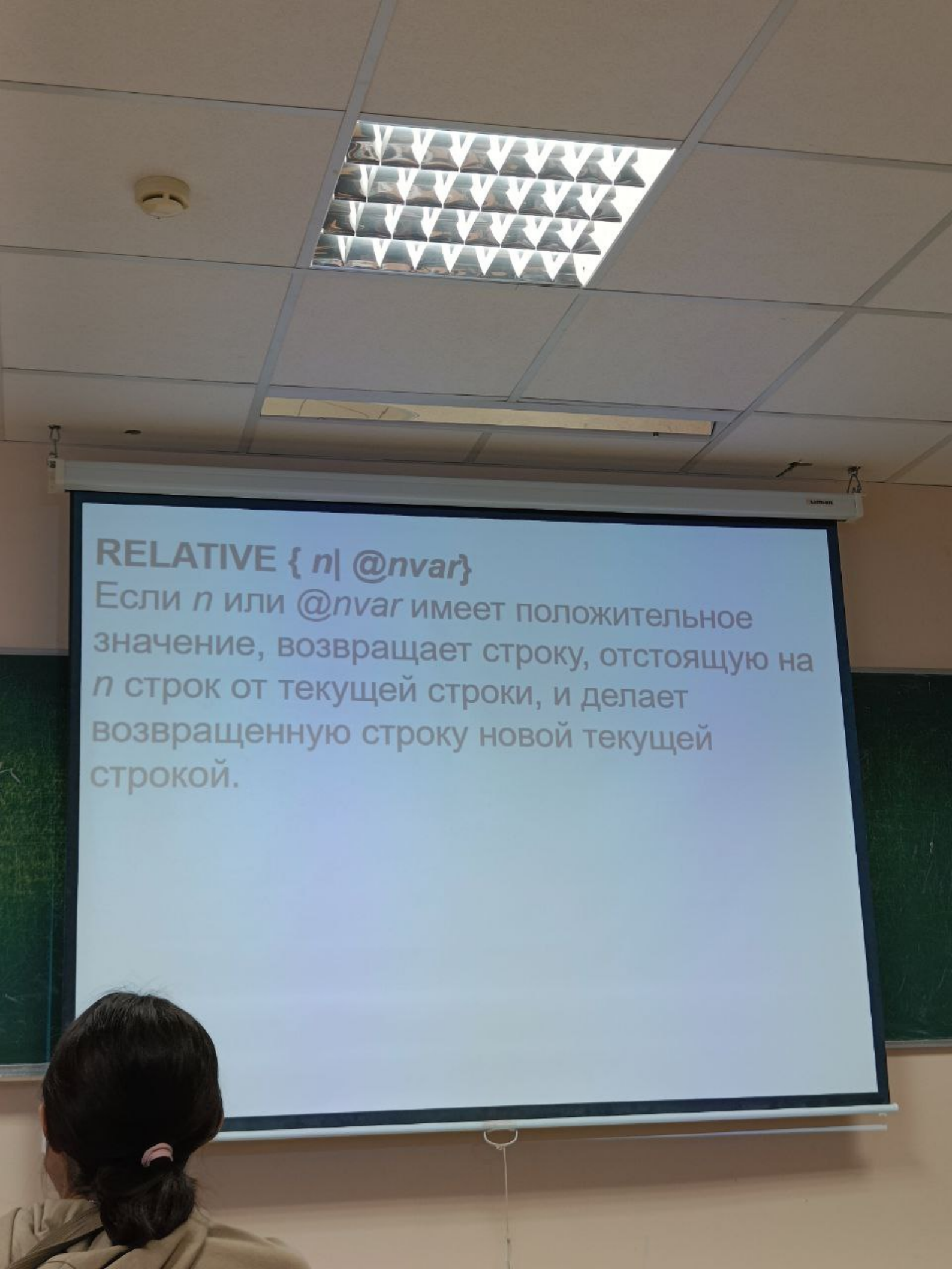

FETCH
FETCH
[[NEXT | PRIOR | FIRST | LAST
| ABSOLUTE { n | @nvar }
| RELATIVE { n | @nvar }
]
FROM
]
{ { [GLOBAL] cursor_name } |
@cursor_variable_name }
[INTO @variable_name [,...n]]

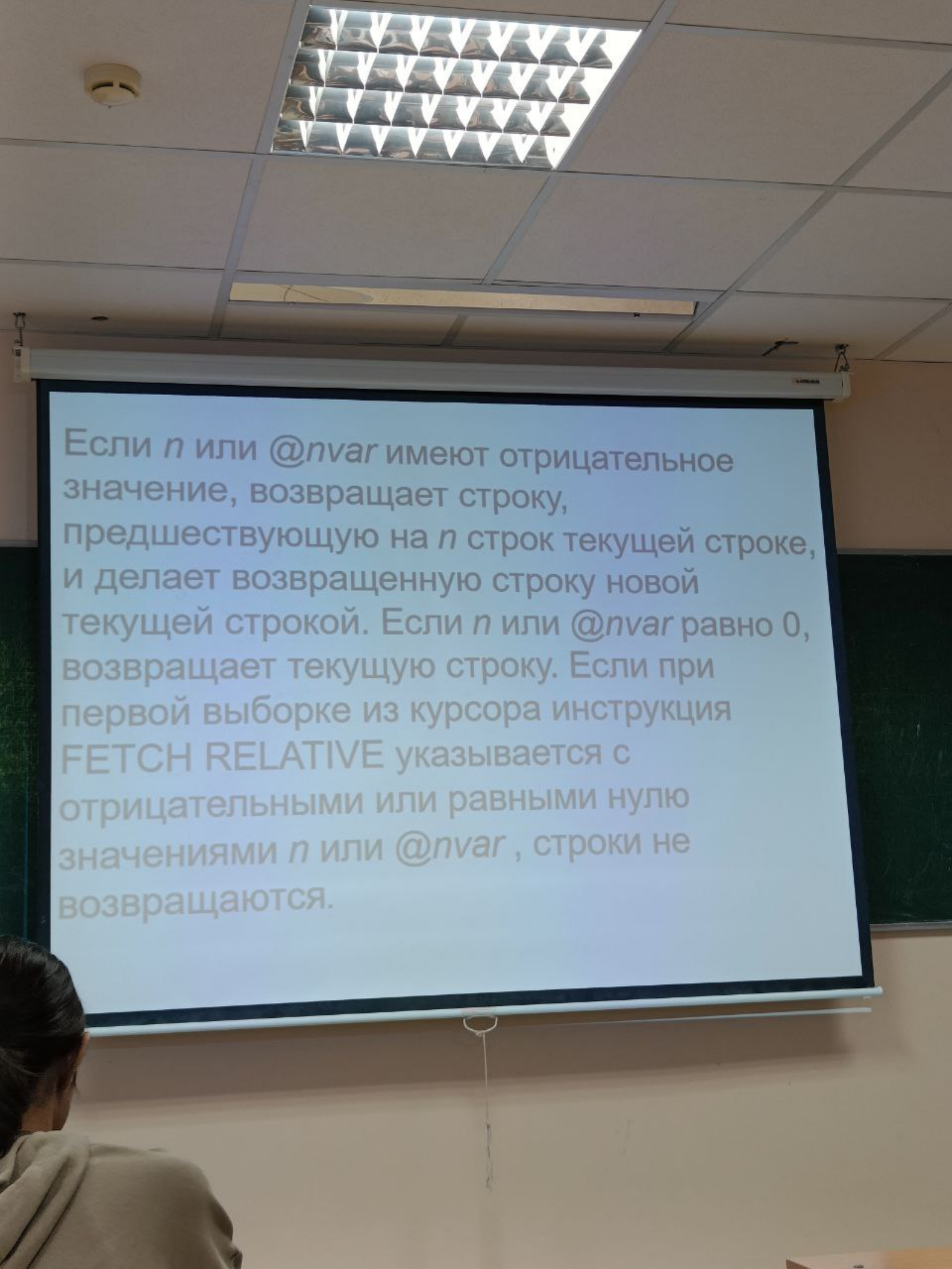

NEXT
Возвращает строку результата сразу же за текущей строкой и перемещает указатель текущей строки на возвращенную строку. Если инструкция FETCH NEXT выполняет первую выборку в отношении курсора, она возвращает первую строку в результирующем наборе. NEXT является параметром по умолчанию выборки из курсора.

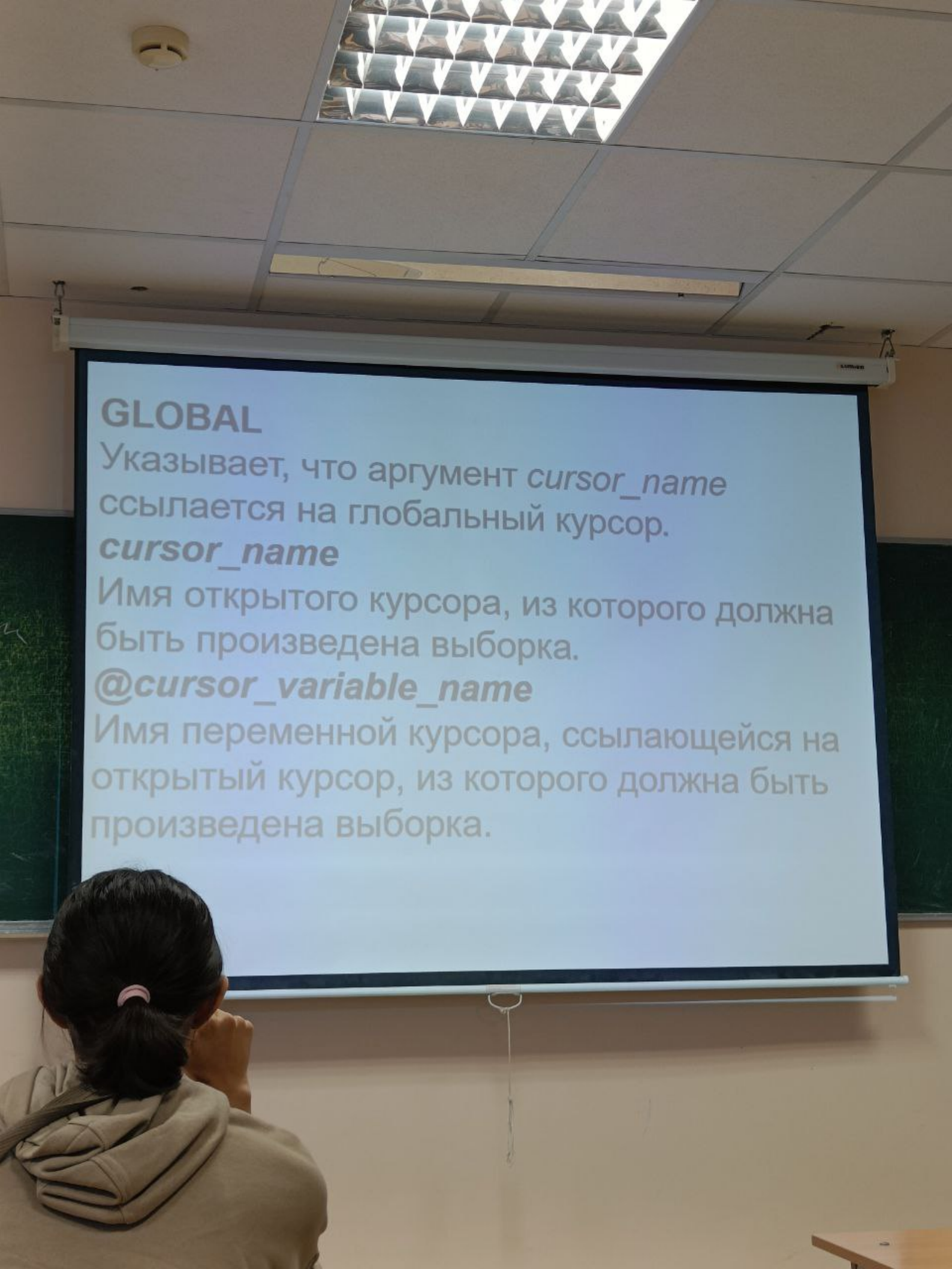

PRIOR
Возвращает строку результата, находящуюся непосредственно перед текущей строкой и перемещает указатель текущей строки на возвращенную строку. Если инструкция FETCH PRIOR выполняет первую выборку из курсора, не возвращается никакая строка и положение курсора остается перед первой строкой.

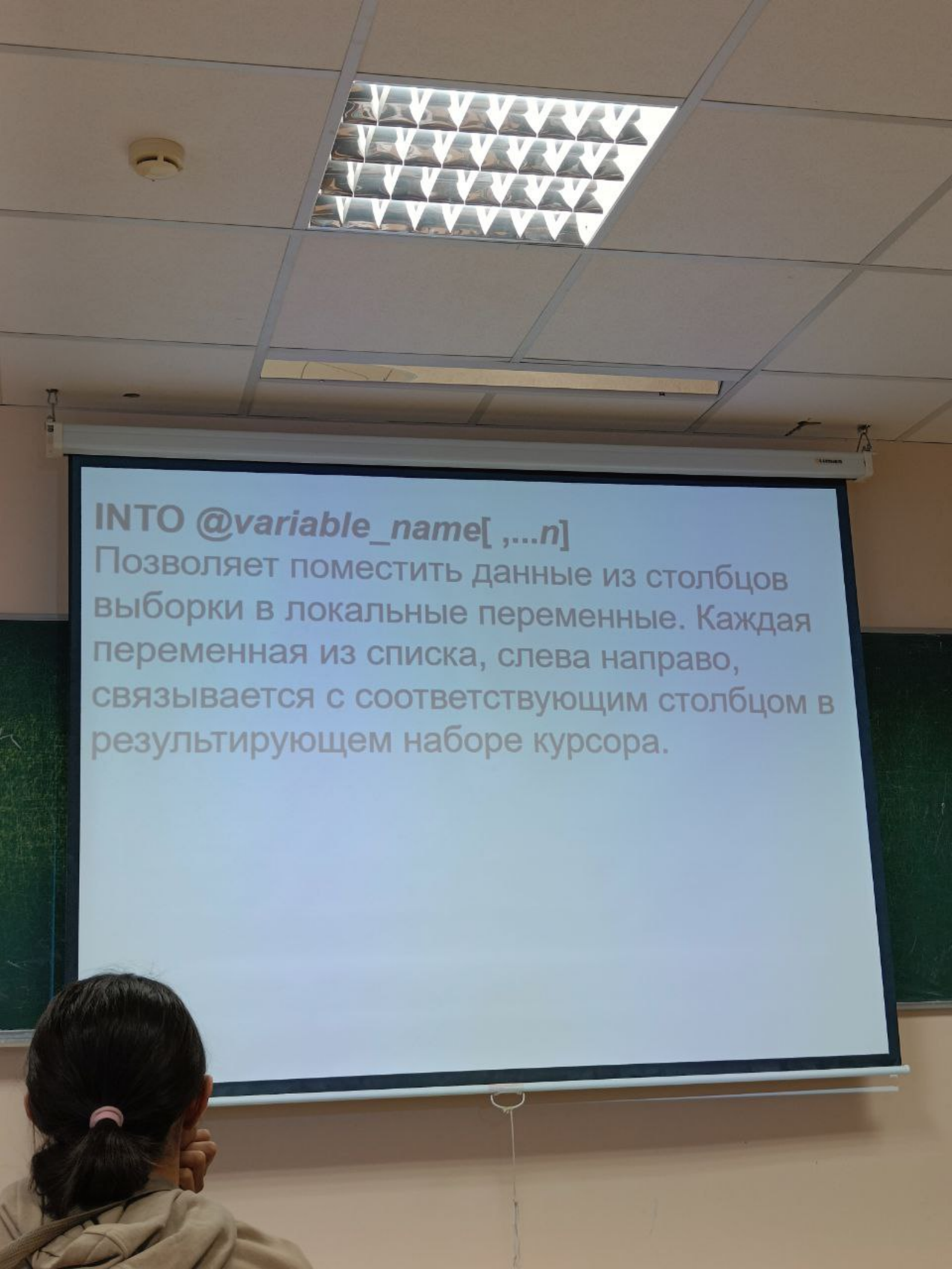

ABSOLUTE { *n*| @*nvar*}
Если *n* или @*nvar* имеет положительное значение, возвращает строку, отстоящую на *n* строк от начала курсора, и делает возвращенную строку новой текущей строкой.

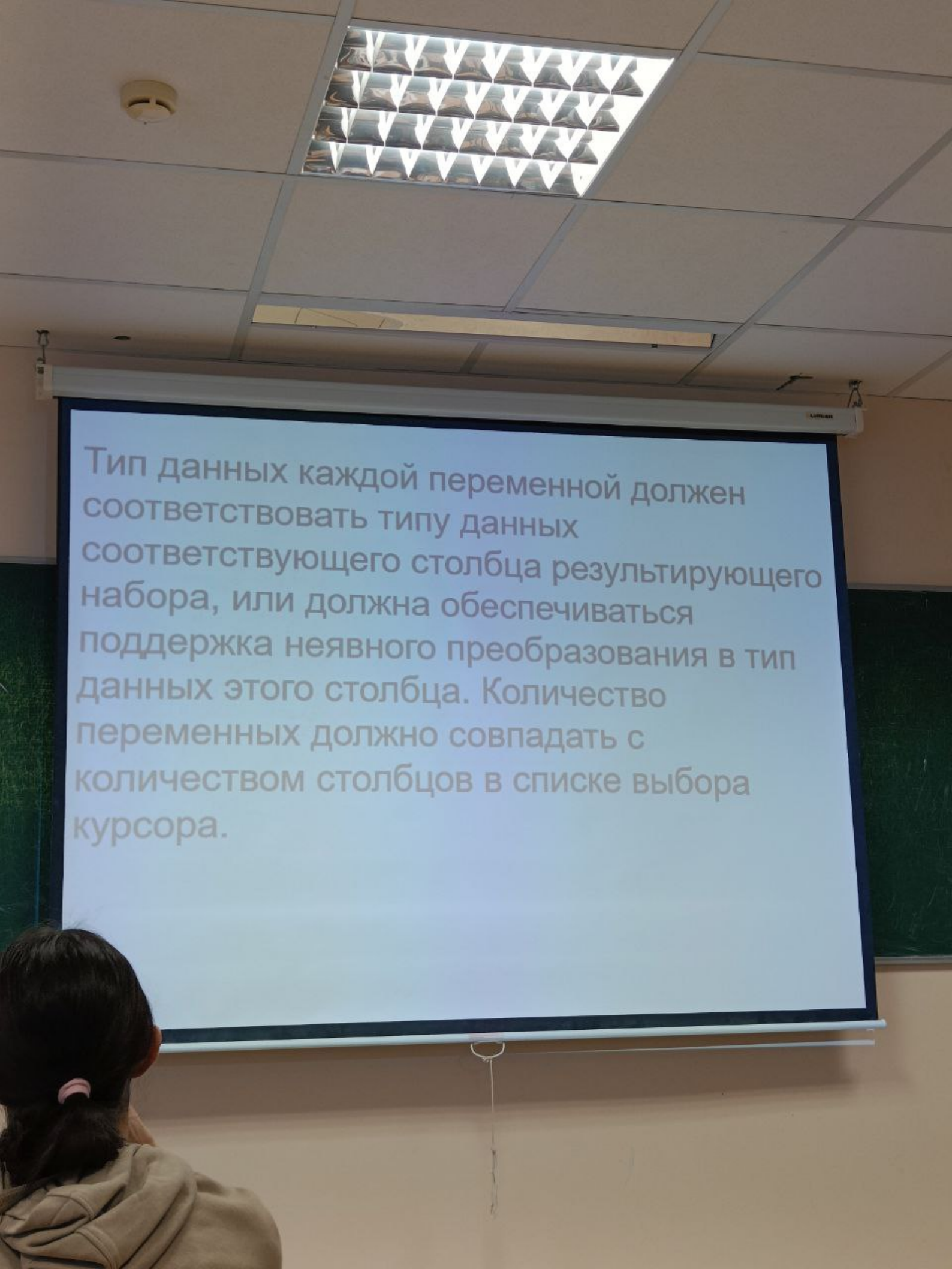

Если *n* или @*nvar* имеет отрицательное значение, возвращает строку, отстоящую на *n* строк от конца курсора, и делает возвращенную строку новой текущей строкой. Если *n* или @*nvar* равно 0, строки не возвращаются. Значение *n* должно быть целочисленной константой, а значение @*nvar* должно иметь тип smallint, tinyint или int.

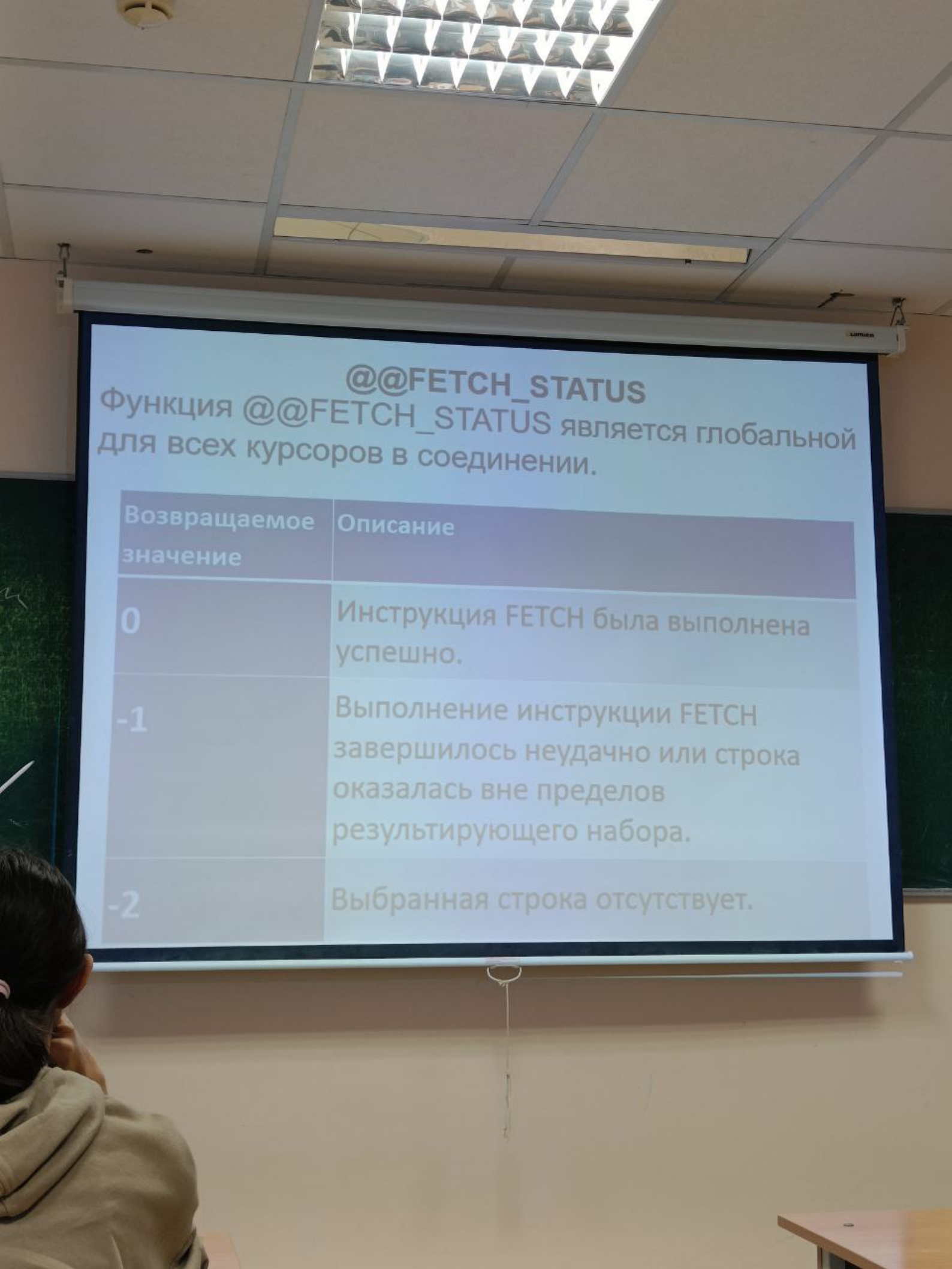

RELATIVE { n| @nvar}
Если n или @nvar имеет положительное значение, возвращает строку, отстоящую на n строк от текущей строки, и делает возвращенную строку новой текущей строкой.

