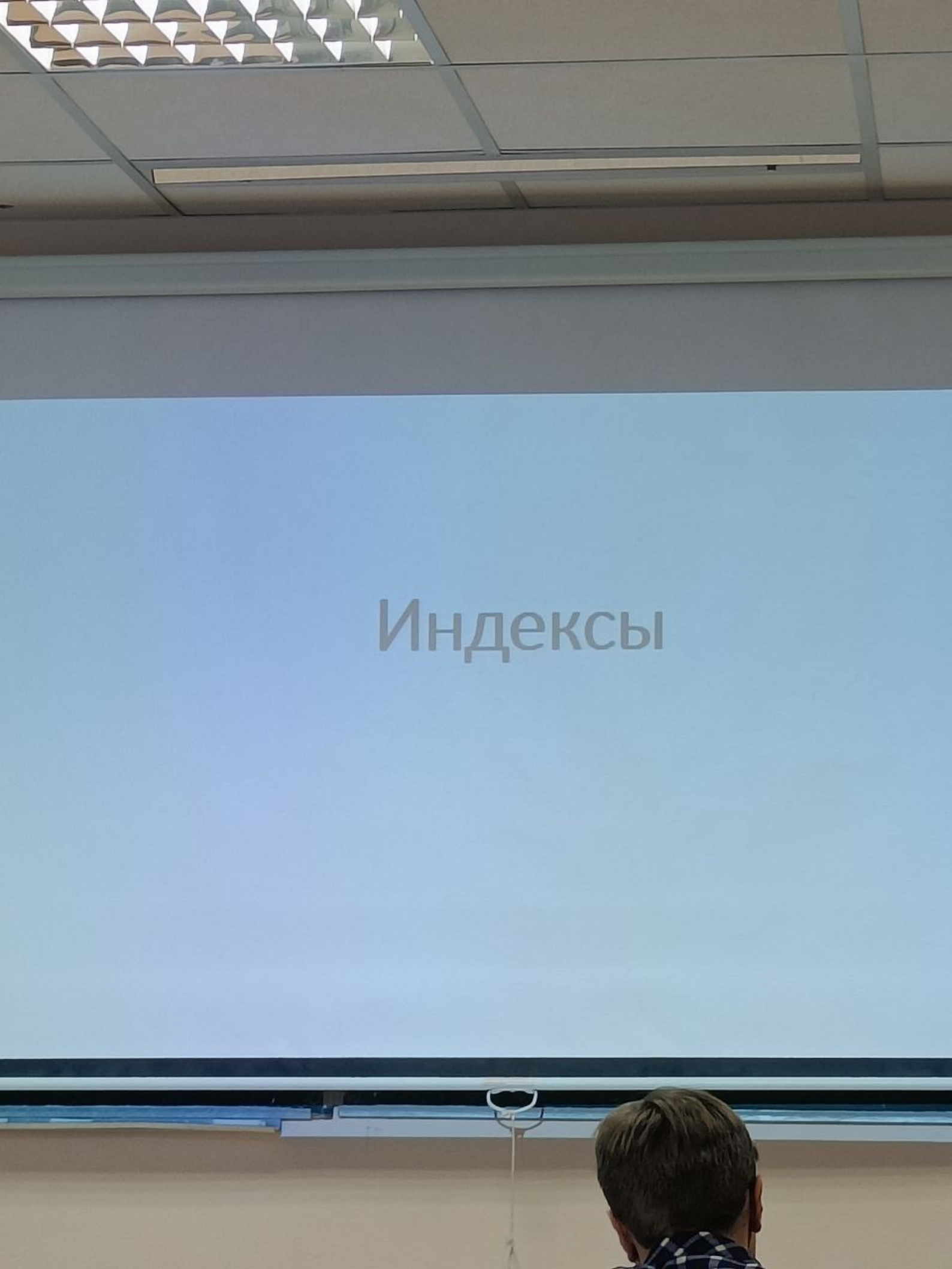

Индексы

Индекс в SQL Server определяется таким образом, чтобы можно было обнаружить строки, необходимые для выполнения запроса к базе данных.

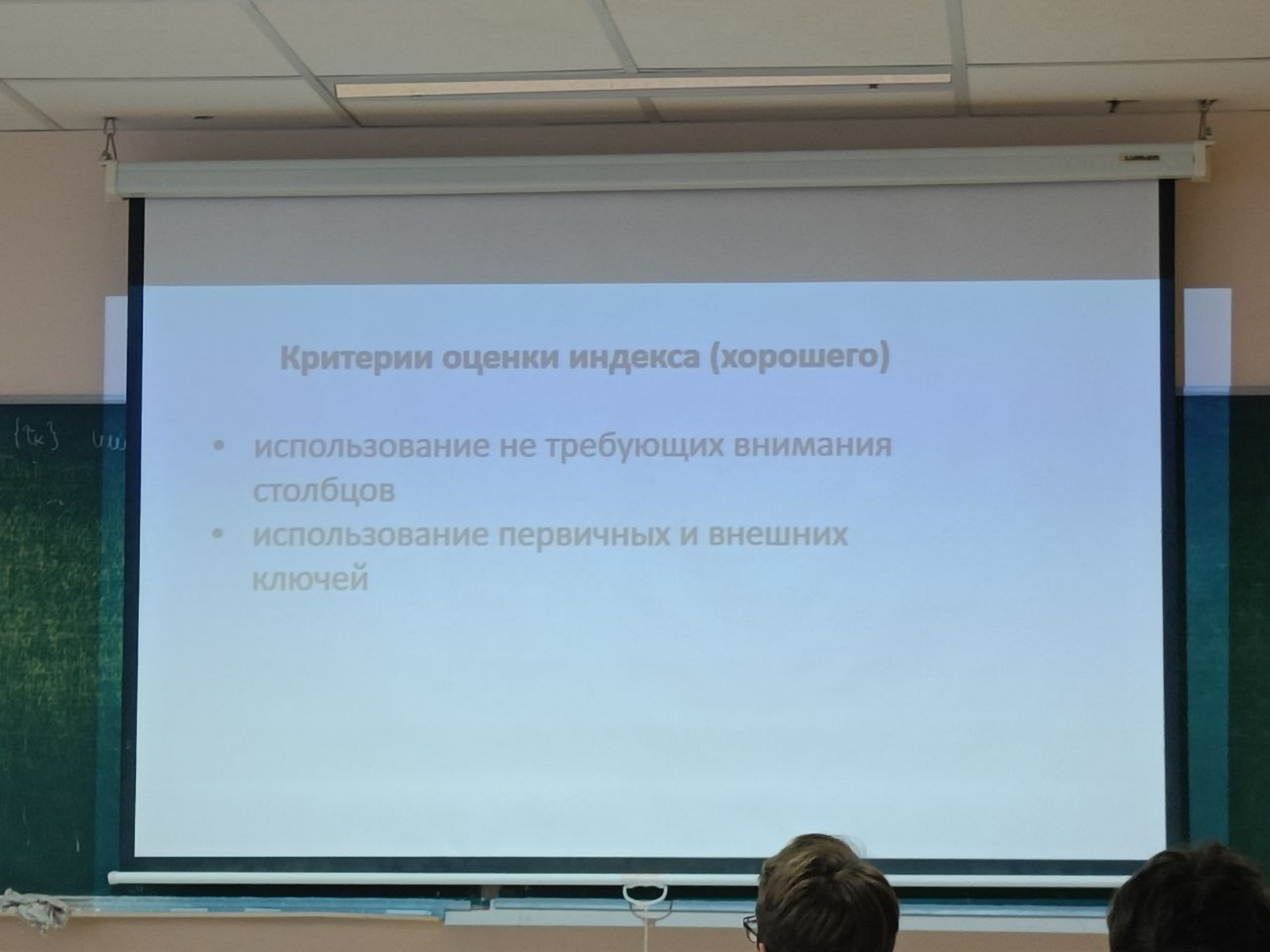
Простой индекс (simple index).

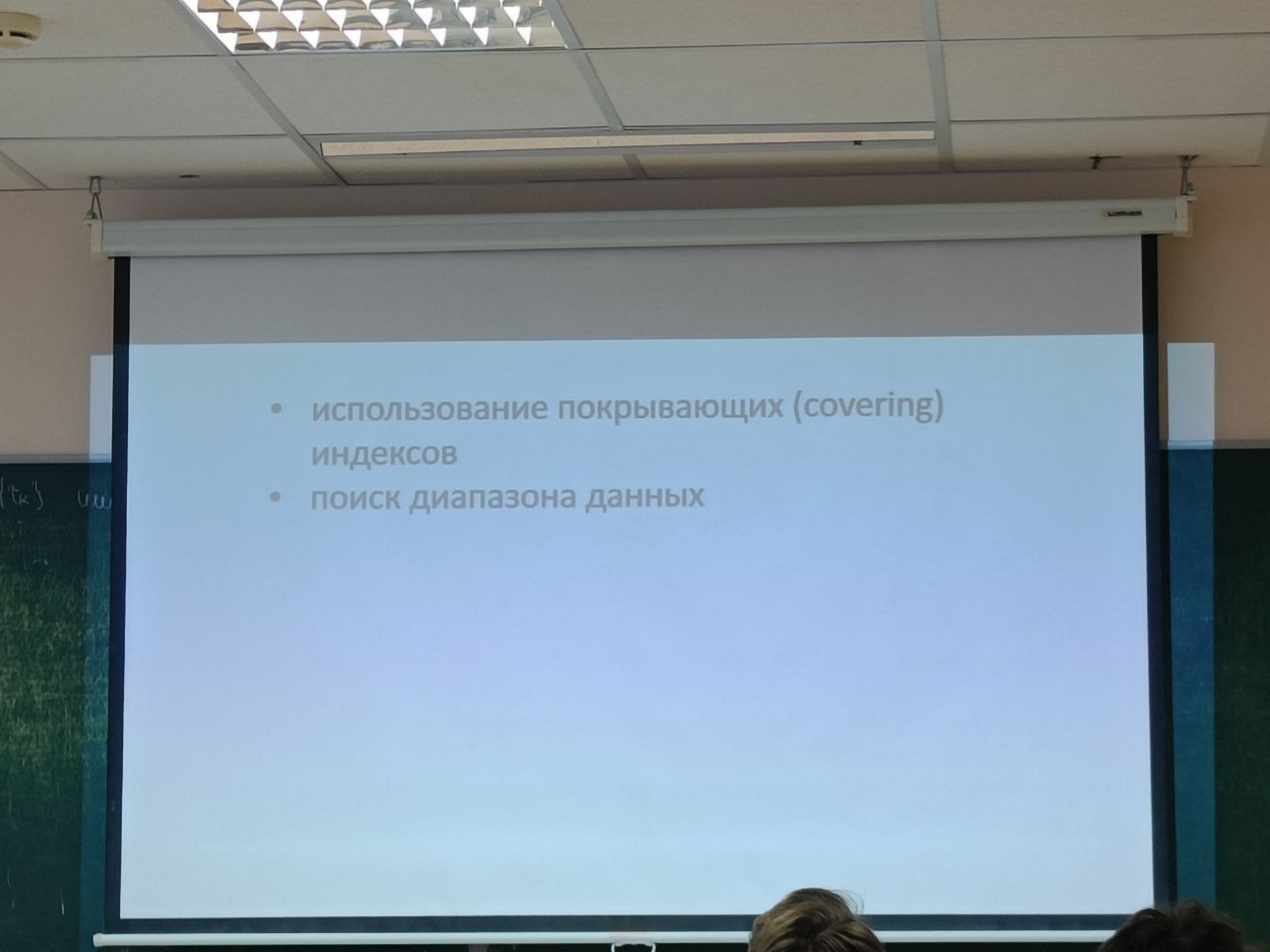
Комбинированный индекс (compound index).

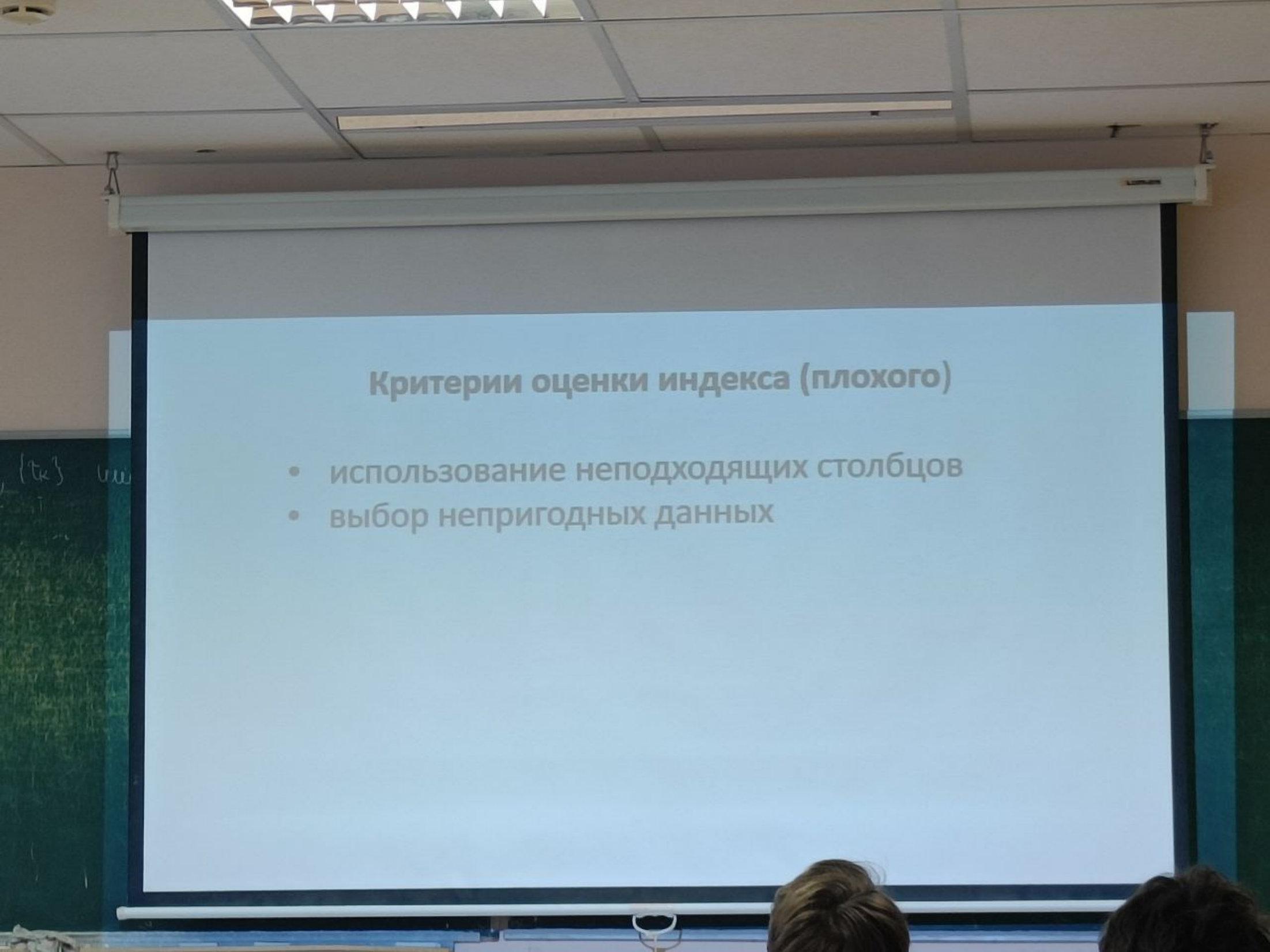
Типы индексов:

