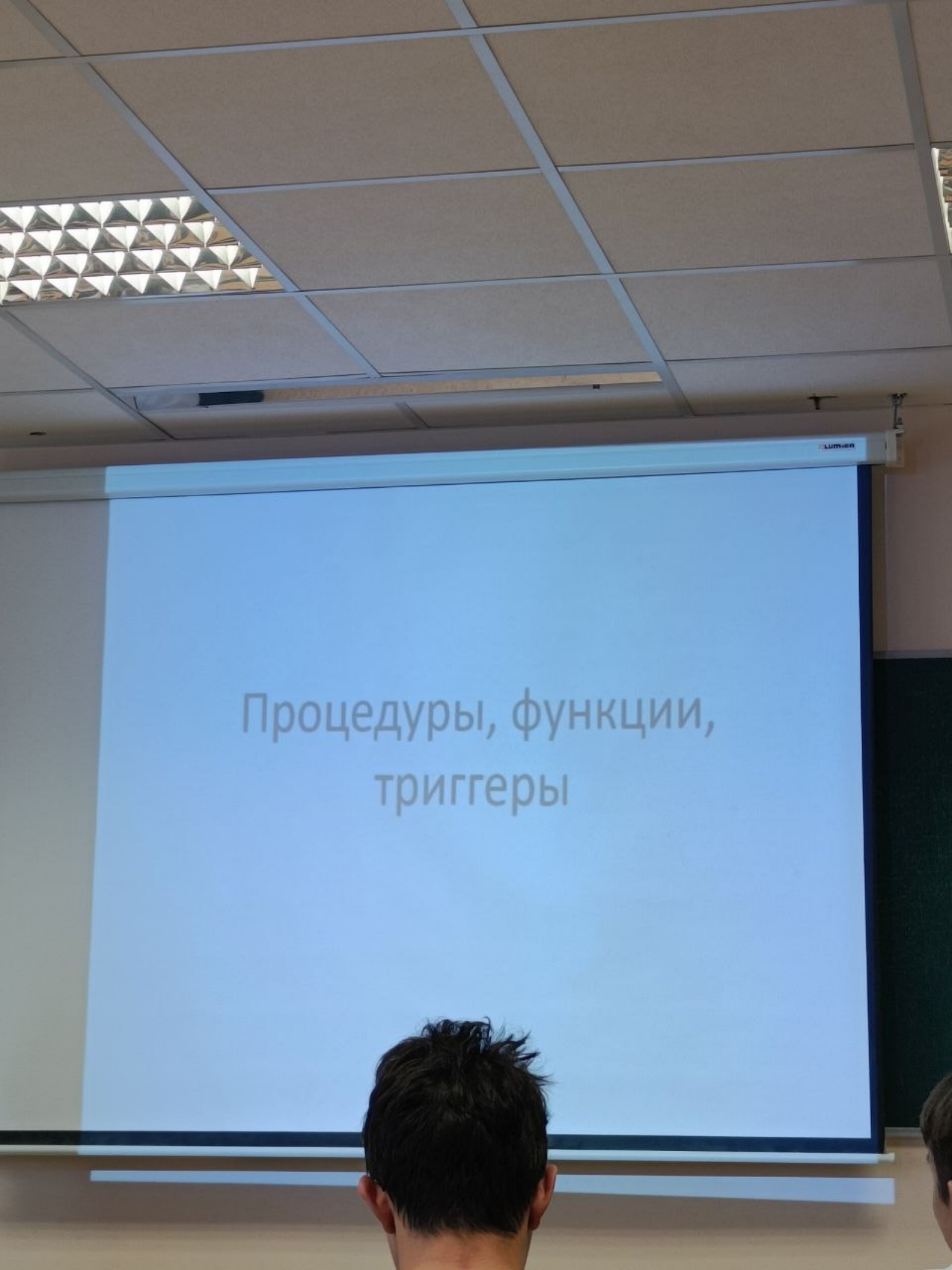
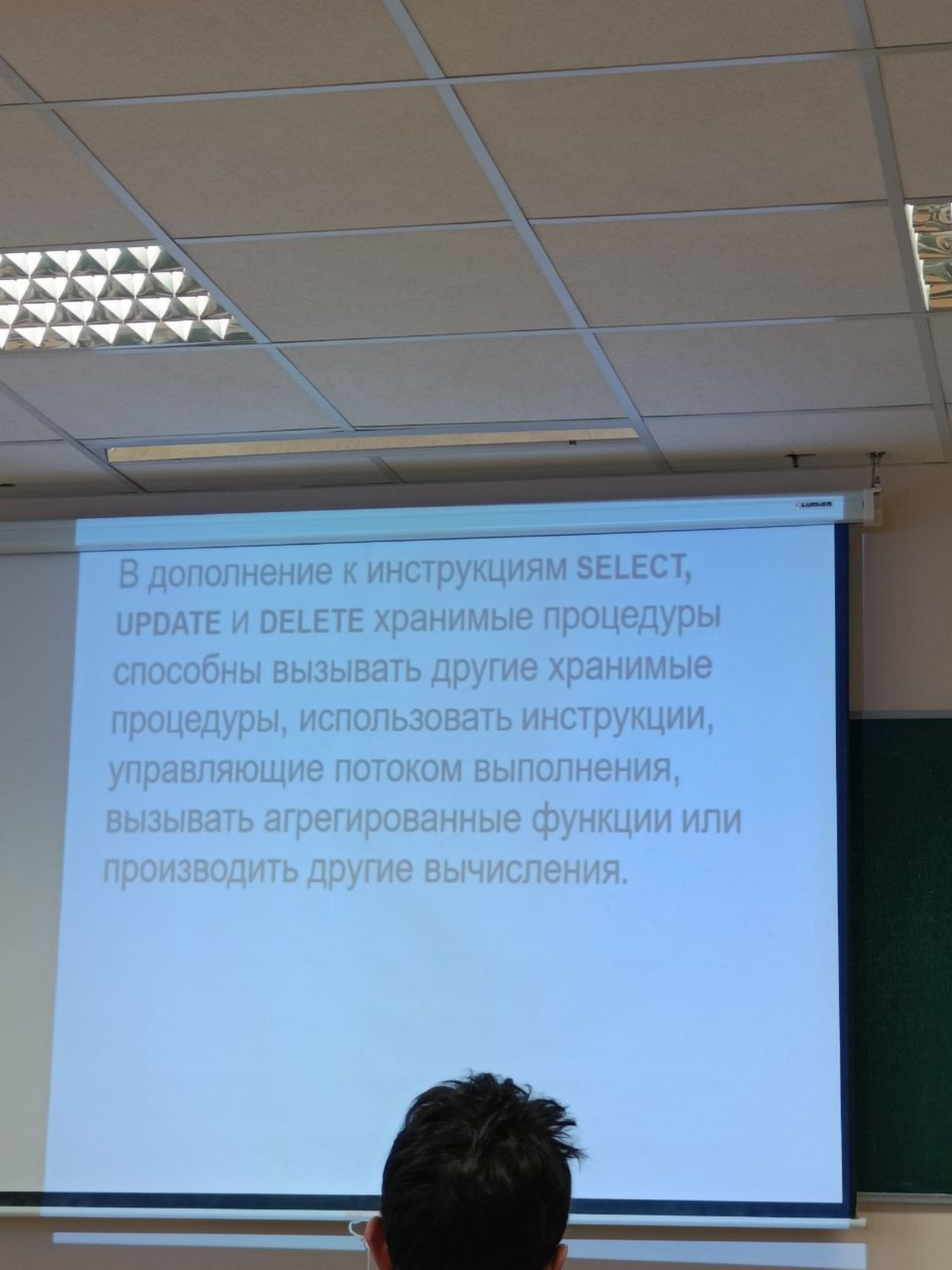
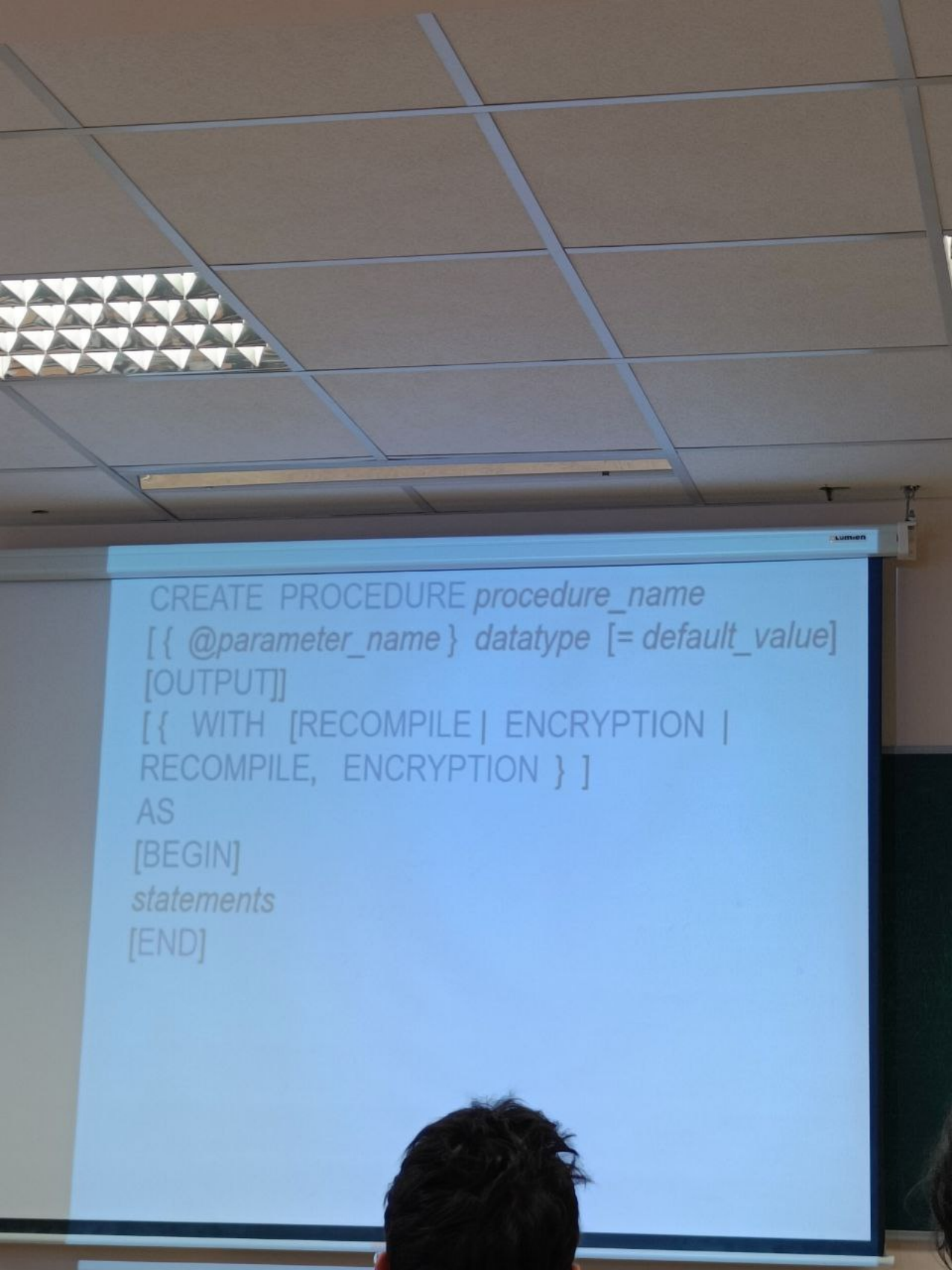

Процедуры, функции,
триггеры

Хранимые процедуры

В самых простых терминах *хранимая процедура*—это набор скомпилированных инструкций T-SQL, непосредственно доступный в SQL Server. Инструкции, помещенные в хранимую процедуру, выполняются как отдельная единица, или *пакет* (batch), обработки— преимущество пакета состоит в значительном уменьшении сетевого трафика.


В дополнение к инструкциям SELECT, UPDATE и DELETE хранимые процедуры способны вызывать другие хранимые процедуры, использовать инструкции, управляющие потоком выполнения, вызывать агрегированные функции или производить другие вычисления.

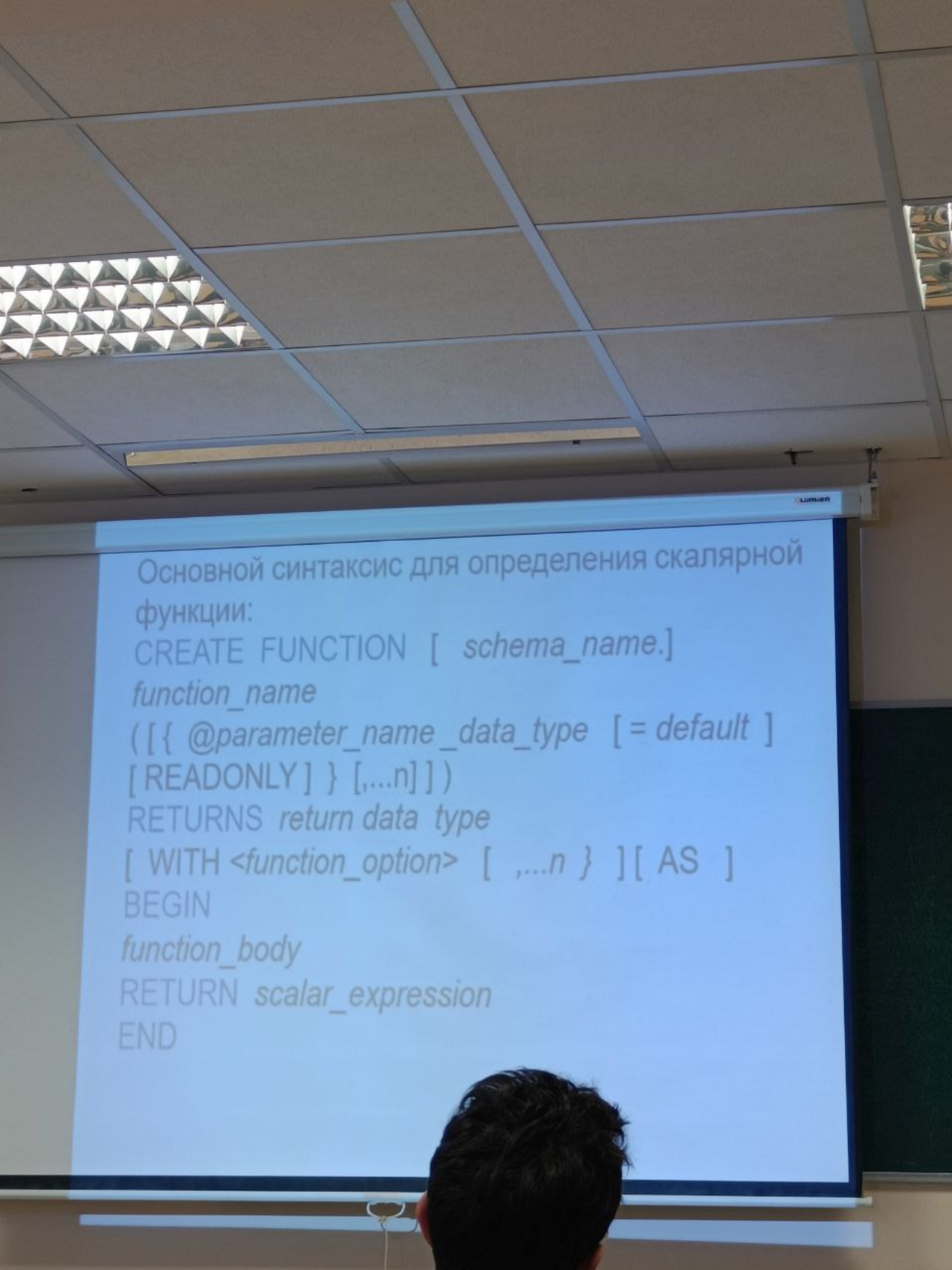

CREATE PROCEDURE procedure_name
[{ @parameter_name } datatype [= default_value]
[OUTPUT]]
[{ WITH [RECOMPILE | ENCRYPTION |
RECOMPILE, ENCRYPTION }]
AS
[BEGIN]
statements
[END]