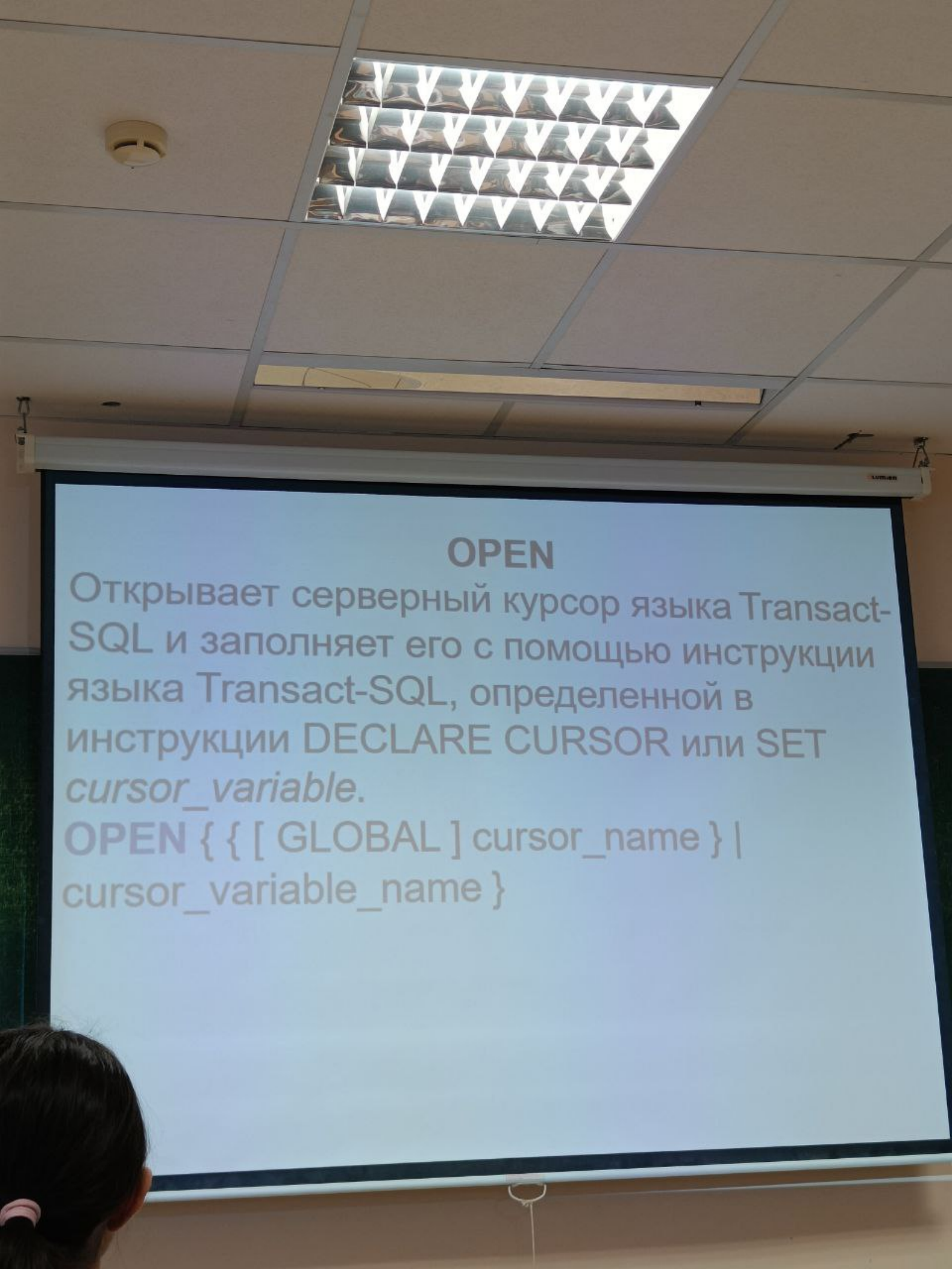

Если *n* или @*nvar* имеют отрицательное значение, возвращает строку, предшествующую на *n* строк текущей строке, и делает возвращенную строку новой текущей строкой. Если *n* или @*nvar* равно 0, возвращает текущую строку. Если при первой выборке из курсора инструкция FETCH RELATIVE указывается с отрицательными или равными нулю значениями *n* или @*nvar* , строки не возвращаются.

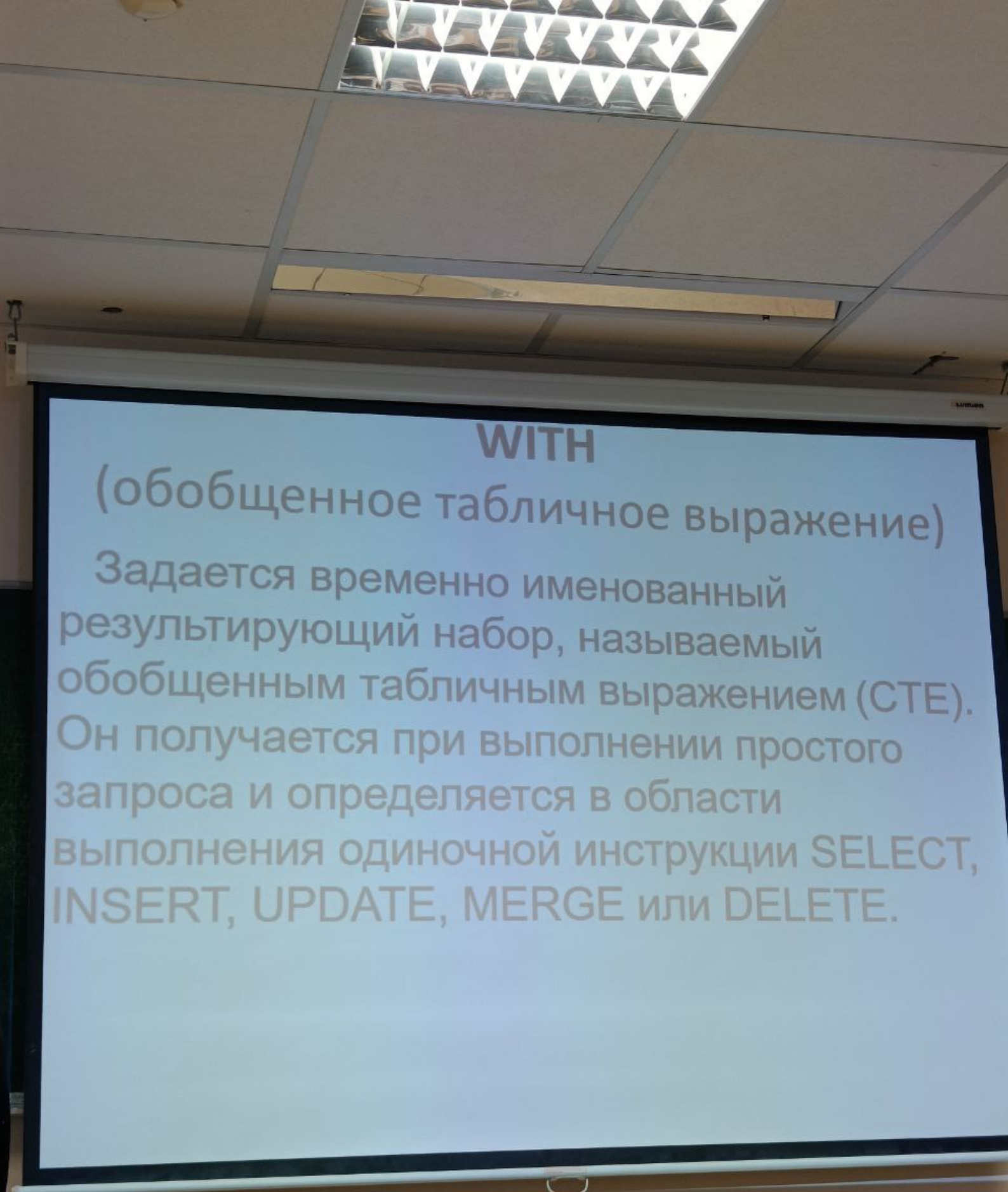

GLOBAL
Указывает, что аргумент cursor_name ссылается на глобальный курсор.
cursor_name
Имя открытого курсора, из которого должна быть произведена выборка.
@cursor_variable_name
Имя переменной курсора, ссылающейся на открытый курсор, из которого должна быть произведена выборка.

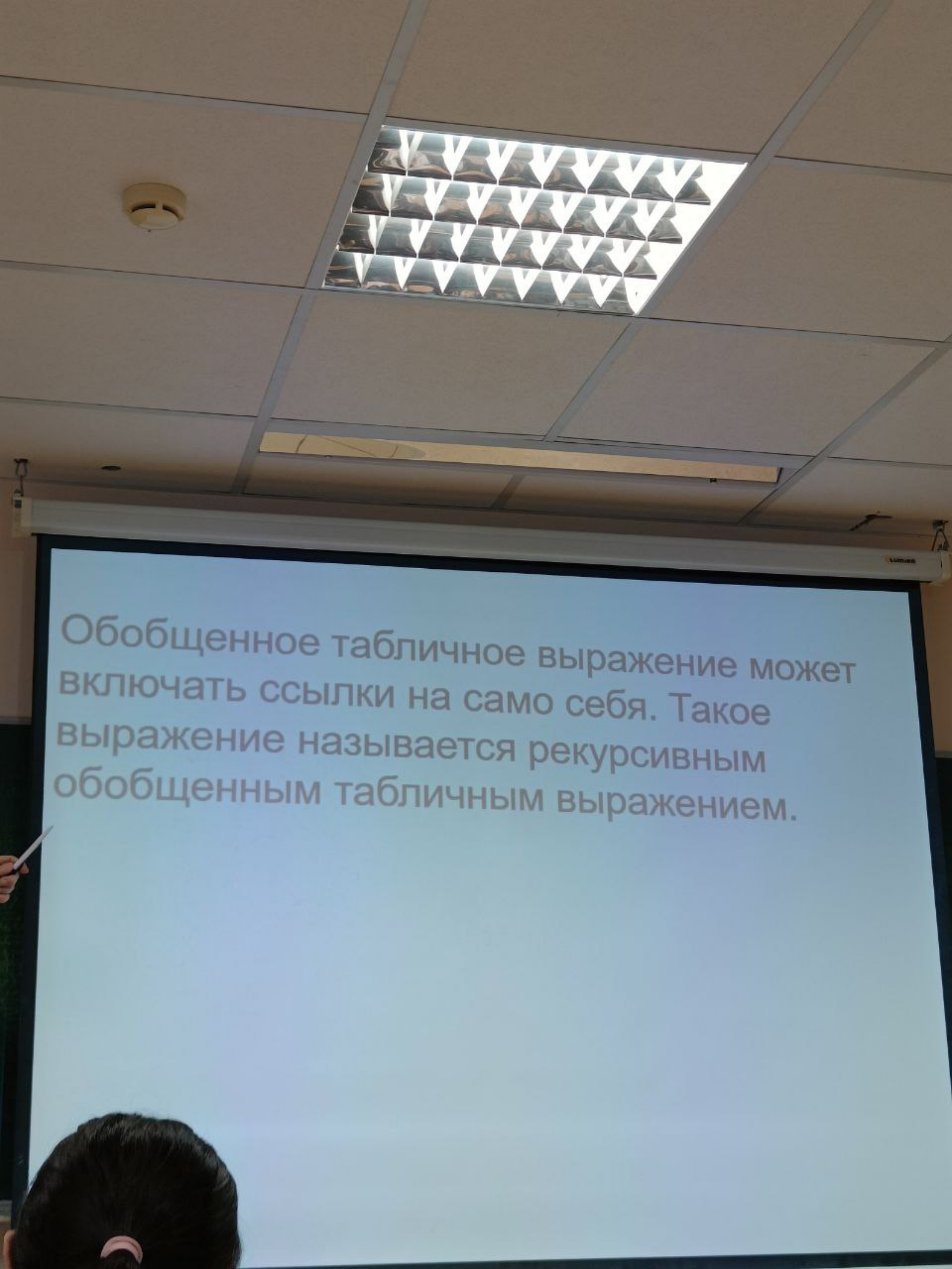

INTO @variable_name[,...n]
Позволяет поместить данные из столбцов выборки в локальные переменные. Каждая переменная из списка, слева направо, связывается с соответствующим столбцом в результирующем наборе курсора.

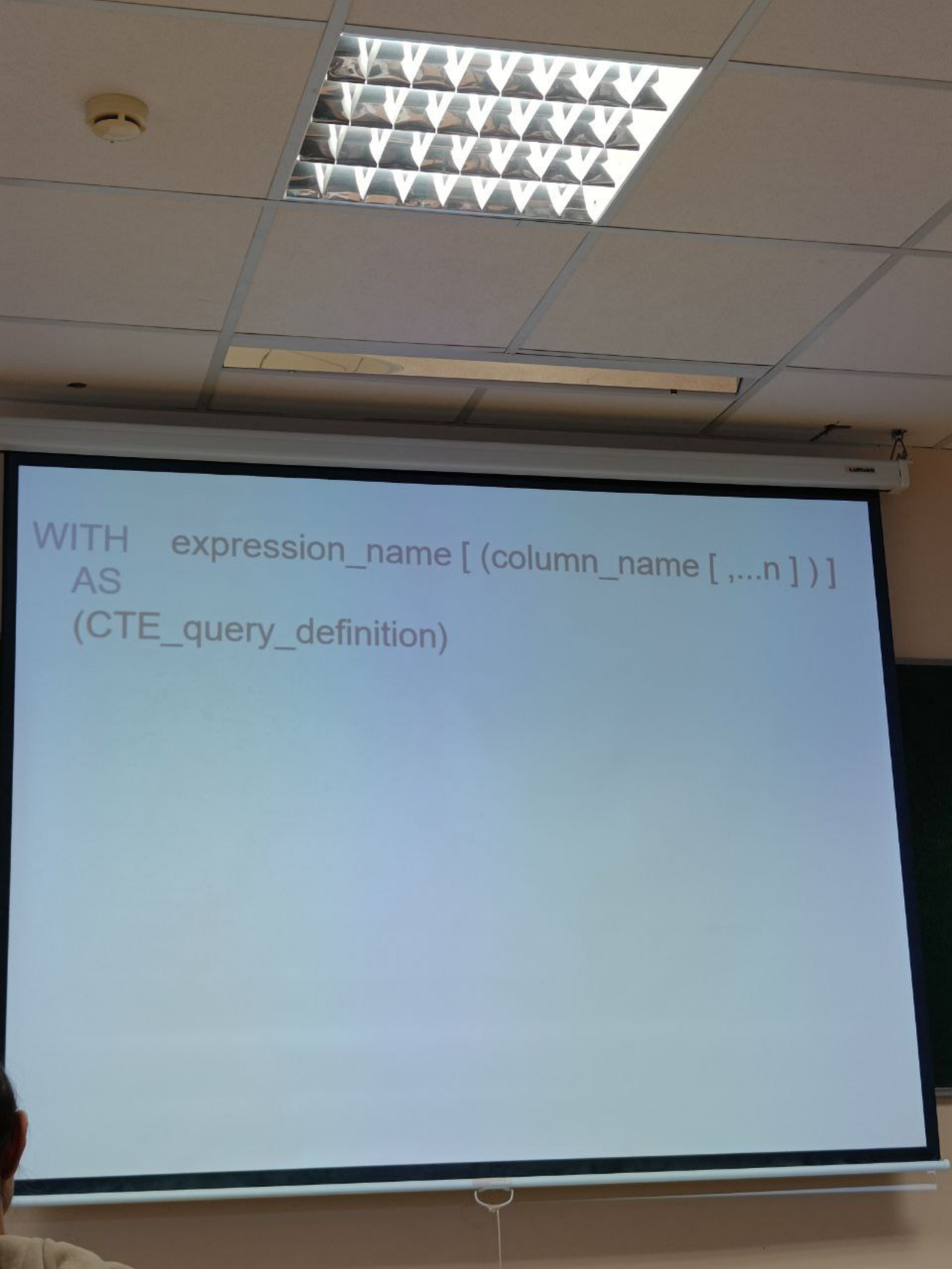

Тип данных каждой переменной должен соответствовать типу данных соответствующего столбца результирующего набора, или должна обеспечиваться поддержка неявного преобразования в тип данных этого столбца. Количество переменных должно совпадать с количеством столбцов в списке выбора курсора.

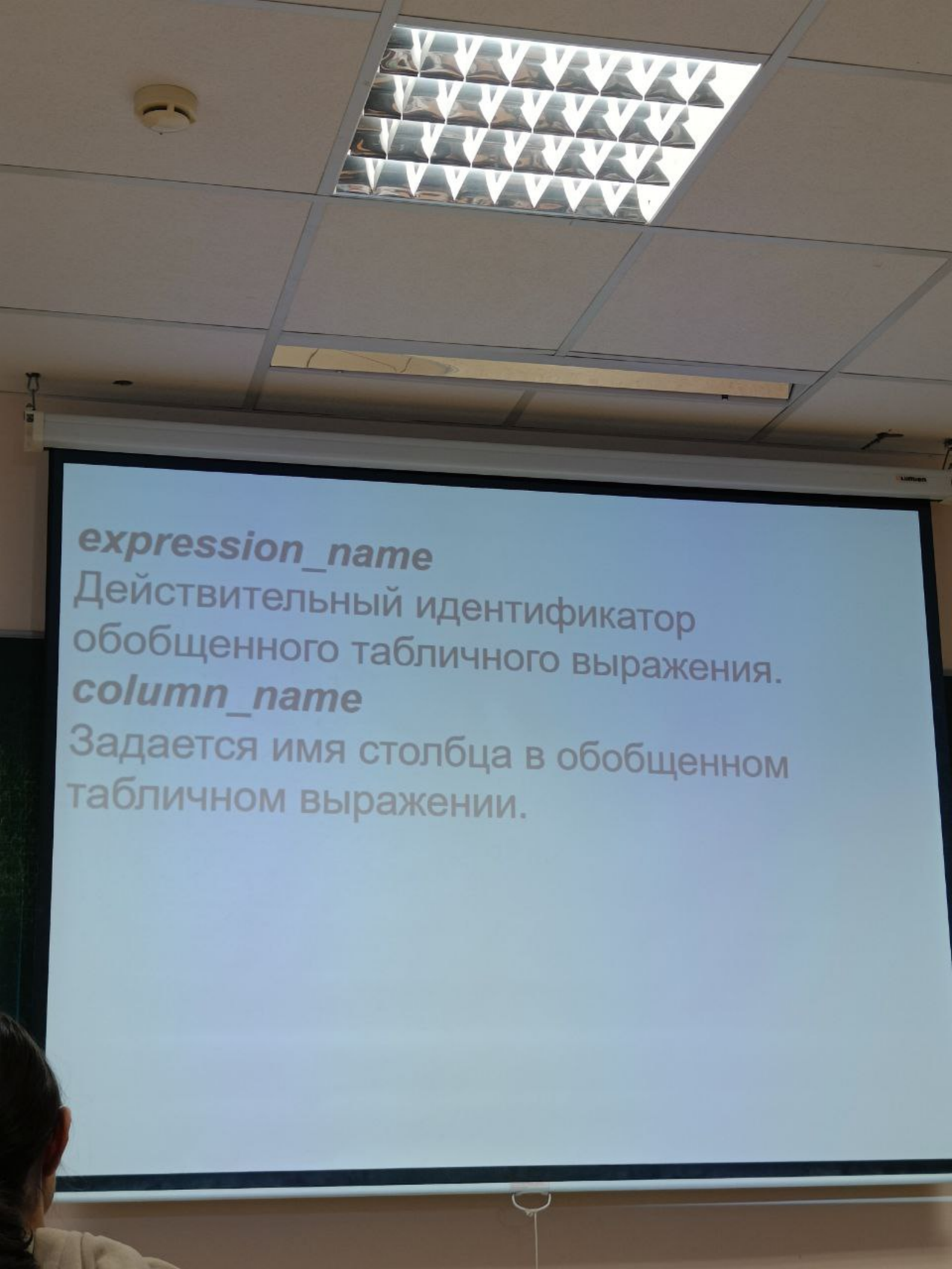

@@FETCH_STATUS
Функция @@FETCH_STATUS является глобальной для всех курсоров в соединении.
Возвращаемое значение
Описание
0
Инструкция FETCH была выполнена успешно.
-1
Выполнение инструкции FETCH завершилось неудачно или строка оказалась вне пределов результирующего набора.
-2
Выбранная строка отсутствует.

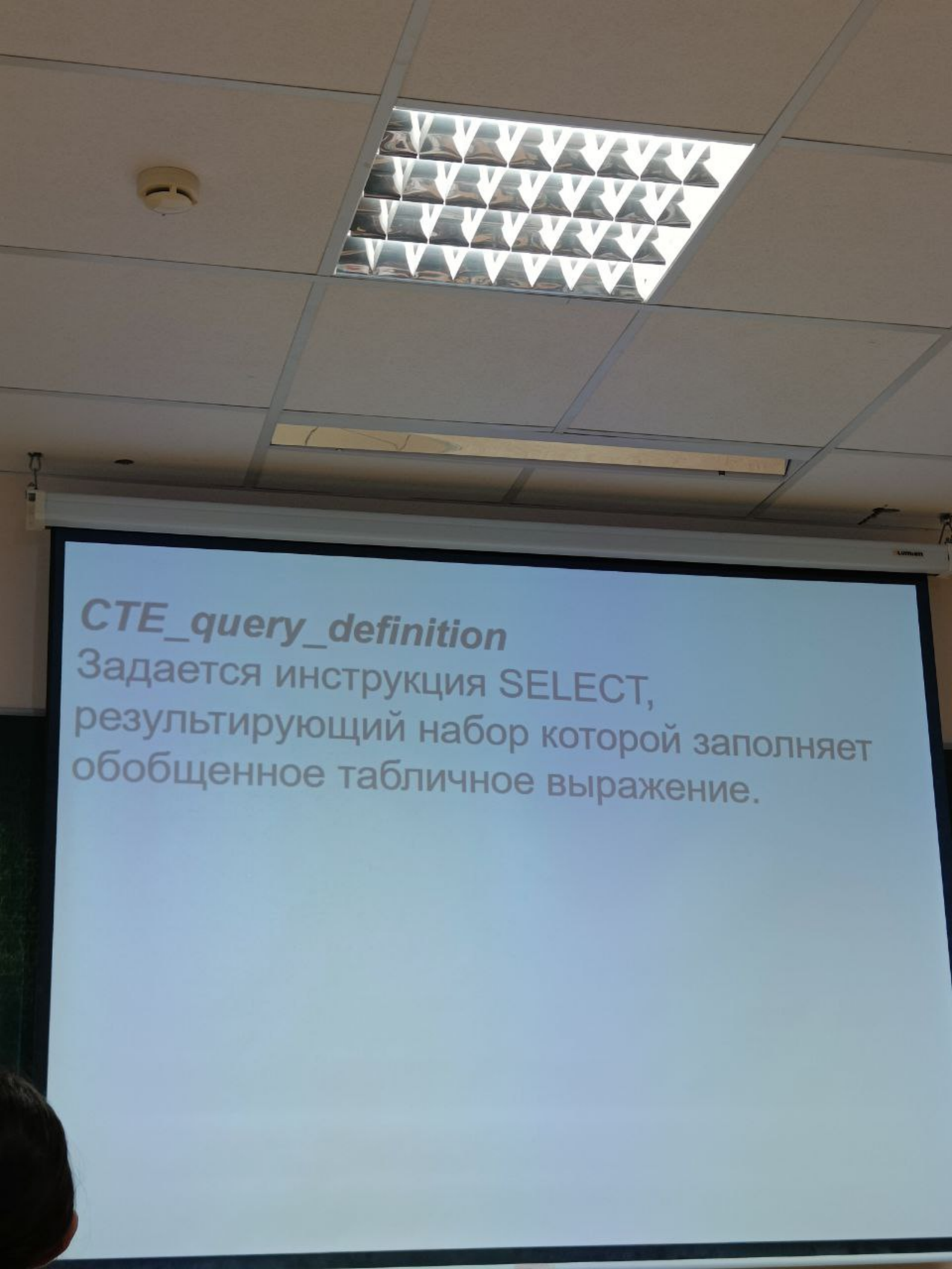

OPEN
Открывает серверный курсор языка Transact-SQL и заполняет его с помощью инструкции языка Transact-SQL, определенной в инструкции DECLARE CURSOR или SET cursor_variable.
OPEN { { [GLOBAL] cursor_name } | cursor_variable_name }