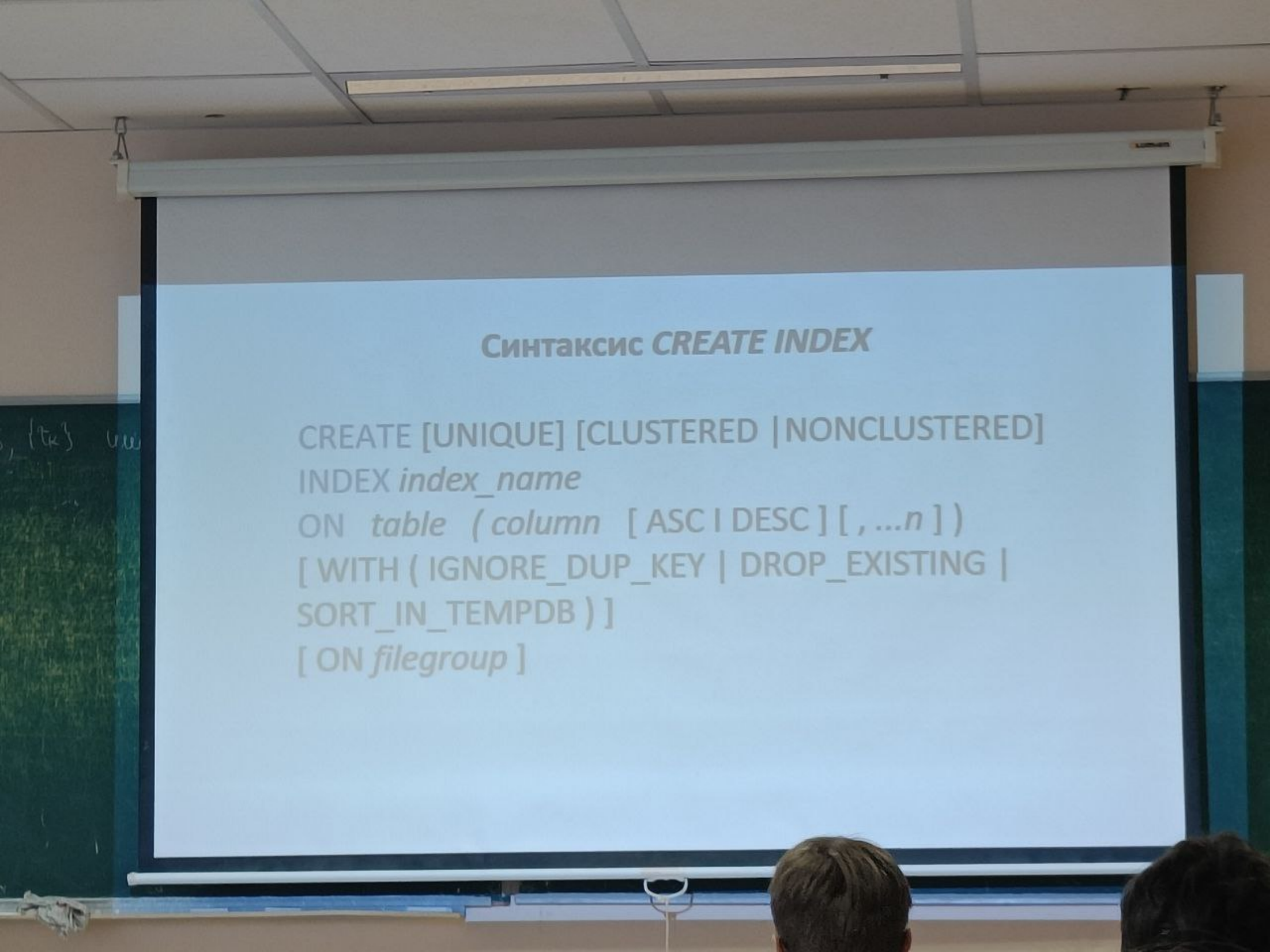
- кластеризованные (*только один на таблицу*)
- некластеризованные
- первичные и вторичные индексы XML

Уникальный (unique) и
неуникальный (nonunique) индексы


Критерии оценки индекса (хорошего)
• использование не требующих внимания столбцов
• использование первичных и внешних ключей


• использование покрывающих (covering) индексов
• поиск диапазона данных

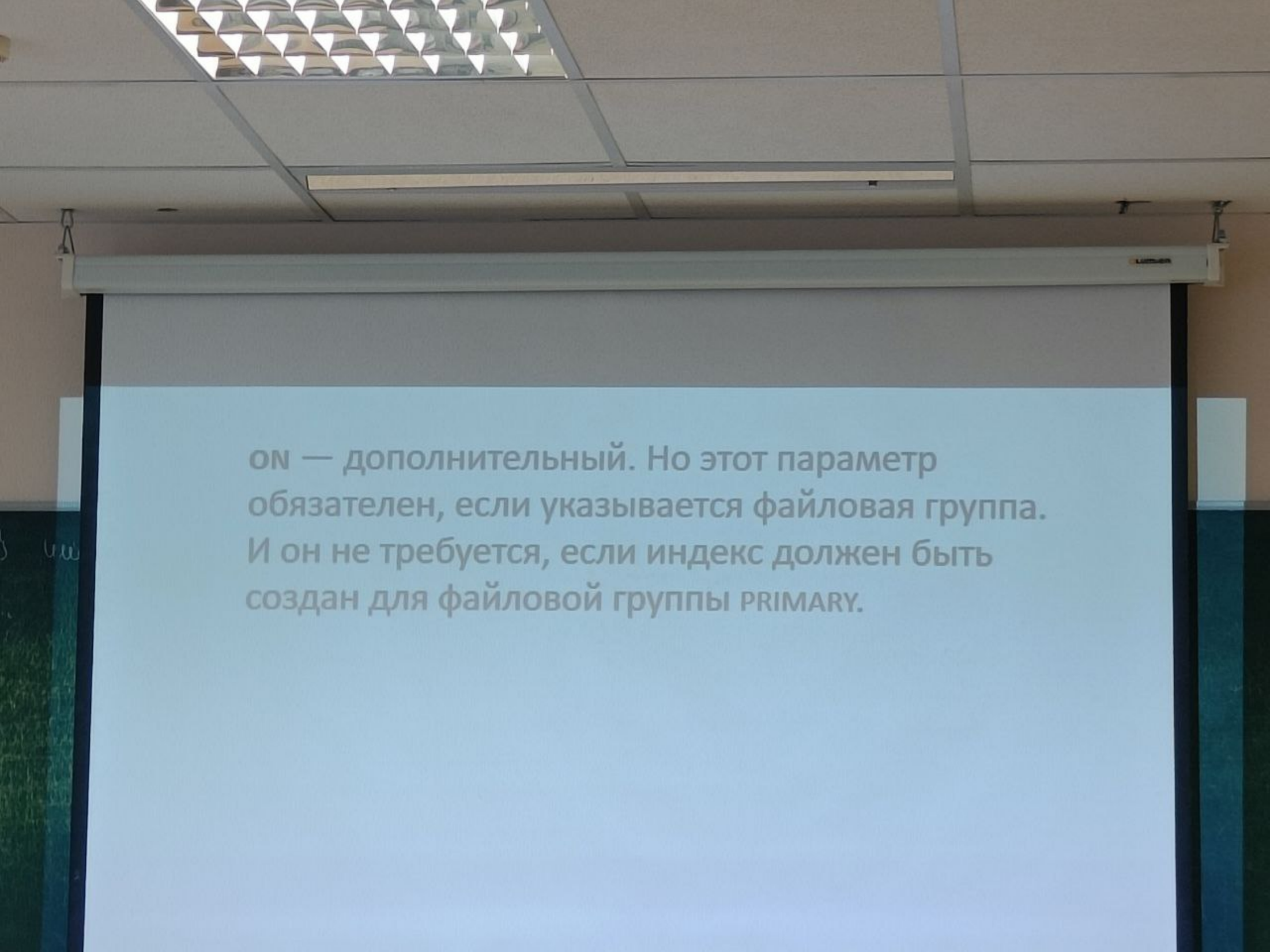

Критерии оценки индекса (плохого)
• использование неподходящих столбцов
• выбор непригодных данных

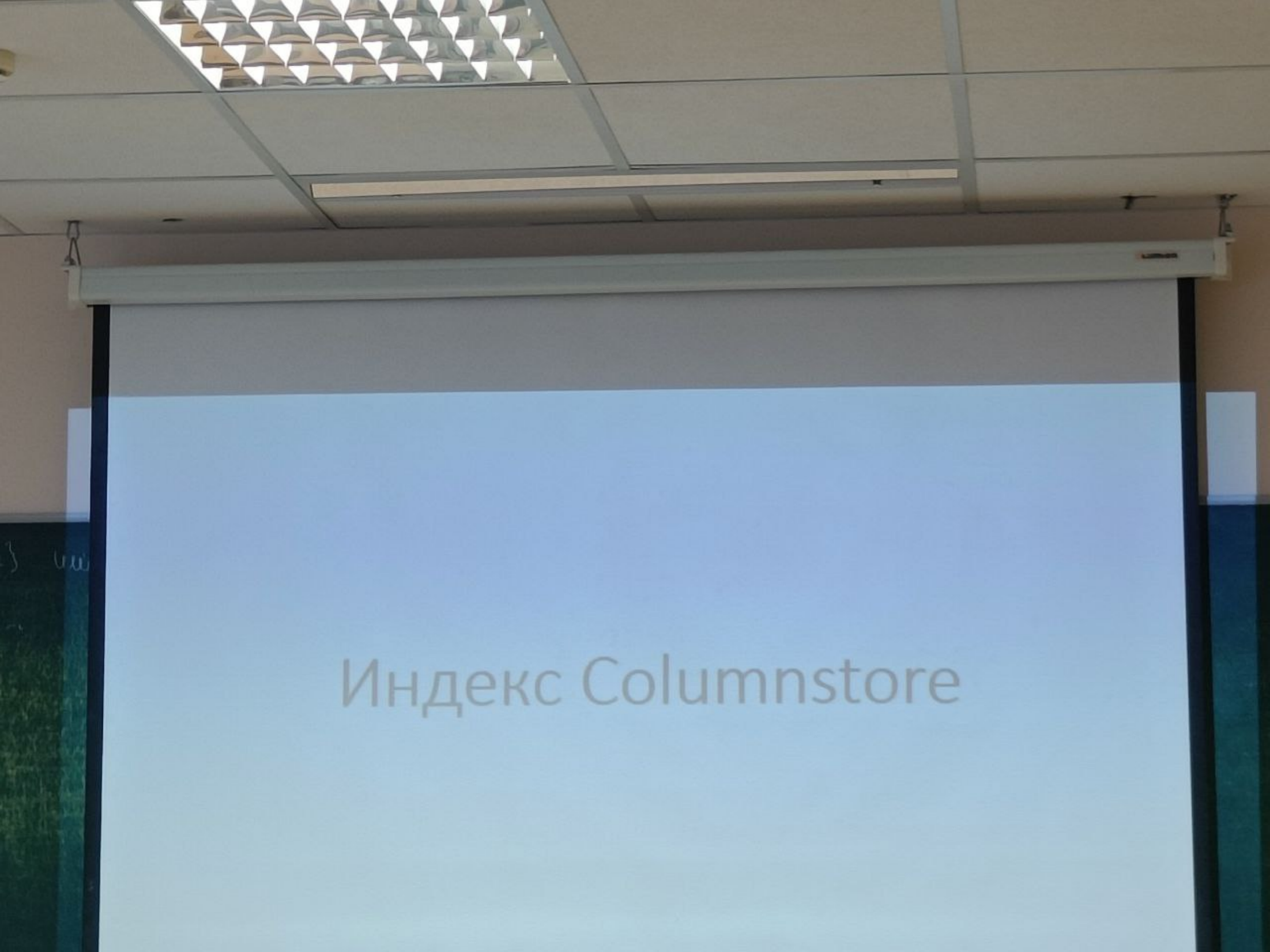

Синтаксис CREATE INDEX
CREATE [UNIQUE] [CLUSTERED |NONCLUSTERED]
INDEX index_name
ON table (column [ASC I DESC] [, ...n])
[WITH (IGNORE_DUP_KEY | DROP_EXISTING |
SORT_IN_TEMPDB)]
[ON filegroup]

- IGNORE_DUP_KEY требуется только в том случае, если индекс определен как UNIQUE. Если этот параметр не был использован раньше, то недоступен.

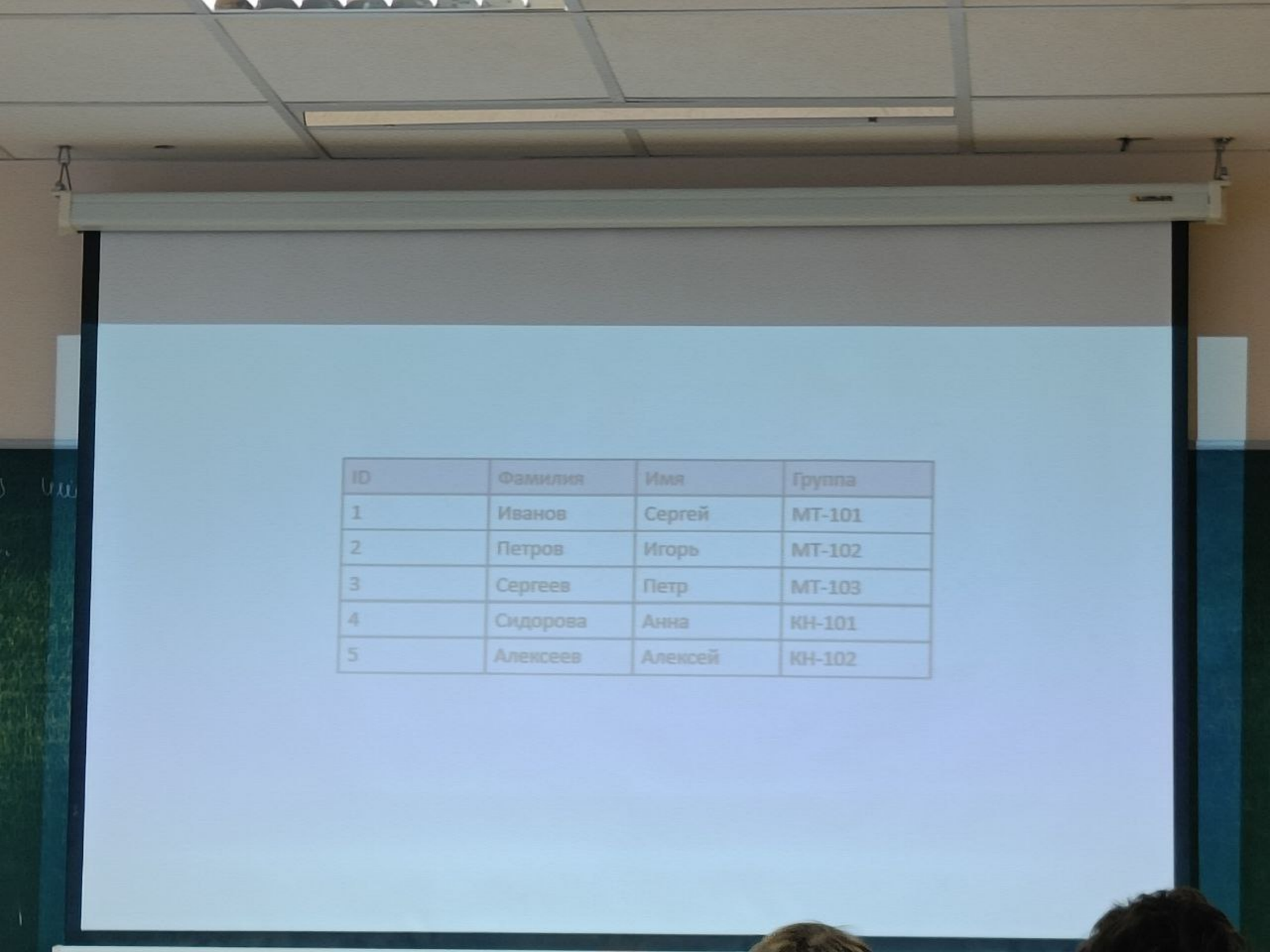
- DROP_EXISTING указывается, если в базе данных существует индекс с таким же именем, то этот индекс удаляется перед созданием нового. Удобно, если фактически столбцы в индексе не изменяются.

- **SORT_IN_TEMPDB.** Когда индекс создается при наличии данных в таблице, возможно, если таблица большая, целесообразно получить для индекса данные, отсортированные во временной базе данных tempdb. Применяется этот параметр, если таблица большая, или если tempdb находится с базой данных на разных жестких дисках.

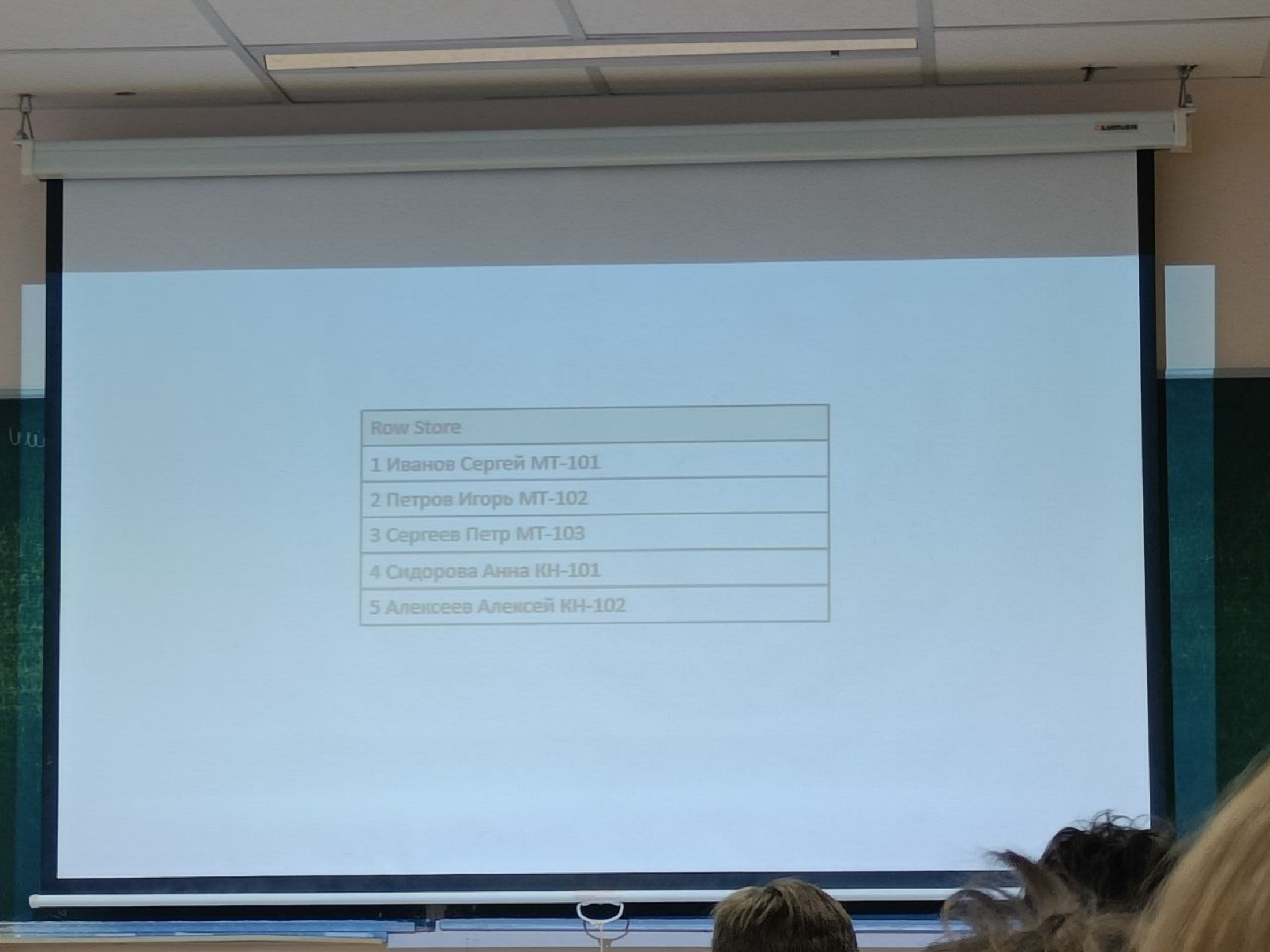

ON — дополнительный. Но этот параметр
обязателен, если указывается файловая группа.
И он не требуется, если индекс должен быть
создан для файловой группы PRIMARY.