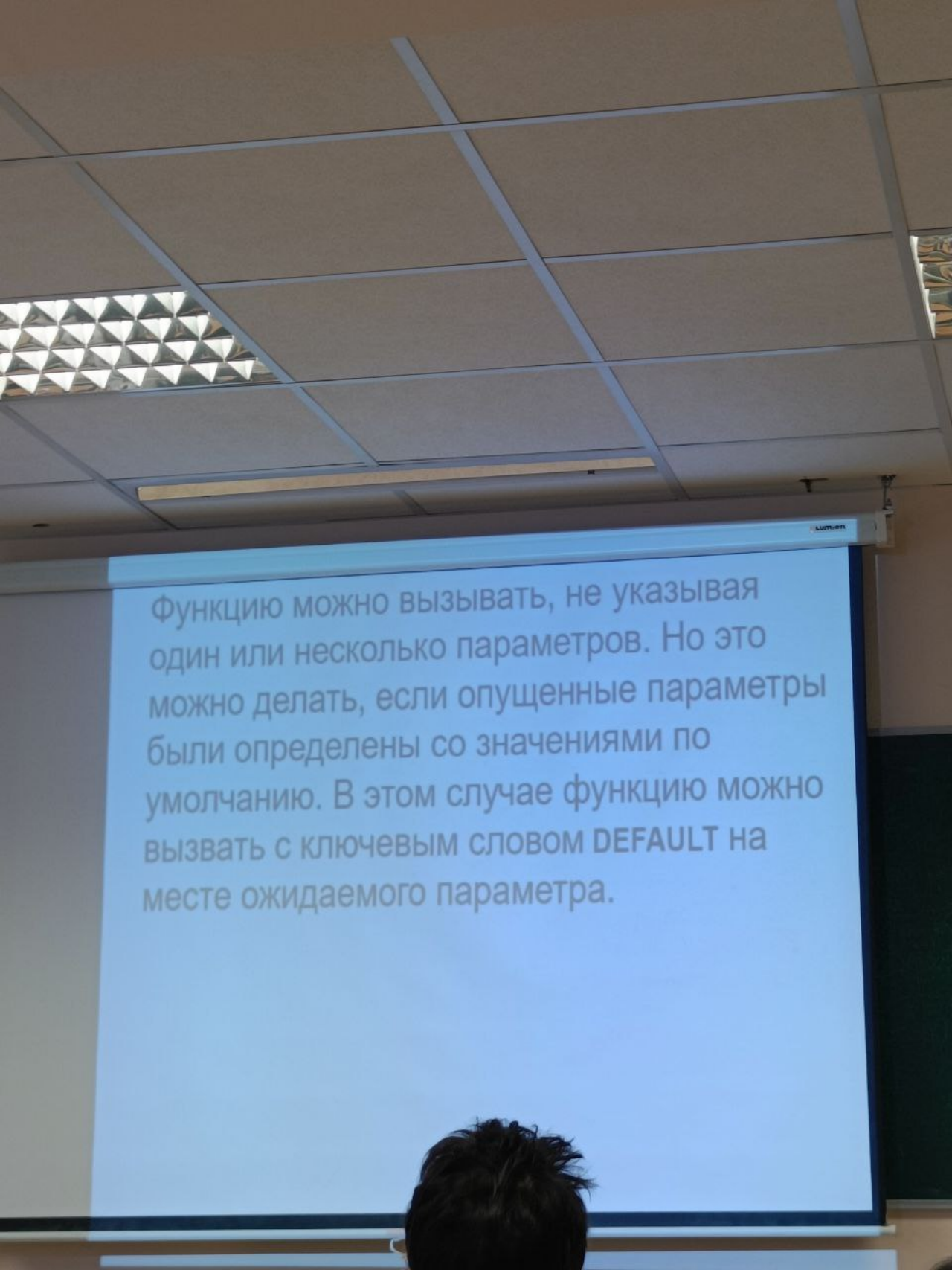
Хранимая процедура принимает набор данных, выполняет требуемую обработку, а затем завершается. Невозможно взять хранимую процедуру и выполнить ее, например, в инструкции *SELECT*.

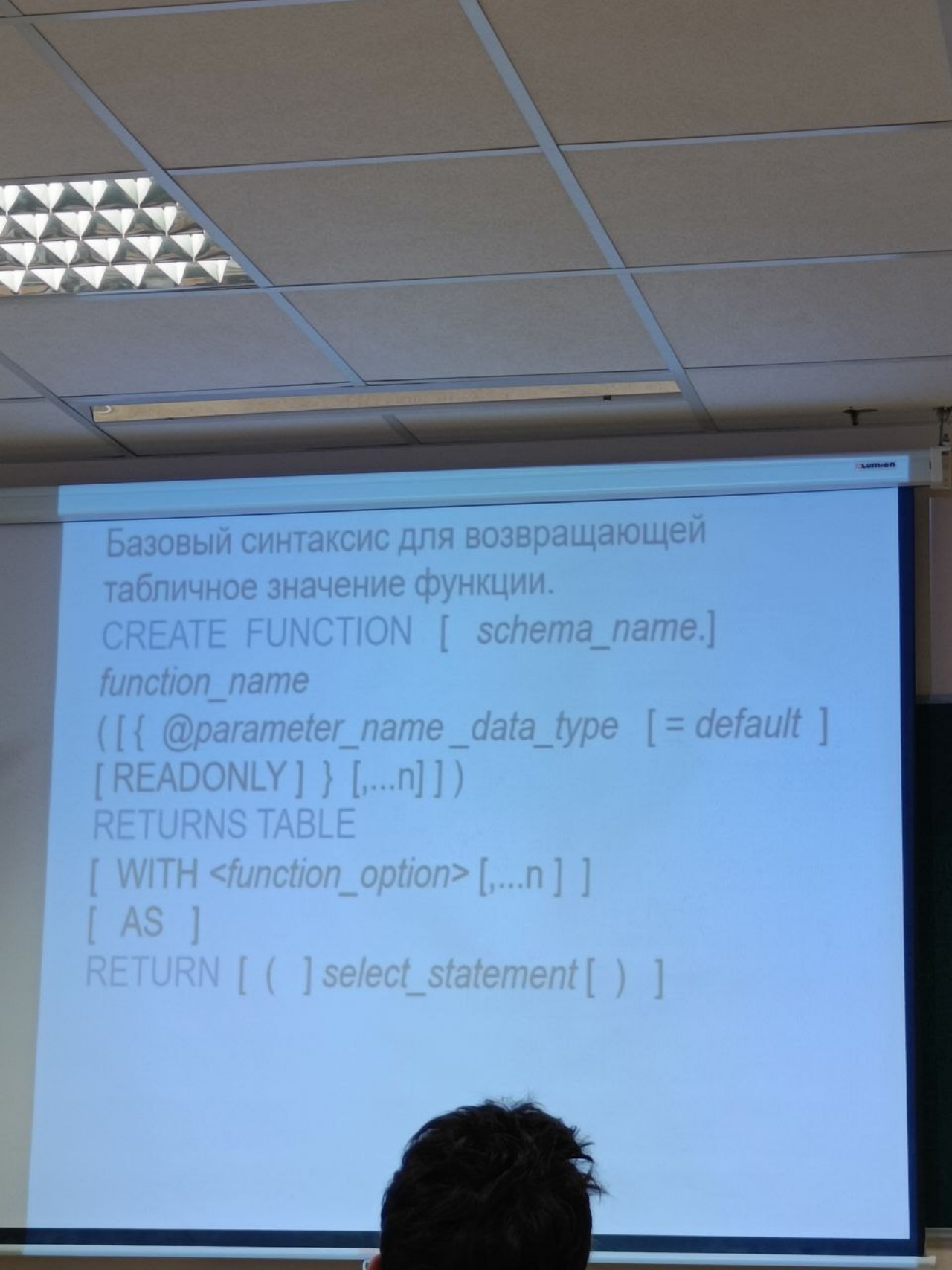
Пользовательские функции

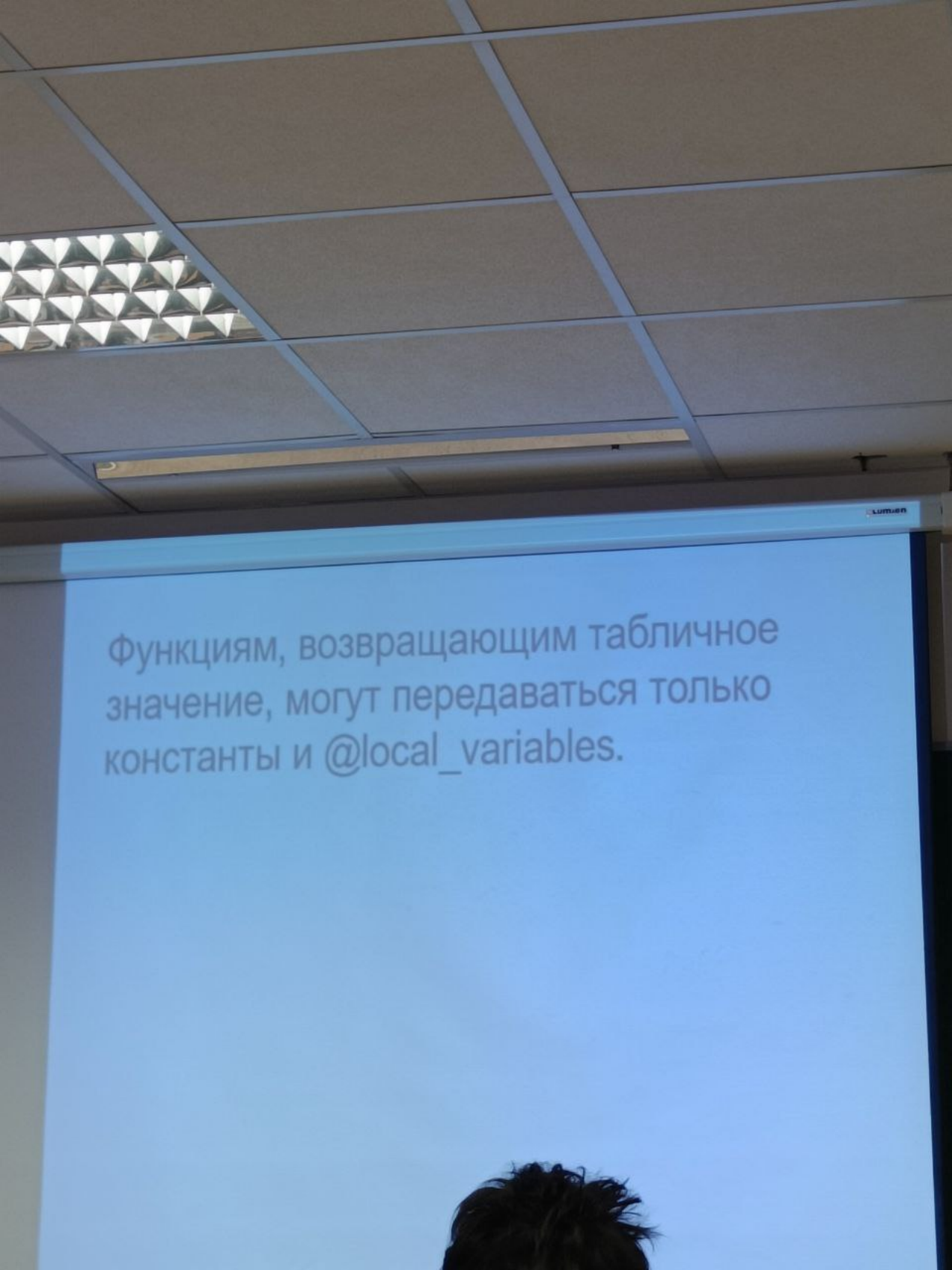
В таких случаях применяются *определяемые пользователем функции* (UDF, user-defined functions).

Функции бывают двух типов: скалярные функции (возвращающие единственное значение) и возвращающие табличное значение функции (table-valued).


Основной синтаксис для определения скалярной функции:
CREATE FUNCTION [schema_name.]
function_name
([{ @parameter_name _data_type [= default]
[READONLY] } [,...n]])
RETURNS return data type
[WITH <function_option> [,...n }] [AS]
BEGIN
function_body
RETURN scalar_expression
END


Функцию можно вызывать, не указывая один или несколько параметров. Но это можно делать, если опущенные параметры были определены со значениями по умолчанию. В этом случае функцию можно вызвать с ключевым словом DEFAULT на месте ожидаемого параметра.

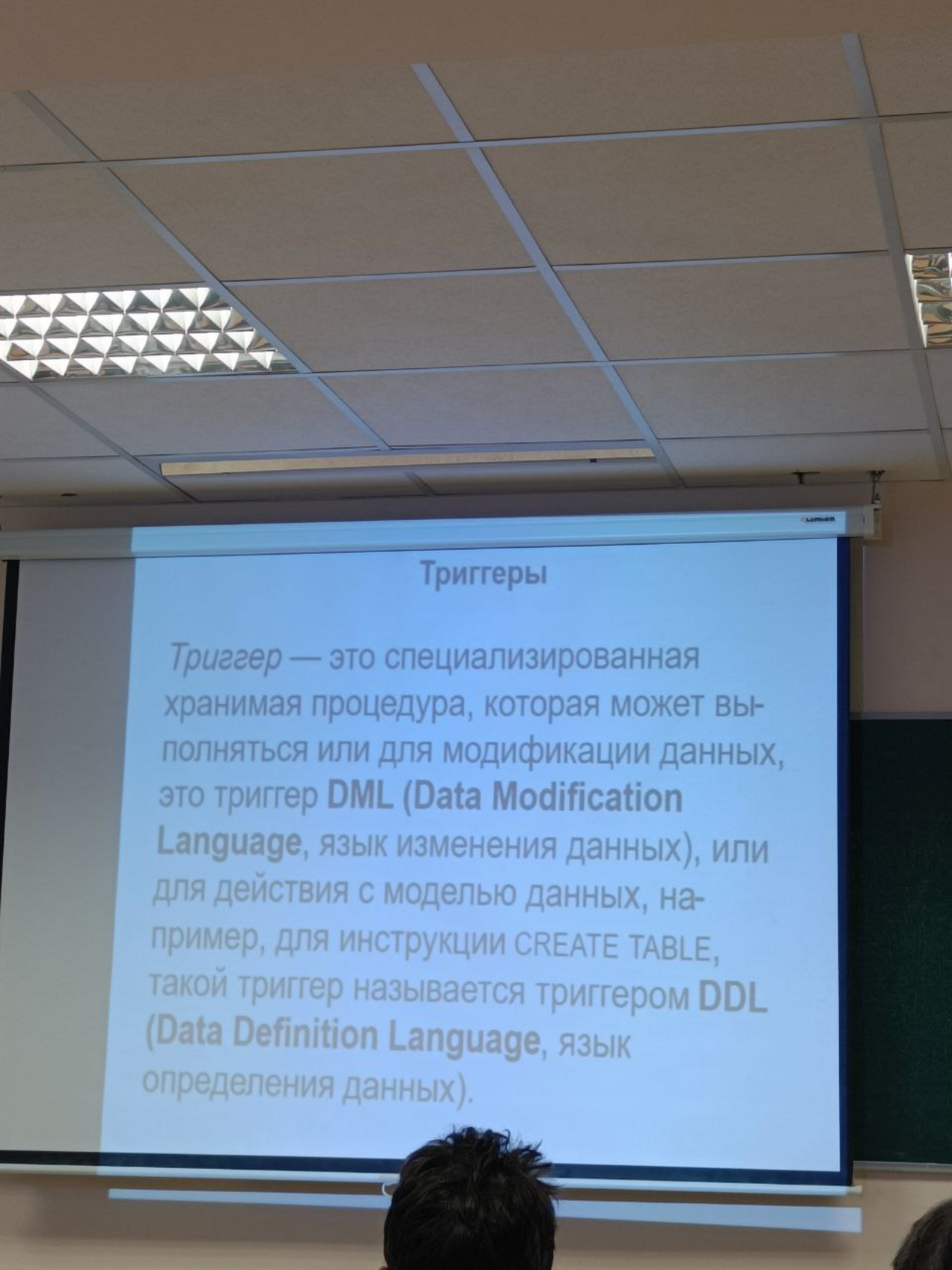

Базовый синтаксис для возвращающей
табличное значение функции.
CREATE FUNCTION [schema_name.]
function_name
([{ @parameter_name _data_type [= default]
[READONLY] } [,...n]])
RETURNS TABLE
[WITH <function_option> [,...n]]
[AS]
RETURN [( ] select_statement [ )]