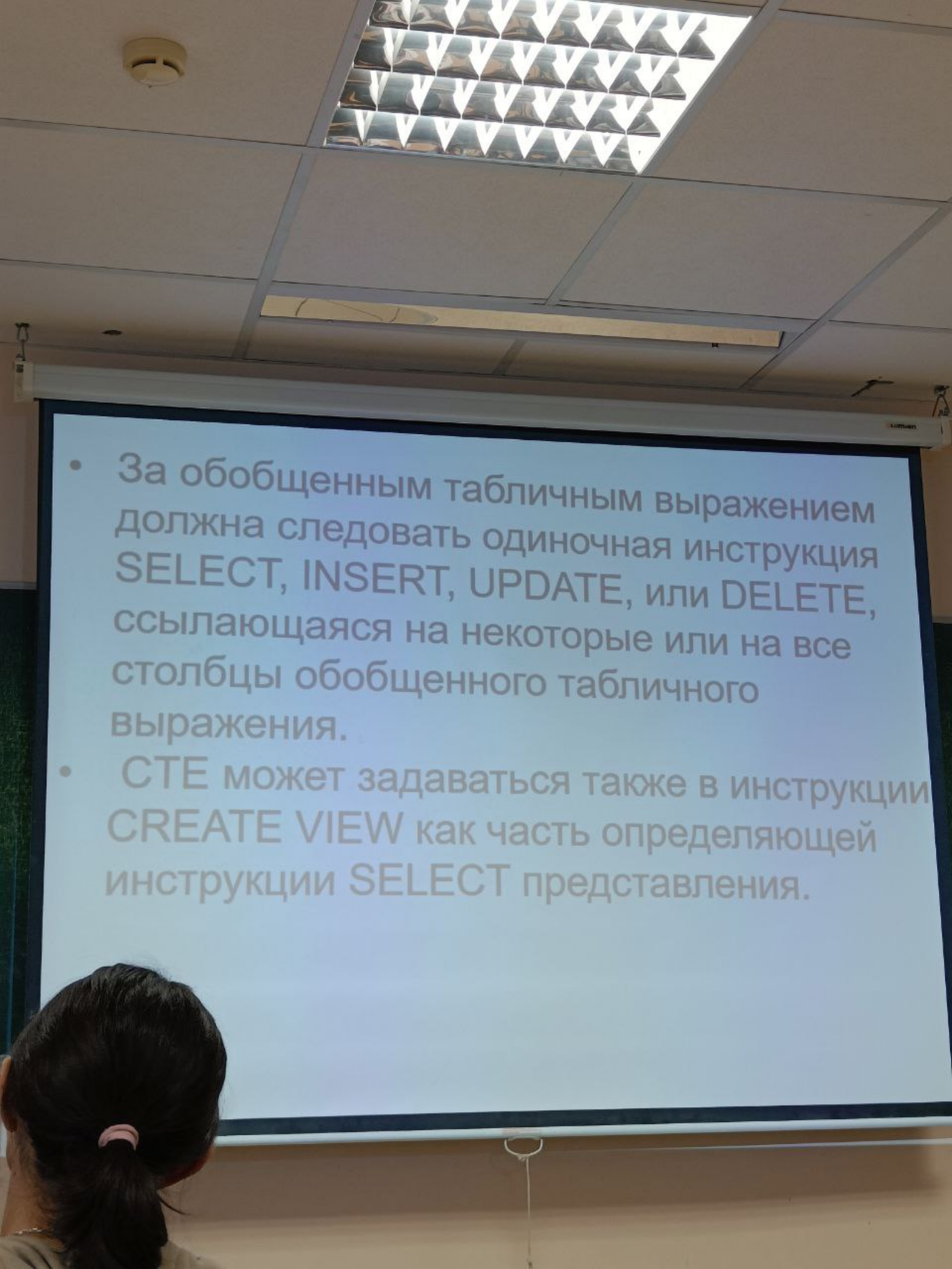

WITH
(обобщенное табличное выражение)
Задается временно именованный результирующий набор, называемый обобщенным табличным выражением (CTE). Он получается при выполнении простого запроса и определяется в области выполнения одиночной инструкции SELECT, INSERT, UPDATE, MERGE или DELETE.

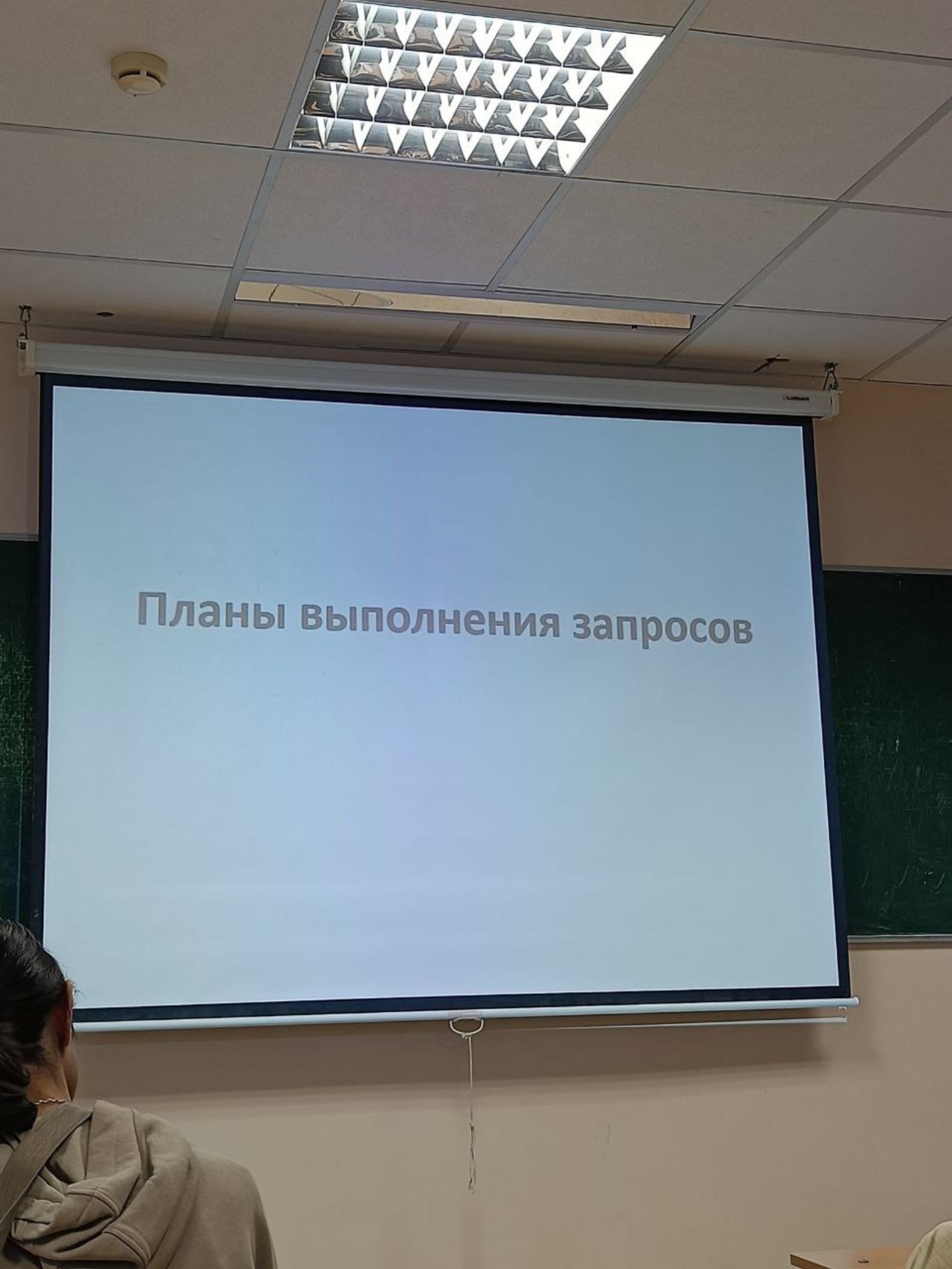

Обобщенное табличное выражение может включать ссылки на само себя. Такое выражение называется рекурсивным обобщенным табличным выражением.

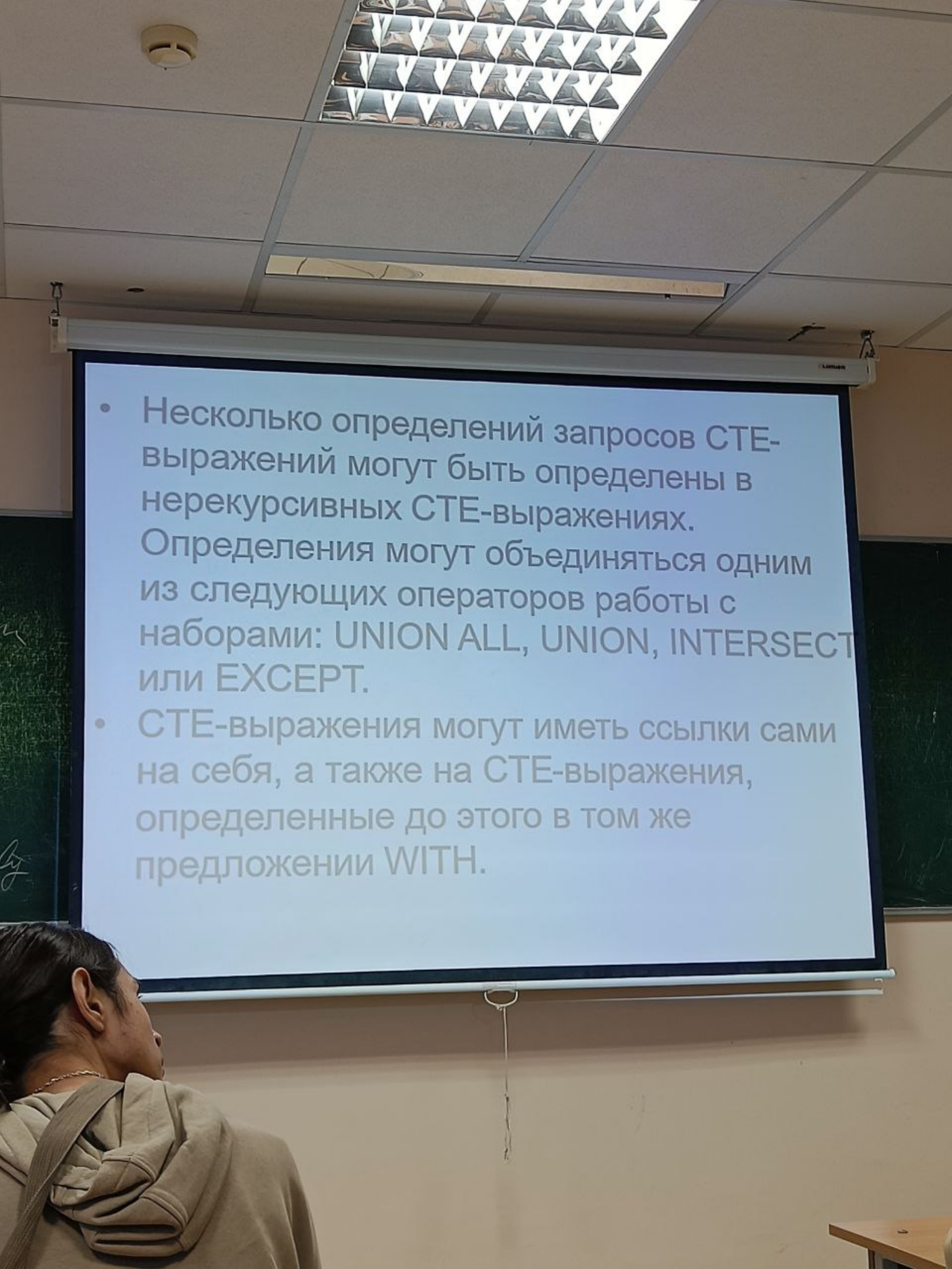

WITH expression_name [(column_name [,...n])]
AS
(CTE_query_definition)

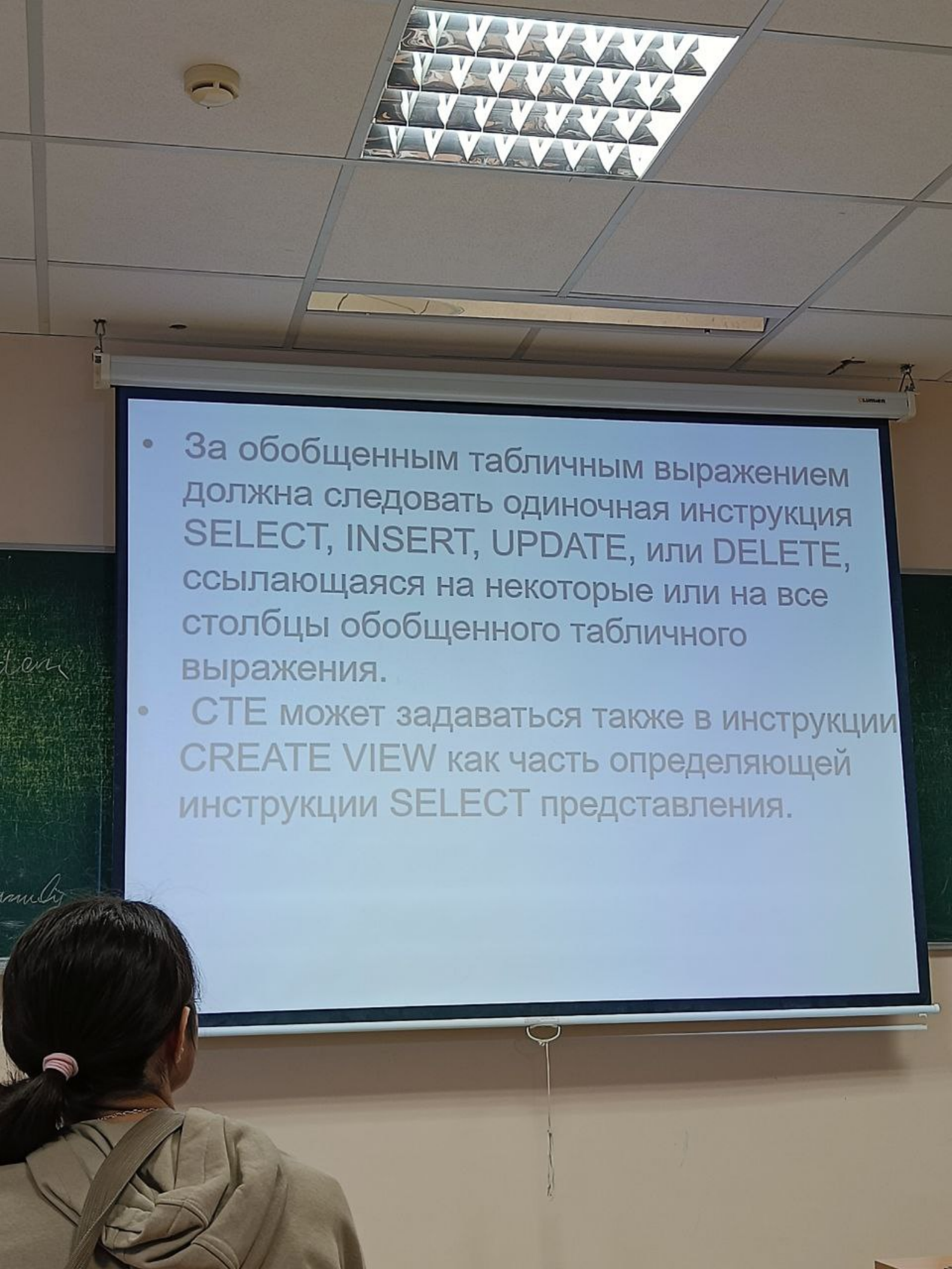

expression_name
Действительный идентификатор обобщенного табличного выражения.
column_name
Задается имя столбца в обобщенном табличном выражении.

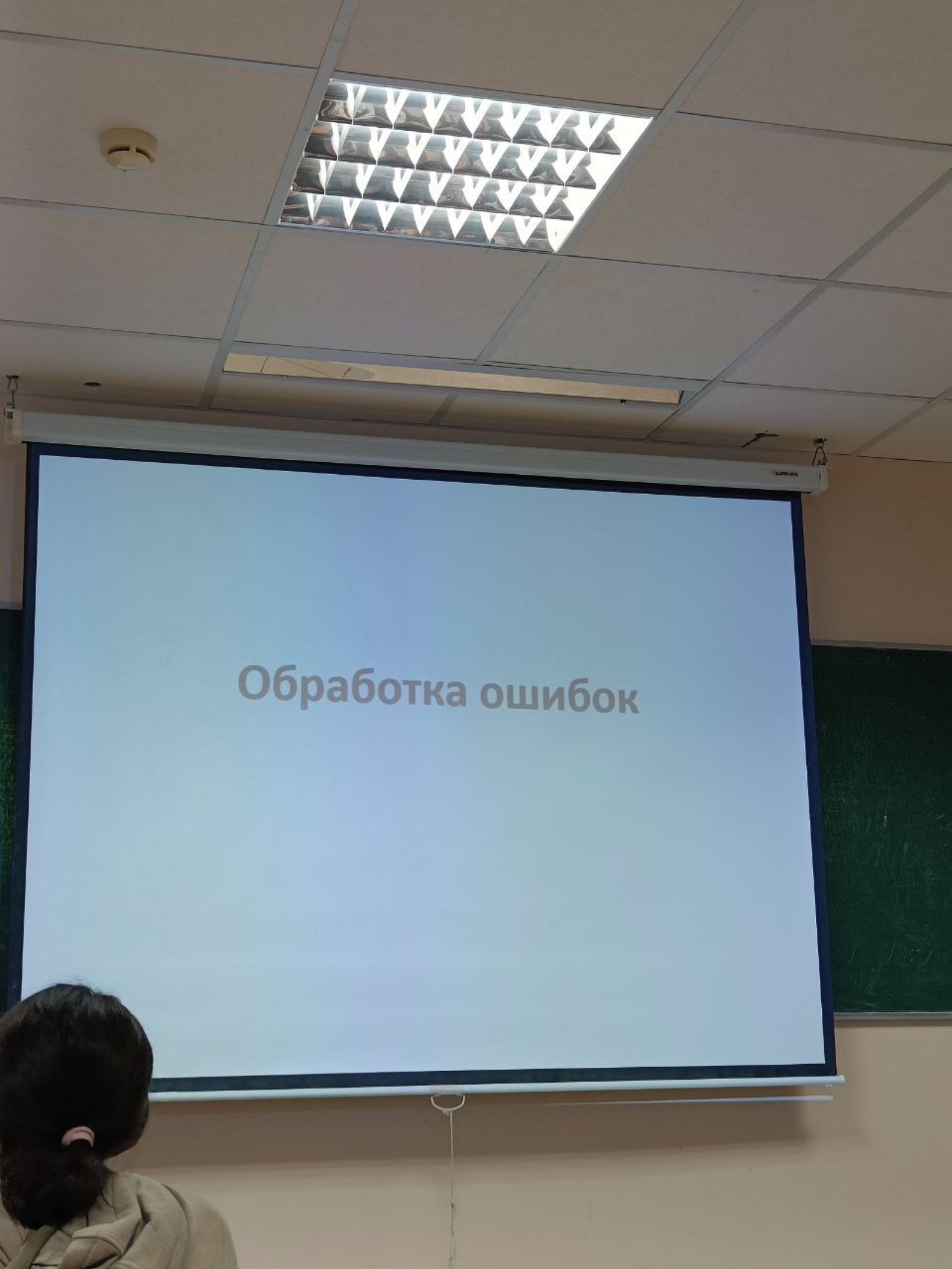

CTE_query_definition
Задается инструкция SELECT, результирующий набор которой заполняет обобщенное табличное выражение.

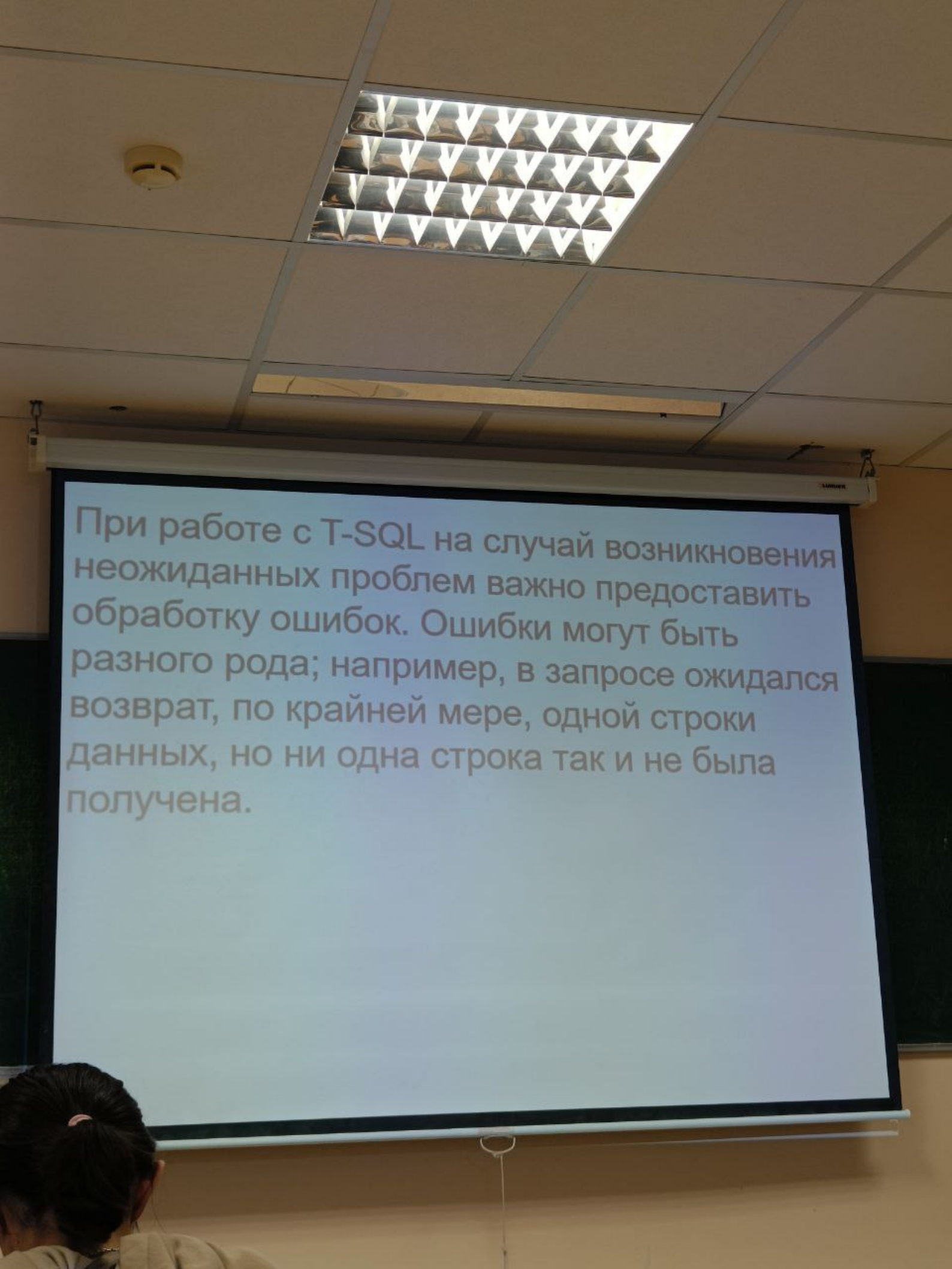

• За обобщенным табличным выражением должна следовать одиночная инструкция SELECT, INSERT, UPDATE, или DELETE, ссылающаяся на некоторые или на все столбцы обобщенного табличного выражения.
• CTE может задаваться также в инструкции CREATE VIEW как часть определяющей инструкции SELECT представления.

