Università Politecnica Delle Marche

Tirocinio

Polimarche Racing Team

Nome: Alessandro Seghini
Dipartimento di riferimento: DII
Corso di riferimento : Programmazione Mobile

Sommario

Il tirocinio consiste nello sviluppo di un'applicazione mobile per il team Polimarche del nostro ateneo che consentirà di visualizzare le prestazioni e le parti tecniche, tra cui il setup, della macchina sviluppata dal team stesso.

Contents

List of Figures

Chapter 1

Introduzione

Il progetto consiste nello sviluppo di un'applicazione mobile per i componenti del team del Polimarche utile ad analizzare dati, visualizzare statistiche e altre informazioni riguardo i test e la parte puramente tecnica della monoposto.

Per maggiore chiarezza in questa parte iniziale viene riportata parte dell'intervista a termine di questo capitolo che riguarda formalmente l'ambito di sviluppo dell'applicazione.

Inizialmente verrà progettato un nuovo database partendo da un modello pre-esistente sviluppato come progetto di **Basi di dati** poiché a seguito del nuovo regolamento della formula SAE[1], la monoposto, a partire dal 2023, deve essere completamente elettrica, mentre il vecchio database era stato progettato per una macchina con motore a combustione.

Telemetria

Domanda: "Come vengono raccolti i dati ottenuti tramite i sensori della monoposto?"

Direttore Tecnico: "I dati ottenuti dai sensori vengono in inizialmente istanziati all'interno di un computer presente sulla monoposto. Per visualizzarli, invece, viene raccolta fisicamente la memoria che contiene i vari dati sotto forma di un file CSV."

La struttura dei setup e test

Domanda: "Che tipo di altri dati vengono salvati durante un test? Che

[1] Competizione tra varie università europee che consiste nella progettazione e nella realizzazione di una monoposto atta a gareggiare con le altre macchine sviluppate dagli altri partecipanti.

struttura hanno?"

Direttore Tecnico: "I dati dei test vengono registrati con:

- tracciato in cui è stato svolto
- lunghezza del tracciato
- data e ora in cui è stato svolto
- tipo di test effettuato
- condizioni del tracciato e del meteo
- setup della macchina
- tempistiche dei singoli settori per giro
- pilota che ha effettuato il giro.

Il tipo di test effettuato fa intendere che esistano 4 diverse tipologie di prove che vengono fatte. Queste sono rispettivamente:

- Autocross
- Skidpad
- Acceleration
- Endurance."

Altre informazioni sui test e setup

Domanda: "Ci sono altri aspetti non ancora nominati che potrebbero essere utili in merito ai test?"

Direttore Tecnico: "Sarebbe comodo raccogliere i dati e i commenti dei singoli piloti alla fine di ogni test, oltre ai dati biometrici dei singoli. Inoltre sarebbe importante per noi capire se, alla fine della sessione, si è rotto qualcosa e la causa di tale rottura. Così facendo in futuro non si commetteranno gli stessi errori essendone già a conoscenza."