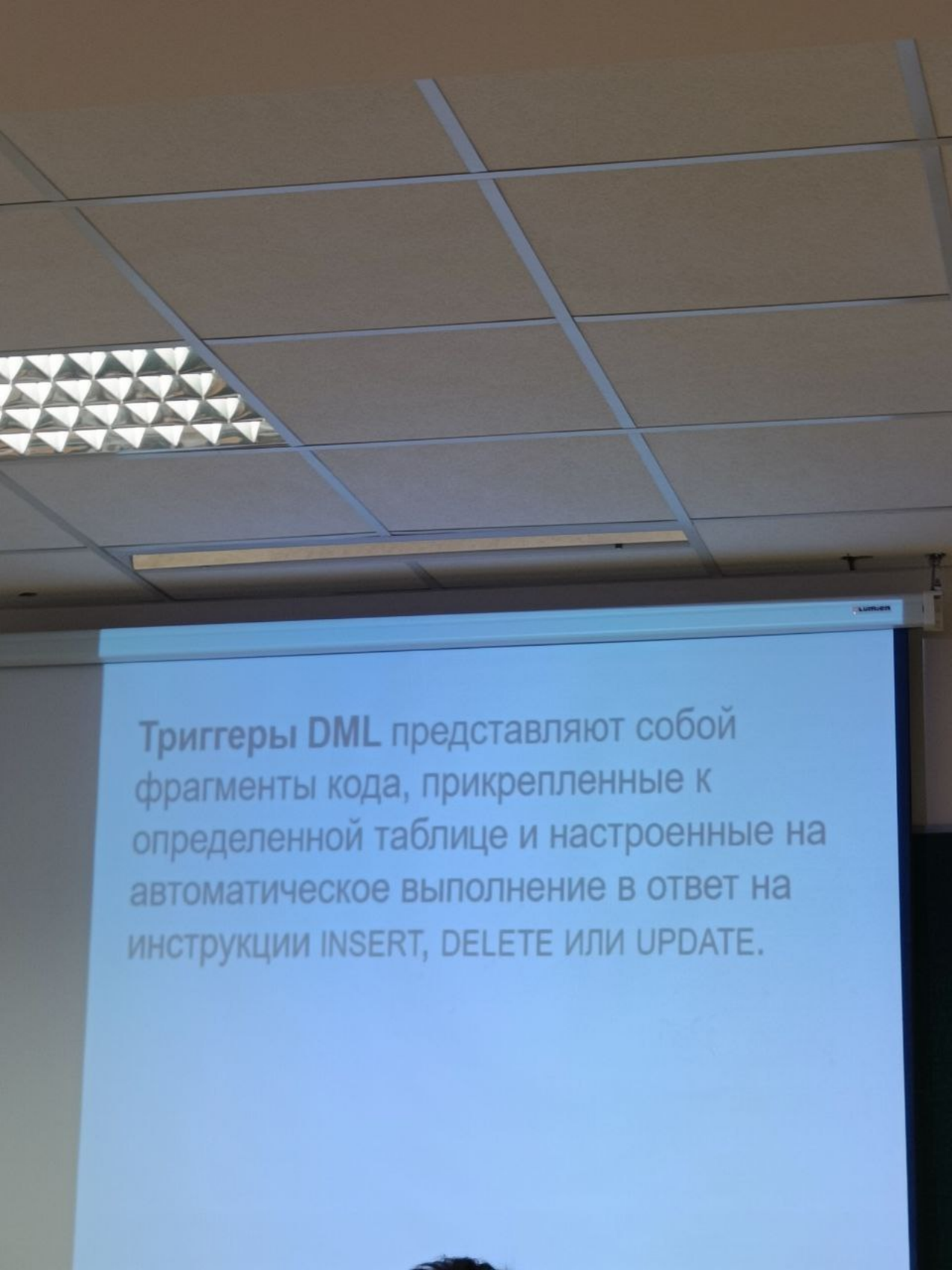

Функциям, возвращающим табличное значение, могут передаваться только константы и @local_variables.

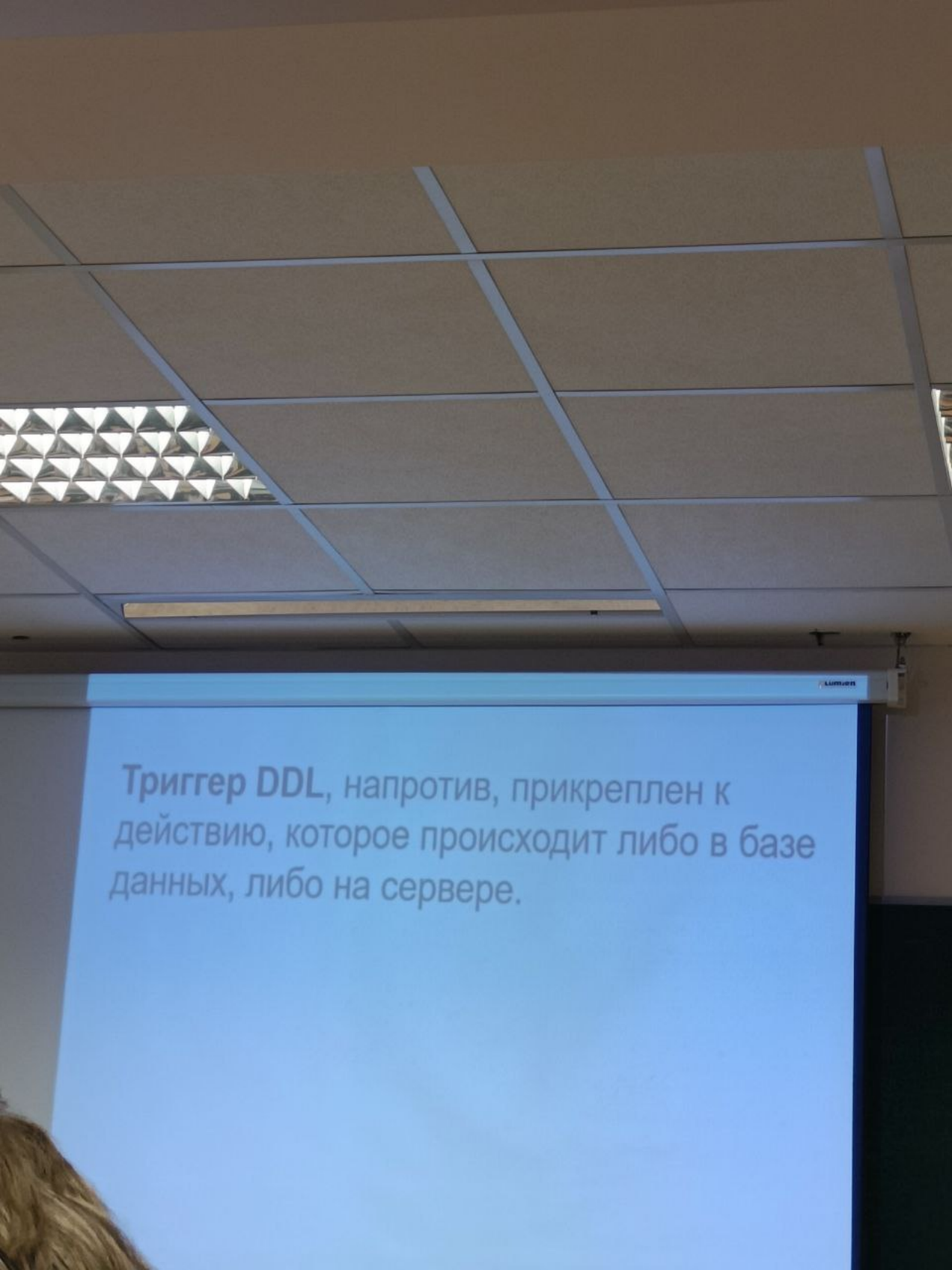
При помощи этих инструкций можно создать подпрограмму, которую можно повторно использовать следующими способами.

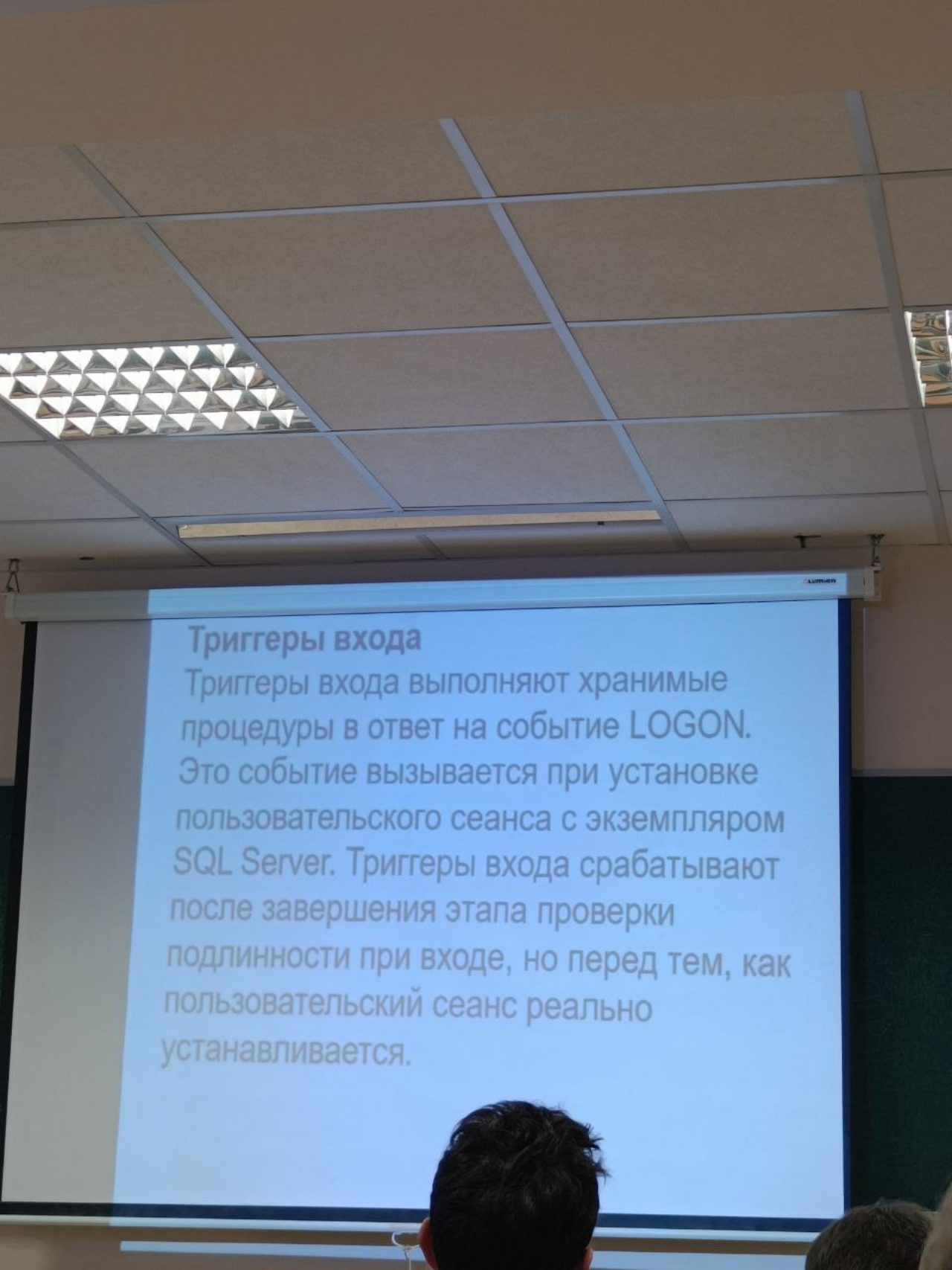
1. В инструкциях Transact-SQL, например SELECT.
2. В приложениях, вызывающих функцию.
3. В определении другой пользовательской функции.

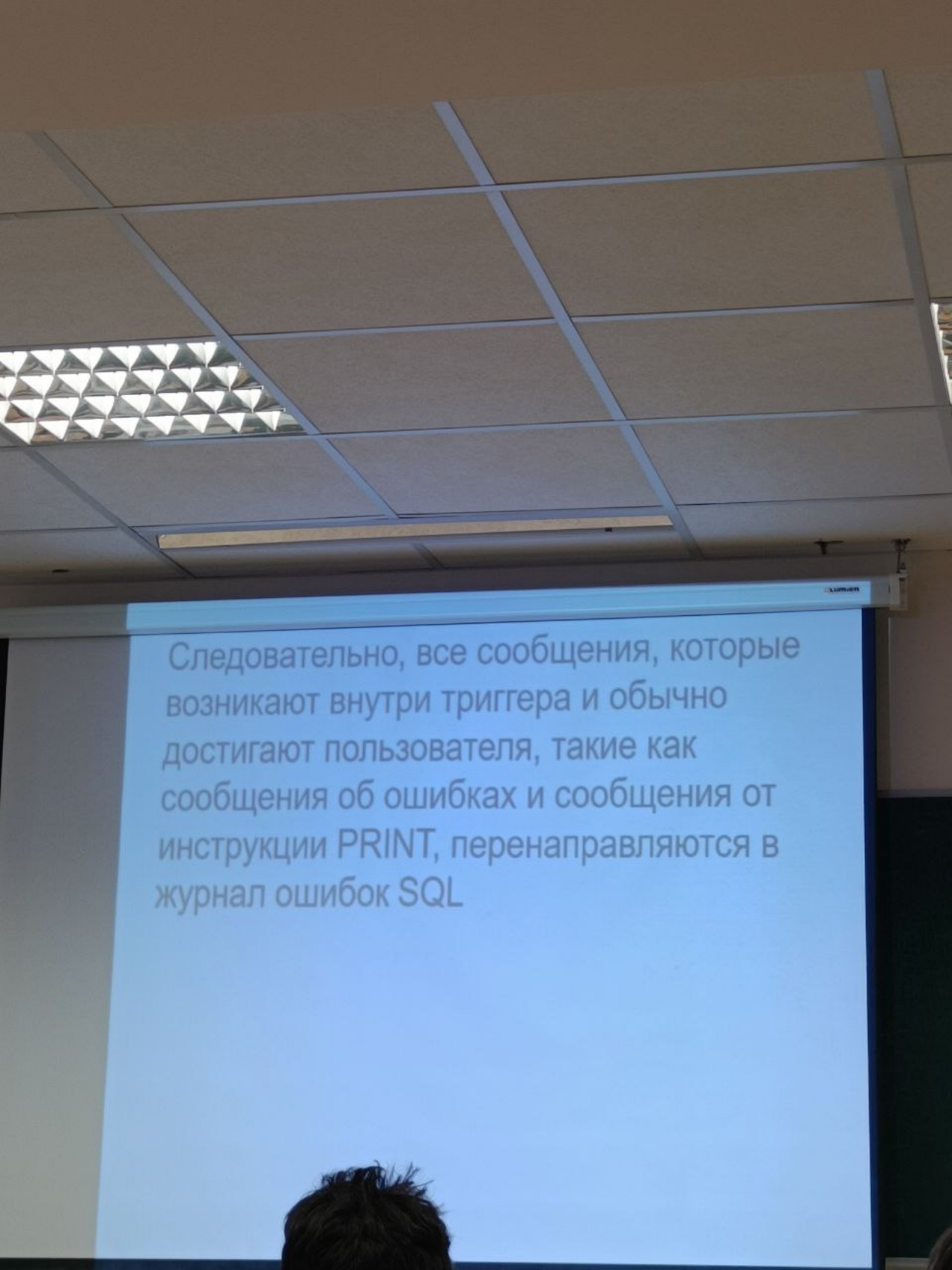
4. Для параметризации представления или улучшения функциональности индексированного представления.
5. Для определения столбца таблицы.
6. Для определения ограничения CHECK на столбец.
7. Для замены хранимой процедуры.