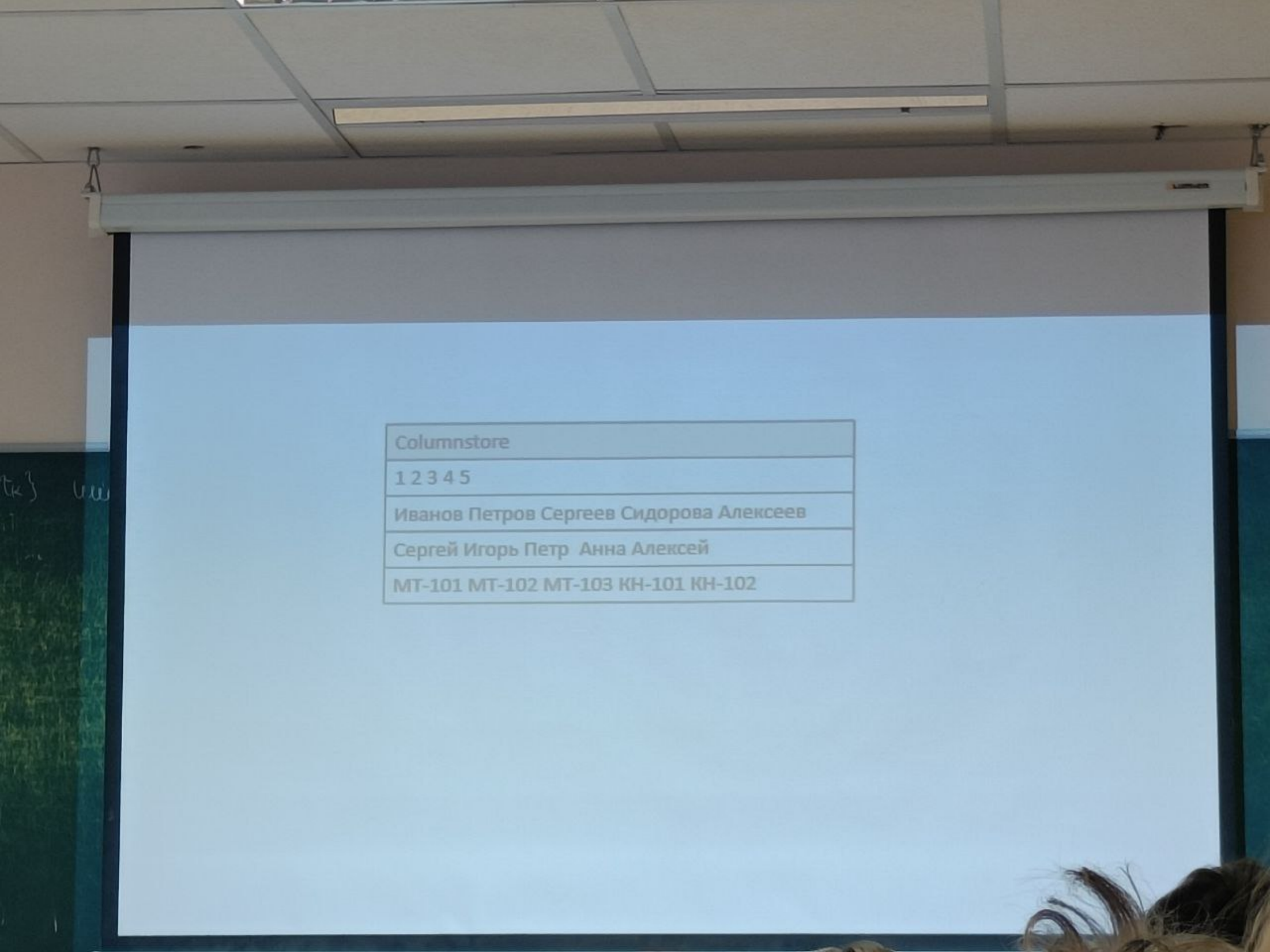

Индекс Columnstore

Индекс Columnstore группирует и сохраняет данные каждого столбца с последующим соединением всех столбцов для завершения индекса в целом. Они отличаются от традиционных индексов, которые группируют и сохраняют данные для каждого ряда, затем объединяют все ряды, чтобы завершить создание всего индекса.



ID	Фамилия	Имя	Группа
1	Иванов	Сергей	МТ-101
2	Петров	Игорь	МТ-102
3	Сергеев	Петр	МТ-103
4	Сидорова	Анна	КН-101
5	Алексеев	Алексей	КН-102


Row Store
1 Иванов Сергей МТ-101
2 Петров Игорь МТ-102
3 Сергеев Петр МТ-103
4 Сидорова Анна КН-101
5 Алексеев Алексей КН-102


Columnstore
1 2 3 4 5
Иванов Петров Сергеев Сидорова Алексеев
Сергей Игорь Петр Анна Алексей
МТ-101 МТ-102 МТ-103 КН-101 КН-102

Индексы columnstore предоставляют улучшенную производительность для таких стандартных запросов хранилища данных, как фильтрация, статистическая обработка, группирование и запросы соединения типа «звезда».

Следующие типы данных могут быть включены в индекс columnstore.

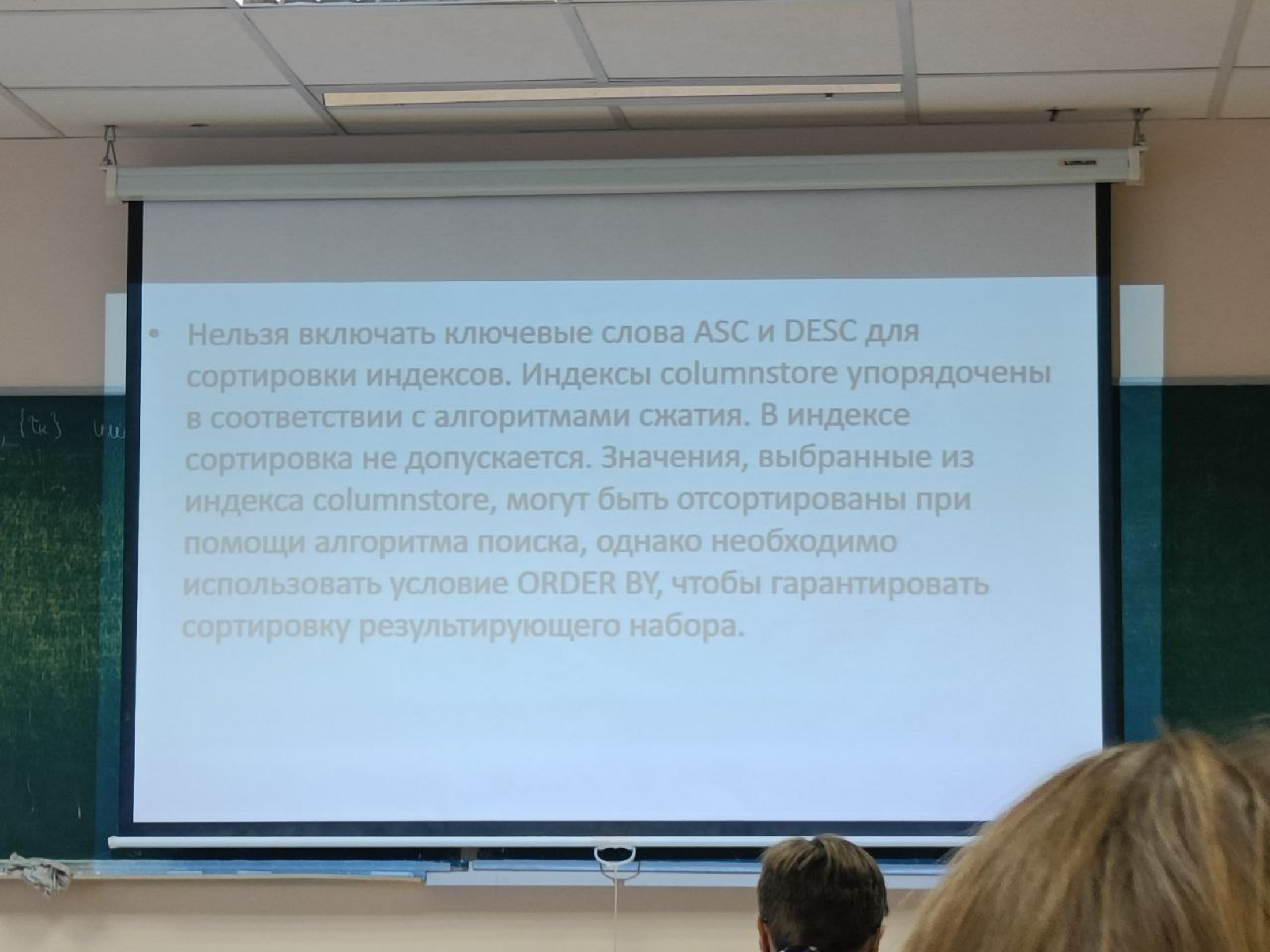
- char и varchar
- nchar и nvarchar (кроме varchar(max) и nvarchar(max))
- decimal (и numeric) (кроме как при точности более 18 цифр)

Основные ограничения индекса columnstore.

- Не более 1024 столбцов.
- Не пригоден для кластеризации. Доступны только некластеризованные индексы columnstore.
- Не может быть уникальным индексом.

- Не может быть создан для представления или индексированного представления.
- Не может содержать разреженный столбец.
- Не может использоваться в качестве первичного ключа или внешнего ключа.

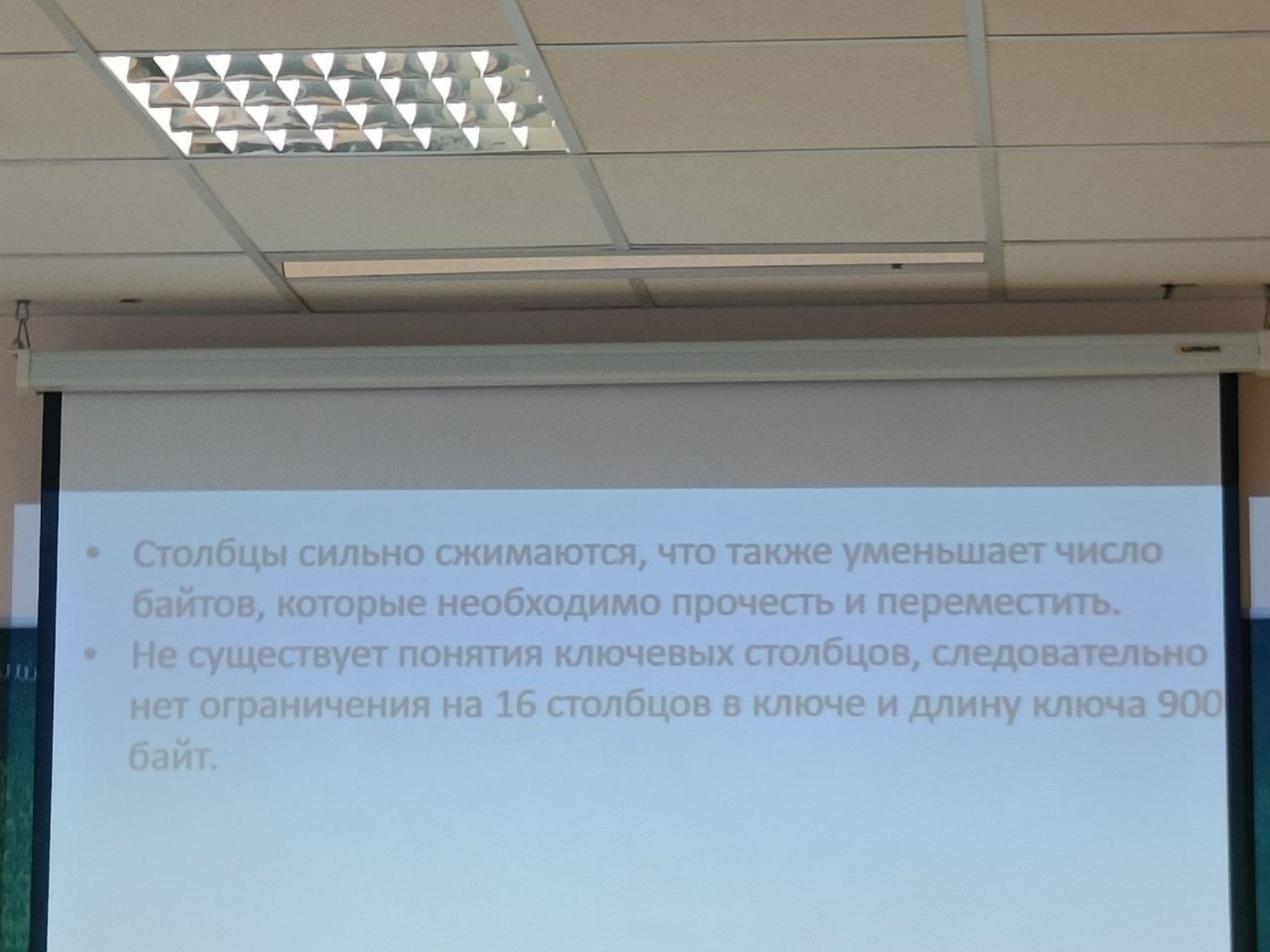
- Не может быть изменен с помощью инструкции ALTER INDEX. Вместо этого удалите и заново создайте индекс columnstore.
- Не может быть создан с помощью ключевого слова INCLUDE.

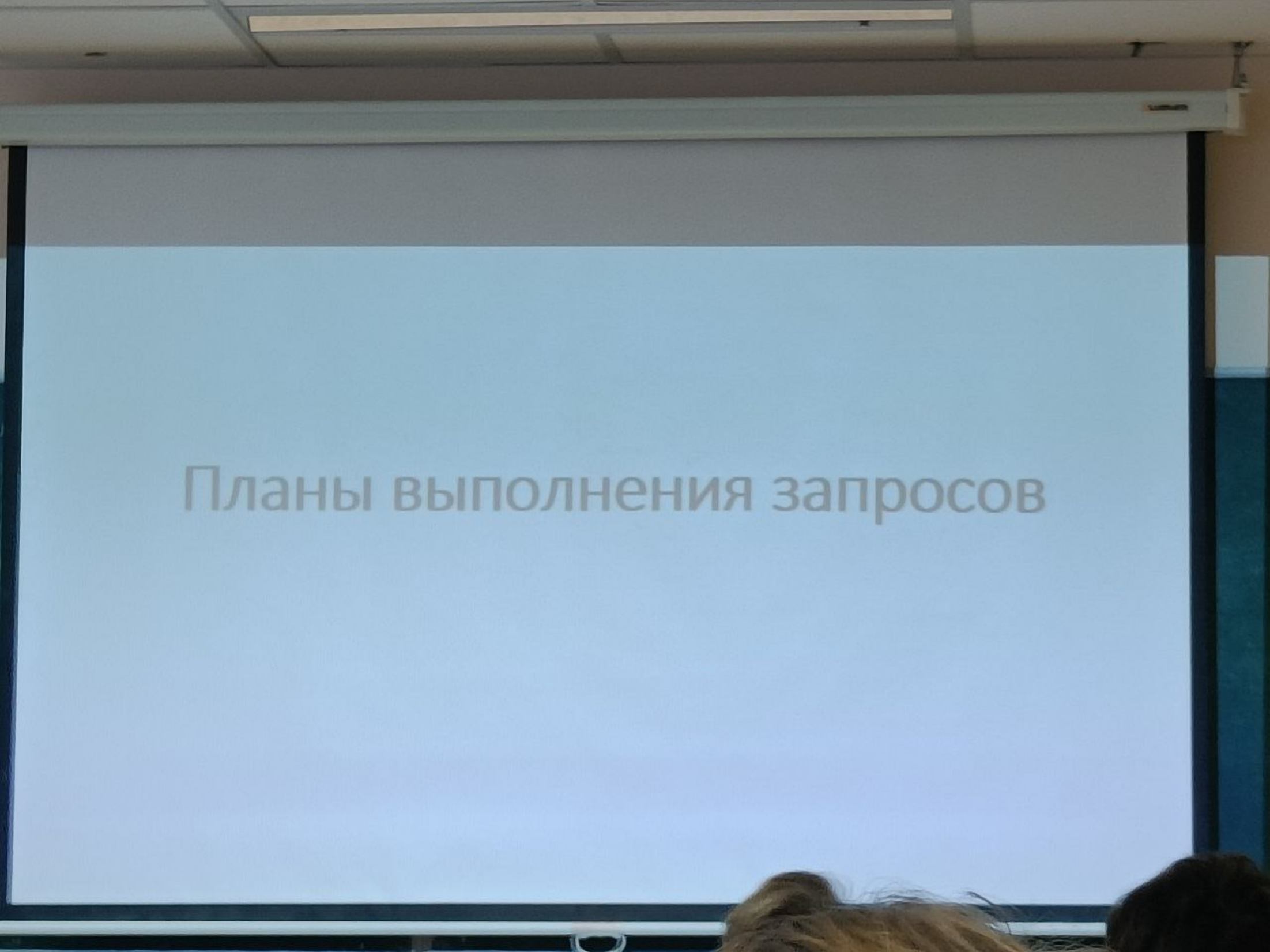

• Нельзя включать ключевые слова ASC и DESC для сортировки индексов. Индексы columnstore упорядочены в соответствии с алгоритмами сжатия. В индексе сортировка не допускается. Значения, выбранные из индекса columnstore, могут быть отсортированы при помощи алгоритма поиска, однако необходимо использовать условие ORDER BY, чтобы гарантировать сортировку результирующего набора.

