Normalization and DB Creation

Normalization

How you organize your data has a profound effect on how efficiently the data is stored, updated and selected.

Normalization is the process of:

- eliminating data redundancy
- enhancing data integrity
- eliminating unstructured data

Traditionally, there are 5 forms of normalization. Neatly named 1st, 2nd, 3rd, 4th, 5th and even 6th normal form.

In practice, only 1st through 3rd are typically used.

4th, 5th and 6th normal form are too restrictive for real use.

Even 3rd normal form is too restrictive, and therefore we will learn *Jeff Normal Form*, which is a less restrictive form of 3rd normal form.

In all normal forms each row of a table is uniquely identified by a primary key.

A primary key may be a single field or a combination of keys.

Normalization – 1st Normal Form

1st Normal Form:

- breaks data into the smallest units possible
- data should be atomic
- each record must be unique

Observe the following table

People		
Name	**Address**	**Phone**
Jeff Salvage	1313 Mockingbird Lane, Mockingbird Heights, CA, 33004	(555) 555-1212
Ahsoka Tano	500 Republica Street, Coruscant City, Coruscant, 11111	(555) 101-1977
Thurman Munson	1977 Championship Lane, Bronx, NY 11002	(888) YAN-KEES

Therefore, any field that is non-atomic should be broken into separate fields.

People						
FirstName	**LastName**	**Address**	**City**	**State**	**Zip**	**Phone**
Jeff	Salvage	1313 Mockingbird Lane	Mockingbird Heights	CA	33004	(555) 555-1212
Ahsoka	Tano	500 Republica Street	Coruscant City	Coruscant	11111	(555) 101-1977
Thurman	Munson	1977 Championship Lane	Bronx	NY	11002	(888) YAN-KEES

Technically we should break up address, but we will leave that as is for now.

Normalization – 1st Normal Form

Additionally, data should not be stored with repetitious groups of fields:

Observe the following table

ForceUsers		
Allegiance	**ForceUser1**	**ForceUser2**
Jedi	Qui Gon Jinn	Ahsoka Tano
Sith	Darth Maul	Count Dooko

When data is set up in this manner unused fields are wasted. In addition, a set number of values for the repeated field is hard coded into the table instead of allowing an arbitrary number of values.

Instead, you should create a record for each value as shown in the following table:

ForceAllegiance		
Allegiance	**FirstName**	**LastName**
Jedi	Qui Gon	Jinn
Jedi	Ahsoka	Tano
Sith	Darth	Maul
Sith	Count	Dooko

Normalization – 1st Normal Form

However, now the table does not have a simple primary key.

Either we can add a field to the table to identify the force user or we can make the force user's first and last name part of the primary key.

ForceUser			
IDForceUser	**Allegiance**	**FirstName**	**LastName**
1	Jedi	Qui Gon	Jinn
2	Jedi	Ahsoka	Tano
3	Sith	Darth	Maul
4	Sith	Count	Dooko

Normalization – 2nd Normal Form

Each progressive normal form builds on the previous normal form.

2nd Normal Form includes all of the rules that apply to 1st Normal Form.

2nd normal form states that in tables with compound primary keys, each non-key field should relate to a fact about all the keys (not a single part of the key) in the compound primary key.

Otherwise, the data should be reorganized into another table.

The following table is in 1st normal form and contains a compound primary key of *IDForceUser* and *IDAllegiance*.

ForceUser					
IDForceUser	**IDAllegiance**	**Allegiance**	**FirstName**	**LastName**	**MidichlorianCount**
1	1	Jedi	Qui Gon	Jinn	10,000
2	1	Jedi	Ahsoka	Tano	14,000
3	2	Sith	Darth	Maul	14,000
4	2	Sith	Count	Dooko	16,500

Normalization – 2nd Normal Form

Observe the table broken into three tables that are in 2nd normal form.

ForceAttributes		
IDForceUser	**IDAllegiance**	**MidichlorianCount**
1	1	10,000
2	1	14,000
3	2	14,000
4	2	16,500

ForceUser		
IDForceUser	**FirstName**	**LastName**
1	Qui Gon	Jinn
2	Ahsoka	Tano
3	Darth	Maul
4	Count	Dooko

Allegiance	
IDAllegiance	**Allegiance**
1	Jedi
2	Sith

Normalization – 3rd Normal Form

3rd normal form is similar to 2nd normal form.
The only difference is it applies to non-compound primary keys.
Each non key field should be a fact about the primary key.
Otherwise, it should be placed in a separate table.

Since the combination of *City*, *State* and *Zip* are repetitious, they do not belong in the table.

People							
idPerson	**FirstName**	**LastName**	**Address**	**City**	**State**	**Zip**	**Phone**
1	Jeff	Salvage	1313 Mockingbird Lane	Mockingbird Heights	CA	33004	(555) 555-1212
2	Ahsoka	Tano	500 Republica Street	Coruscant City	Coruscant	11111	(555) 101-1977
3	Thurman	Munson	1977 Championship Lane	Bronx	NY	11002	(888) YAN-KEES
4	Mon	Mothma	520 Republica Street	Coruscant City	Coruscant	11111	(555) 421-0608

City, *State*, and *Zip* should be in their own table.

Normalization – 3rd Normal Form

The correct 3rd normal form is:

People

idPerson	FirstName	LastName	Address	Zip	Phone
1	Jeff	Salvage	1313 Mockingbird Lane	33004	(555) 555-1212
2	Ahsoka	Tano	500 Republica Street	11111	(555) 101-1977
3	Thurman	Munson	1977 Championship Lane	11002	(888) YAN-KEES
4	Mon	Mothma	520 Republica Street	11111	(555) 421-0608

Address

City	State	Zip
Mockingbird Heights	CA	33004
Coruscant City	Coruscant	11111
Bronx	NY	11002
Coruscant City	Coruscant	11111

Normalization – Jeff Normal Form

Common sense.

Examples like the zip code take normalization too far.

Usually leave it as it was in 2nd normal form.

There are times it's ok not to normalize.

When the data is coming in as a unit as city, state, and zip are, that it's ok to leave it as is.

Also, zips do some strange things.

Not really a 1 to 1 relationship there.

Normalization – Creating a Data Model

Let's create a data model for tracking international competitions in race walking.
Specifically, we will focus on the judging of the event.

- Race walking is a progression of steps so taken that the walker makes contact with the ground so that no visible (to the human eye) loss of contact occurs.

- The advancing leg must be straightened (i.e., not bent at the knee) from the moment of first contact with the ground until in the vertical upright position.

Normalization – Creating a Data Model

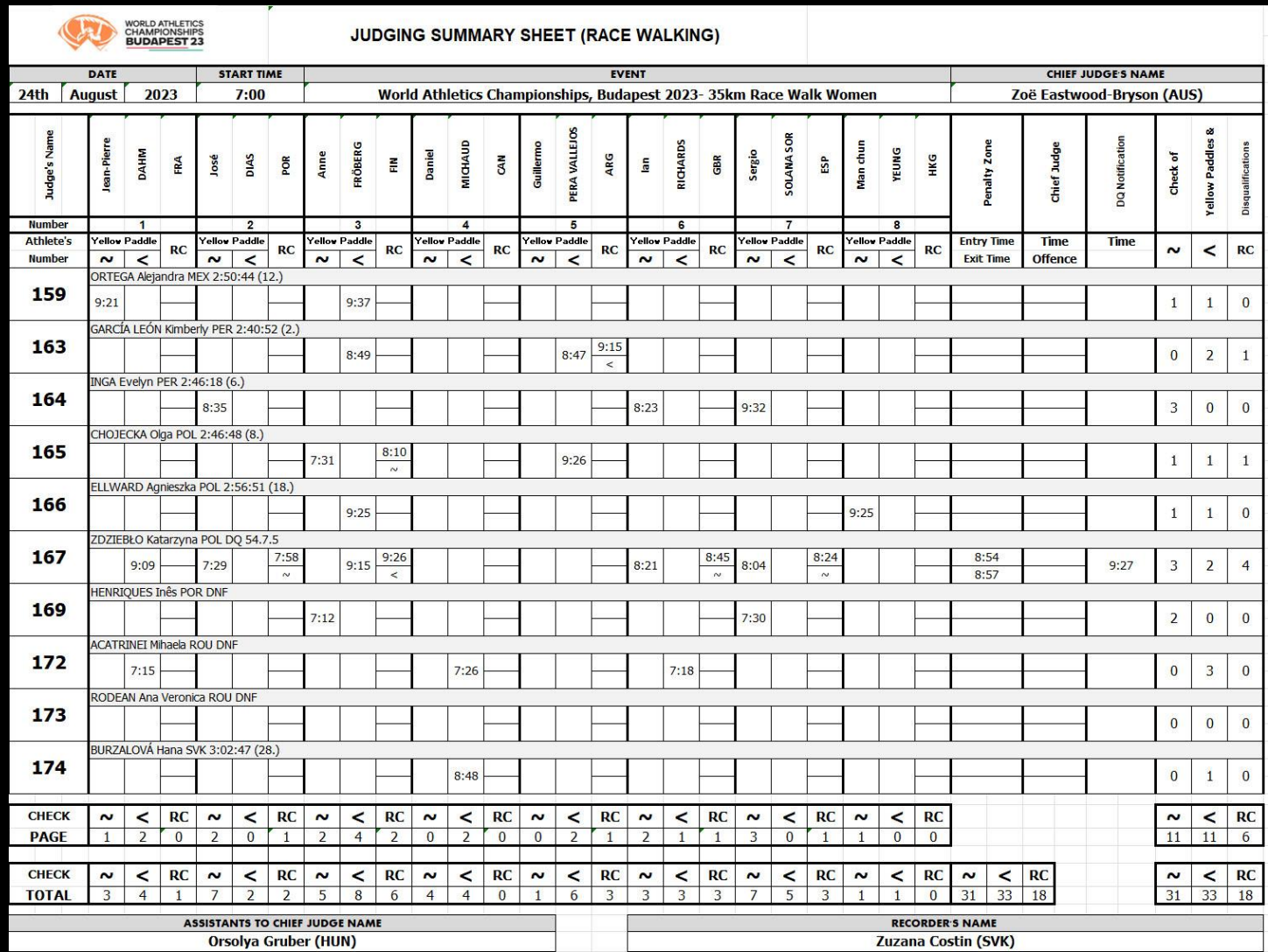
What is a "Bent Knee" in race walking:

Normalization – Creating a Data Model

What is a "Loss of Contact" in race walking:

Normalization – Creating a Data Model

All the events that occur in the race are tracking with a Summary Sheet
Our job is to develop a data model for the race



WORLD ATHLETICS CHAMPIONSHIPS BUDAPEST 23

JUDGING SUMMARY SHEET (RACE WALKING)

DATE	START TIME	EVENT	CHIEF JUDGE'S NAME
24th August 2023	7:00	World Athletics Championships, Budapest 2023- 35km Race Walk Women	Zoë Eastwood-Bryson (AUS)

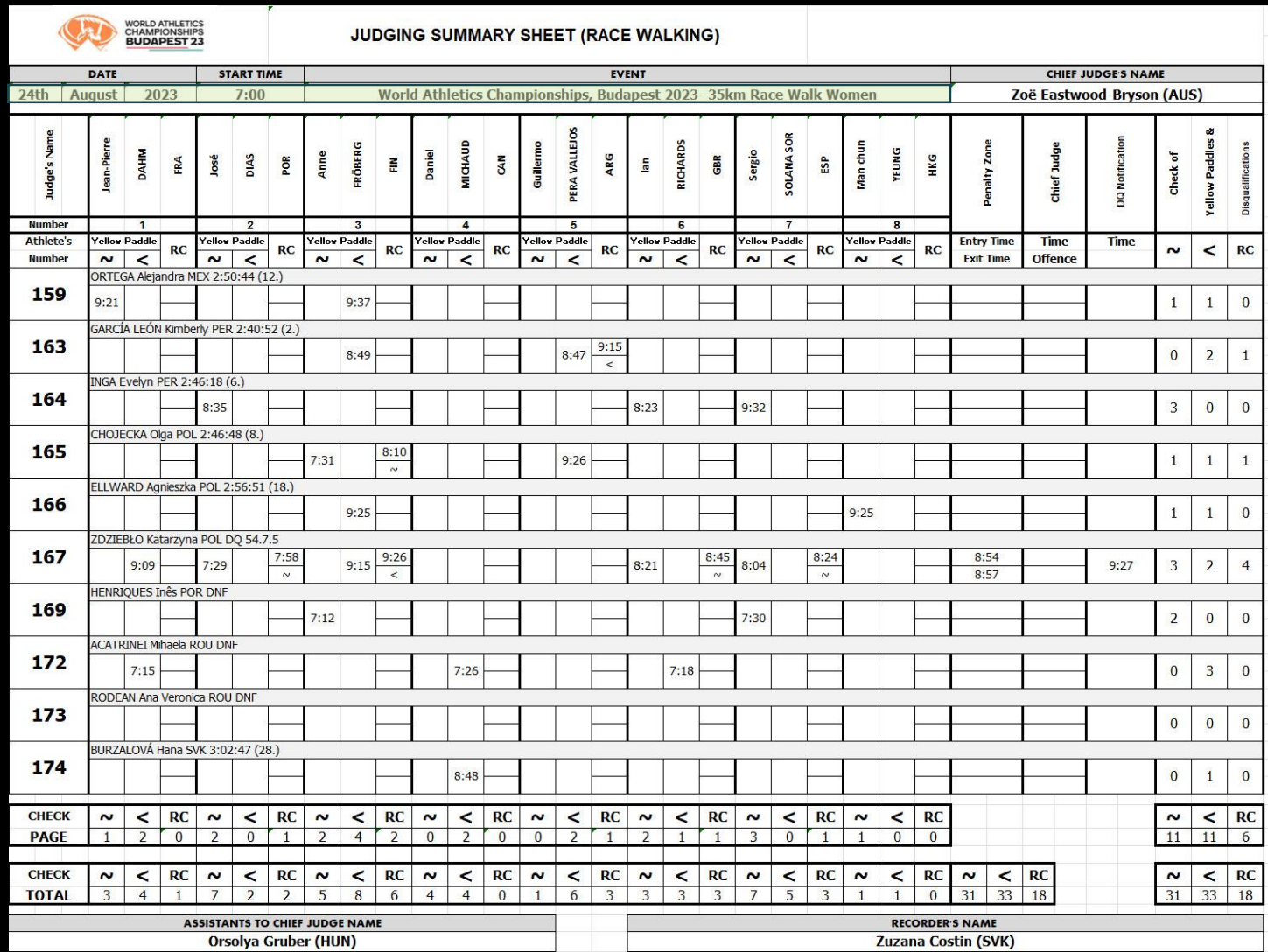
Judge's Name	Jean-Pierre DAHM FRA			José DIAS POR			Anne FRÖBERG FIN			Daniel MICHAUD CAN			Guillermo PERA VALLEJOS ARG			Ian RICHARDS GBR			Sergio SOLANA SOR ESP			Man chun YEUNG HKG			Penalty Zone	Chief Judge	DQ Notification	Check of	Yellow Paddles &	Disqualifications
Number	1			2			3			4			5			6			7			8									
Athlete's Number	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Entry Time / Exit Time	Time / Offence	Time	~	<	RC
159	ORTEGA Alejandra MEX 2:50:44 (12.)																													
	9:21							9:37																				1	1	0
163	GARCÍA LEÓN Kimberly PER 2:40:52 (2.)																													
								8:49						8:47	9:15 <													0	2	1
164	INGA Evelyn PER 2:46:18 (6.)																													
				8:35												8:23			9:32									3	0	0
165	CHOJECKA Olga POL 2:46:48 (8.)																													
							7:31		8:10 ~					9:26														1	1	1
166	ELLWARD Agnieszka POL 2:56:51 (18.)																													
								9:25														9:25						1	1	0
167	ZDZIEBŁO Katarzyna POL DQ 54.7.5																													
		9:09		7:29		7:58 ~		9:15	9:26 <							8:21		8:45 ~	8:04		8:24 ~				8:54 / 8:57		9:27	3	2	4
169	HENRIQUES Inês POR DNF																													
							7:12												7:30									2	0	0
172	ACATRINEI Mihaela ROU DNF																													
		7:15									7:26						7:18											0	3	0
173	RODEAN Ana Veronica ROU DNF																													
																												0	0	0
174	BURZALOVÁ Hana SVK 3:02:47 (28.)																													
											8:48																	0	1	0

	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC				~	<	RC
CHECK PAGE	1	2	0	2	0	1	2	4	2	0	2	0	0	2	1	2	1	1	3	0	1	1	0	0				11	11	6
CHECK TOTAL	3	4	1	7	2	2	5	8	6	4	4	0	1	6	3	3	3	3	7	5	3	1	1	0	31	33	18	31	33	18

ASSISTANTS TO CHIEF JUDGE NAME	RECORDER'S NAME
Orsolya Gruber (HUN)	Zuzana Costin (SVK)

Normalization – Creating a Data Model

What attributes define a race?



WORLD ATHLETICS CHAMPIONSHIPS BUDAPEST 23

JUDGING SUMMARY SHEET (RACE WALKING)

DATE	START TIME	EVENT	CHIEF JUDGE'S NAME
24th August 2023	7:00	World Athletics Championships, Budapest 2023- 35km Race Walk Women	Zoë Eastwood-Bryson (AUS)

Judge's Name	Jean-Pierre DAHM FRA			José DIAS POR			Anne FRÖBERG FIN			Daniel MICHAUD CAN			Guillermo PERA VALLEJOS ARG			Ian RICHARDS GBR			Sergio SOLANA SOR ESP			Man chun YEUNG HKG			Penalty Zone	Chief Judge	DQ Notification	Check of	Yellow Paddles &	Disqualifications
Number	1			2			3			4			5			6			7			8									
Athlete's Number	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Entry Time / Exit Time	Time / Offence	Time	~	<	RC
159	ORTEGA Alejandra MEX 2:50:44 (12.)																													
	9:21							9:37																				1	1	0
163	GARCÍA LEÓN Kimberly PER 2:40:52 (2.)																													
								8:49						8:47	9:15 <													0	2	1
164	INGA Evelyn PER 2:46:18 (6.)																													
				8:35												8:23			9:32									3	0	0
165	CHOJECKA Olga POL 2:46:48 (8.)																													
							7:31		8:10 ~					9:26														1	1	1
166	ELLWARD Agnieszka POL 2:56:51 (18.)																													
								9:25														9:25						1	1	0
167	ZDZIEBŁO Katarzyna POL DQ 54.7.5																													
		9:09		7:29		7:58 ~		9:15	9:26 <							8:21		8:45 ~	8:04		8:24 ~				8:54 / 8:57		9:27	3	2	4
169	HENRIQUES Inês POR DNF																													
							7:12												7:30									2	0	0
172	ACATRINEI Mihaela ROU DNF																													
		7:15									7:26						7:18											0	3	0
173	RODEAN Ana Veronica ROU DNF																													
																												0	0	0
174	BURZALOVÁ Hana SVK 3:02:47 (28.)																													
											8:48																	0	1	0

	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC				~	<	RC
CHECK PAGE	1	2	0	2	0	1	2	4	2	0	2	0	0	2	1	2	1	1	3	0	1	1	0	0				11	11	6
CHECK TOTAL	3	4	1	7	2	2	5	8	6	4	4	0	1	6	3	3	3	3	7	5	3	1	1	0	31	33	18	31	33	18

ASSISTANTS TO CHIEF JUDGE NAME	RECORDER'S NAME
Orsolya Gruber (HUN)	Zuzana Costin (SVK)

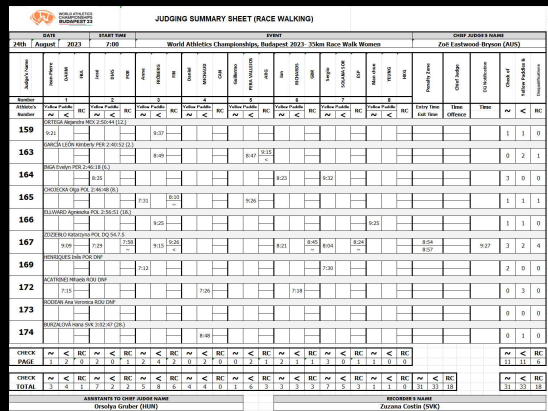
Normalization – Creating a Data Model

What attributes define a race?

The event field on the spreadsheet is a compound field.
It contains:
The event name: World Athletics Championships
City: Budapest
Year: 2023, repeated elsewhere
Distance: 35km
Gender: Women
RaceDate: 8/24/2023
Start Time: 7:00
Country: Not listed, but should be

This informs the creation of:

Race								
IDRace	**Event**	**City**	**Country**	**RaceDate**	**StartTime**	**Distance**	**DistanceUnits**	**Gender**
1	World Athletics Championships	Budapest	Hungary	2023-08-24	7:00 AM	35	km	F
2	World Athletics Championships	Budapest	Hungary	2023-08-24	7:00 AM	35	km	M
3	World Athletics Championships	Budapest	Hungary	2023-08-20	7:15 AM	20	km	F
4	World Athletics Championships	Budapest	Hungary	2023-08-19	10:50 AM	20	km	M


JUDGING SUMMARY SHEET (RACE WALKING)

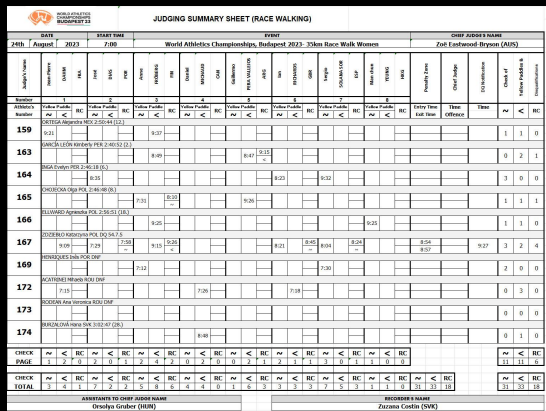
Normalization – Creating a Data Model

We already have issues of repetitious data related to event, city and country.

This informs the creation of:

Event			
IDEvent	**Event**	**City**	**Country**
1	World Athletics Championships	Budapest	Hungary
2	World Athletics Team Championships	Anatalya	Turkey
3	Olympics Games	Paris	France

Race						
IDRace	**IDEvent**	**RaceDate**	**StartTime**	**Distance**	**DistanceUnits**	**Gender**
1	1	2023-08-24	7:00 AM	35	km	F
2	1	2023-08-24	7:00 AM	35	km	M
3	1	2023-08-20	7:15 AM	20	km	F
4	1	2023-08-19	10:50 AM	20	km	M


JUDGING SUMMARY SHEET (RACE WALKING)

DATE: 24th August 2023 | START TIME: 7:00 | EVENT: World Athletics Championships, Budapest 2023 - 35km Race Walk Women | CHIEF JUDGE'S NAME: Zoë Eastwood Bryson (AUS)

Athlete's Number	159	163	164	165	166	167	169	172	173	174
CHECK PAGE	1	0	2	1	3	0	0	2	0	0
CHECK TOTAL	3	1	7	2	5	3	1	6	0	1

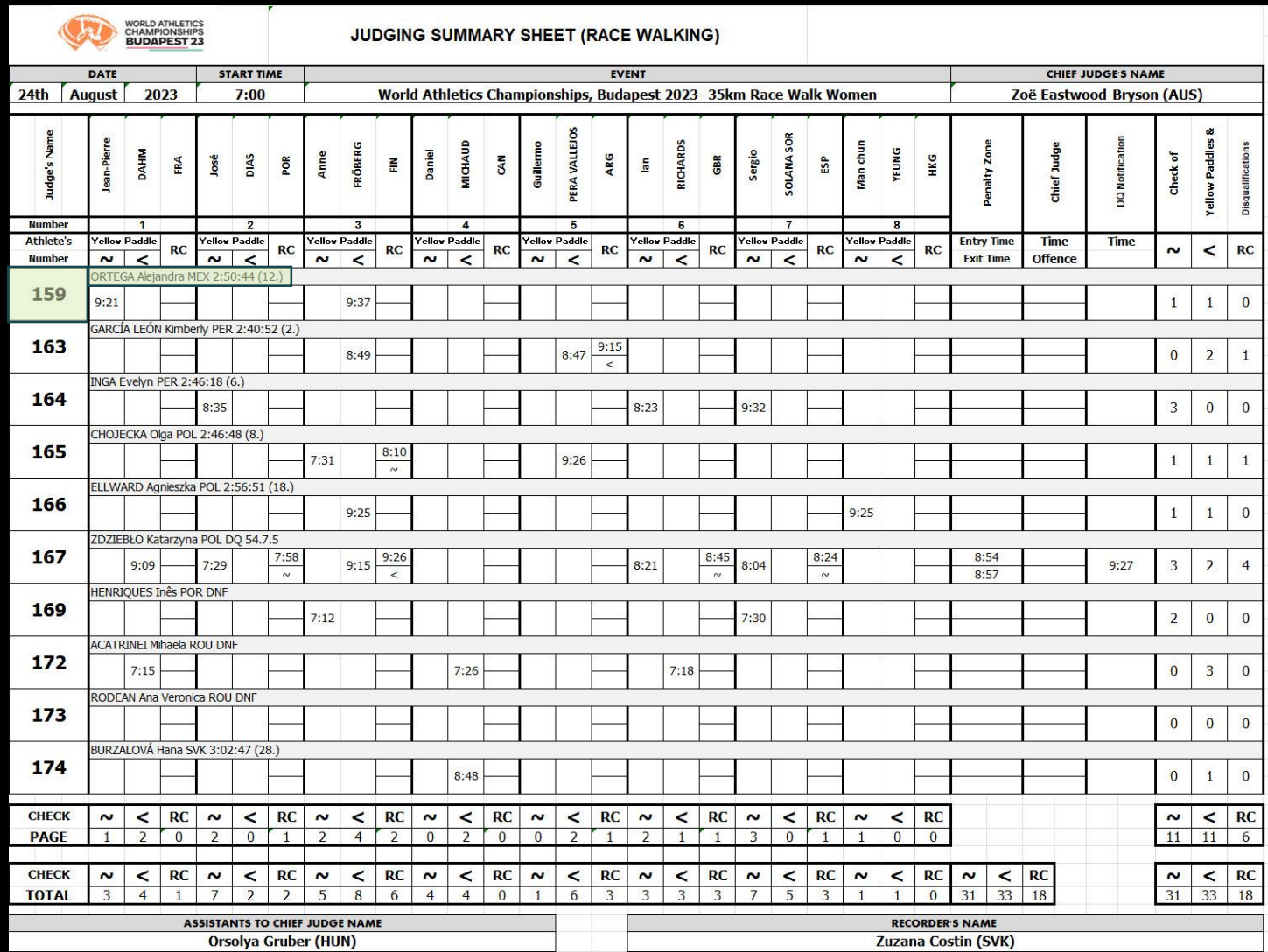
ASSISTANTS TO CHIEF JUDGE NAME: Orsolya Gruber (HUN)

RECORDER'S NAME: Zuzana Costin (SVK)

Normalization – Creating a Data Model

Let's create a relation for Athletes

We need to track:
FirstName: Alejandra
LastName: ORTEGA
Country: Mex
BibNumber: 159
FinishingTime: 2:50:44
FinishingPlace: 12



WORLD ATHLETICS CHAMPIONSHIPS BUDAPEST 23

JUDGING SUMMARY SHEET (RACE WALKING)

DATE	START TIME	EVENT	CHIEF JUDGE'S NAME
24th August 2023	7:00	World Athletics Championships, Budapest 2023- 35km Race Walk Women	Zoë Eastwood-Bryson (AUS)

Judge's Name	Jean-Pierre DAHM FRA			José DIAS POR			Anne FRÖBERG FIN			Daniel MICHAUD CAN			Guillermo PERA VALLEJOS ARG			Ian RICHARDS GBR			Sergio SOLANA SOR ESP			Man chun YEUNG HKG			Penalty Zone	Chief Judge	DQ Notification	Check of	Yellow Paddles &	Disqualifications
Number	1			2			3			4			5			6			7			8										
Athlete's Number	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Entry Time / Exit Time	Time Offence	Time	~	<	RC
159	ORTEGA Alejandra MEX 2:50:44 (12.)																														
	9:21							9:37																				1	1	0
163	GARCÍA LEÓN Kimberly PER 2:40:52 (2.)																														
								8:49						8:47	9:15 <													0	2	1
164	INGA Evelyn PER 2:46:18 (6.)																														
				8:35												8:23			9:32									3	0	0
165	CHOJECKA Olga POL 2:46:48 (8.)																														
							7:31		8:10 ~					9:26														1	1	1
166	ELLWARD Agnieszka POL 2:56:51 (18.)																						9:25					1	1	0
								9:25																							
167	ZDZIEBŁO Katarzyna POL DQ 54.7.5																														
		9:09		7:29		7:58 ~		9:15	9:26 <							8:21		8:45 ~	8:04		8:24 ~					8:54 / 8:57		9:27	3	2	4
169	HENRIQUES Inês POR DNF																														
							7:12												7:30									2	0	0
172	ACATRINEI Mihaela ROU DNF																														
		7:15									7:26						7:18											0	3	0
173	RODEAN Ana Veronica ROU DNF																														
																												0	0	0
174	BURZALOVÁ Hana SVK 3:02:47 (28.)																														
											8:48																	0	1	0

	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC				~	<	RC
CHECK PAGE	1	2	0	2	0	1	2	4	2	0	2	0	0	2	1	2	1	1	3	0	1	1	0	0				11	11	6
CHECK TOTAL	3	4	1	7	2	2	5	8	6	4	4	0	1	6	3	3	3	3	7	5	3	1	1	0	31	33	18	31	33	18

ASSISTANTS TO CHIEF JUDGE NAME	RECORDER'S NAME
Orsolya Gruber (HUN)	Zuzana Costin (SVK)

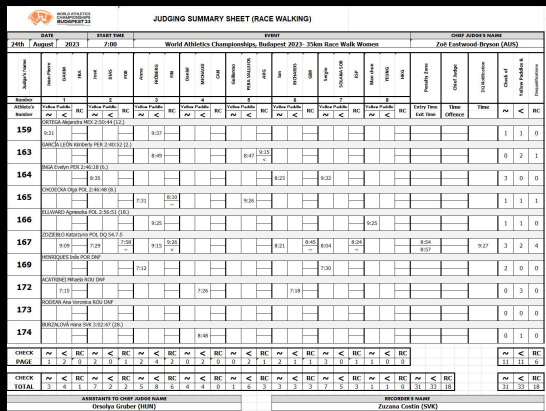
Normalization – Creating a Data Model

Let's create a relation for Athletes

We need to track:
FirstName: Alejandra
LastName: ORTEGA
Country: Mex
BibNumber: 159
FinishingTime: 2:50:44
FinishingPlace: 12
Finished: Yes / DNF / DQ

This informs the creation of:

Race							
idAthlete	**FirstName**	**LastName**	**CountryCode**	**BibNumber**	**FinishingTime**	**FinishingPlace**	**Finished**
1	Alejandra	ORTEGA	MEX	159	2:50:44	12	Yes
2	Kimberly	GARCÍA LEÓN	PER	163	2:40:52	2	Yes
3	Evelyn	INGA	PER	164	2:46:18	6	Yes
4	Olga	CHOKECKA	POL	165	2:46:48	20	Yes



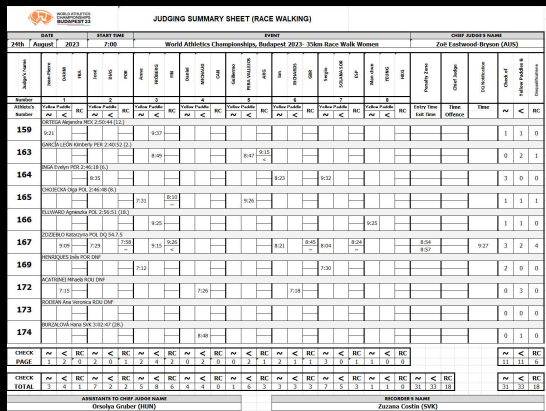
Normalization – Creating a Data Model

Again, we have issues related to *BibNumber*, *FinishingTime*, *FinishingPlace*
Athletes will compete in more than one race/event.
Race specific data must be separated from athlete specific data

Refactor as follows:

Athlete			
IDAthlete	**FirstName**	**LastName**	**CountryCode**
1	Alejandra	ORTEGA	MEX
2	Kimberly	GARCÍA LEÓN	PER
3	Evelyn	INGA	PER
4	Olga	CHOKECKA	POL

Bib					
IDRace	**IDAthlete**	**BibNumber**	**FinishingTime**	**FinishingPlace**	**Finished**
1	1	159	2:50:44	12	Yes
1	2	163	2:40:52	2	Yes
1	3	164	2:46:18	6	Yes
1	4	165	2:46:48	20	Yes


JUDGING SUMMARY SHEET (RACE WALKING)

World Athletics Championships, Budapest 2023 - 35km Race Walk Women

Zoë Eastwood Bryson (AUS)

Normalization – Creating a Data Model

Judges need to be tracked:

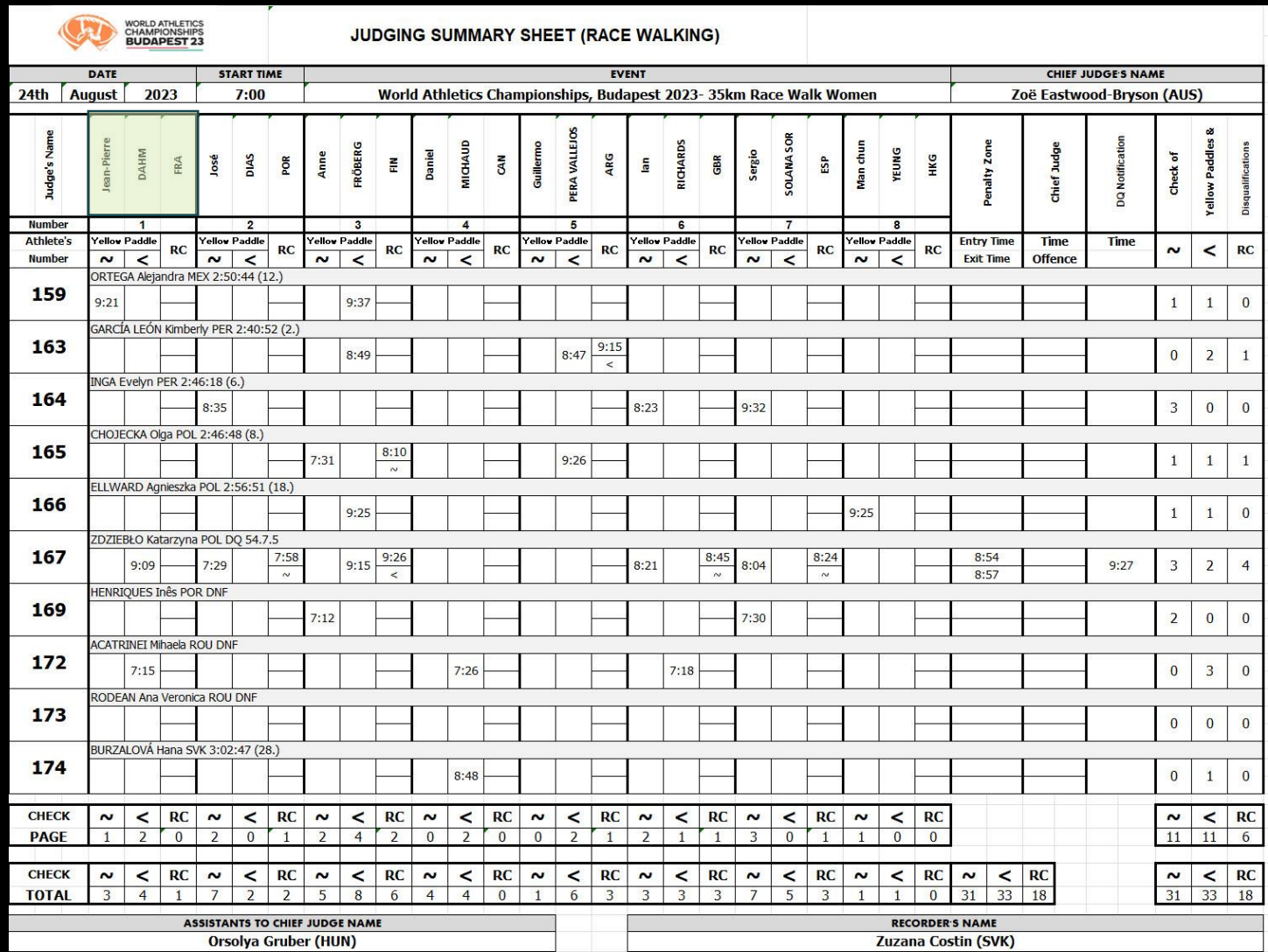
Judge			
IDJudge	**FirstName**	**LastName**	**CountryCode**
1	Jean Pierre	DAHM	FRA
2	José	DIAS	PER
3	Anne	FRÖBERG	FIN
4	Daniel	MICHAUD	CAN

This is not enough.

Judges work more than one event/race

A table is needed to connect the judge to the race.

RaceJudge	
IDRace	**IDJudge**
1	1
2	2
3	3
4	4



WORLD ATHLETICS CHAMPIONSHIPS BUDAPEST 23

JUDGING SUMMARY SHEET (RACE WALKING)

DATE	START TIME	EVENT	CHIEF JUDGE'S NAME
24th August 2023	7:00	World Athletics Championships, Budapest 2023- 35km Race Walk Women	Zoë Eastwood-Bryson (AUS)

Judge's Name	Jean-Pierre DAHM FRA			José DIAS POR			Anne FRÖBERG FIN			Daniel MICHAUD CAN			Guillermo PERA VALLEJOS ARG			Ian RICHARDS GBR			Sergio SOLANA SOR ESP			Man chun YEUNG HKG			Penalty Zone	Chief Judge	DQ Notification	Check of	Yellow Paddles &	Disqualifications
Number	1			2			3			4			5			6			7			8										
Athlete's Number	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Yellow Paddle ~	Yellow Paddle <	RC	Entry Time / Exit Time	Time / Offence	Time	~	<	RC
159	ORTEGA Alejandra MEX 2:50:44 (12.)																														
	9:21							9:37																				1	1	0
163	GARCÍA LEÓN Kimberly PER 2:40:52 (2.)																														
								8:49						8:47	9:15 <													0	2	1
164	INGA Evelyn PER 2:46:18 (6.)																														
				8:35												8:23			9:32									3	0	0
165	CHOJECKA Olga POL 2:46:48 (8.)																														
							7:31		8:10 ~					9:26														1	1	1
166	ELLWARD Agnieszka POL 2:56:51 (18.)																														
								9:25														9:25						1	1	0
167	ZDZIEBŁO Katarzyna POL DQ 54.7.5																														
		9:09		7:29		7:58 ~		9:15	9:26 <							8:21		8:45 ~	8:04		8:24 ~				8:54 / 8:57		9:27	3	2	4
169	HENRIQUES Inês POR DNF																														
							7:12												7:30									2	0	0
172	ACATRINEI Mihaela ROU DNF																														
		7:15									7:26						7:18											0	3	0
173	RODEAN Ana Veronica ROU DNF																														
																												0	0	0
174	BURZALOVÁ Hana SVK 3:02:47 (28.)																														
											8:48																	0	1	0

CHECK PAGE	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC				~	<	RC
	1	2	0	2	0	1	2	4	2	0	2	0	0	2	1	2	1	1	3	0	1	1	0	0				11	11	6
CHECK TOTAL	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC	~	<	RC
	3	4	1	7	2	2	5	8	6	4	4	0	1	6	3	3	3	3	7	5	3	1	1	0	31	33	18	31	33	18

ASSISTANTS TO CHIEF JUDGE NAME	RECORDER'S NAME
Orsolya Gruber (HUN)	Zuzana Costin (SVK)

Normalization – Creating a Data Model

Judges can take no action.

Judges can make calls.

When a judge makes a call, it can in one of four variations, but they can only make each call once per athlete.

Yellow (caution) for < (bent knee)
Yellow (caution) for ~ (loss of contact)
Red (proposal for disqualification) for < (bent knee)
Red (proposal for disqualification) for ~ (loss of contact)

When a judge makes a call, they also record:

- the time of day
- the bib number of the athlete
- their judge number
- the race

We can store it in a relation like:

JudgeCall

IDRace	IDJudge	Color	Infraction	TOD	BibNumber
1	1	Yellow	~	9:33 AM	181
1	2	Yellow	~	8:04 AM	142
1	4	Yellow	<	8:20 AM	183
1	1	Red	~	08:10 AM	181

Normalization – Creating a Data Model

Technically, `Yellow` and `Red` could be moved out to a reference table and IDs used instead.
`Red` and `Yellow` and not likely to change, they are small text fields, and make readability easier.
Either way would be fine.

JudgeCall					
IDRace	**IDJudge**	**IDColor**	**Infraction**	**TOD**	**BibNumber**
1	1	1	~	9:33 AM	181
1	2	1	~	8:04 AM	142
1	4	1	<	8:20 AM	183
1	1	2	~	08:10 AM	181

Color	
IDColor	**Color**
1	Yellow
2	Red

Normalization – Creating a Data Model

Race Walking Data Model:

Event

idEvent	Event	City	Country
1	World Athletics Championships	Budapest	Hungary
2	World Athletics Team Championships	Anatalya	Turkey
3	Olympics Games	Paris	France

Race

idRace	IDEvent	RaceDate	StartTime	Distance	DistanceUnits	Gender
1	1	2023-08-24	7:00 AM	35	km	F
2	1	2023-08-24	7:00 AM	35	km	M
3	1	2023-08-20	7:15 AM	20	km	F
4	1	2023-08-19	10:50 AM	20	km	M

Athlete

idAthlete	FirstName	LastName	CountryCode	Gender
1	Alejandra	ORTEGA	MEX	F
2	Kimberly	GARCÍA LEÓN	PER	F
3	Evelyn	INGA	PER	F
4	Olga	CHOKECKA	POL	F

Bib

IDRace	IDAthlete	BibNumber	FinishingTime	FinishingPlace	Finished
1	1	159	2:50:44	12	Yes
1	2	163	2:40:52	2	Yes
1	3	164	2:46:18	6	Yes
1	4	165	2:46:48	20	Yes

Judge

IDJudge	FirstName	LastName	CountryCode
1	Jean Pierre	DAHM	FRA
2	José	DIAS	PER
3	Anne	FRÖBERG	FIN
4	Daniel	MICHAUD	CAN

RaceJudge

IDRace	IDJudge
1	1
2	2
3	3
4	4

JudgeCall

idRace	IDJudge	Color	Infraction	TOD	BibNumber
1	1	Yellow	~	9:33 AM	181
1	2	Yellow	~	8:04 AM	142
1	4	Yellow	<	8:20 AM	183
1	1	Red	~	08:10 AM	181

We have more work to do, let's identify the primary keys of each table by underlining the fields that comprise the primary key

Normalization – Creating a Data Model

Race Walking Data Model:

Event

IDEvent	Event	City	Country
1	World Athletics Championships	Budapest	Hungary
2	World Athletics Team Championships	Anatalya	Turkey
3	Olympics Games	Paris	France

Race

IDRace	IDEvent	RaceDate	StartTime	Distance	DistanceUnits	Gender
1	1	8/24/2023	7:00 AM	35	km	F
2	1	8/24/2023	7:00 AM	35	km	M
3	1	8/20/2023	7:15 AM	20	km	F
4	1	8/19/2023	10:50 AM	20	km	M

Athlete

IDAthlete	FirstName	LastName	CountryCode
1	Alejandra	ORTEGA	MEX
2	Kimberly	GARCÍA LEÓN	PER
3	Evelyn	INGA	PER
4	Olga	CHOKECKA	POL

Bib

IDRace	IDAthlete	BibNumber	FinishingTime	FinishingPlace	Finished
1	1	159	2:50:44	12	Yes
1	2	163	2:40:52	2	Yes
1	3	164	2:46:18	6	Yes
1	4	165	2:46:48	20	Yes

Judge

IDJudge	FirstName	LastName	CountryCode
1	Jean Pierre	DAHM	FRA
2	José	DIAS	PER
3	Anne	FRÖBERG	FIN
4	Daniel	MICHAUD	CAN

RaceJudge

IDRace	IDJudge
1	1
2	2
3	3
4	4

JudgeCall

IDRace	IDJudge	Color	Infraction	TOD	BibNumber
1	1	Yellow	~	9:33 AM	181
1	2	Yellow	~	8:04 AM	142
1	4	Yellow	<	8:20 AM	183
1	1	Red	~	08:10 AM	181

We have more work to do, let's identify the primary keys of each table by underlining the fields that comprise the primary key

Normalization – Creating a Data Model

We need two more tables to store manual visual video observations made about the race
Create an *Observer* table to store the people making the observations with a Primary Key of *IDObserver*:

Observer		
IDObserver	**FirstName**	**LastName**
1	Jeff	Salvage
2	David	Harriman
3	Gary	Westerfield

Create an *VideoObserveration* table to store observations of the race walkers with a Primary Key of *ID*:

VideoObservation												
ID	**IDRace**	**BibNumber**	**IDObserver**	**LOCAverage**	**LOC1**	**LOC2**	**LOC3**	**LOC4**	**BentKneeAngle**	**Comment**	**VideoFile**	**TOD**
1	1	159	2	46.9	5	6	5.5	6	0		C1002	09:10 AM
2	1	163	1		6	6	6	6.5	0			
3	1	164	1						176.4			

Normalization – Installing SQLite & DBVisualizer

"Install" SQLite"
https://www.sqlite.org/download.html

Download the file for your OS containing the following:
A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff.exe program, and the sqlite3_analyzer.exe program. 64-bit.

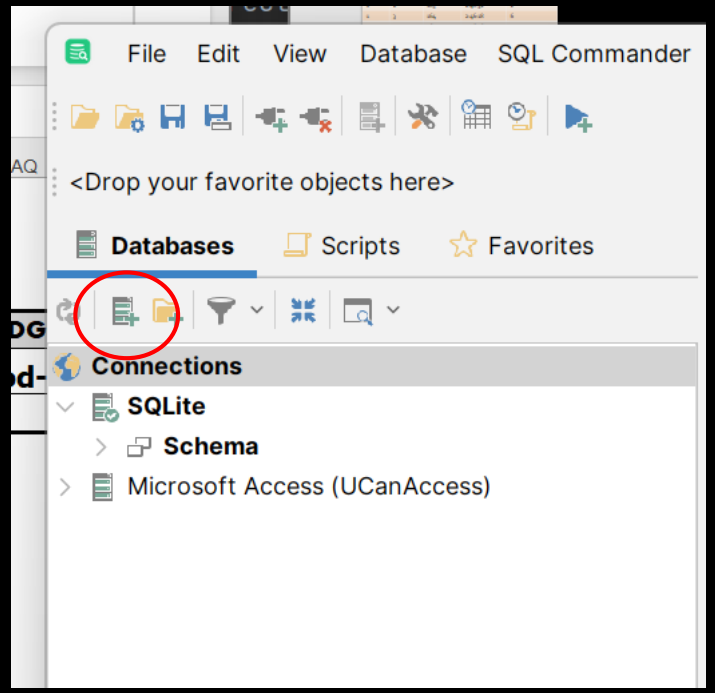
Unzip it.

Download and install DBVisualizer:
https://www.dbvis.com/download/

Use the license located in Learn.
Please do not share it.