Триггеры
Триггер — это специализированная хранимая процедура, которая может выполняться или для модификации данных, это триггер **DML (Data Modification Language**, язык изменения данных), или для действия с моделью данных, например, для инструкции CREATE TABLE, такой триггер называется триггером **DDL (Data Definition Language**, язык определения данных).


Триггеры DML представляют собой фрагменты кода, прикрепленные к определенной таблице и настроенные на автоматическое выполнение в ответ на инструкции INSERT, DELETE ИЛИ UPDATE.


Триггер DDL, напротив, прикреплен к действию, которое происходит либо в базе данных, либо на сервере.


Триггеры входа
Триггеры входа выполняют хранимые процедуры в ответ на событие LOGON. Это событие вызывается при установке пользовательского сеанса с экземпляром SQL Server. Триггеры входа срабатывают после завершения этапа проверки подлинности при входе, но перед тем, как пользовательский сеанс реально устанавливается.


Следовательно, все сообщения, которые возникают внутри триггера и обычно достигают пользователя, такие как сообщения об ошибках и сообщения от инструкции PRINT, перенаправляются в журнал ошибок SQL

В отличие от хранимых процедур

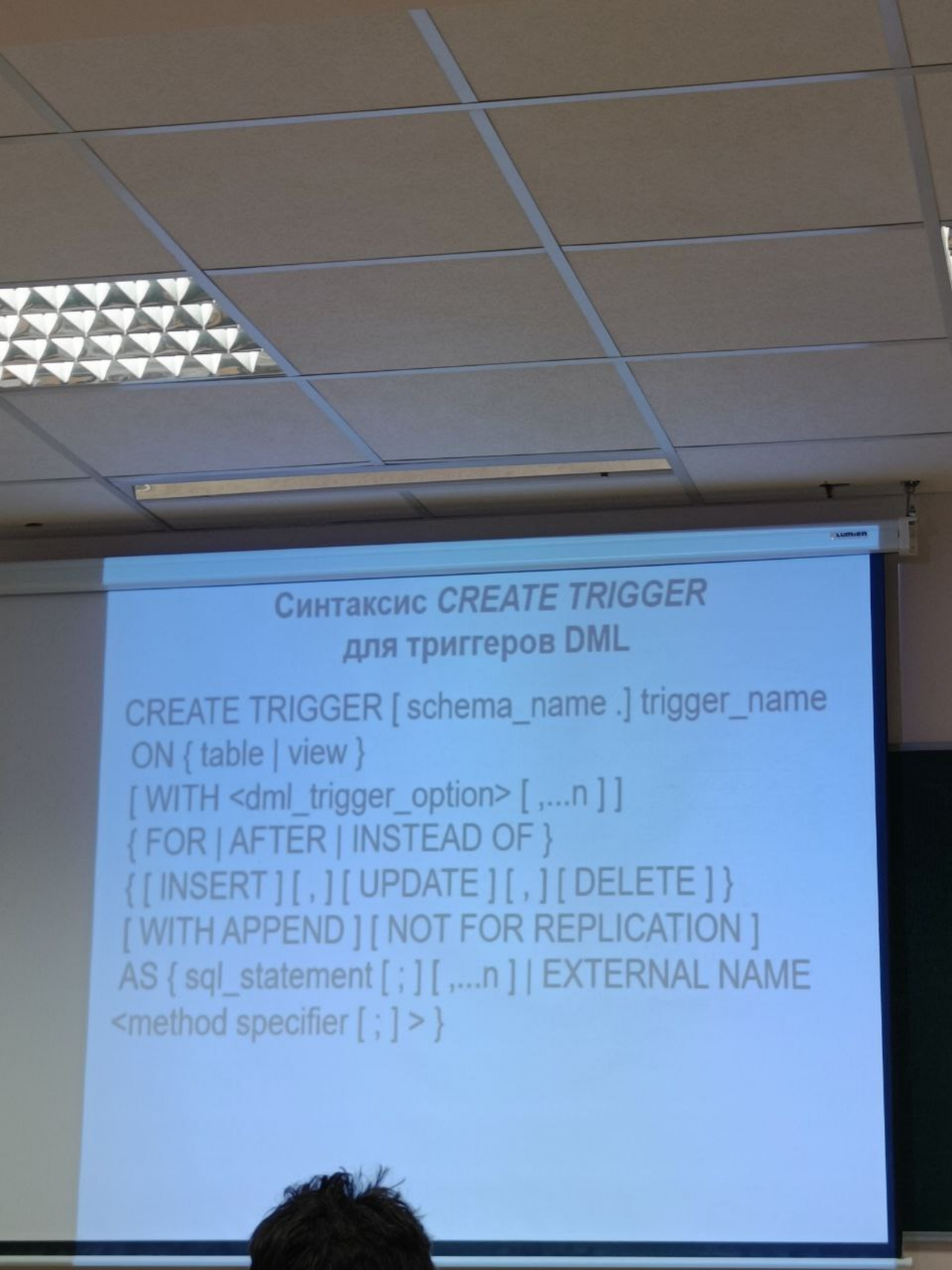
- триггер невозможно запустить вручную,
- с триггерами нельзя использовать параметры,
- из триггеров невозможно вернуть значения.

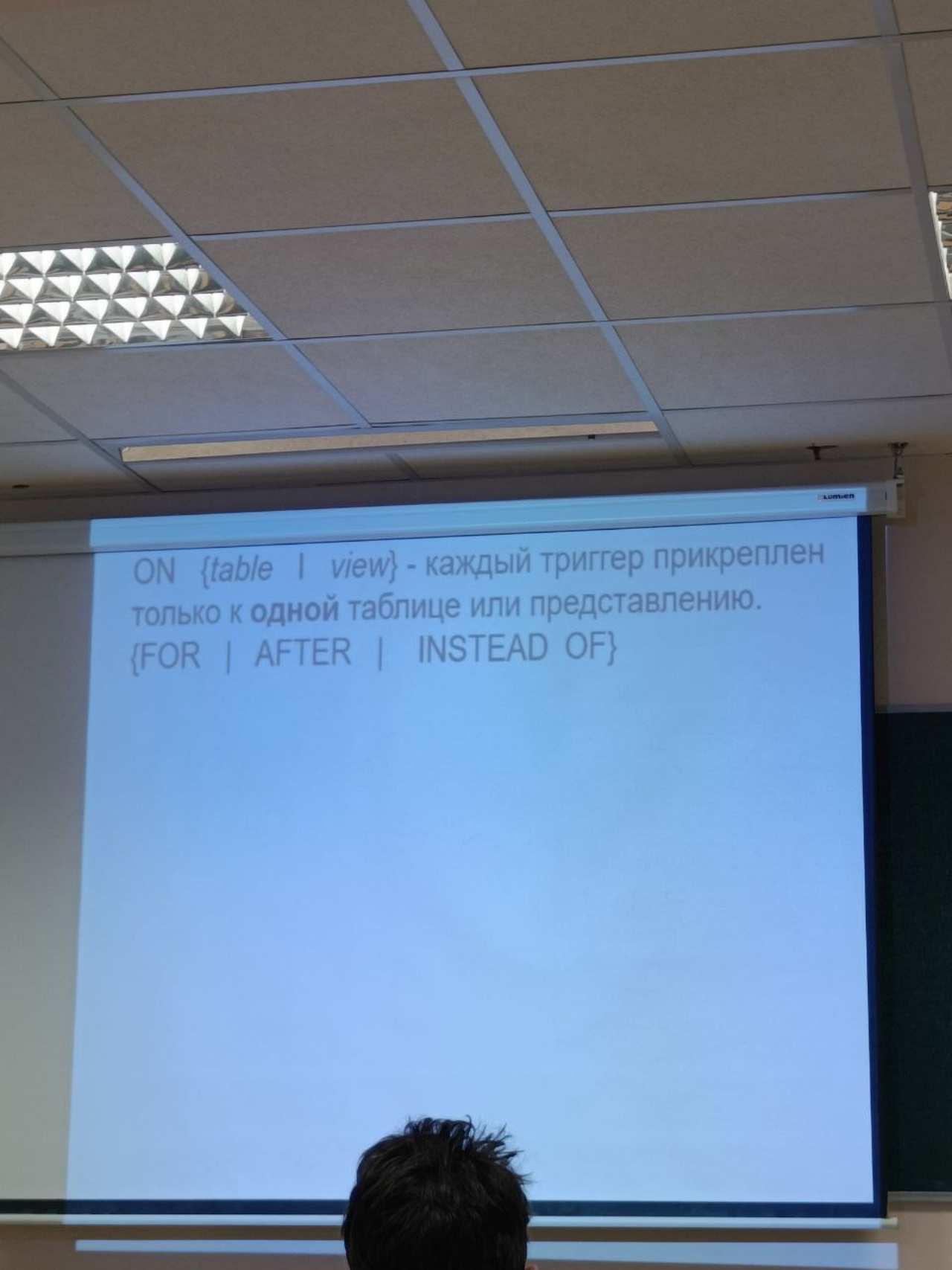
Для любого действия с таблицами за исключением SELECT можно создать отдельные триггеры, а также триггеры, которые будут запускаться по какой-либо комбинации действий.

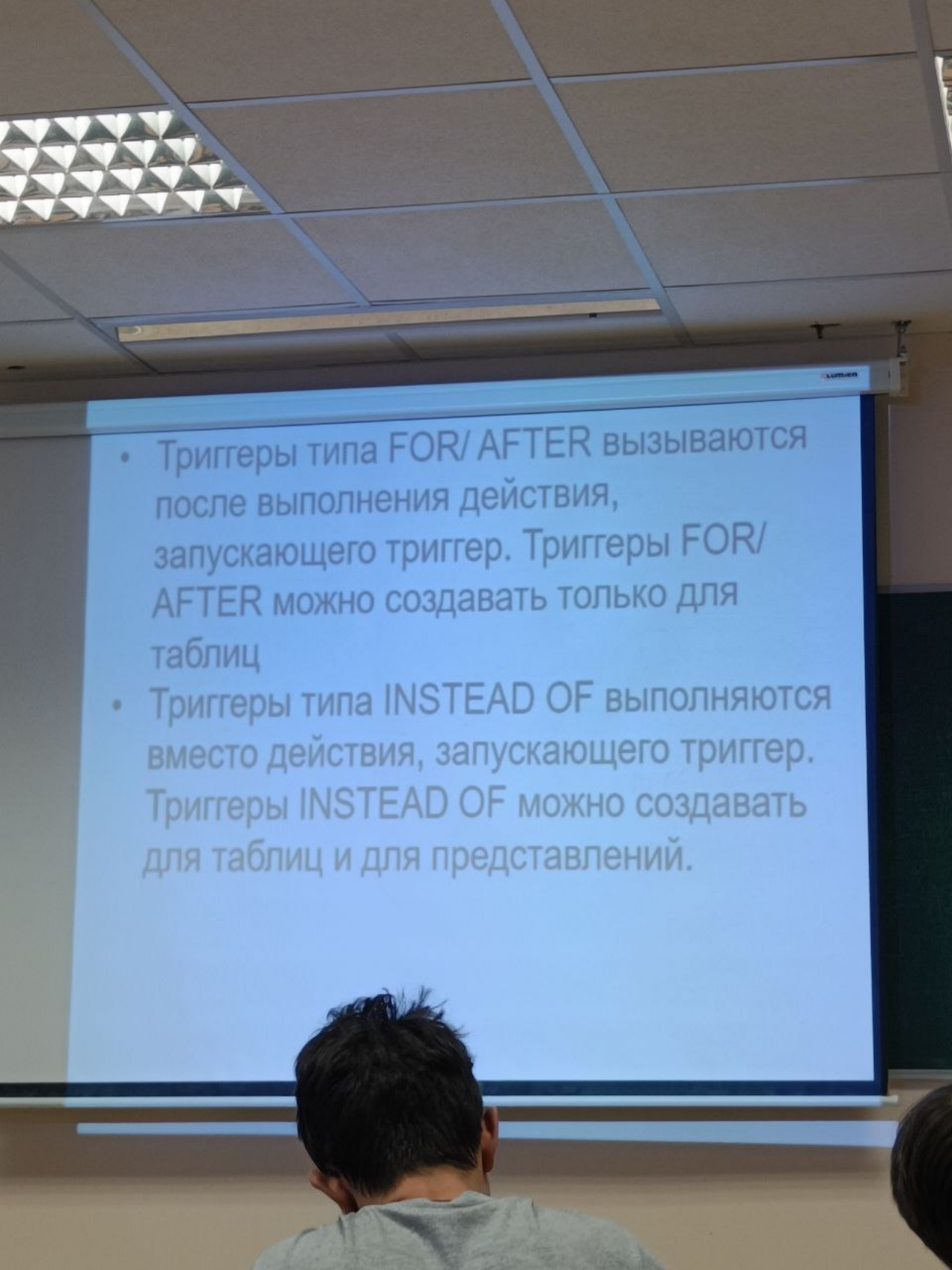
Существуют три основных типа триггеров:

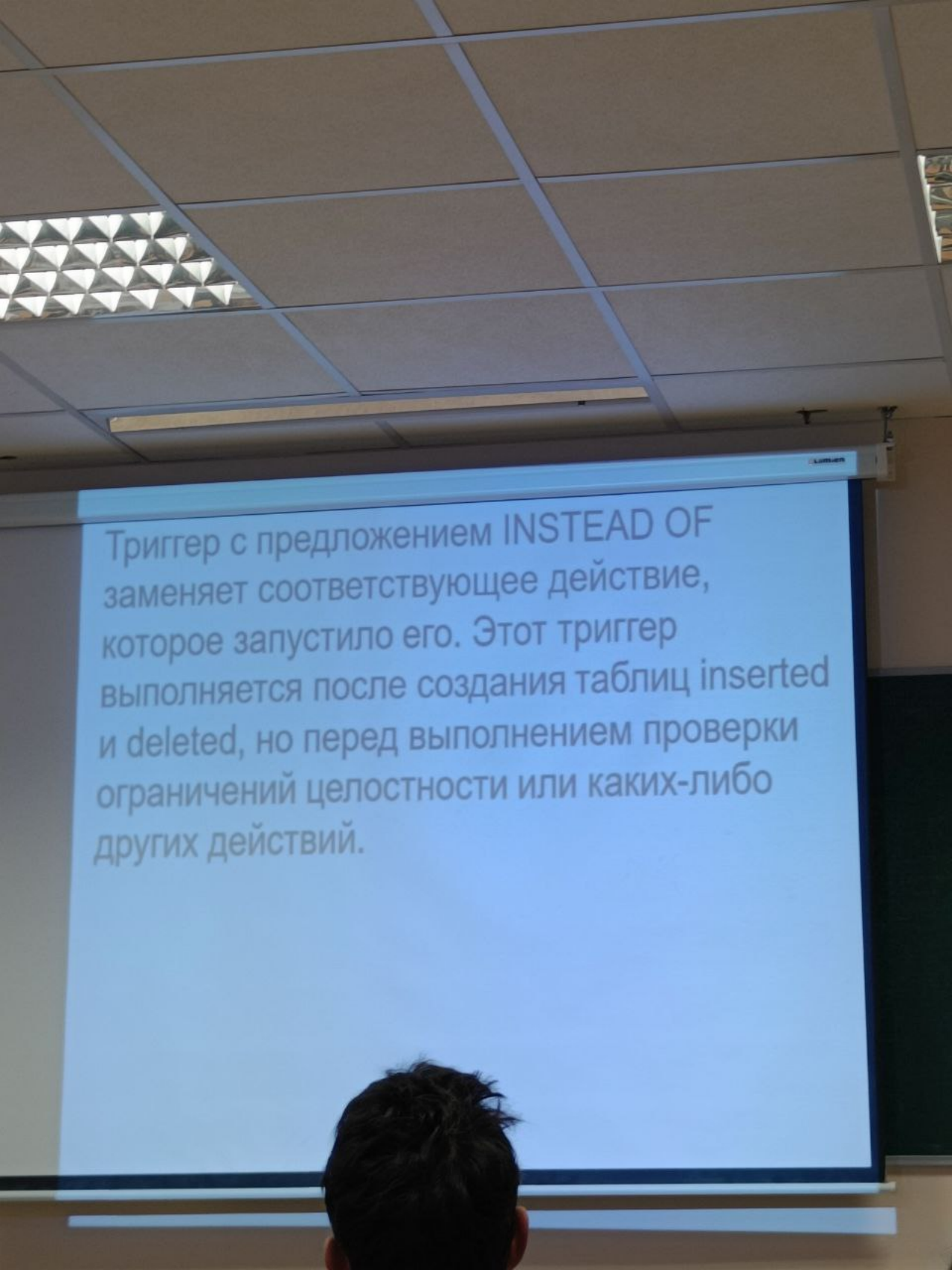
- триггер **INSERT**;
- триггер **DELETE**;
- триггер **UPDATE**.

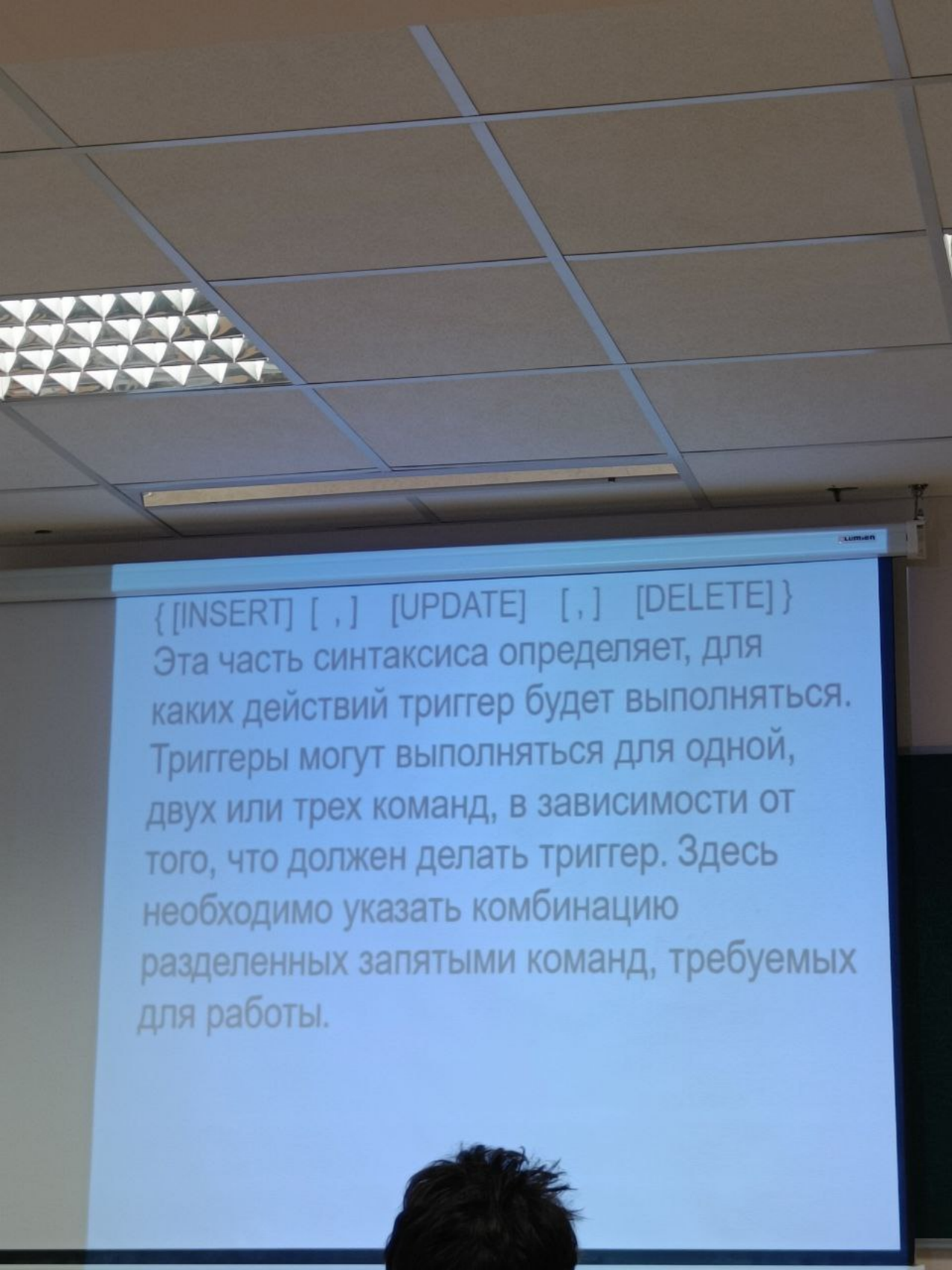
Типы триггеров можно комбинировать.


Синтаксис CREATE TRIGGER
для триггеров DML
CREATE TRIGGER [schema_name .] trigger_name
ON { table | view }
[WITH <dml_trigger_option> [,...n]]
{ FOR | AFTER | INSTEAD OF }
{ [INSERT] [,] [UPDATE] [,] [DELETE] }
[WITH APPEND] [NOT FOR REPLICATION]
AS { sql_statement [;] [,...n] | EXTERNAL NAME
<method specifier [;] > }

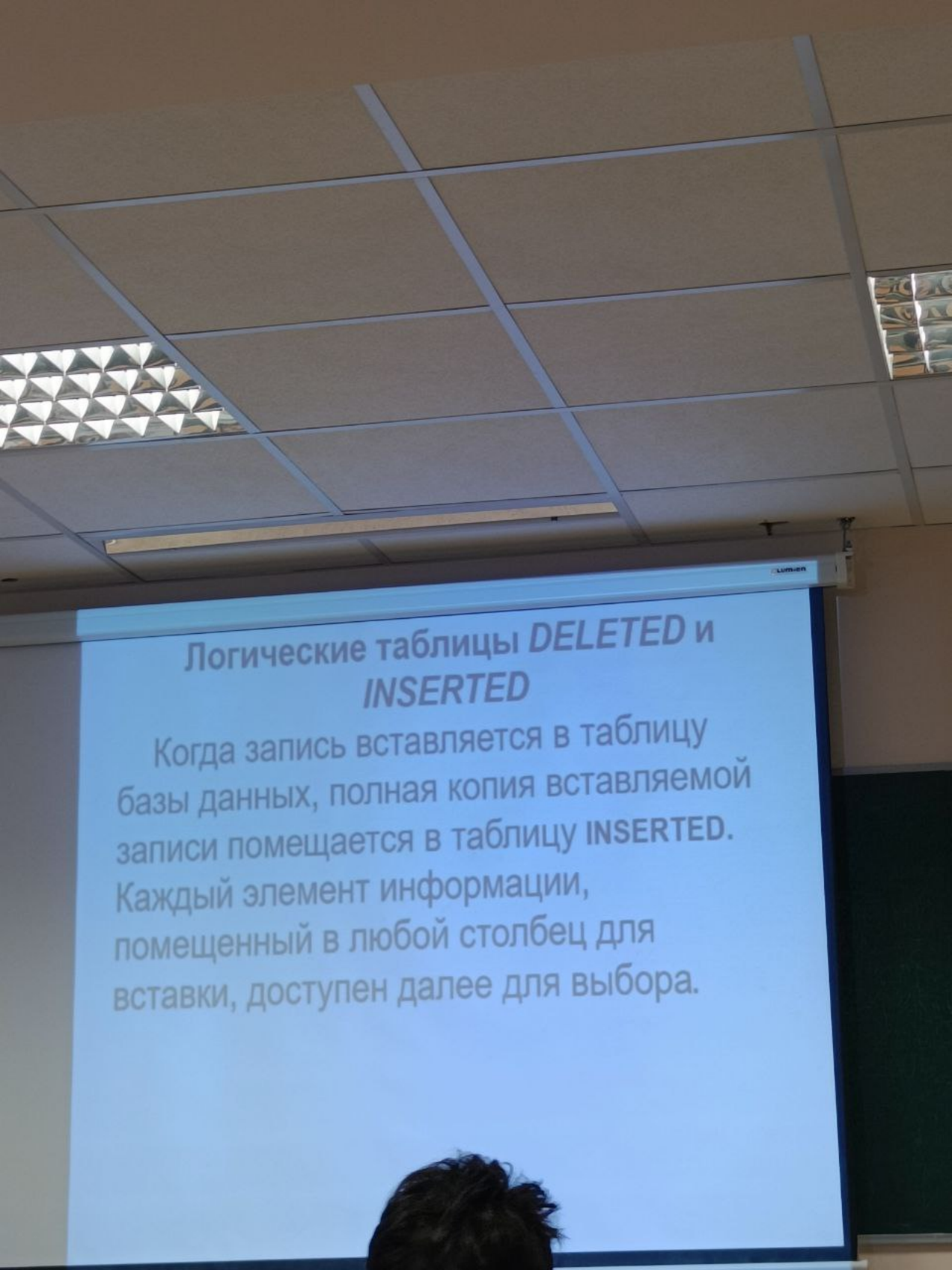

ON {table | view} - каждый триггер прикреплен только к **одной** таблице или представлению.
{FOR | AFTER | INSTEAD OF}


• Триггеры типа FOR/ AFTER вызываются после выполнения действия, запускающего триггер. Триггеры FOR/ AFTER можно создавать только для таблиц
• Триггеры типа INSTEAD OF выполняются вместо действия, запускающего триггер. Триггеры INSTEAD OF можно создавать для таблиц и для представлений.


Триггер с предложением INSTEAD OF заменяет соответствующее действие, которое запустило его. Этот триггер выполняется после создания таблиц inserted и deleted, но перед выполнением проверки ограничений целостности или каких-либо других действий.


{ [INSERT] [,] [UPDATE] [,] [DELETE] }
Эта часть синтаксиса определяет, для каких действий триггер будет выполняться. Триггеры могут выполняться для одной, двух или трех команд, в зависимости от того, что должен делать триггер. Здесь необходимо указать комбинацию разделенных запятыми команд, требуемых для работы.

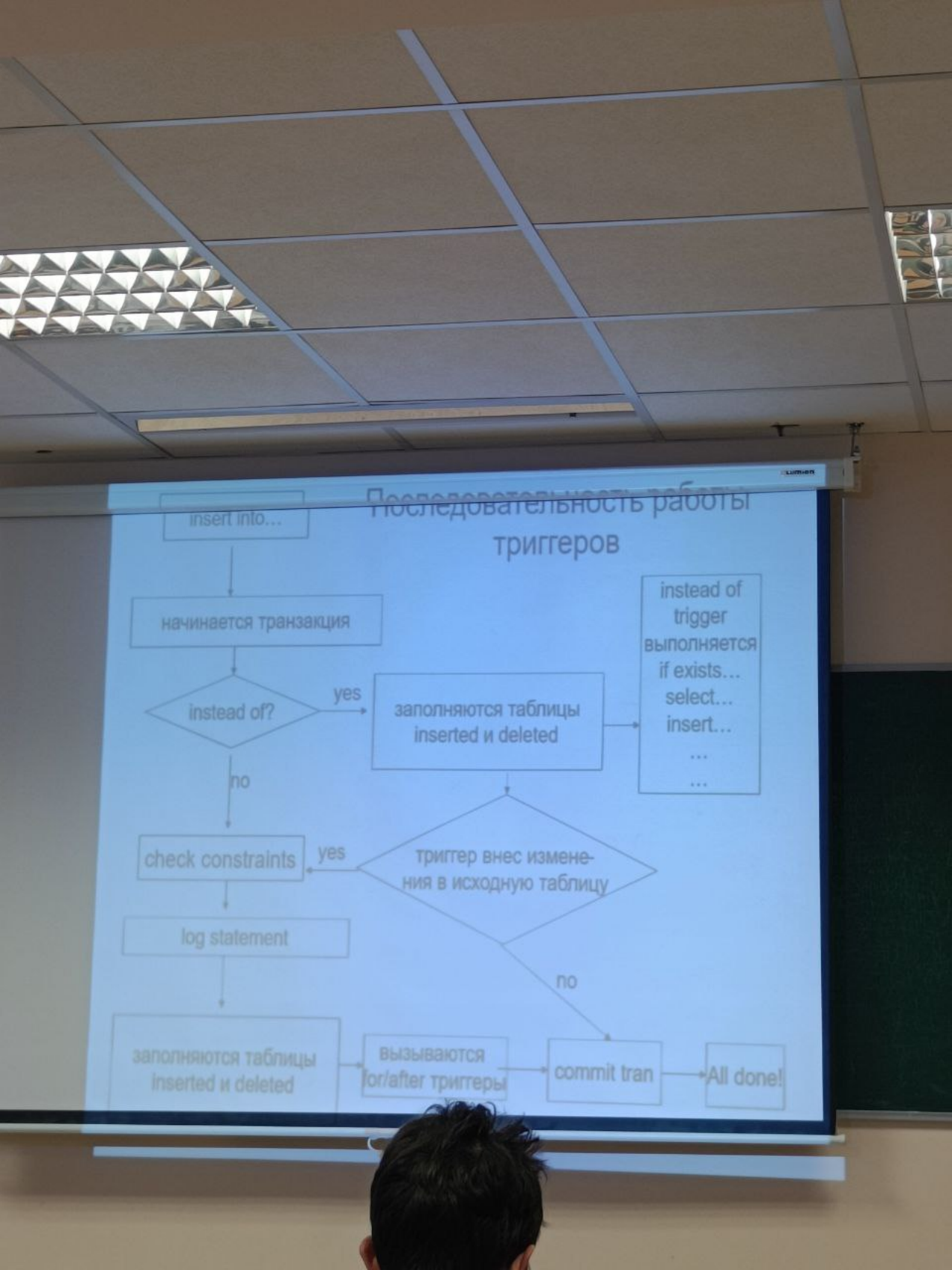
AS. Ключевое слово AS определяет начало кода триггера, точно так же, как ключевое слово AS определяет начало хранимой процедуры. Ведь триггер - это просто *специализированная хранимая процедура*.