- Не использует и не хранит статистические данные в форме традиционного индекса.
- Не может содержать столбец с атрибутом FILESTREAM. Другие столбцы в таблице, неиспользуемые в индексе, могут содержать атрибут FILESTREAM.

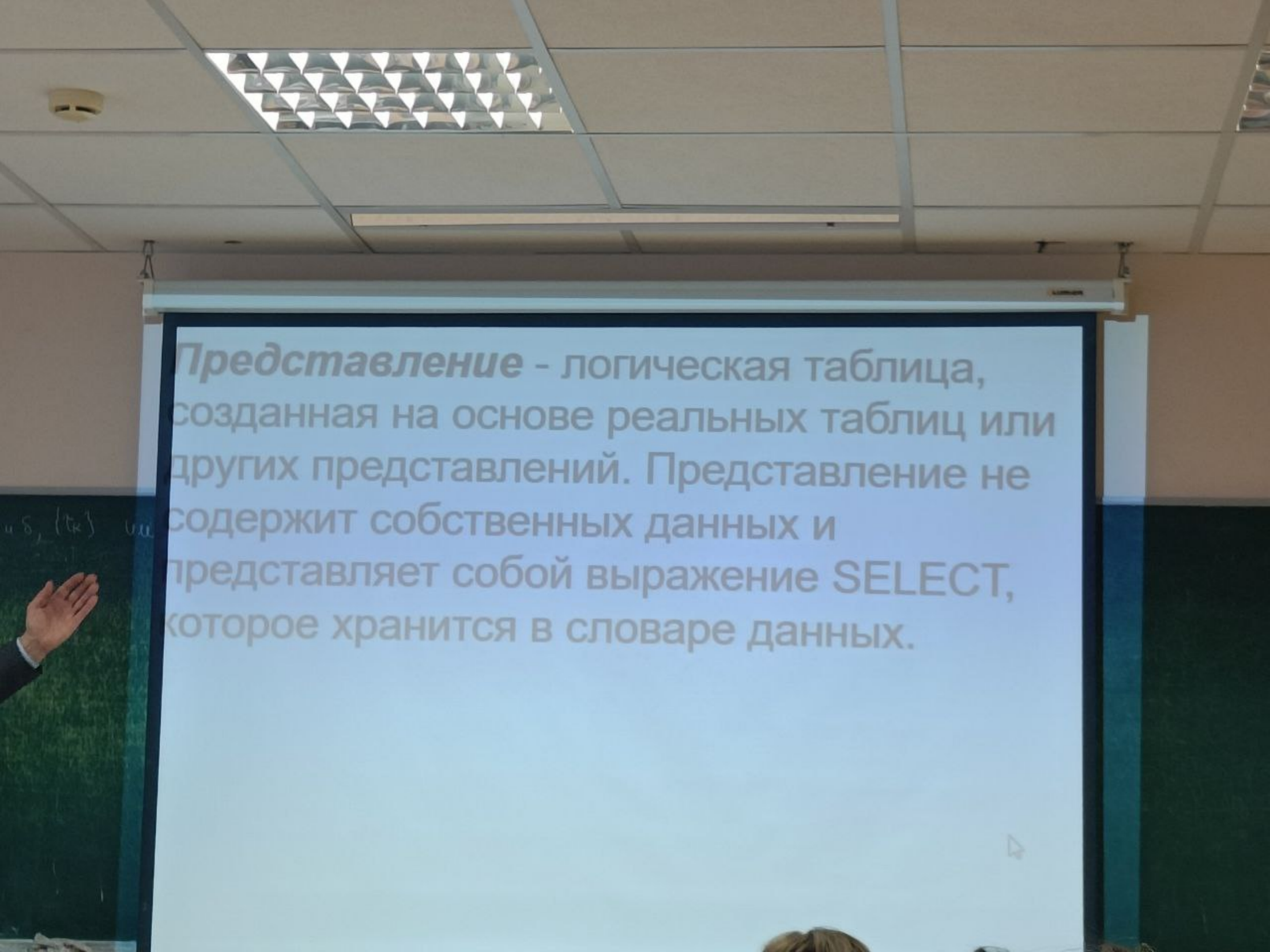
Преимущества

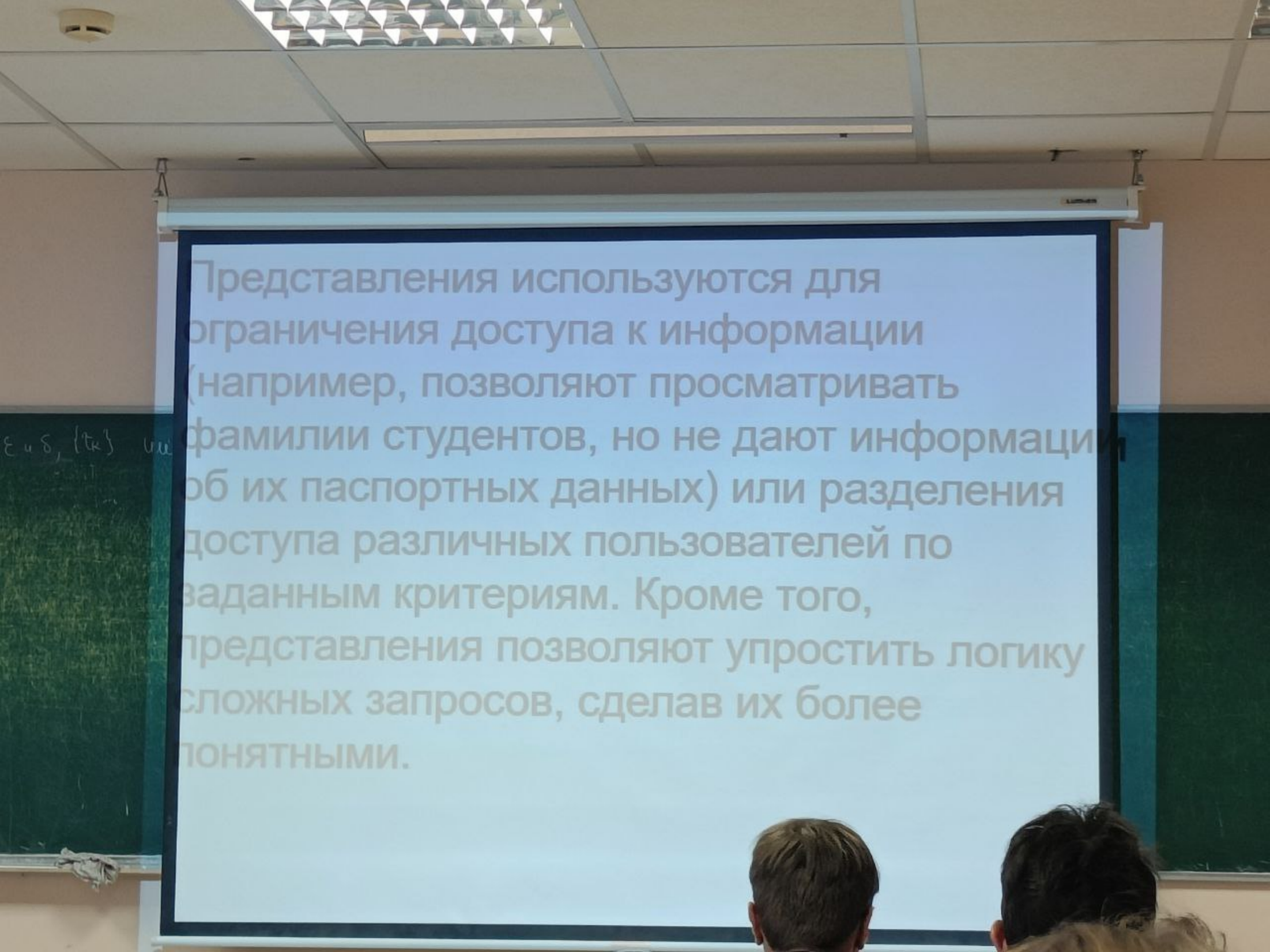
- Большинство запросов не затрагивают все столбцы таблицы. Таким образом, многие поля не будут передаваться в память. Это улучшает использование буферного пула, что уменьшает общее число операций ввода вывода.
- Должны быть прочитаны только необходимые столбцы, следовательно меньше данных читается из диска в память и затем перемещается из памяти в кэш процессора

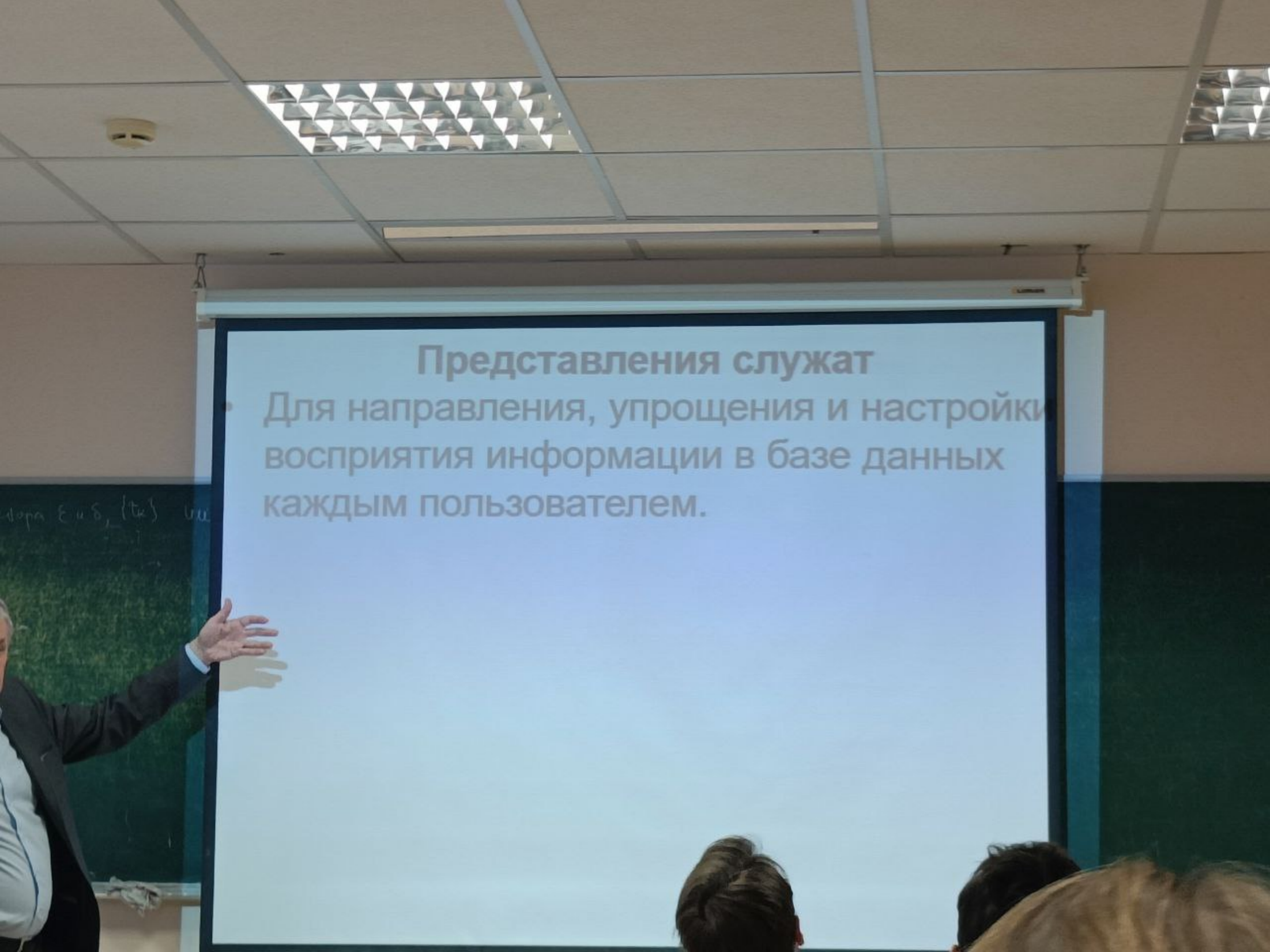

• Столбцы сильно сжимаются, что также уменьшает число байтов, которые необходимо прочесть и переместить.
• Не существует понятия ключевых столбцов, следовательно нет ограничения на 16 столбцов в ключе и длину ключа 900 байт.

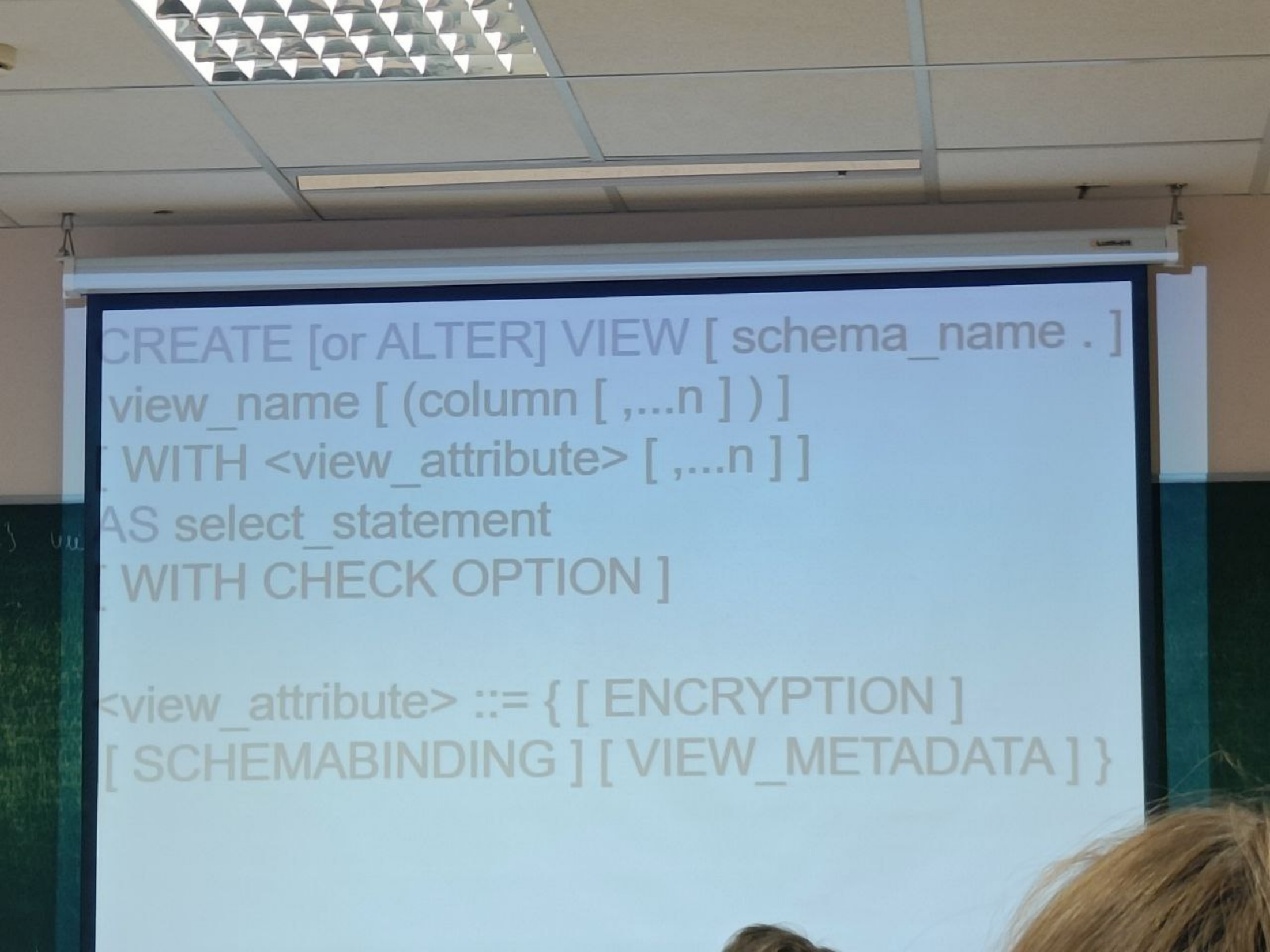

Планы выполнения запросов

Представления


Представление - логическая таблица, созданная на основе реальных таблиц или других представлений. Представление не содержит собственных данных и представляет собой выражение SELECT, которое хранится в словаре данных.