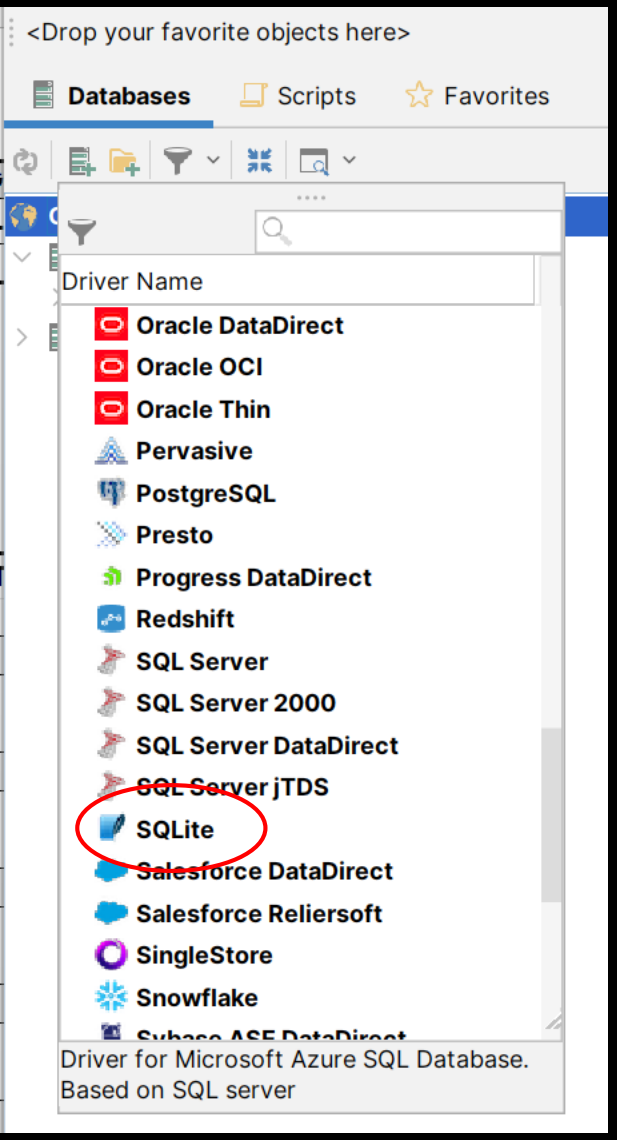
Normalization – Creating a Database Connection

Click on the icon to create a new database connection:



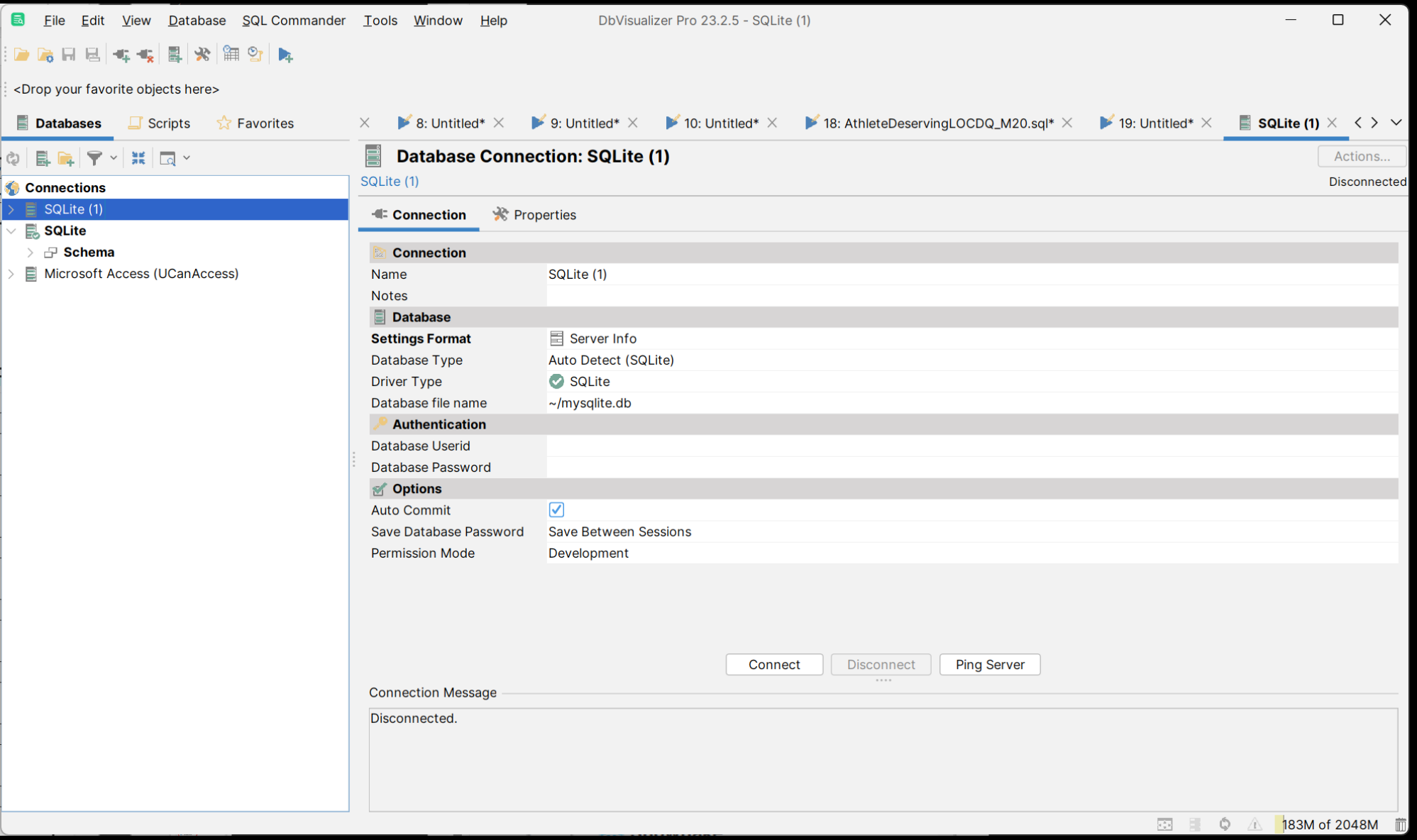
Normalization – Creating a Database Connection

Select *SQLite* from the *Drive Name* pulldown:



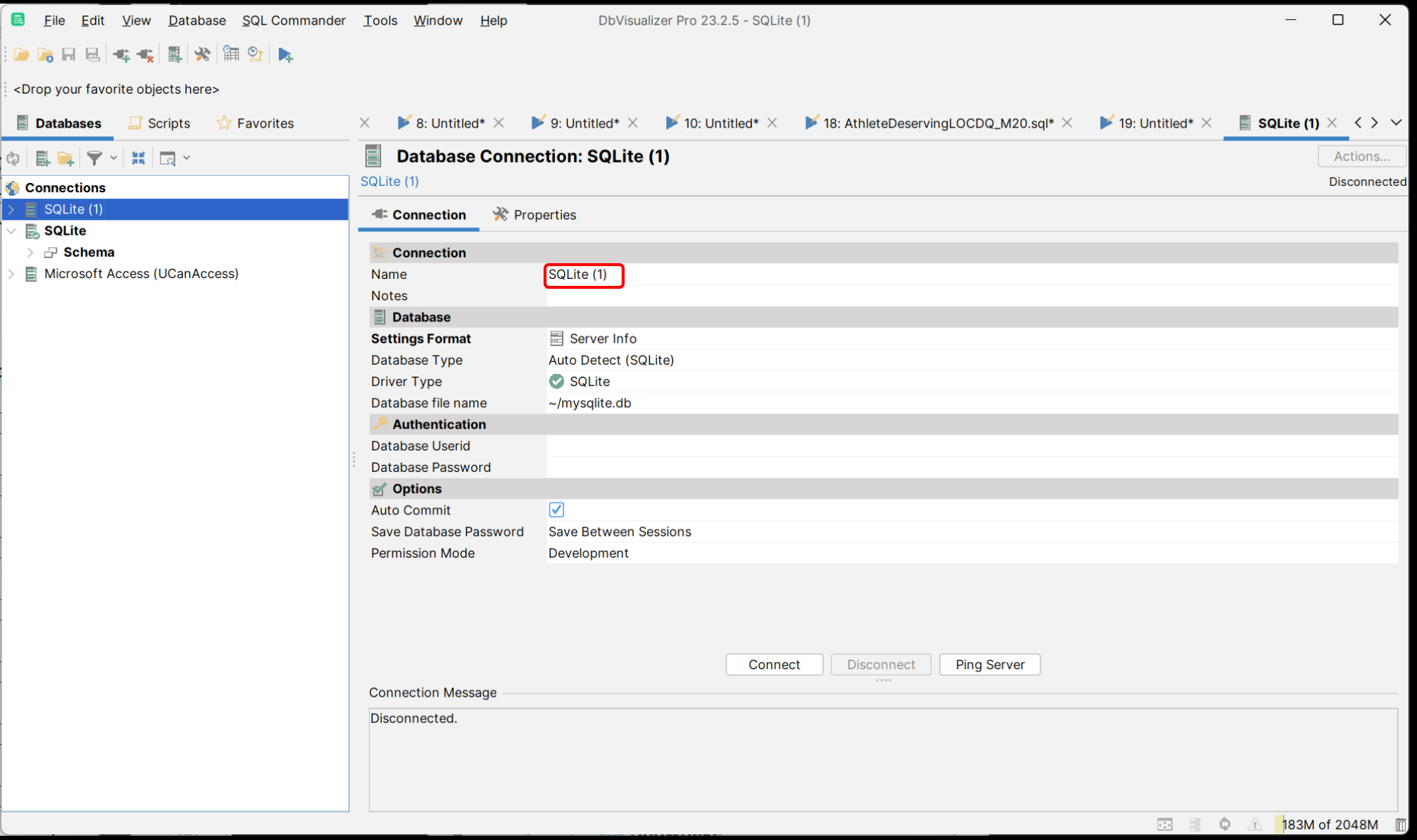
Normalization – Creating a Database Connection

A new *Connection* dialog is presented:



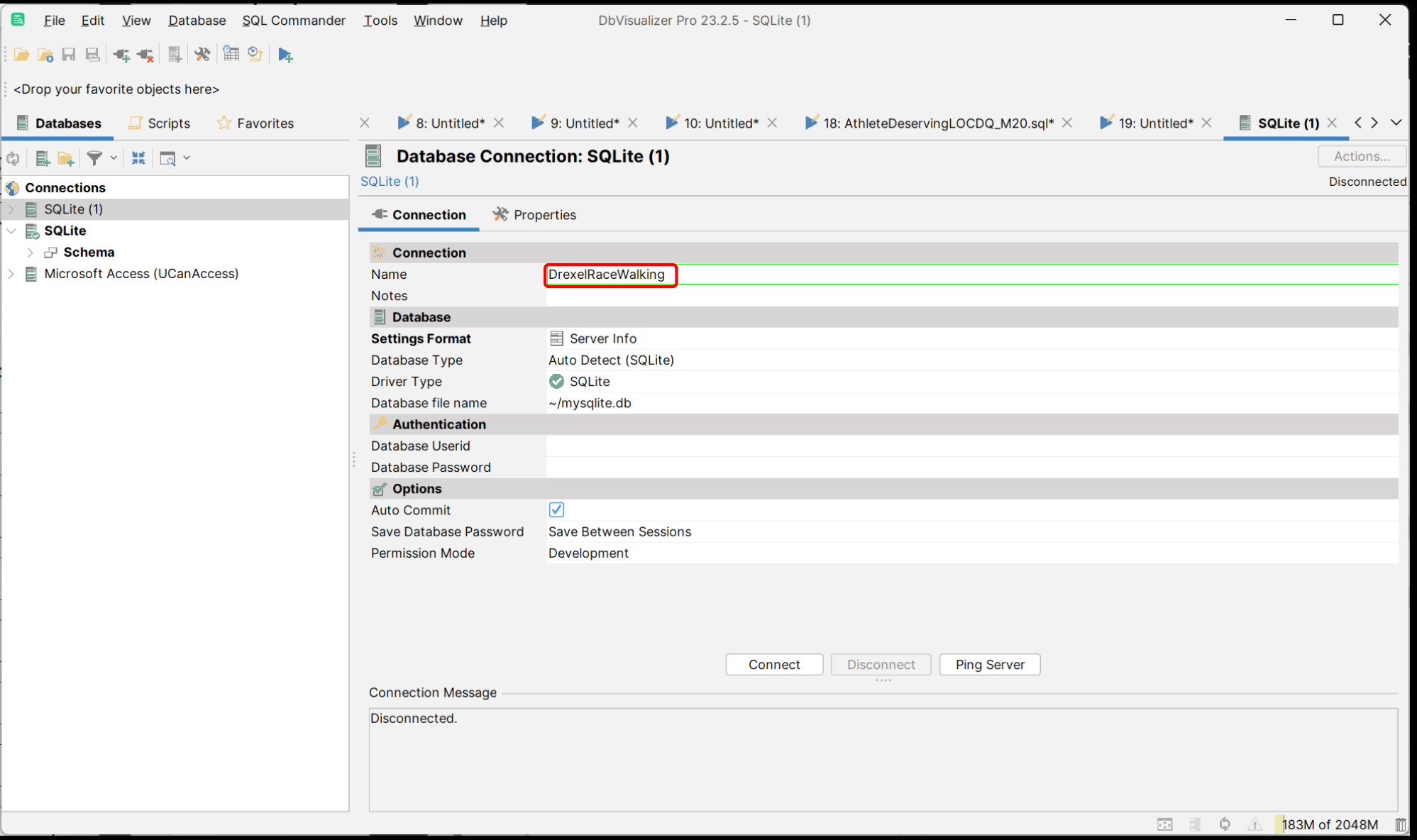
Normalization – Creating a Database Connection

Change the *Name* of the connection to something appropriate like *DrexelRaceWalking*



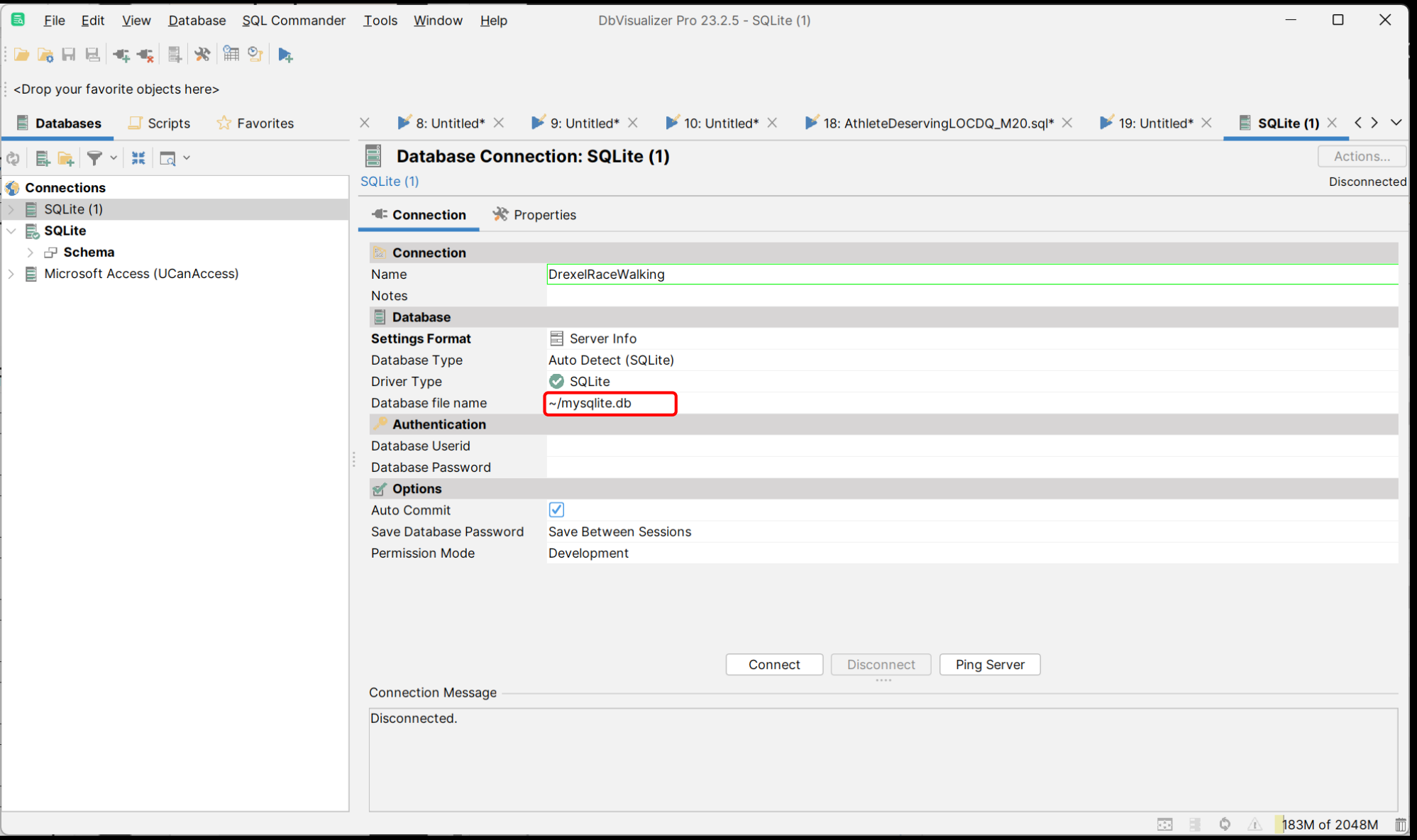
Normalization – Creating a Database Connection

Change the *Name* of the connection to something appropriate like *DrexelRaceWalking*



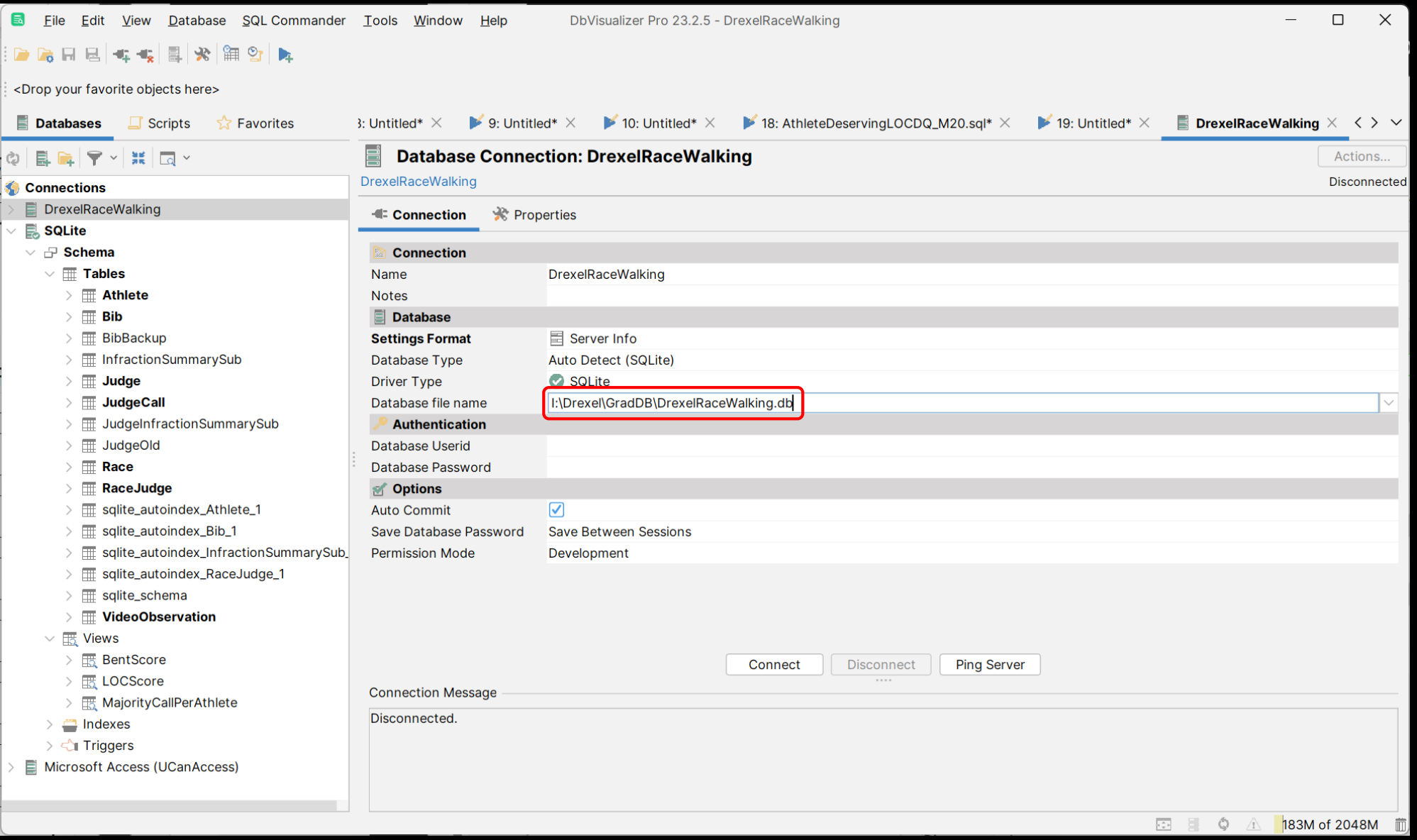
Normalization – Creating a Database Connection

Change the location / name of the physical file the database is stored within:



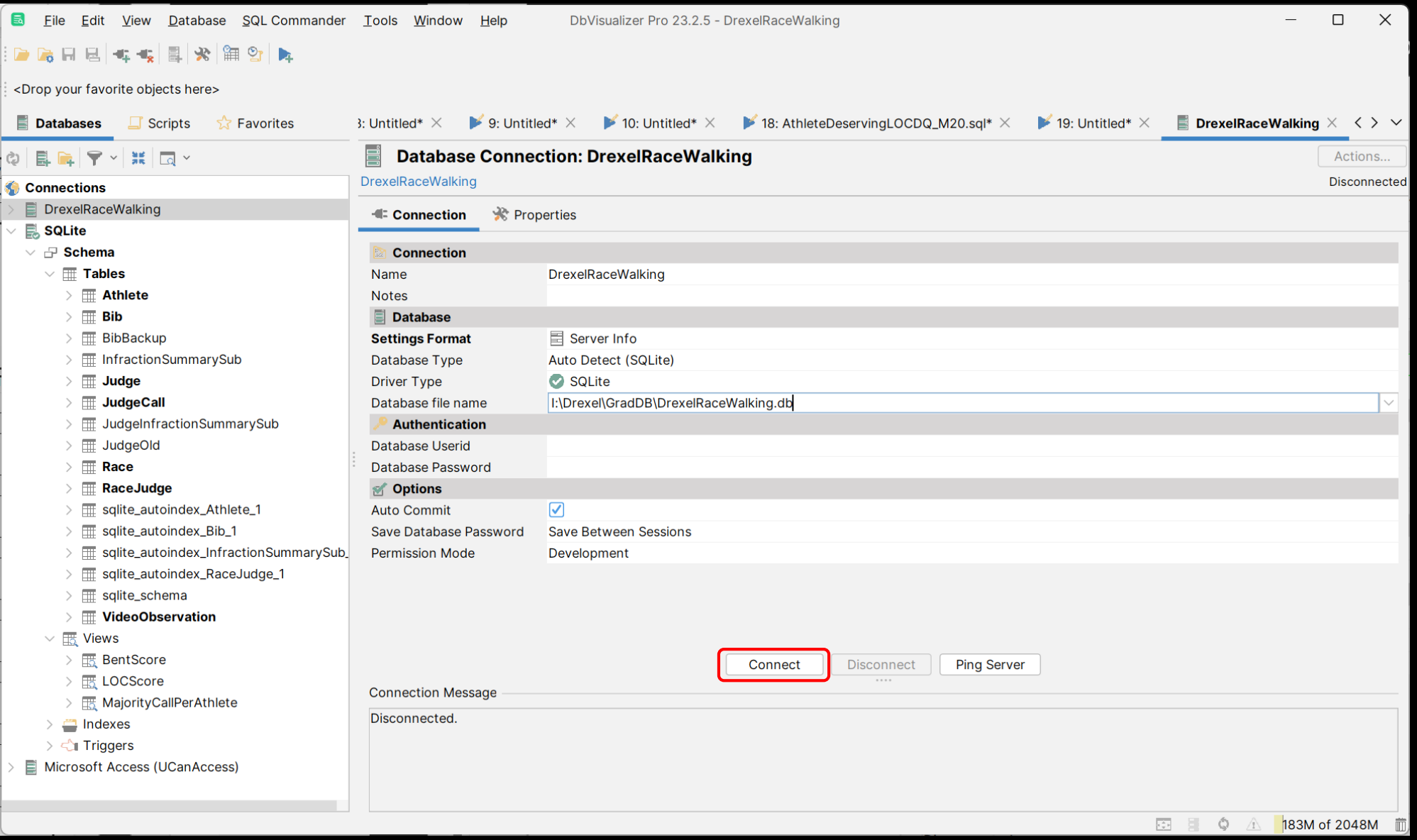
Normalization – Creating a Database Connection

Change the location / name of the physical file the database is stored within:



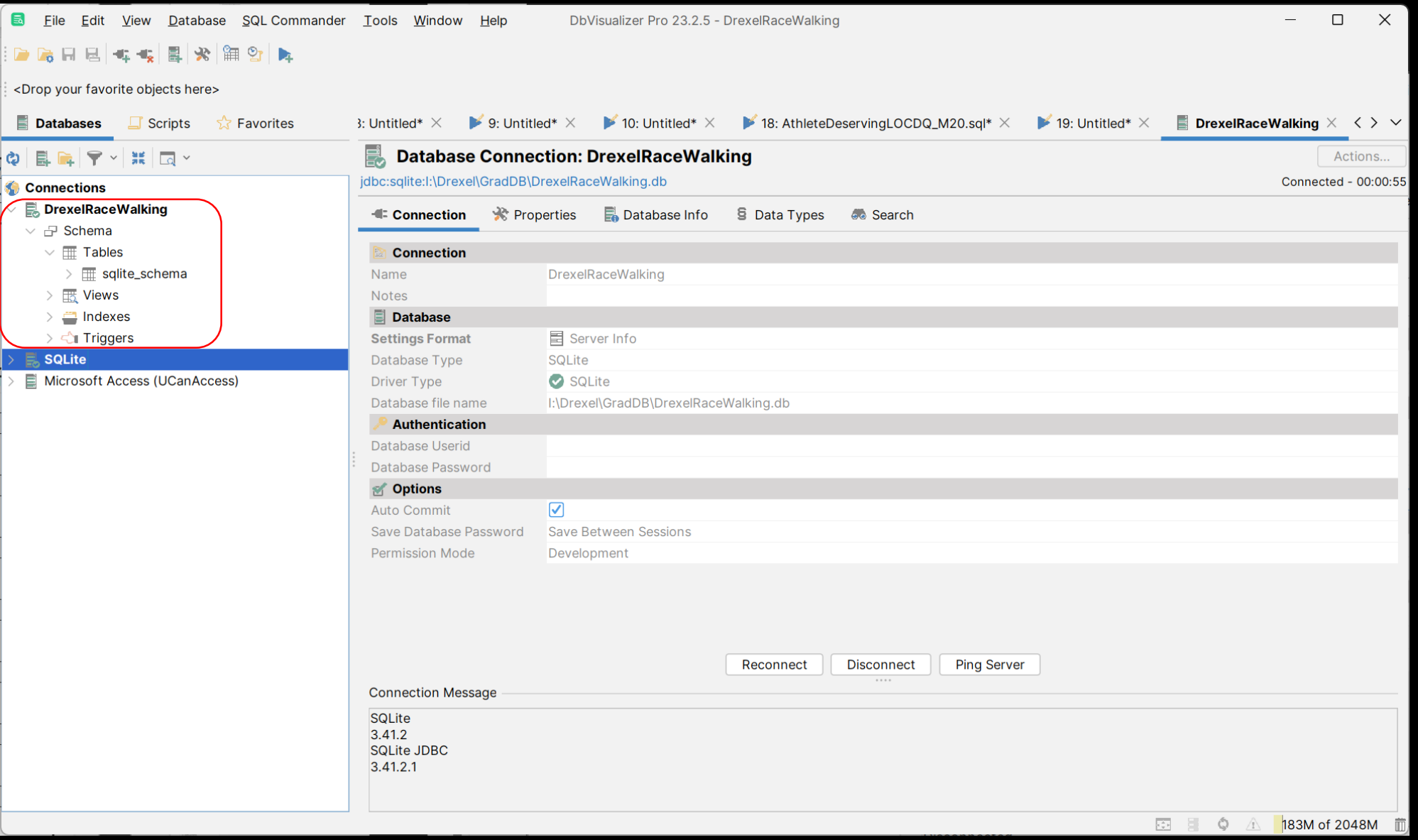
Normalization – Creating a Database Connection

Click on the *Connect* button:



Normalization – Creating a Database Connection

Your new database appears and can be expanded on the left:

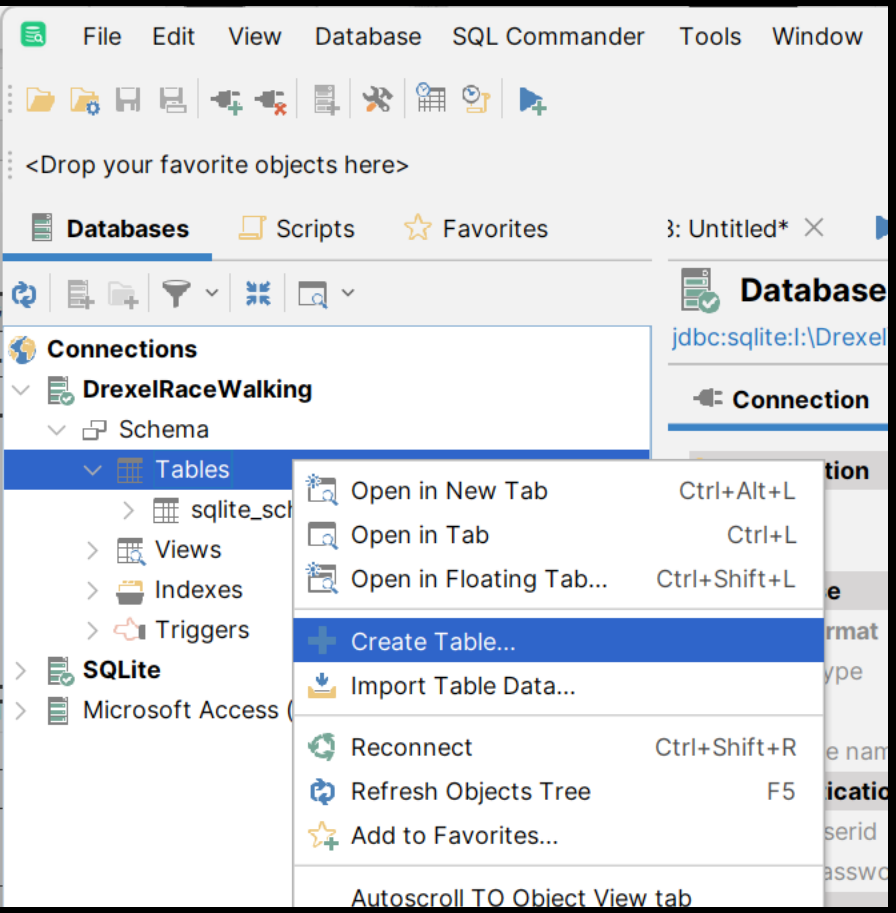


Currently, there are no tables.

Creating Tables - GUI

We can create tables with the GUI or by SQL.

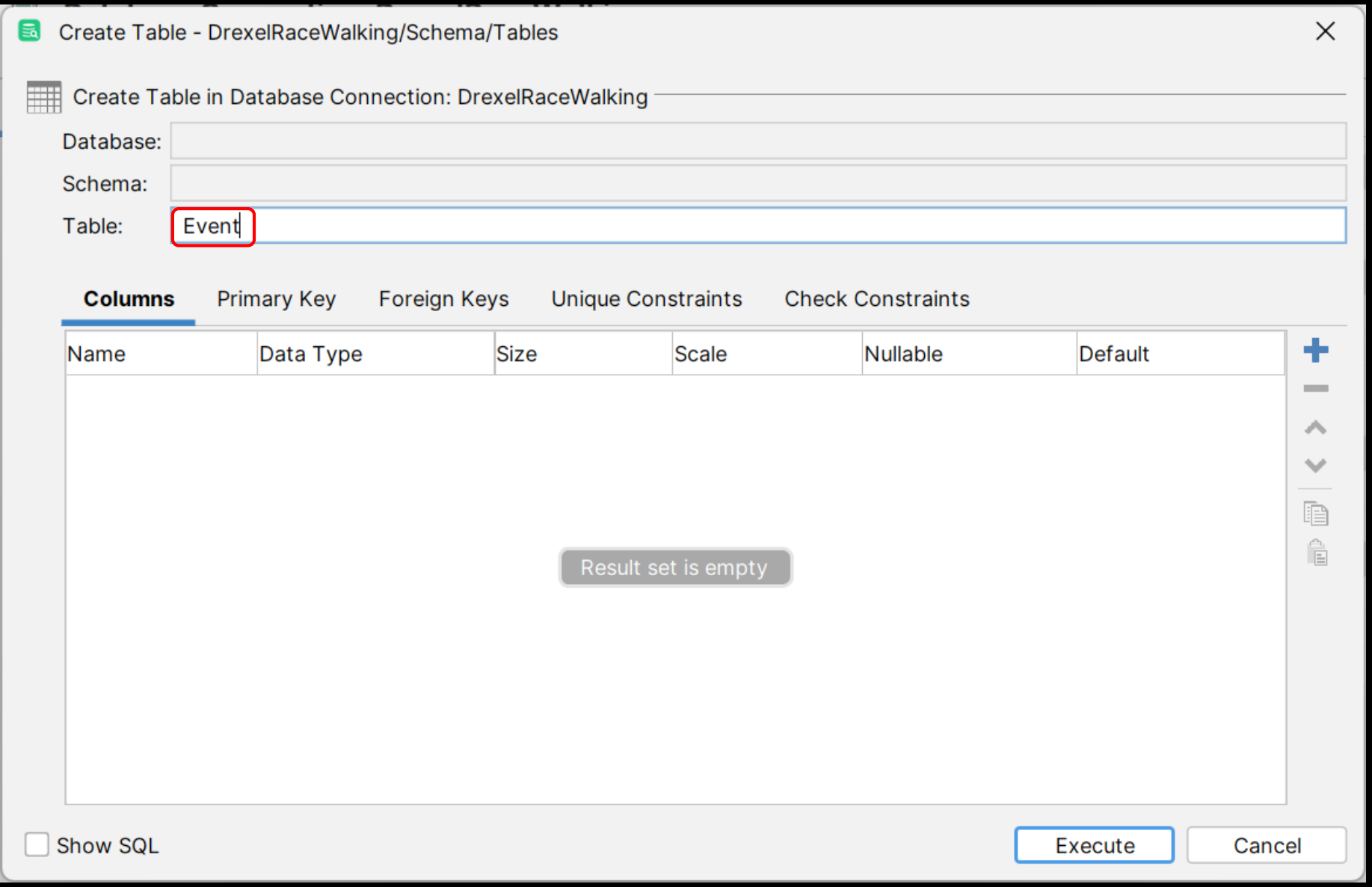
Use the GUI.
Right Click on *Tables*.
Select *Create Table…* from the popup:



Event			
idEvent	**Event**	**City**	**Country**
1	World Athletics Championships	Budapest	Hungary
2	World Athletics Team Championships	Anatalya	Turkey
3	Olympics Games	Paris	France

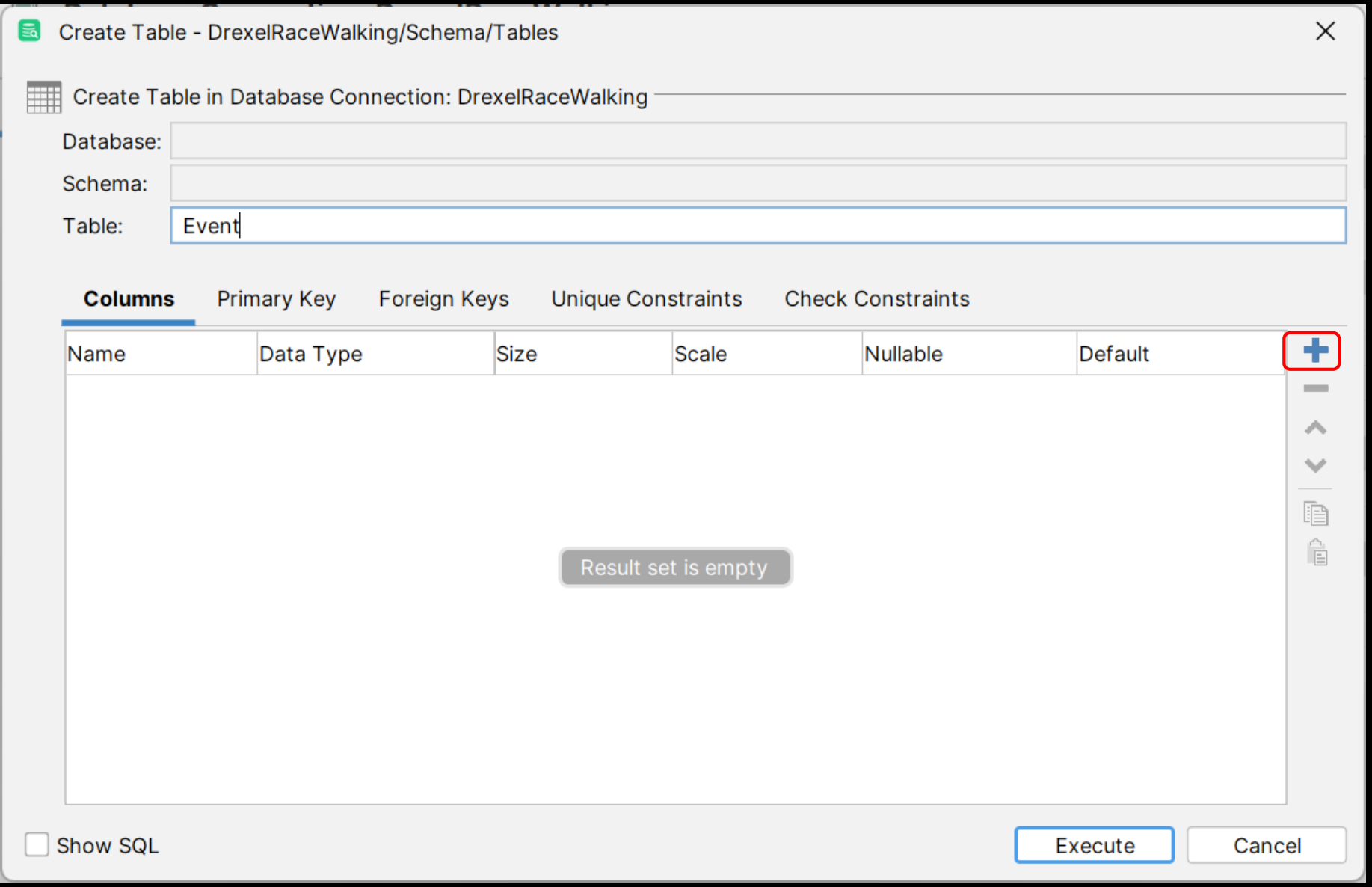
Creating Tables - GUI

Enter the table name, *Event*:



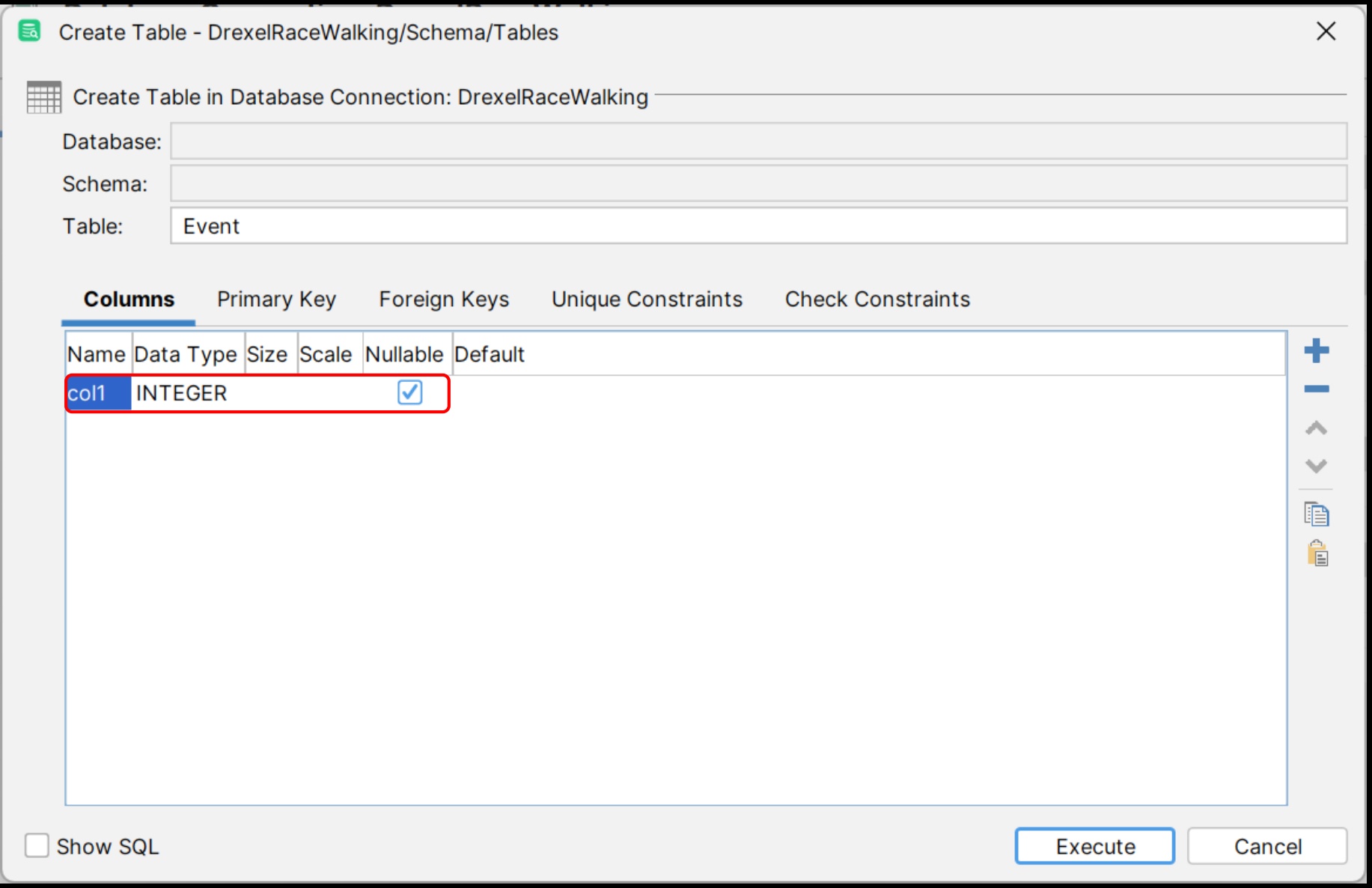
Creating Tables - GUI

Add the columns / fields, one at a time, by clicking on the + icon:



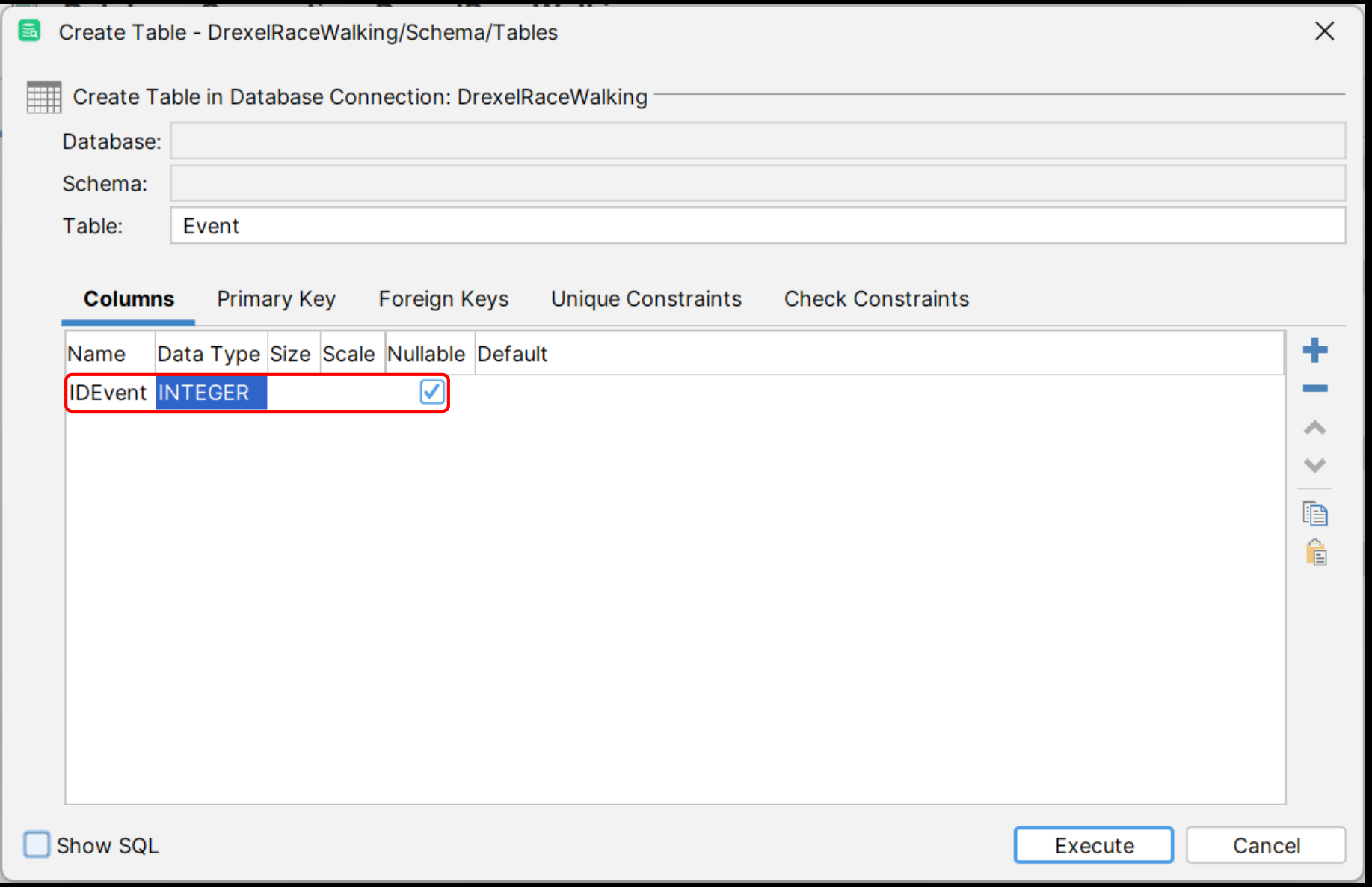
Creating Tables - GUI

Add the *IDEvent* field, after clicking on the +, default values appear:



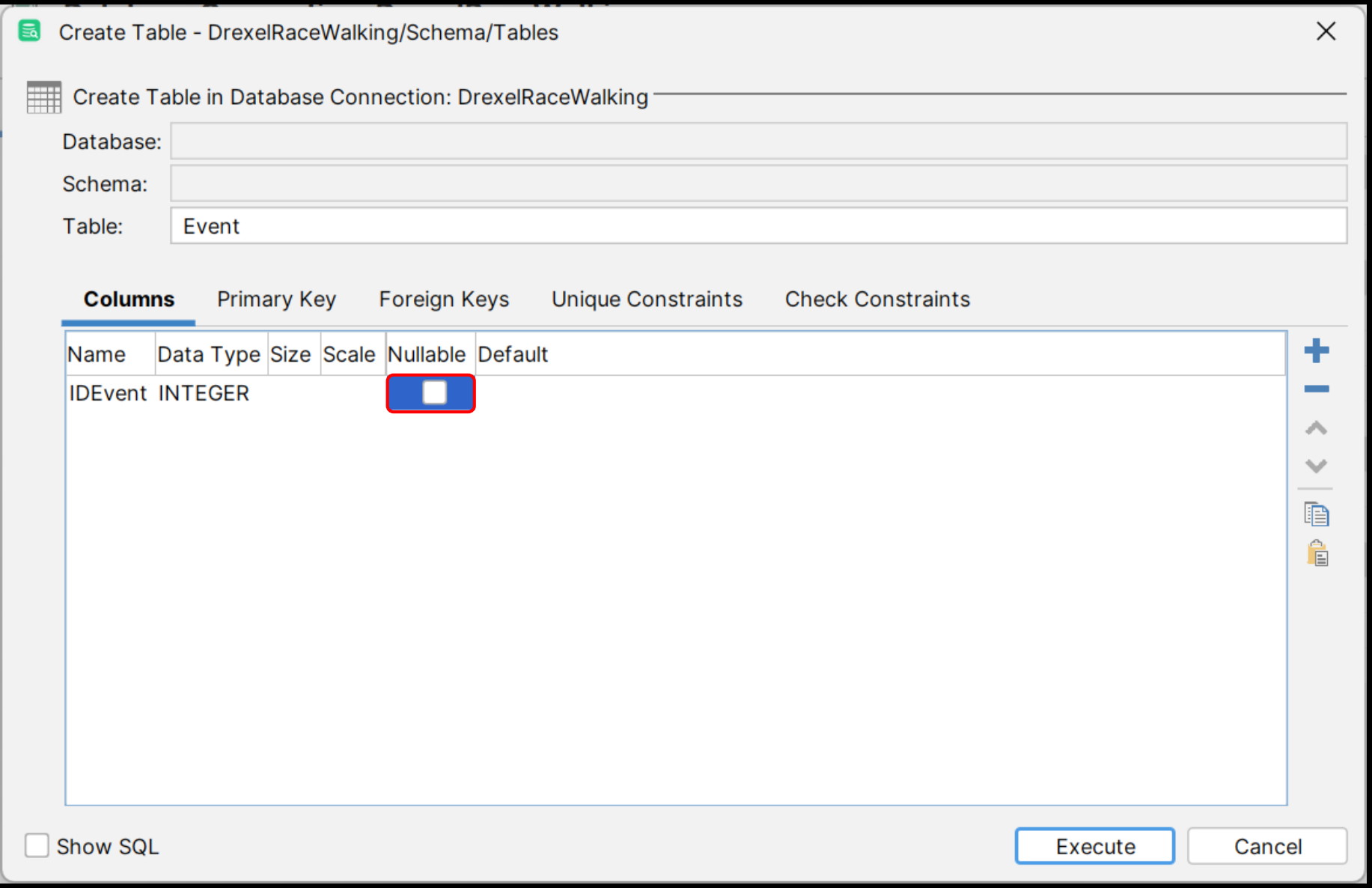
Creating Tables - GUI

Add the *IDEvent* field, change the values to *IDEvent*, leave *INTEGER*:



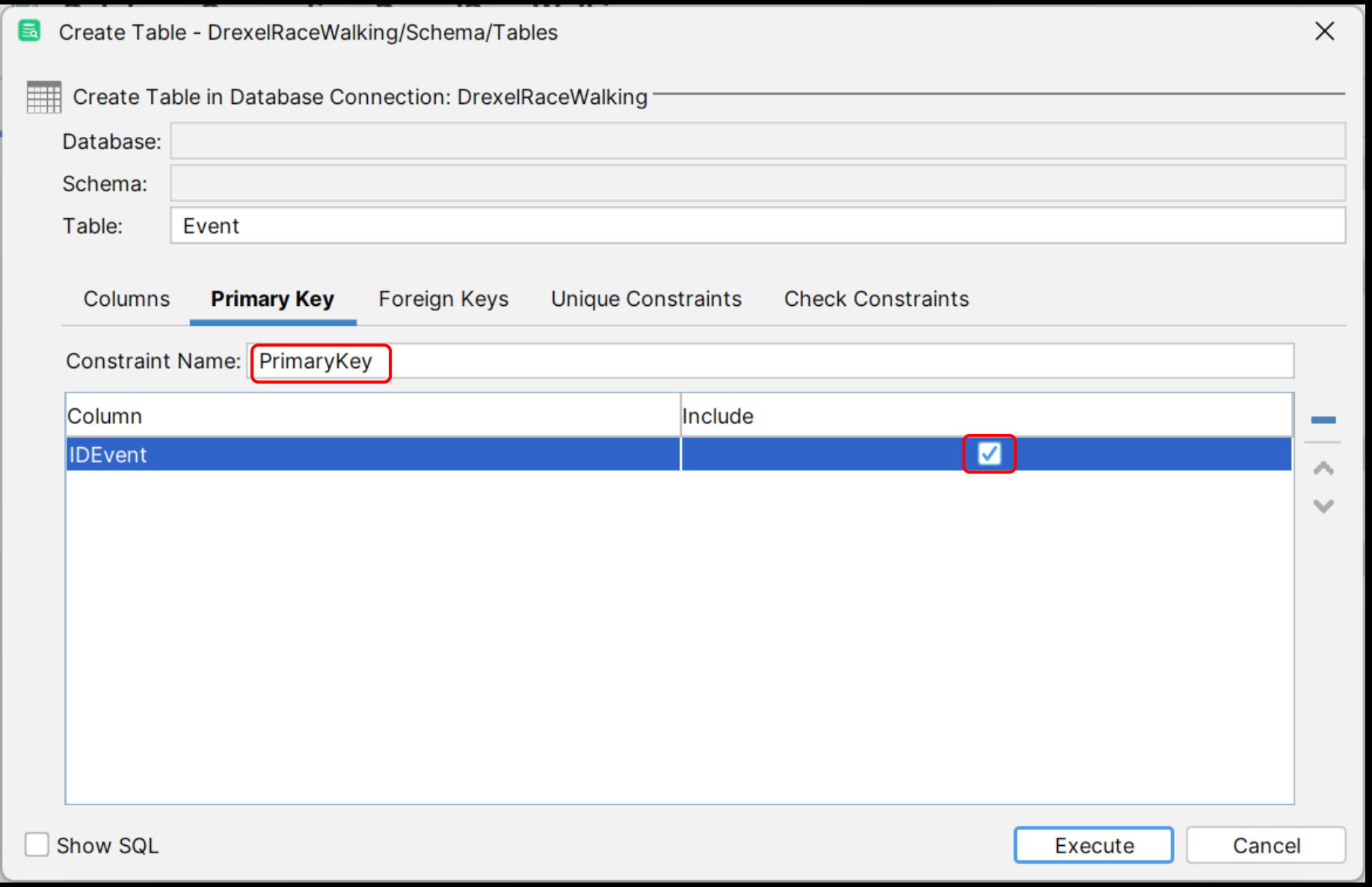
Creating Tables - GUI

Uncheck *Nullable*, because *IDEvent* is the primary key.