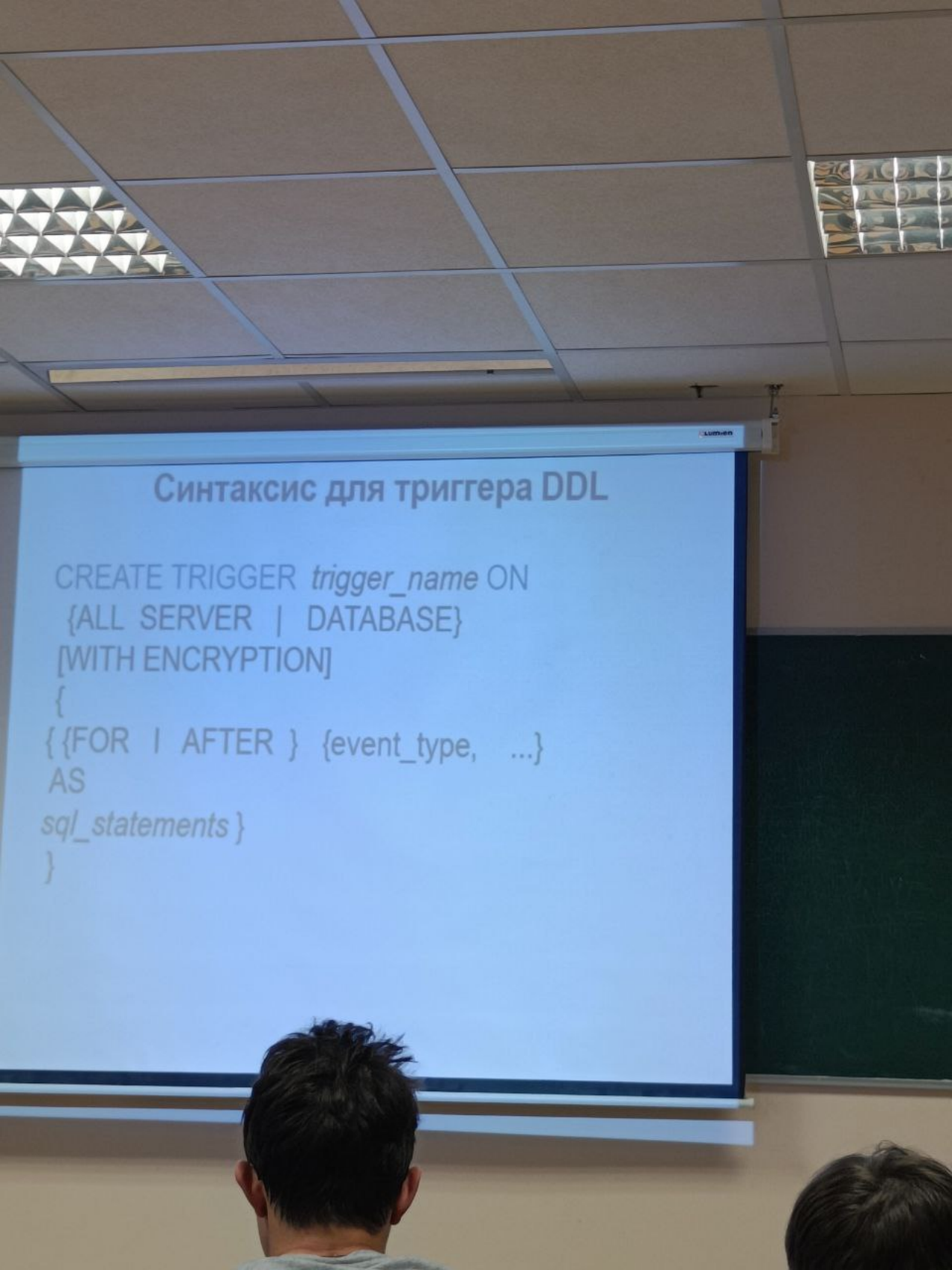

Логические таблицы DELETED и INSERTED
Когда запись вставляется в таблицу базы данных, полная копия вставляемой записи помещается в таблицу INSERTED. Каждый элемент информации, помещенный в любой столбец для вставки, доступен далее для выбора.

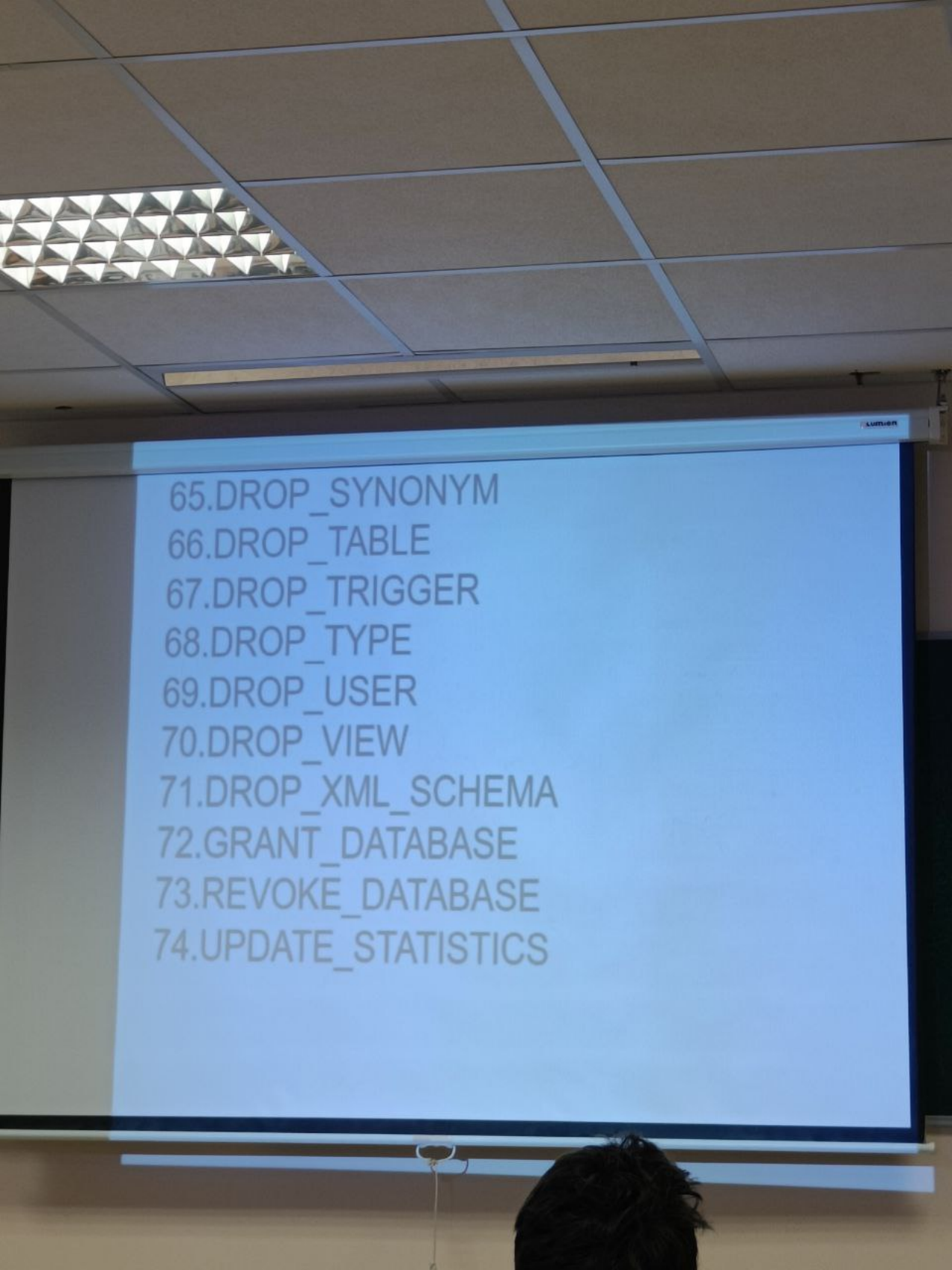
Если выполняется удаление, то запись строки данных помещается в таблицу **DELETED**.

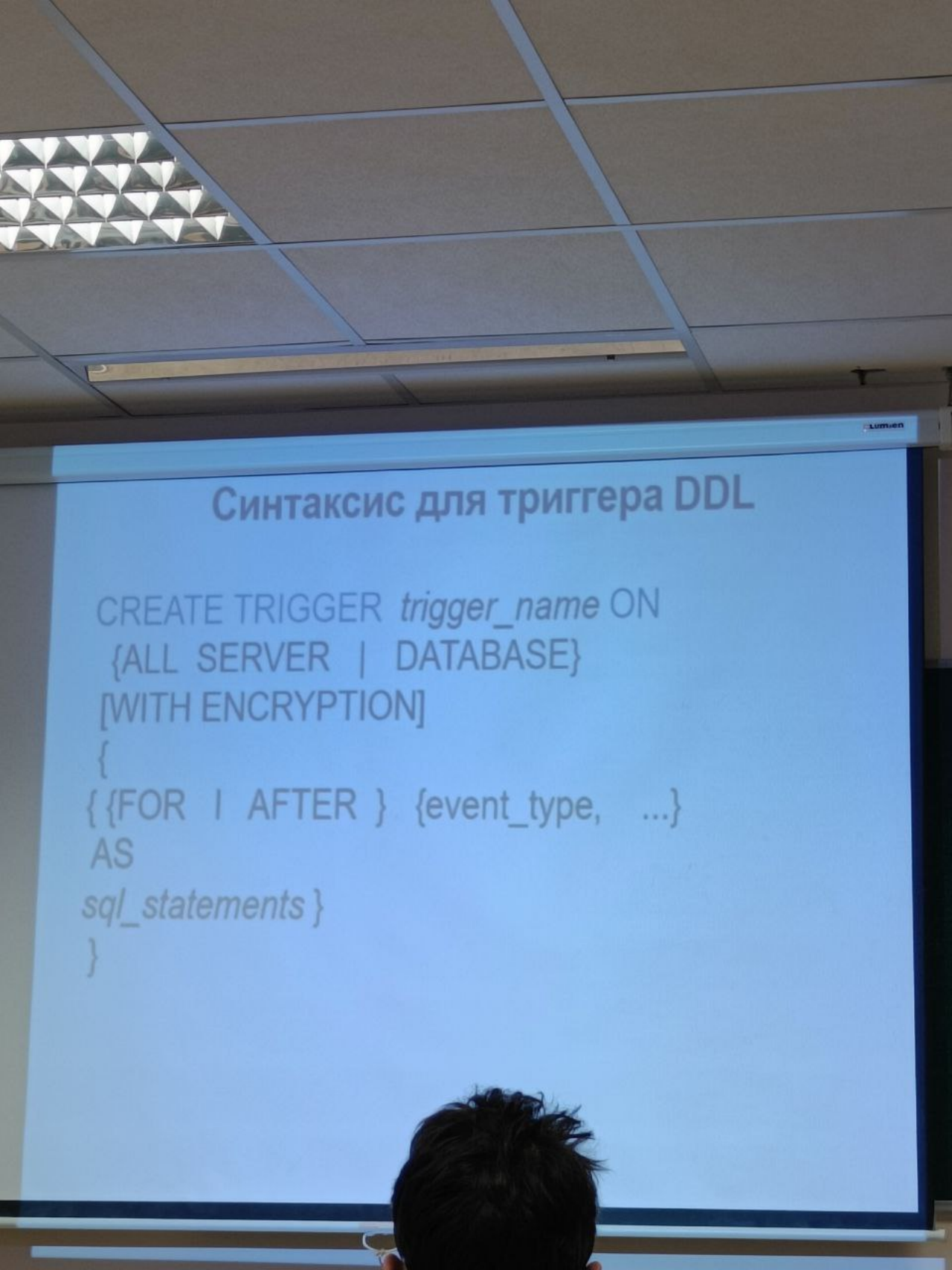
Если происходит обновление строки данных, то перед изменением запись строки помещается в таблицу **DELETED**, а после изменения копия строки данных помещается в таблицу **INSERTED**.

Эти таблицы хранятся во временной базе данных **tempdb**.


Последовательность работы триггеров
insert into…
начинается транзакция
instead of?
yes
заполняются таблицы inserted и deleted
instead of trigger выполняется if exists… select… insert… … …
no
check constraints
yes
триггер внес изменения в исходную таблицу
log statement
no
заполняются таблицы inserted и deleted
вызываются for/after триггеры
commit tran
All done!


Синтаксис для триггера DDL
CREATE TRIGGER trigger_name ON
{ALL SERVER | DATABASE}
[WITH ENCRYPTION]
{
{{FOR | AFTER } {event_type, ...}
AS
sql_statements }
}

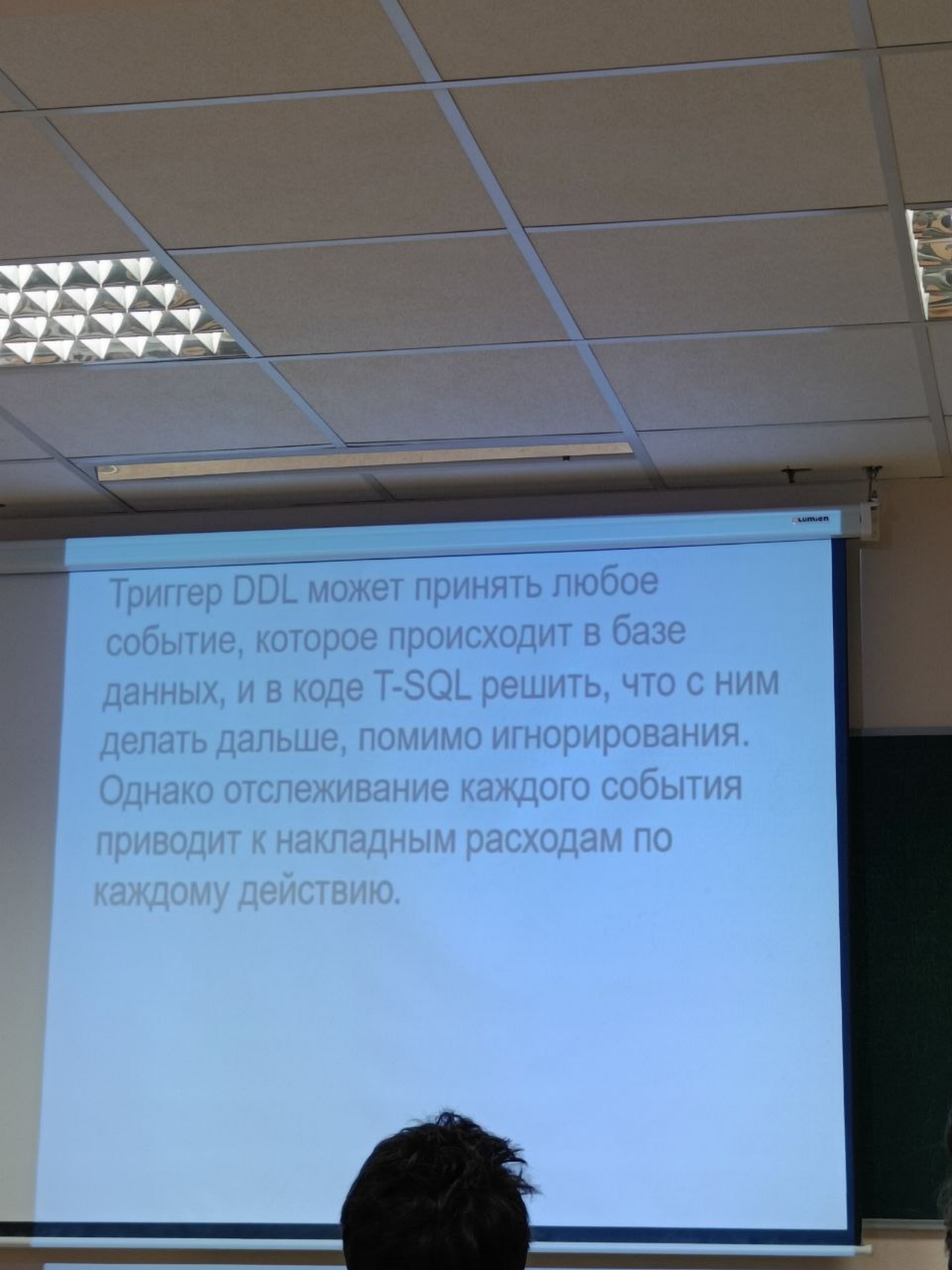

65.DROP_SYNONYM
66.DROP_TABLE
67.DROP_TRIGGER
68.DROP_TYPE
69.DROP_USER
70.DROP_VIEW
71.DROP_XML_SCHEMA
72.GRANT_DATABASE
73.REVOKE_DATABASE
74.UPDATE_STATISTICS