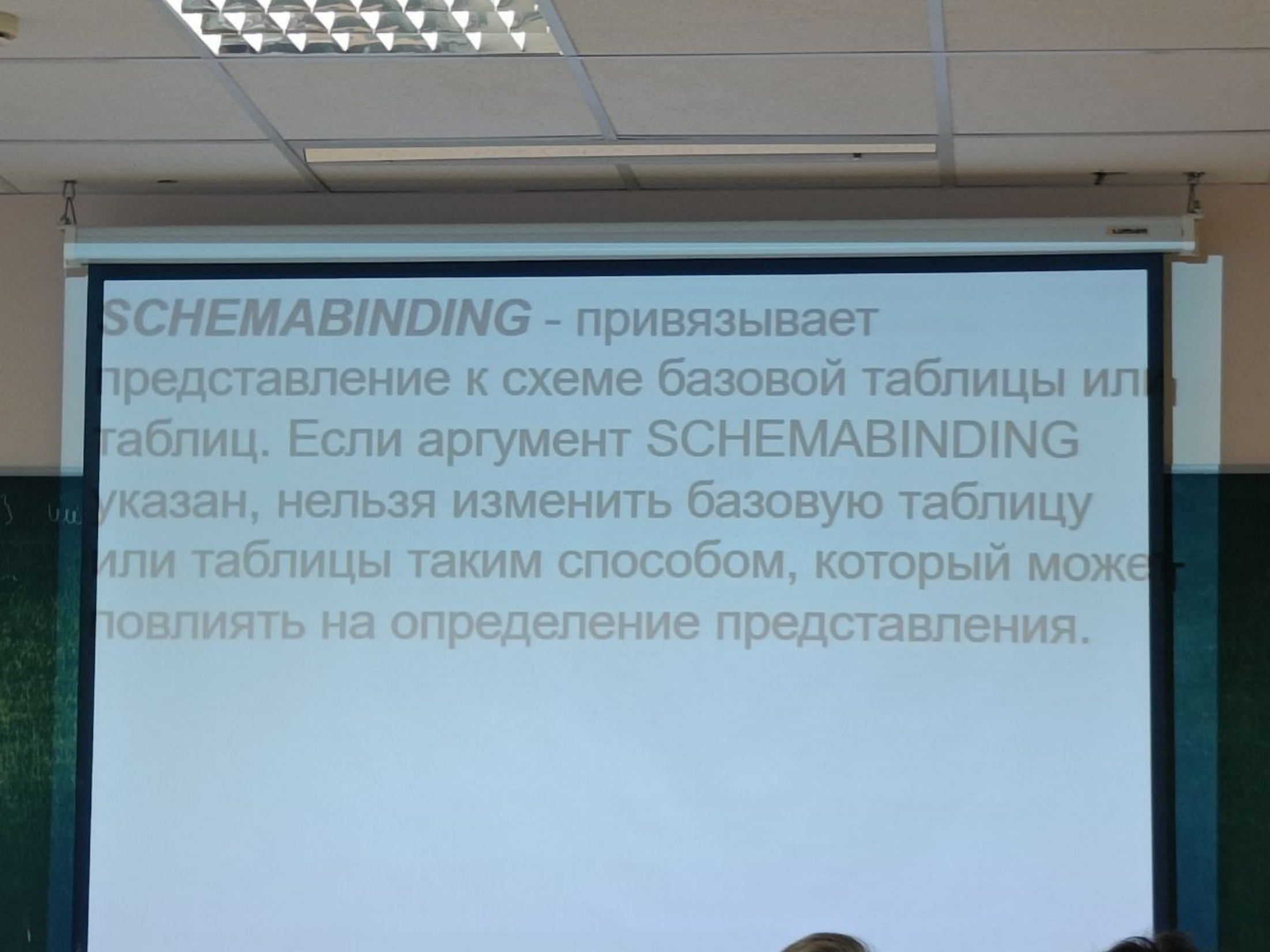

Представления используются для
ограничения доступа к информации
(например, позволяют просматривать
фамилии студентов, но не дают информации
об их паспортных данных) или разделения
доступа различных пользователей по
заданным критериям. Кроме того,
представления позволяют упростить логику
сложных запросов, сделав их более
понятными.

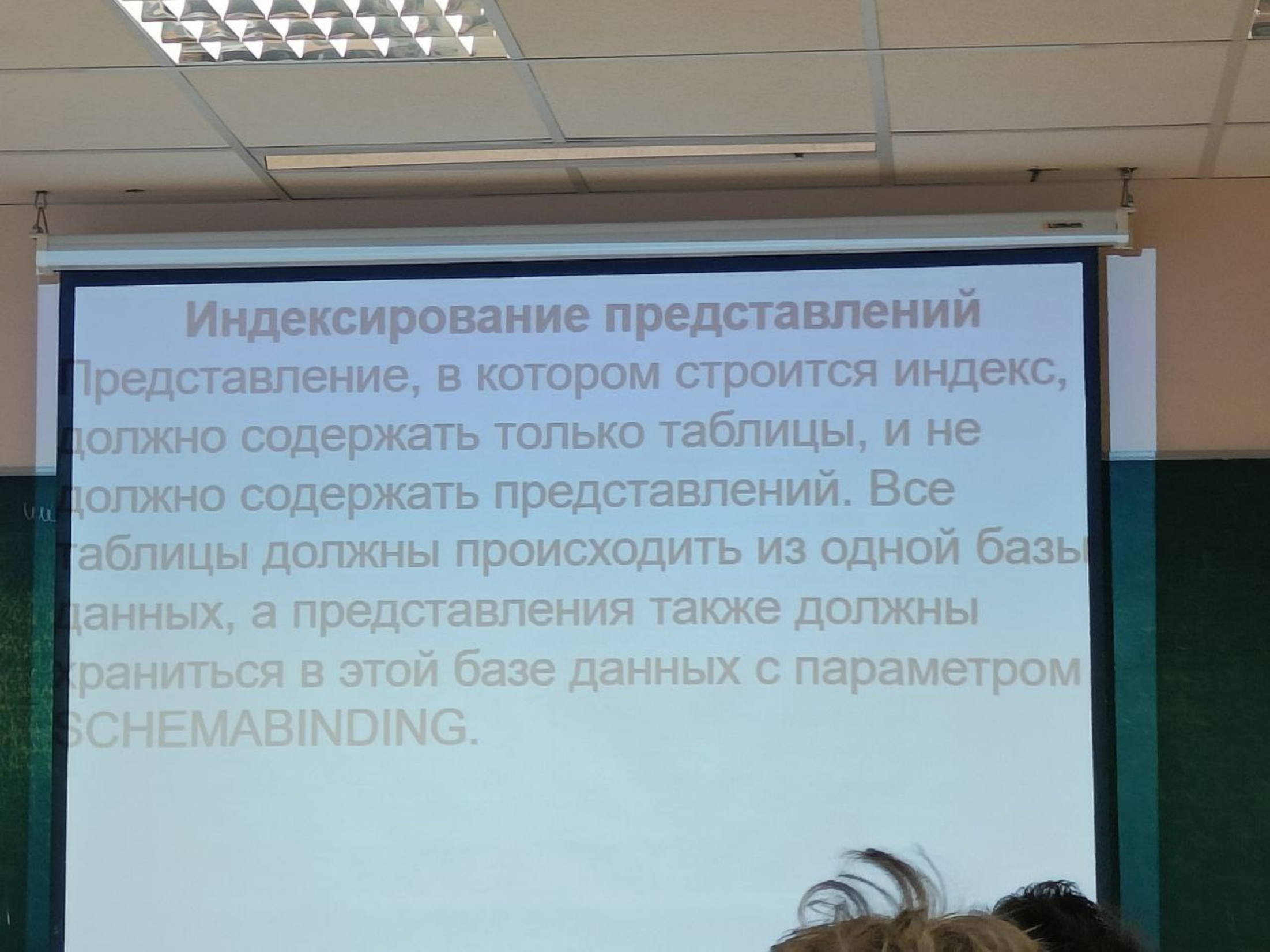

Представления служат
Для направления, упрощения и настройки
восприятия информации в базе данных
каждым пользователем.

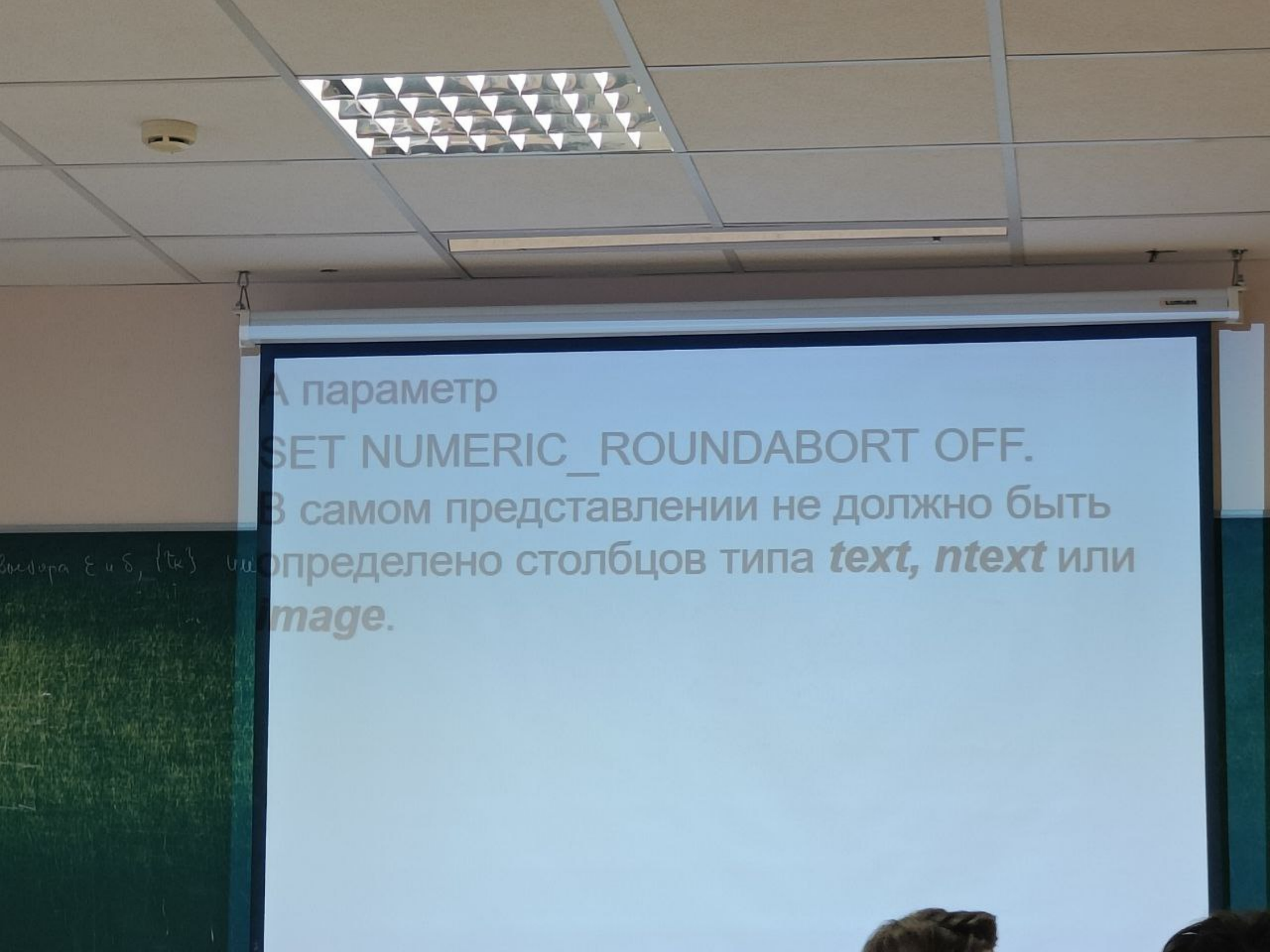

CREATE [or ALTER] VIEW [schema_name .]
view_name [(column [,...n])]
[WITH <view_attribute> [,...n]]
AS select_statement
[WITH CHECK OPTION]
<view_attribute> ::= { [ENCRYPTION]
[SCHEMABINDING] [VIEW_METADATA] }

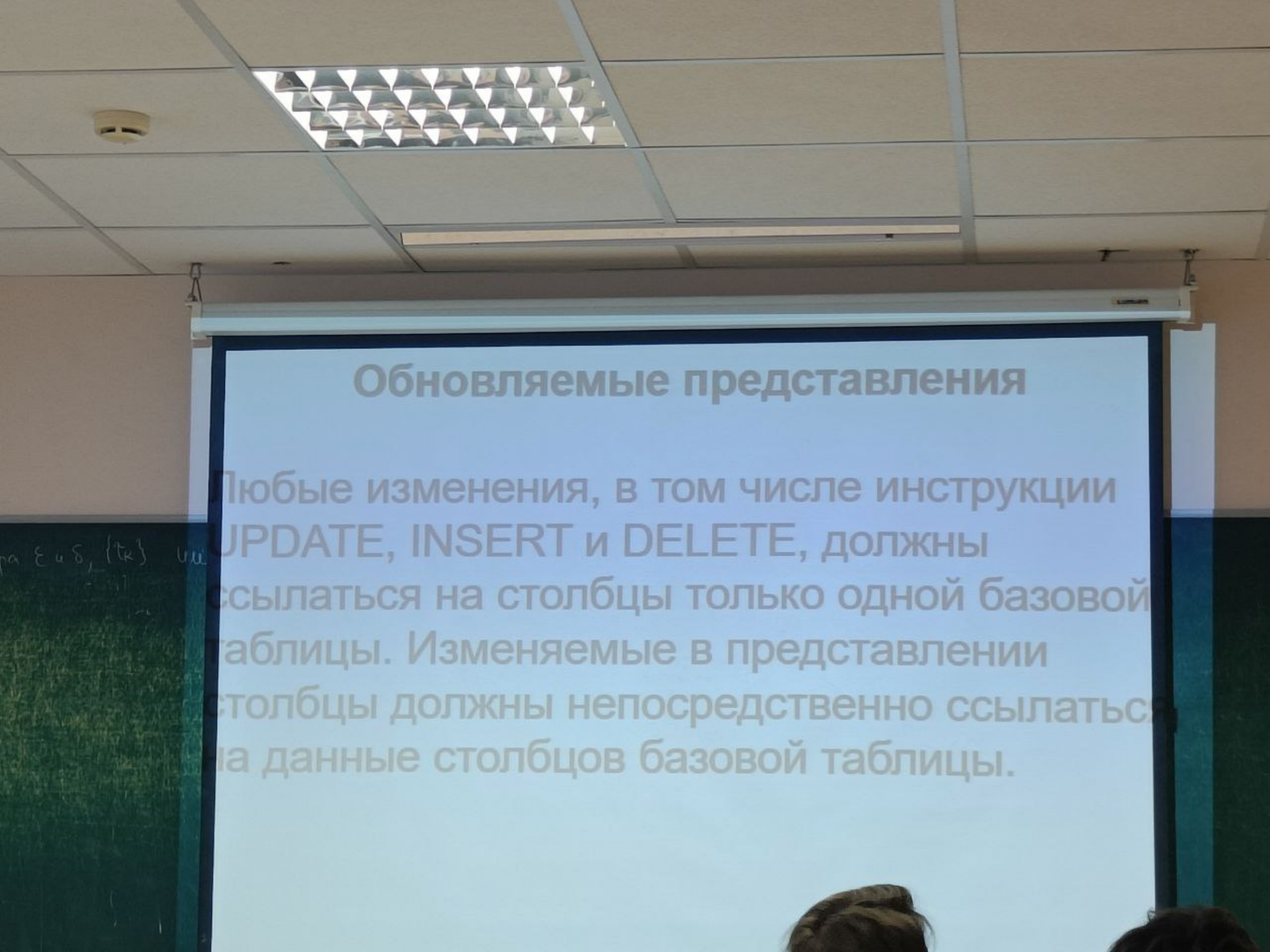
CHECK OPTION - обеспечивает соответствие всех выполняемых для представления инструкций модификации данных критериям, заданным при помощи аргумента select_statement.

Если строка изменяется посредством представления, предложение WITH CHECK OPTION гарантирует, что после фиксации изменений доступ к данным из представления сохранится.


SCHEMABINDING - привязывает
представление к схеме базовой таблицы или
таблиц. Если аргумент SCHEMABINDING
указан, нельзя изменить базовую таблицу
или таблицы таким способом, который может
повлиять на определение представления.


Индексирование представлений
Представление, в котором строится индекс,
должно содержать только таблицы, и не
должно содержать представлений. Все
таблицы должны происходить из одной базы
данных, а представления также должны
храниться в этой базе данных с параметром
SCHEMABINDING.