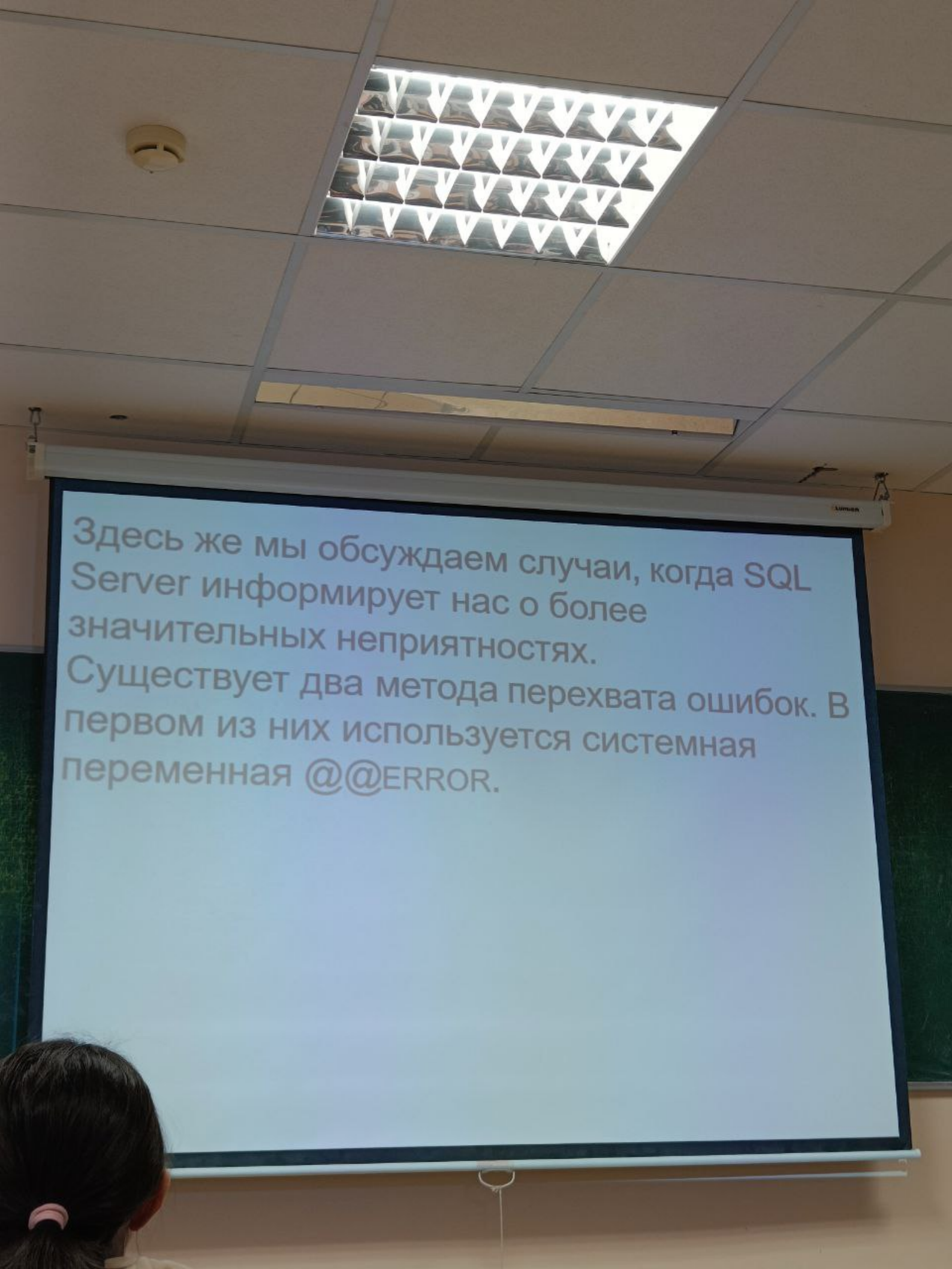

Планы выполнения запросов

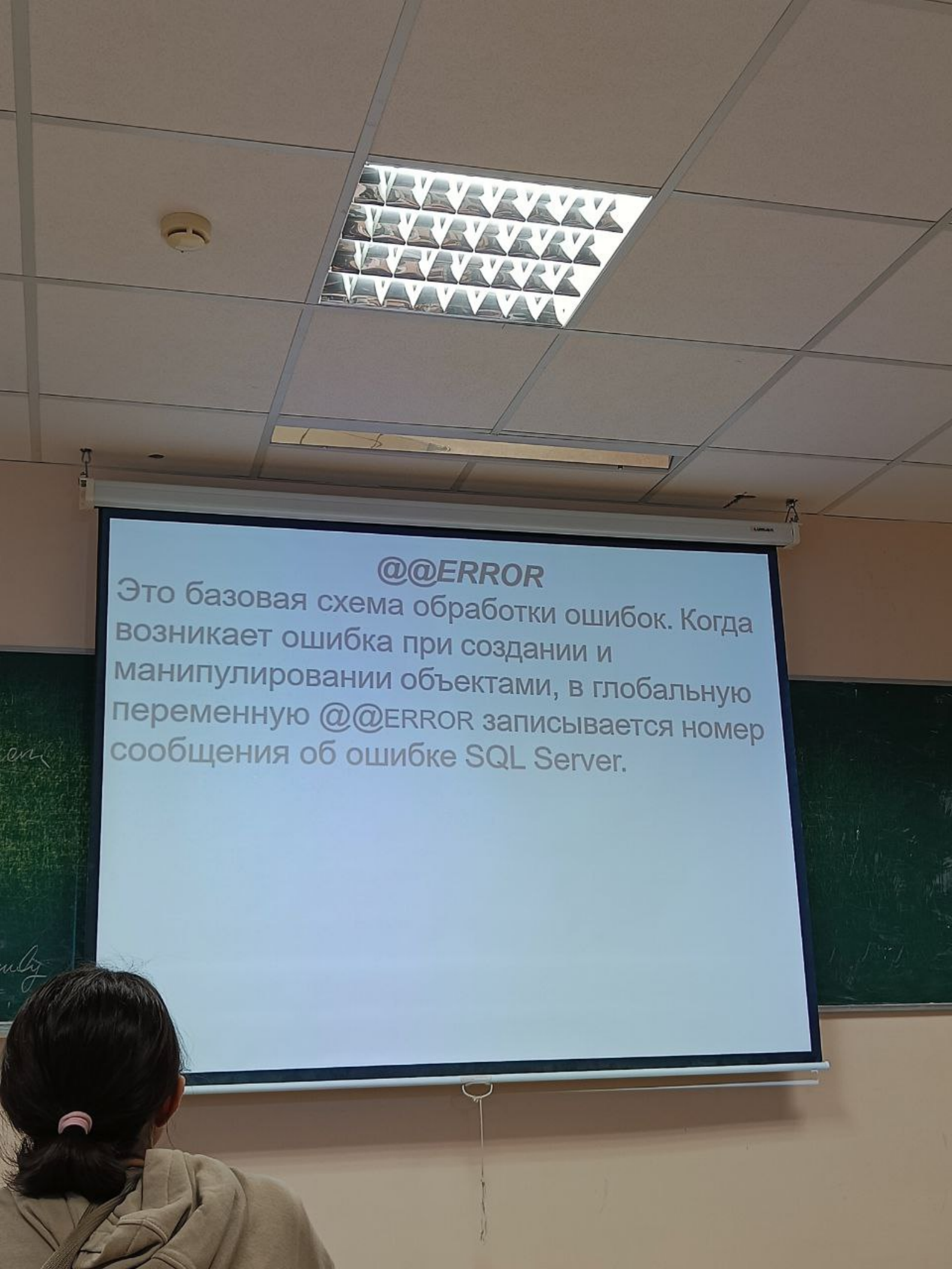

• Несколько определений запросов CTE-выражений могут быть определены в нерекурсивных CTE-выражениях. Определения могут объединяться одним из следующих операторов работы с наборами: UNION ALL, UNION, INTERSECT или EXCEPT.
• CTE-выражения могут иметь ссылки сами на себя, а также на CTE-выражения, определенные до этого в том же предложении WITH.

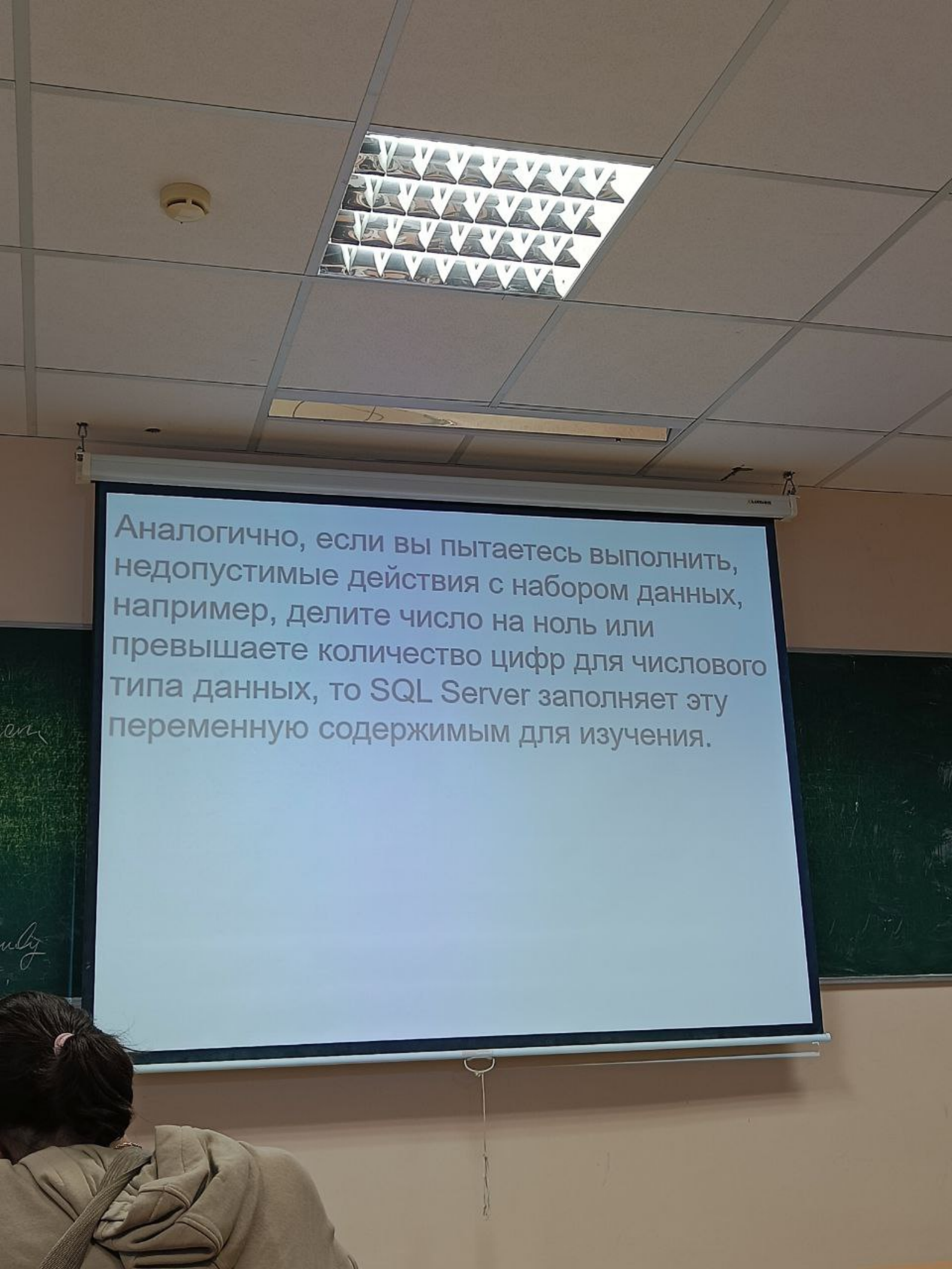

За обобщенным табличным выражением должна следовать одиночная инструкция SELECT, INSERT, UPDATE, или DELETE, ссылающаяся на некоторые или на все столбцы обобщенного табличного выражения.
CTE может задаваться также в инструкции CREATE VIEW как часть определяющей инструкции SELECT представления.

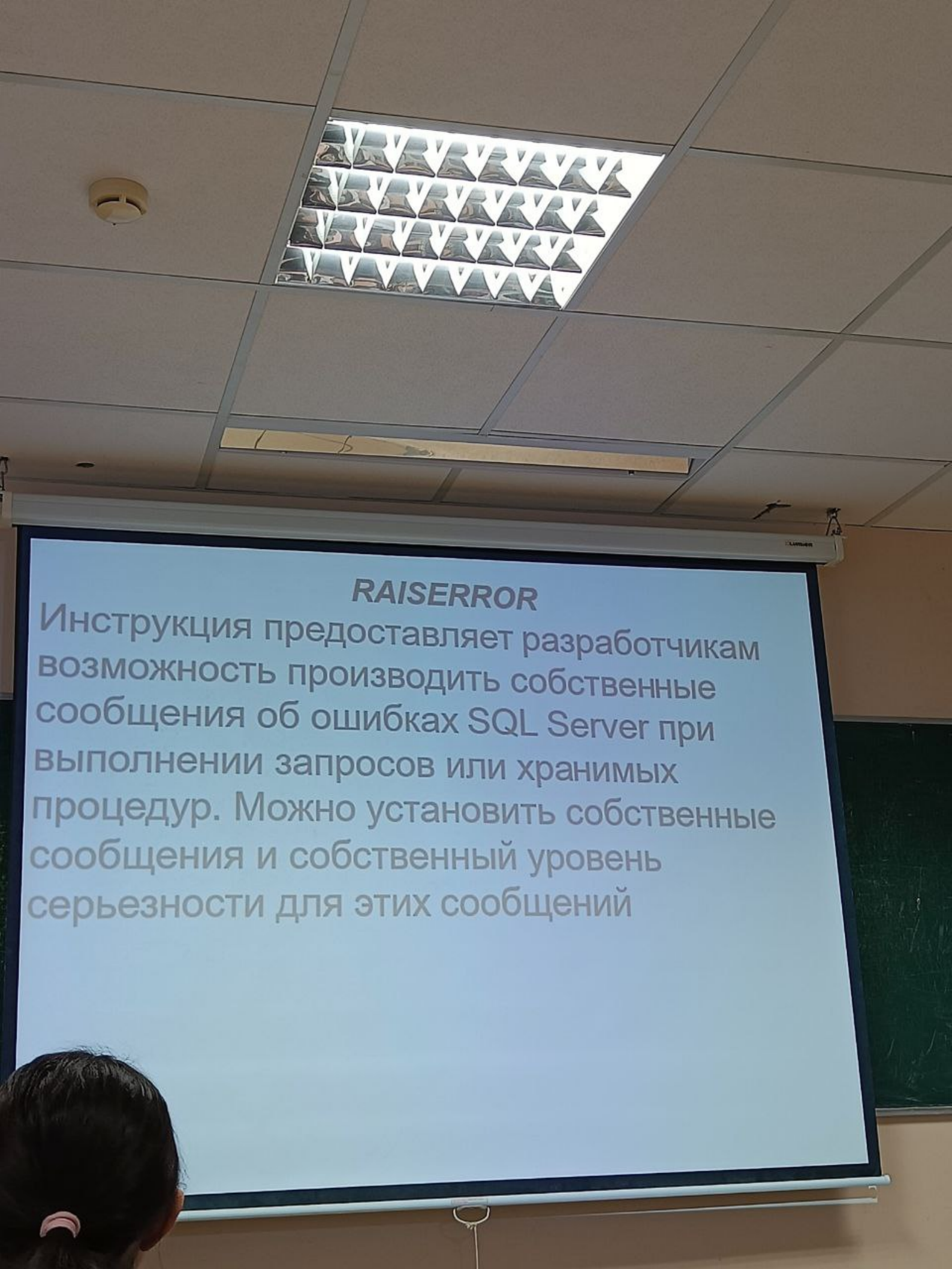

Обработка ошибок

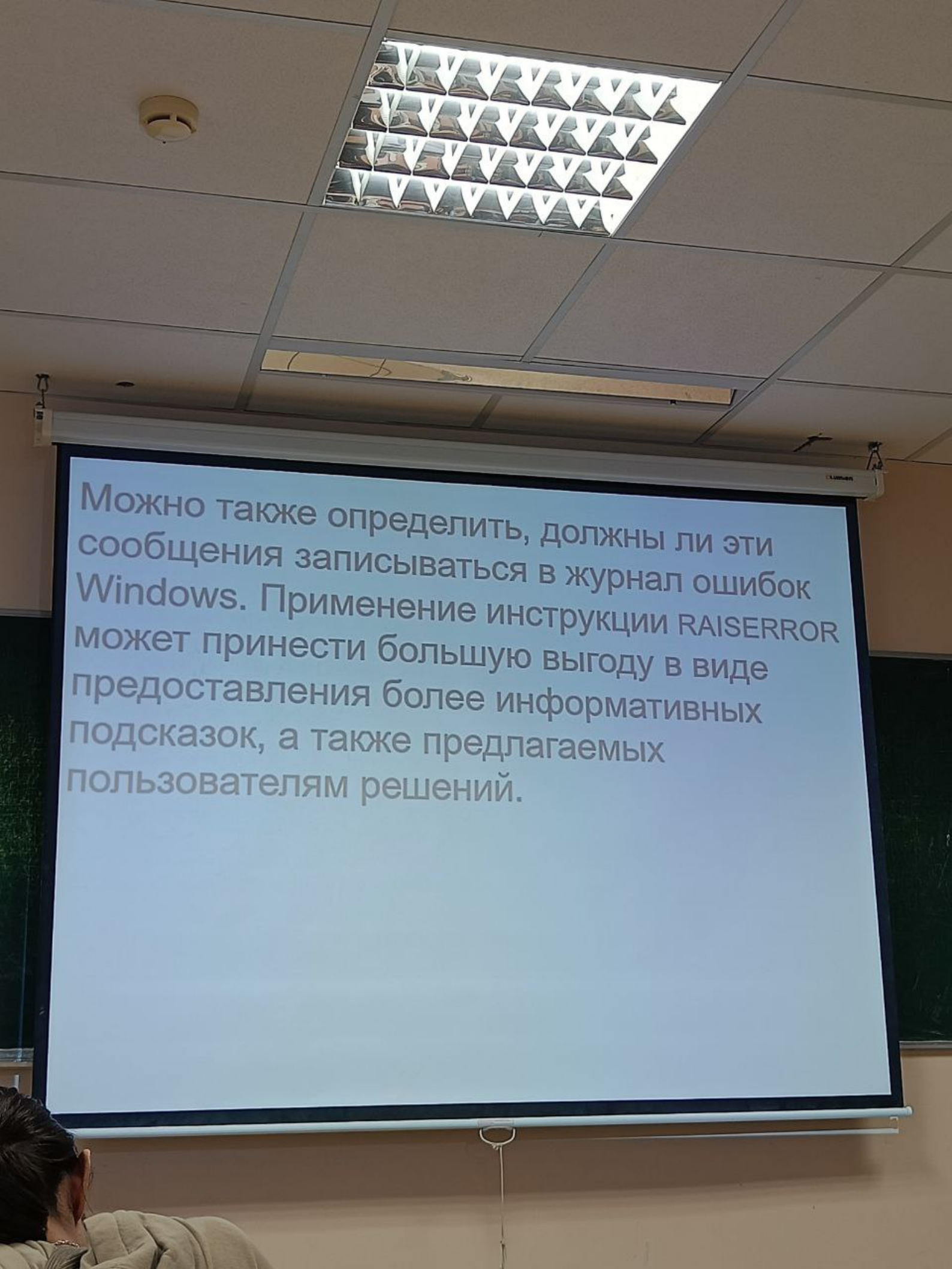

При работе с T-SQL на случай возникновения неожиданных проблем важно предоставить обработку ошибок. Ошибки могут быть разного рода; например, в запросе ожидался возврат, по крайней мере, одной строки данных, но ни одна строка так и не была получена.

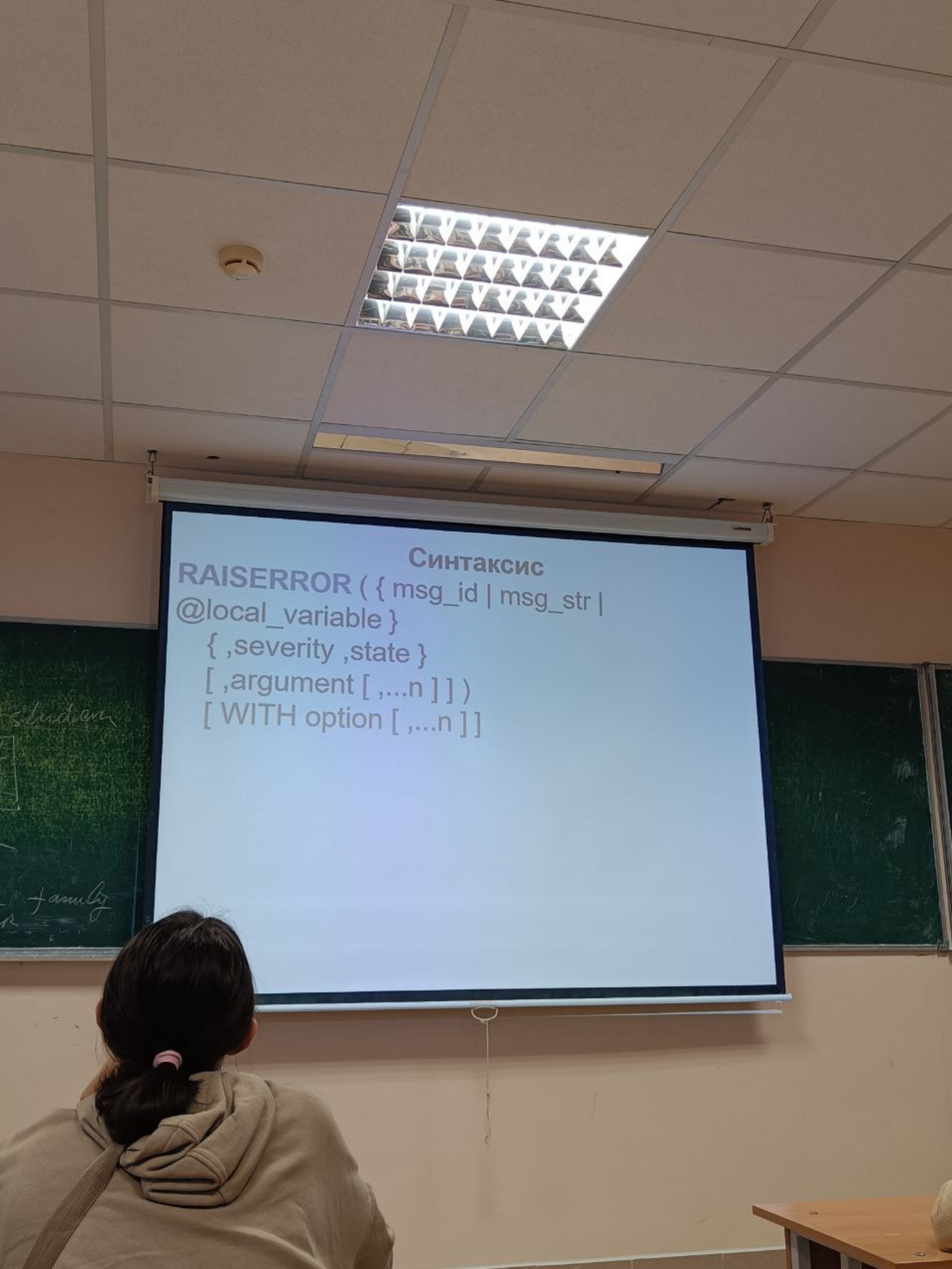

Здесь же мы обсуждаем случаи, когда SQL Server информирует нас о более значительных неприятностях.
Существует два метода перехвата ошибок. В первом из них используется системная переменная @@ERROR.