Chapter 2

Sviluppo Database

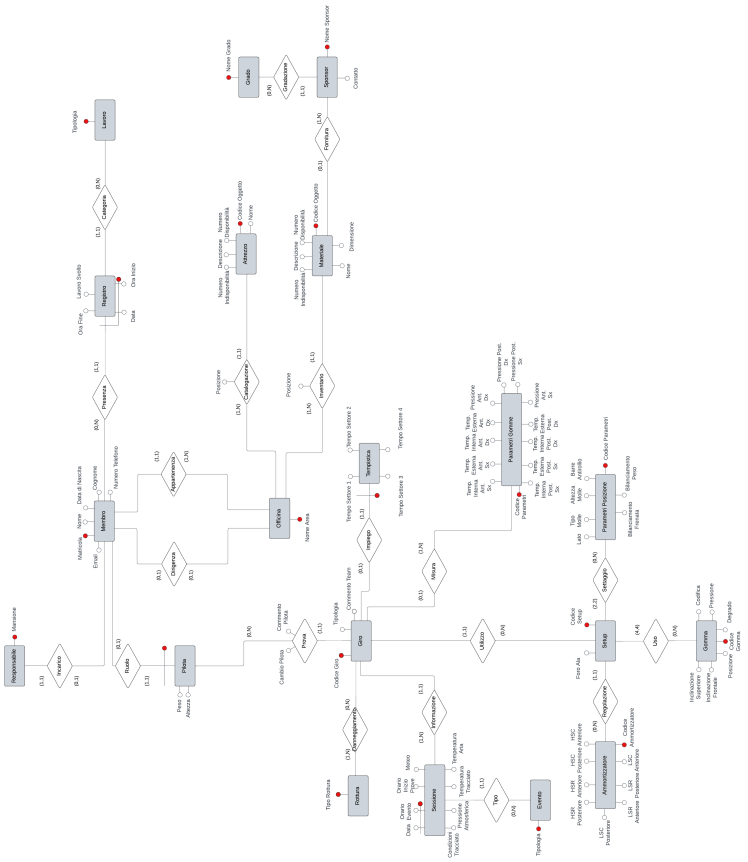

Figure 2.1: Modello logico iniziale

Il database che si fonda sul modello logico sopra presentato verrà ristrutturato dalla zona sotto all'entità **Giro**.

Il lavoro deve essere svolto perché, a seguito del nuovo regolamento della formula Sae, dall'anno 2023 le macchine costruite hanno l'obbligo di essere completamente elettriche. Ciò rende necessario un adattamento del database già esistente alla nuova norma.

Nello specifico la parte da ristrutturare risulta la seguente.

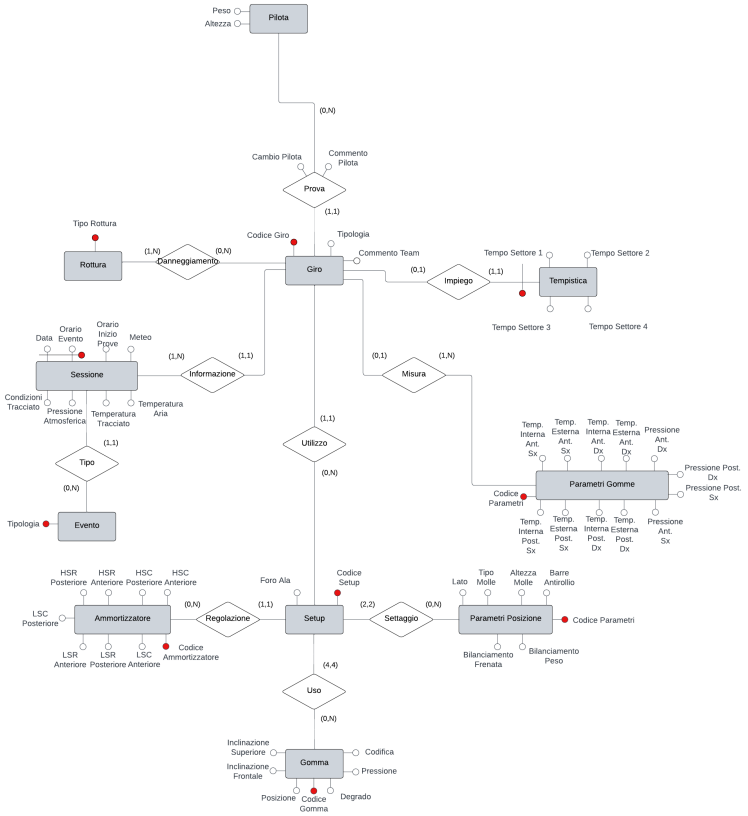

Figure 2.2: Modello da modificare

2.1 Telemetria

Una prima modifica consiste nell'aggiunta al database una parte che consenta la raccolta dei dati, ottenuti da tutti i sensori utilizzati nella monoposto, per effettuare statistiche o calcolare dei parametri.