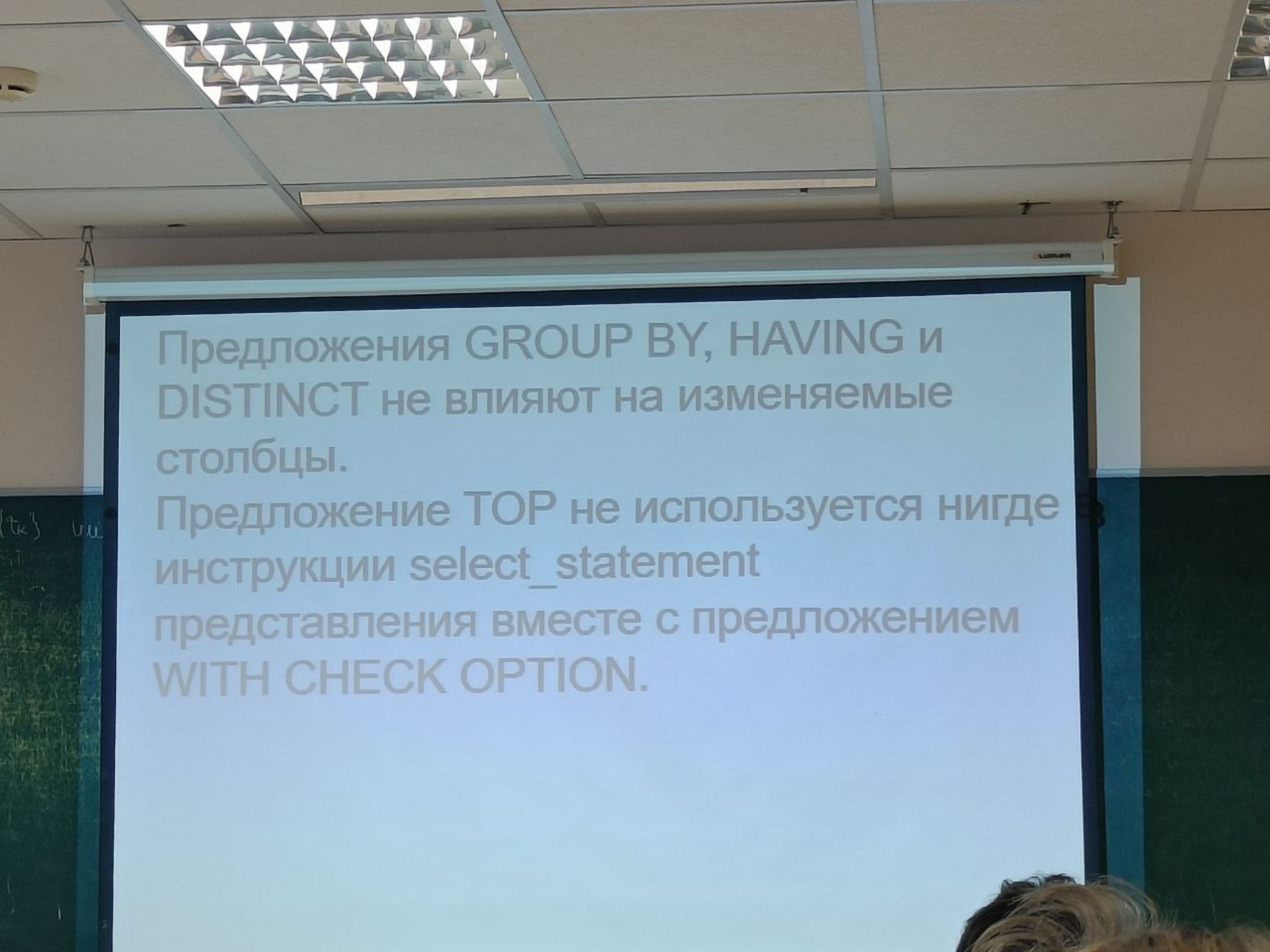

А параметр
SET NUMERIC_ROUNDABORT OFF.
В самом представлении не должно быть
определено столбцов типа text, ntext или
image.

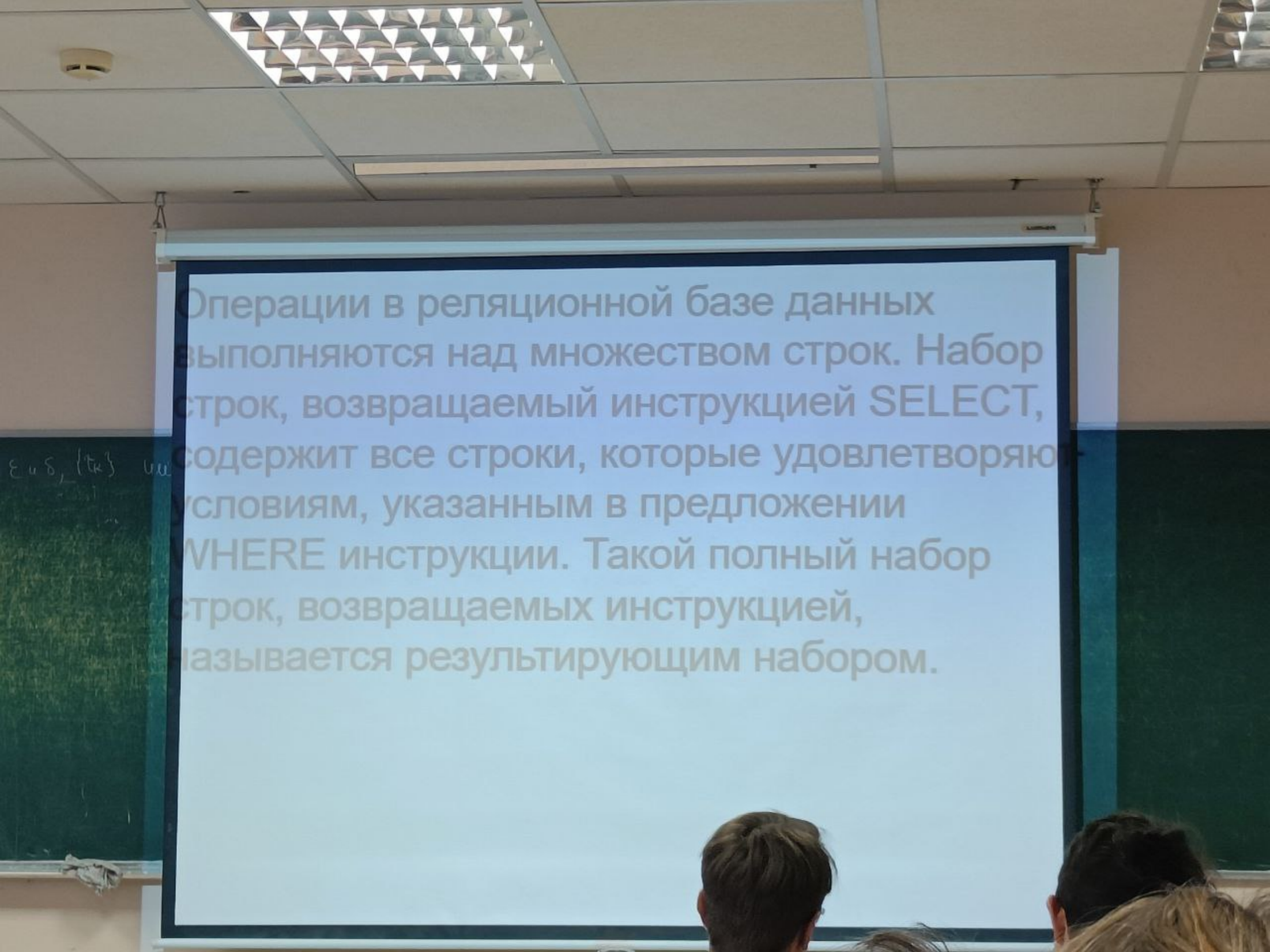

Обновляемые представления
Любые изменения, в том числе инструкции
UPDATE, INSERT и DELETE, должны
ссылаться на столбцы только одной базовой
таблицы. Изменяемые в представлении
столбцы должны непосредственно ссылаться
на данные столбцов базовой таблицы.

Столбцы нельзя сформировать каким-либо другим образом, в том числе:

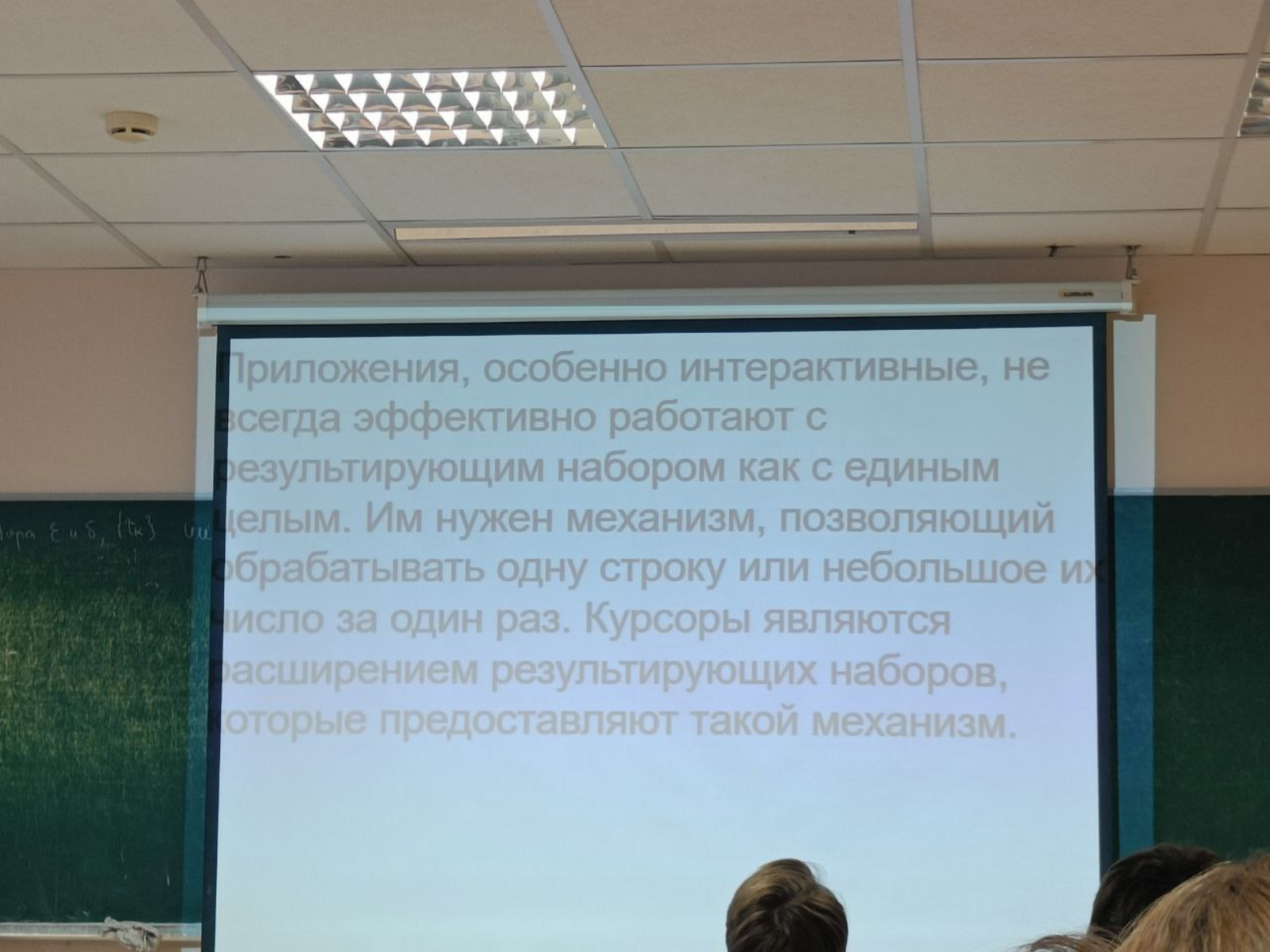
- при помощи агрегатной функции: AVG, COUNT, SUM, MIN, MAX, GROUPING, STDEV, STDEVP, VAR и VARP;
- на основе вычисления. Столбец нельзя вычислить по выражению, включающему другие столбцы.

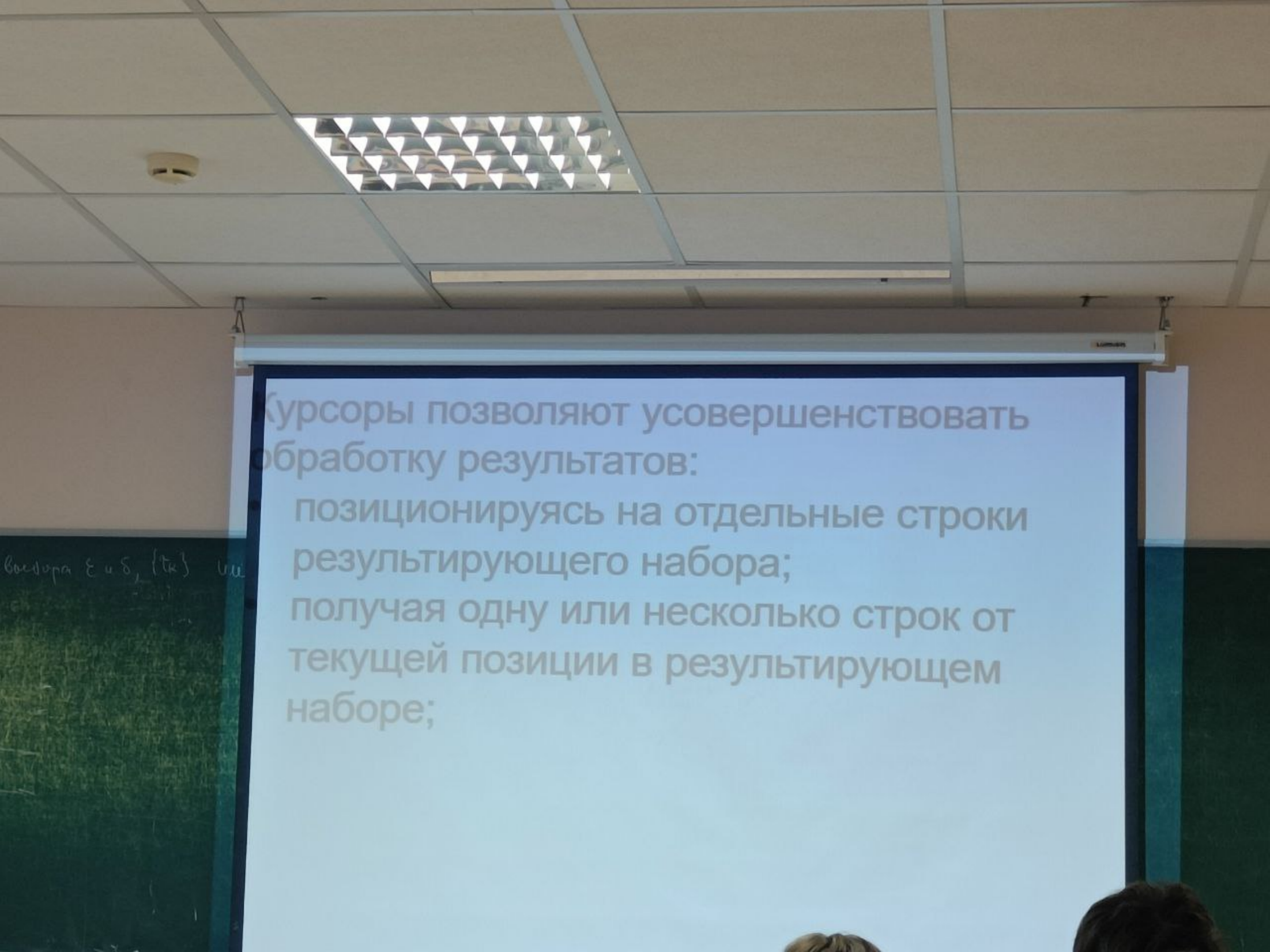
Столбцы, сформированные при помощи операторов UNION, UNION ALL, CROSSJOIN, EXCEPT и INTERSECT, считаются вычисляемыми и также не являются обновляемыми.