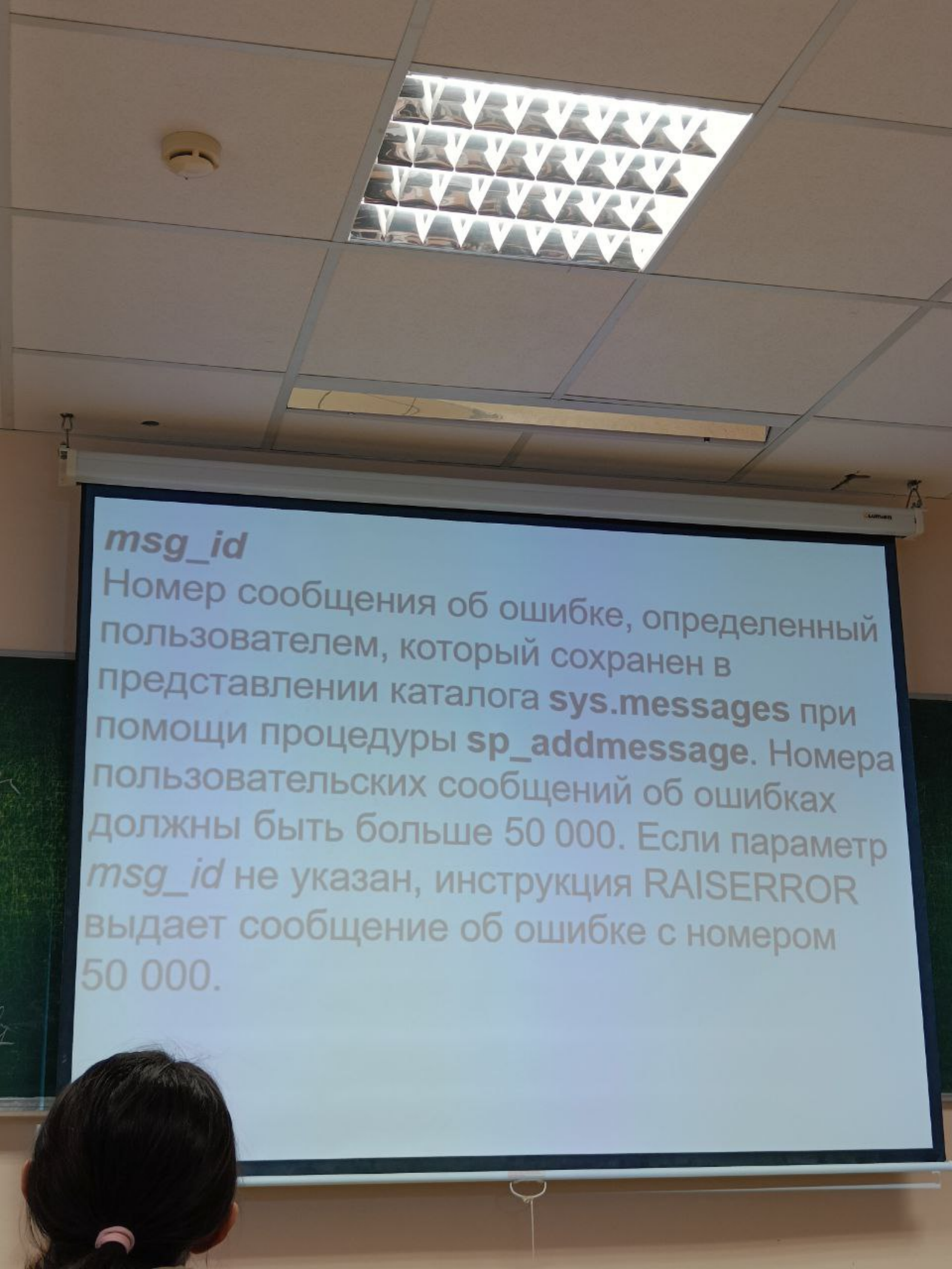

@@ERROR
Это базовая схема обработки ошибок. Когда возникает ошибка при создании и манипулировании объектами, в глобальную переменную @@ERROR записывается номер сообщения об ошибке SQL Server.

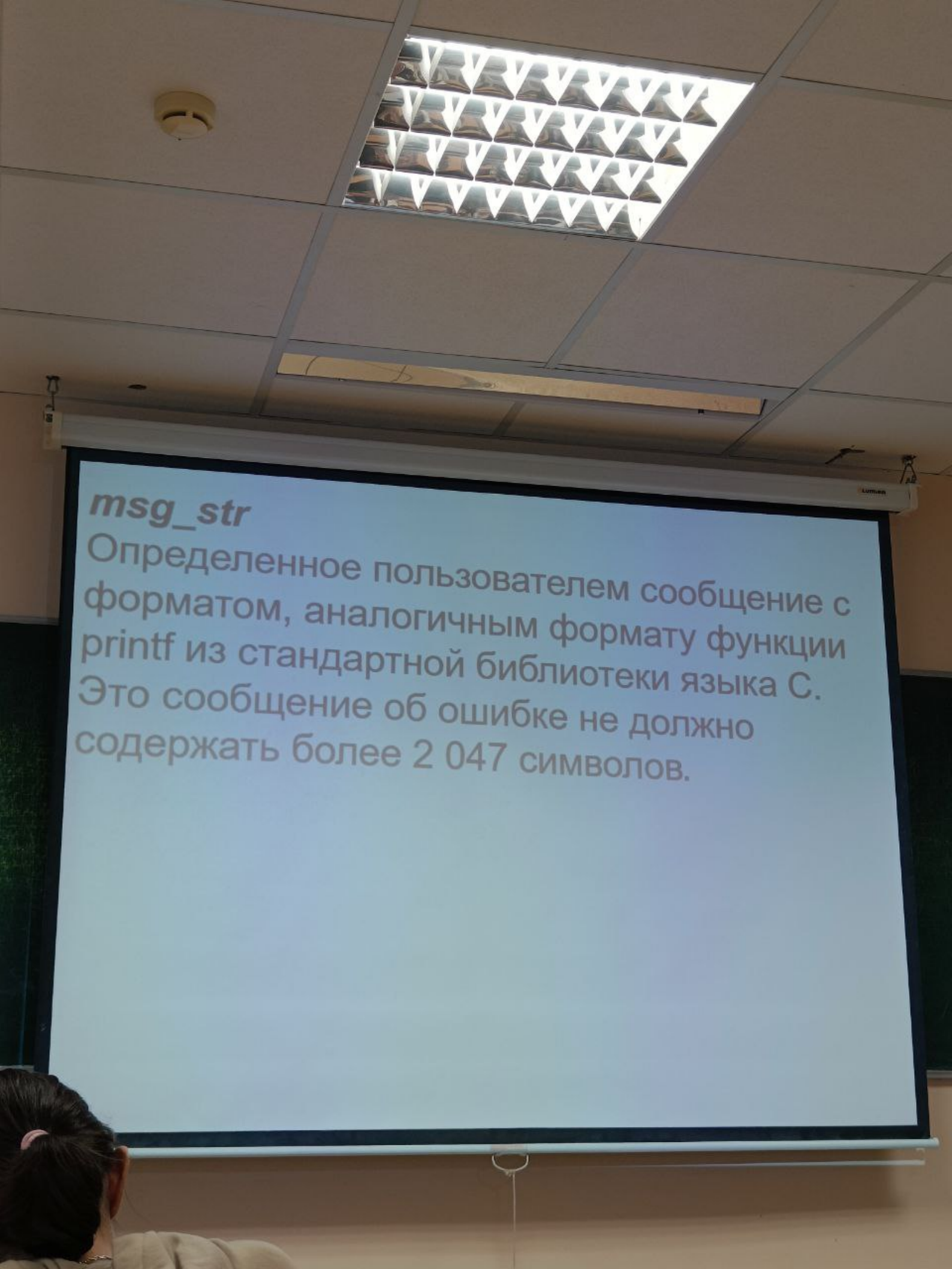

Аналогично, если вы пытаетесь выполнить, недопустимые действия с набором данных, например, делите число на ноль или превышаете количество цифр для числового типа данных, то SQL Server заполняет эту переменную содержимым для изучения.

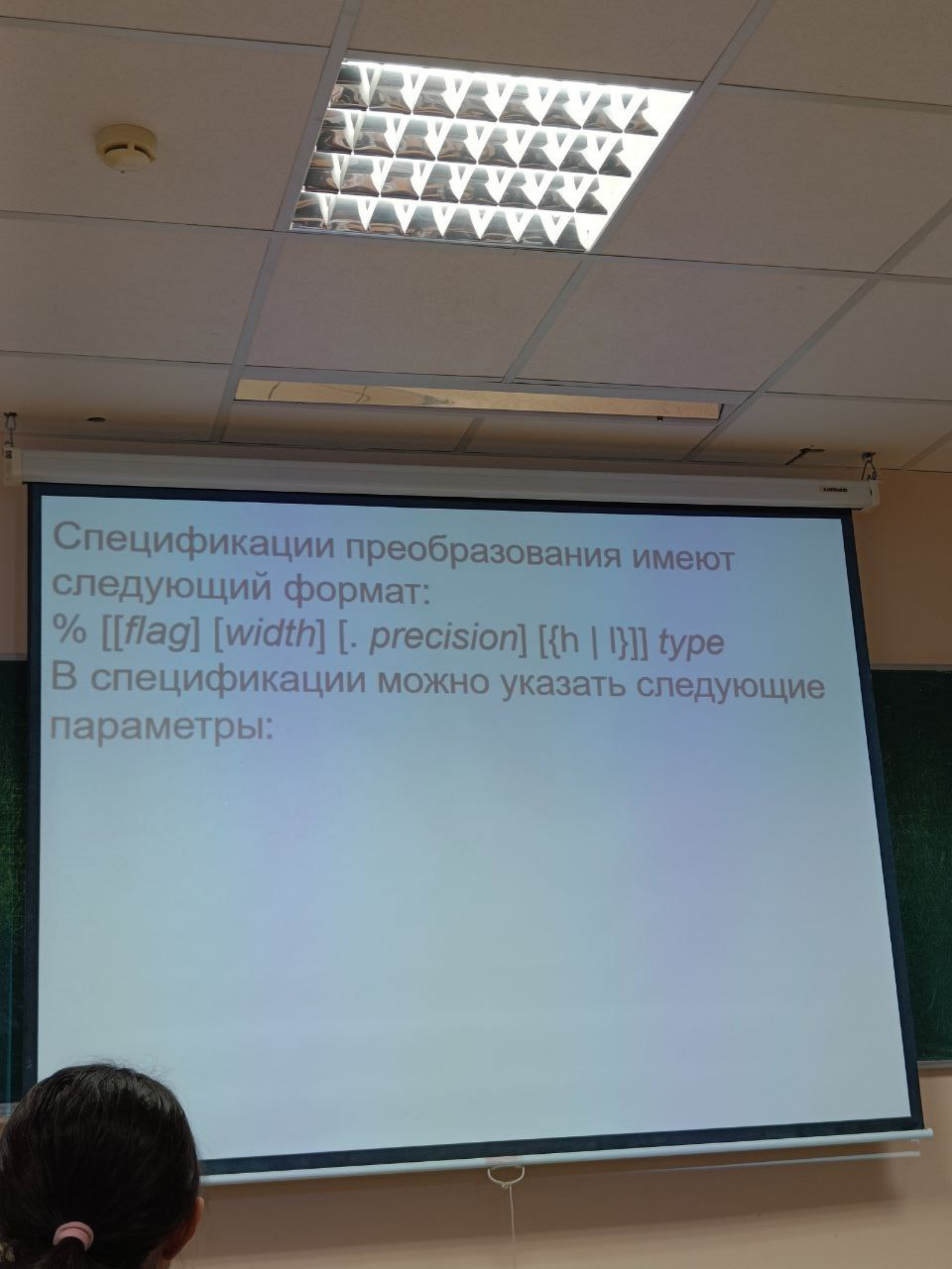

RAISERROR
Инструкция предоставляет разработчикам возможность производить собственные сообщения об ошибках SQL Server при выполнении запросов или хранимых процедур. Можно установить собственные сообщения и собственный уровень серьезности для этих сообщений

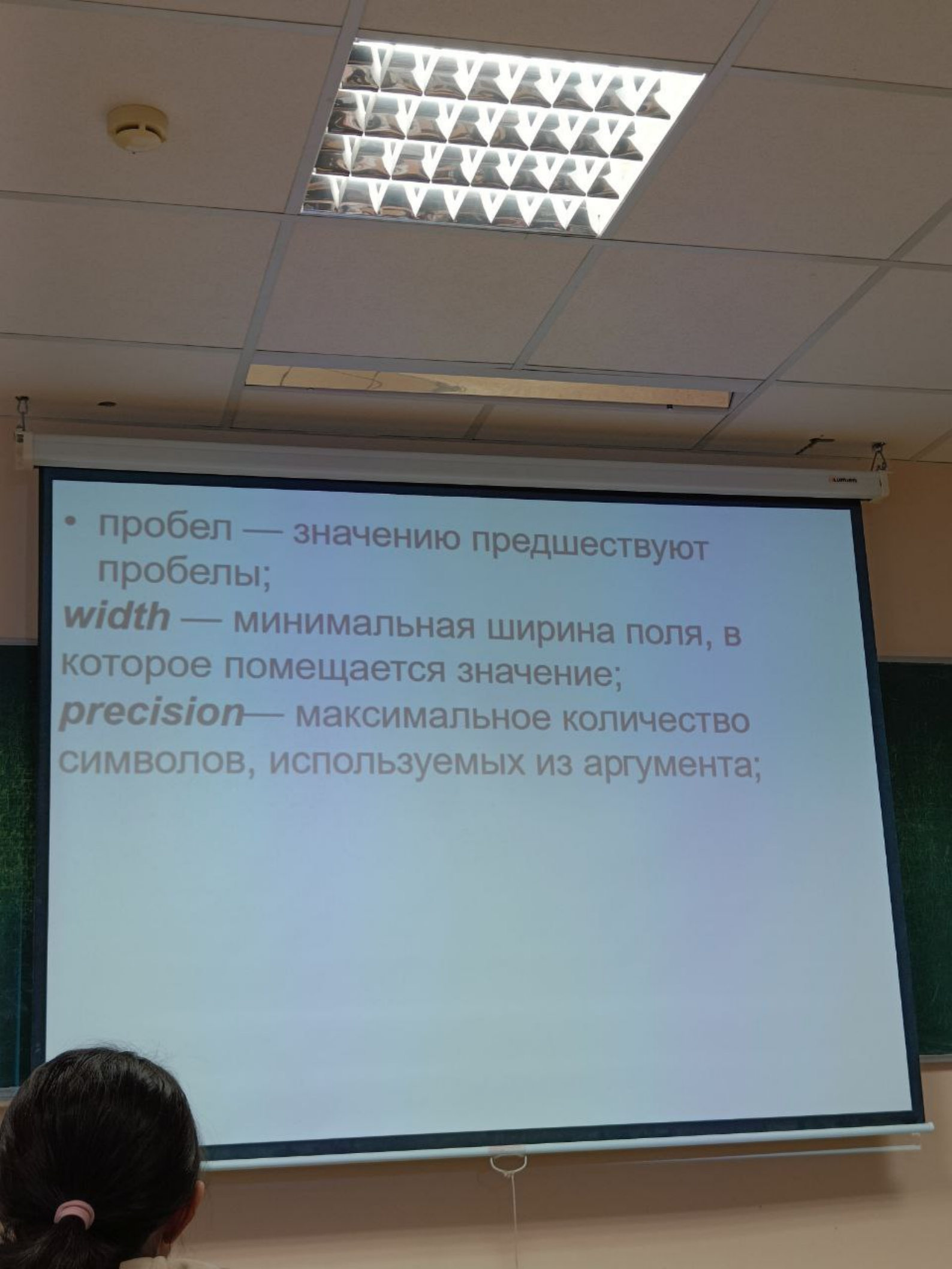

Можно также определить, должны ли эти сообщения записываться в журнал ошибок Windows. Применение инструкции RAISERROR может принести большую выгоду в виде предоставления более информативных подсказок, а также предлагаемых пользователям решений.

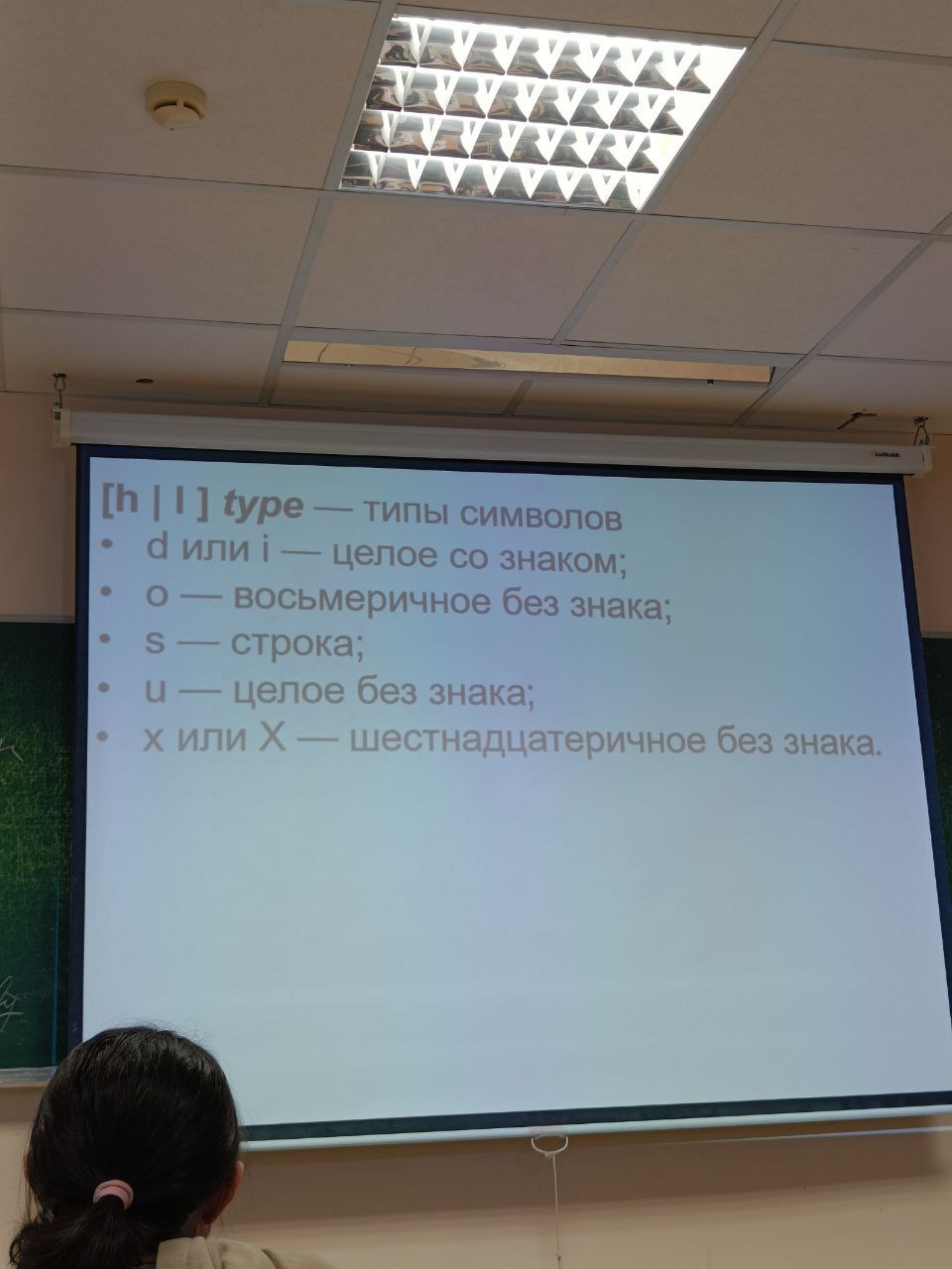

Синтаксис
RAISERROR ({ msg_id | msg_str |
@local_variable }
{ ,severity ,state }
[,argument [,...n]])
[WITH option [,...n]]

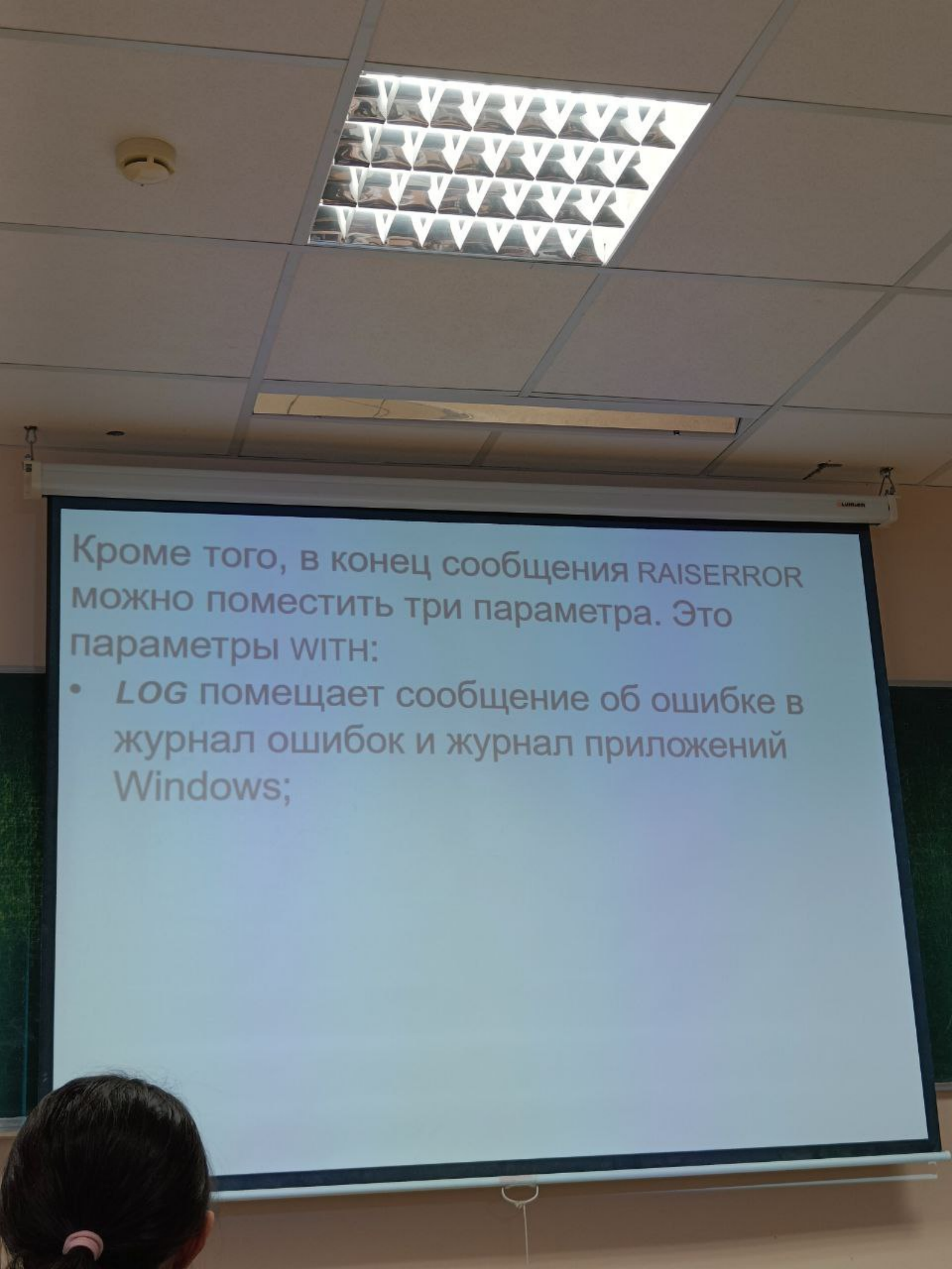

msg_id
Номер сообщения об ошибке, определенный пользователем, который сохранен в представлении каталога sys.messages при помощи процедуры sp_addmessage. Номера пользовательских сообщений об ошибках должны быть больше 50 000. Если параметр msg_id не указан, инструкция RAISERROR выдает сообщение об ошибке с номером 50 000.