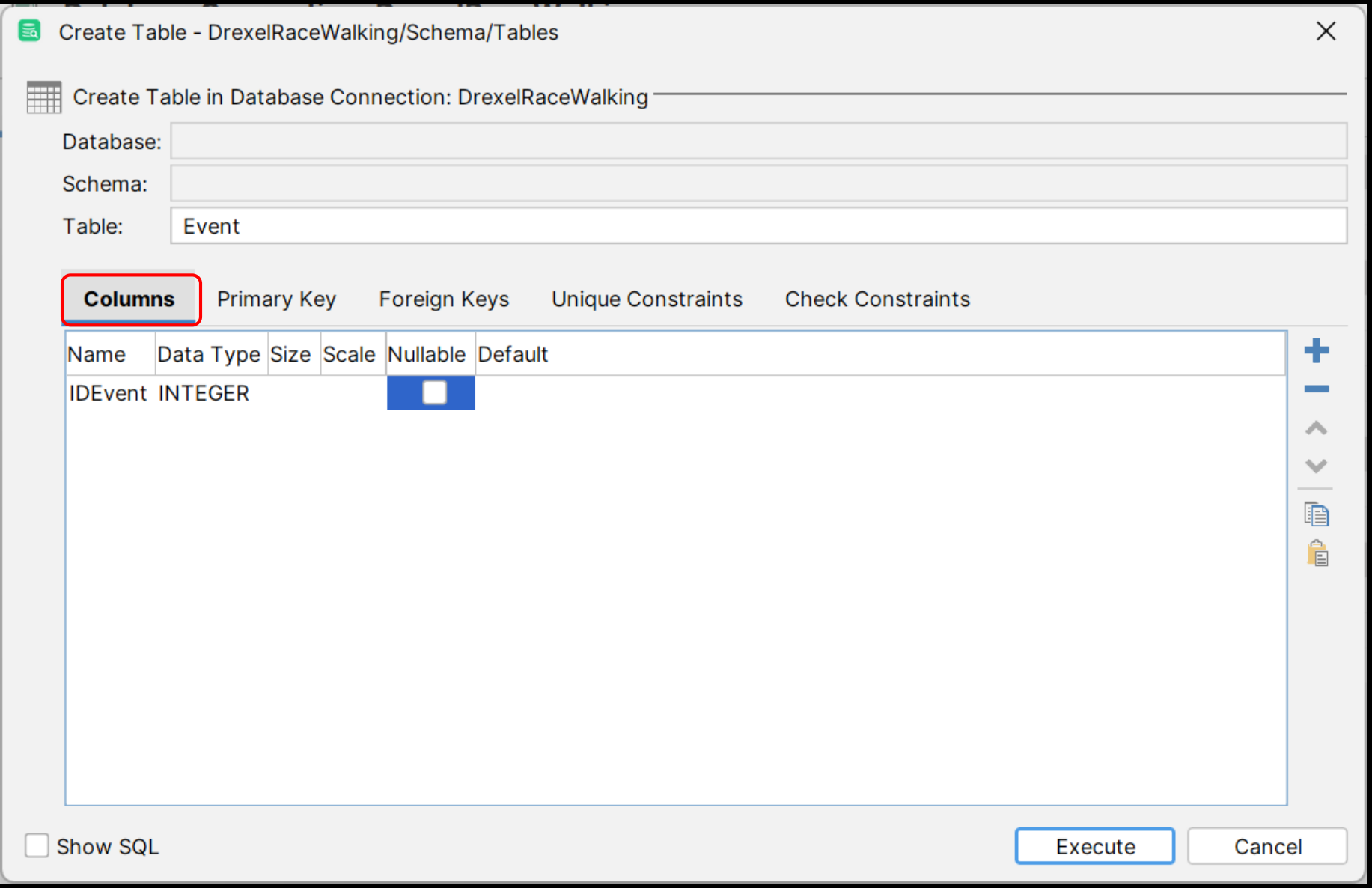
Creating Tables - GUI

Click on the *Primary Key* tab, give the *Constraint Name* a value, and check the box to include the *IDEvent* field:



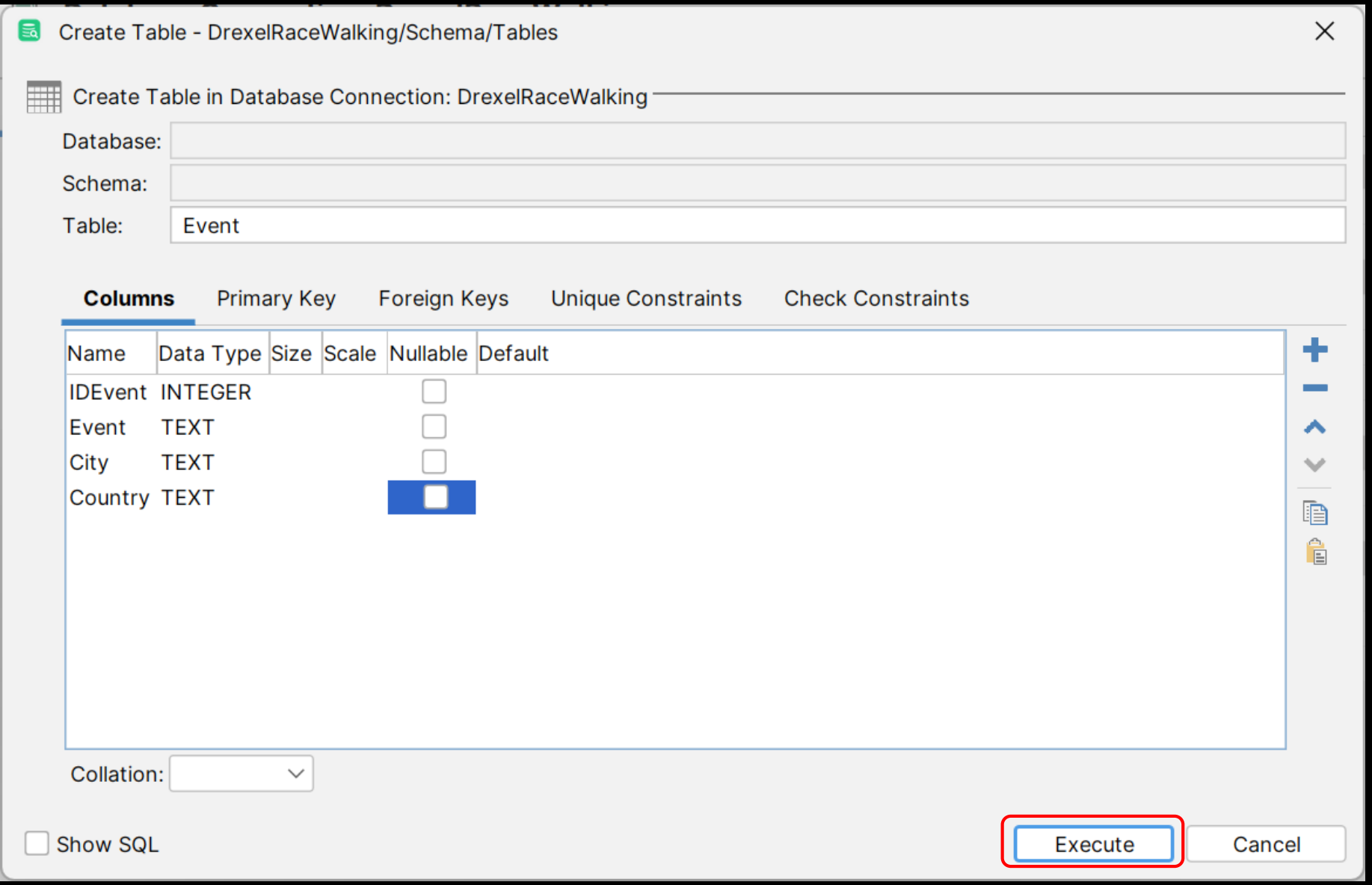
Creating Tables - GUI

Click on the *Columns* tab so we can enter the rest of the fields:



Creating Tables - GUI

Enter the remaining fields, all should not be allowed to contain nulls, then click *Execute*:

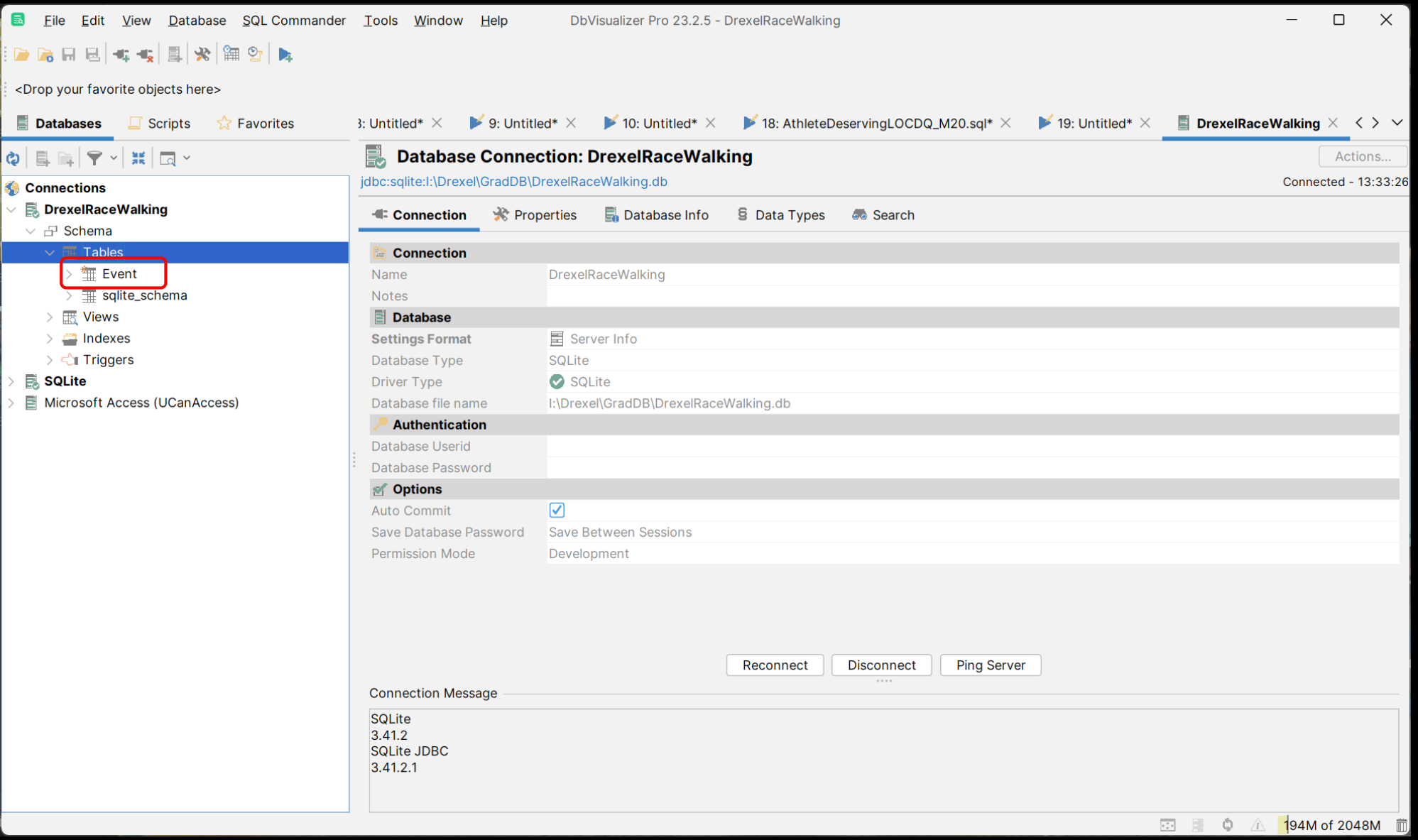


Creating Tables - GUI

Click *Yes* to confirm the table creation:

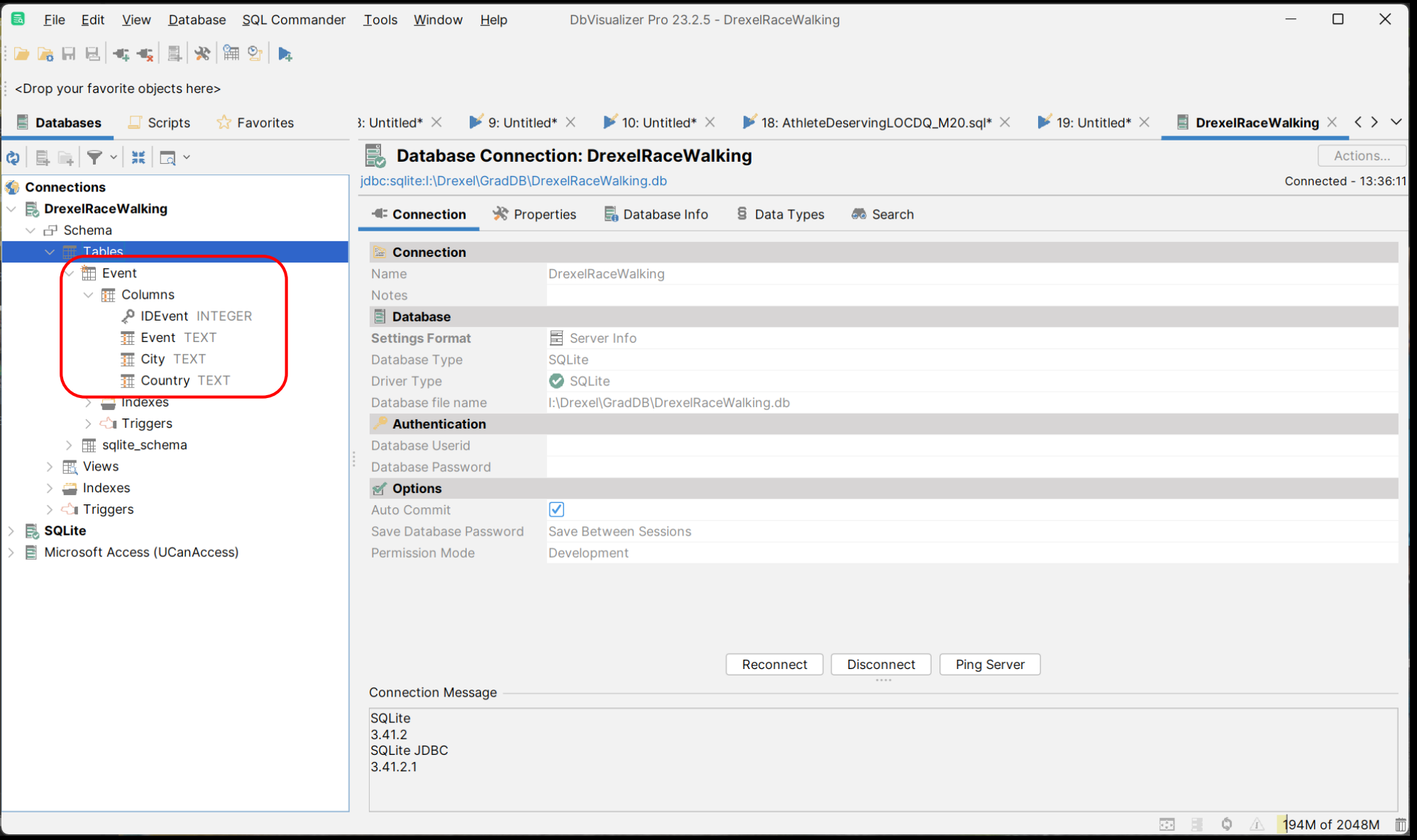
Creating Tables - GUI

Enter the remaining fields, all should not be allowed to contain nulls, then click *Execute*:



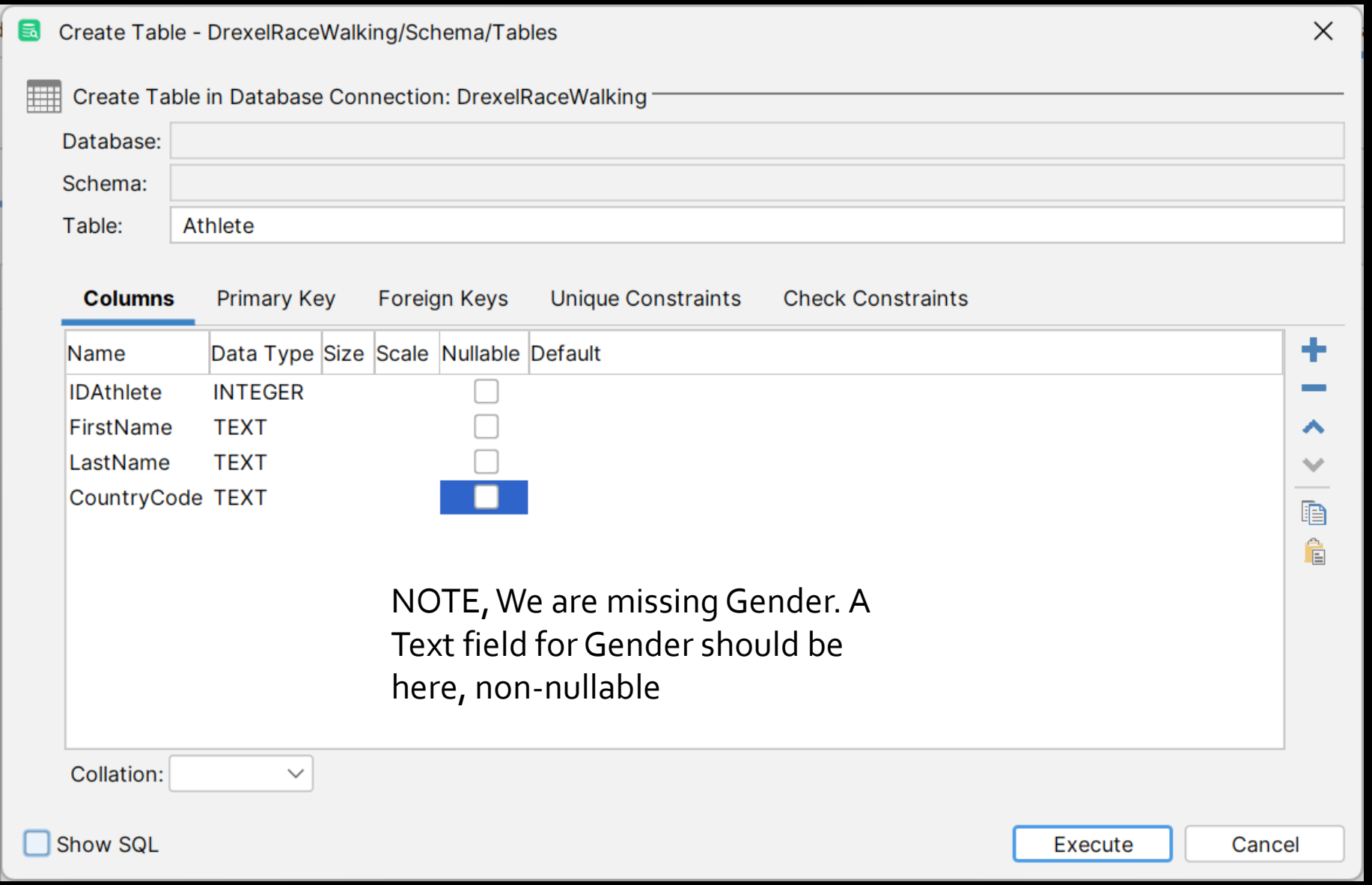
Creating Tables - GUI

Quickly spy the columns using the arrows:



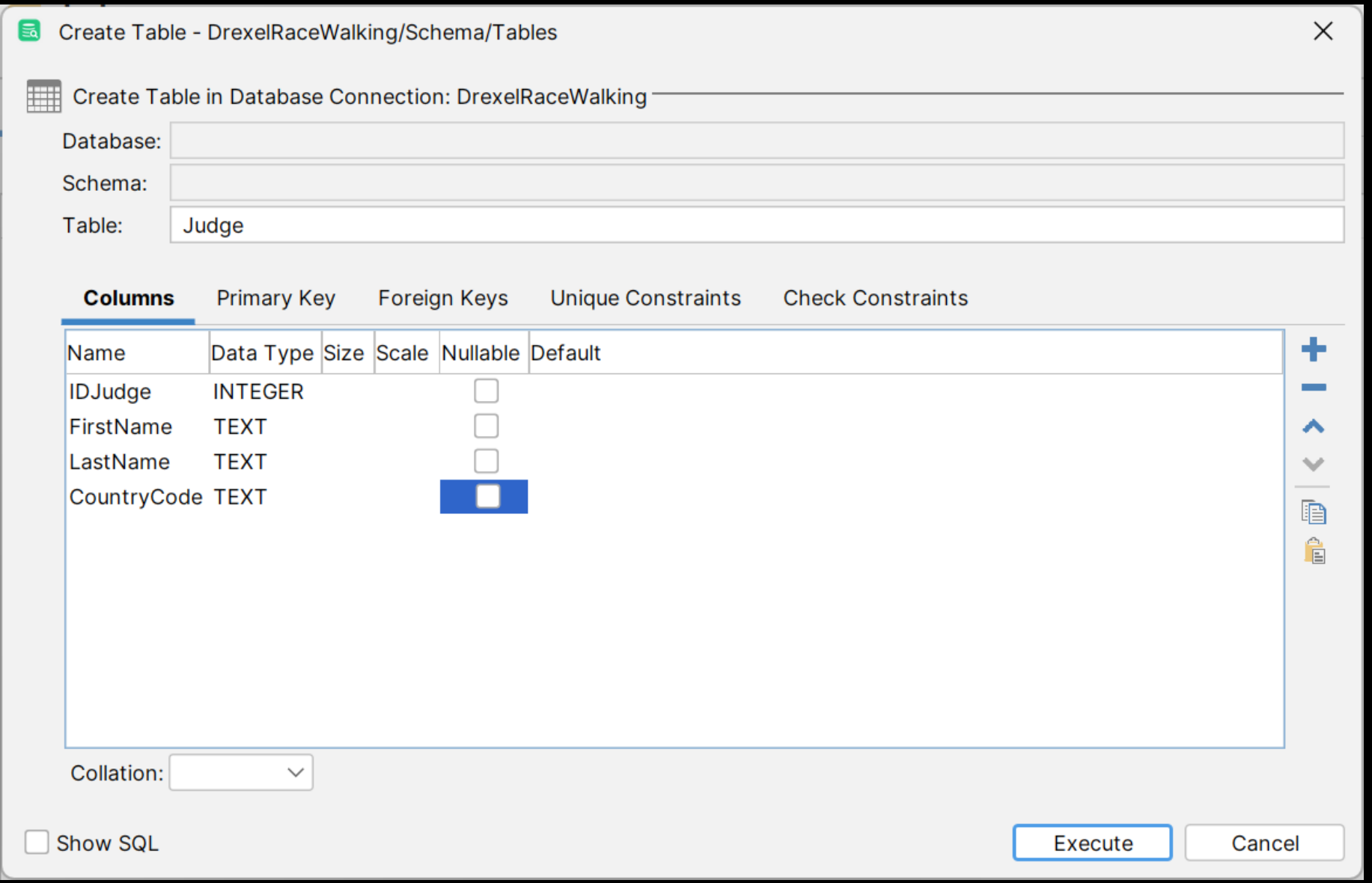
Creating Tables - GUI

Add the *Athlete* table in the same manner (Primary Key is *IDAthlete*):



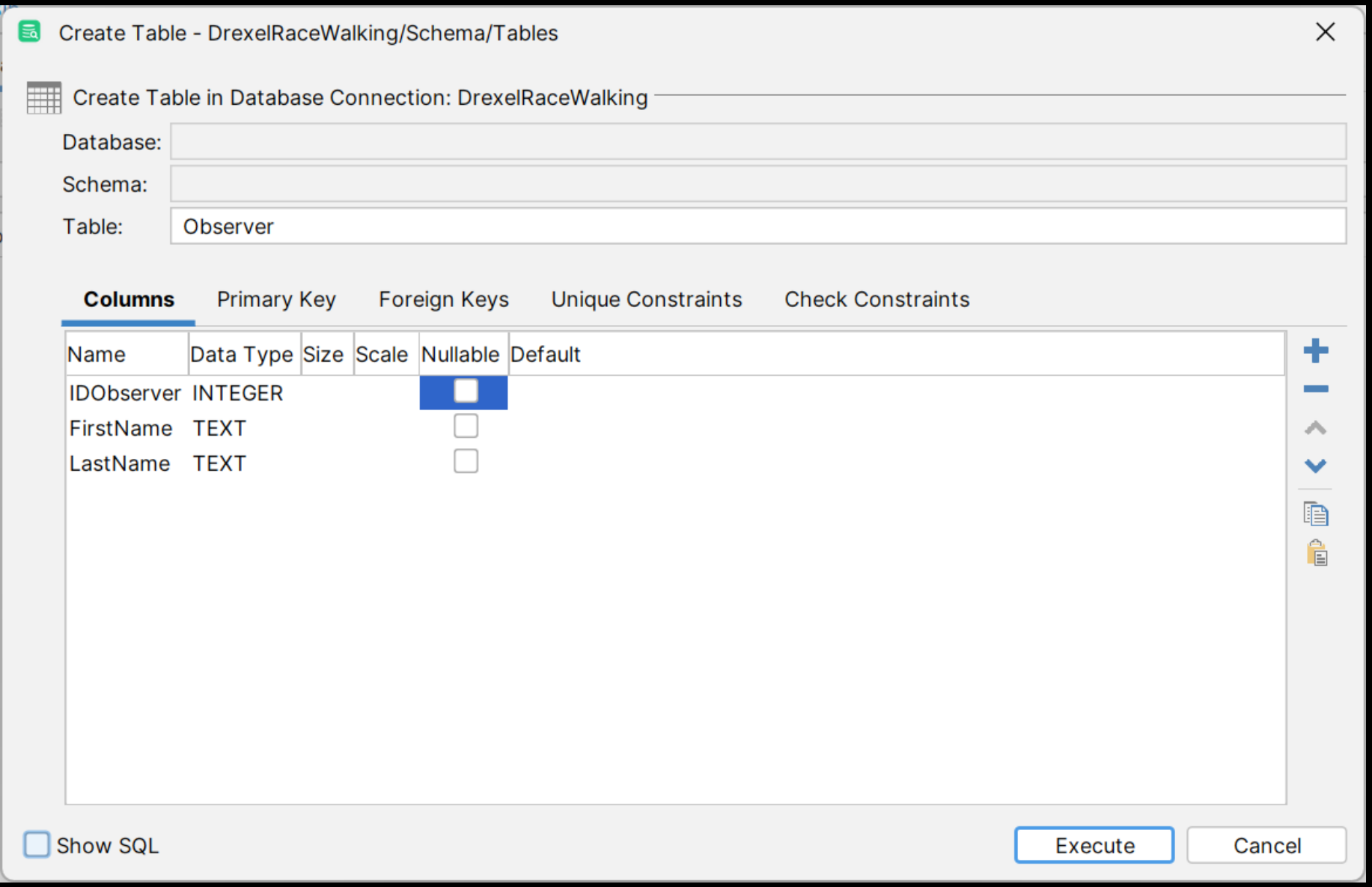
Creating Tables - GUI

Add the *Judge* table in the same manner (Primary Key is *IDJudge*):



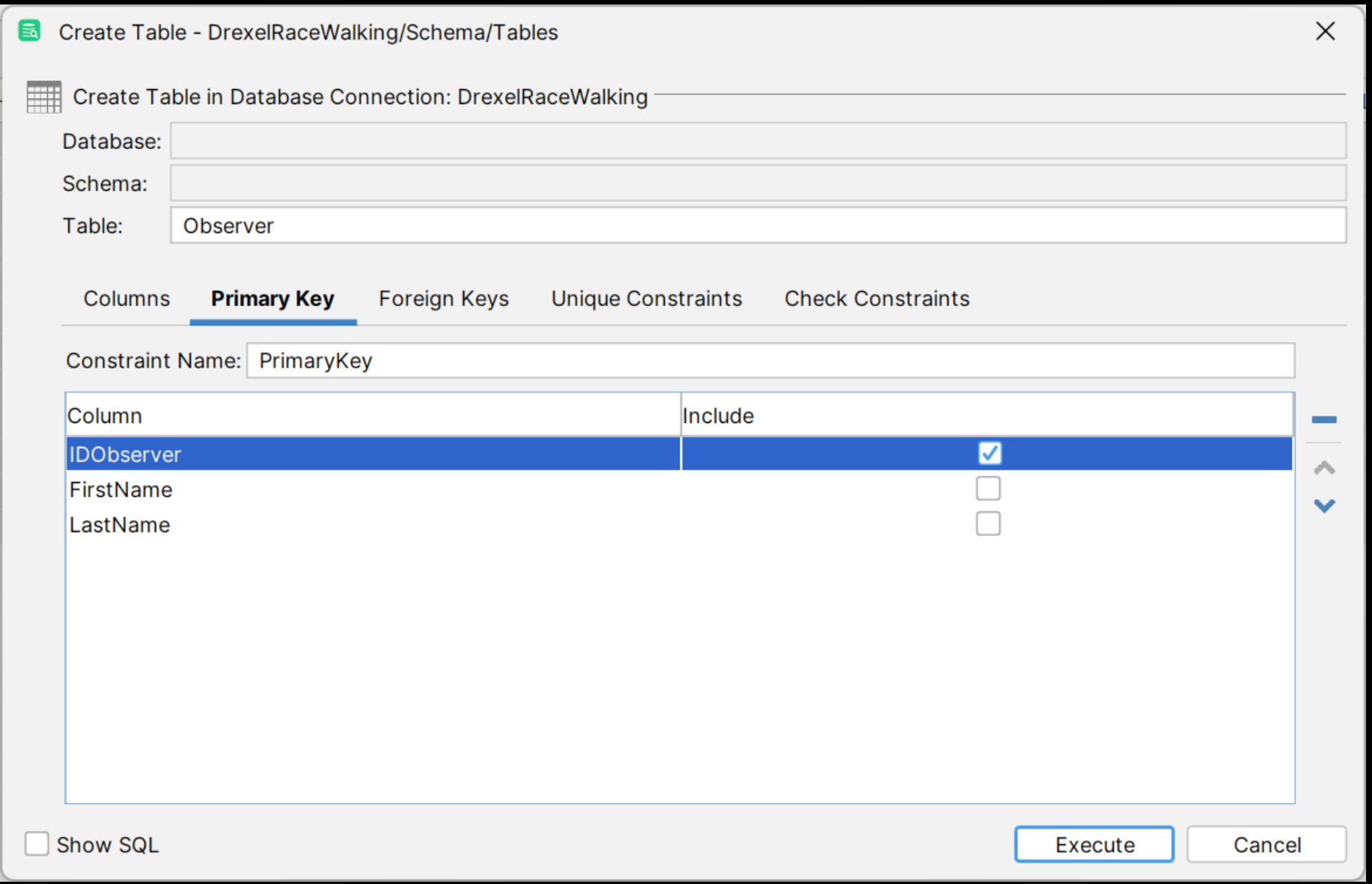
Creating Tables - GUI

Add the rows for the *Observer* table in the same manner:



Creating Tables - GUI

Add the Primary Key for the *Observer* table (*IDObserver*):



Creating Tables - GUI

The tables:

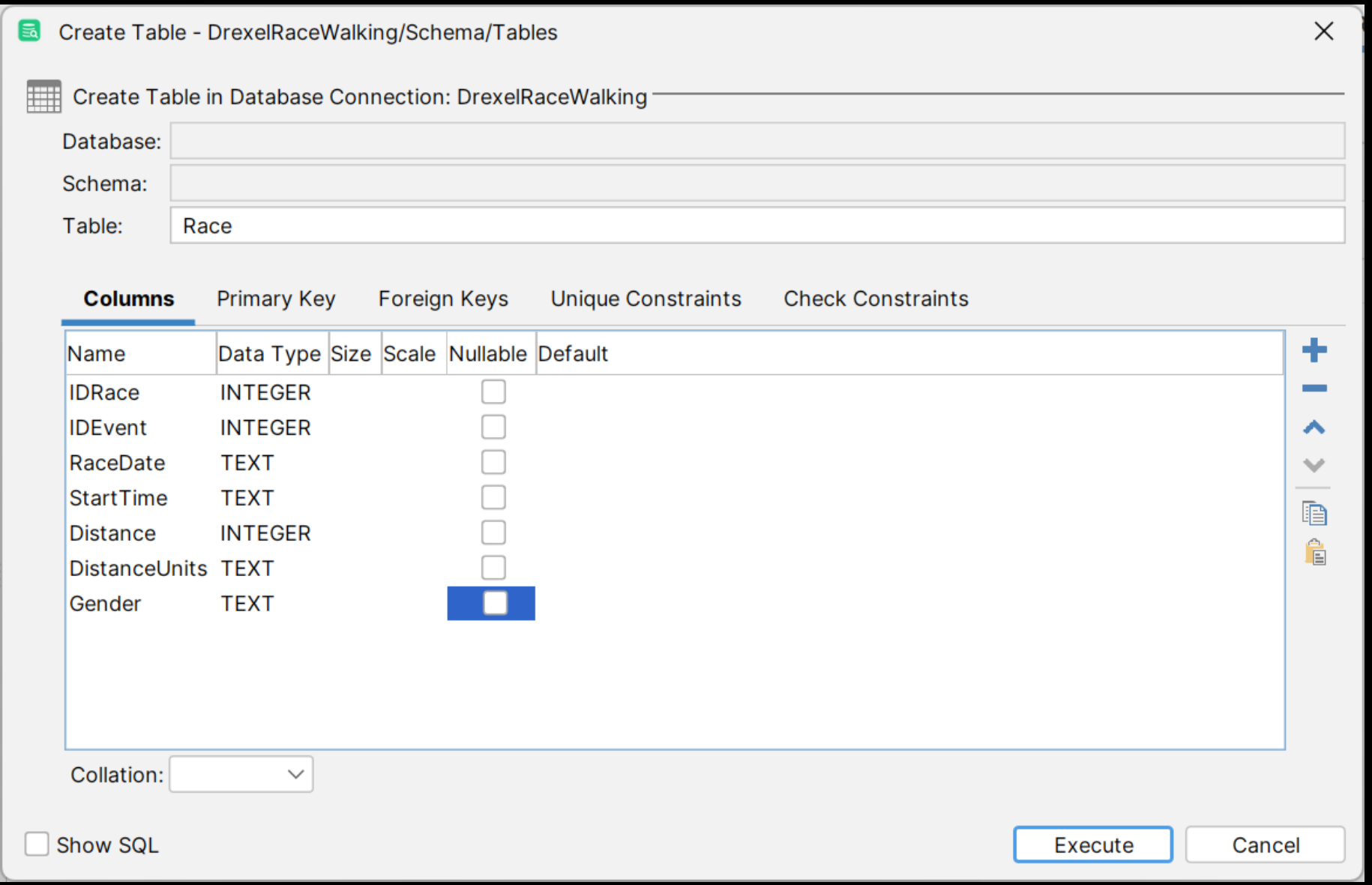
- Event
- Athlete
- Judge
- Observer

All had one thing in common:

There were no fields defined / related in other tables.
They were entity defining tables.

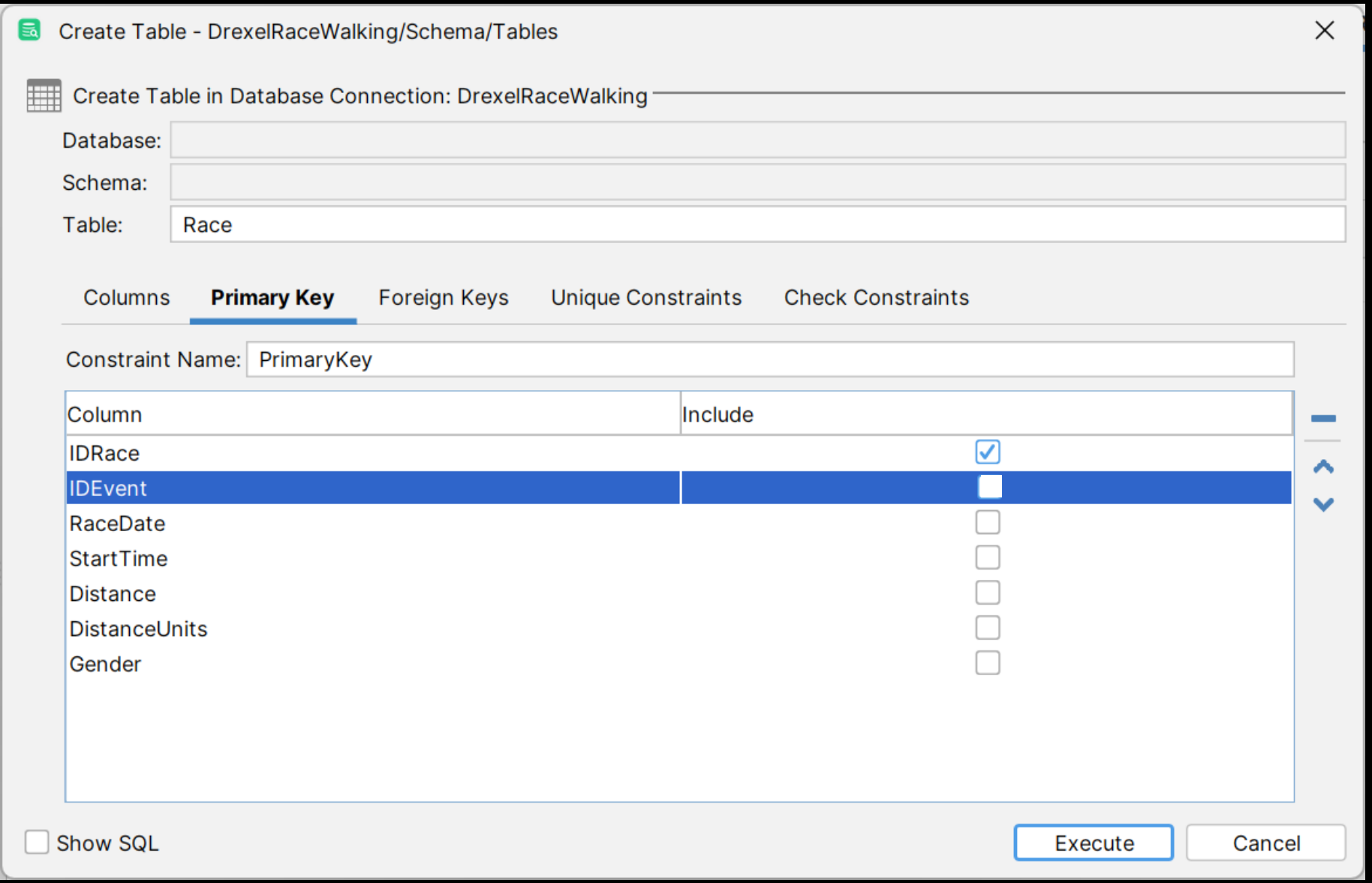
Creating Tables - GUI

The next table, *Race* is a little different. Start creating the basic fields, all not nullable.



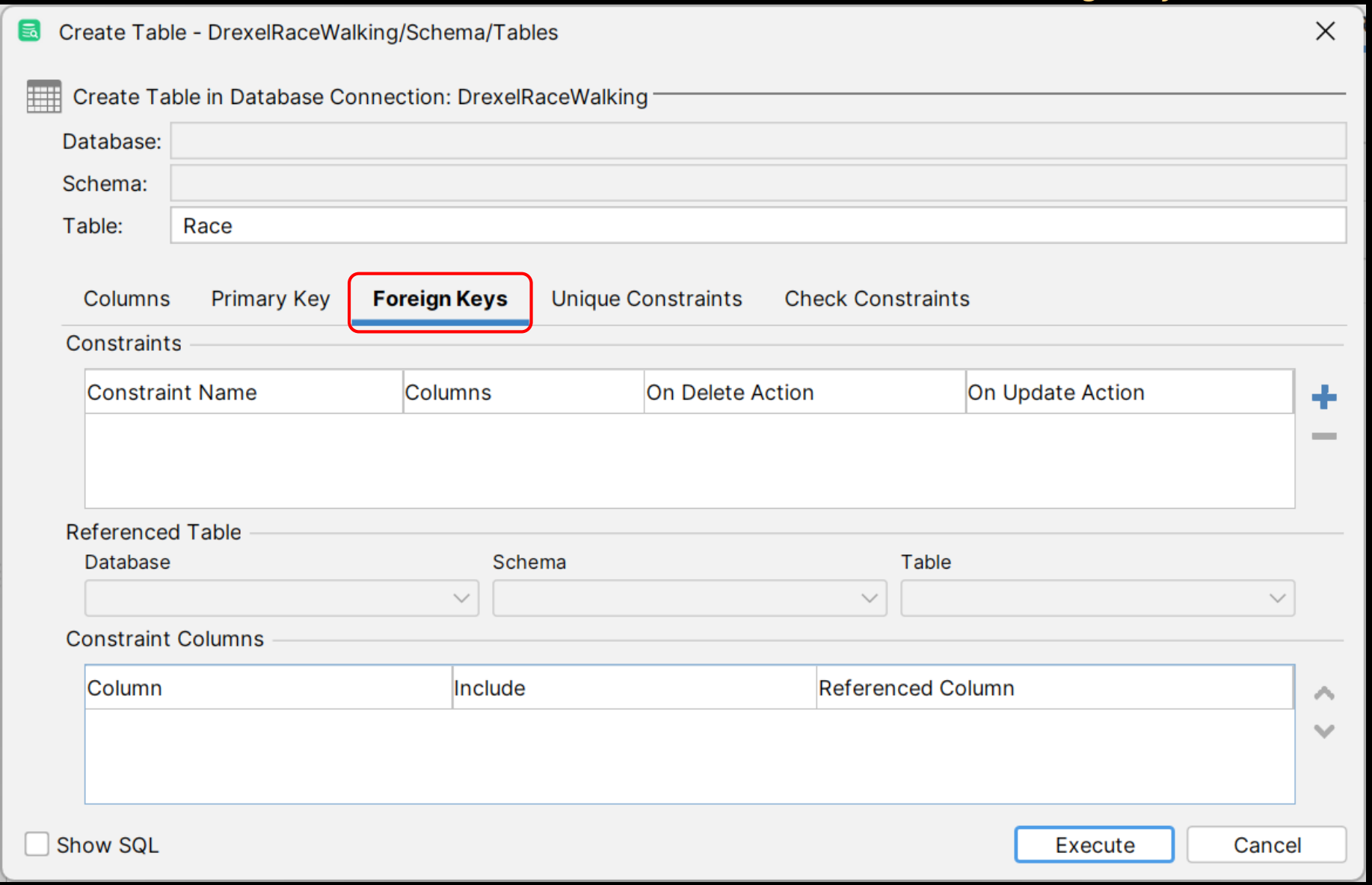
Creating Tables - GUI

The Primary Key is *IDRace*. Note the original video showed this as a compound key including IDEvent. That was a typo



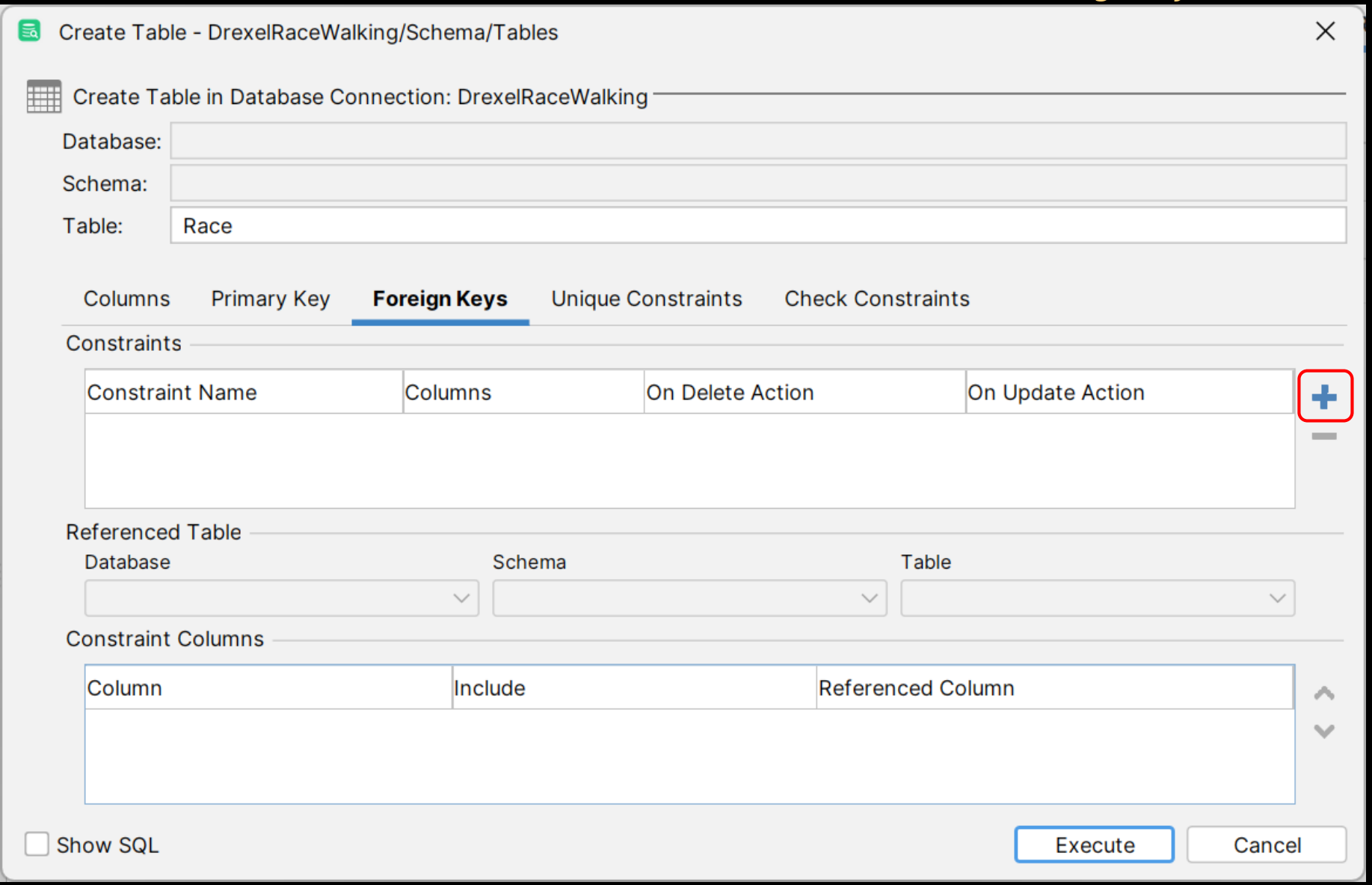
Creating Tables - GUI

We are not done. We need to ensure that *IDRace* exists. Click on the *Foreign Keys* tab:



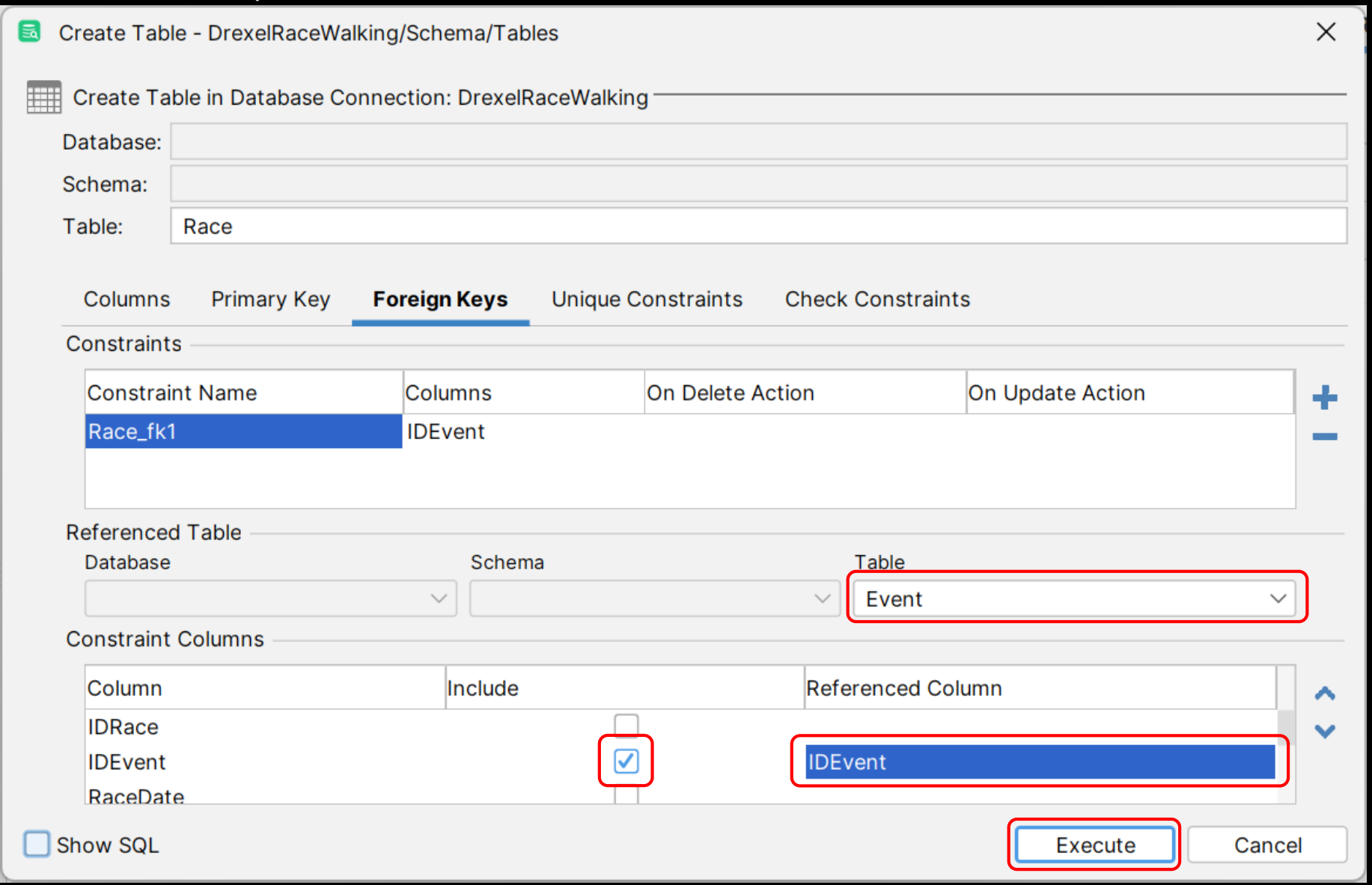
Creating Tables - GUI

We are not done. We need to ensure that *IDEvent* exists. Click on the *Foreign Keys* tab:



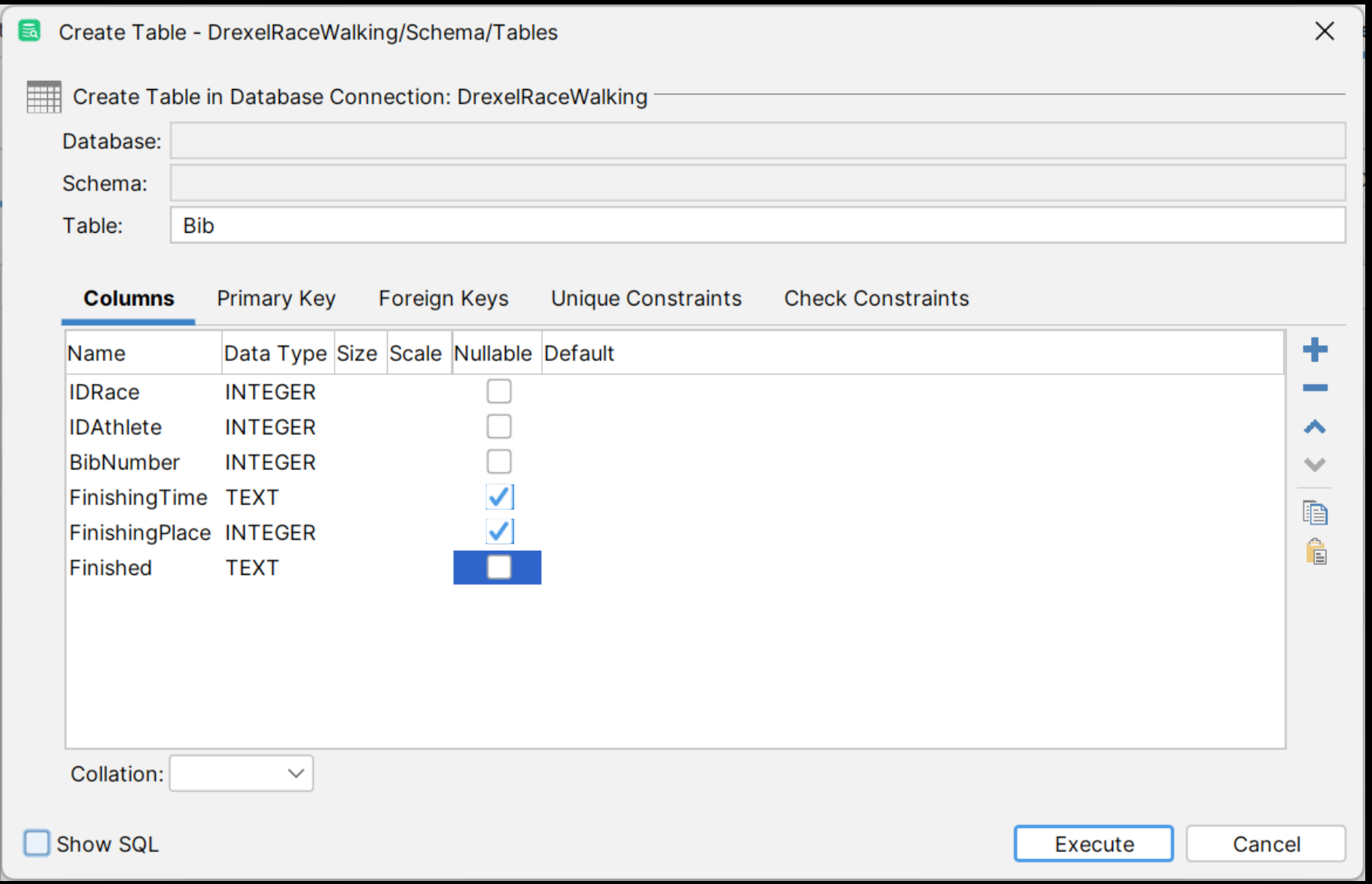
Creating Tables - GUI

Select the lookup table, Event. Include the *IDEvent* column. Select the *IDEvent* reference column. Hit the *Execute* button:



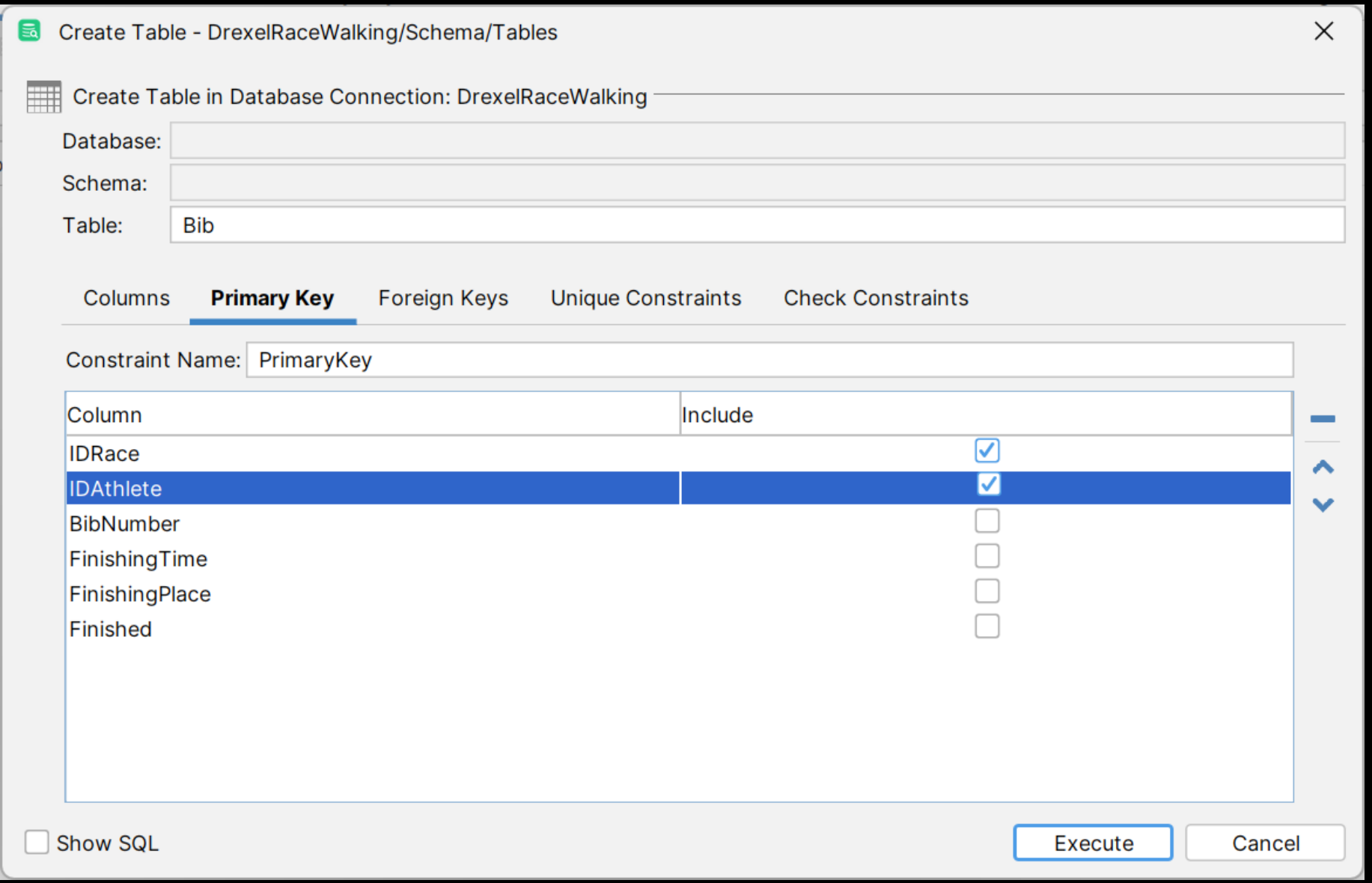
Creating Tables - GUI

Create the *Bib* table. Add the fields:



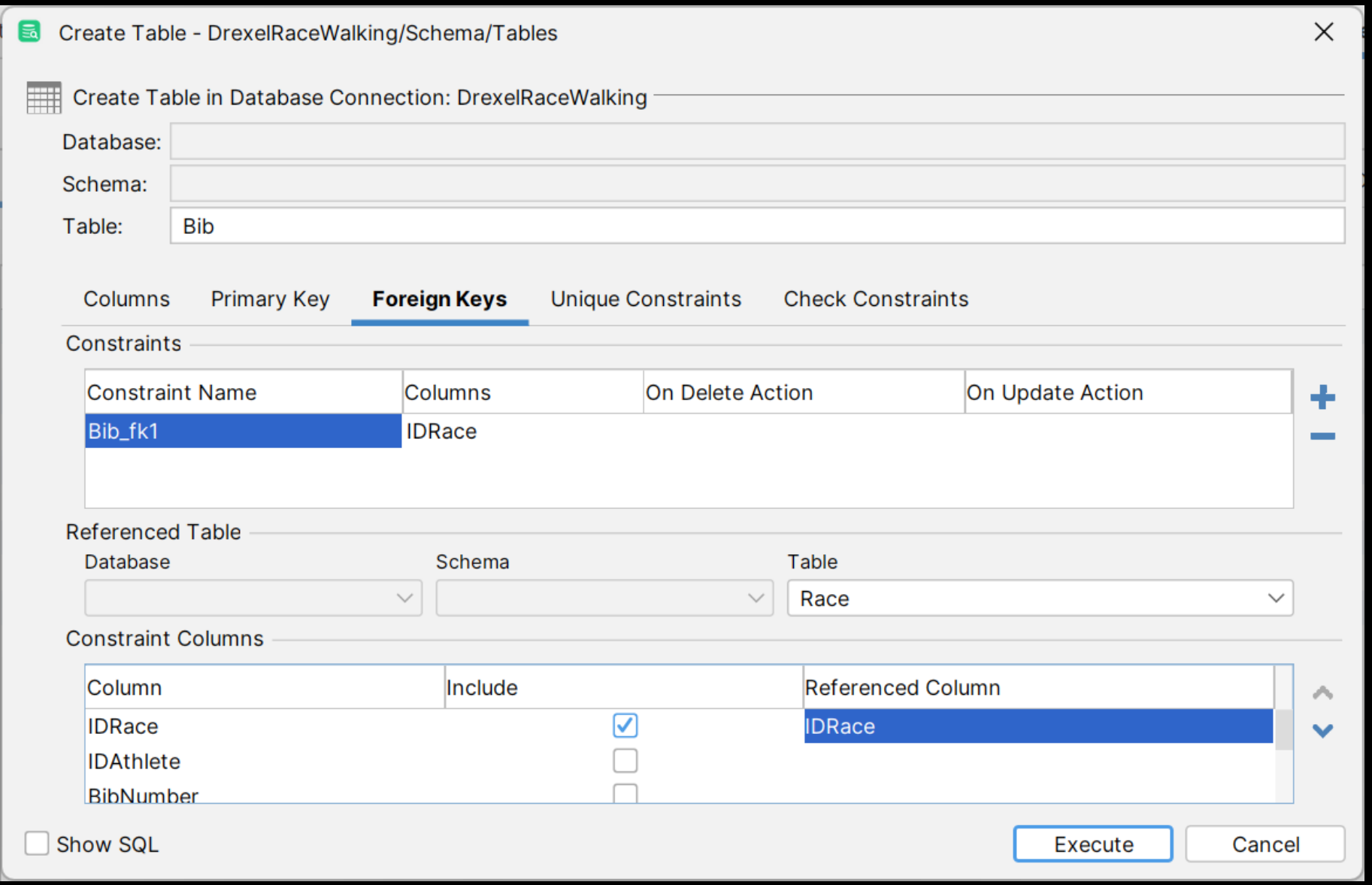
Creating Tables - GUI

Add a compound primary key with *IDRace* & *IDAthlete*:



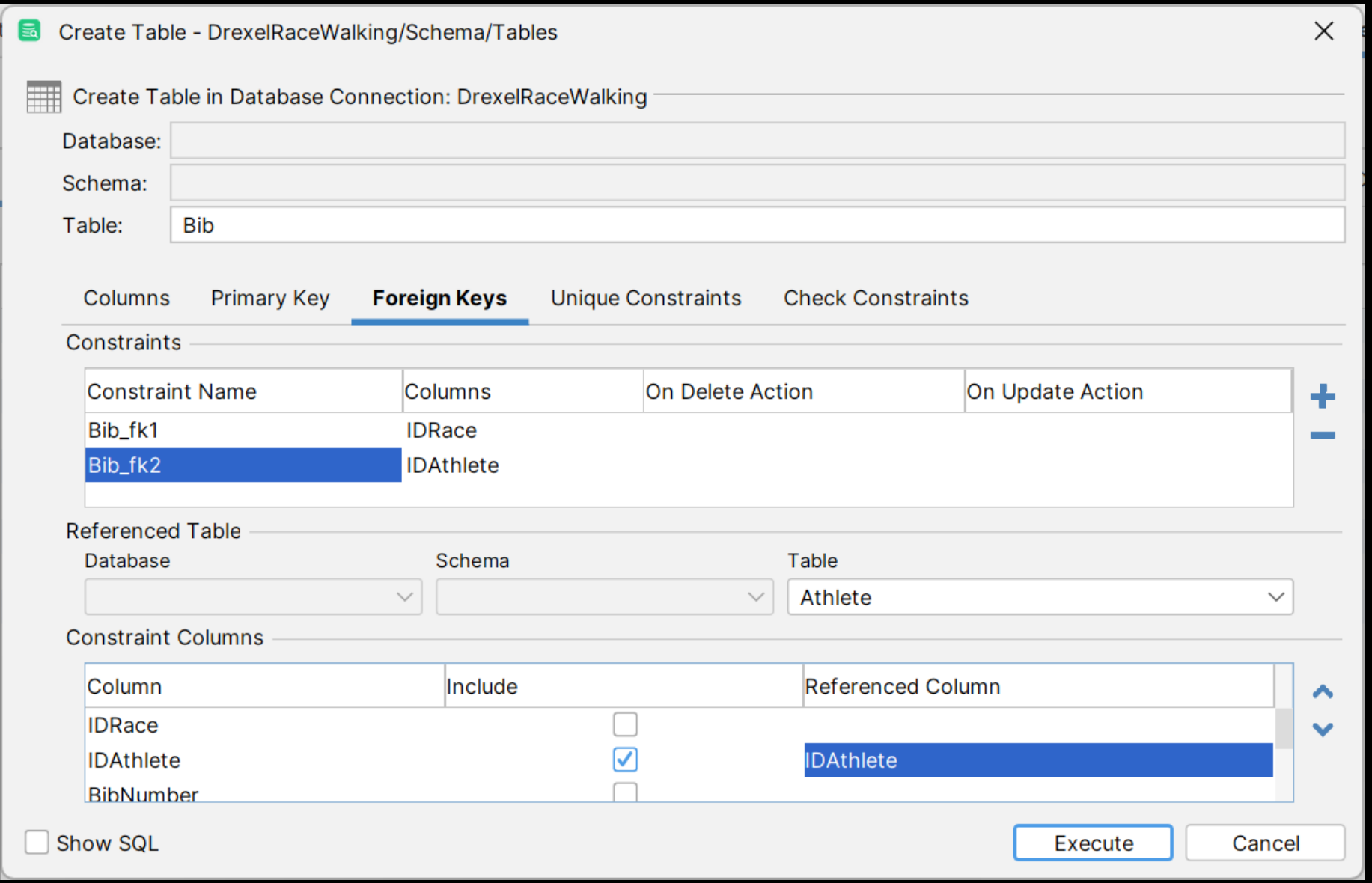
Creating Tables - GUI

Add a foreign key for *IDRace* to the *Race* table:



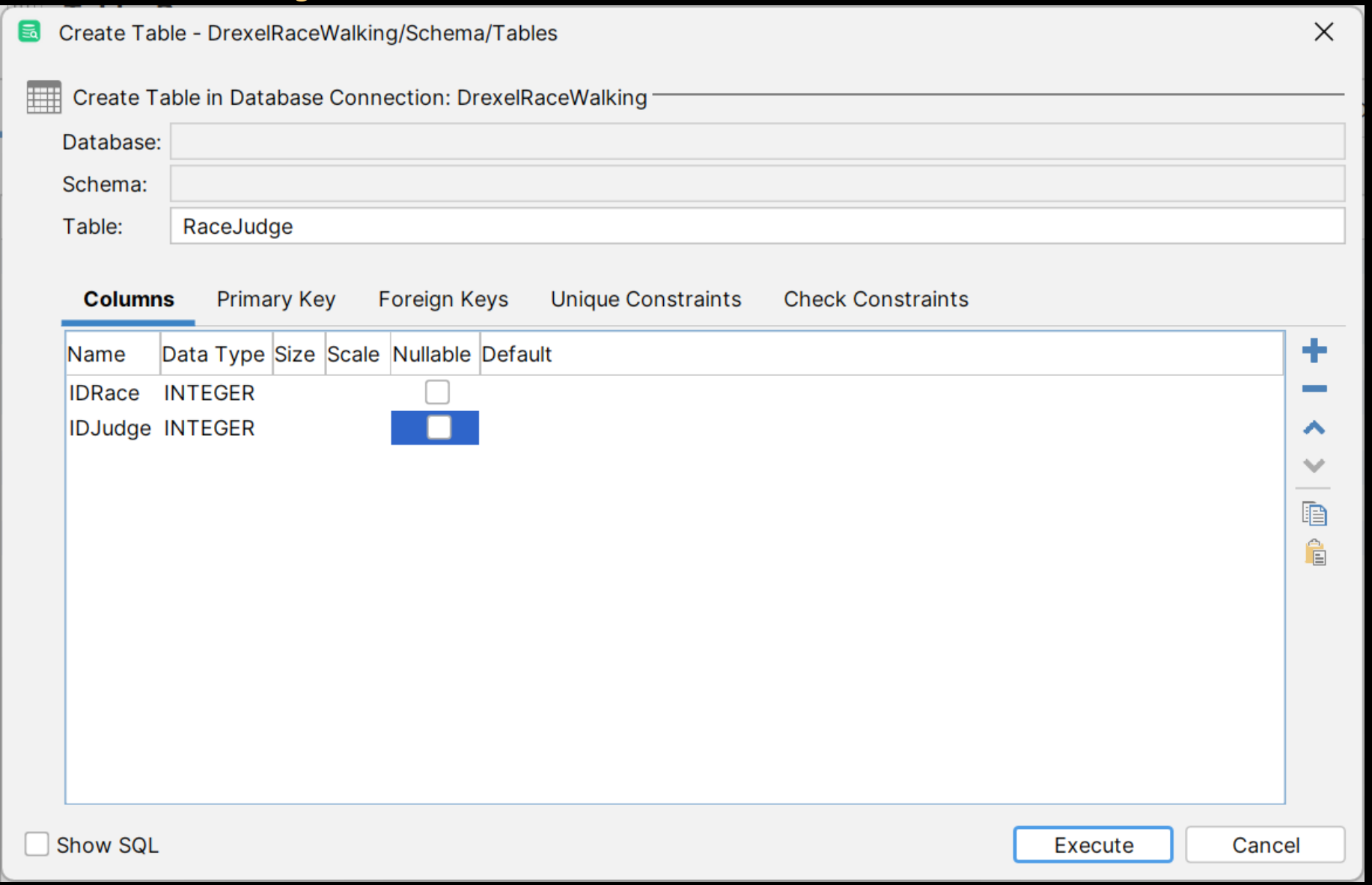
Creating Tables - GUI

Add a foreign key for *IDAthlete* to the *Athlete* table:



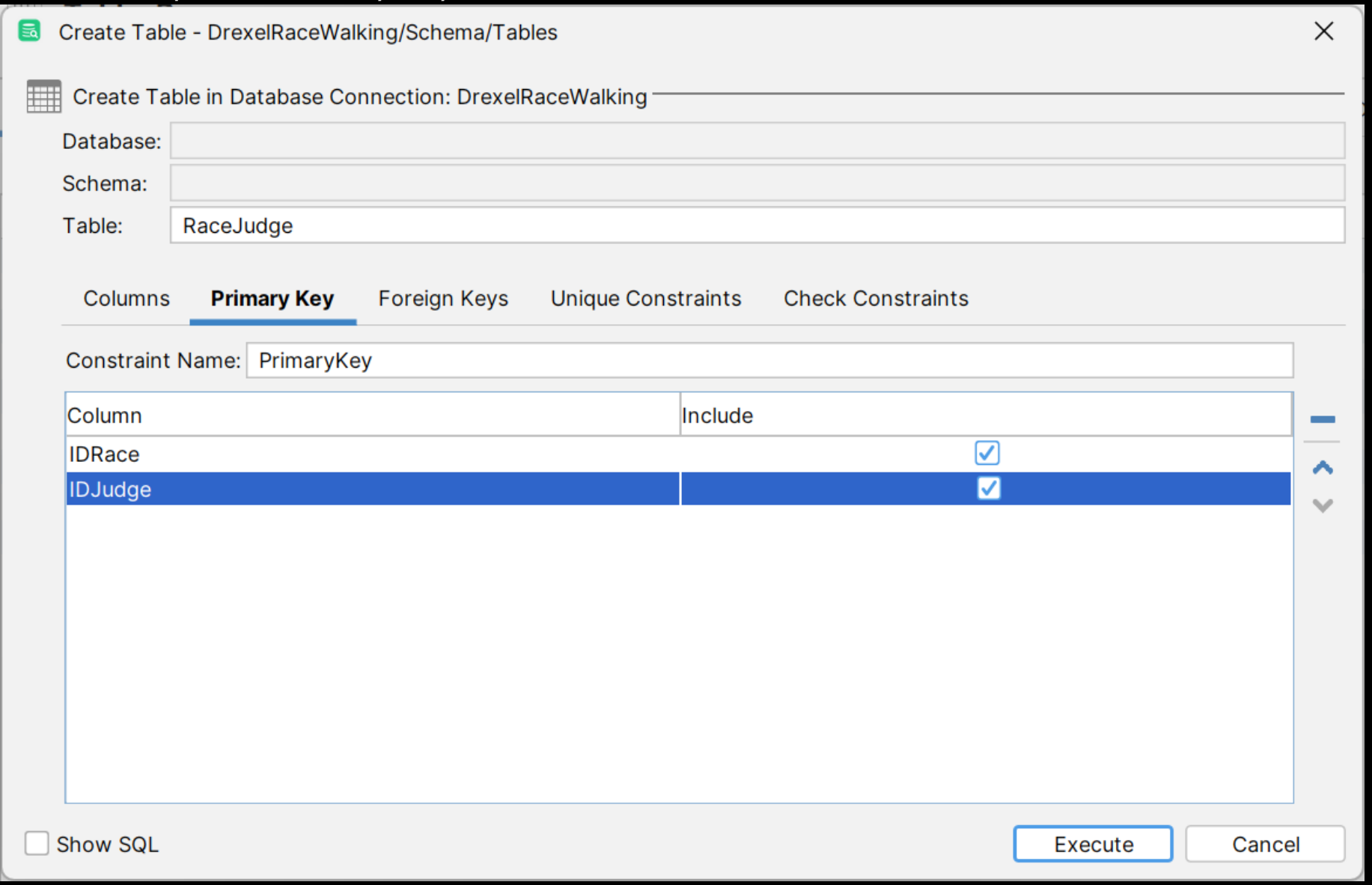
Creating Tables - GUI

Create the *RaceJudge* table:



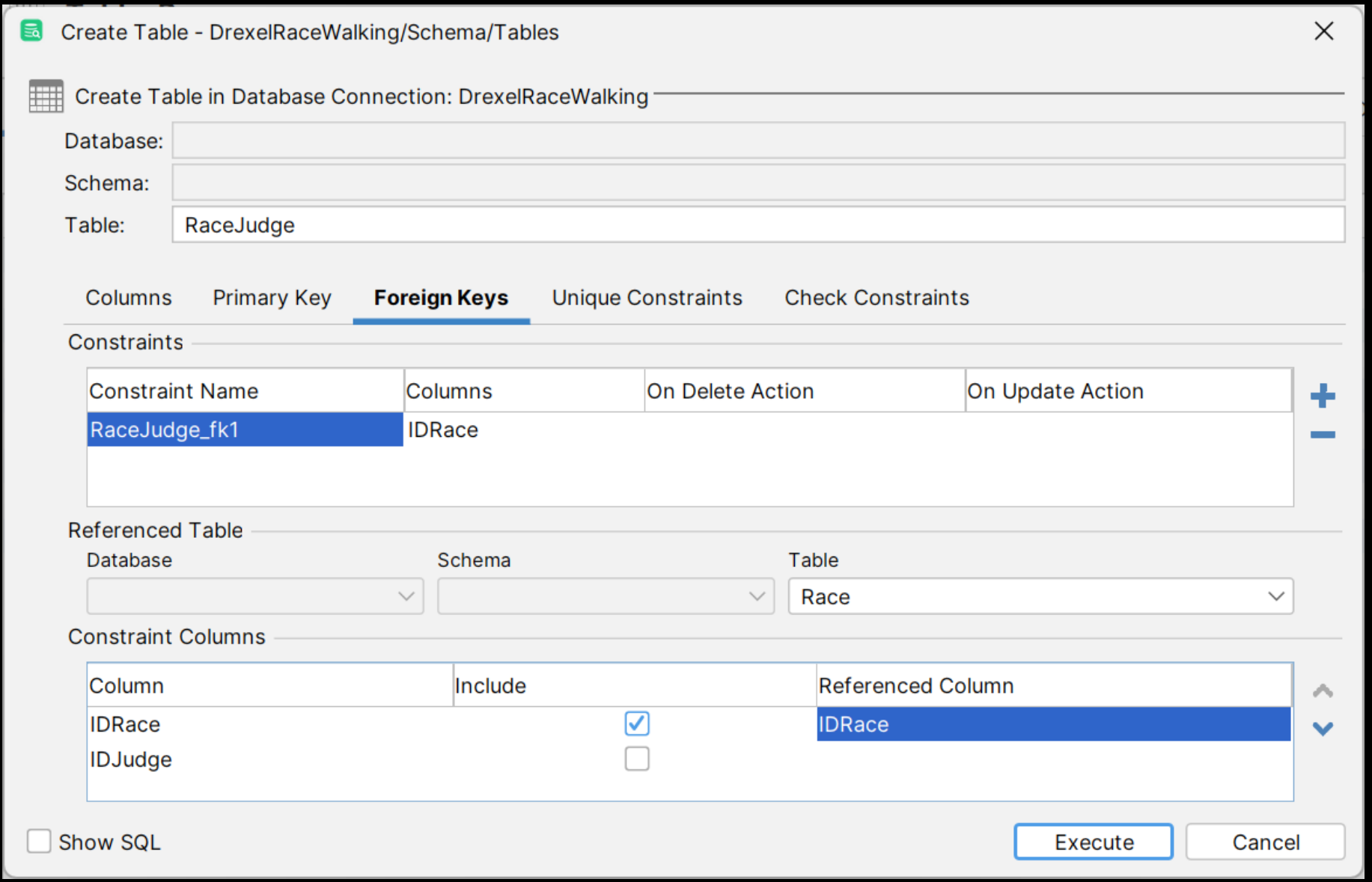
Creating Tables - GUI

Add a compound Primary Key:



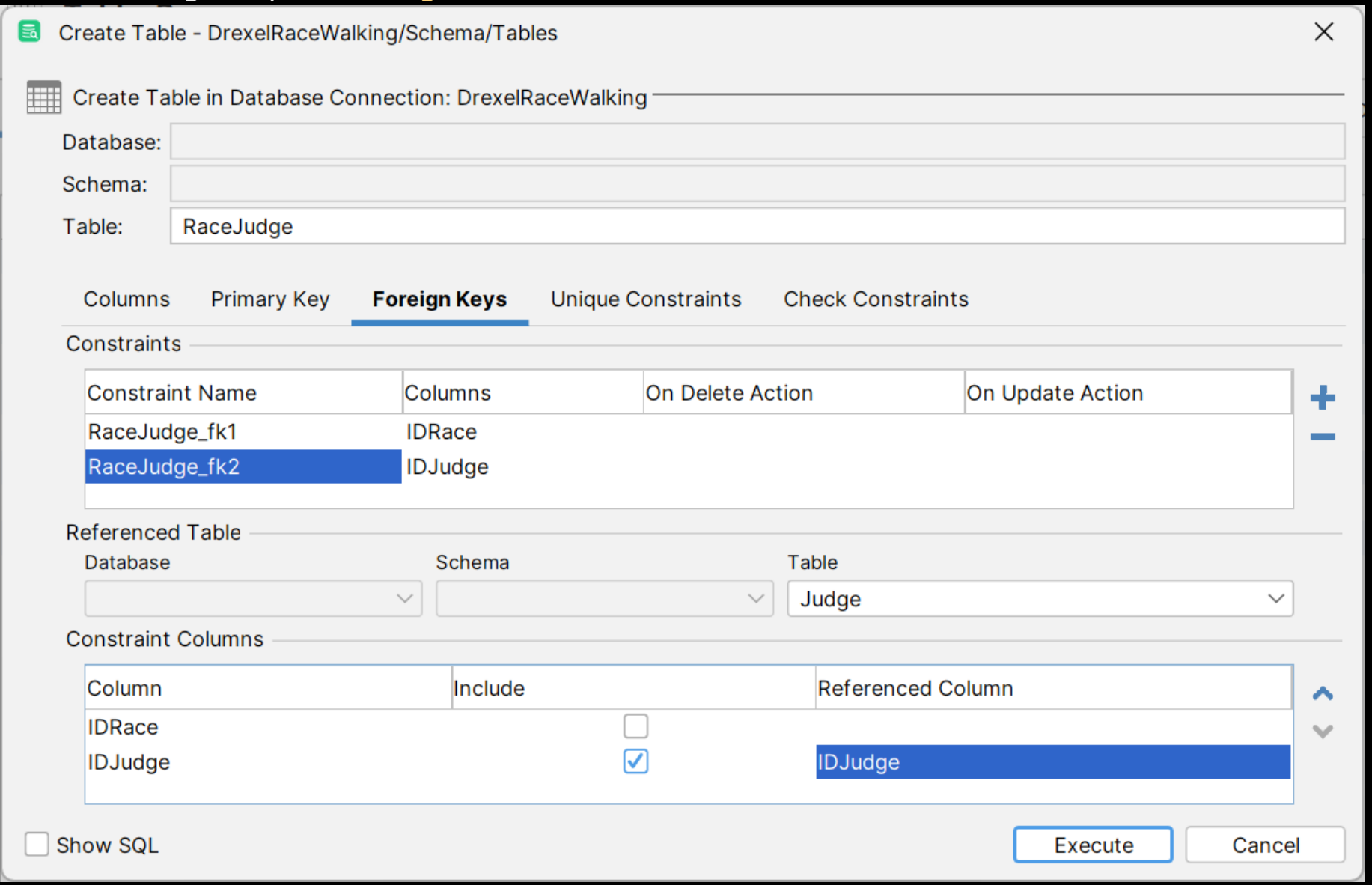
Creating Tables - GUI

Add a Foreign Key for *IDRace*:



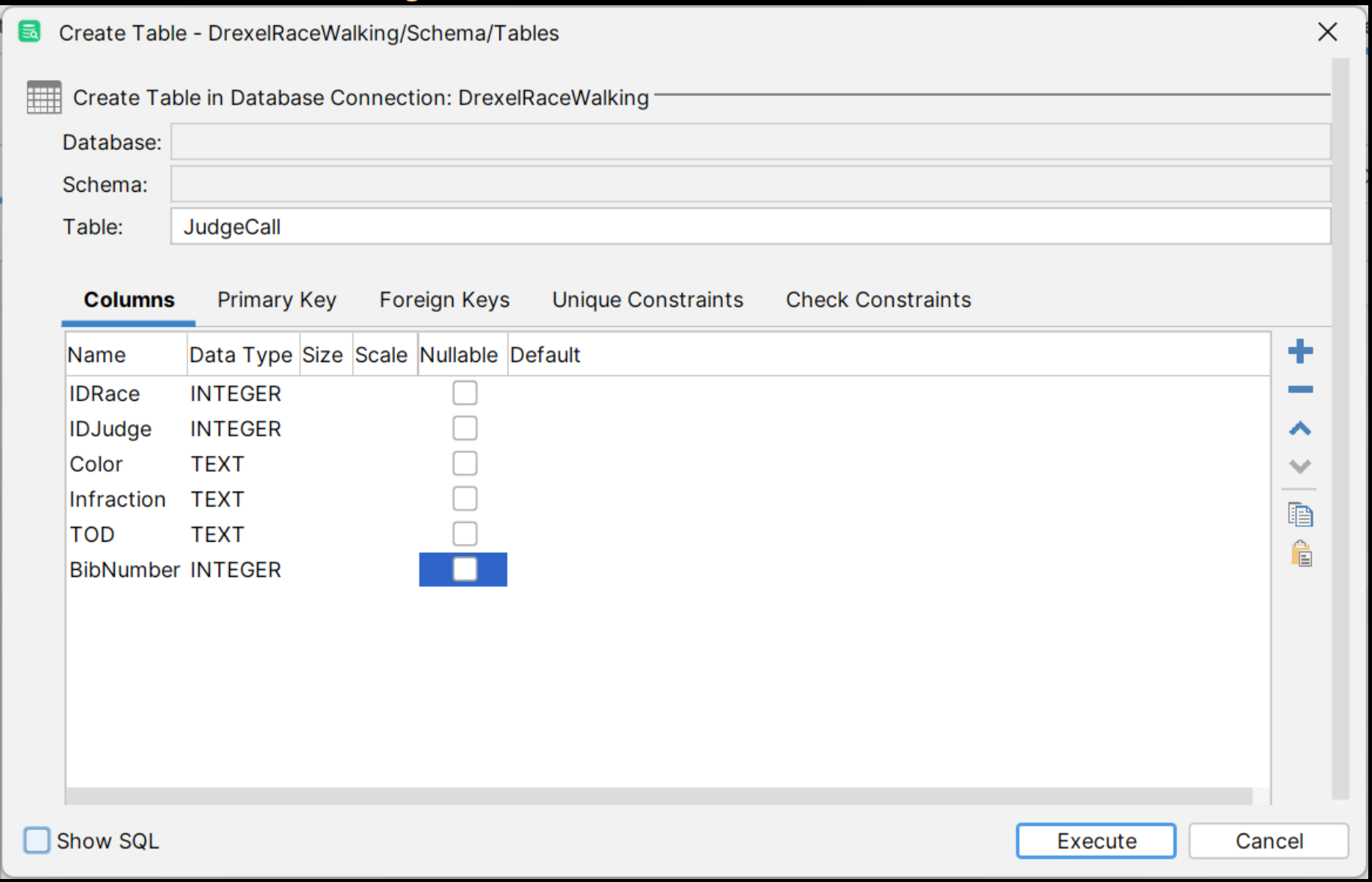
Creating Tables - GUI

Add a Foreign Key for *IDJudge*:



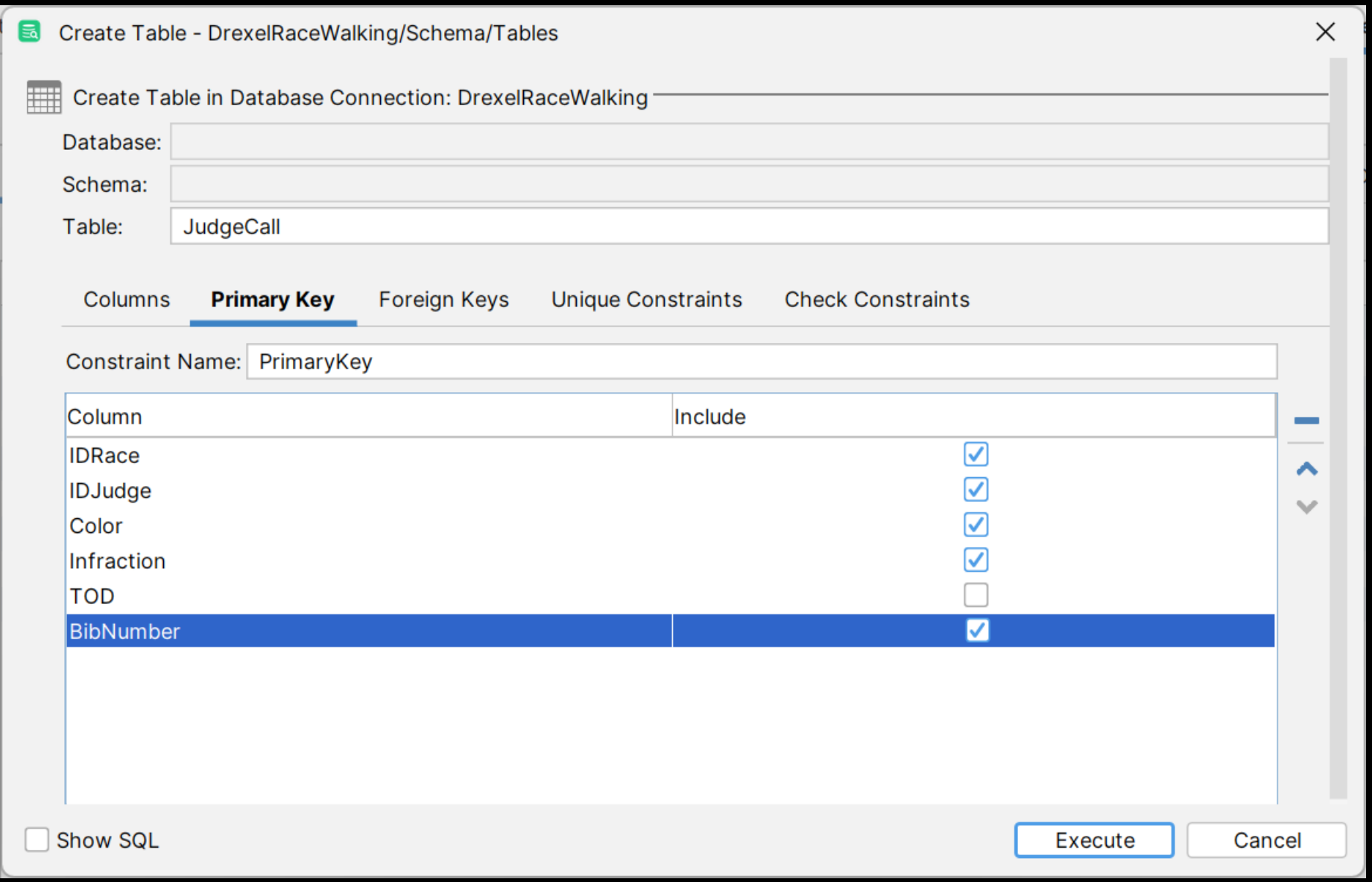
Creating Tables - GUI

Create the fields of the *JudgeCall* table:



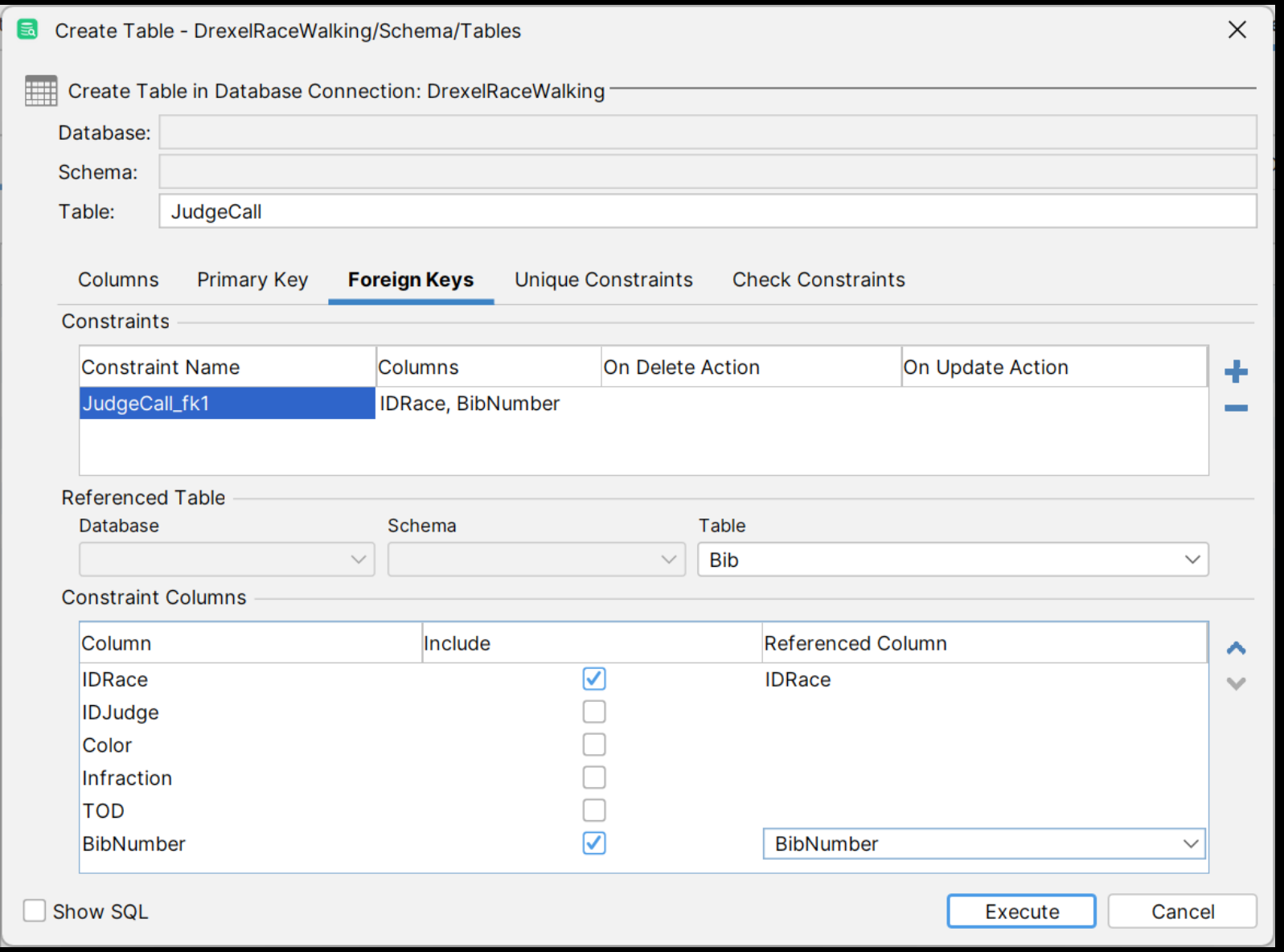
Creating Tables - GUI

Add a compound Primary Key on *IDRace*, *IDJudge*, *BibNumber*, *Color*, & *Infraction*:



Creating Tables - GUI

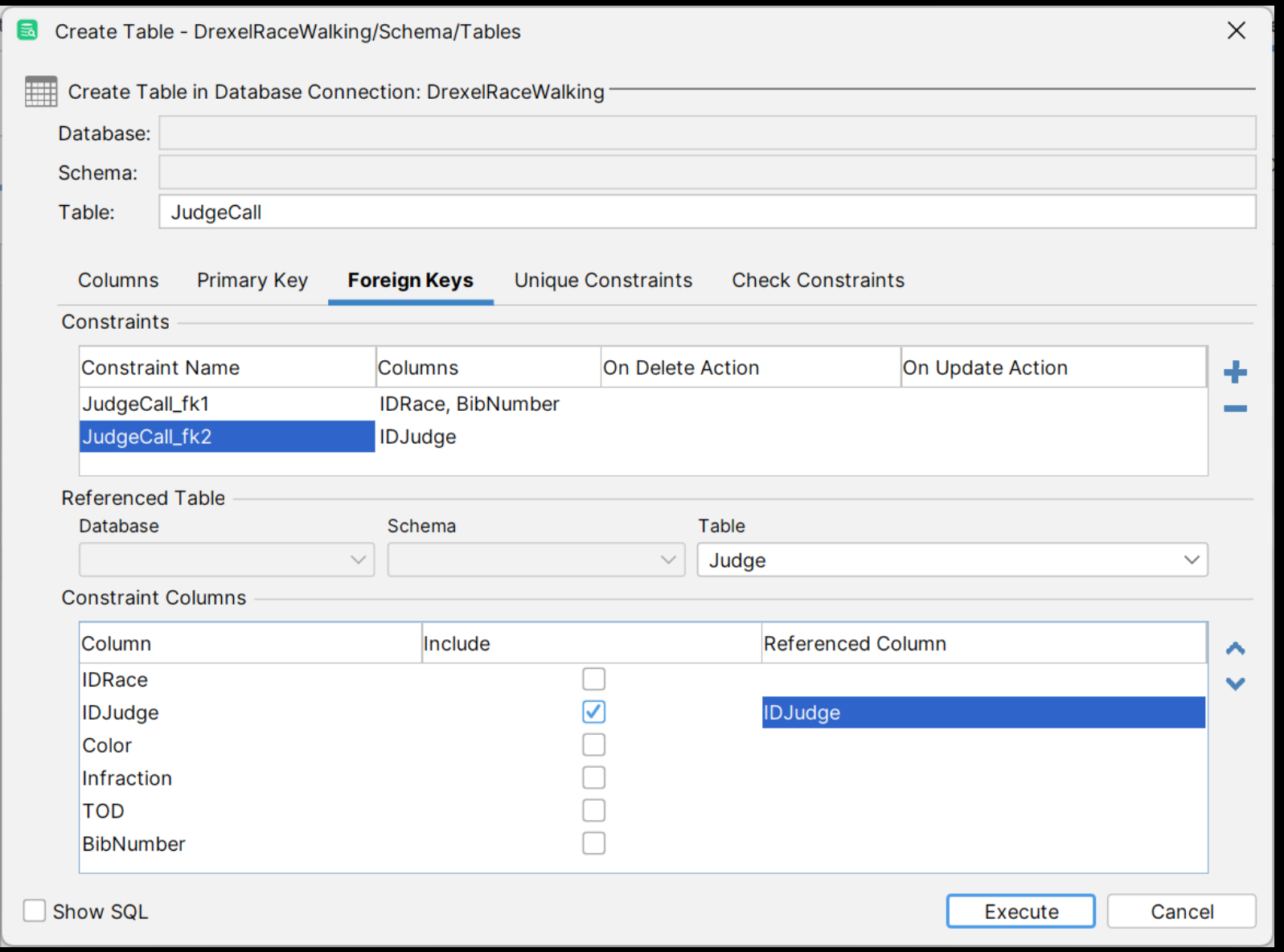
Create a Foreign Key for *IDRace* & *BibNumber* from the *BibNumber* table



We can't simply check if the bib number exists, because bib numbers are reused from race to race.