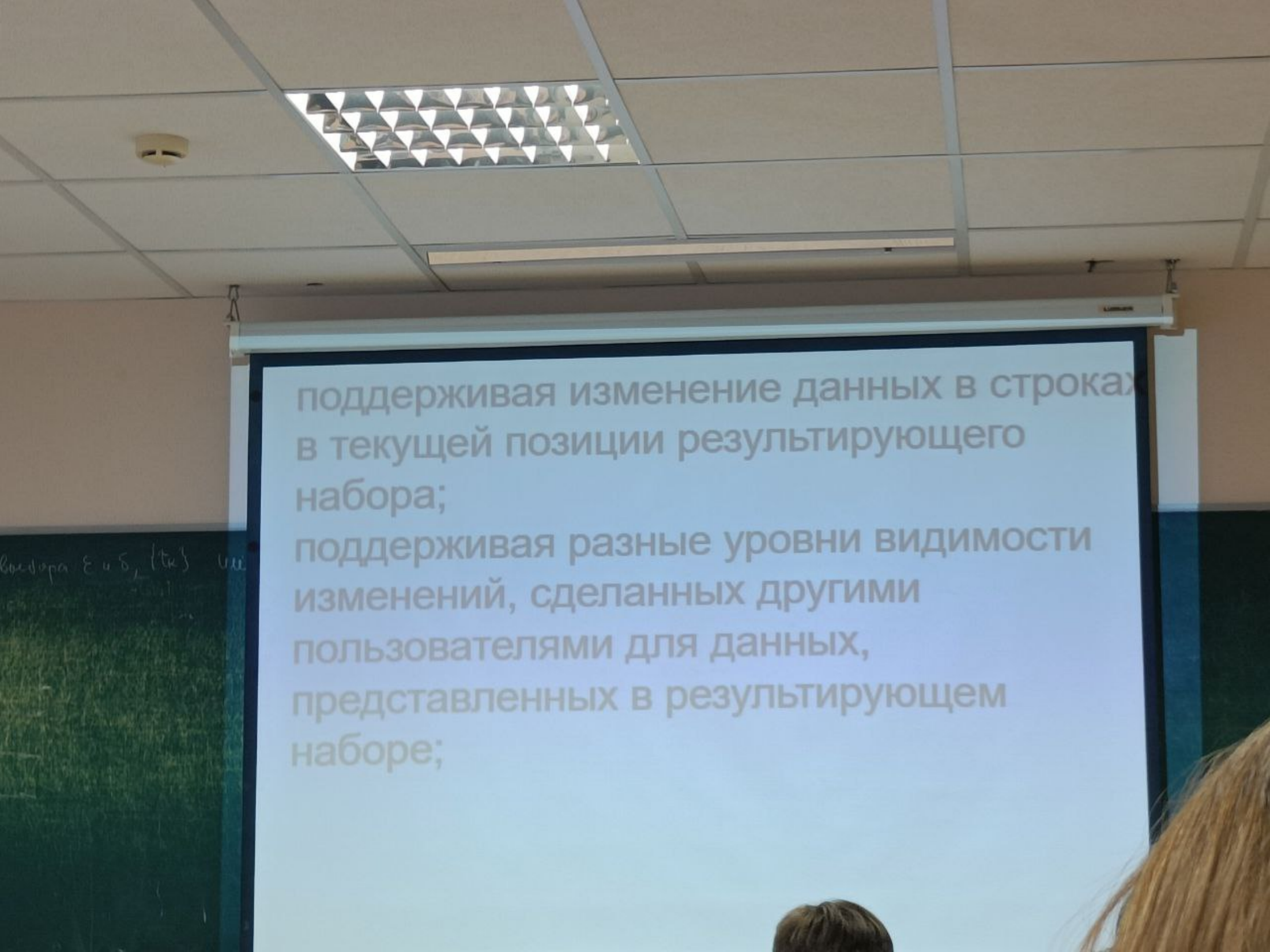

Предложения GROUP BY, HAVING и DISTINCT не влияют на изменяемые столбцы.
Предложение TOP не используется нигде инструкции select_statement представления вместе с предложением WITH CHECK OPTION.


Операции в реляционной базе данных
выполняются над множеством строк. Набор
строк, возвращаемый инструкцией SELECT,
содержит все строки, которые удовлетворяют
условиям, указанным в предложении
WHERE инструкции. Такой полный набор
строк, возвращаемых инструкцией,
называется результирующим набором.

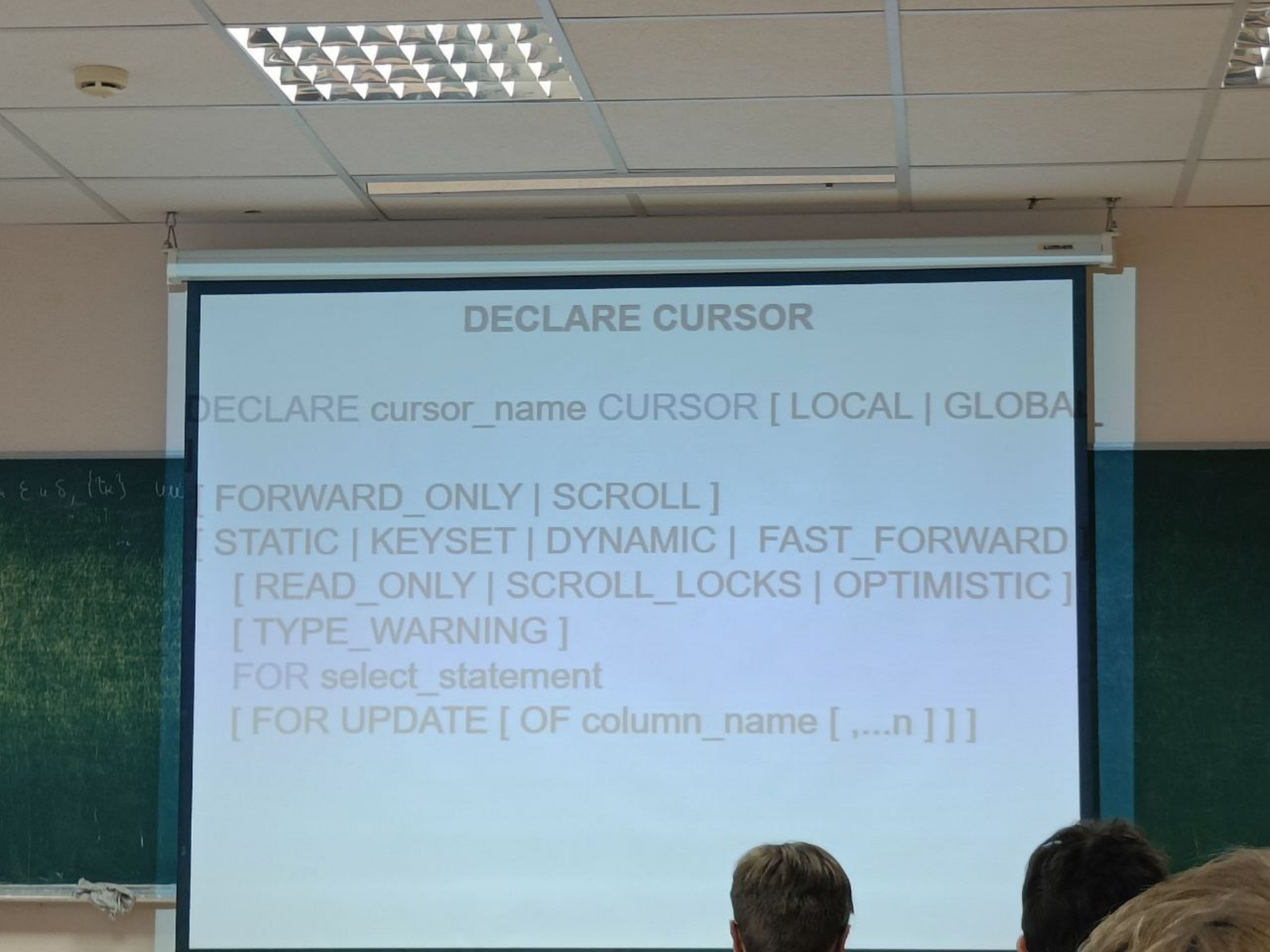
Операции в реляционной базе данных
выполняются над множеством строк. Набор
строк, возвращаемый инструкцией SELECT,
содержит все строки, которые удовлетворяю
условиям, указанным в предложении
WHERE инструкции. Такой полный набор
строк, возвращаемых инструкцией,
называется результирующим набором.

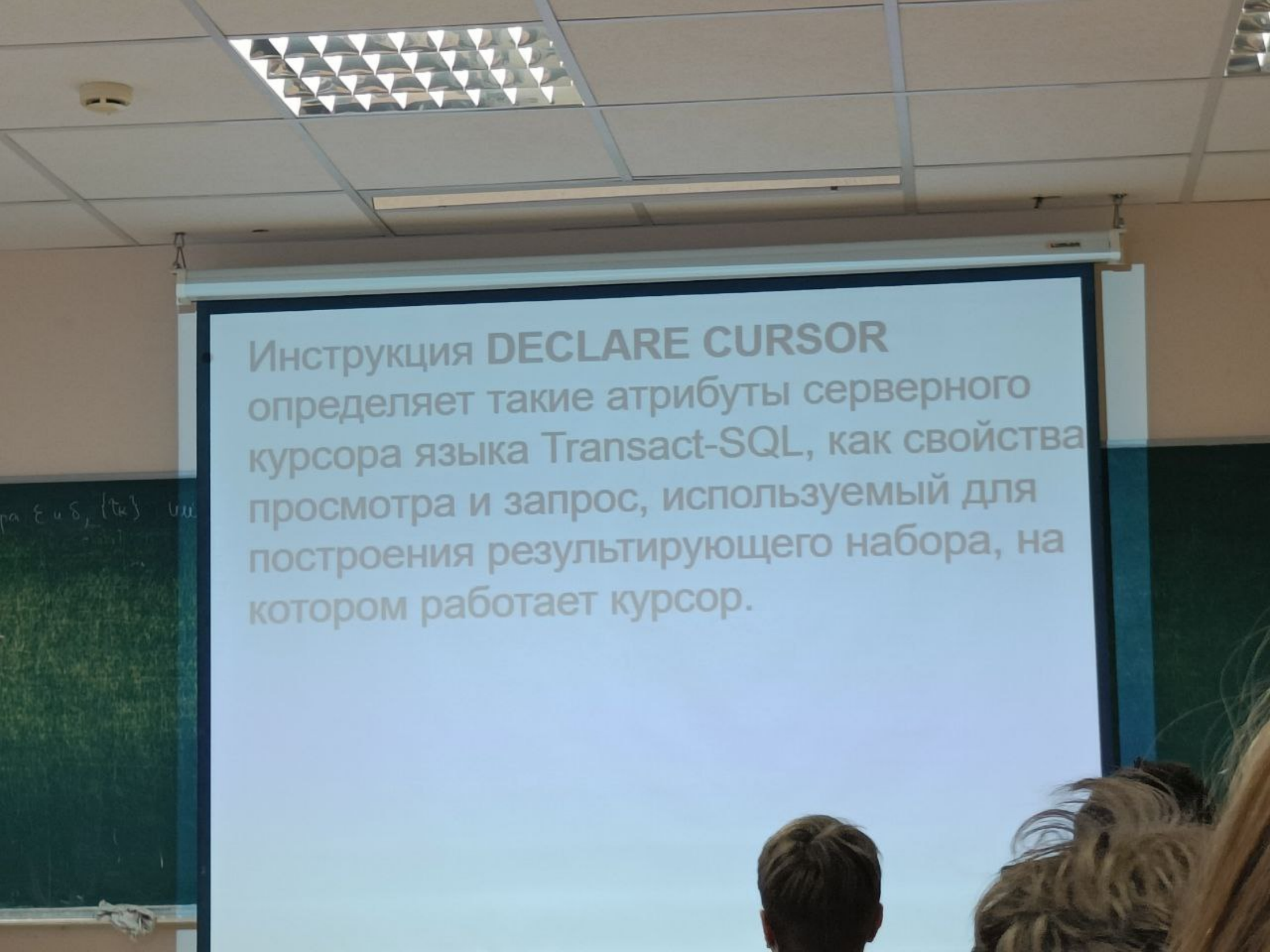

Приложения, особенно интерактивные, не
всегда эффективно работают с
результирующим набором как с единым
целым. Им нужен механизм, позволяющий
обрабатывать одну строку или небольшое их
число за один раз. Курсоры являются
расширением результирующих наборов,
которые предоставляют такой механизм.

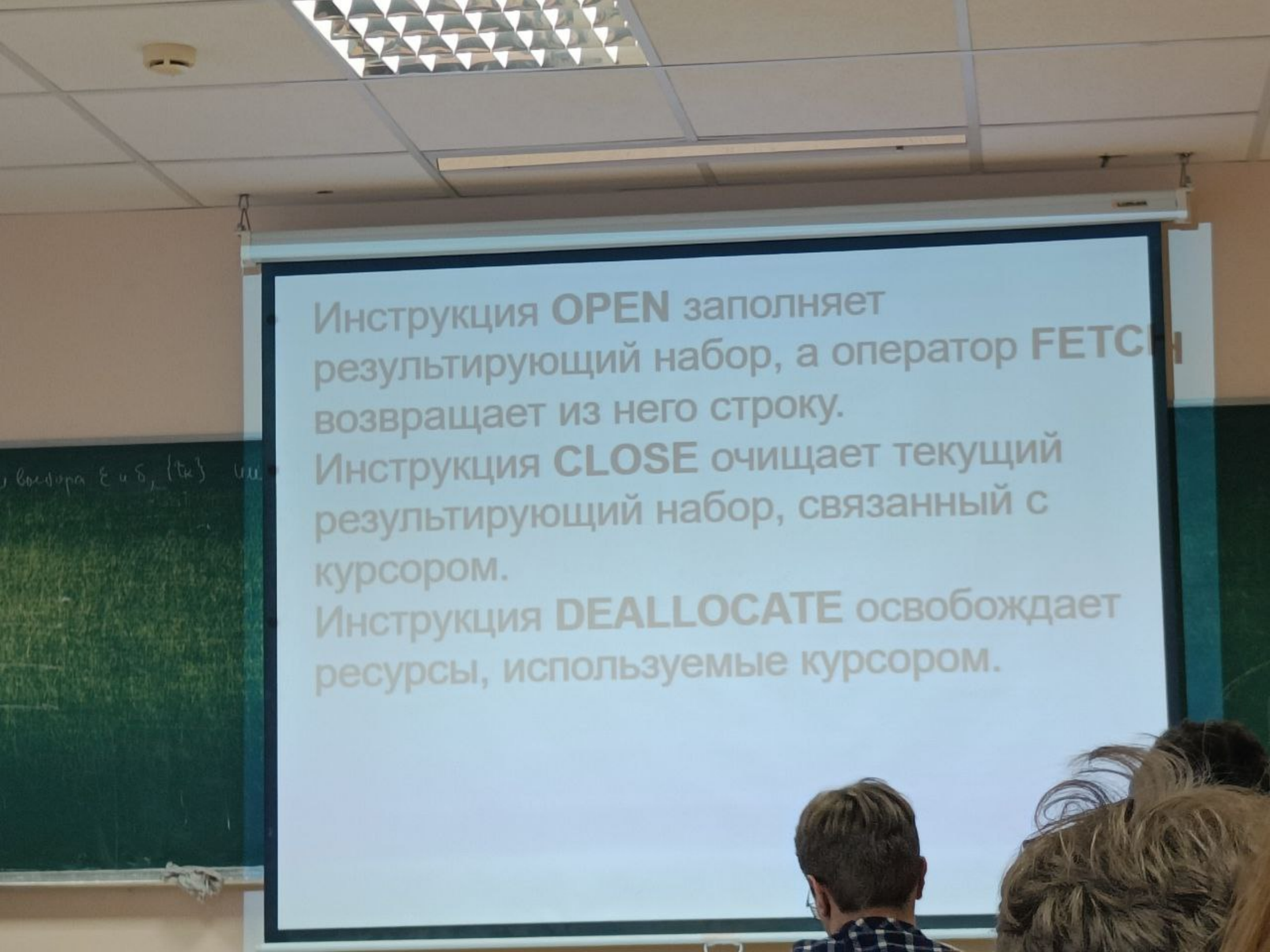

Курсоры позволяют усовершенствовать
обработку результатов:
позиционируясь на отдельные строки
результирующего набора;
получая одну или несколько строк от
текущей позиции в результирующем
наборе;

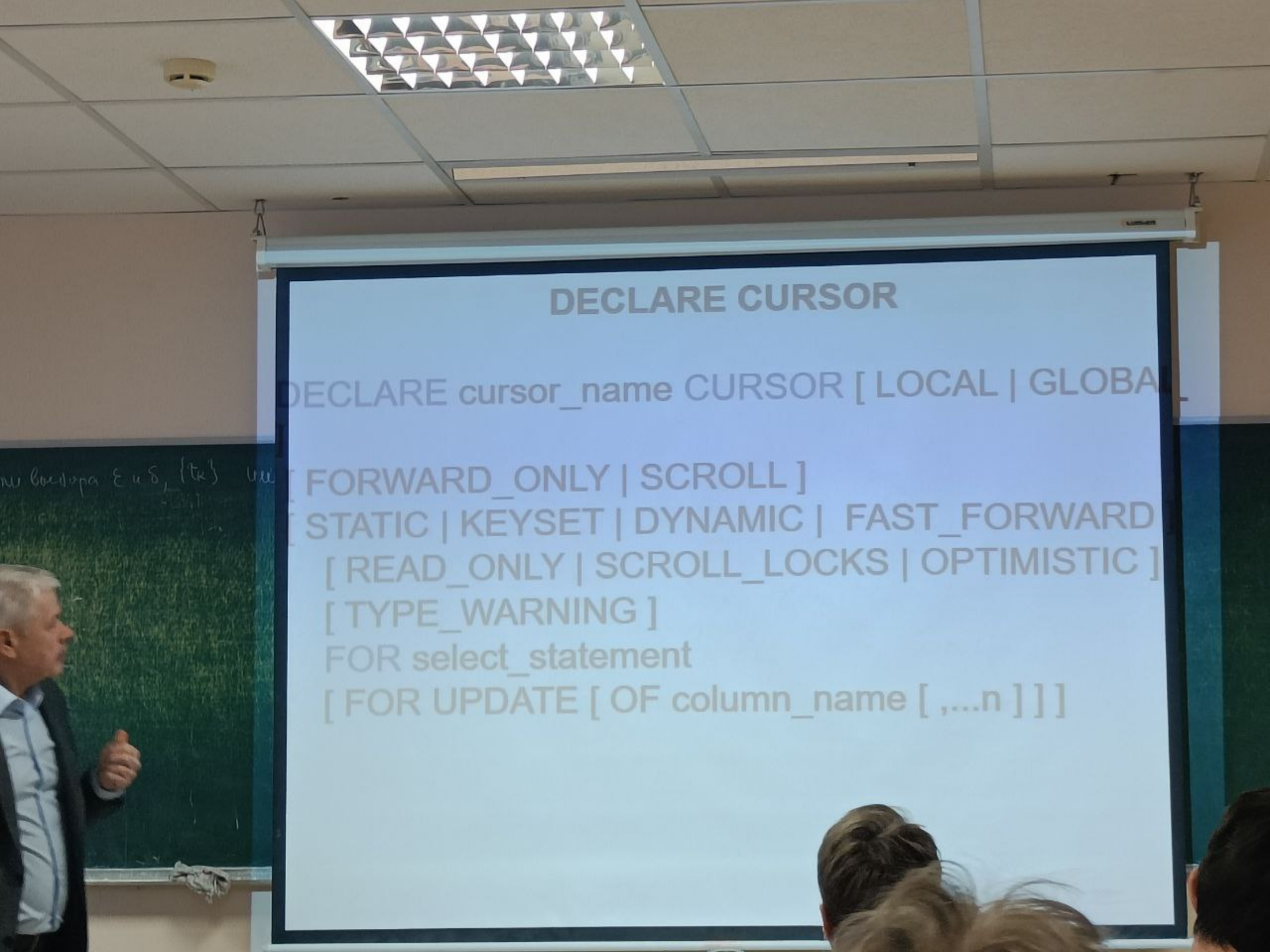

поддерживая изменение данных в строках
в текущей позиции результирующего набора;
поддерживая разные уровни видимости изменений, сделанных другими пользователями для данных, представленных в результирующем наборе;