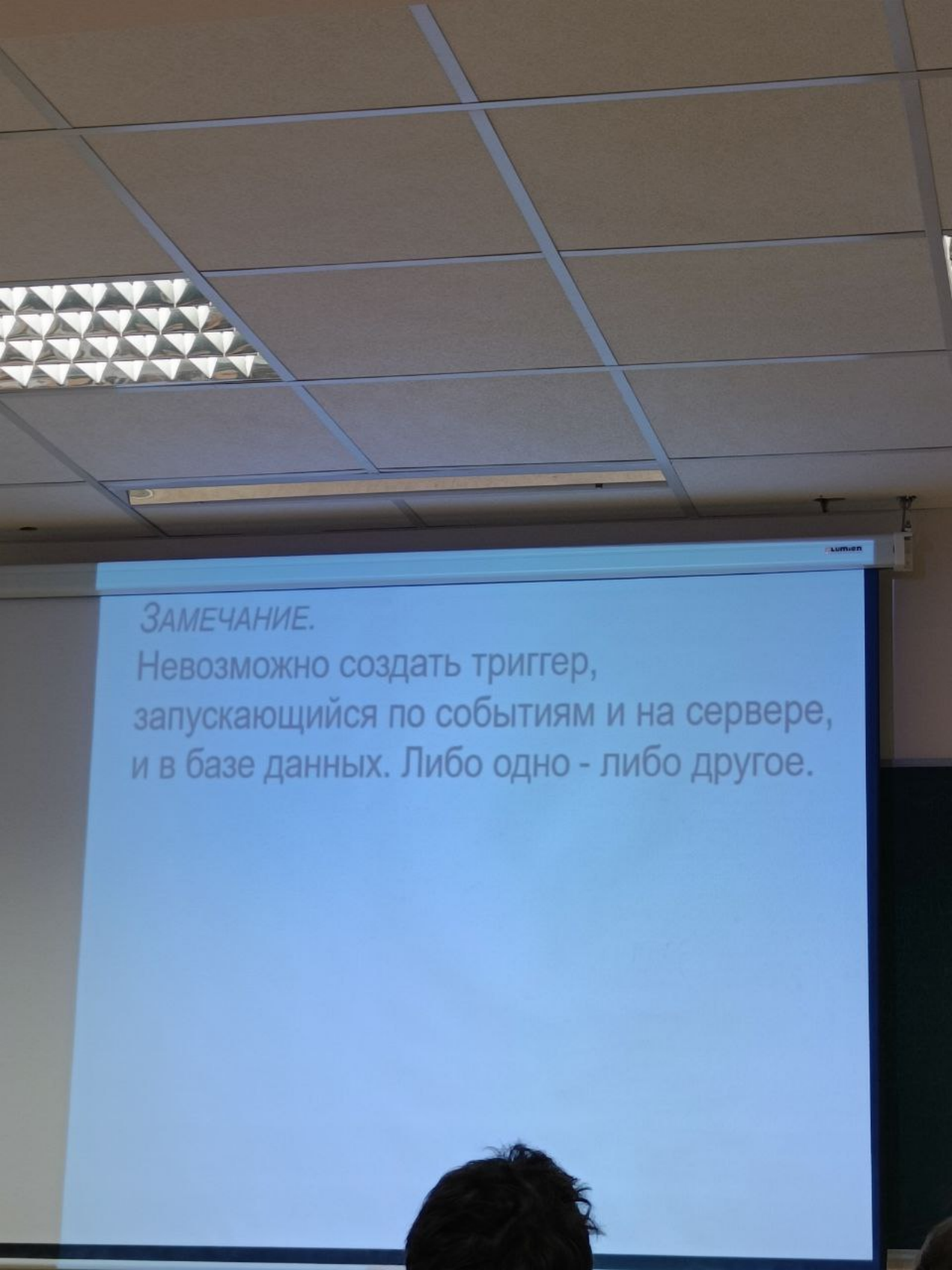

Синтаксис для триггера DDL
CREATE TRIGGER trigger_name ON
{ALL SERVER | DATABASE}
[WITH ENCRYPTION]
{
{ {FOR | AFTER } {event_type, ...}
AS
sql_statements }
}

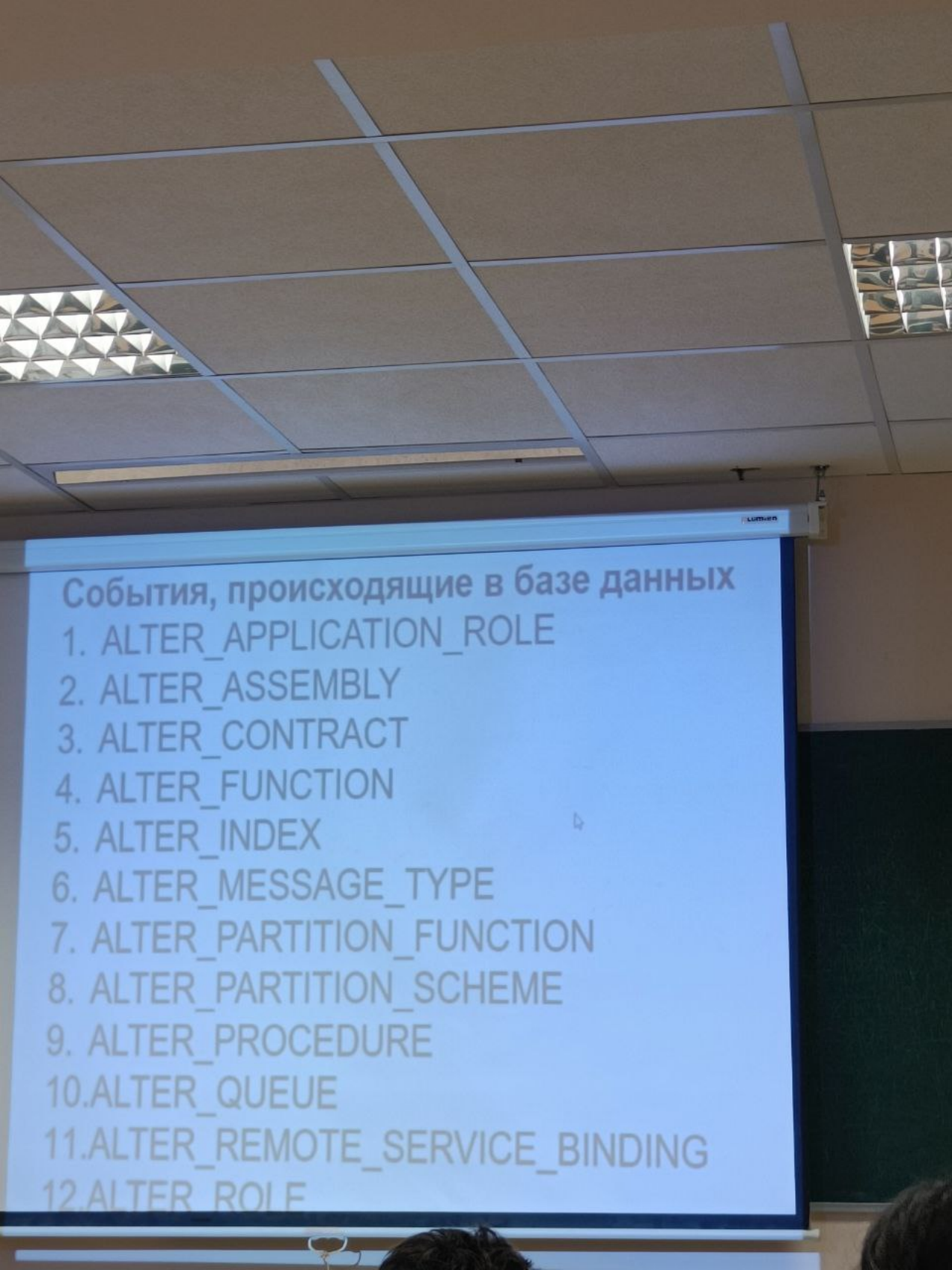
Основные параметры

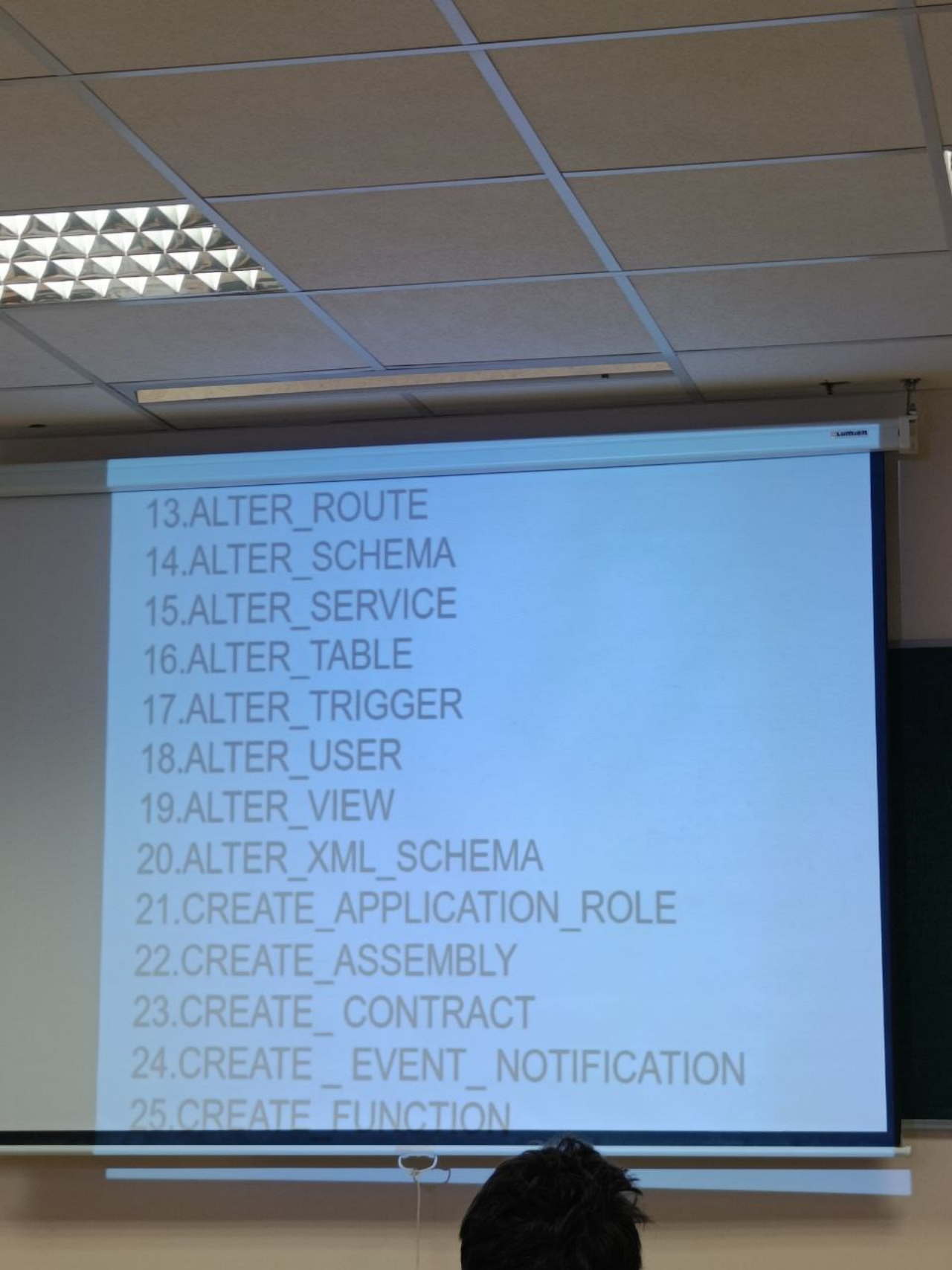
ALL SERVER | DATABASE — триггер запускается либо для сервера, либо для базы данных, с которой имелось соединение при создании триггера;

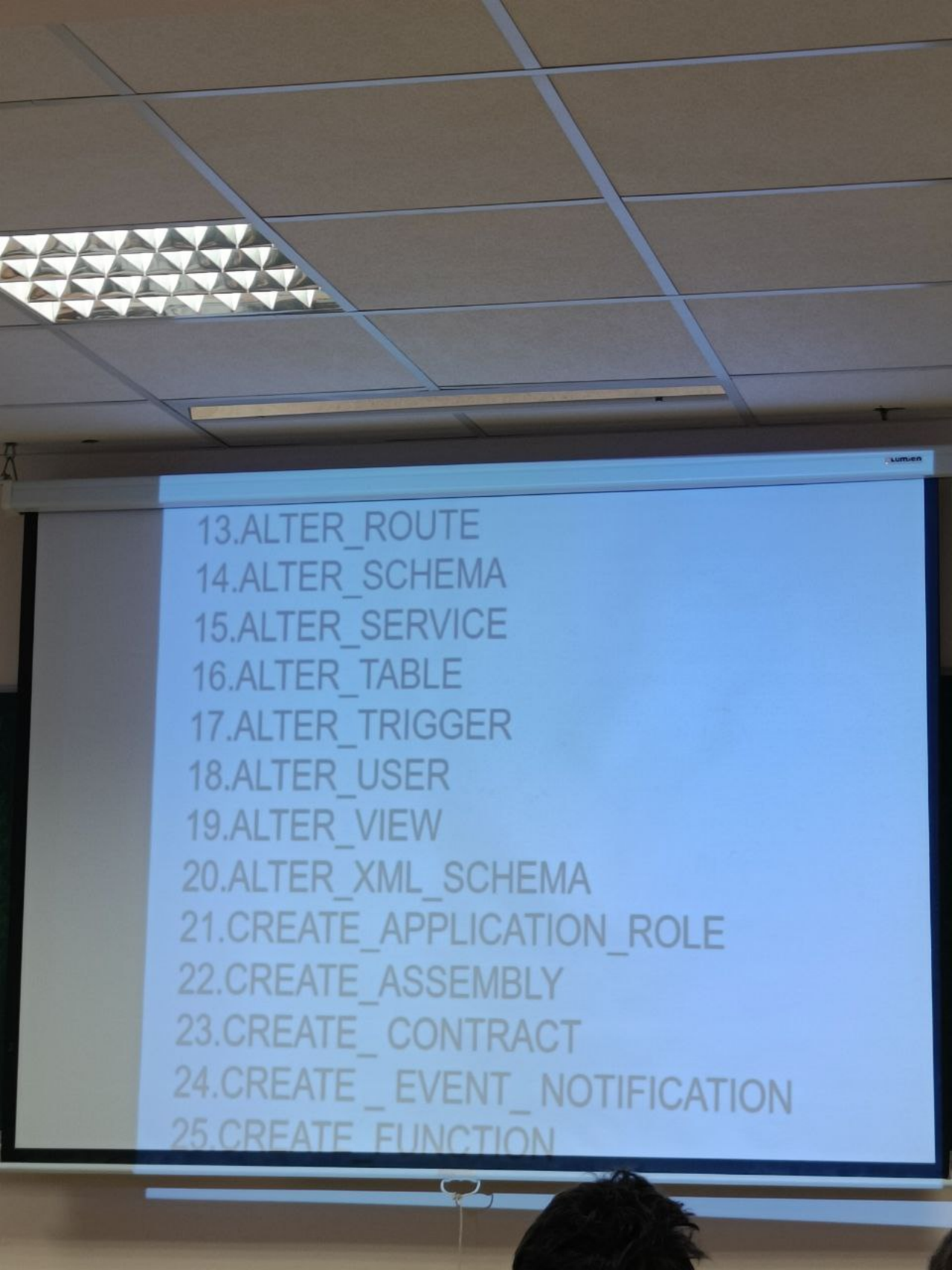
event_type - в этом списке через запятую перечислены действия, которые могут быть перехвачены, из списка действий DDL или базы данных или сервера.