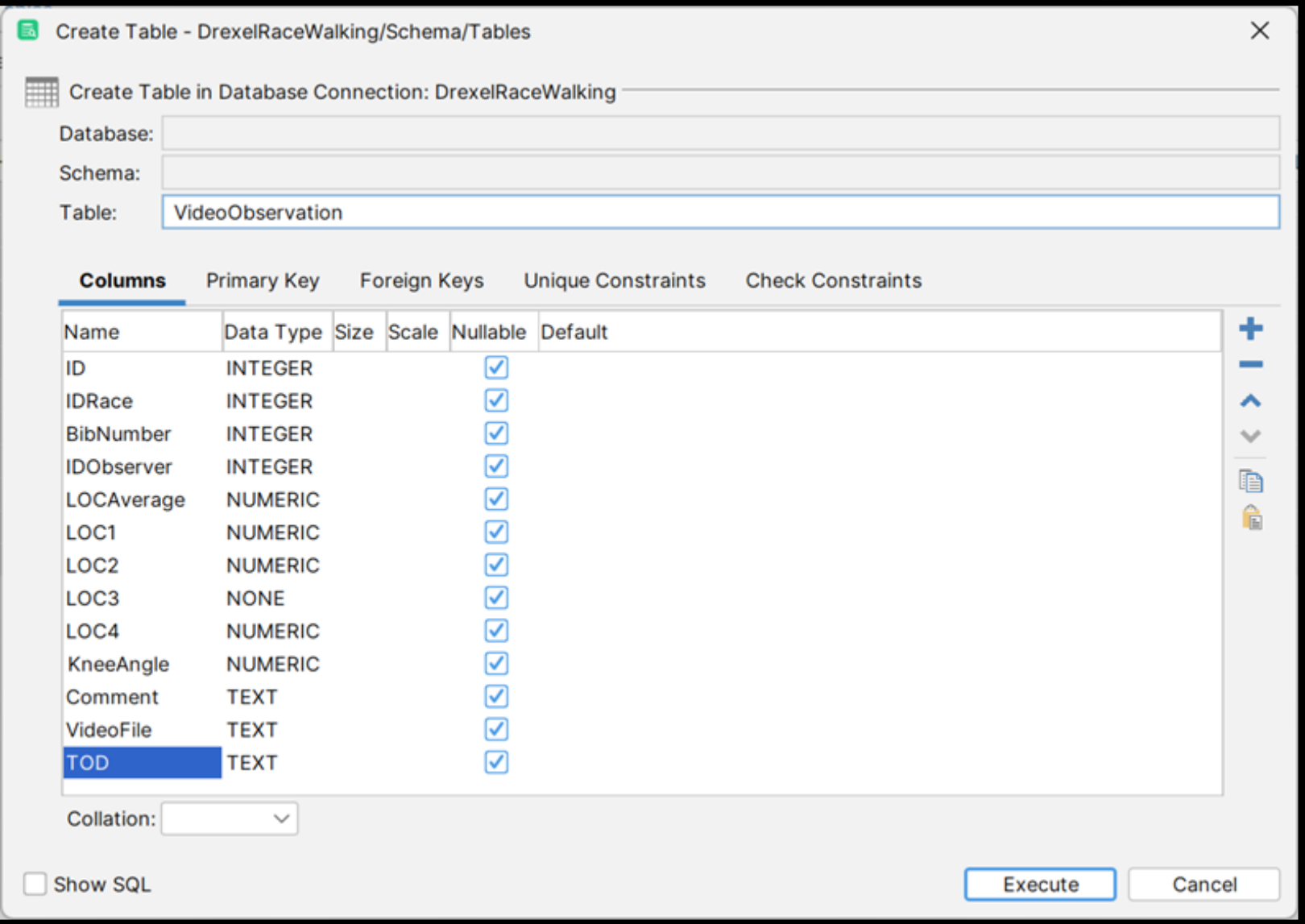
Creating Tables - GUI

Create a Foreign Key for *IDJudge*



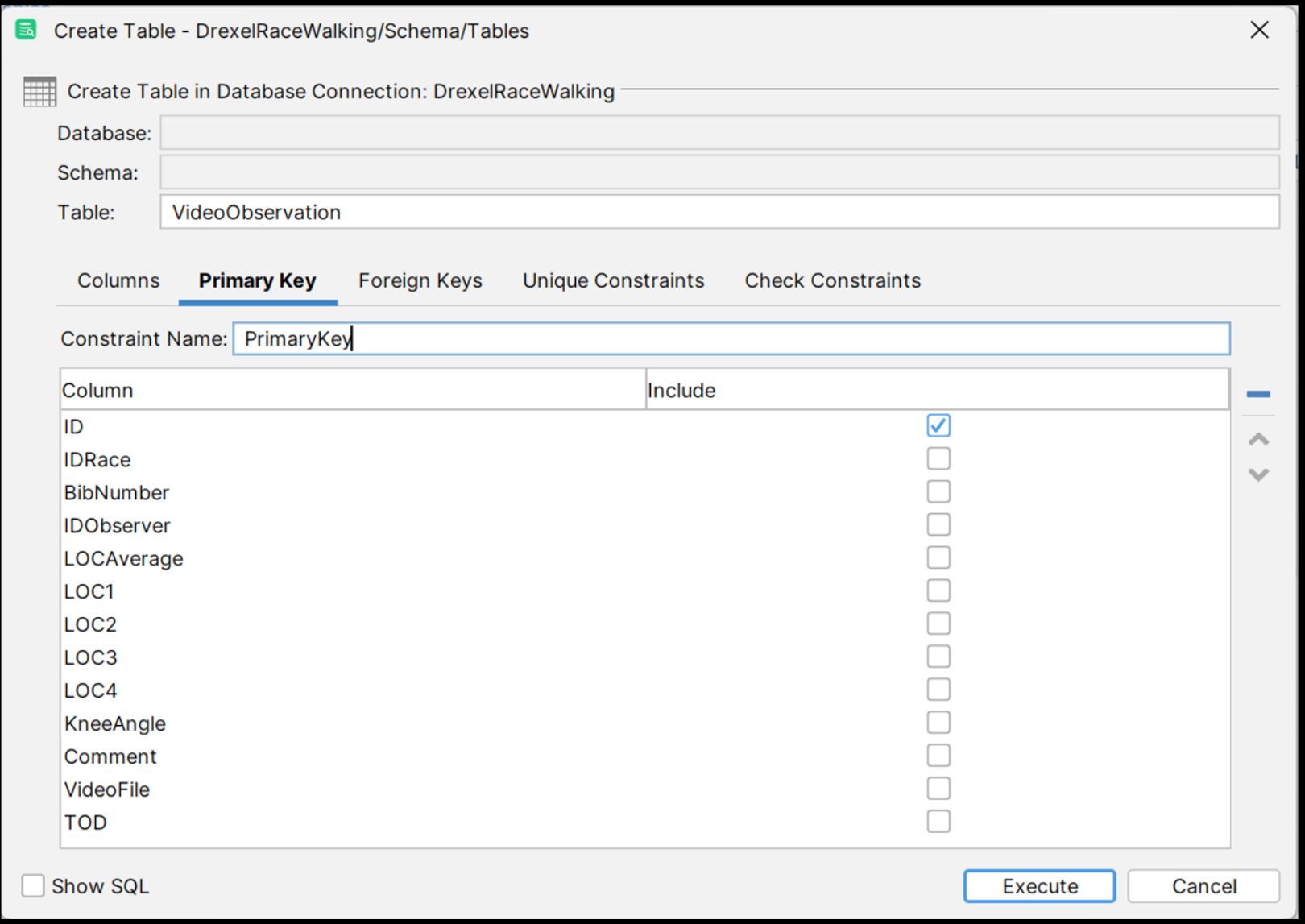
Creating Tables - GUI

Create the rows for the *VideoObservation* table:



Creating Tables - GUI

Create the Primary Key for the *VideoObservation* table:



The choice of Primary Key could be debated.

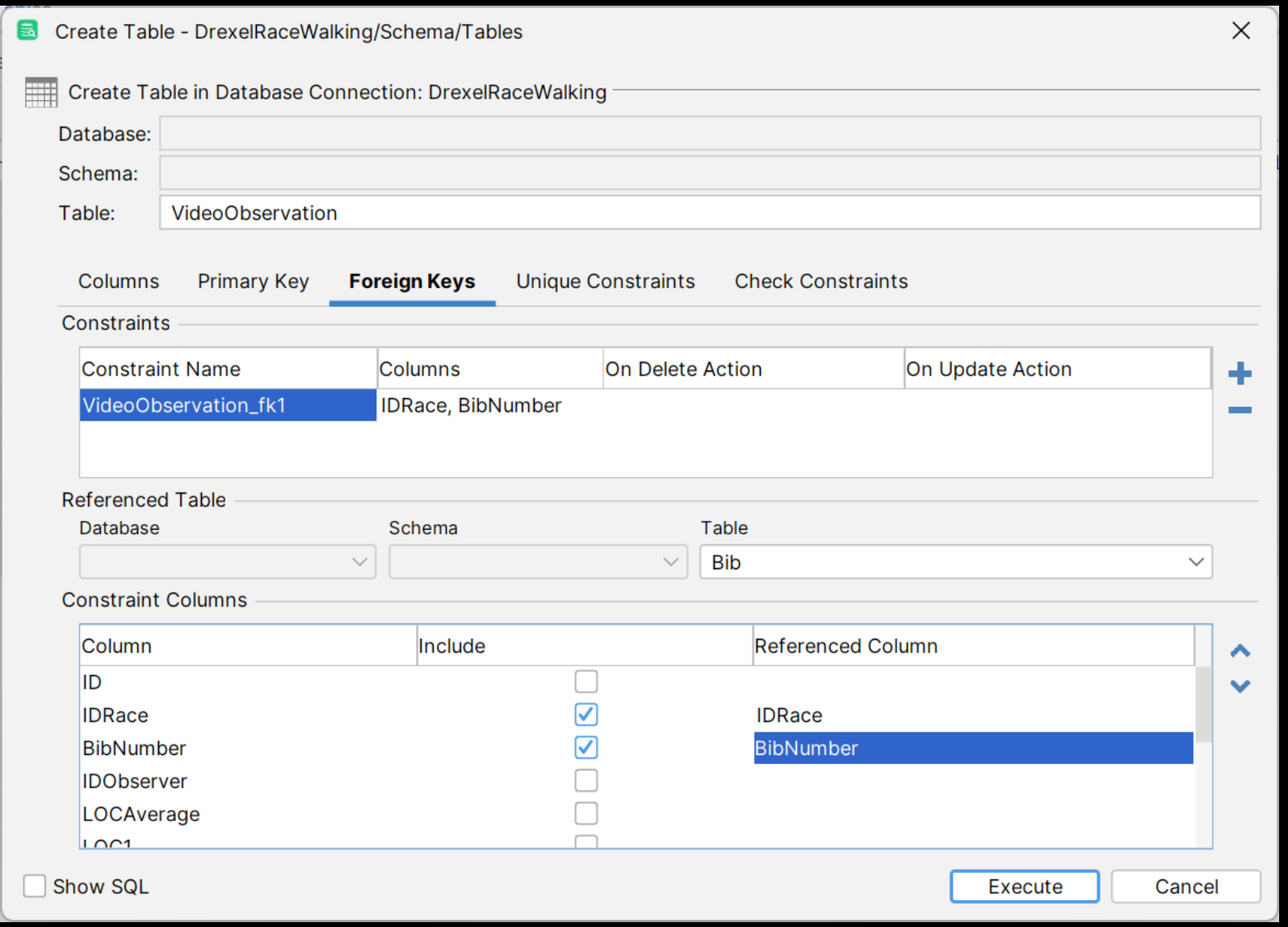
Do we need an *ID* field?

We could say that *IDRace*, *BibNumber*, *VideoFile*, & *TOD* make a unique record.

Sometimes the choice is style.

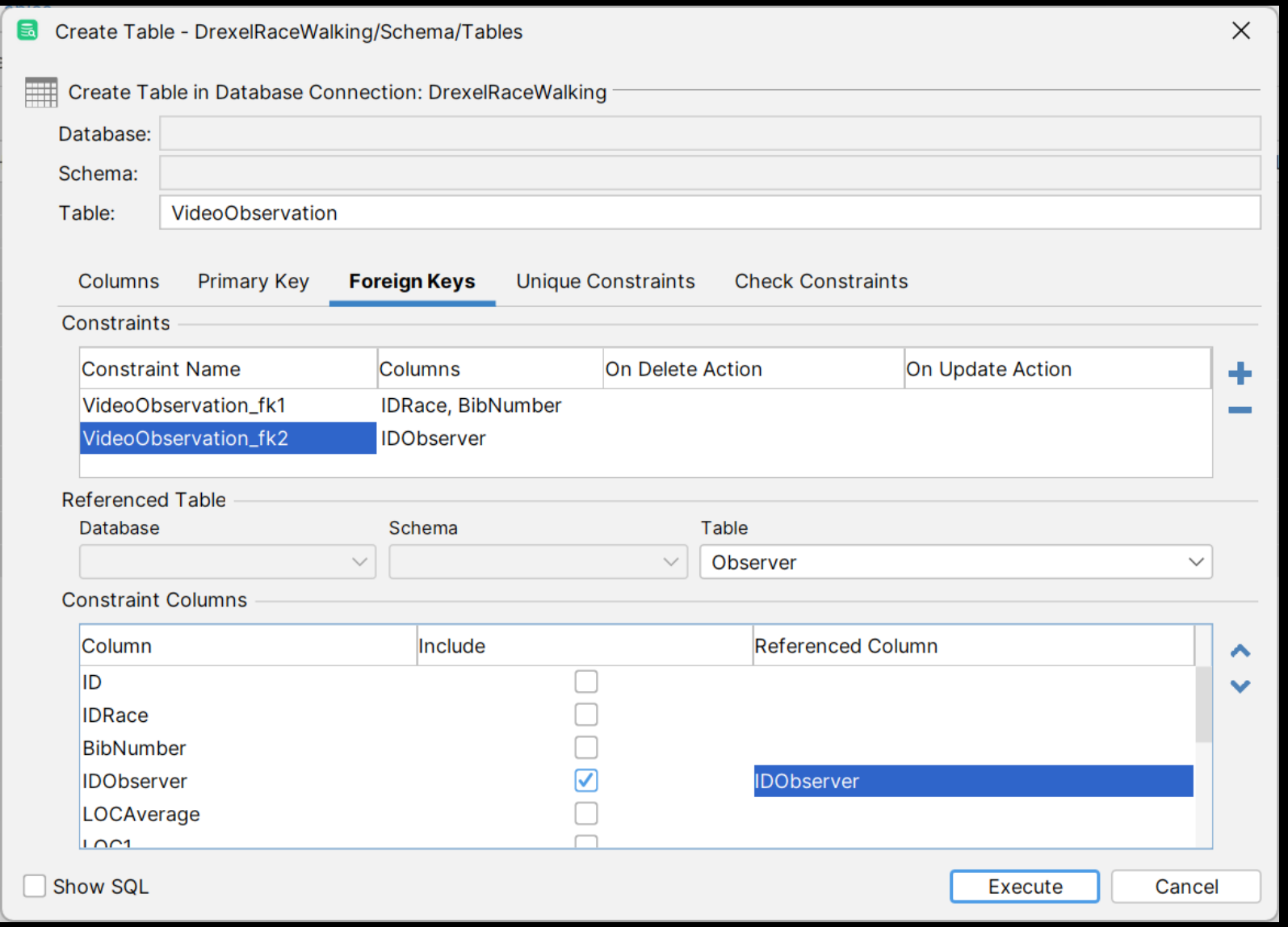
Creating Tables - GUI

Create a Foreign Key for the *IDRace* and *BibNumber* fields to the *Bib* table:



Creating Tables - GUI

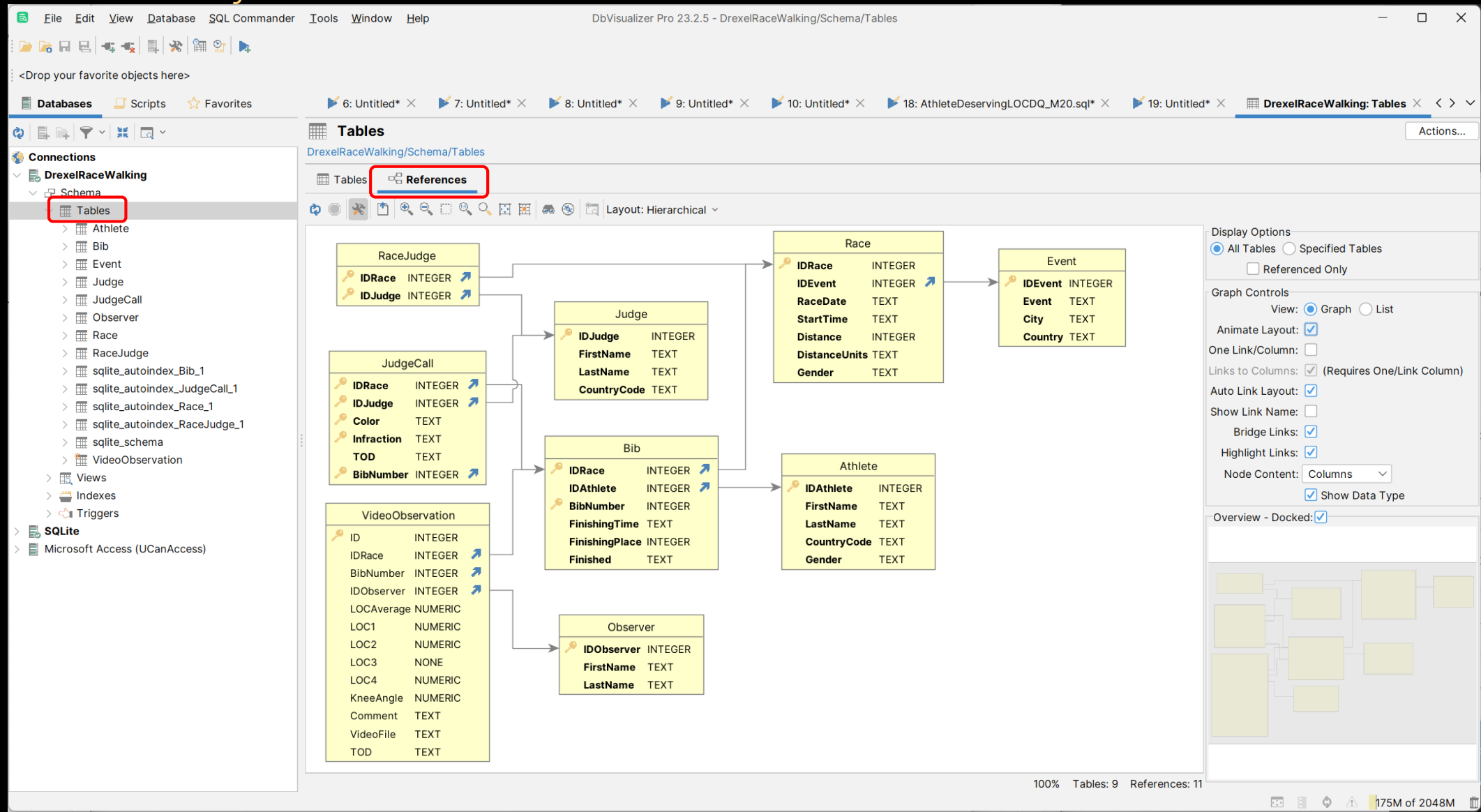
Create a Foreign Key for the *IDObserver* field to the *Observer* table:



Visualizing the Database

Let's looks at what we created.

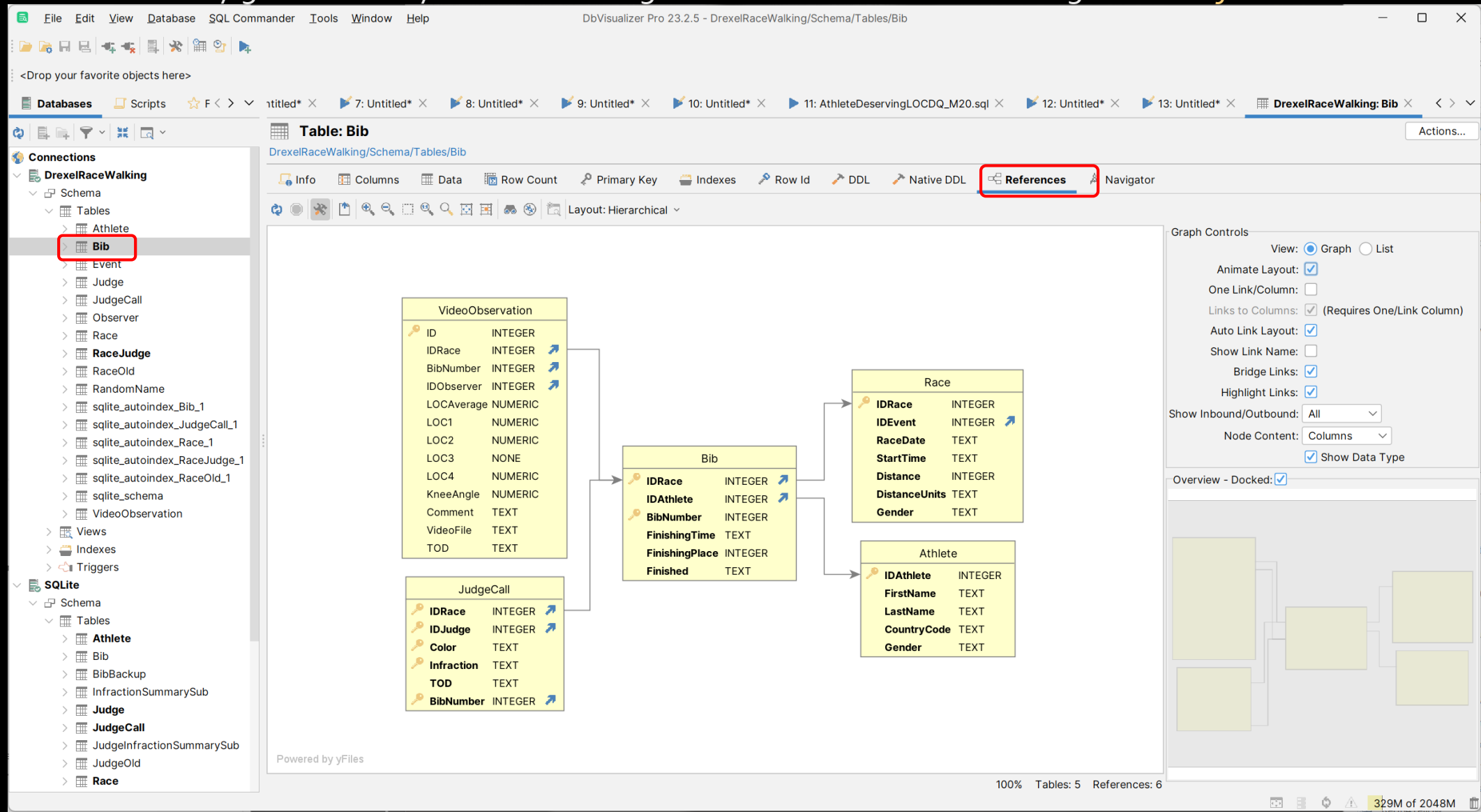
Double-click on *Tables* then click on *References*:



Visualizing the Database

Let's looks at what we created.

We can also see the references for any given table by Double-clicking on the table name and clicking on the *References* tab:



Creating Tables - Code

Create the *Event* table in code:

```
CREATE TABLE
    Event
    (
        IDEvent INTEGER NOT NULL,
        Event   TEXT NOT NULL,
        City    TEXT NOT NULL,
        Country TEXT NOT NULL,
        CONSTRAINT PrimaryKey PRIMARY KEY (IDEvent)
    )
```

Creating Tables - Code

Create the *Athlete* table in code:

```
CREATE TABLE
    AthleteTest
    (
        IDAthlete   INTEGER NOT NULL,
        FirstName   TEXT NOT NULL,
        LastName    TEXT NOT NULL,
        CountryCode TEXT NOT NULL,
        Gender  TEXT NOT NULL,
        CONSTRAINT PrimaryKey PRIMARY KEY (IDAthlete)
    )
```

Creating Tables - Code

Create the *Judge* table in code:

```
CREATE TABLE
    Judge
    (
        IDJudge     INTEGER NOT NULL,
        FirstName   TEXT NOT NULL,
        LastName    TEXT NOT NULL,
        CountryCode TEXT NOT NULL,
        CONSTRAINT PrimaryKey PRIMARY KEY (IDJudge)
    )
```

Creating Tables - Code

Create the *Race* table in code:

```
CREATE TABLE
    Race
    (
        IDRace        INTEGER NOT NULL,
        IDEvent       INTEGER NOT NULL,
        RaceDate      TEXT NOT NULL,
        StartTime     TEXT NOT NULL,
        Distance      INTEGER NOT NULL,
        DistanceUnits TEXT NOT NULL,
        Gender        TEXT NOT NULL,
        CONSTRAINT PrimaryKey PRIMARY KEY (IDRace),
        CONSTRAINT Race_fk1 FOREIGN KEY (IDEvent) REFERENCES "Event" ("IDEvent")
    )
```

Creating Tables - Code

Create the *Bib* table in code:

```
CREATE TABLE
    Bib
    (
        IDRace          INTEGER NOT NULL,
        IDAthlete       INTEGER NOT NULL,
        BibNumber       INTEGER NOT NULL,
        FinishingTime   TEXT,
        FinishingPlace  INTEGER,
        Finished        TEXT NOT NULL,
        CONSTRAINT PrimaryKey PRIMARY KEY (IDRace, BibNumber),
        CONSTRAINT Bib_fk1 FOREIGN KEY (IDRace) REFERENCES "Race" ("IDRace"),
        CONSTRAINT Bib_fk2 FOREIGN KEY (IDAthlete) REFERENCES "Athlete" ("IDAthlete")
    )
```

PLEASE NOTE, Prof. Salvage updated this slide to reflect `FinishingTime` & `FinishingPlace` allowing `NULL`.

Creating Tables - Code

Create the *RaceJudge* table in code:

```
CREATE TABLE
    RaceJudge
    (
        IDRace  INTEGER NOT NULL,
        IDJudge INTEGER NOT NULL,
        CONSTRAINT PrimaryKey PRIMARY KEY (IDRace, IDJudge),
        CONSTRAINT RaceJudge_fk1 FOREIGN KEY (IDRace) REFERENCES "Race" ("IDRace"),
        CONSTRAINT RaceJudge_fk2 FOREIGN KEY (IDJudge) REFERENCES "Judge" ("IDJudge")
    )
```

Creating Tables - Code

Create the *JudgeCall* table in code:

```
CREATE TABLE
    JudgeCall
    (
        IDRace     INTEGER NOT NULL,
        IDJudge    INTEGER NOT NULL,
        Color      TEXT NOT NULL,
        Infraction TEXT NOT NULL,
        TOD        TEXT NOT NULL,
        BibNumber  INTEGER NOT NULL,
        CONSTRAINT PrimaryKey PRIMARY KEY (IDRace, IDJudge, Color, Infraction, BibNumber),
        CONSTRAINT JudgeCall_fk1 FOREIGN KEY (IDRace, BibNumber) REFERENCES "Bib" ("IDRace",
        "BibNumber"),
        CONSTRAINT JudgeCall_fk2 FOREIGN KEY (IDJudge) REFERENCES "Judge" ("IDJudge")
    )
```

Creating Tables - Code

Create the *Observer* table in code:

```
CREATE TABLE
    Observer
    (
        IDObserver INTEGER,
        FirstName  INTEGER,
        LastName   INTEGER,
        CONSTRAINT PrimaryKey PRIMARY KEY (IDObserver)
    )
```

Creating Tables - Code

Create the *VideoObservation* table in code:

```
CREATE TABLE
    VideoObservation
    (
        ID          INTEGER,
        IDRace      INTEGER,
        BibNumber   INTEGER,
        IDObserver  INTEGER,
        LOCAverage  NUMERIC,
        LOC1        NUMERIC,
        LOC2        NUMERIC,
        LOC3        NUMERIC,
        LOC4        NUMERIC,
        KneeAngle   NUMERIC,
        Comment     TEXT,
        VideoFile   TEXT,
        TOD         TEXT,
        CONSTRAINT PrimaryKey PRIMARY KEY (ID),
        CONSTRAINT VideoObservation_fk1 FOREIGN KEY (IDRace, BibNumber) REFERENCES "Bib" ("IDRace"
        , "BibNumber"),
        CONSTRAINT VideoObservation_fk2 FOREIGN KEY (IDObserver) REFERENCES "Observer"
        ("IDObserver")
    )
```

Creating Tables - Code

Tables can be altered using the `ALTER TABLE` command.

The basic syntax is

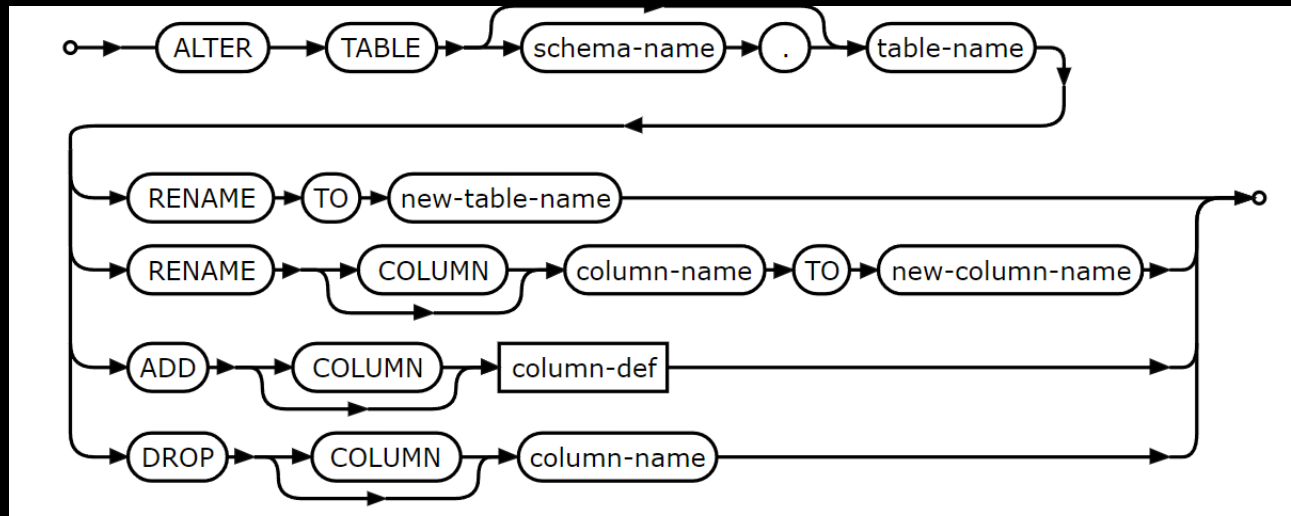
```
ALTER TABLE TableName Operation OperationSpecifics;
```

Operations can include:

- `Rename`
- `Add`
- `Drop`

Sqlite doesn't support all expected operations.

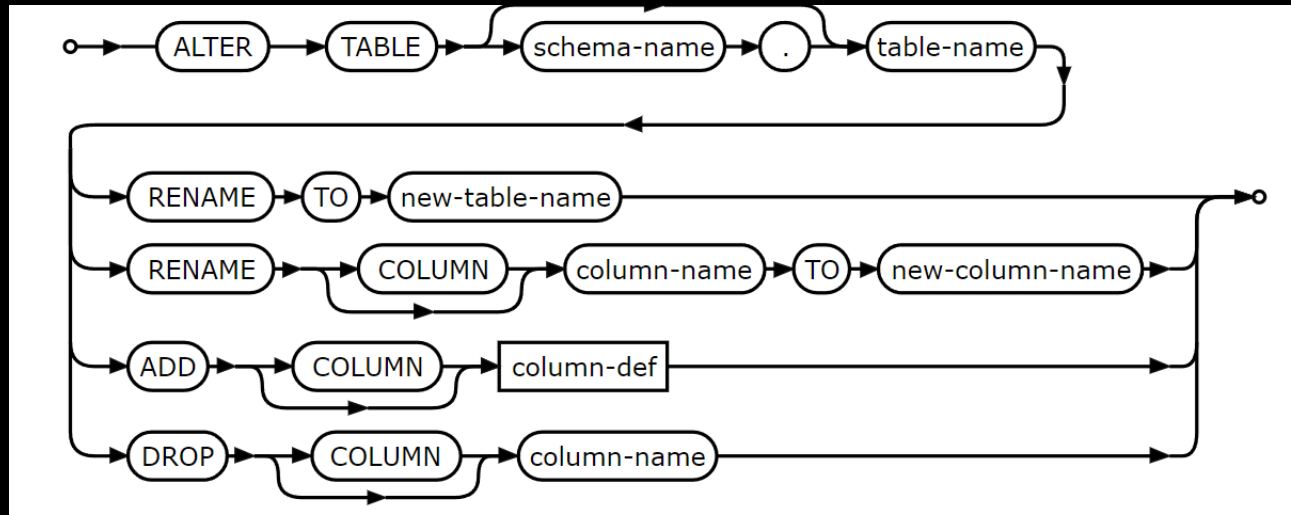
Unsupported operations include `DROP COLUMN` in certain situations as well as changing constraints.
DBVisualizer may be limiting access to changes as well.



Creating Tables - Code

The following is an example of adding a new column to the *JudgeCall* table called *NewField*:

```
ALTER TABLE JudgeCall ADD COLUMN NewField INTEGER
```

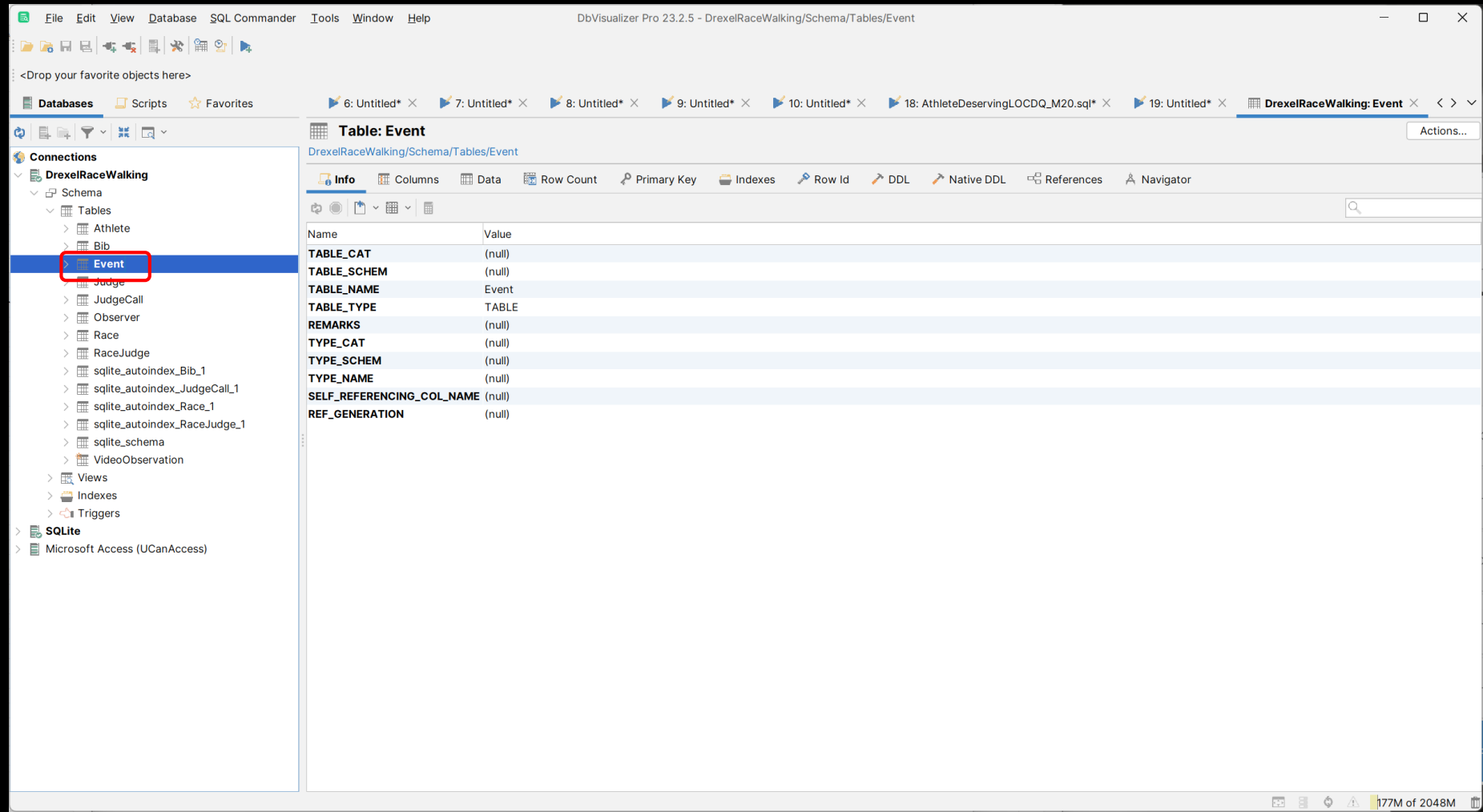


Inserting Data - GUI

Inserting data manually via DBVisualizer is a simple process

Let's add data to the *Event* table.

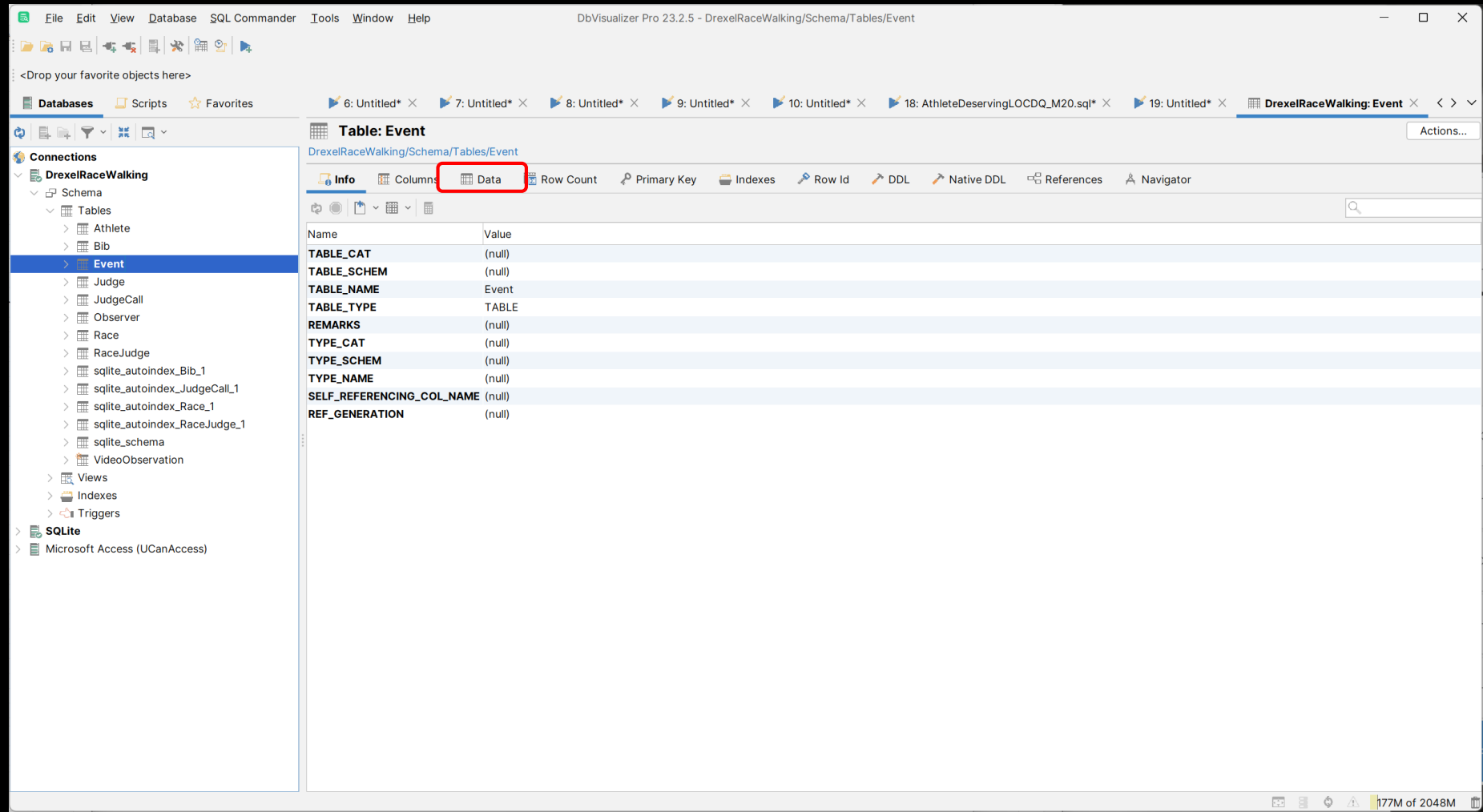
Double Click on the *Event* table:



Inserting Data - GUI

Let's add data to the *Event* table.

Click on the *Data* tab:

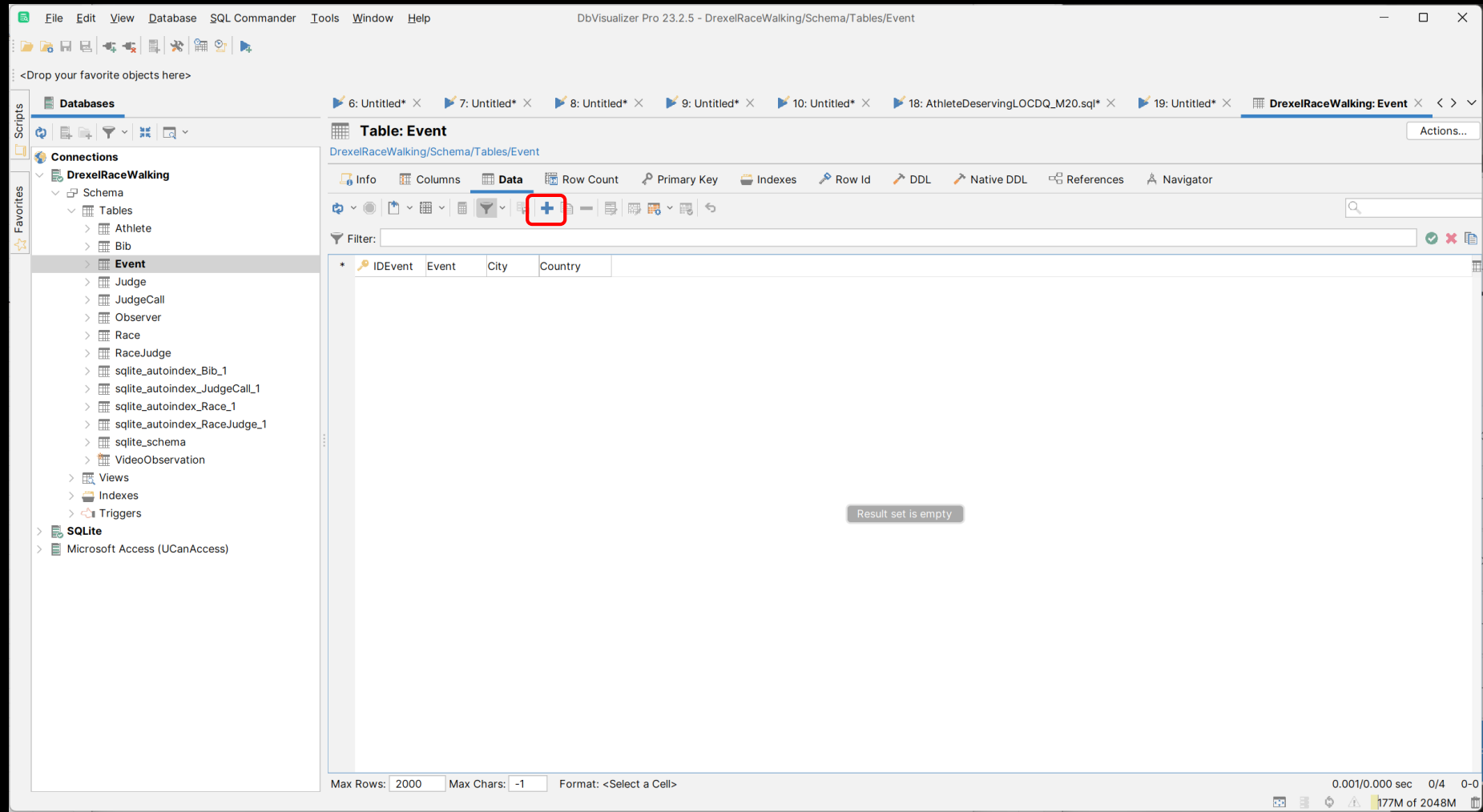


Inserting Data - GUI

Let's add data to the *Event* table.

An empty table appears.

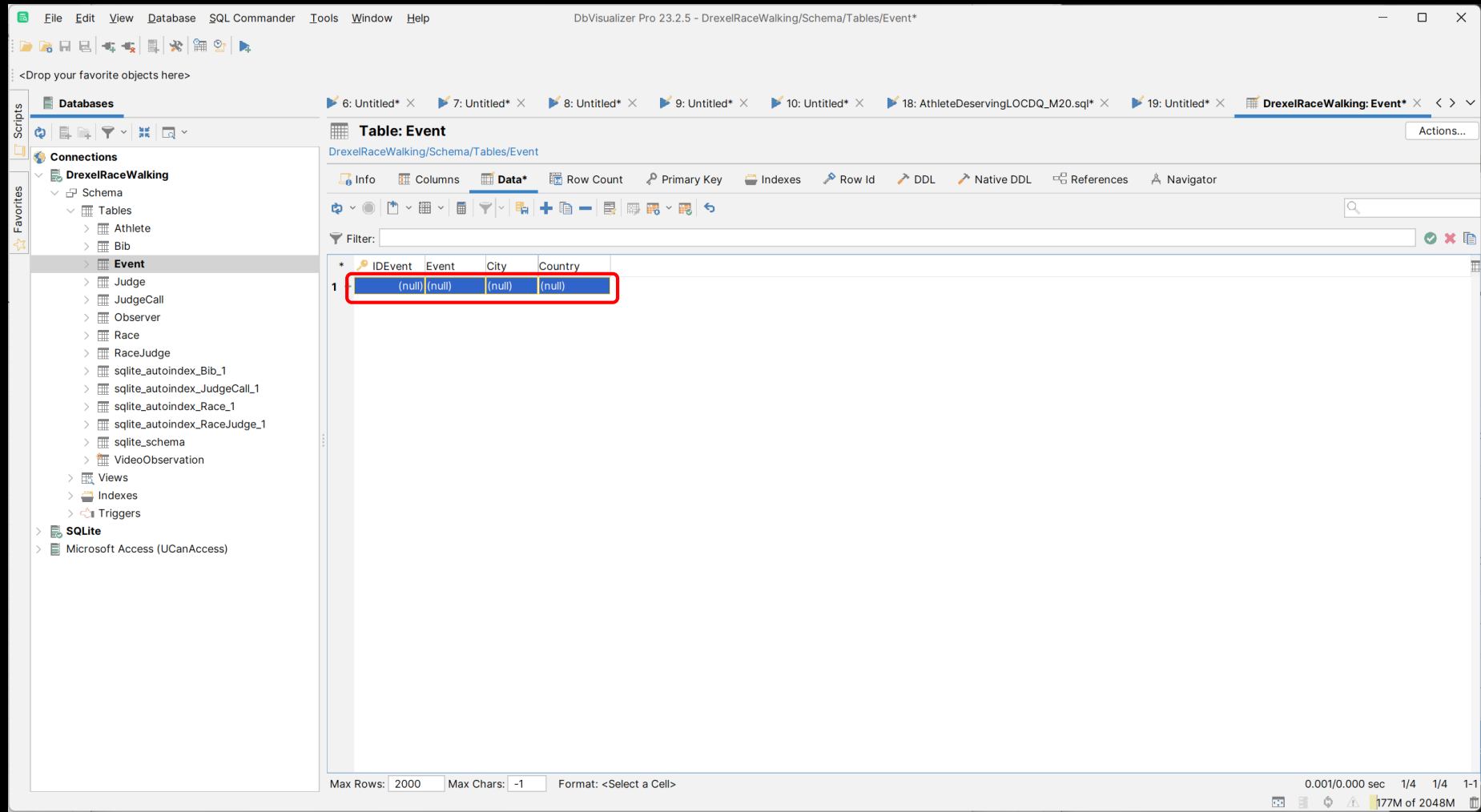
Click on the + icon.



Inserting Data - GUI

Let's add data to the *Event* table.