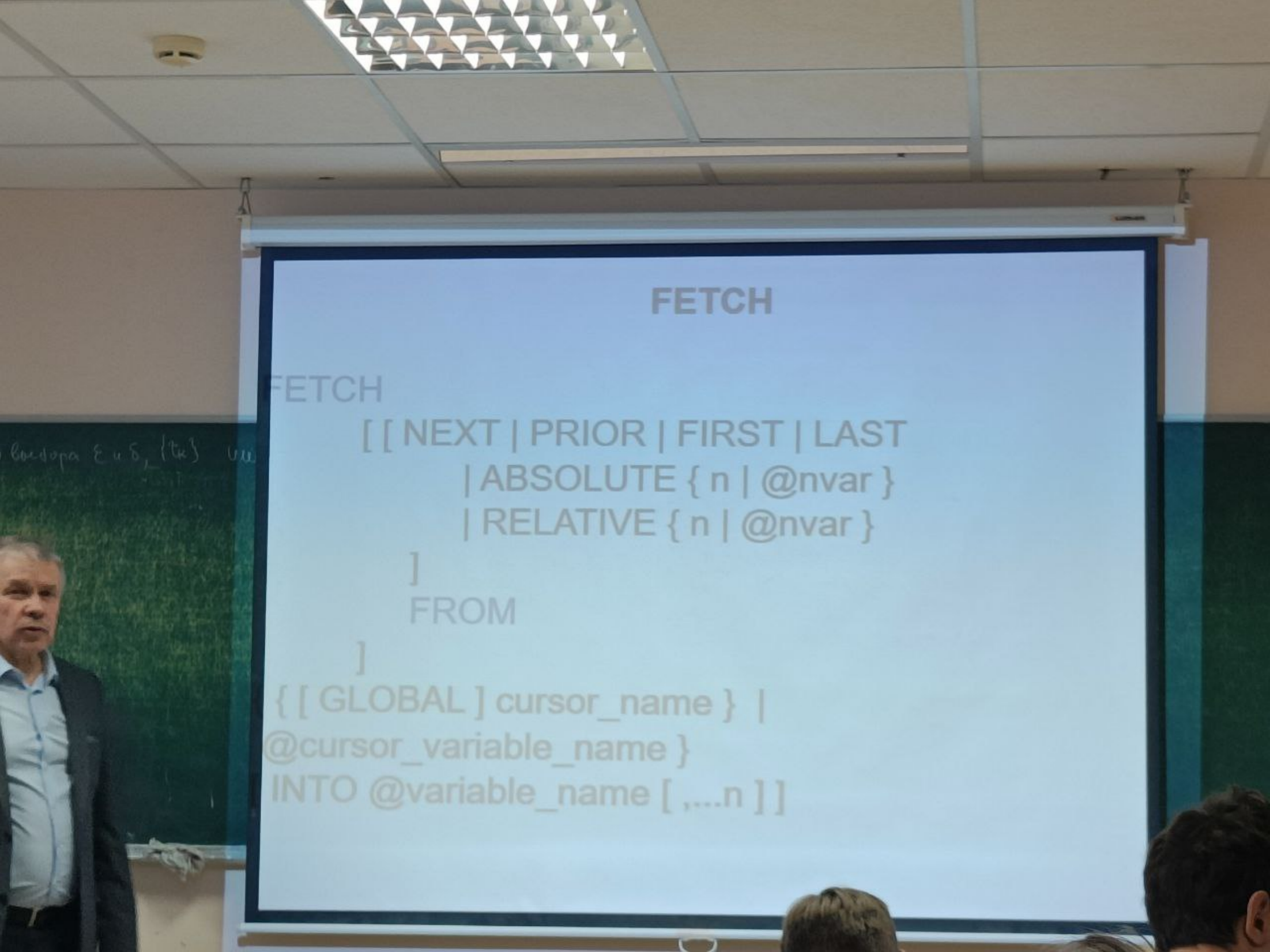
предоставляя инструкциям Transact-SQL в
сценариях, хранимых процедурах и
тригерах доступ к данным
результирующего набора.

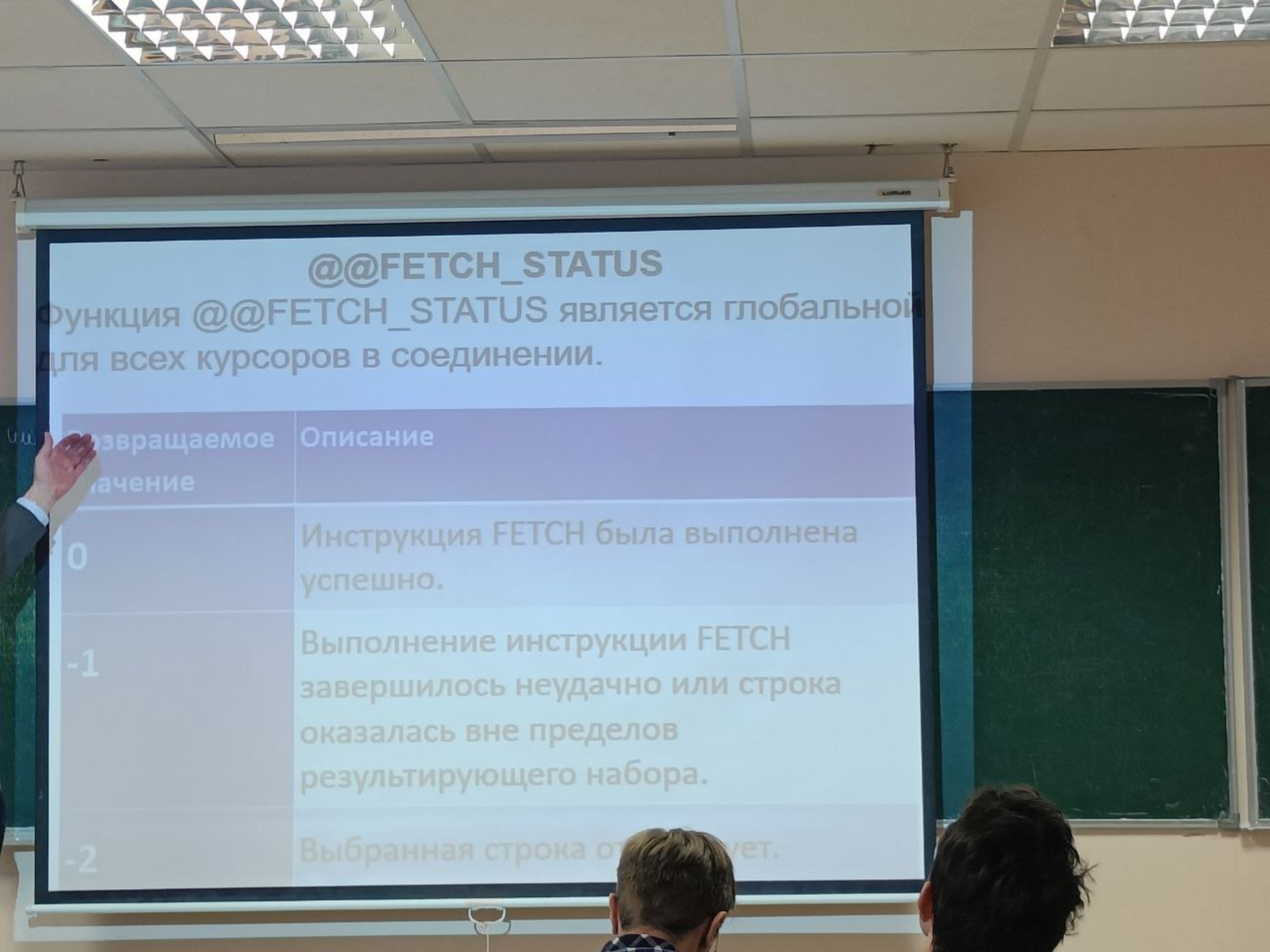

DECLARE CURSOR
DECLARE cursor_name CURSOR [LOCAL | GLOBAL
[FORWARD_ONLY | SCROLL]
[STATIC | KEYSET | DYNAMIC | FAST_FORWARD]
[READ_ONLY | SCROLL_LOCKS | OPTIMISTIC]
[TYPE_WARNING]
FOR select_statement
[FOR UPDATE [OF column_name [,...n]]]

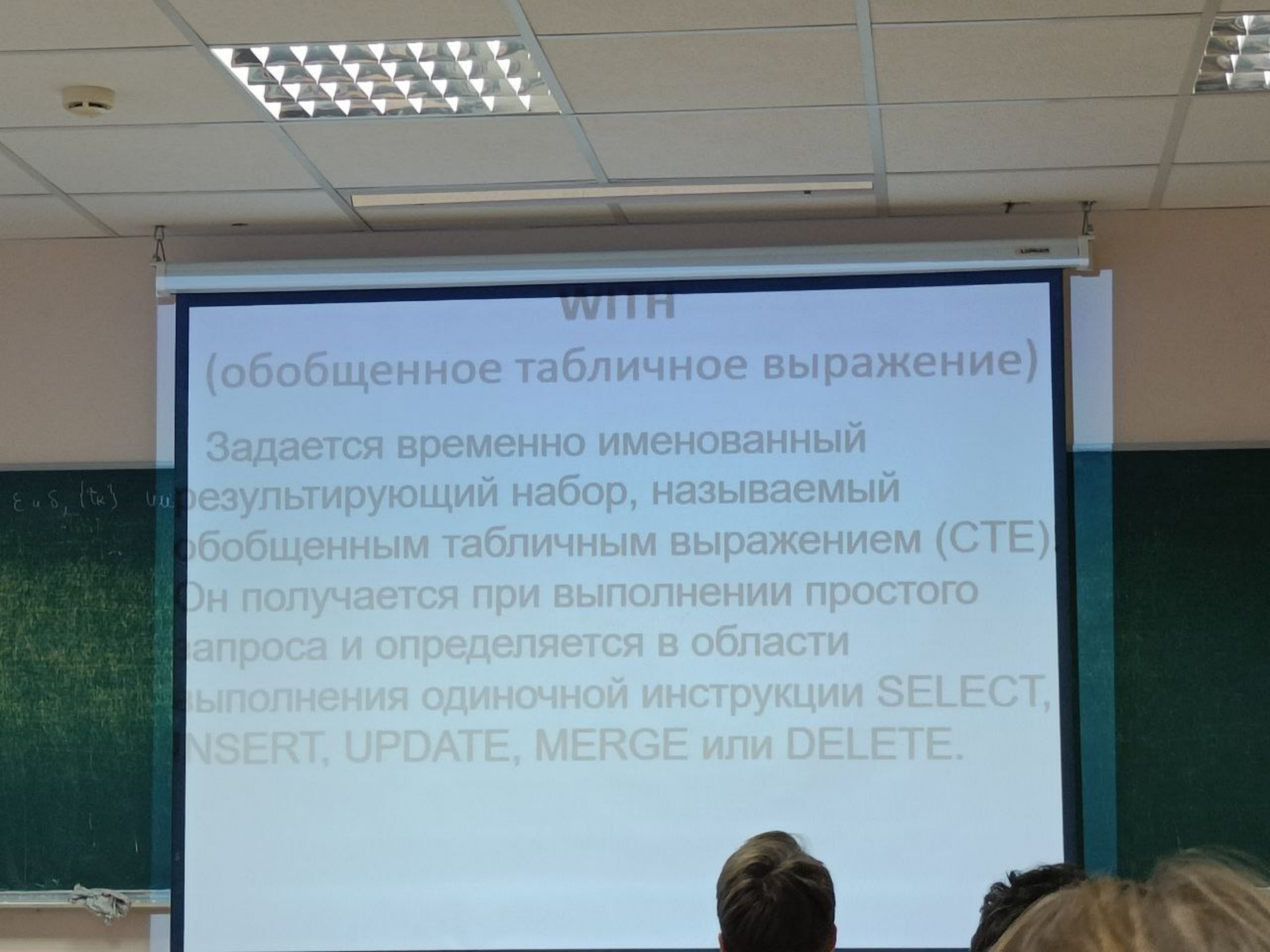

Инструкция DECLARE CURSOR определяет такие атрибуты серверного курсора языка Transact-SQL, как свойства просмотра и запрос, используемый для построения результирующего набора, на котором работает курсор.

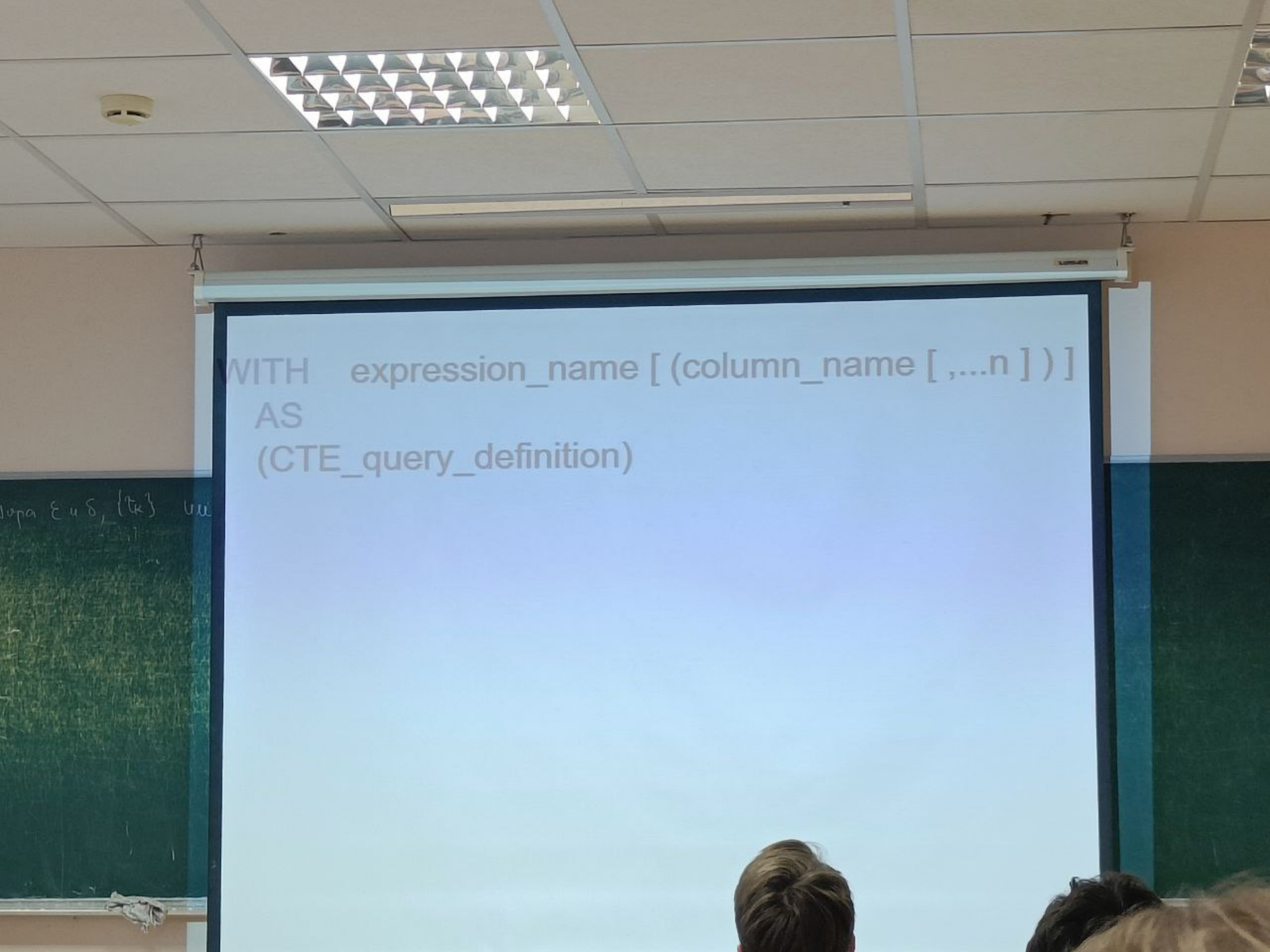

Инструкция **OPEN** заполняет результирующий набор, а оператор **FETCH** возвращает из него строку.
Инструкция **CLOSE** очищает текущий результирующий набор, связанный с курсором.
Инструкция **DEALLOCATE** освобождает ресурсы, используемые курсором.


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


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


@@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 от


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.


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