I vari sensori permettono di misurare:

- temperatura;
- pressione;
- carico;
- velocità;
- posizione;
- corrente;
- voltaggio.

Per ognuna di queste categorie vengono disposte più tipologie di sensori in modo tale da misurare la stessa grandezza fisica in più zone della macchina.

Inizialmente quindi si sono determinate le entità principali del nuovo schema riguardanti la telemetria.

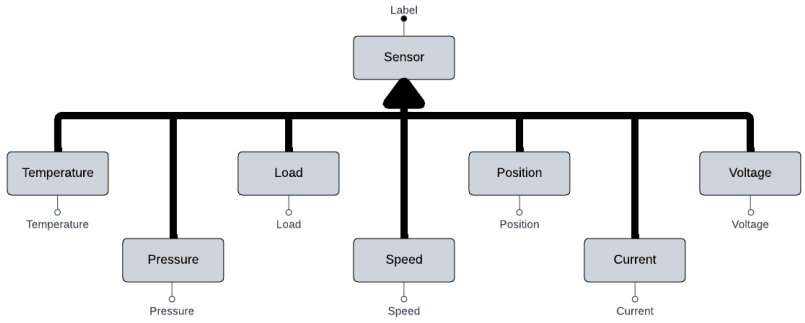

Figure 2.3: Telemetria generale

Abbozzato un primo schema qualitativo, bisogna distinguere i sensori in base alle componenti meccaniche, liquide o fisiche della monoposto su cui determinano la grandezza fisica di riferimento. Ad esempio la temperatura è letta da più sensori su più parti della vettura come, per esempio, il motore o sui vari freni.

Allora per ogni tipo di sensore vengono sviluppate nuove entità che conterranno tutti i dati relativi ad una singola componente.