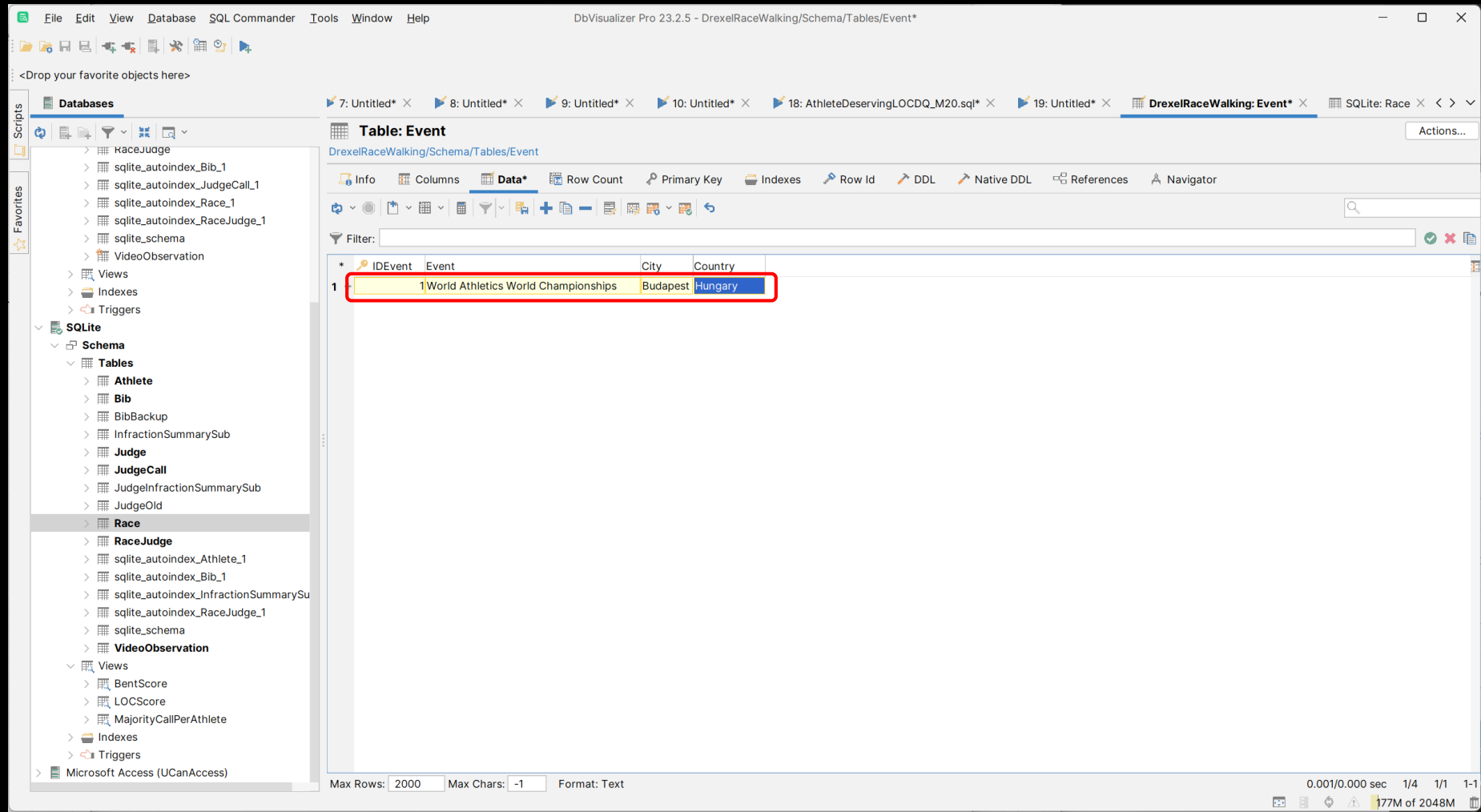
A blank row appears, enter your data:



Inserting Data - GUI

Let's add data to the *Event* table.

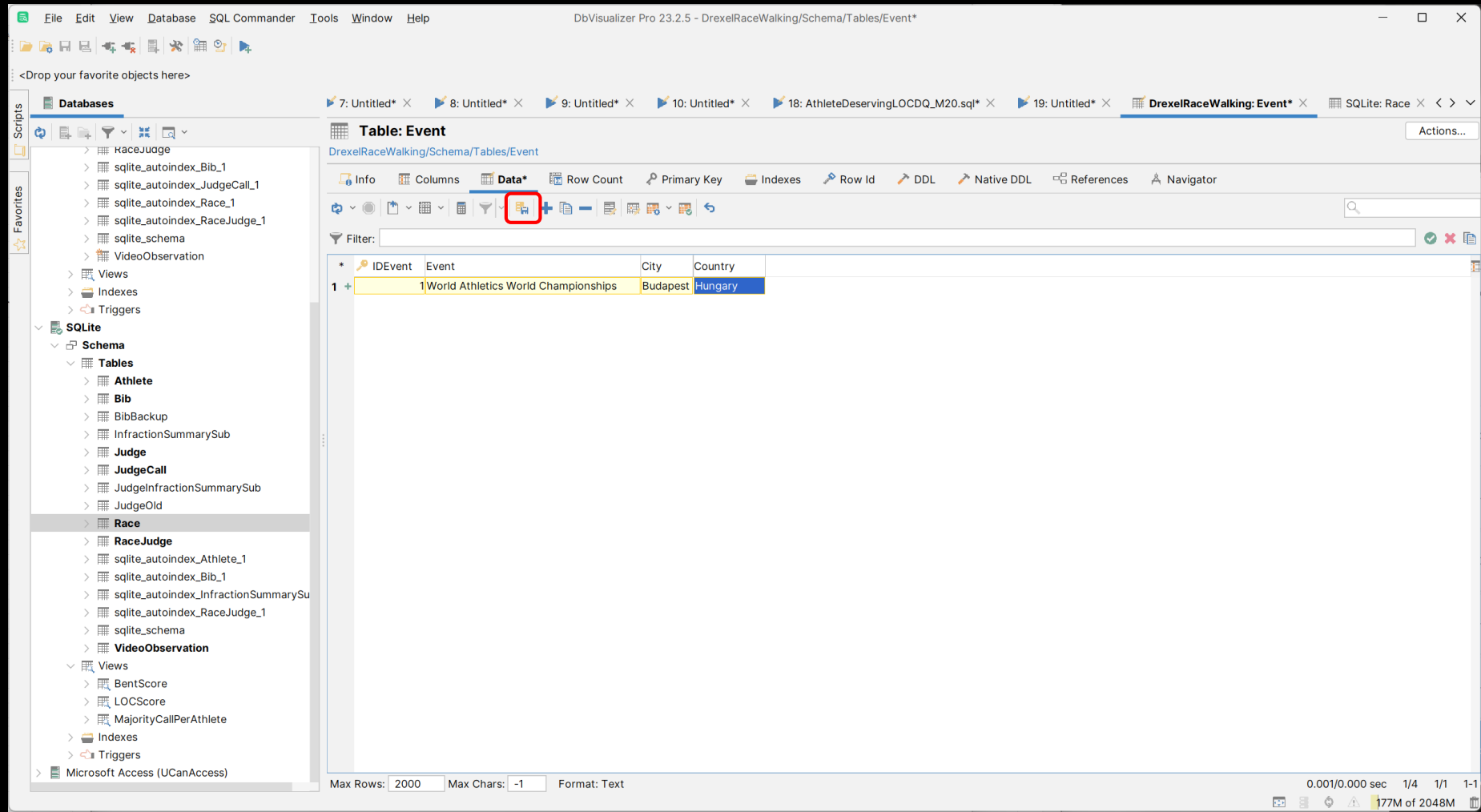
The data is entered, but it is not saved:



Inserting Data - GUI

Let's add data to the *Event* table.

Click on the *Save* icon:

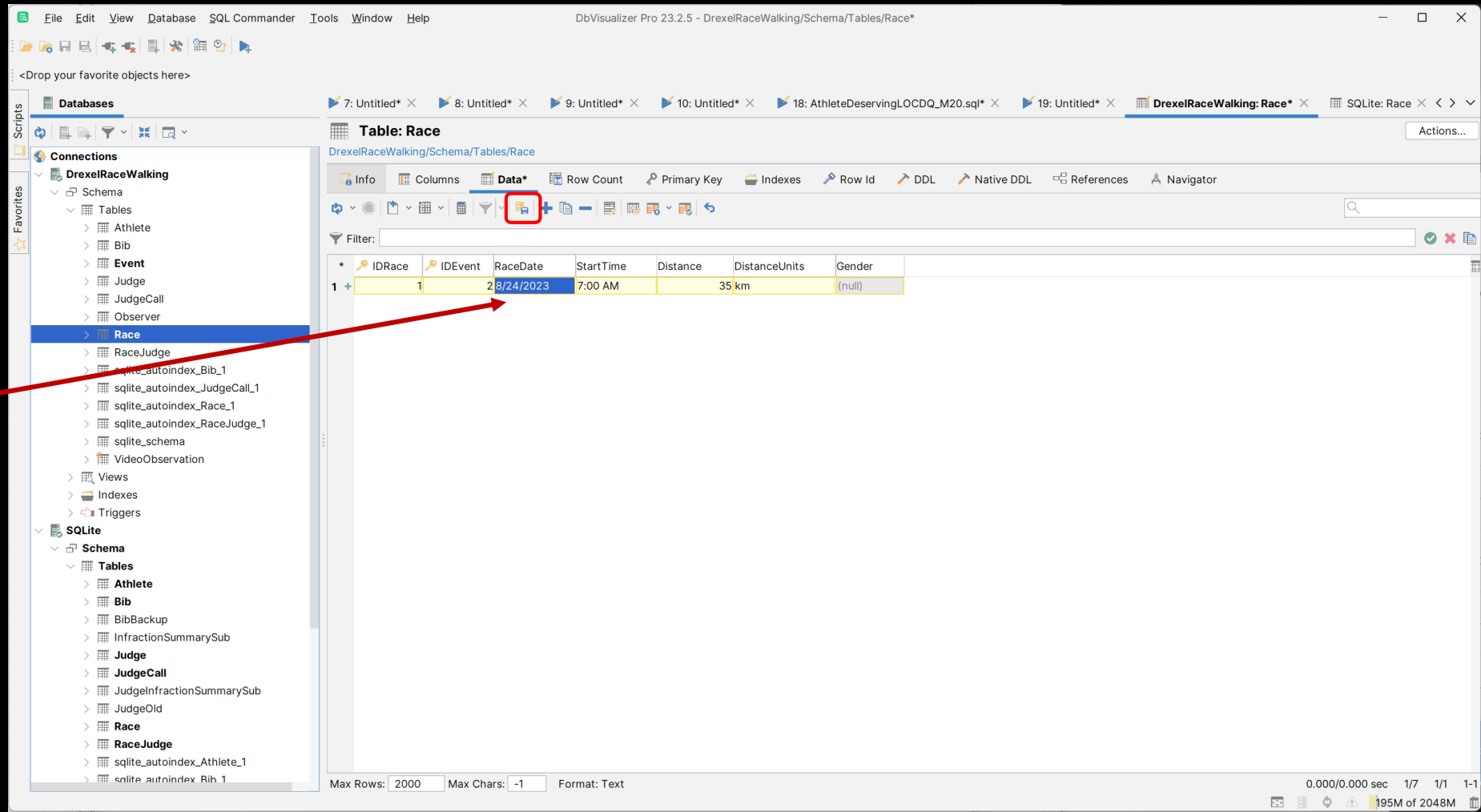


Inserting Data - GUI

Let's add data to the *Race* table.

We have 4 races, we enter them the same way

Let's see what happens though if we make mistakes. Enter:



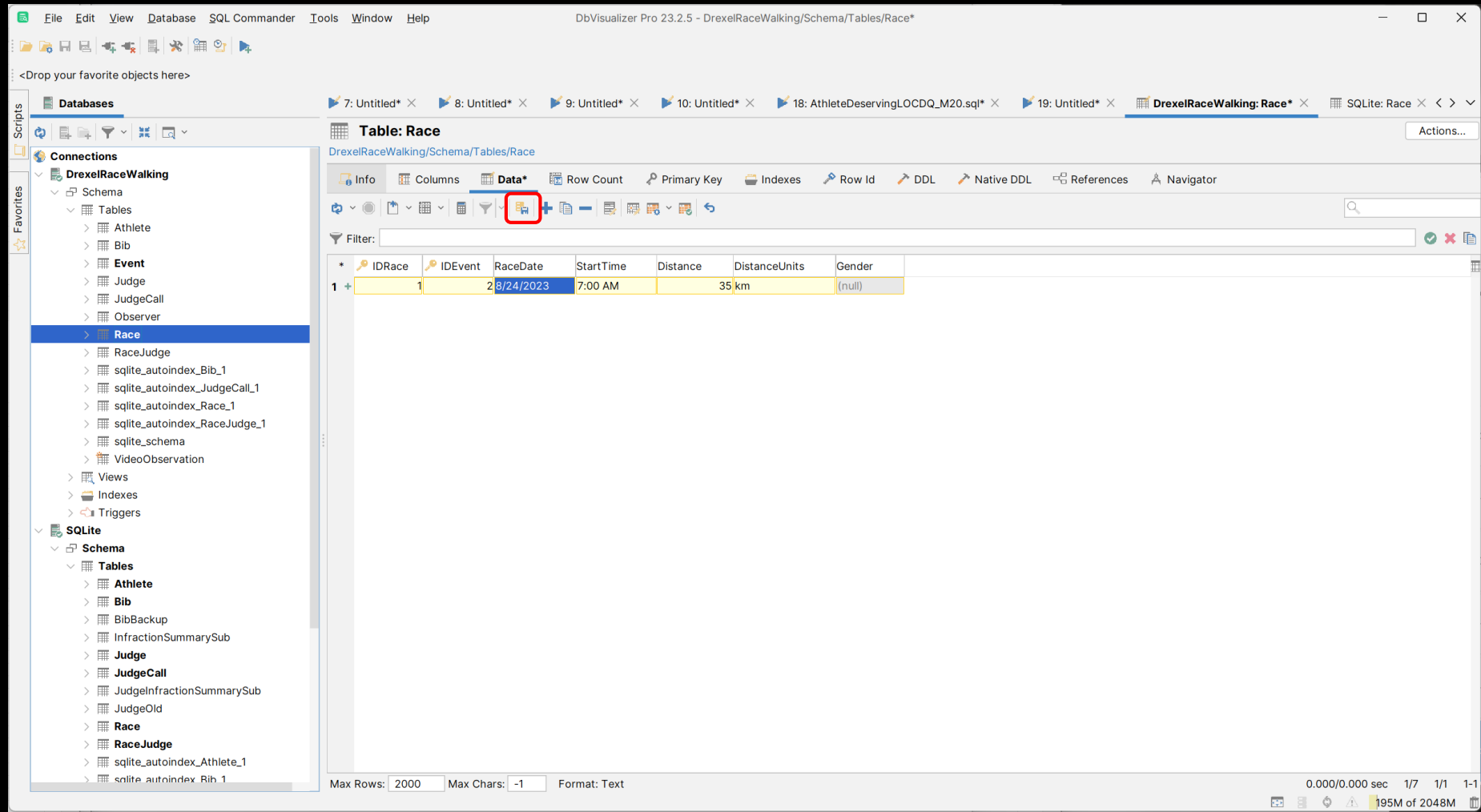
Note, the incorrect date format. ☹

Inserting Data - GUI

What's wrong with the data?

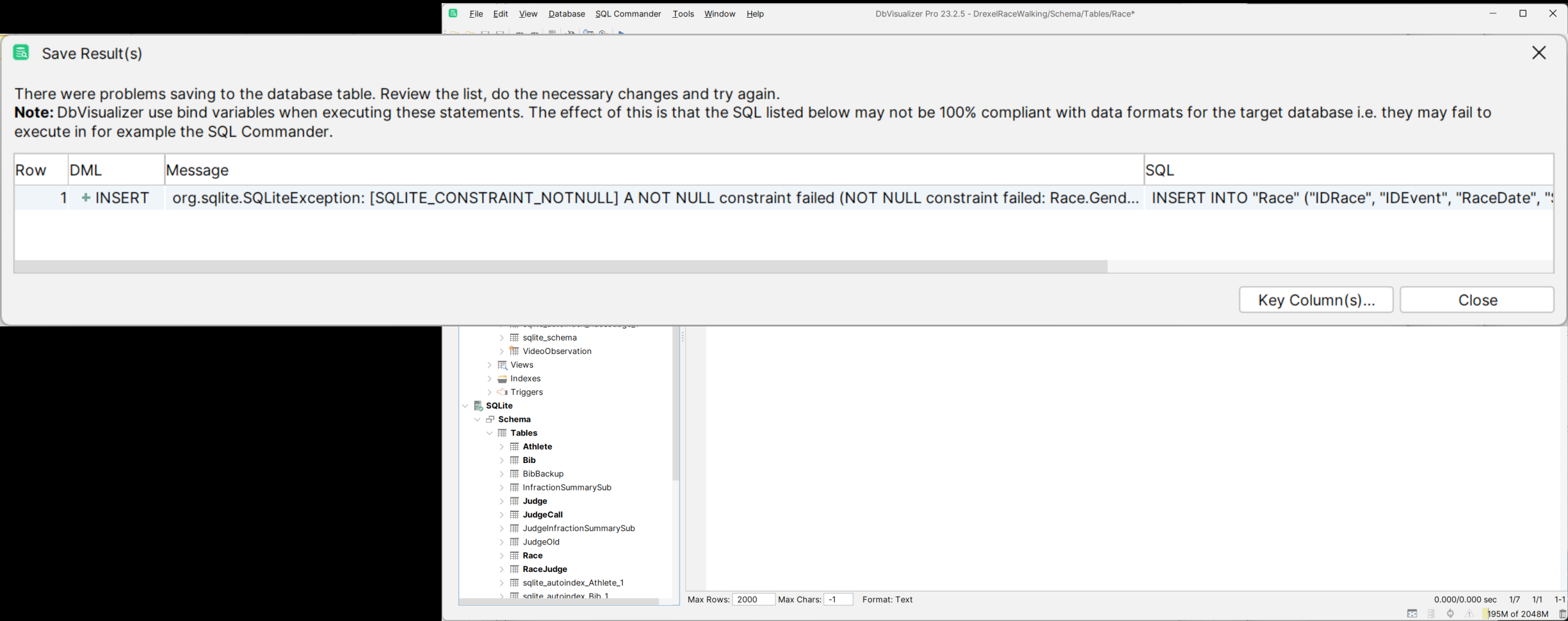
1) There is no *IDEvent* in the *Event* table where `IDEvent = 2`
2) *Gender* is a required field

Click the save icon anyway:



Inserting Data - GUI

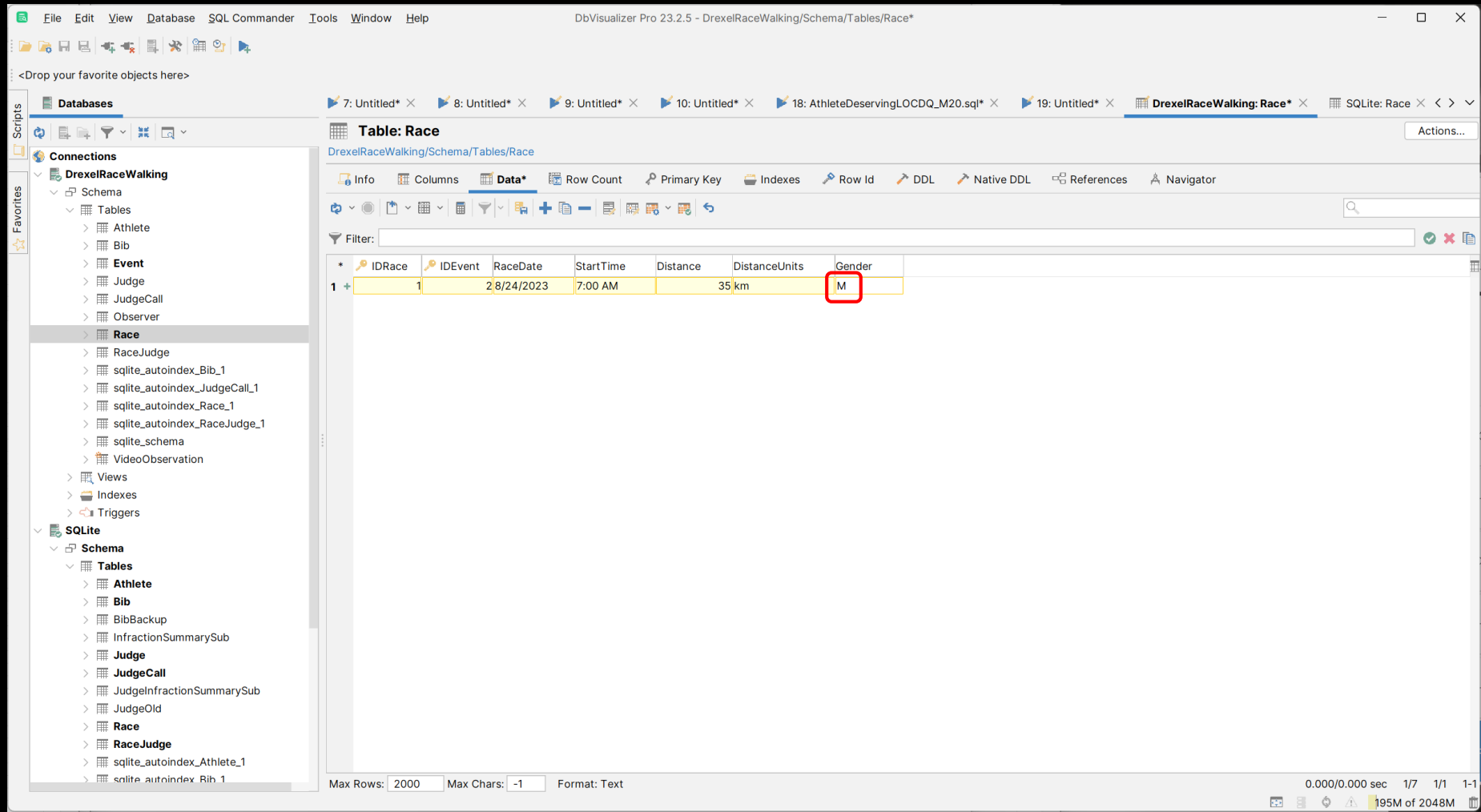
The follow error appears:



Inserting Data - GUI

Hit *Close* and enter the *Gender* as `"M"`

Hit the *Save* icon again:

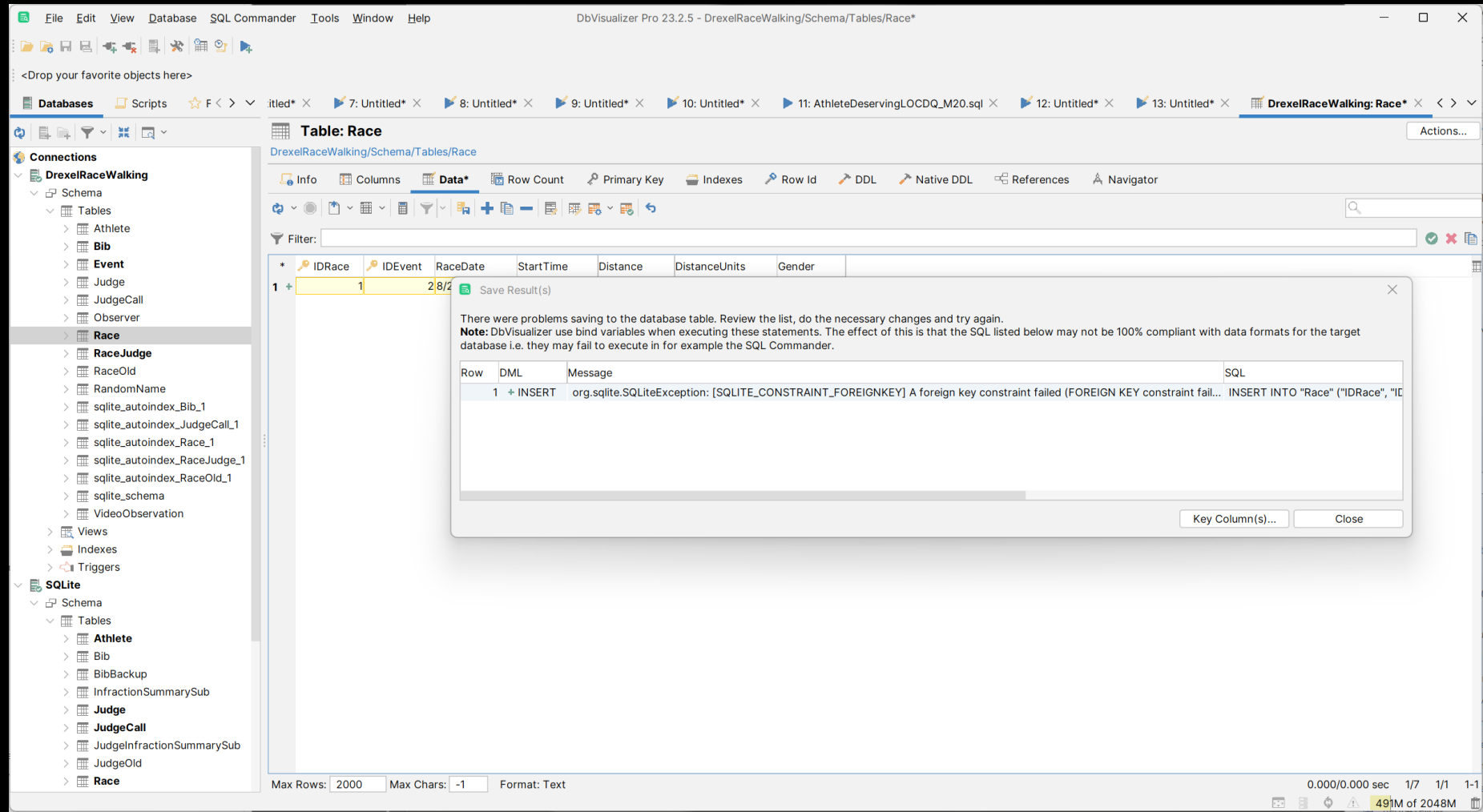


Inserting Data - GUI

Hit *Close* and enter the *Gender* as `"M"`

We should get an error because the Foreign Keys are violated.

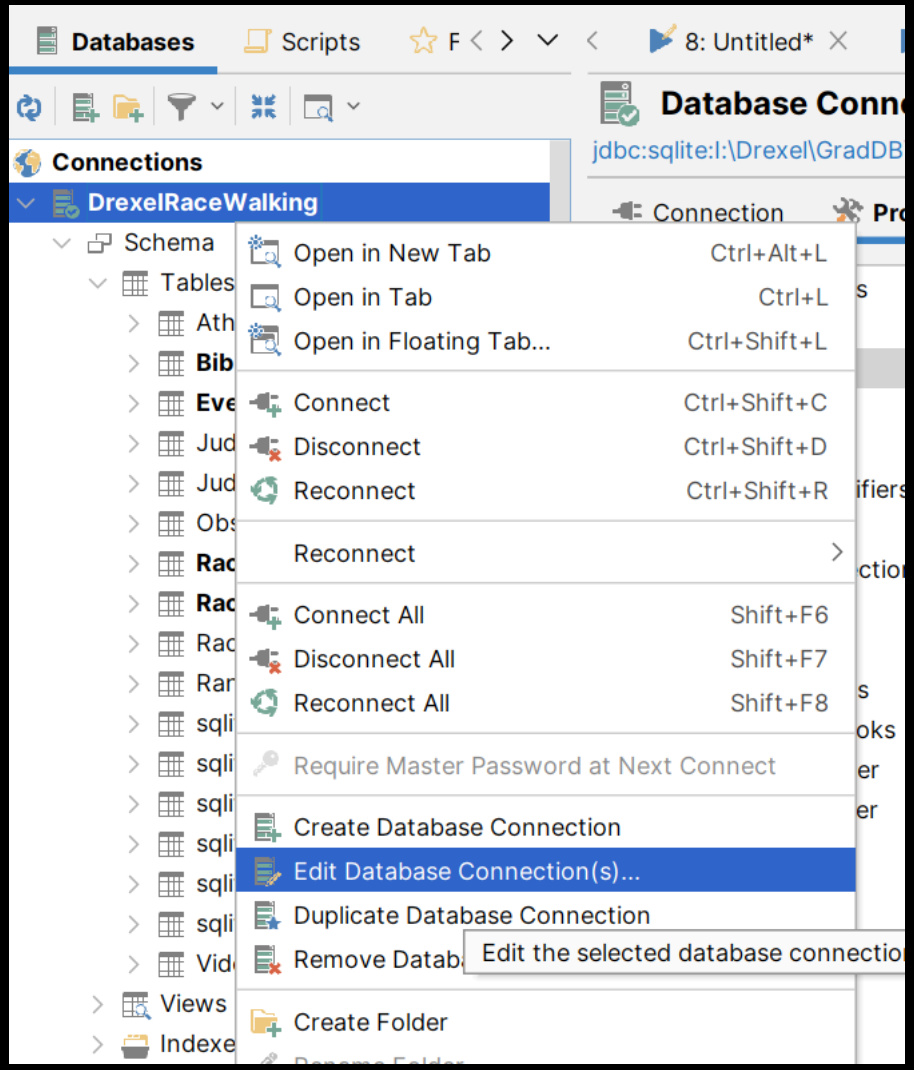
This is a good warning. It prevents accidentally entered data errors.



Inserting Data - GUI

If the error doesn't appear, it's likely that enforcing Foreign Keys is not on by default.

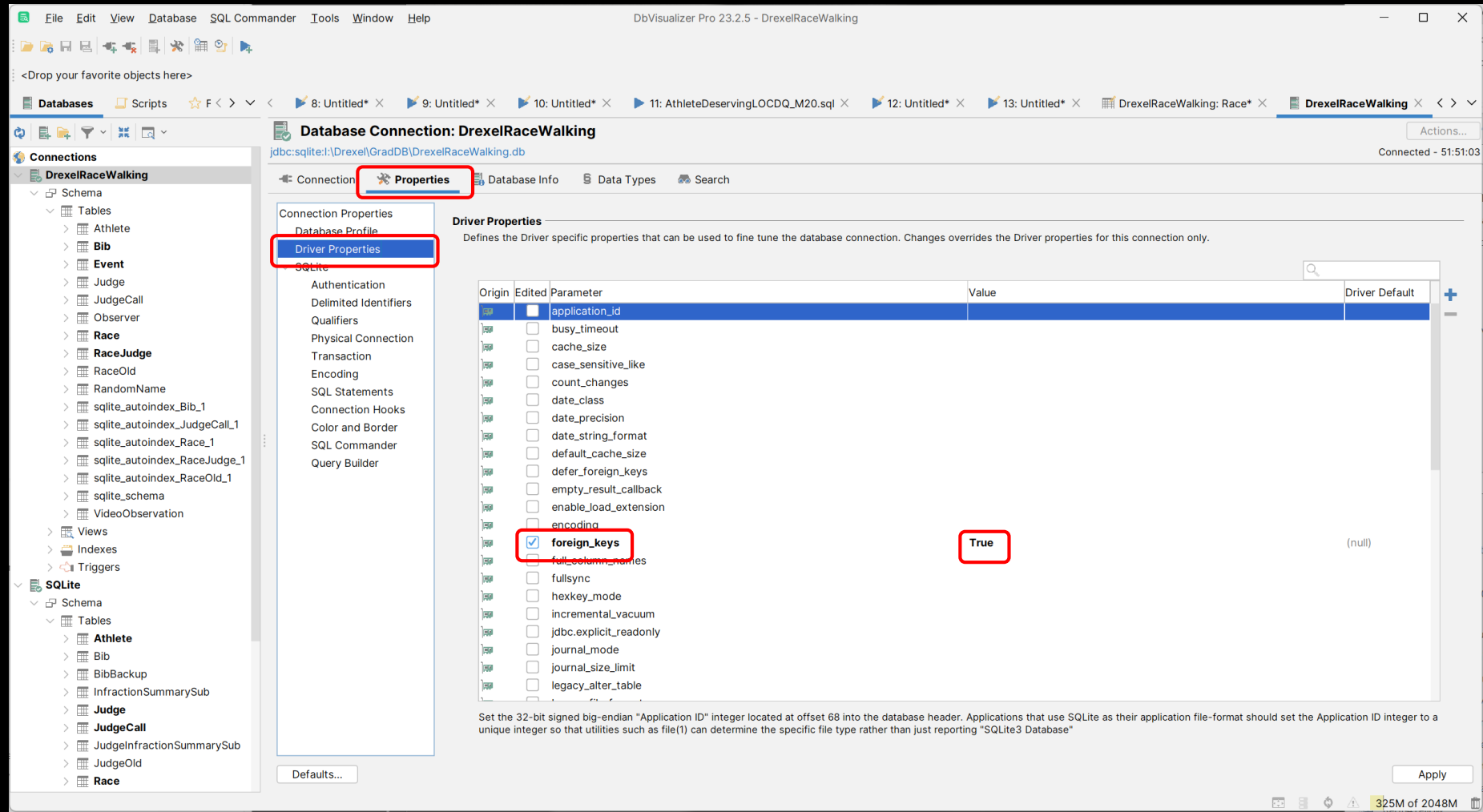
Right-click on the connection, then click on *Edit Database Connections (s)...*



Inserting Data - GUI

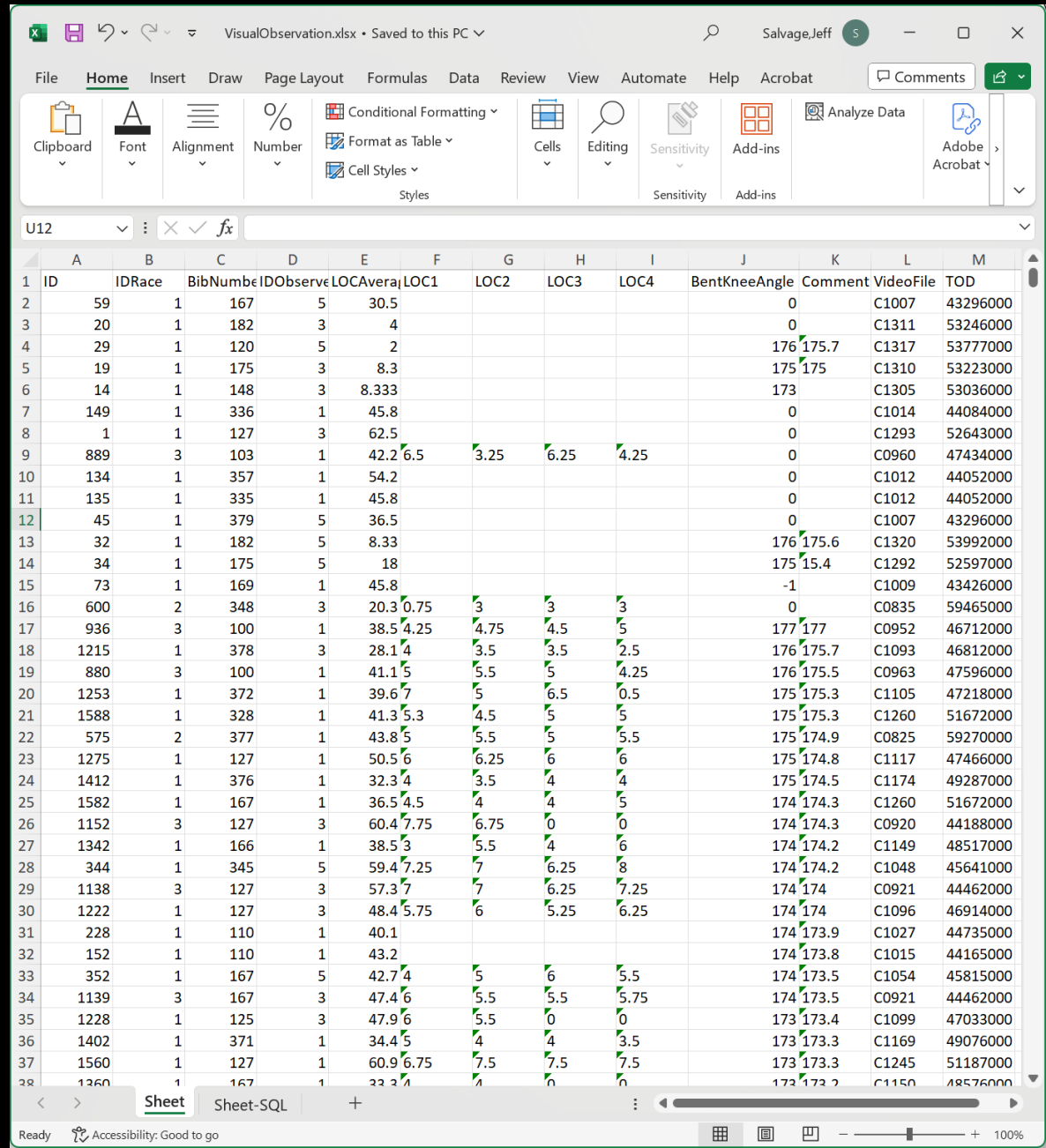
Click on *Properties* and then *Driver Properties*.

Enter True for the value of *foreign_keys*.



Inserting Data - GUI

Sometimes it is required to import data from an external source, like a spreadsheet:



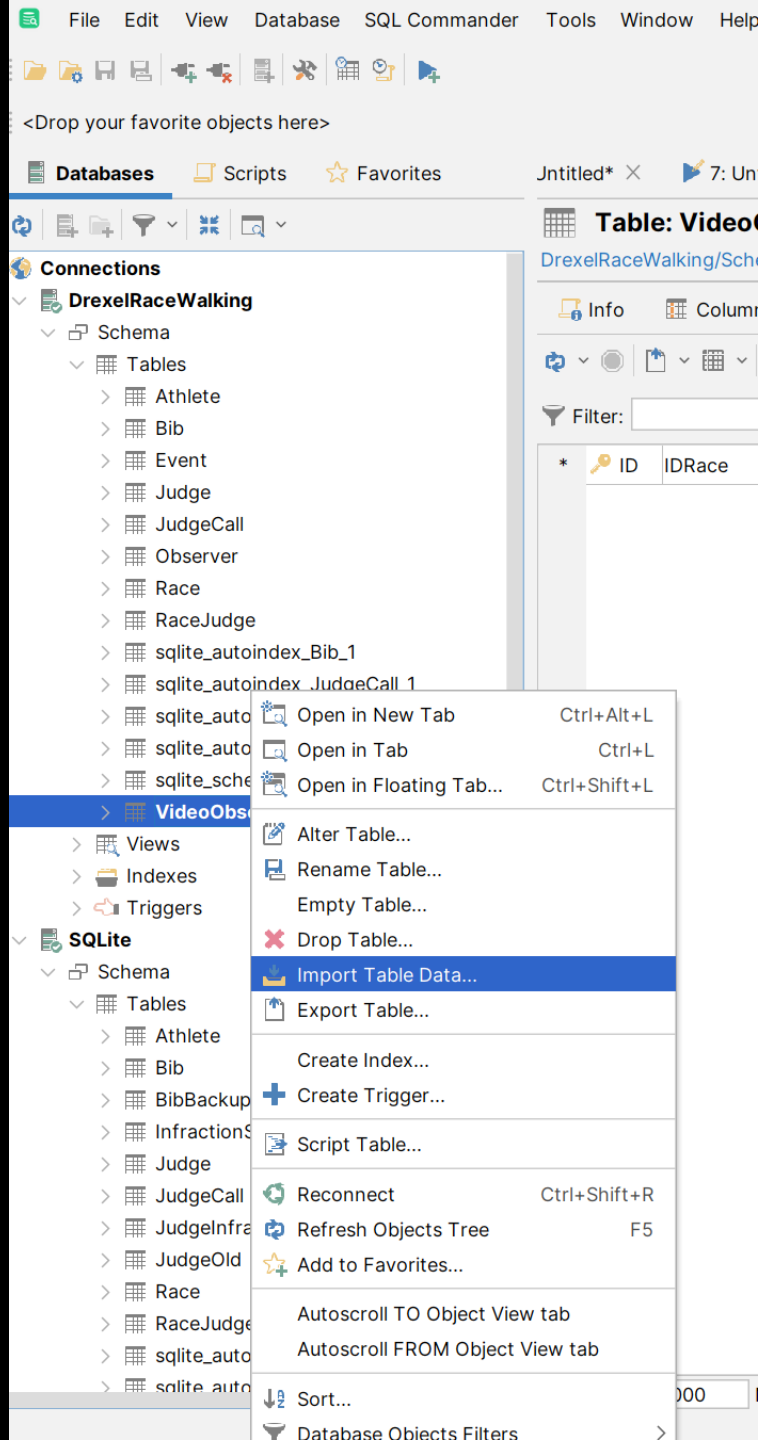
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	ID	IDRace	BibNumbe	IDObserve	LOCAvera	LOC1	LOC2	LOC3	LOC4	BentKneeAngle	Comment	VideoFile	TOD
2	59	1	167	5	30.5					0		C1007	43296000
3	20	1	182	3	4					0		C1311	53246000
4	29	1	120	5	2					176	175.7	C1317	53777000
5	19	1	175	3	8.3					175	175	C1310	53223000
6	14	1	148	3	8.333					173		C1305	53036000
7	149	1	336	1	45.8					0		C1014	44084000
8	1	1	127	3	62.5					0		C1293	52643000
9	889	3	103	1	42.2	6.5	3.25	6.25	4.25	0		C0960	47434000
10	134	1	357	1	54.2					0		C1012	44052000
11	135	1	335	1	45.8					0		C1012	44052000
12	45	1	379	5	36.5					0		C1007	43296000
13	32	1	182	5	8.33					176	175.6	C1320	53992000
14	34	1	175	5	18					175	15.4	C1292	52597000
15	73	1	169	1	45.8					-1		C1009	43426000
16	600	2	348	3	20.3	0.75	3	3	3	0		C0835	59465000
17	936	3	100	1	38.5	4.25	4.75	4.5	5	177	177	C0952	46712000
18	1215	1	378	3	28.1	4	3.5	3.5	2.5	176	175.7	C1093	46812000
19	880	3	100	1	41.1	5	5.5	5	4.25	176	175.5	C0963	47596000
20	1253	1	372	1	39.6	7	5	6.5	0.5	175	175.3	C1105	47218000
21	1588	1	328	1	41.3	5.3	4.5	5	5	175	175.3	C1260	51672000
22	575	2	377	1	43.8	5	5.5	5	5.5	175	174.9	C0825	59270000
23	1275	1	127	1	50.5	6	6.25	6	6	175	174.8	C1117	47466000
24	1412	1	376	1	32.3	4	3.5	4	4	175	174.5	C1174	49287000
25	1582	1	167	1	36.5	4.5	4	4	5	174	174.3	C1260	51672000
26	1152	3	127	3	60.4	7.75	6.75	0	0	174	174.3	C0920	44188000
27	1342	1	166	1	38.5	3	5.5	4	6	174	174.2	C1149	48517000
28	344	1	345	5	59.4	7.25	7	6.25	8	174	174.2	C1048	45641000
29	1138	3	127	3	57.3	7	7	6.25	7.25	174	174	C0921	44462000
30	1222	1	127	3	48.4	5.75	6	5.25	6.25	174	174	C1096	46914000
31	228	1	110	1	40.1					174	173.9	C1027	44735000
32	152	1	110	1	43.2					174	173.8	C1015	44165000
33	352	1	167	5	42.7	4	5	6	5.5	174	173.5	C1054	45815000
34	1139	3	167	3	47.4	6	5.5	5.5	5.75	174	173.5	C0921	44462000
35	1228	1	125	3	47.9	6	5.5	0	0	173	173.4	C1099	47033000
36	1402	1	371	1	34.4	5	4	4	3.5	173	173.3	C1169	49076000
37	1560	1	127	1	60.9	6.75	7.5	7.5	7.5	173	173.3	C1245	51187000
38	1360	1	167	1	33.3	4	4	0	0	173	173.2	C1150	48576000

Inserting Data - GUI

Another way to import data is to use the *Import Table Data* feature.

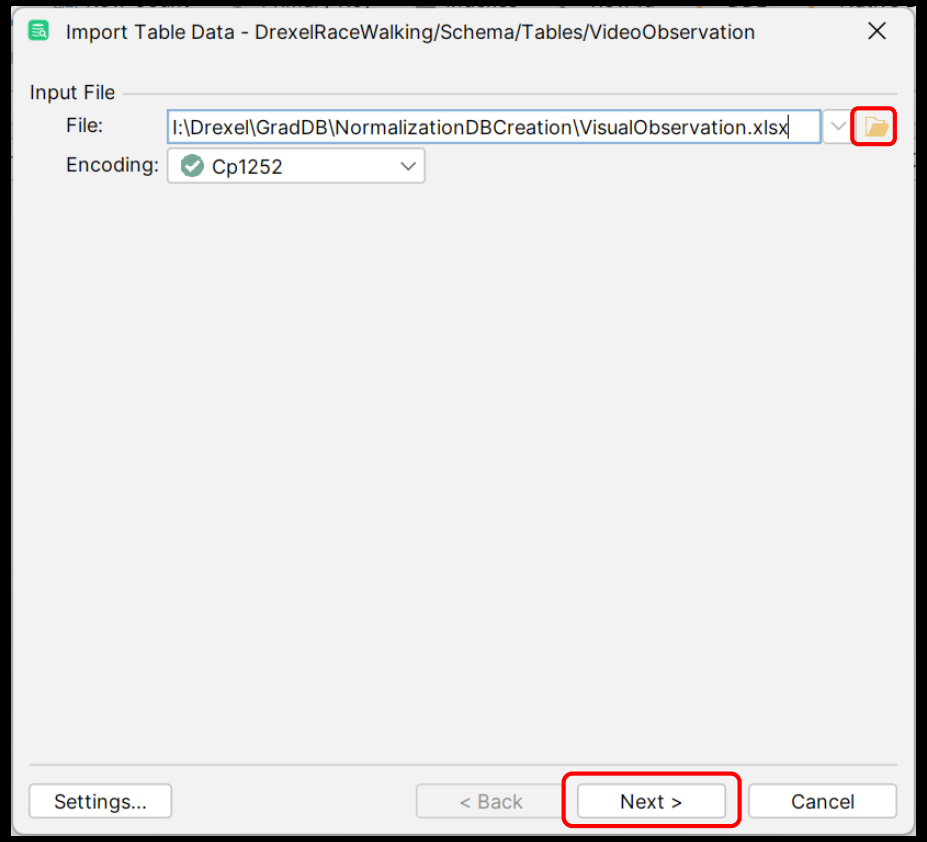
Right-click on the destination table.

Select *Import Table Data* from the popup menu:



Inserting Data - GUI

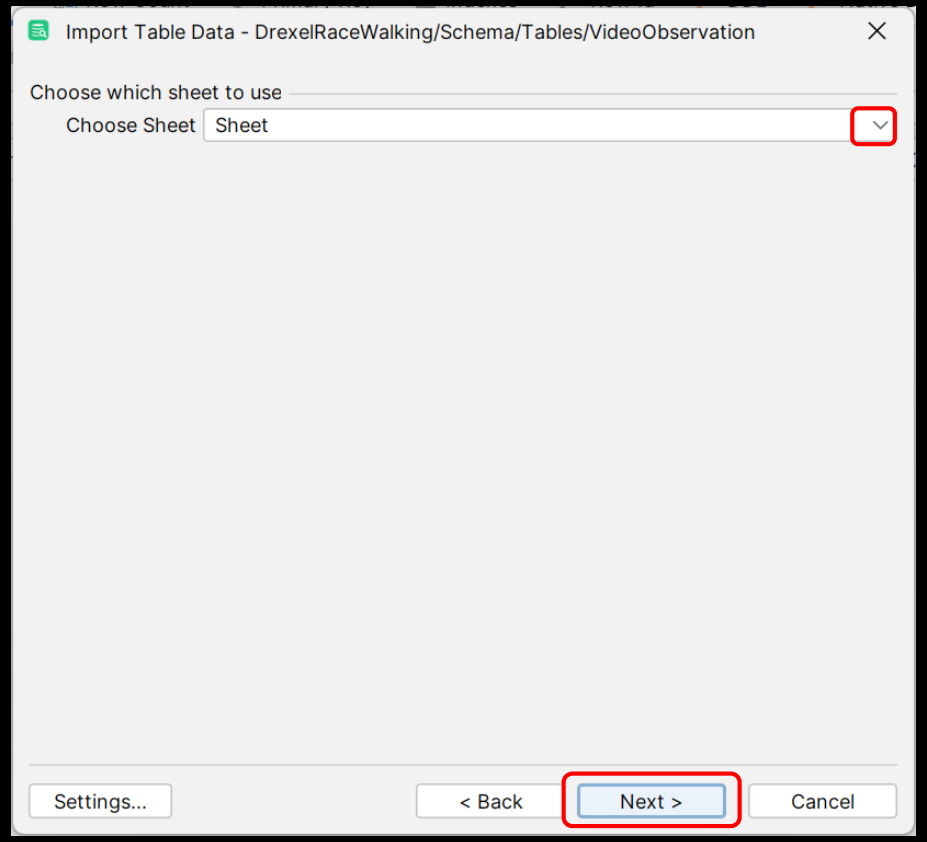
Select the file you wish to import and click next:



Inserting Data - GUI

In this case, there is only one sheet in the spreadsheet, but we could select the other sheets if they existed.

Click on the *Next* button.