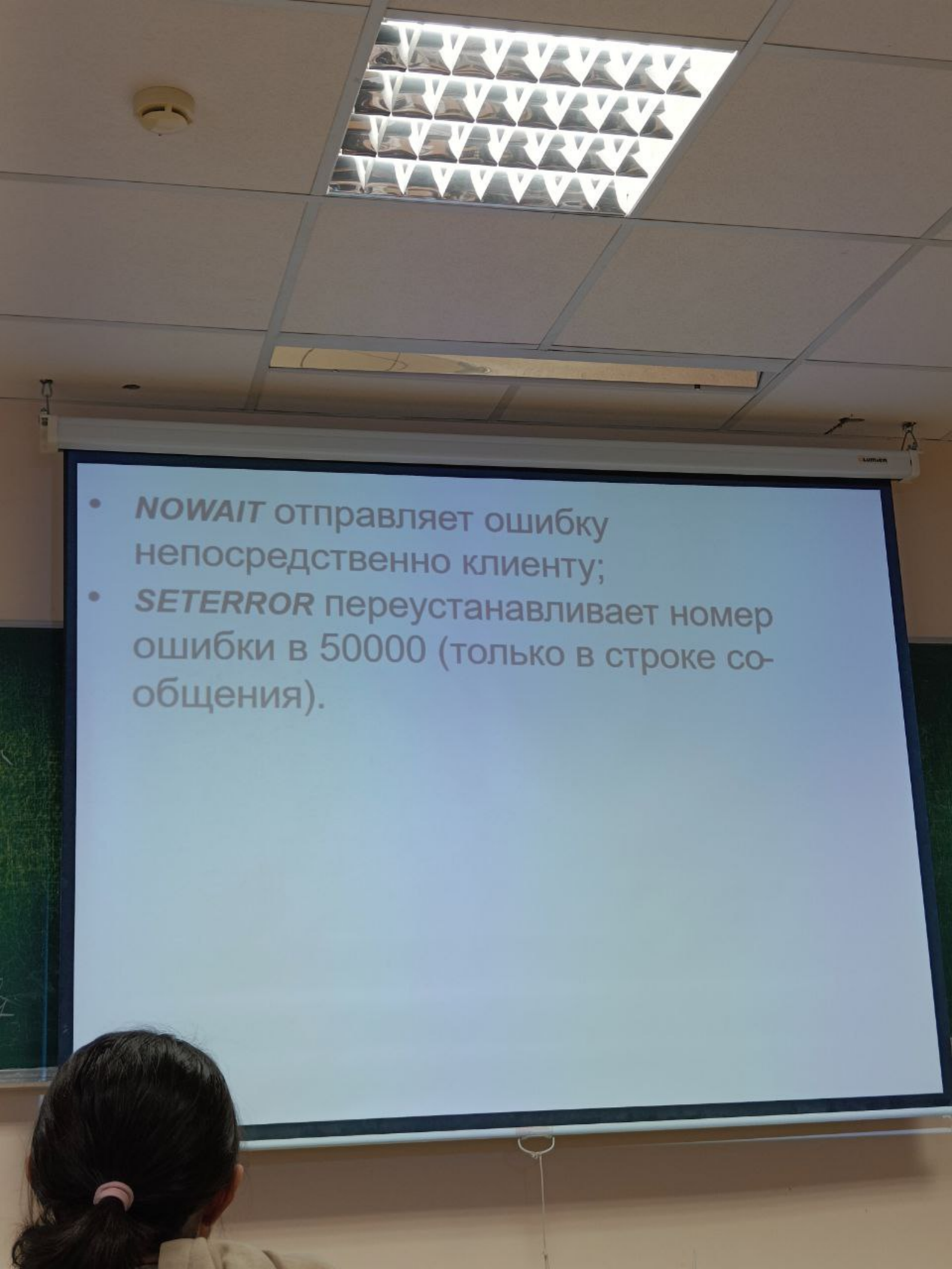

msg_str
Определенное пользователем сообщение с форматом, аналогичным формату функции printf из стандартной библиотеки языка C. Это сообщение об ошибке не должно содержать более 2 047 символов.

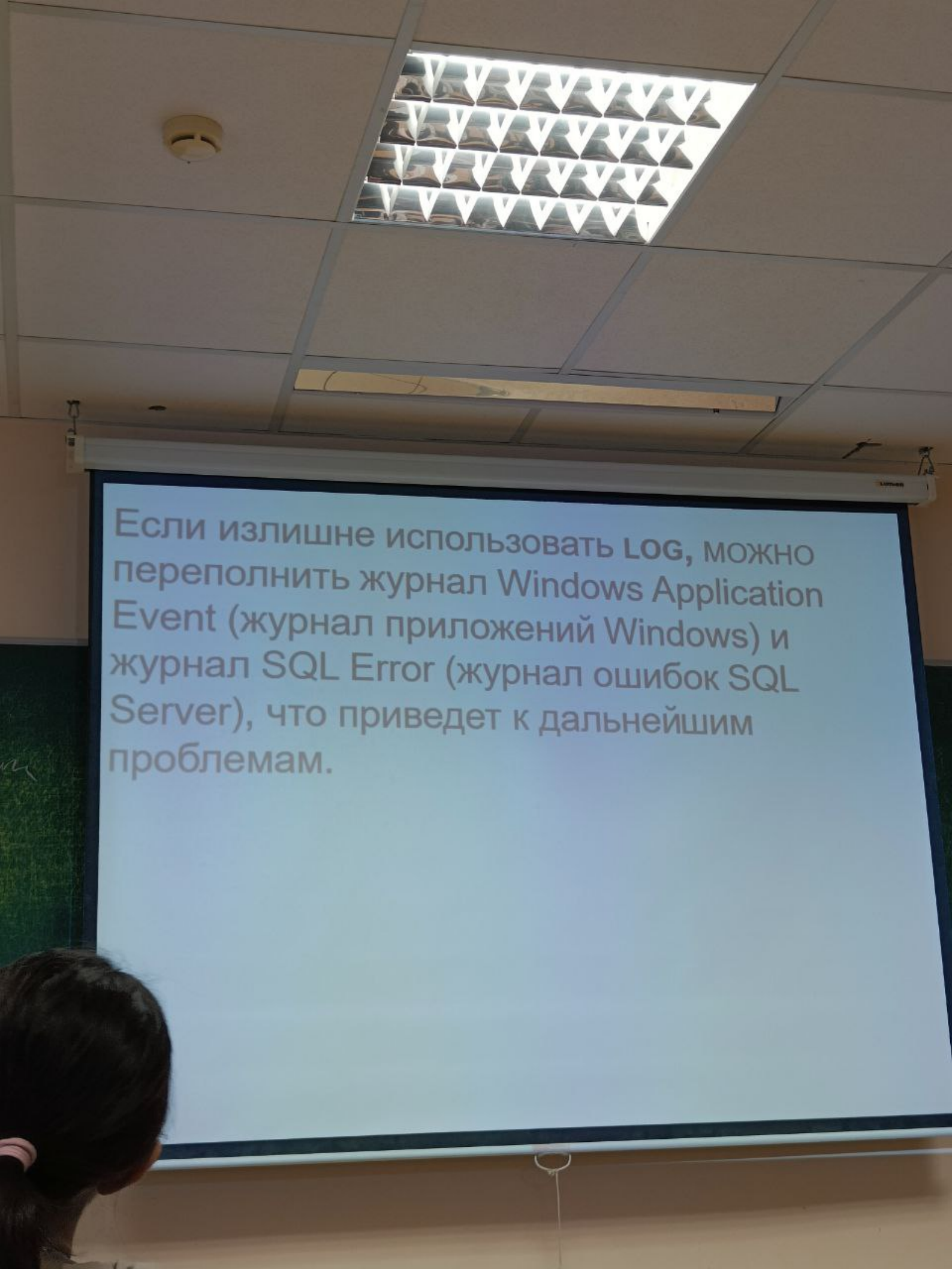

Спецификации преобразования имеют следующий формат:
% [[*flag*] [*width*] [. *precision*] [{h | l}]] *type*
В спецификации можно указать следующие параметры:

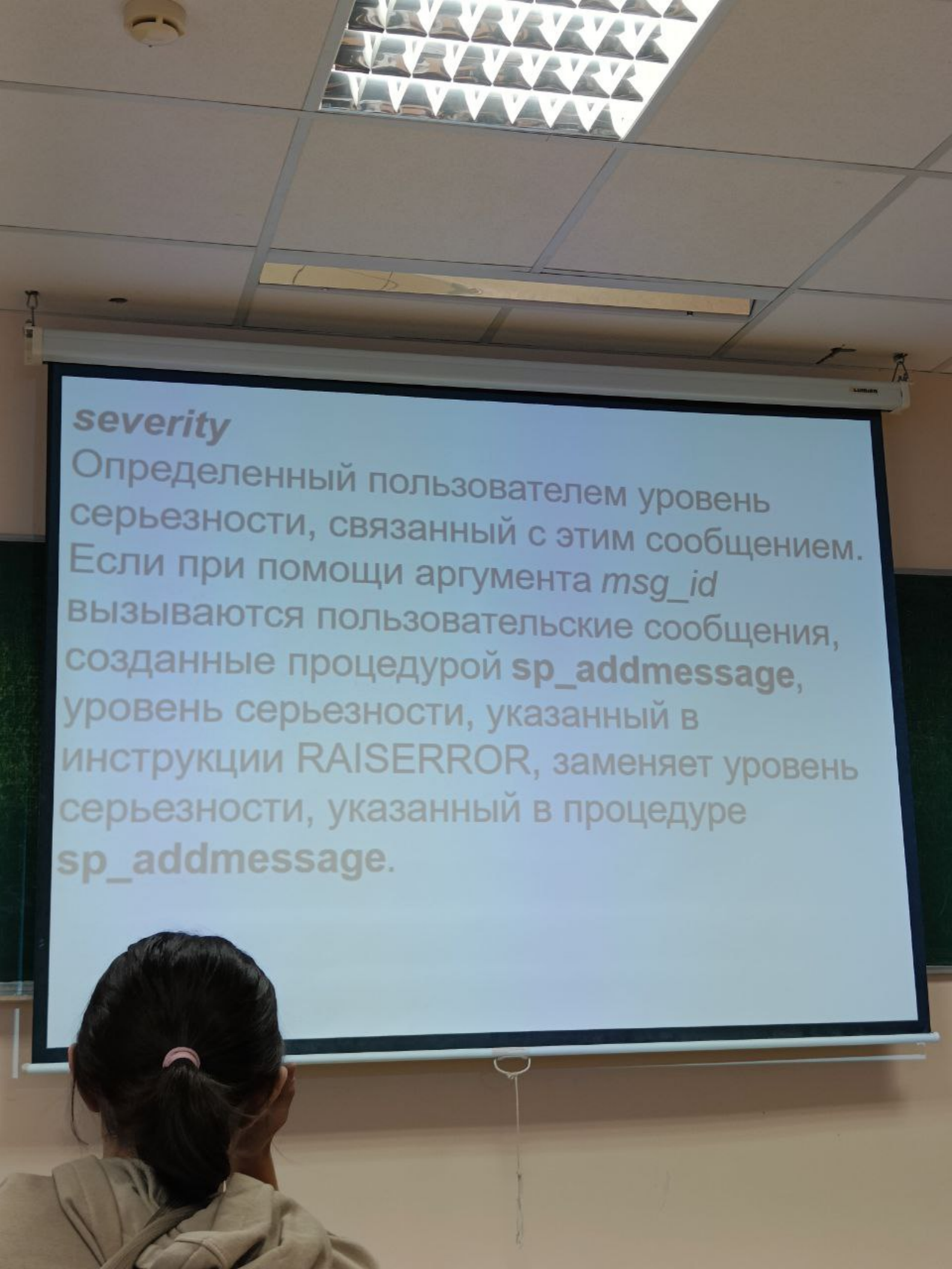

• пробел — значению предшествуют пробелы;
width — минимальная ширина поля, в которое помещается значение;
precision— максимальное количество символов, используемых из аргумента;

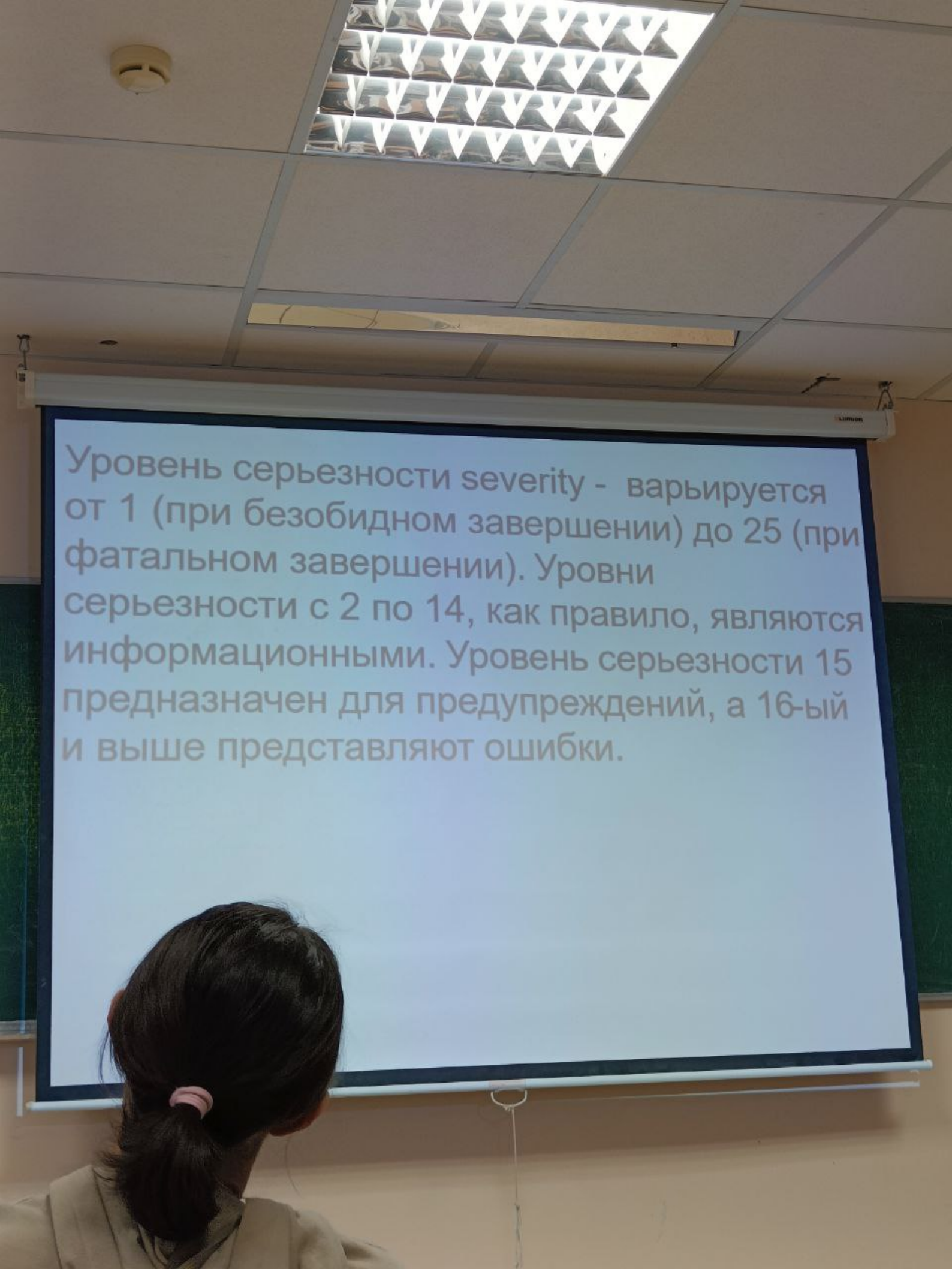

[h | l] type — типы символов
• d или i — целое со знаком;
• o — восьмеричное без знака;
• s — строка;
• u — целое без знака;
• x или X — шестнадцатеричное без знака.

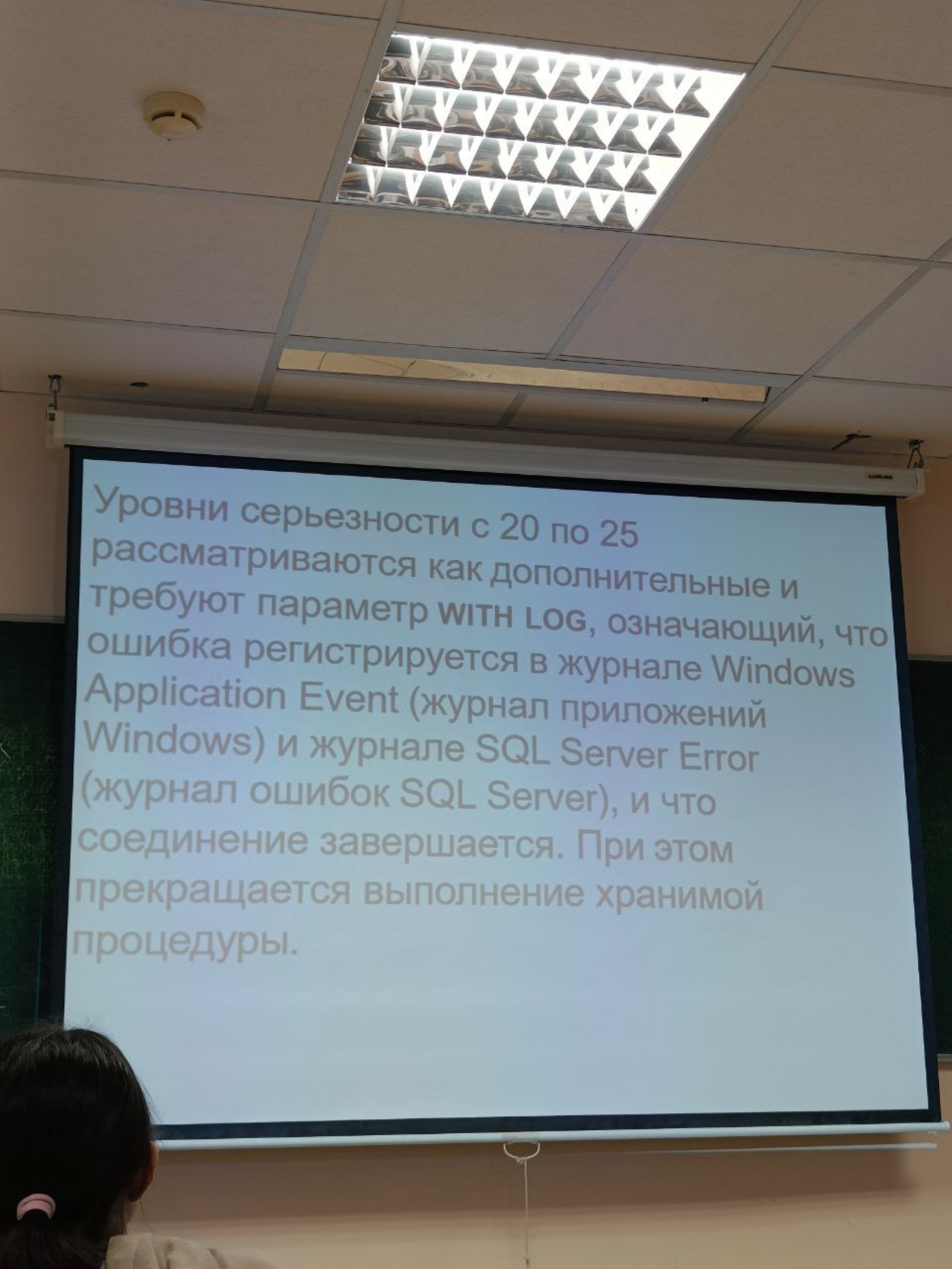

Кроме того, в конец сообщения RAISERROR можно поместить три параметра. Это параметры WITH:
• LOG помещает сообщение об ошибке в журнал ошибок и журнал приложений Windows;

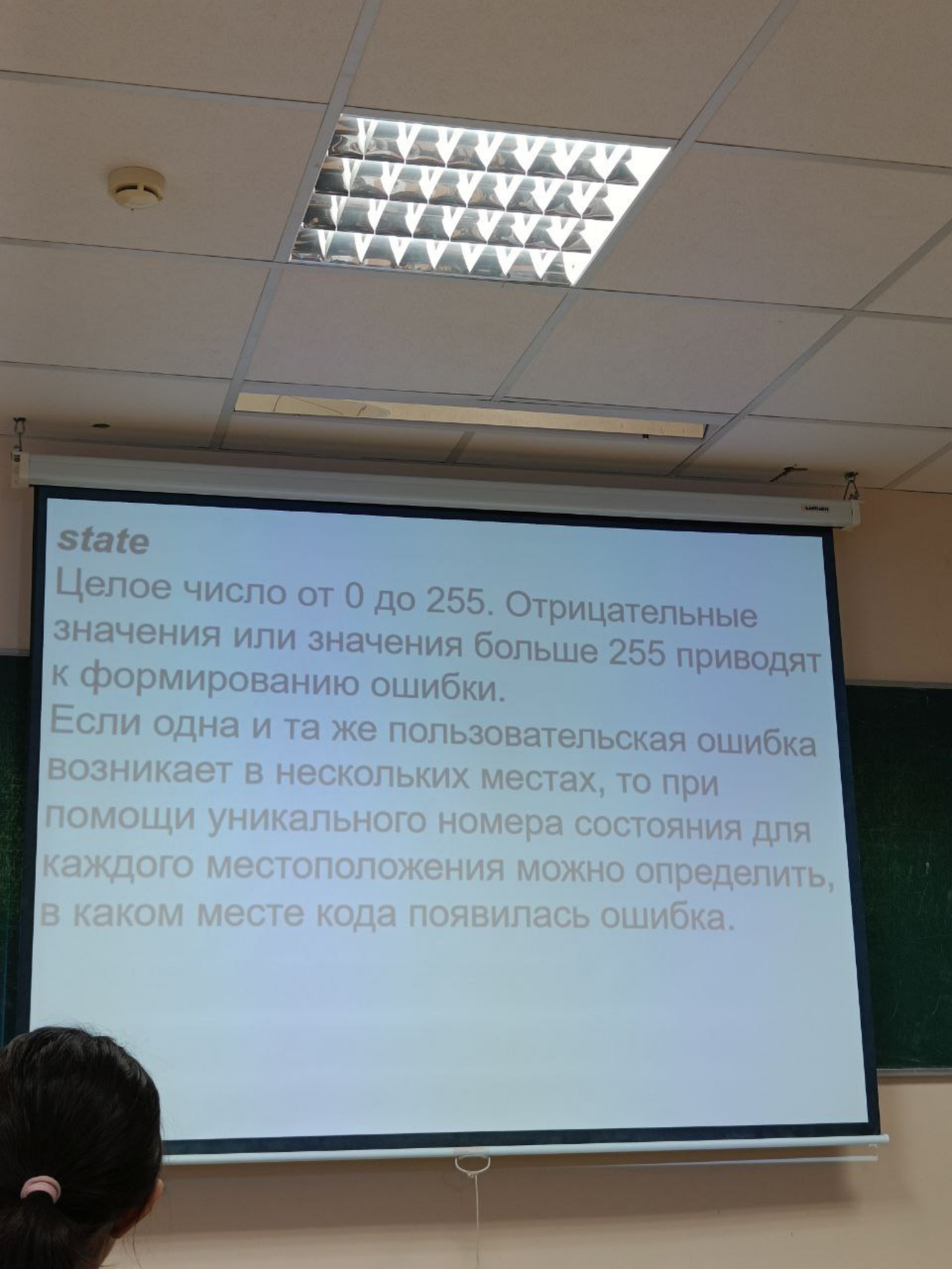

• NOWAIT отправляет ошибку непосредственно клиенту;
• SETERROR переустанавливает номер ошибки в 50000 (только в строке сообщения).