2.1.1 Temperatura

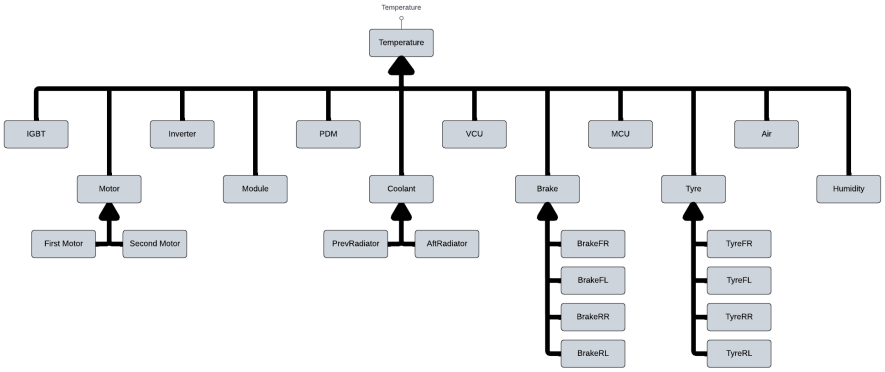

Figure 2.4: Sensori temperatura

2.1.2 Pressione

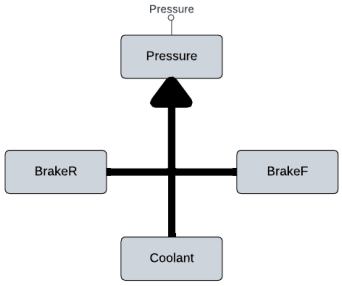

Figure 2.5: Sensori pressione

2.1.3 Carico

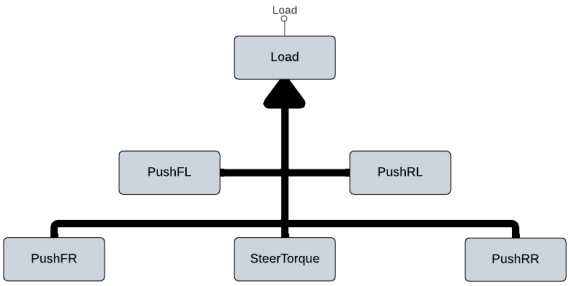

Figure 2.6: Sensori carico

2.1.4 Velocità

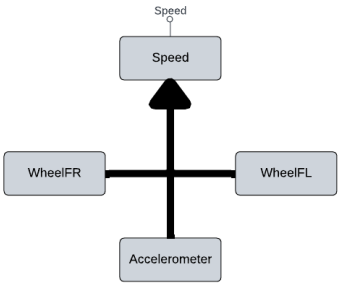

Figure 2.7: Sensori velocità

2.1.5 Posizione

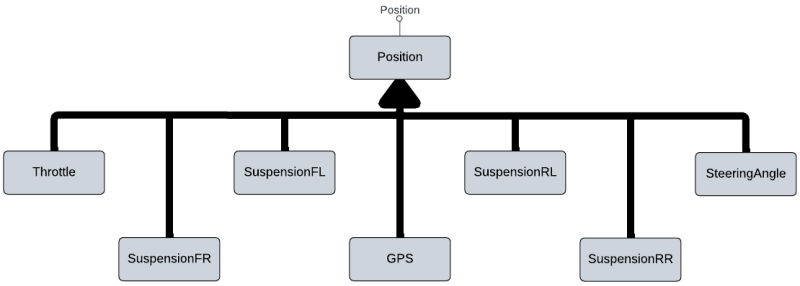

Figure 2.8: Sensori posizione

2.1.6 Corrente

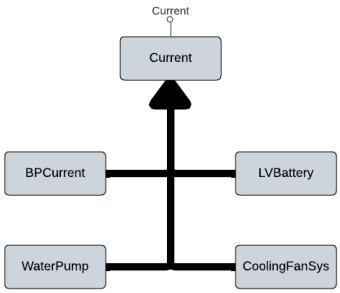

Figure 2.9: Sensori corrente

2.1.7 Voltaggio

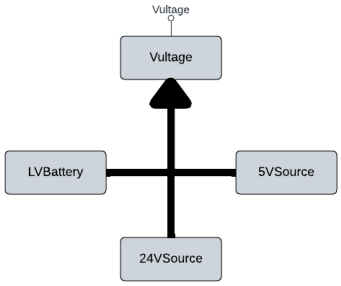

Figure 2.10: Sensori voltaggio

2.1.8 Modifica identificatore

Definite le nuove entità, analizzando l'identificatore primario dell'entità madre di tutte le generalizzazioni, cioè **label** del costrutto **Sensor**, per praticità e correttezza viene inserito un nuovo identificatore, presente in ogni entità figlia, denominato con il nome della grandezza fisica che quel determinato sensore misura.

In questo modo all'interno di ogni sotto-entità saranno presenti solo i dati ottenuti tramite il sensore specifico.

Allora la presenza dell'identificatore **label** diventa superflua e, dunque, si può eliminare.

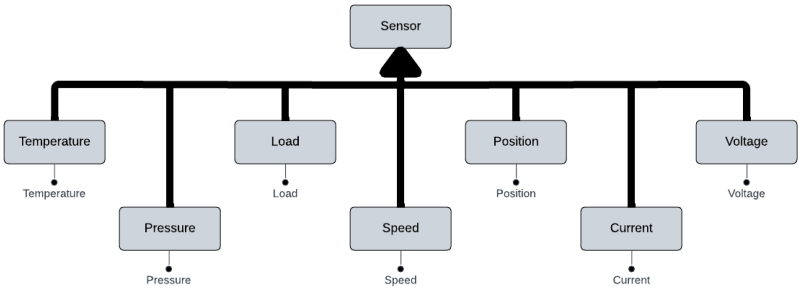

Figure 2.11: Telemetria generale completa

2.2 Modifica concetto Setup

L'entità **Setup** nonostante fosse già presente nel database precedente, viene ristrutturata per favorire sia una nuova relazione con l'entità **Sensor**, sviluppata nella sezione precedente, sia una migliore leggibilità del diagramma stesso e dei dati contenuti all'interno di esso.