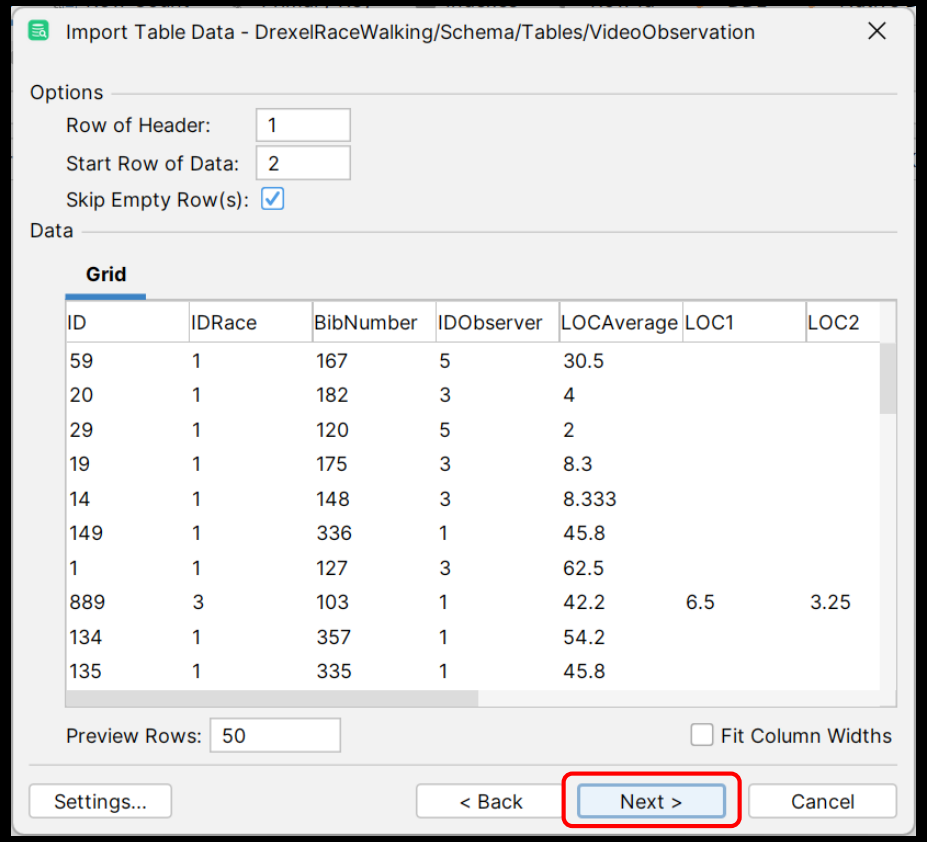


Inserting Data - GUI

We get a preview of what the data looks like.

We can even specify the row for headers and where the data starts.

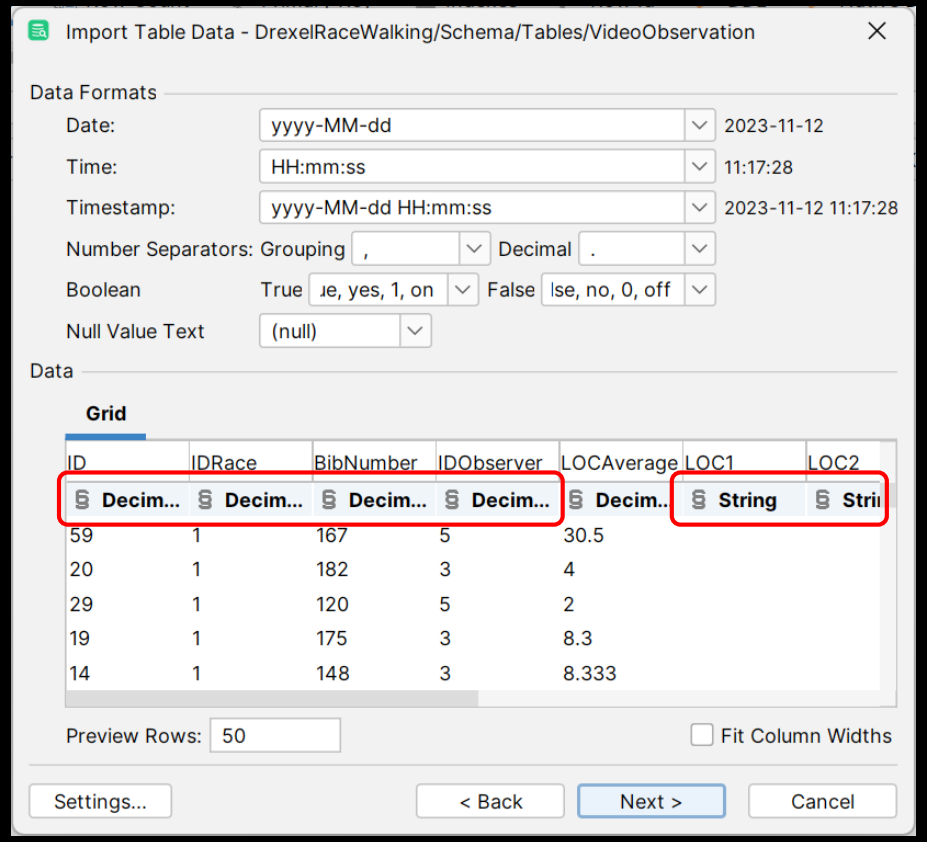
Click on the *Next* button.



Inserting Data - GUI

We can set formats and change data types if needed.

Most in this case, are defaulted incorrectly:

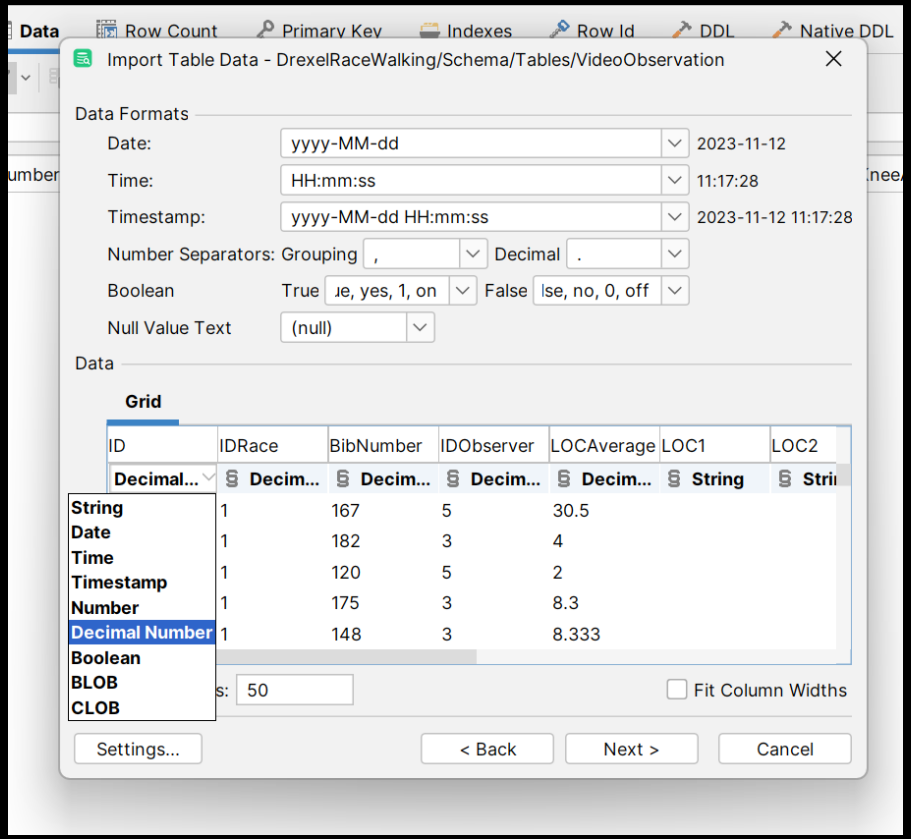


Inserting Data - GUI

Click on each field and correct the data types.

ID: Number
IDRace: Number
BibNumber: Number
IDObserver: Number
LOC1: Decimal
LOC2: Decimal
LOC3: Decimal
LOC4: Decimal
TOD: String (More on this later)

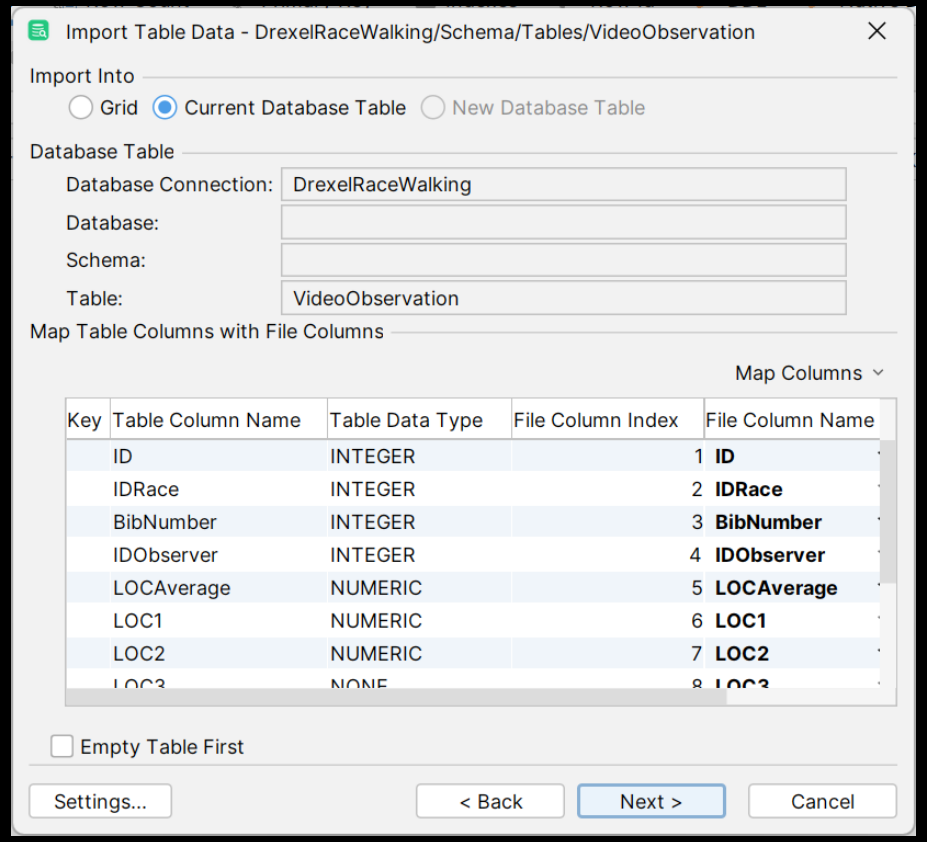
Click on the *Next* button.



Inserting Data - GUI

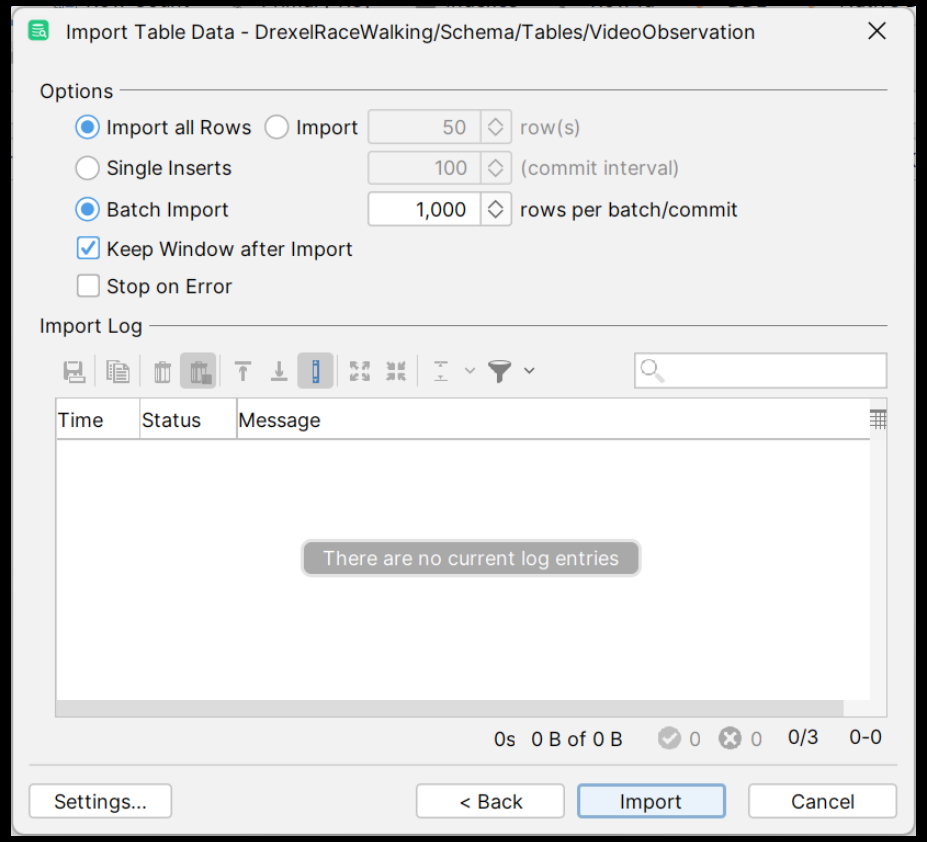
Confirm your changes

Click on the *Next* button.



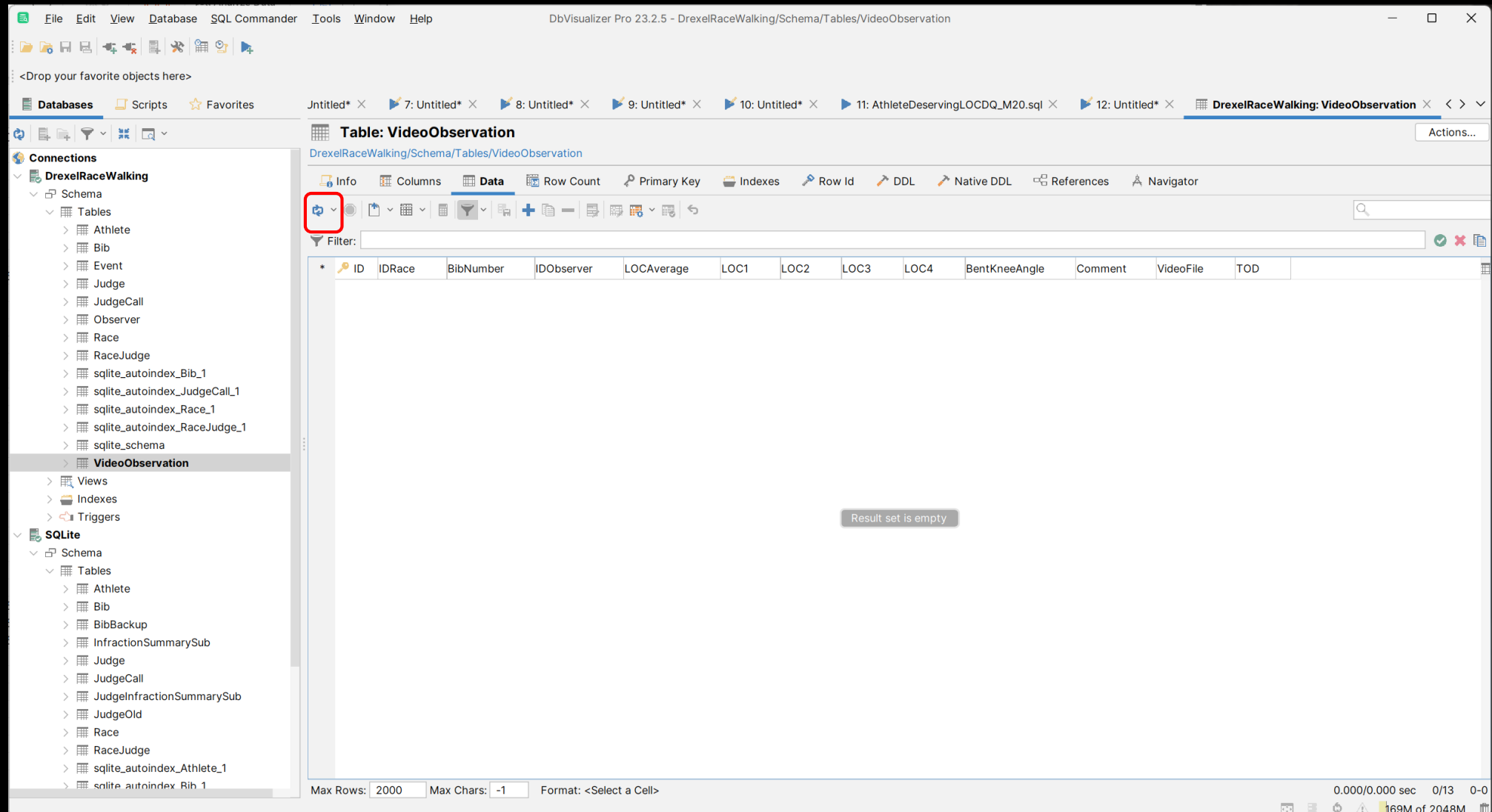
Inserting Data - GUI

Click on the *Import* button.



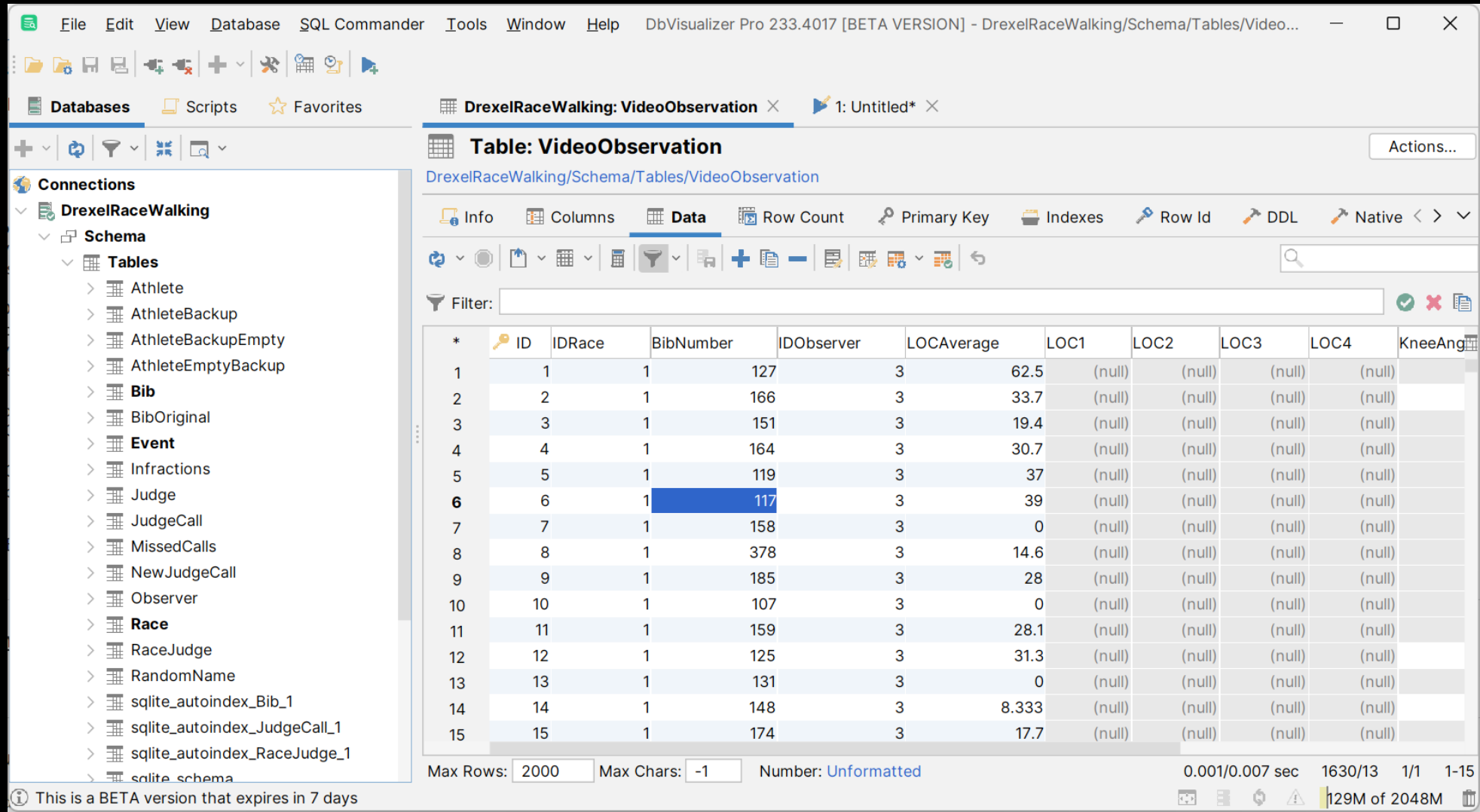
Inserting Data - GUI

To see your changes, click the *Refresh* icon.



Inserting Data - GUI

Your imported data appears:



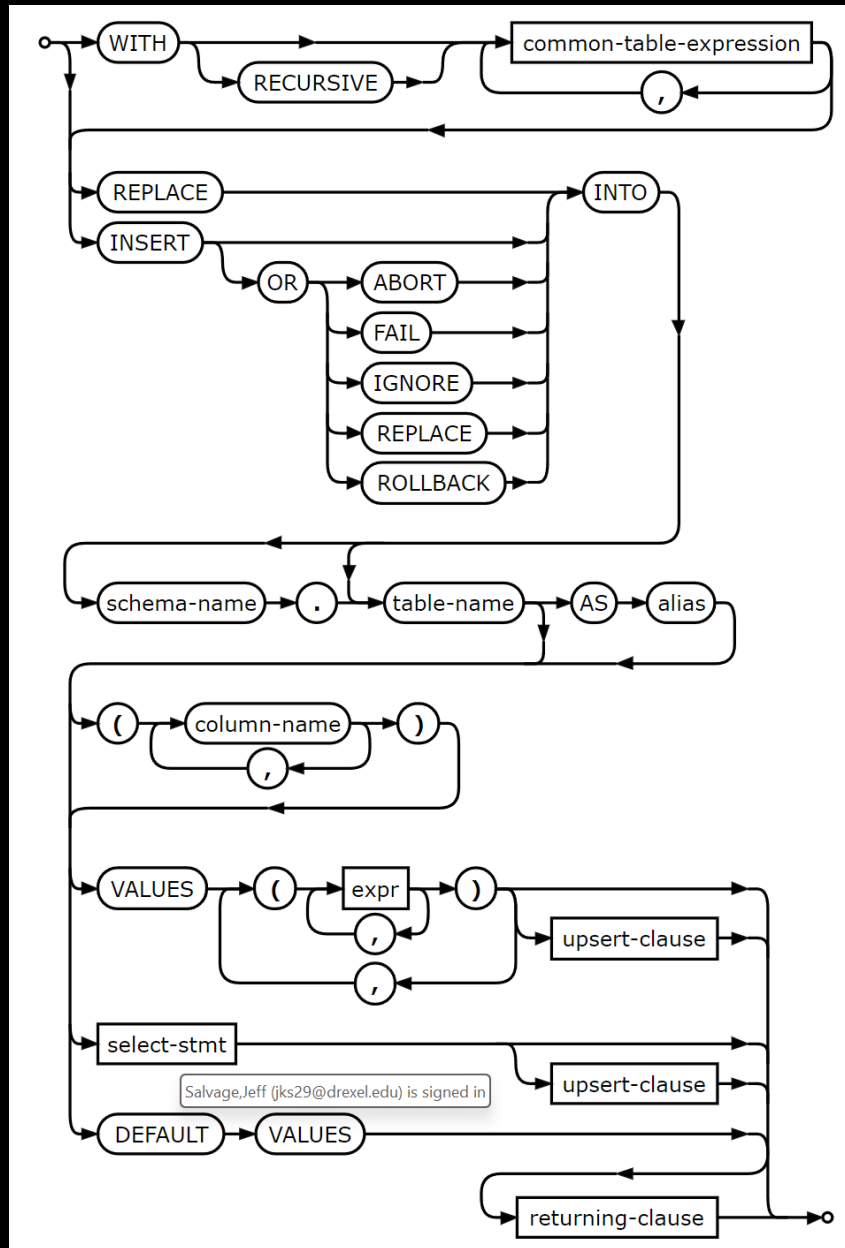
Inserting Data – Via Code

The SQL INSERT statement can be used to insert data into the database one row at a time:

The SQL INSERT statement has many options.

The basic for entering a list of values is:

```
INSERT INTO table VALUES(...);
```



For more information see:
https://www.sqlite.org/lang_insert.html

Inserting Data – Via Code

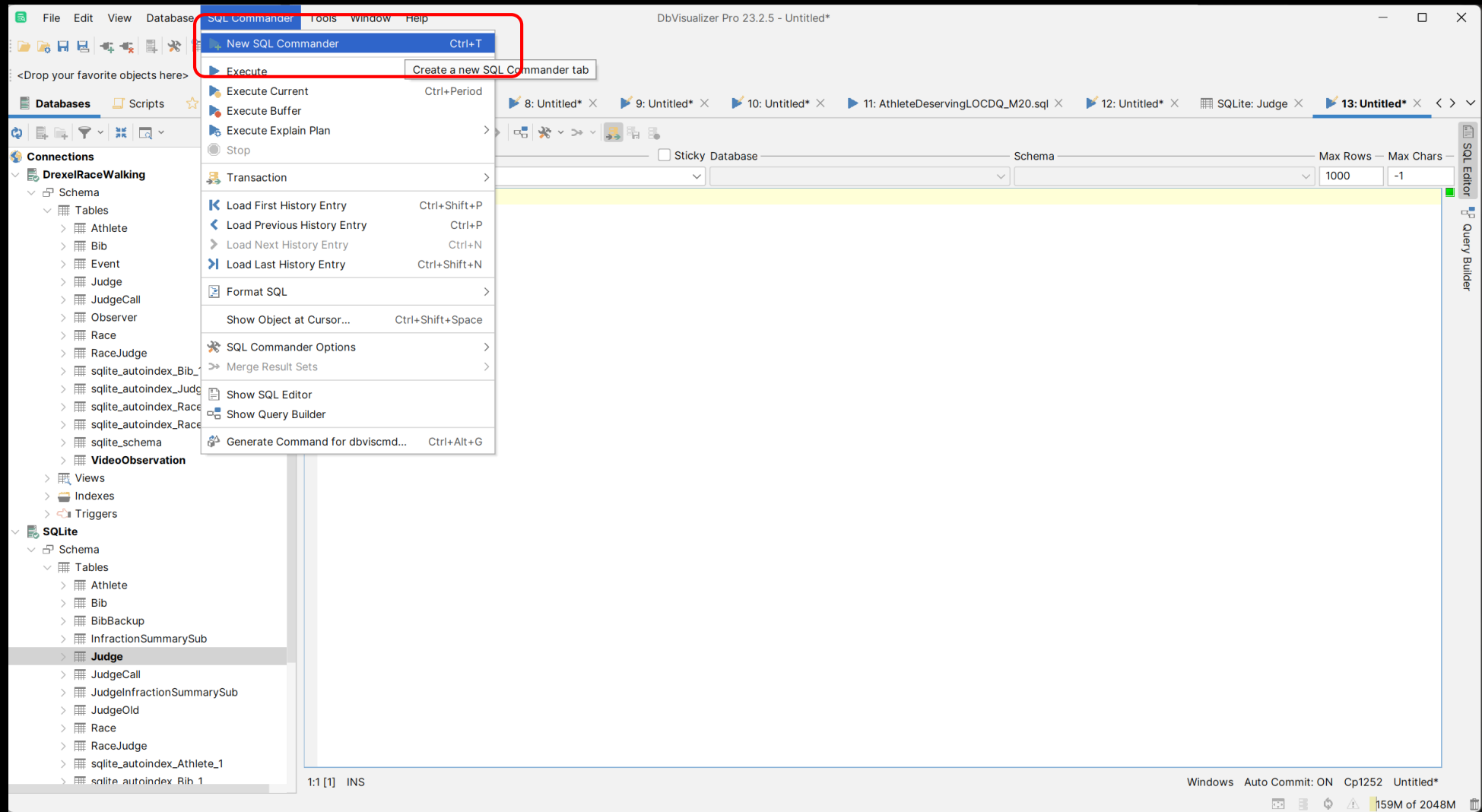
Let's create 8 `INSERT` statements to load the *Judge* table:

```
INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(1, 'FRA', 'Jean-Pierre', 'DAHM');

INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(2, 'POR', 'José', 'DIAS');

INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(3, 'FIN', 'Anne', 'FRÖBERG');

INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(4, 'CAN', 'Daniel', 'MICHAUD');

INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(5, 'ARG', 'Guillermo', 'PERA VALLEJOS');

INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(6, 'GRB', 'Ian', 'RICHARDS');

INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName) VALUES(7, 'ESP', 'Sergio', 'SOLANA SOR');

INSERT INTO Judge (IDJudge, CountryCode, FirstName, LastName)
VALUES(8, 'HKG', 'Man Chun', 'YEUNG');
```

IDJudge	CountryCode	FirstName	LastName
1	FRA	Jean-Pierre	DAHM
2	POR	José	DIAS
3	FIN	Anne	FRÖBERG
4	CAN	Daniel	MICHAUD
5	ARG	Guillermo	PERA VALLEJOS
6	GRB	Ian	RICHARDS
7	ESP	Sergio	SOLANA SOR
8	HKG	Man Chun	YEUNG

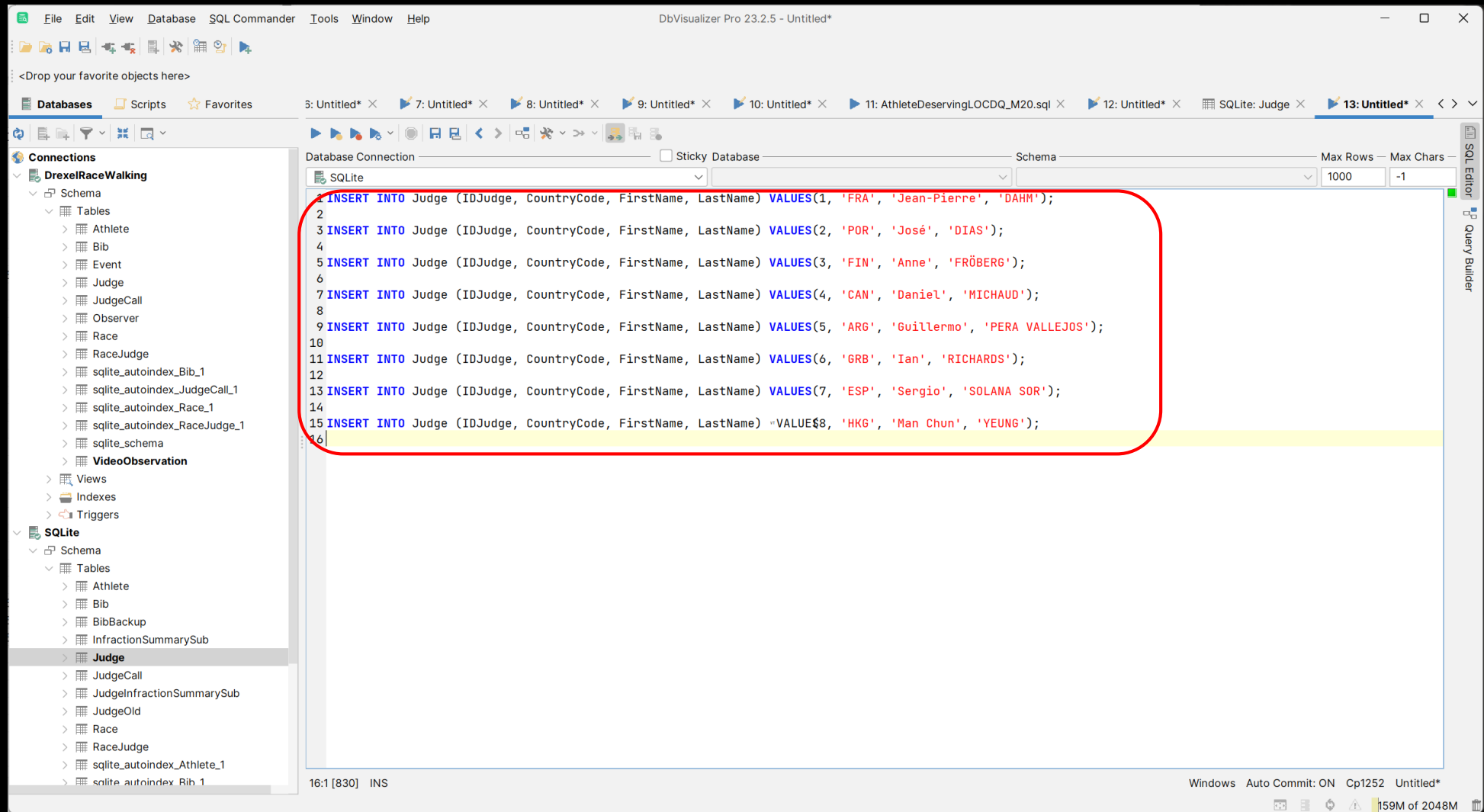
Inserting Data – Via Code

Open the SQL Commander by clicking on the *SQL Commander* menu and the *New SQL Commander* menu item:



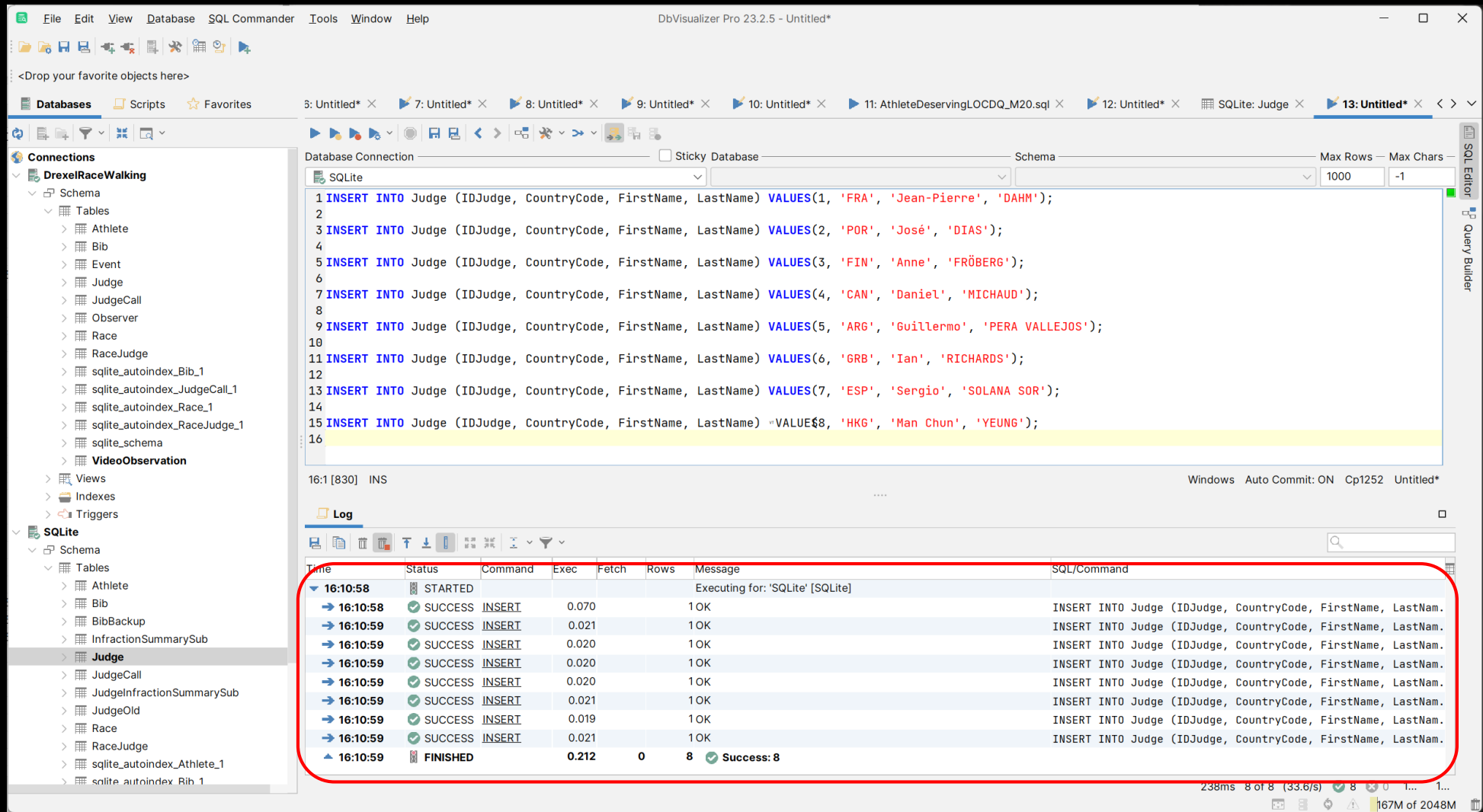
Inserting Data – Via Code

Enter the new code:



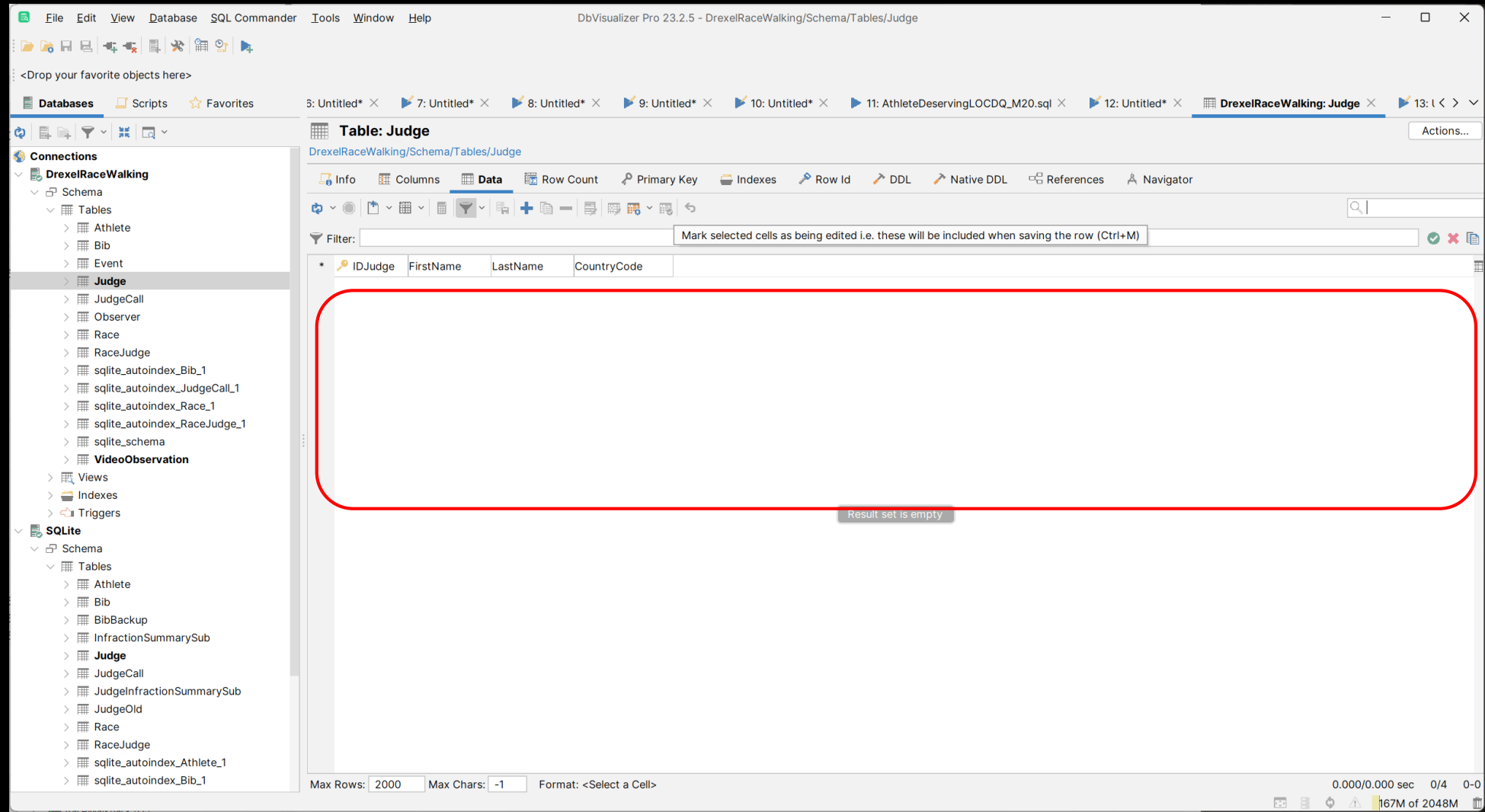
Inserting Data – Via Code

The 8 records are inserted into the database.



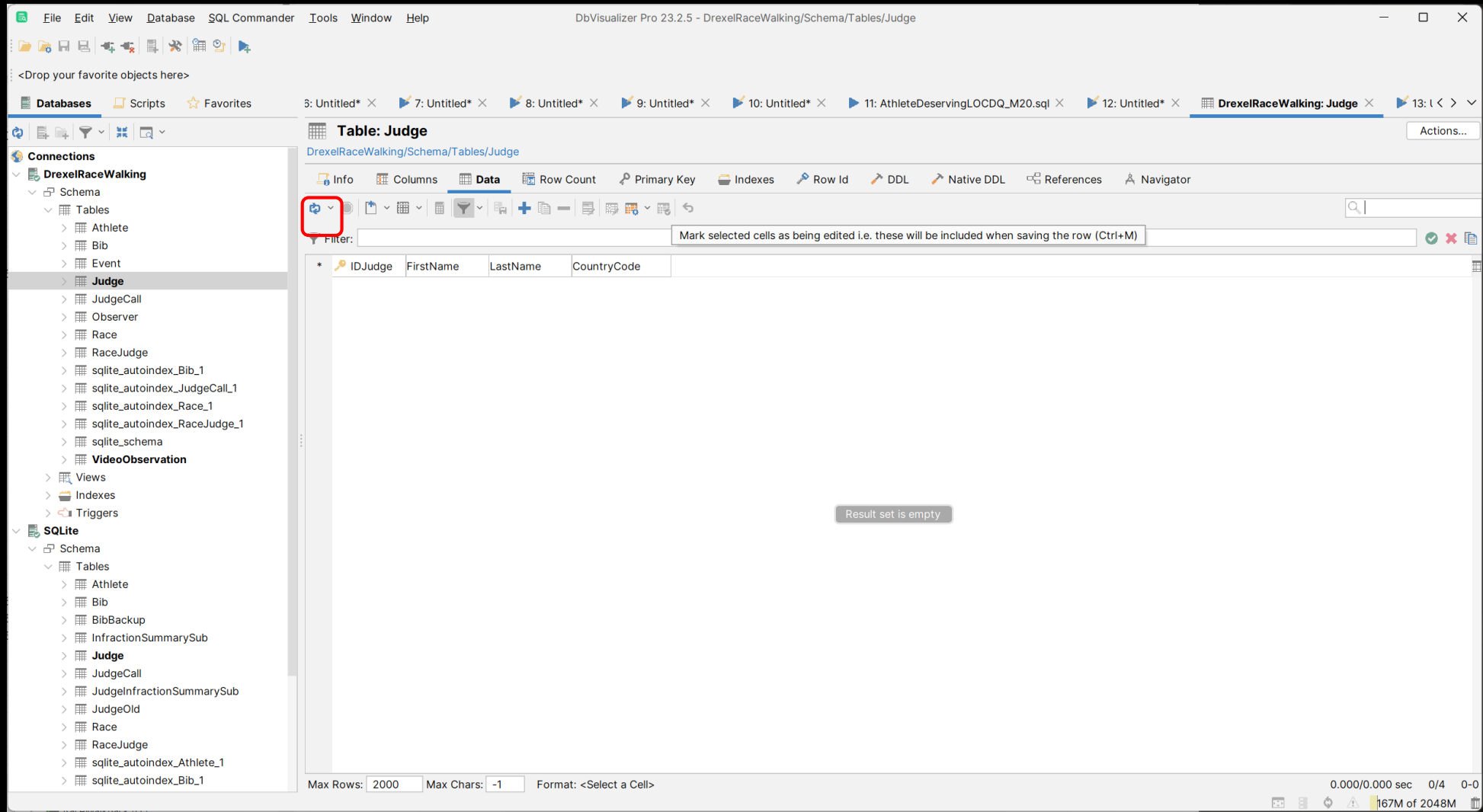
Inserting Data – Via Code

Why is our data not there?



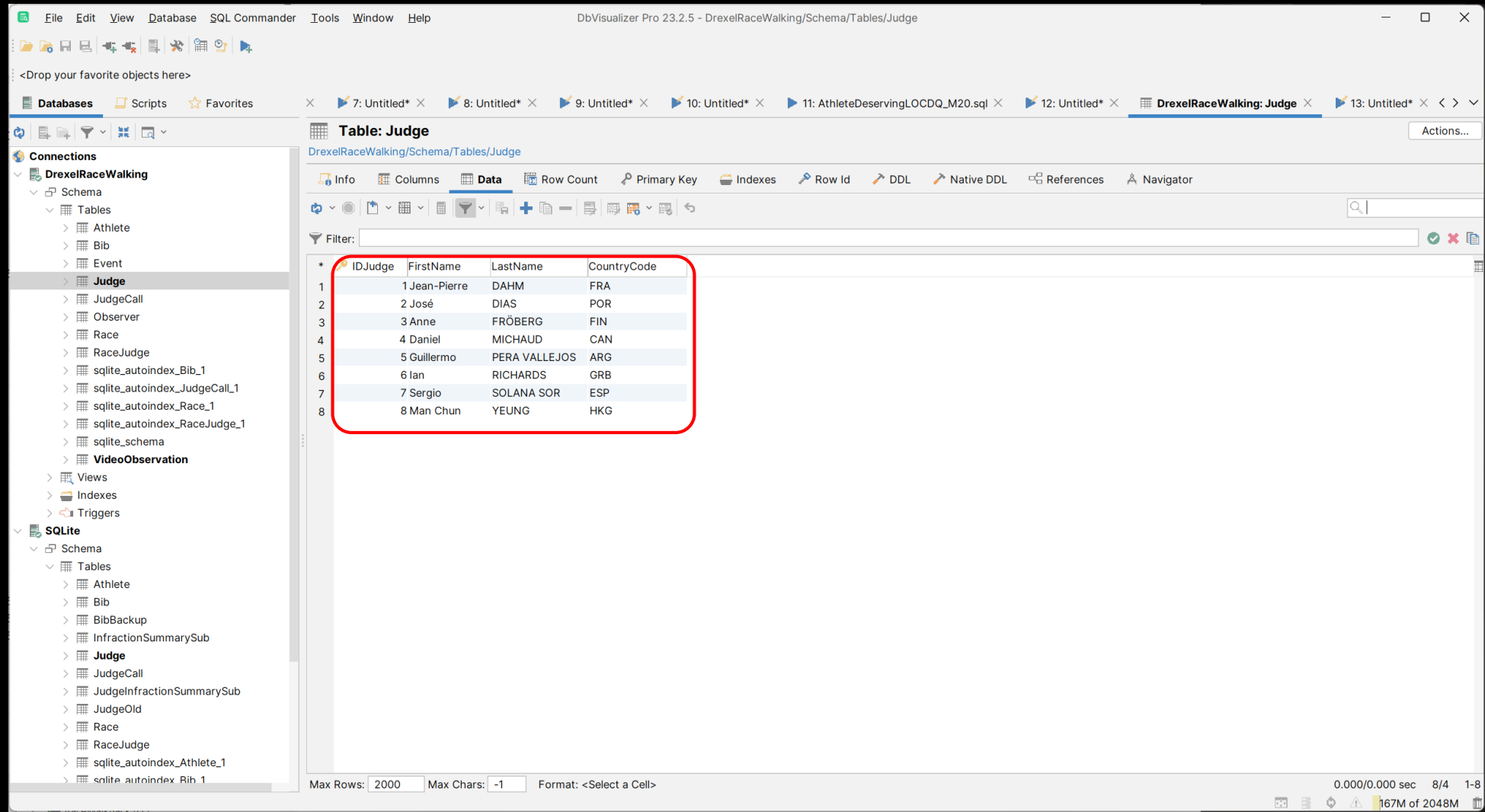
Inserting Data – Via Code

Hit the data *Refresh* button:



Inserting Data – Via Code

The data appears:

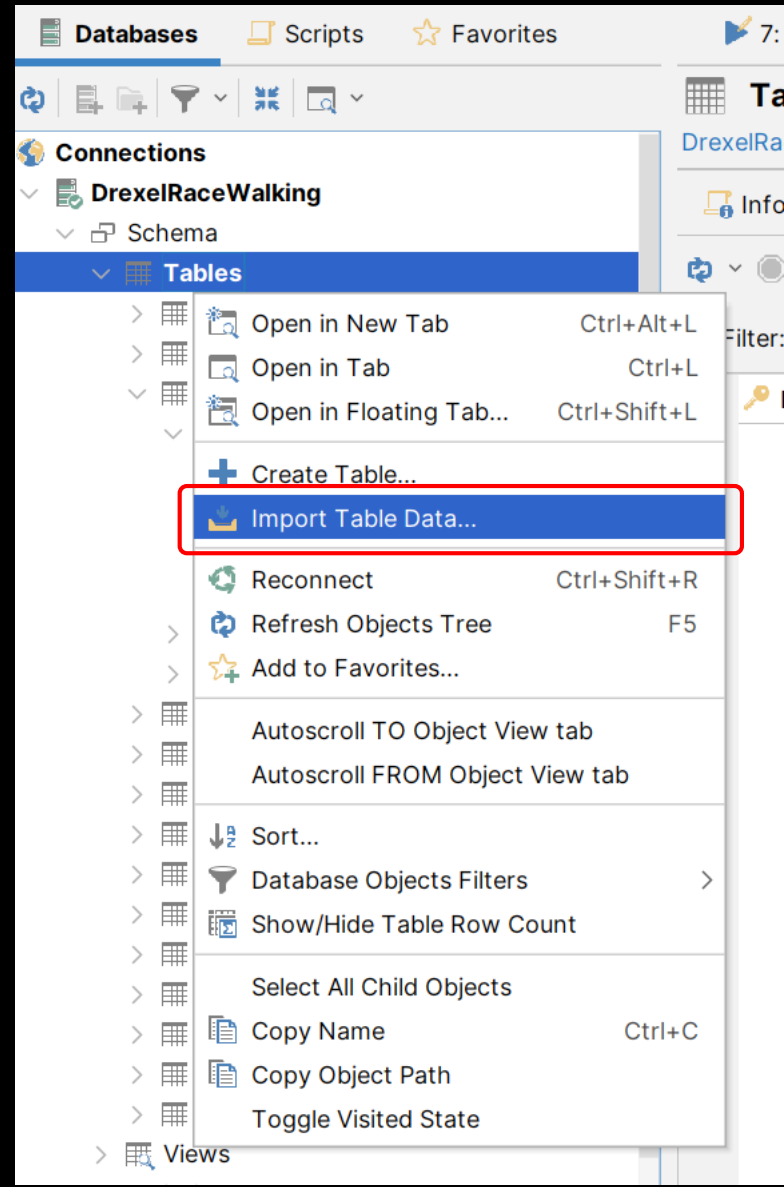


Create a Table from an Import

Shortcut the table creation process by creating it directly from an external data source, like a spreadsheet.