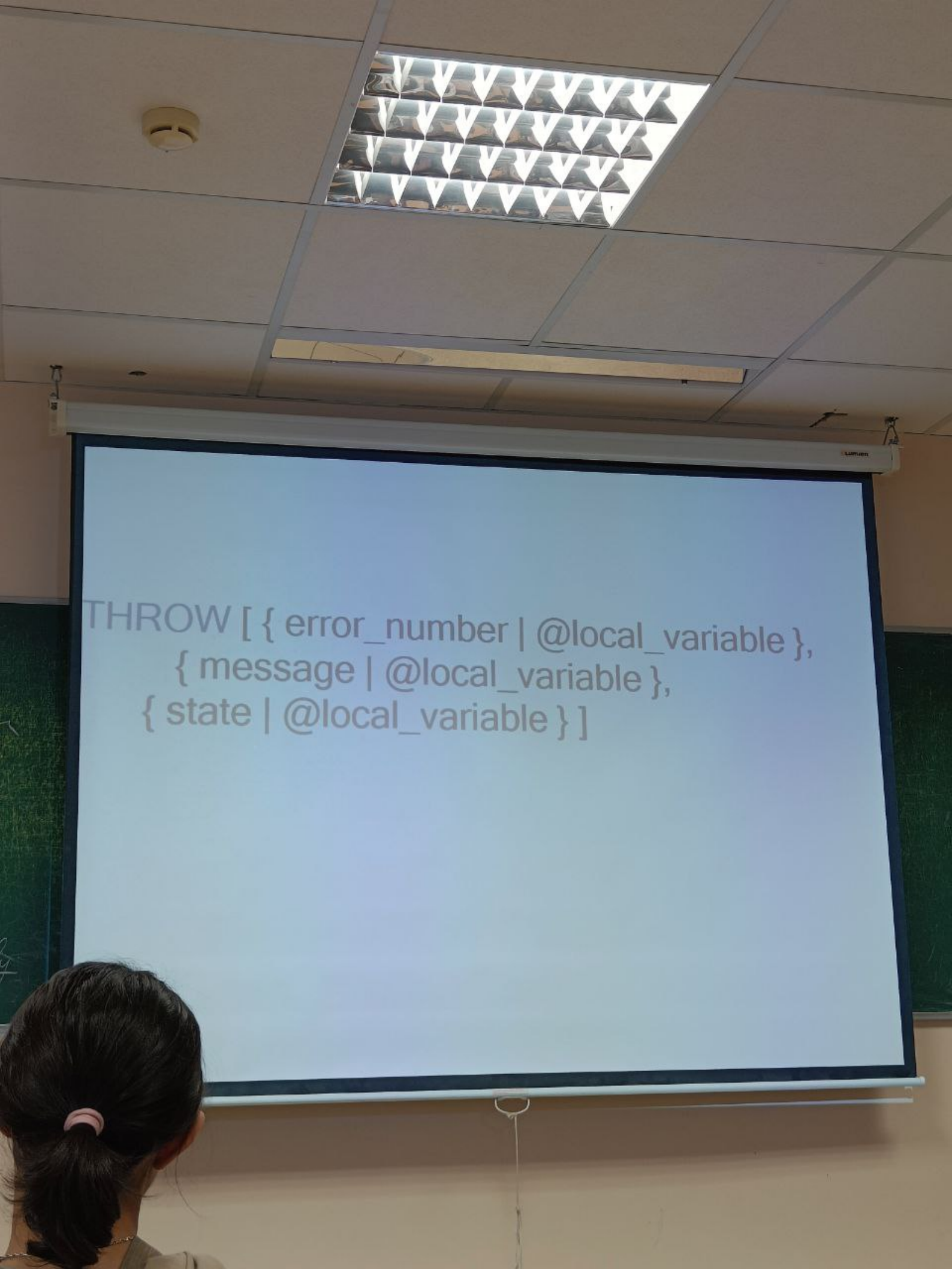

Если излишне использовать LOG, можно переполнить журнал Windows Application Event (журнал приложений Windows) и журнал SQL Error (журнал ошибок SQL Server), что приведет к дальнейшим проблемам.

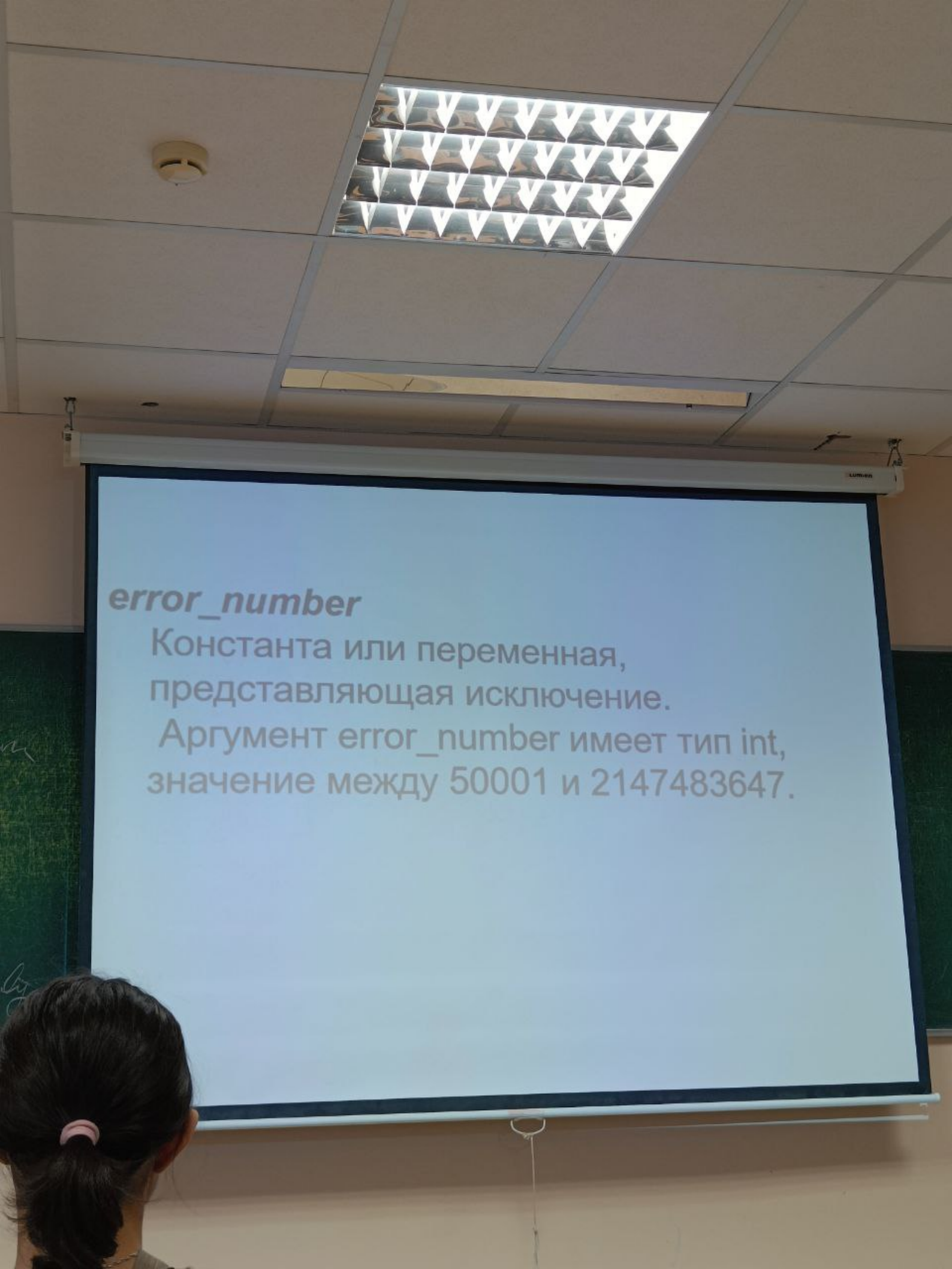

severity
Определенный пользователем уровень серьезности, связанный с этим сообщением. Если при помощи аргумента *msg_id* вызываются пользовательские сообщения, созданные процедурой **sp_addmessage**, уровень серьезности, указанный в инструкции RAISERROR, заменяет уровень серьезности, указанный в процедуре **sp_addmessage**.

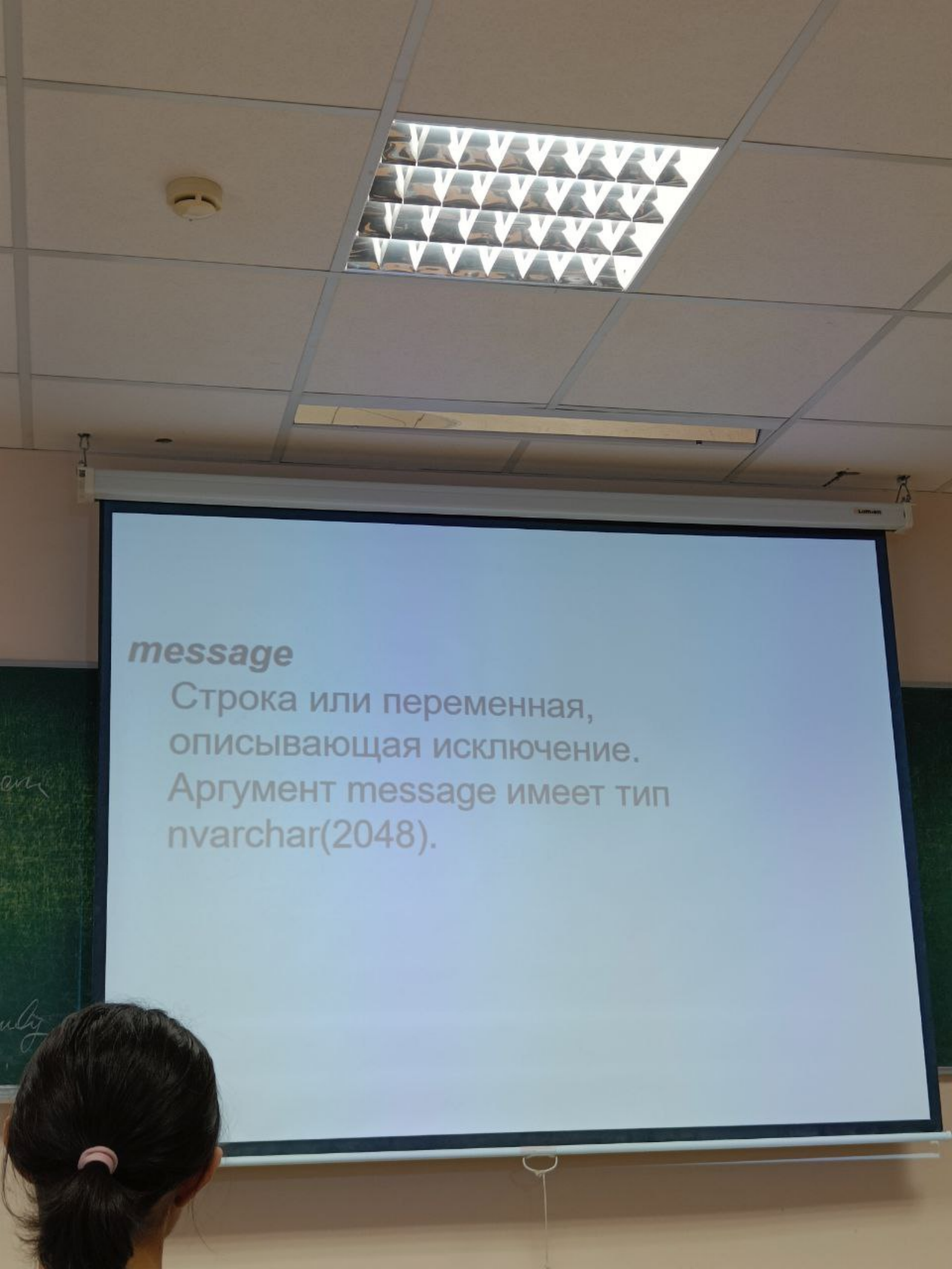

Уровень серьезности severity - варьируется от 1 (при безобидном завершении) до 25 (при фатальном завершении). Уровни серьезности с 2 по 14, как правило, являются информационными. Уровень серьезности 15 предназначен для предупреждений, а 16-ый и выше представляют ошибки.

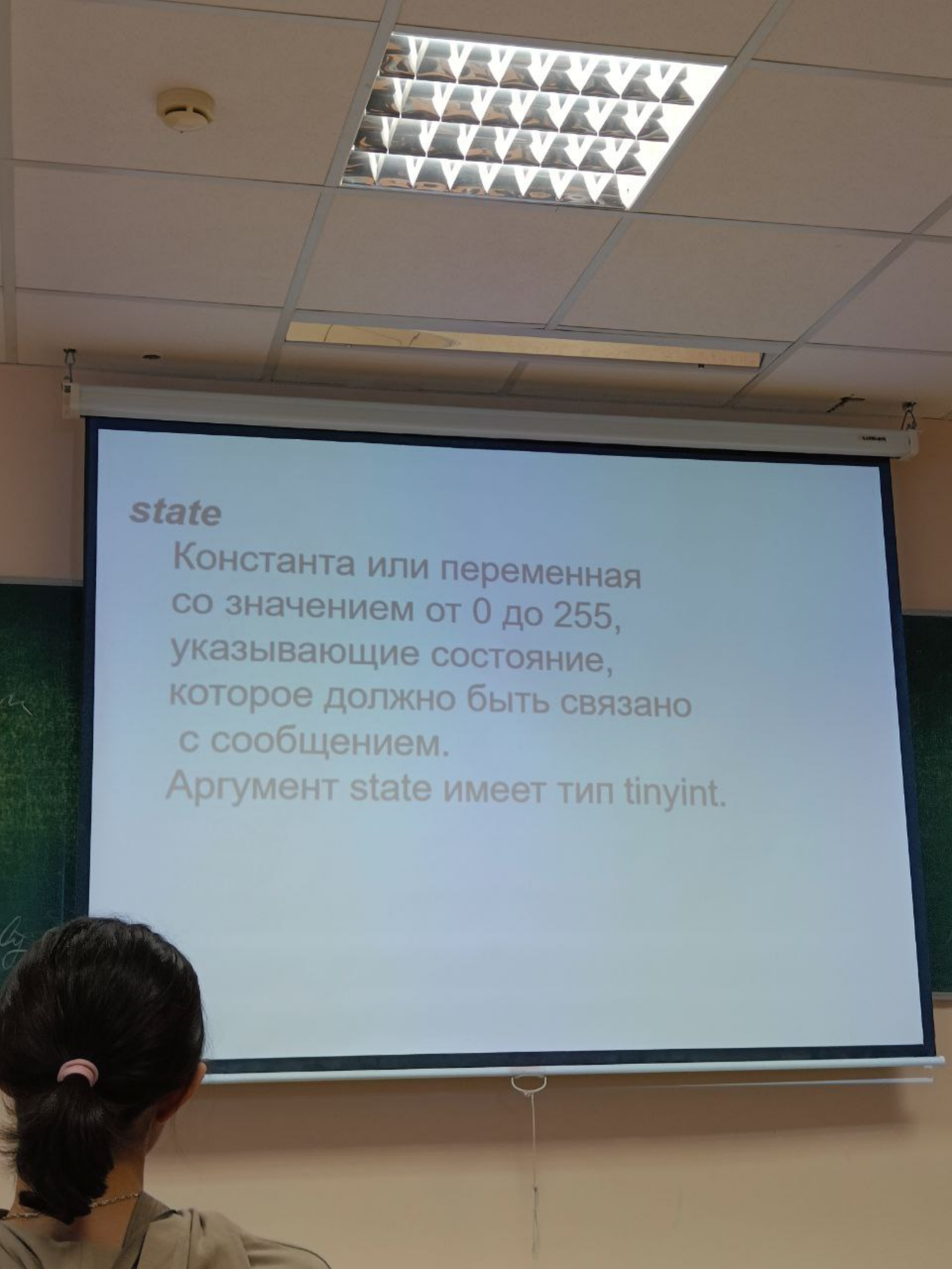

Уровни серьезности с 20 по 25 рассматриваются как дополнительные и требуют параметр WITH LOG, означающий, что ошибка регистрируется в журнале Windows Application Event (журнал приложений Windows) и журнале SQL Server Error (журнал ошибок SQL Server), и что соединение завершается. При этом прекращается выполнение хранимой процедуры.

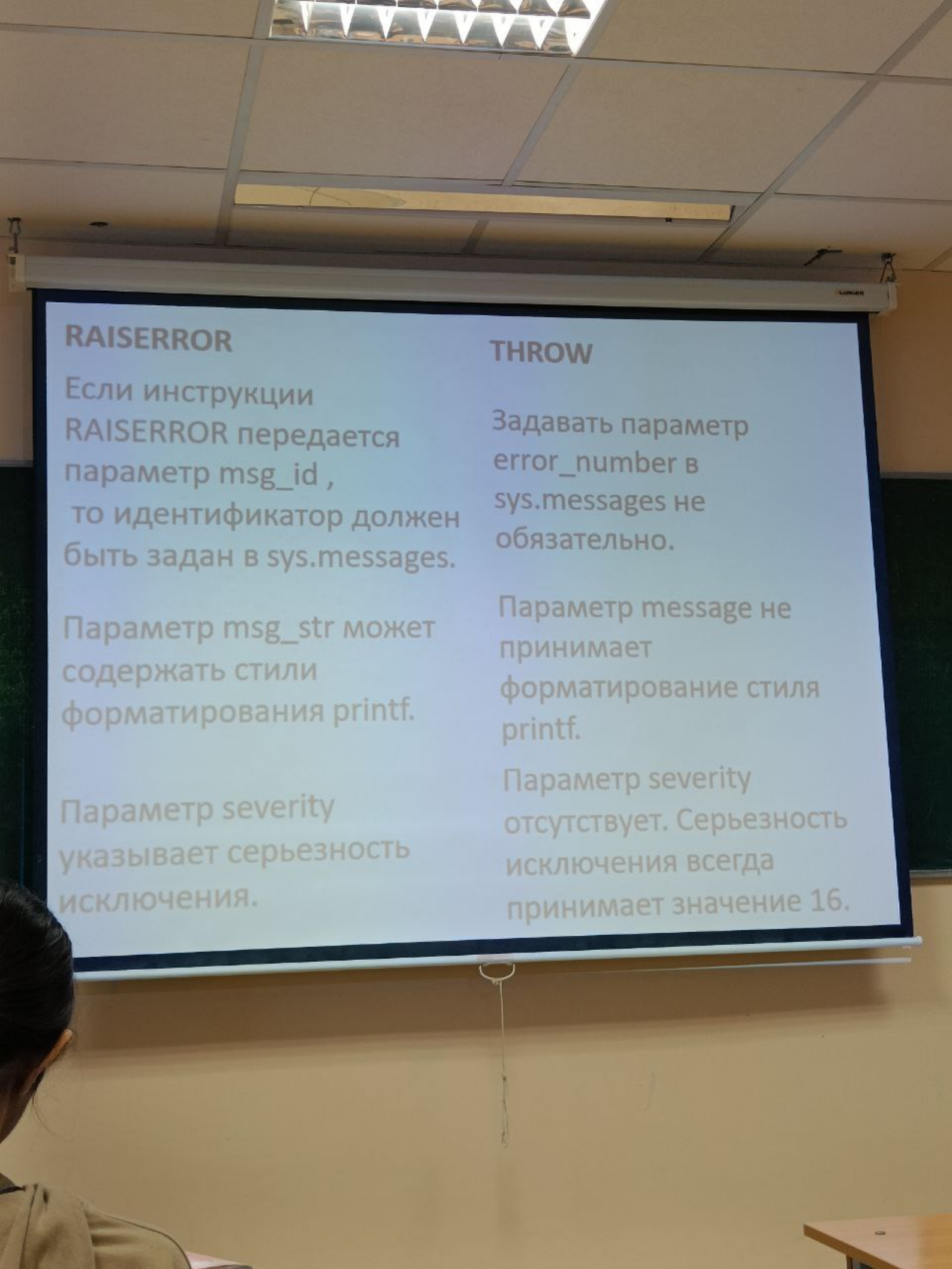

state
Целое число от 0 до 255. Отрицательные значения или значения больше 255 приводят к формированию ошибки.
Если одна и та же пользовательская ошибка возникает в нескольких местах, то при помощи уникального номера состояния для каждого местоположения можно определить, в каком месте кода появилась ошибка.

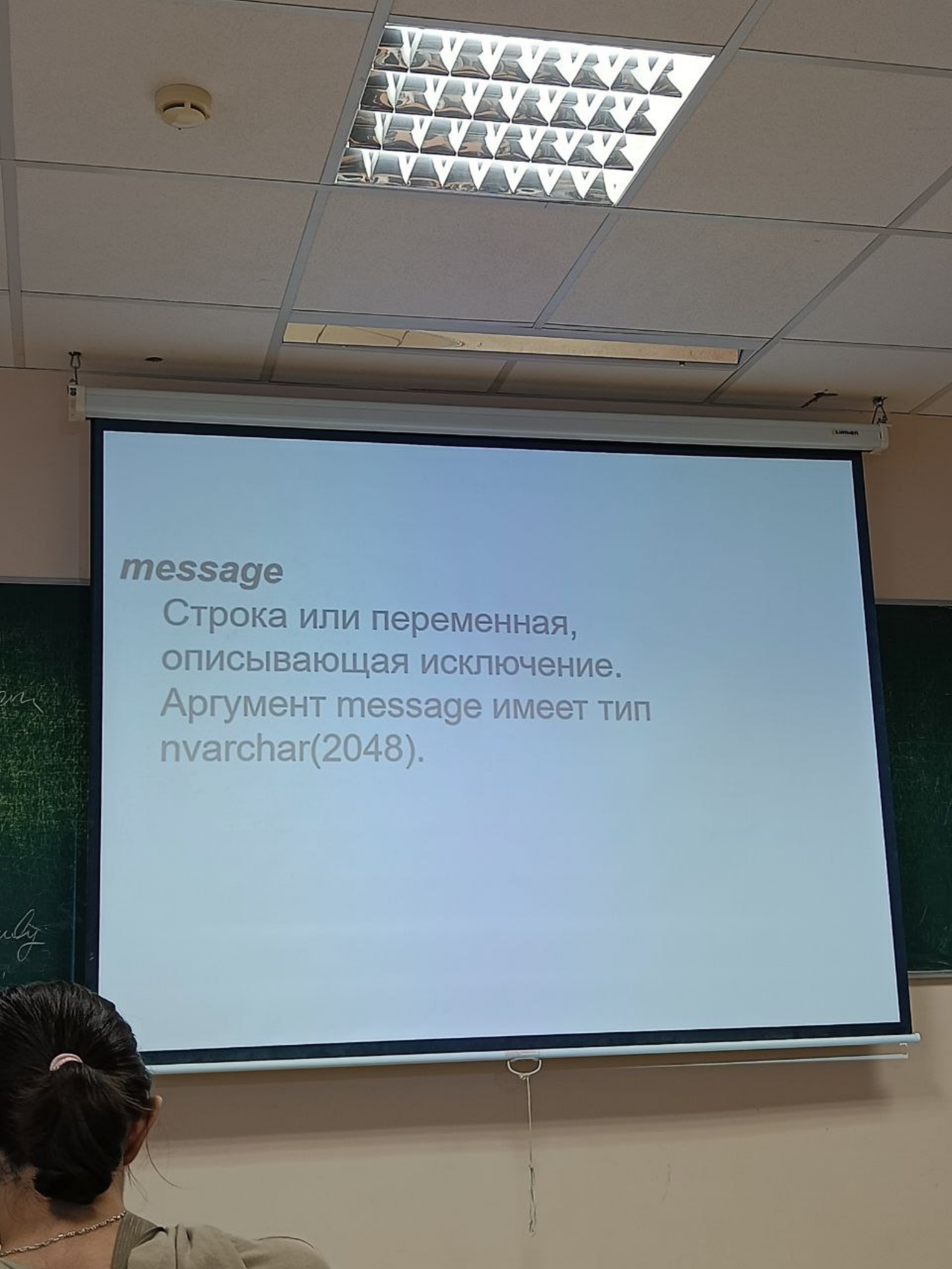

THROW [{ error_number | @local_variable },
{ message | @local_variable },
{ state | @local_variable }]

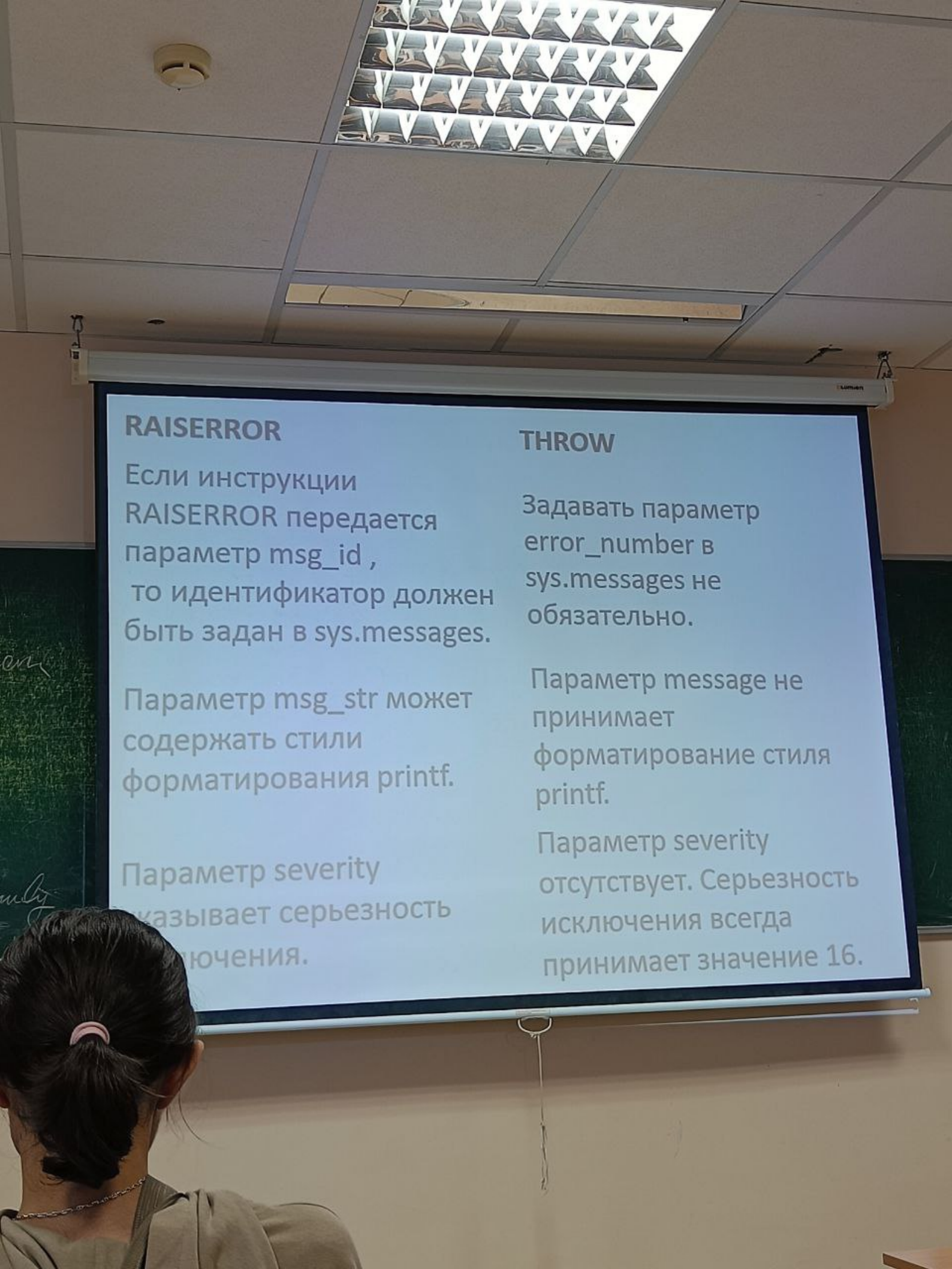

error_number
Константа или переменная,
представляющая исключение.
Аргумент error_number имеет тип int,
значение между 50001 и 2147483647.