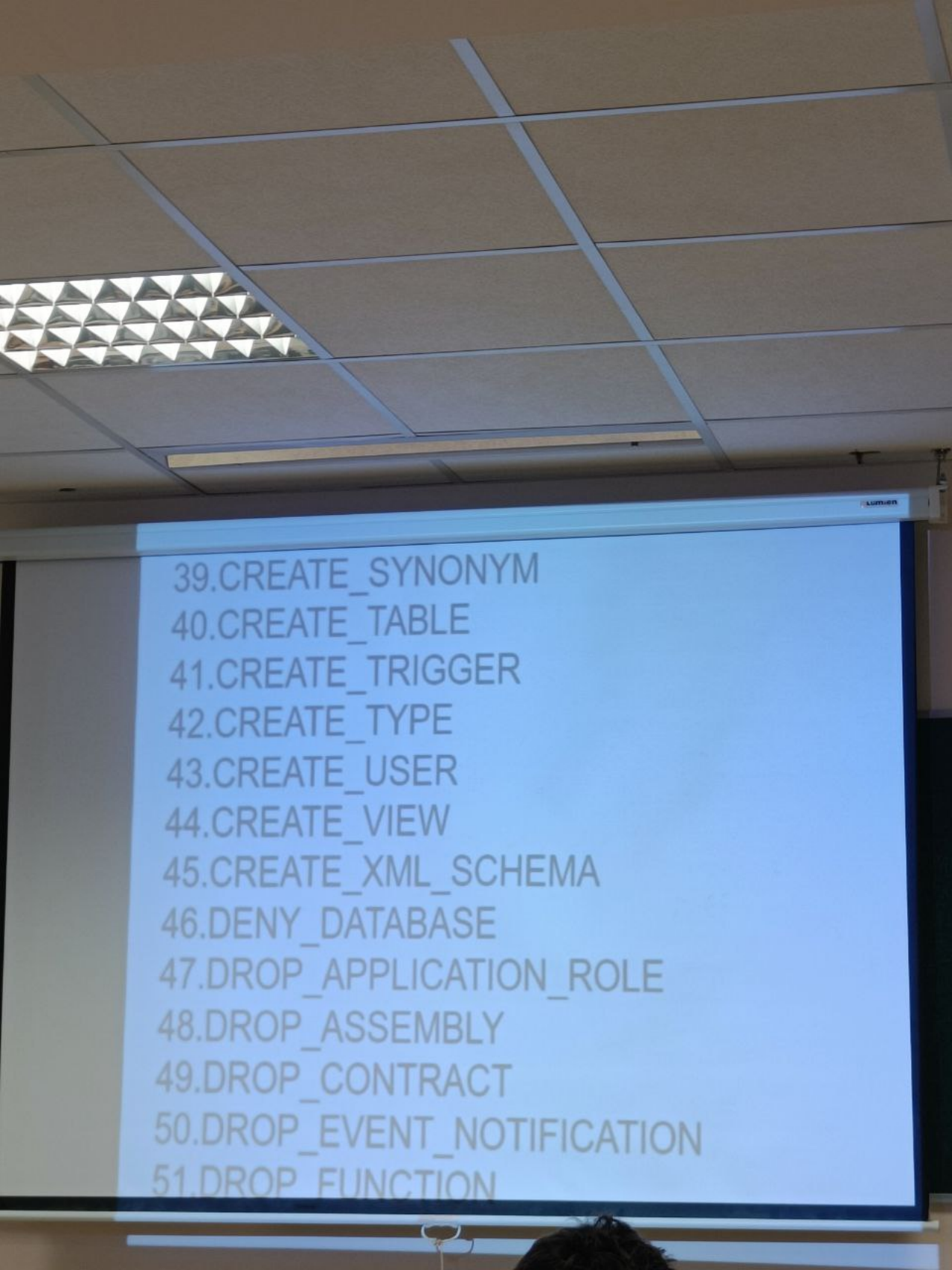

Триггер DDL может принять любое событие, которое происходит в базе данных, и в коде T-SQL решить, что с ним делать дальше, помимо игнорирования. Однако отслеживание каждого события приводит к накладным расходам по каждому действию.

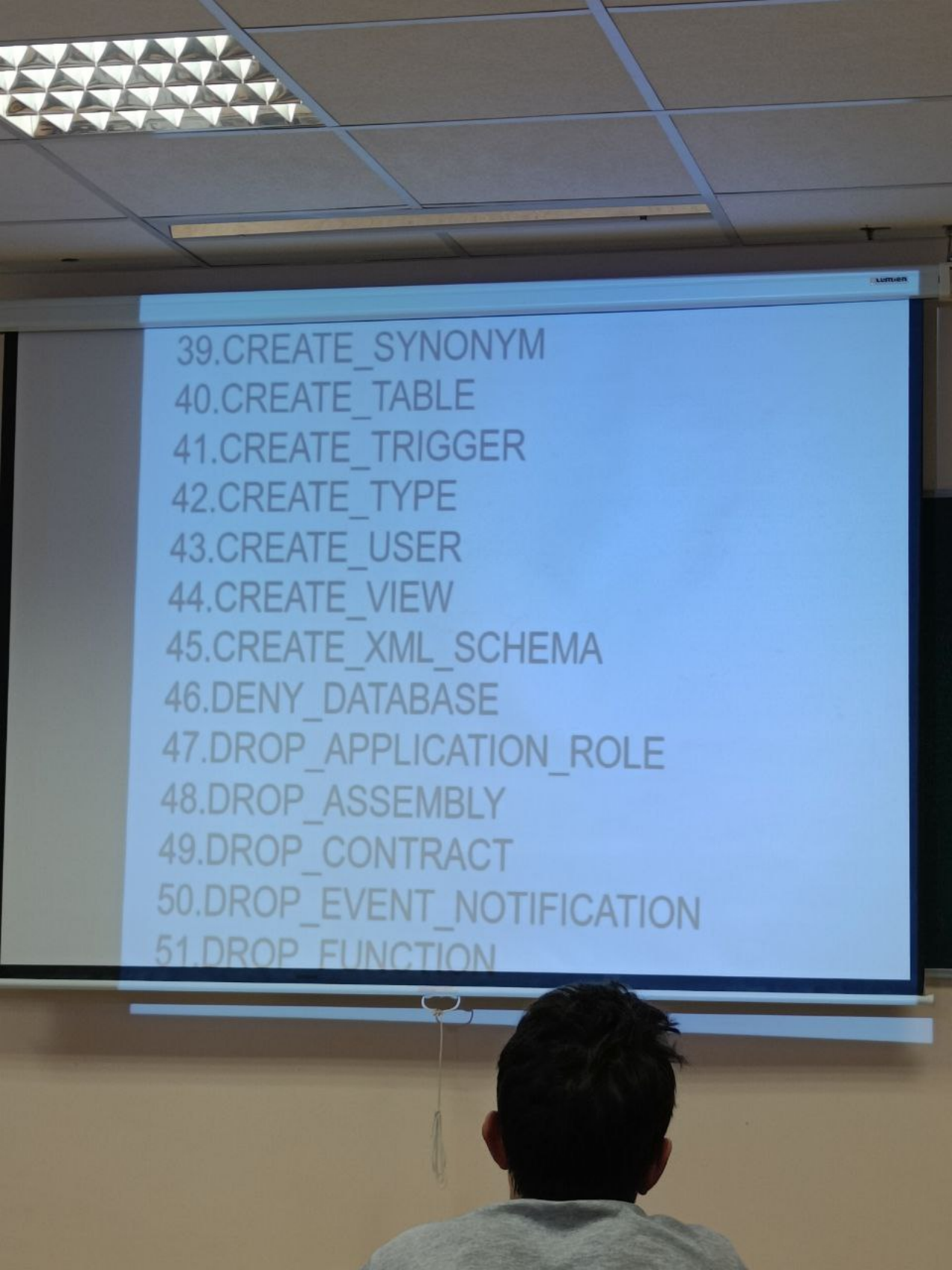

ЗАМЕЧАНИЕ.
Невозможно создать триггер,
запускающийся по событиям и на сервере,
и в базе данных. Либо одно - либо другое.

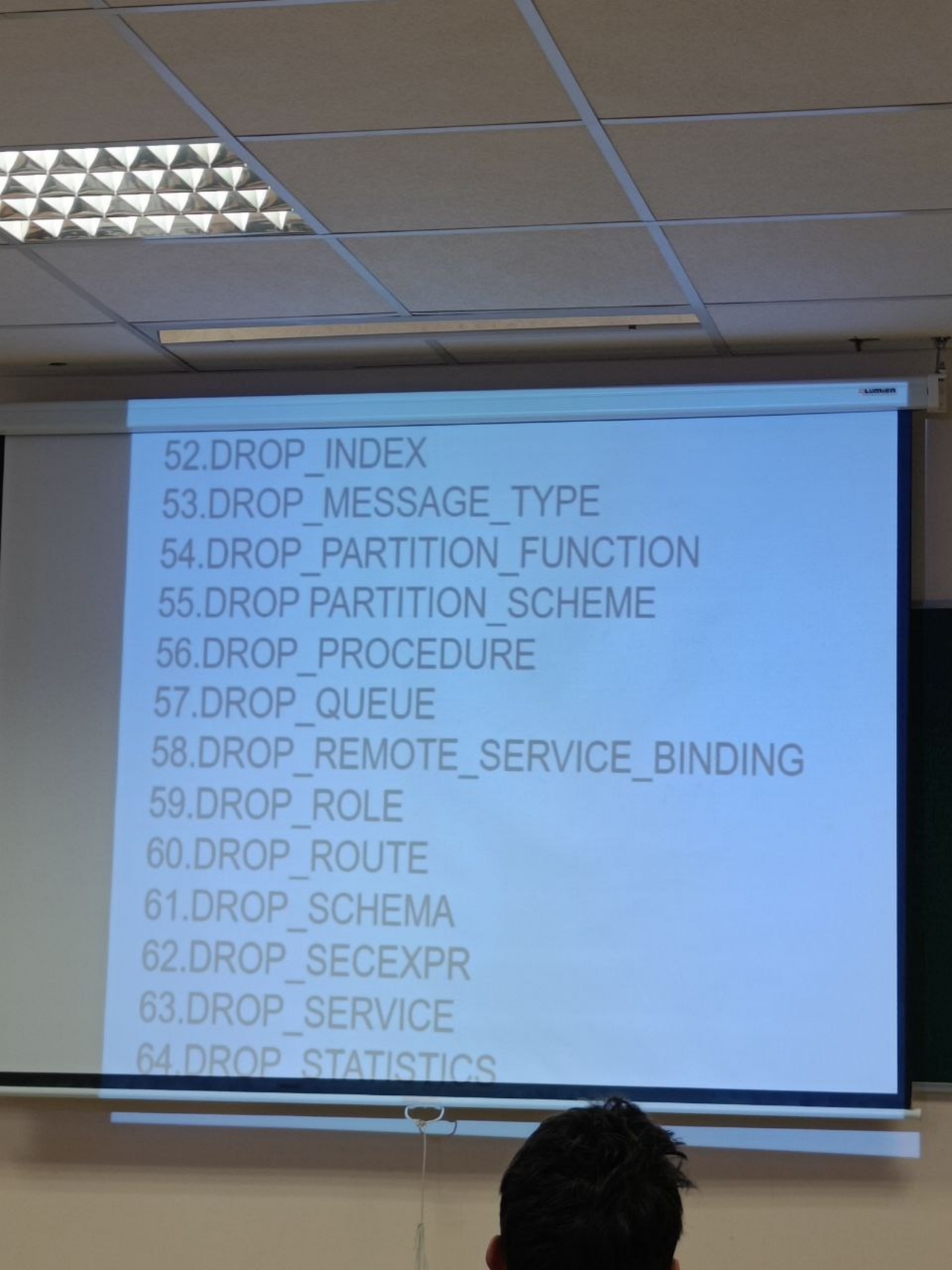

События, происходящие в базе данных
1. ALTER_APPLICATION_ROLE
2. ALTER_ASSEMBLY
3. ALTER_CONTRACT
4. ALTER_FUNCTION
5. ALTER_INDEX
6. ALTER_MESSAGE_TYPE
7. ALTER_PARTITION_FUNCTION
8. ALTER_PARTITION_SCHEME
9. ALTER_PROCEDURE
10.ALTER_QUEUE
11.ALTER_REMOTE_SERVICE_BINDING
12.ALTER_ROLE


13.ALTER_ROUTE
14.ALTER_SCHEMA
15.ALTER_SERVICE
16.ALTER_TABLE
17.ALTER_TRIGGER
18.ALTER_USER
19.ALTER_VIEW
20.ALTER_XML_SCHEMA
21.CREATE_APPLICATION_ROLE
22.CREATE_ASSEMBLY
23.CREATE_ CONTRACT
24.CREATE _ EVENT_ NOTIFICATION
25.CREATE_FUNCTION


13.ALTER_ROUTE
14.ALTER_SCHEMA
15.ALTER_SERVICE
16.ALTER_TABLE
17.ALTER_TRIGGER
18.ALTER_USER
19.ALTER_VIEW
20.ALTER_XML_SCHEMA
21.CREATE_APPLICATION_ROLE
22.CREATE_ASSEMBLY
23.CREATE_ CONTRACT
24.CREATE _ EVENT_ NOTIFICATION
25.CREATE_FUNCTION


39.CREATE_SYNONYM
40.CREATE_TABLE
41.CREATE_TRIGGER
42.CREATE_TYPE
43.CREATE_USER
44.CREATE_VIEW
45.CREATE_XML_SCHEMA
46.DENY_DATABASE
47.DROP_APPLICATION_ROLE
48.DROP_ASSEMBLY
49.DROP_CONTRACT
50.DROP_EVENT_NOTIFICATION
51.DROP_FUNCTION


39.CREATE_SYNONYM
40.CREATE_TABLE
41.CREATE_TRIGGER
42.CREATE_TYPE
43.CREATE_USER
44.CREATE_VIEW
45.CREATE_XML_SCHEMA
46.DENY_DATABASE
47.DROP_APPLICATION_ROLE
48.DROP_ASSEMBLY
49.DROP_CONTRACT
50.DROP_EVENT_NOTIFICATION
51.DROP_FUNCTION


52.DROP_INDEX
53.DROP_MESSAGE_TYPE
54.DROP_PARTITION_FUNCTION
55.DROP PARTITION_SCHEME
56.DROP_PROCEDURE
57.DROP_QUEUE
58.DROP_REMOTE_SERVICE_BINDING
59.DROP_ROLE
60.DROP_ROUTE
61.DROP_SCHEMA
62.DROP_SECEXPR
63.DROP_SERVICE
64.DROP_STATISTICS