Right click on *Tables*

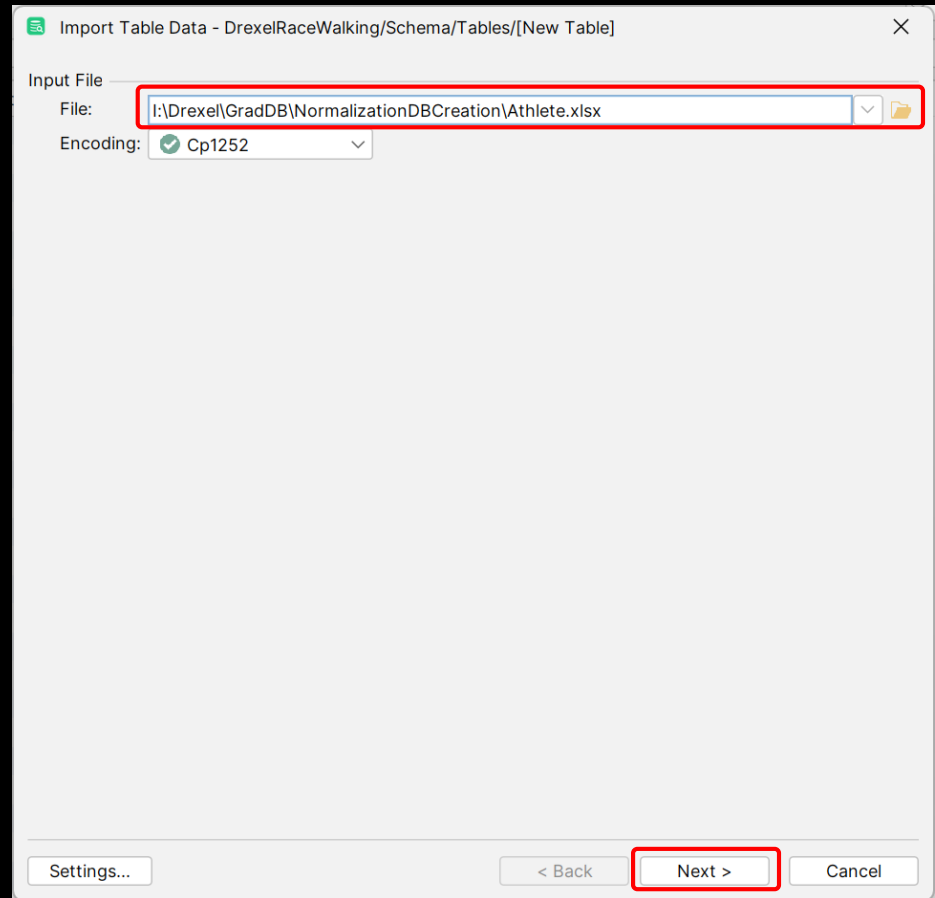
Click on *Import Table Data…* from the popup menu.



Create a Table from an Import

Select a file to import:

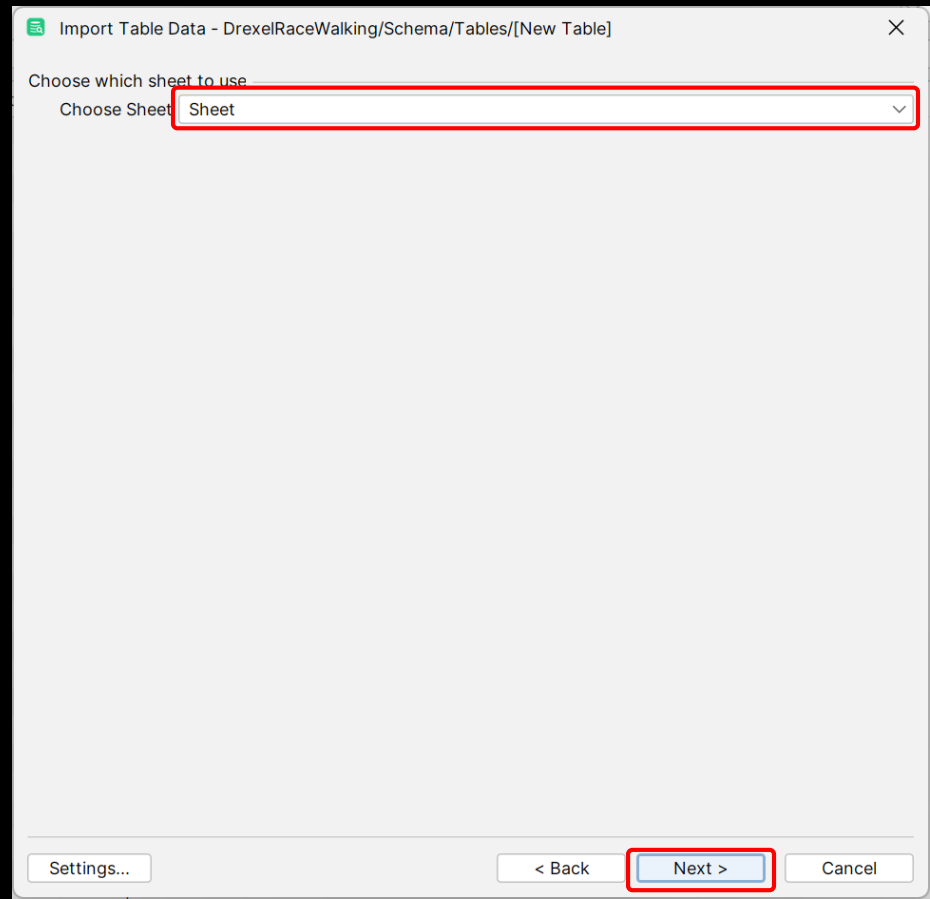
Then click the *Next* button:



Create a Table from an Import

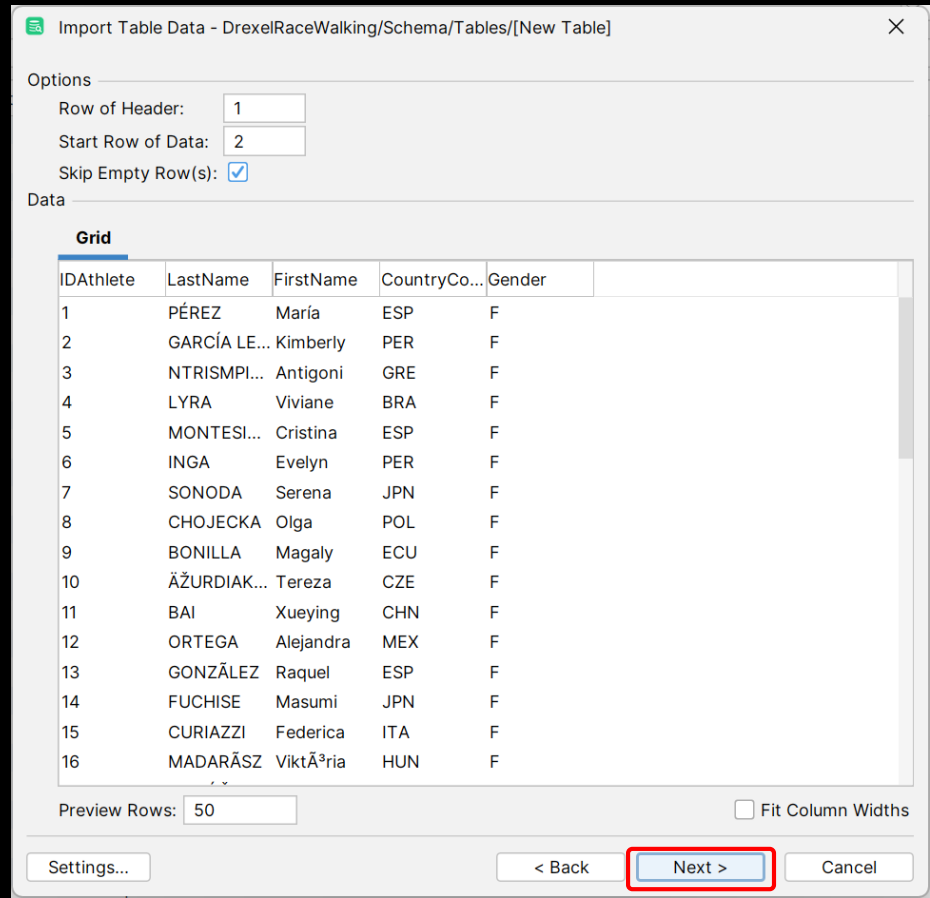
Since this is an Excel file, select the sheet the data is stored.

Then click the *Next* button:



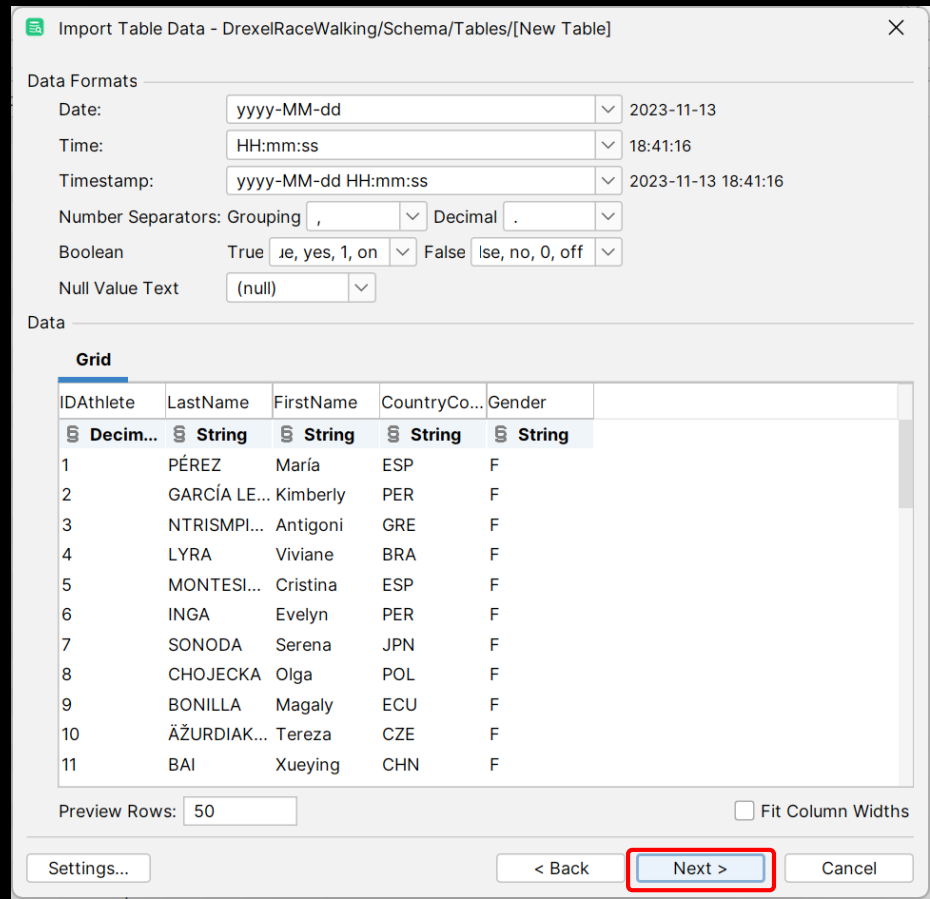
Create a Table from an Import

Assuming your data looks correct, click the *Next* button:



Create a Table from an Import

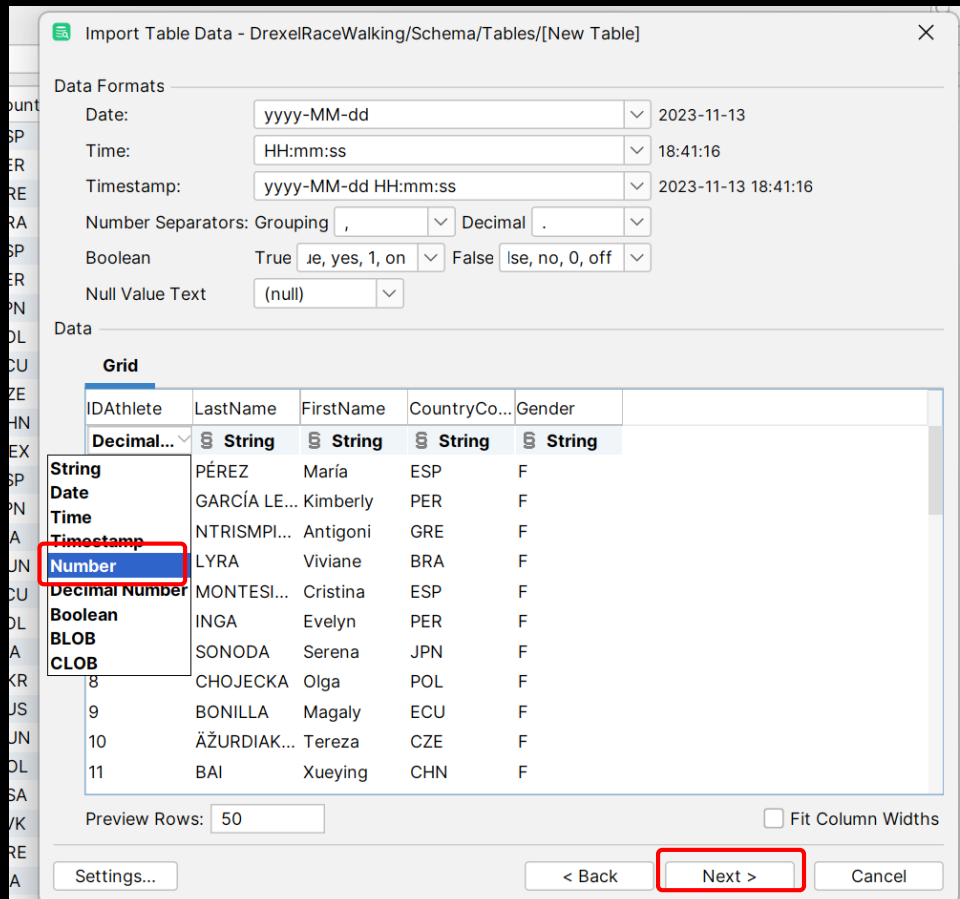
Assuming your data looks correct, click the *Next* button:



Create a Table from an Import

Change any data types DBVisualizer guesses wrong on:

Click the *Next* button:



Create a Table from an Import

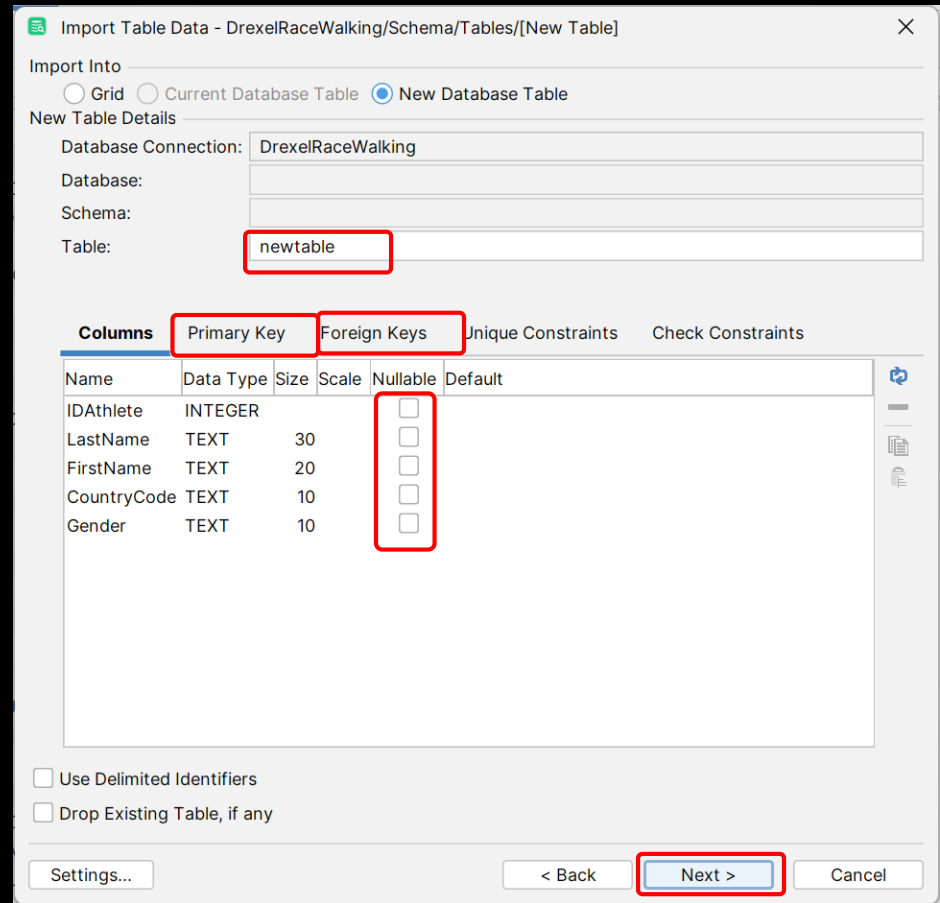
From here the process is the same as creating a table.

Give the table a name:

Indicate which fields are nullable:

Add any Primary and Foreign Keys:

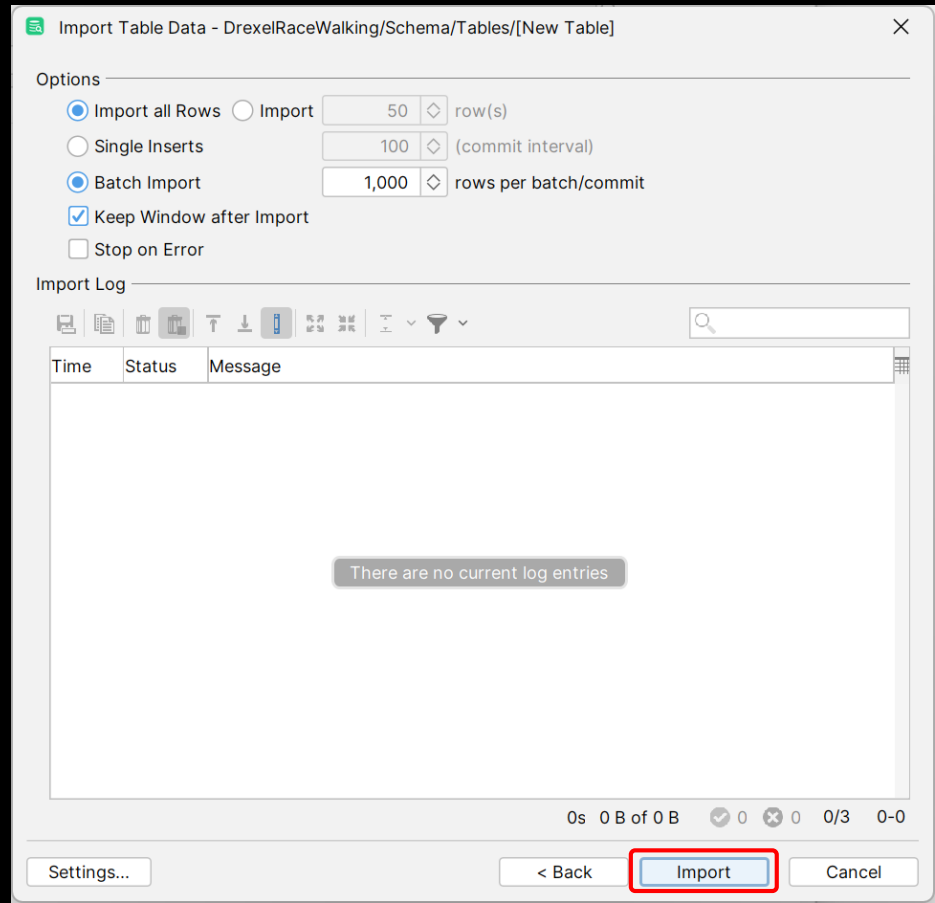
Click the *Next* button:



Create a Table from an Import

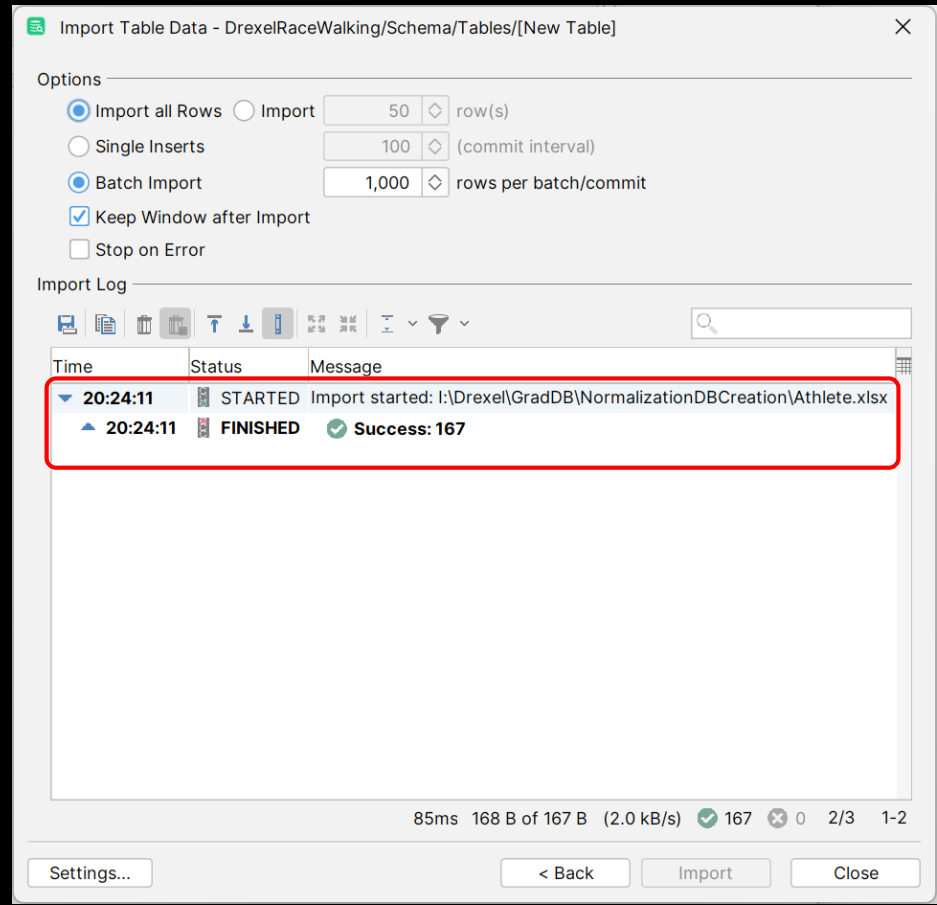
You are now ready to import the data:

Click the *Import* button:



Create a Table from an Import

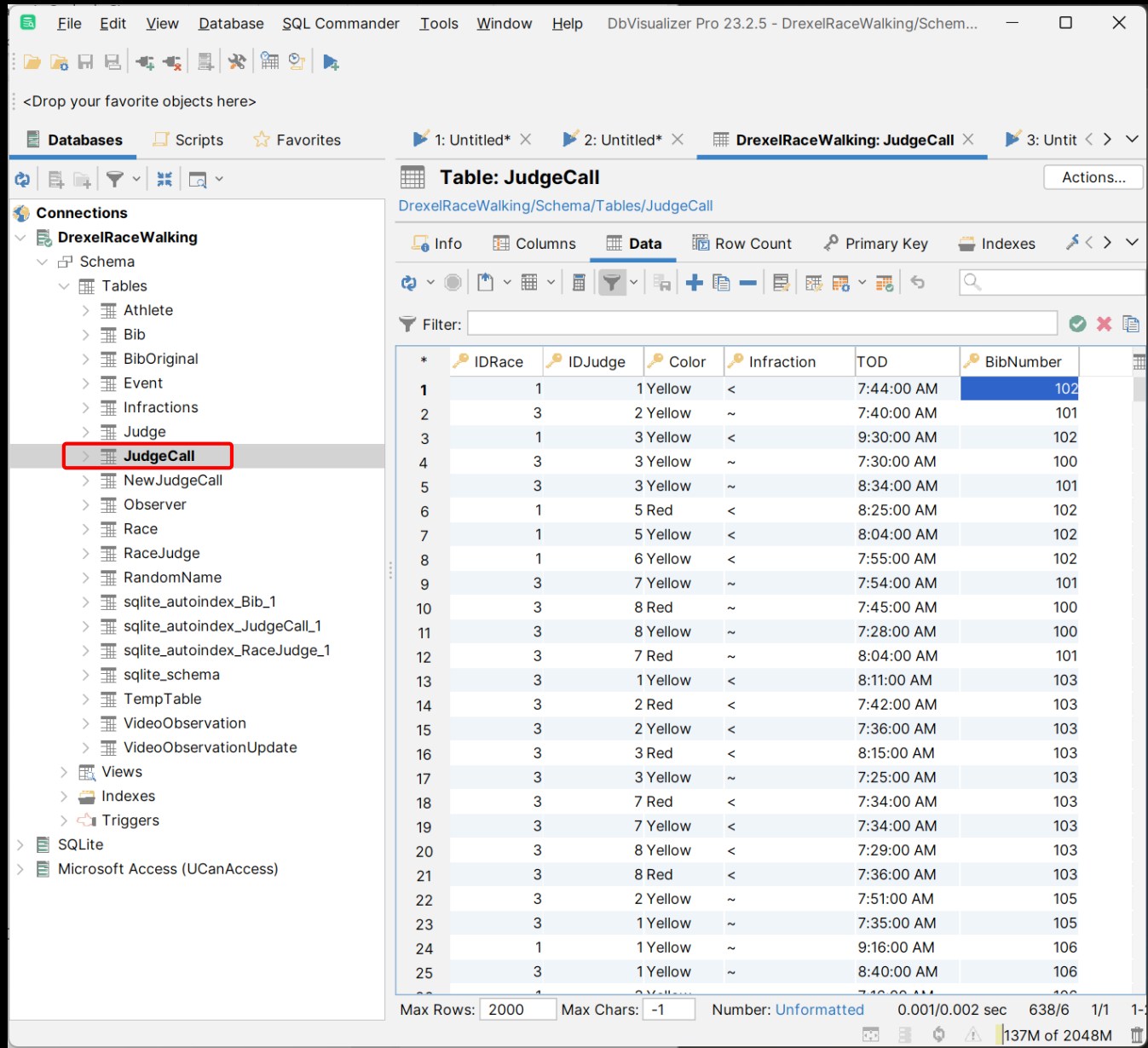
The status of the import is displayed:



Viewing Data within DQVisualizer

Data in a single table can be displayed by double clicking on the table name in the *Schema* tree.

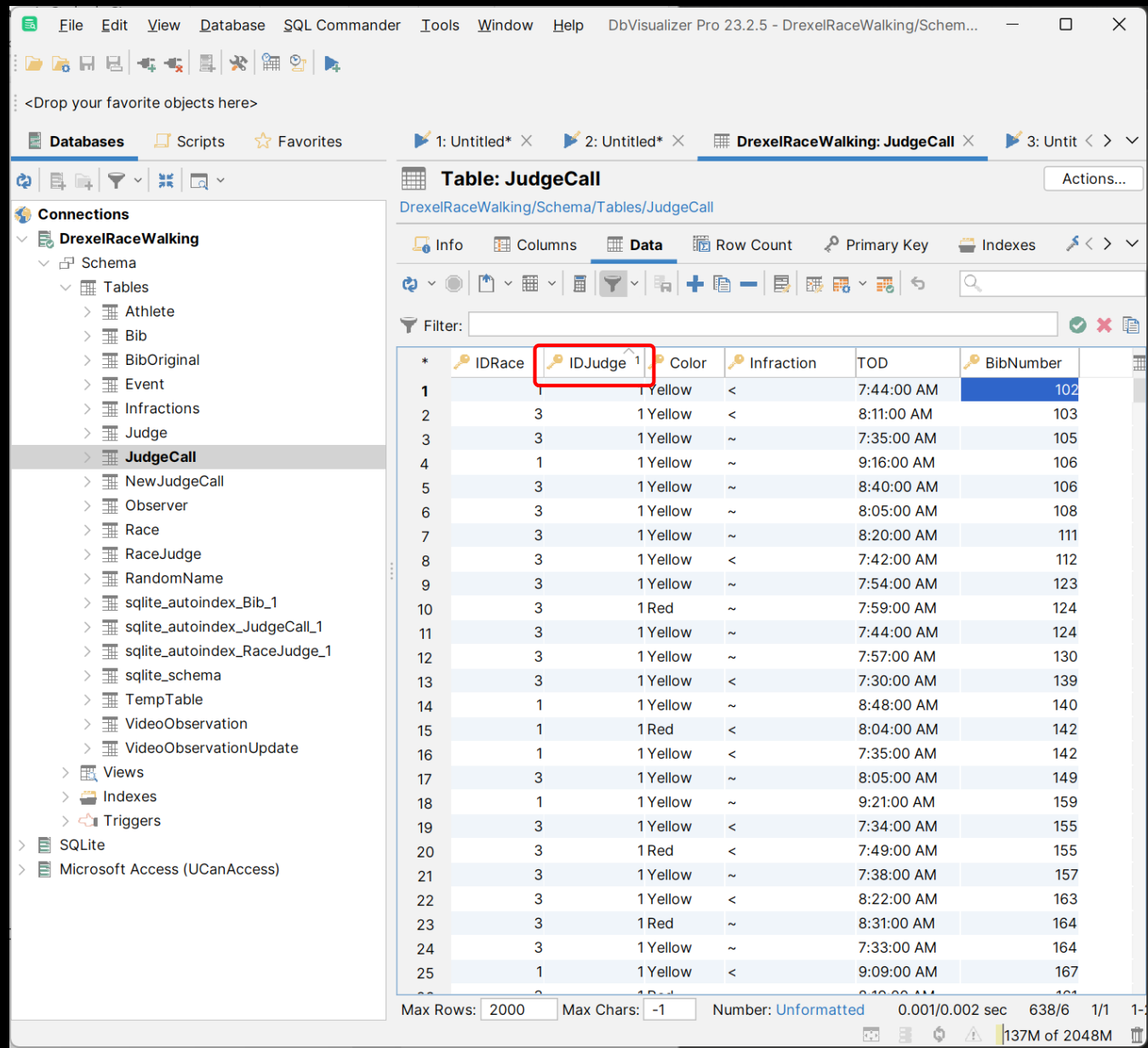
Observe viewing the *JudgeCall* table:



Viewing Data within DQVisualizer

Sort the data on a single field by clicking on the column header.

Observe the table sorted by *IDJudge*:

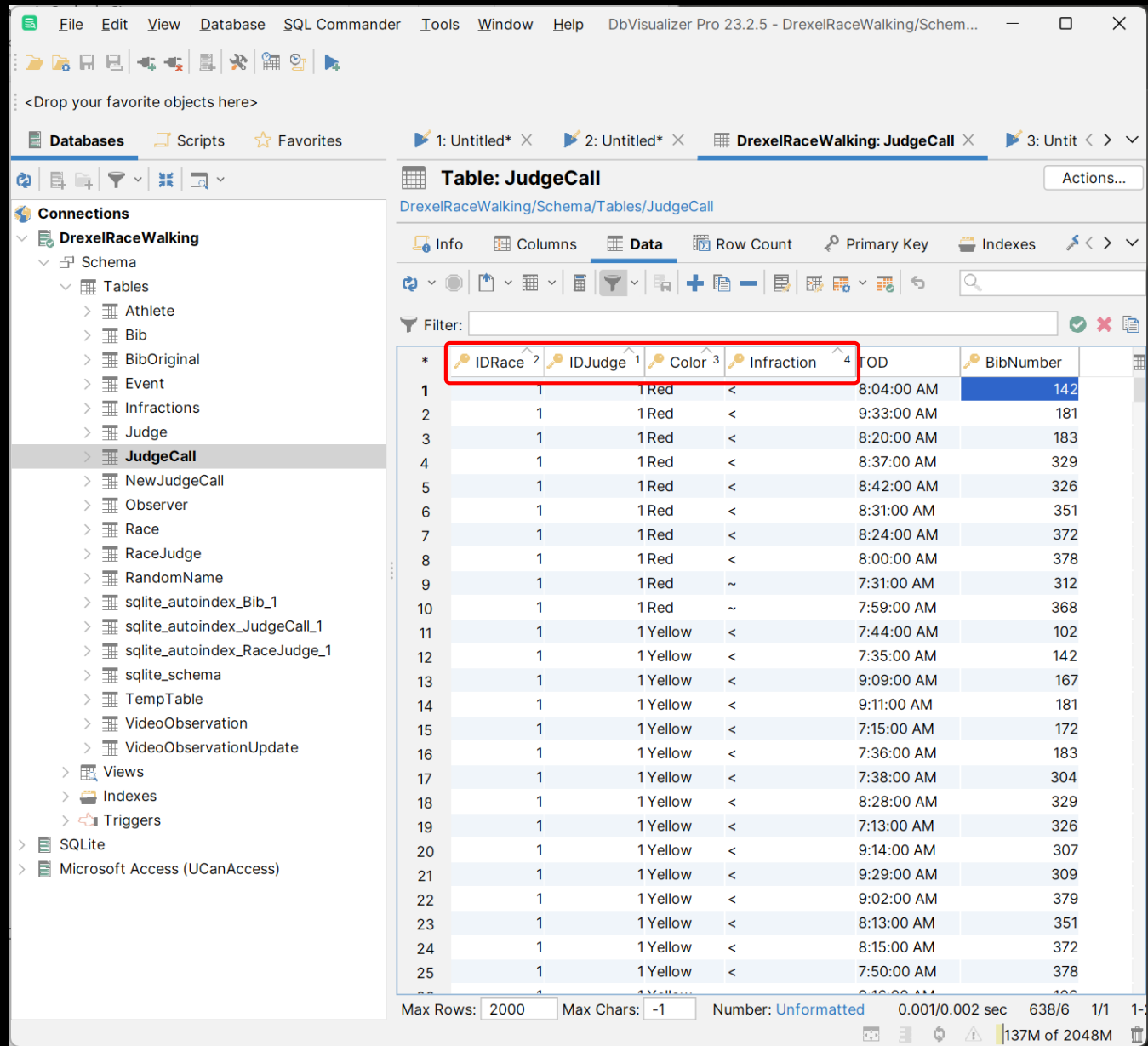


Viewing Data within DQVisualizer

Sort the data on more than a single field by holding down the *ctrl* key while clicking on the column headers (in order) of the fields to sort.

The order of the sort is shown with small superscript numbers next to the column names.

Observe the table sorted by *IDJudge*, *IDRace*, *Color* and then *Infraction*:

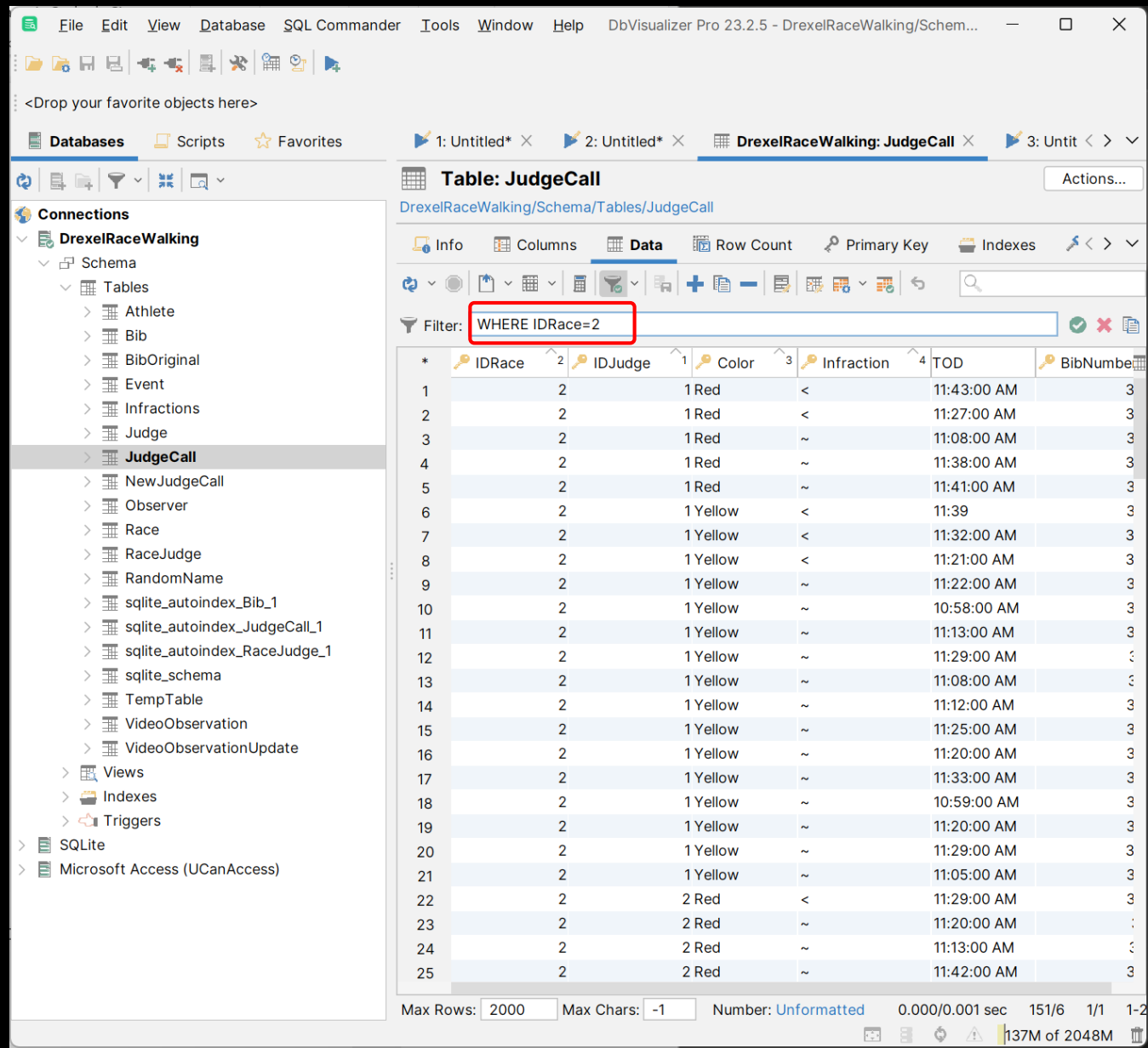


Viewing Data within DQVisualizer

Data can be filtered by adding conditions into the filter textbox.

Observe filtering for only `IDRace=2`:

Remember to click refresh after entering the filter data.

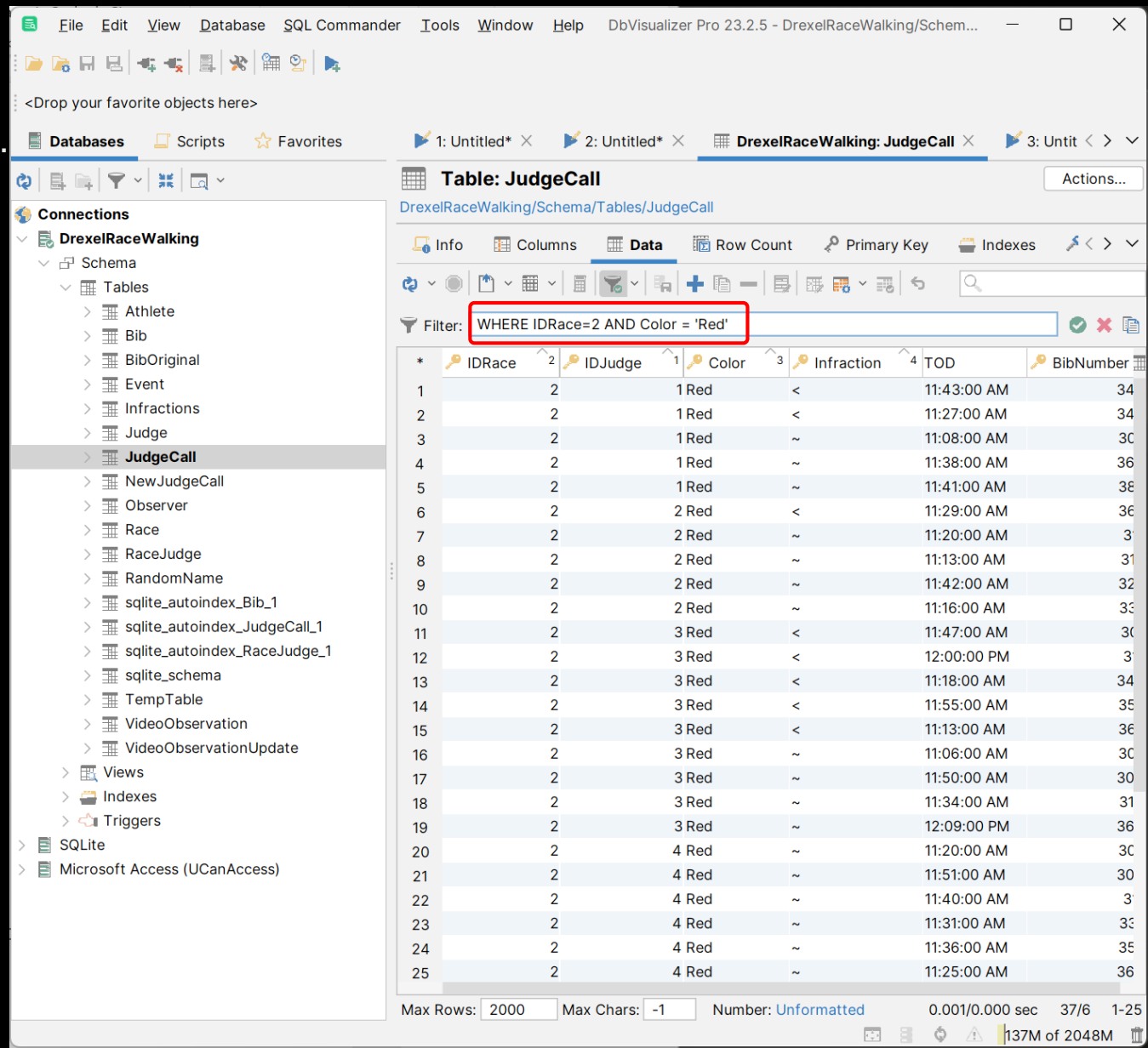


Viewing Data within DQVisualizer

Complex conditions are implemented with Boolean logic like `AND` / `OR`

Observe filtering for only `IDRace=2 AND Color = 'Red'`:

Note, remember to click refresh after entering the filter data.

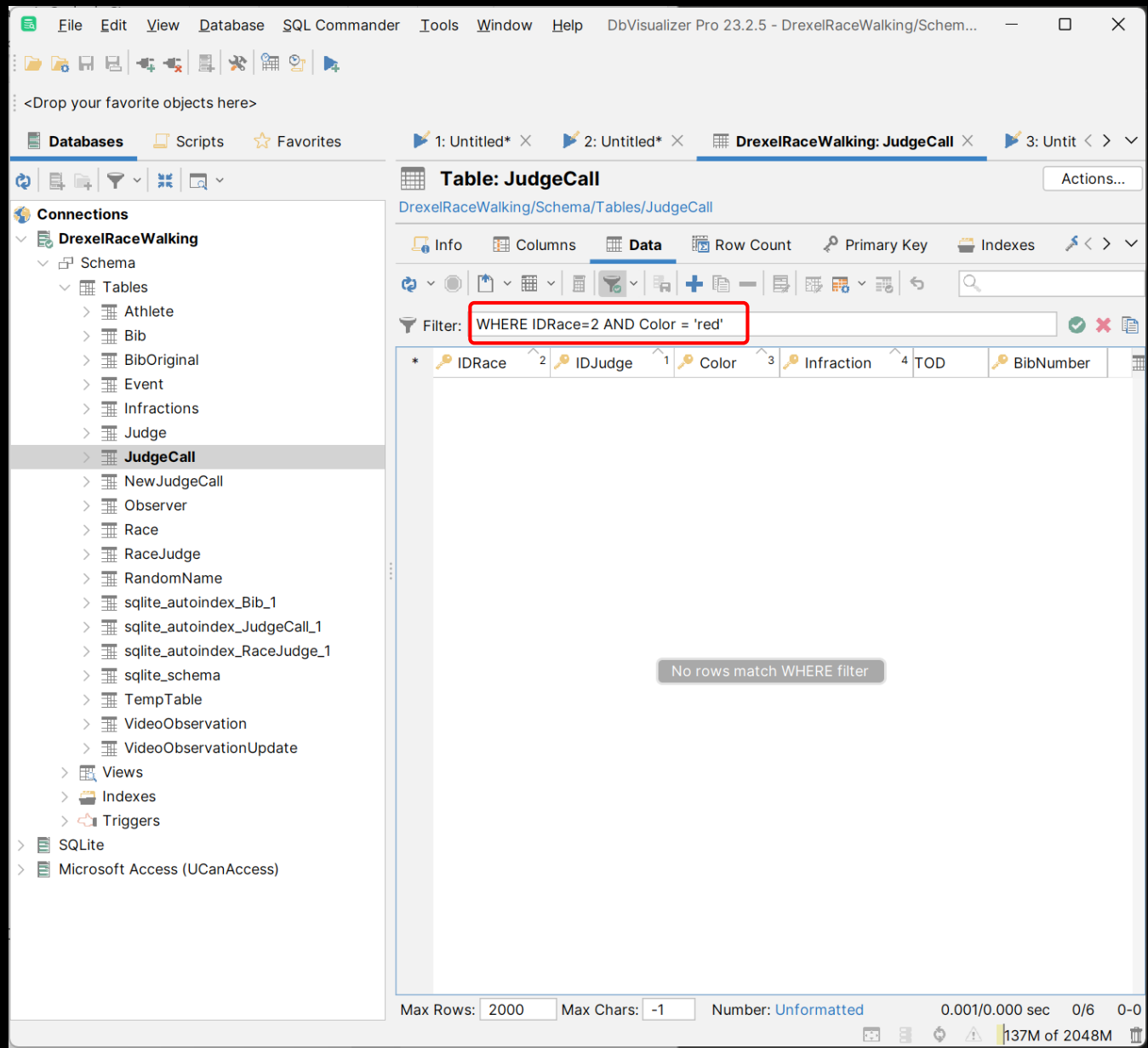


Viewing Data within DQVisualizer

SQLite is case sensitive when comparing data.

Not all databases are this way.

Observe filtering for `IDRace=2 and Color = 'red'`:



Autoincrement

Most databases have an option to create an integer field that is inserted automatically as a unique identifier.

Typically, the field is tagged with a qualifier like AUTO_INCREMENT.

This is shown in the following example:

```
CREATE TABLE Car (
    IDCar int NOT NULL AUTO_INCREMENT,
    Make varchar(255) NOT NULL,
    Model varchar(255),
    Year int,
    PRIMARY KEY (IDCar)
);
```

SQLite doesn't support this.

Autoincrement

SQLite does allow for auto increment, just by default.

To illicit the autoincrement behavior:

- set the auto increment field as the primary key
- this requires the field to not all nulls
- insert all the fields except the primary key

Observe the following table:

Observer		
IDObserver	**FirstName**	**LastName**
1	Jeff	Salvage
2	David	Harriman
3	JeffDavid	SalvageHarriman
4	Gary	Westerfield
5	JeffGary	SalvageWesterfield

IDObserver is the primary key.

To insert values and get the auto increment behavior, execute the following:

```
INSERT INTO OBSERVER (FirstName, LastName)
VALUES ('Shmi', 'Skywalker')
```

Notice `IDObserver` is left off.

Autoincrement

```
INSERT INTO OBSERVER (FirstName, LastName) VALUES ('Shmi', 'Skywalker')
```

Observer		
IDObserver	**FirstName**	**LastName**
1	Jeff	Salvage
2	David	Harriman
3	JeffDavid	SalvageHarriman
4	Gary	Westerfield
5	JeffGary	SalvageWesterfield

Observer		
IDObserver	**FirstName**	**LastName**
1	Jeff	Salvage
2	David	Harriman
3	JeffDavid	SalvageHarriman
4	Gary	Westerfield
5	JeffGary	SalvageWesterfield
6	Shmi	Skywalker

The 6 is inserted automatically and guaranteed to be unique.

Inserting Data – Via Code

Assignment:

Install SQLite.
Install DBVisualizer Pro.
Create the entire database explained here.
This includes all the Primary and Foreign keys.
Save all the SQL statements required to create your table.
Load all the data for each table.
Manually enter the *Event* and *Race* tables.
Load from the given spreadsheets the data for the tables: *Athlete*, *Bib*, *Judge*, *JudgeCall*, *Observer*, *RaceJudge*, and *VideoObservation*.

Let's anonymize the data, use the following for Event and Race.

Event

IDEvent	Event	City	Country
1	Boonta Eve Classic	Mos Espa	Tatooine

Race

IDRace	IDEvent	RaceDate	StartTime	Distance	DistanceUnits	Gender
1	1	2023-08-24	7:00 AM	35	km	MF
2	1	2023-08-19	10:50 AM	20	km	M
3	1	2023-08-20	7:15 AM	20	km	F