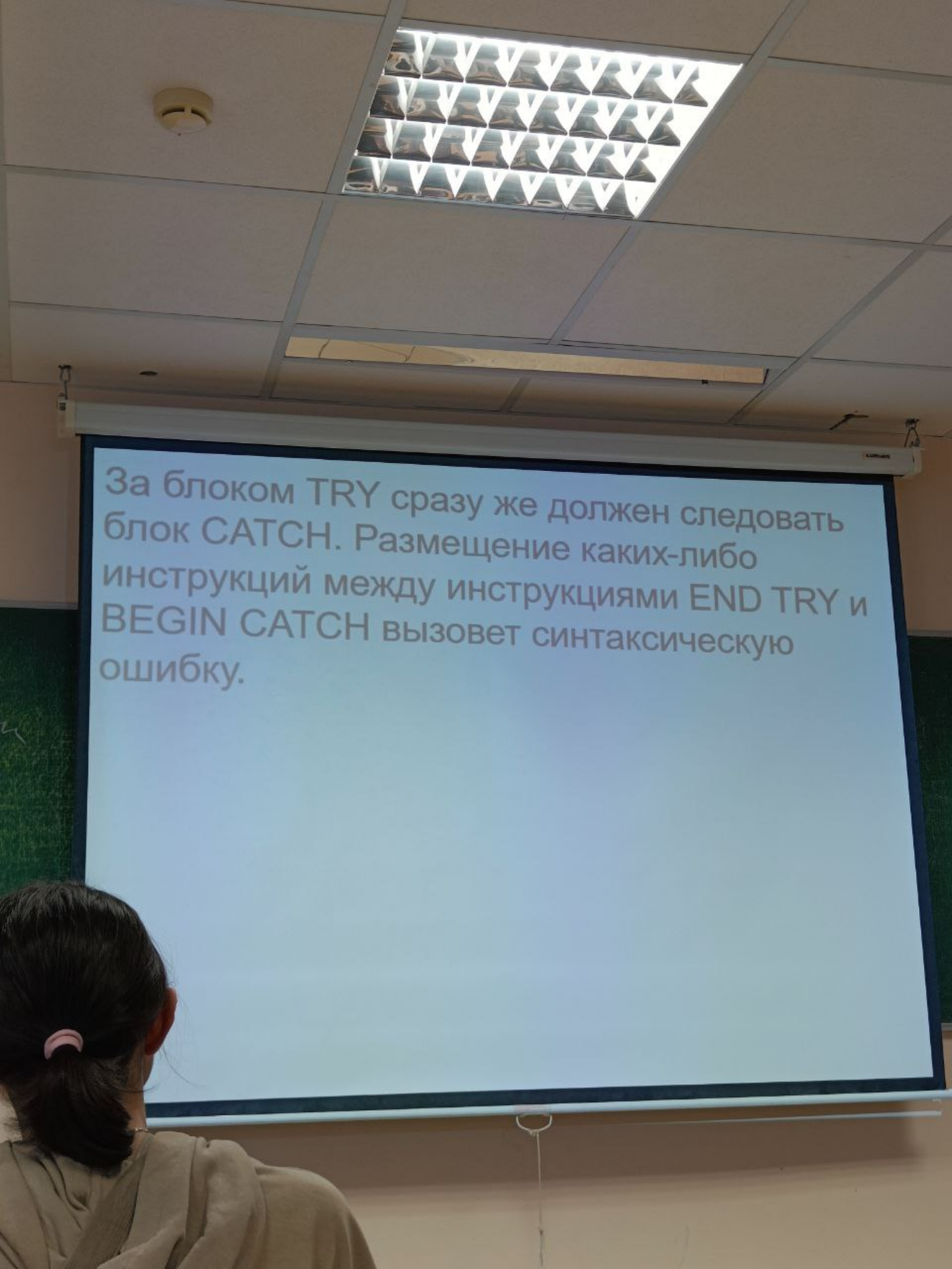

message
Строка или переменная, описывающая исключение. Аргумент message имеет тип nvarchar(2048).

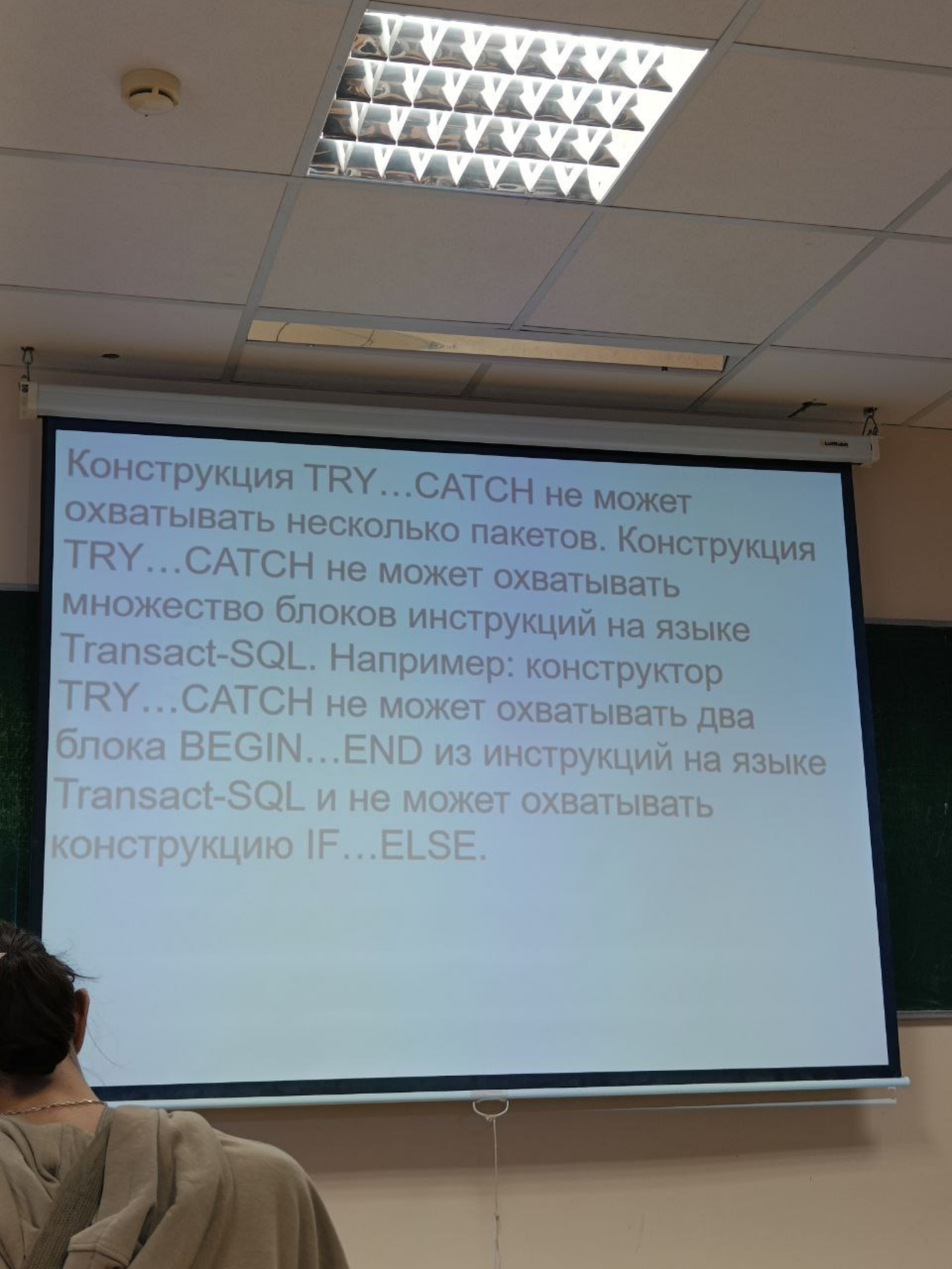

state
Константа или переменная
со значением от 0 до 255,
указывающие состояние,
которое должно быть связано
с сообщением.
Аргумент state имеет тип tinyint.

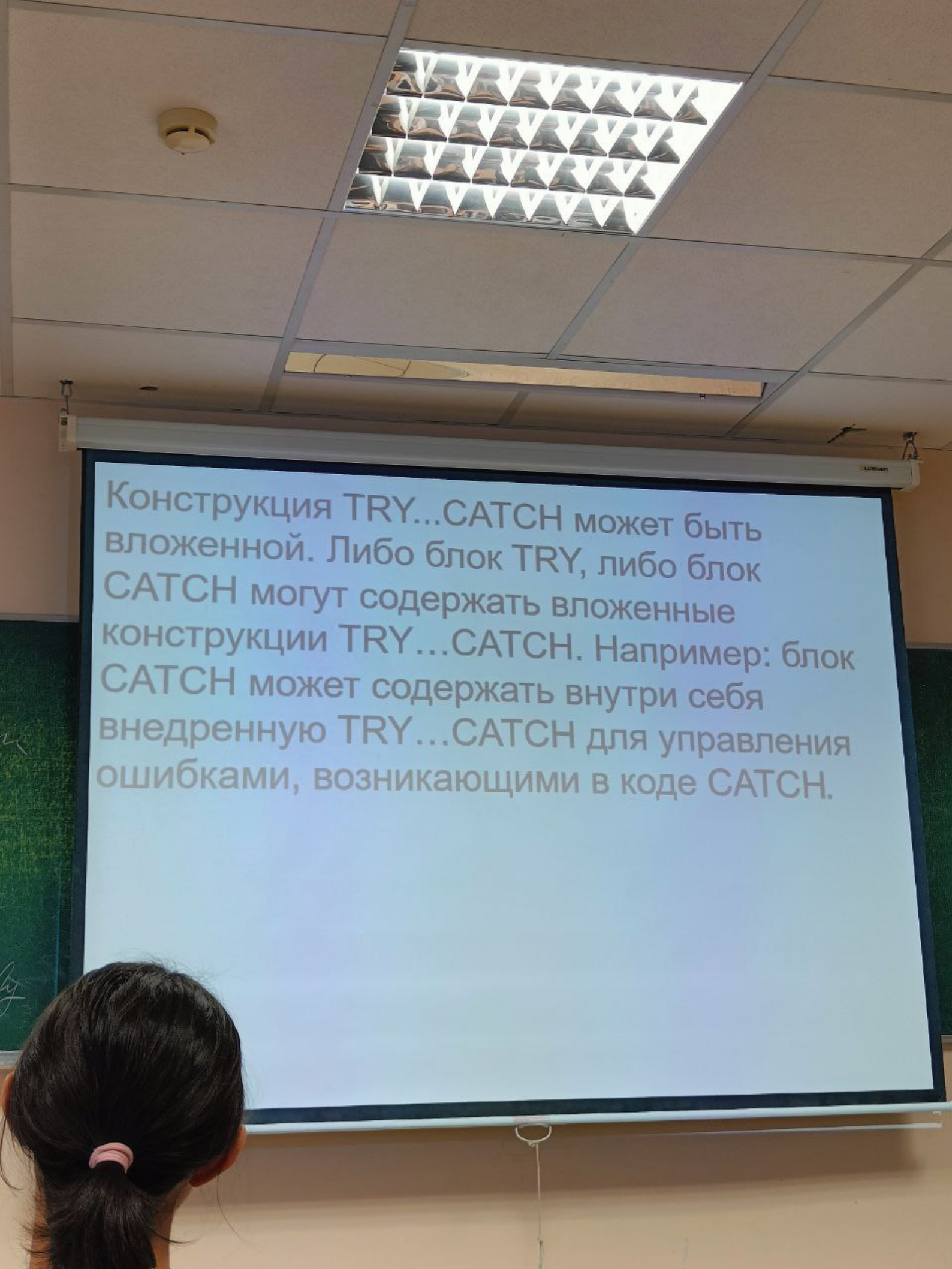

RAISERROR
Если инструкции RAISERROR передается параметр msg_id , то идентификатор должен быть задан в sys.messages.
Параметр msg_str может содержать стили форматирования printf.
Параметр severity указывает серьезность исключения.
THROW
Задавать параметр error_number в sys.messages не обязательно.
Параметр message не принимает форматирование стиля printf.
Параметр severity отсутствует. Серьезность исключения всегда принимает значение 16.

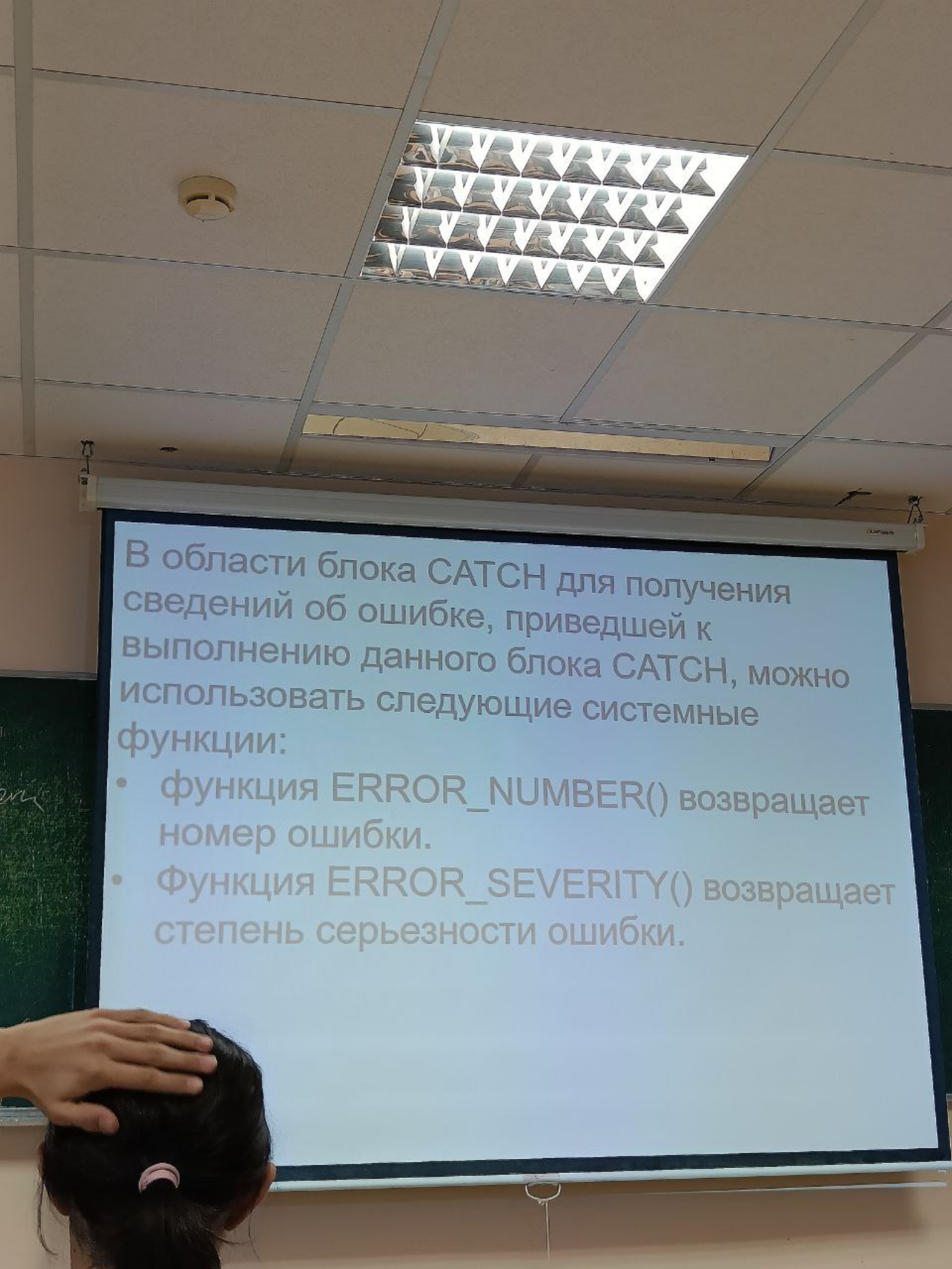

message
Строка или переменная, описывающая исключение. Аргумент message имеет тип nvarchar(2048).

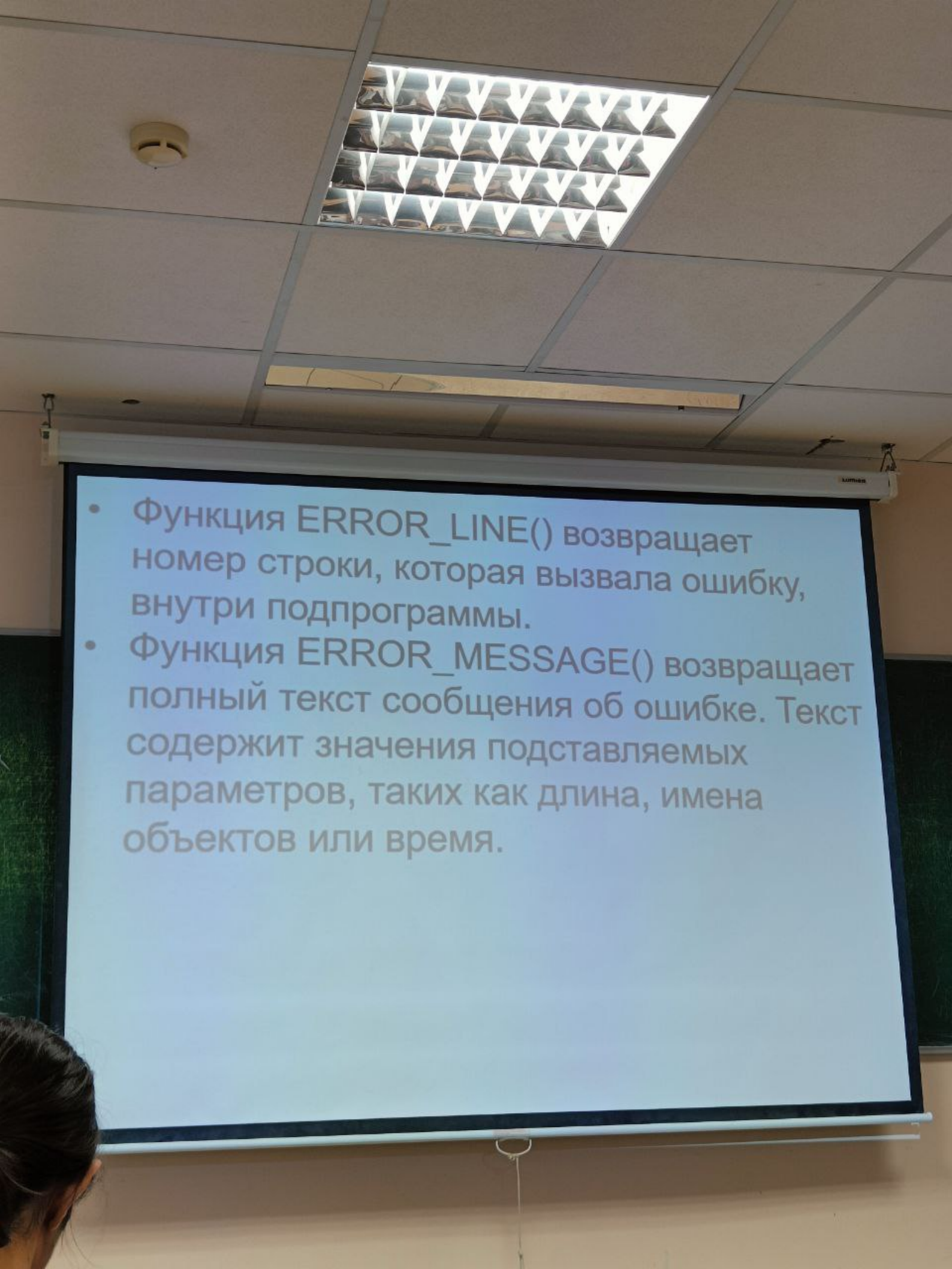

RAISERROR
Если инструкции RAISERROR передается параметр msg_id , то идентификатор должен быть задан в sys.messages.
Параметр msg_str может содержать стили форматирования printf.
Параметр severity
казывает серьезность
ючения.
THROW
Задавать параметр error_number в sys.messages не обязательно.
Параметр message не принимает форматирование стиля printf.
Параметр severity отсутствует. Серьезность исключения всегда принимает значение 16.


За блоком TRY сразу же должен следовать блок CATCH. Размещение каких-либо инструкций между инструкциями END TRY и BEGIN CATCH вызовет синтаксическую ошибку.


Конструкция TRY…CATCH не может охватывать несколько пакетов. Конструкция TRY…CATCH не может охватывать множество блоков инструкций на языке Transact-SQL. Например: конструктор TRY…CATCH не может охватывать два блока BEGIN…END из инструкций на языке Transact-SQL и не может охватывать конструкцию IF…ELSE.


Конструкция TRY...CATCH может быть вложенной. Либо блок TRY, либо блок CATCH могут содержать вложенные конструкции TRY…CATCH. Например: блок CATCH может содержать внутри себя внедренную TRY…CATCH для управления ошибками, возникающими в коде CATCH.


В области блока CATCH для получения сведений об ошибке, приведшей к выполнению данного блока CATCH, можно использовать следующие системные функции:
• функция ERROR_NUMBER() возвращает номер ошибки.
• Функция ERROR_SEVERITY() возвращает степень серьезности ошибки.


• Функция ERROR_LINE() возвращает номер строки, которая вызвала ошибку, внутри подпрограммы.
• Функция ERROR_MESSAGE() возвращает полный текст сообщения об ошибке. Текст содержит значения подставляемых параметров, таких как длина, имена объектов или время.