La prima modifica consiste nell'aggiungere una nuova entità **Problem** al cui interno saranno contenuti tutti i problemi che i vari setup riscontrano.

Una seconda invece riguarda la percezione di una nuova entità denominata **Suspension** che diventa madre di una generalizzazione, le cui figlie risultano **Spring** e **Damper**. Ciò rende possibile la distinzione tra le 2 parti meccaniche presenti nella vettura che, nella versione precedente del database, non era consentita.

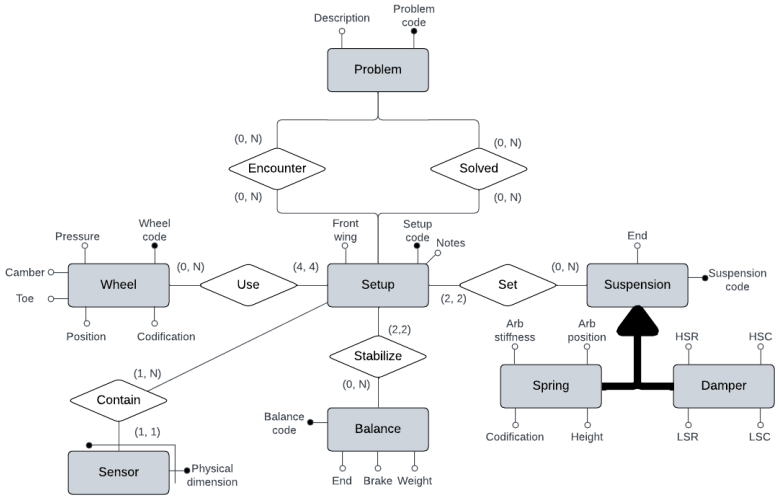

Figure 2.12: Modifica concetto setup

Un particolare da notare è che l'entità **Sensor** rappresenta per intero il diagramma ad albero generato dalla concatenazione delle varie generalizzazioni analizzato nella sezione della *Telemetria*.

Inoltre, sempre per quanto riguarda l'entità **Sensor**, è presente un identificatore esterno perché tutti i dati raccolti dai vari sensori durante le prove vengono collegati direttamente al setup utilizzato con cui, insieme, formano una chiave.

2.3 Modifica concetto Sessione

Per quanto riguarda il concetto di **Sessione** si è scelto di eliminare l'entità **Evento** presente nel vecchio modello del database e immettere all'interno di **Practice Session** un nuovo attributo, denominato Event type, che con la data, l'orario di inizio e di fine, formasse l'identificatore primario.

Inoltre è stata creata un'altra entità, chiamata **Circuit**, per contenere tutti i tracciati su cui è possibile provare la monoposto.

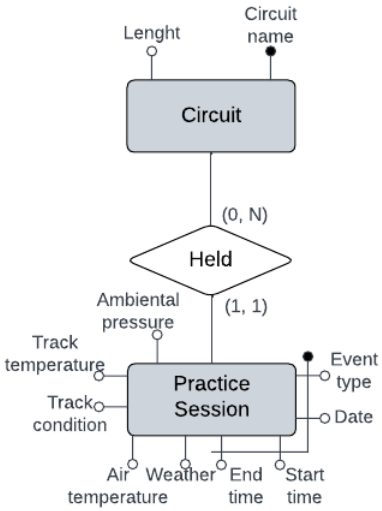

Figure 2.13: Modifica concetto sessione

Una volta sviluppato il concetto di sessione nella sua interezza, bisogna anche metterlo in relazione con i vari piloti e con i giri che vengono completati durante una singola prova. Mettendo tutto in un unico diagramma si ottiene il seguente blocco definito da entità e relazioni.

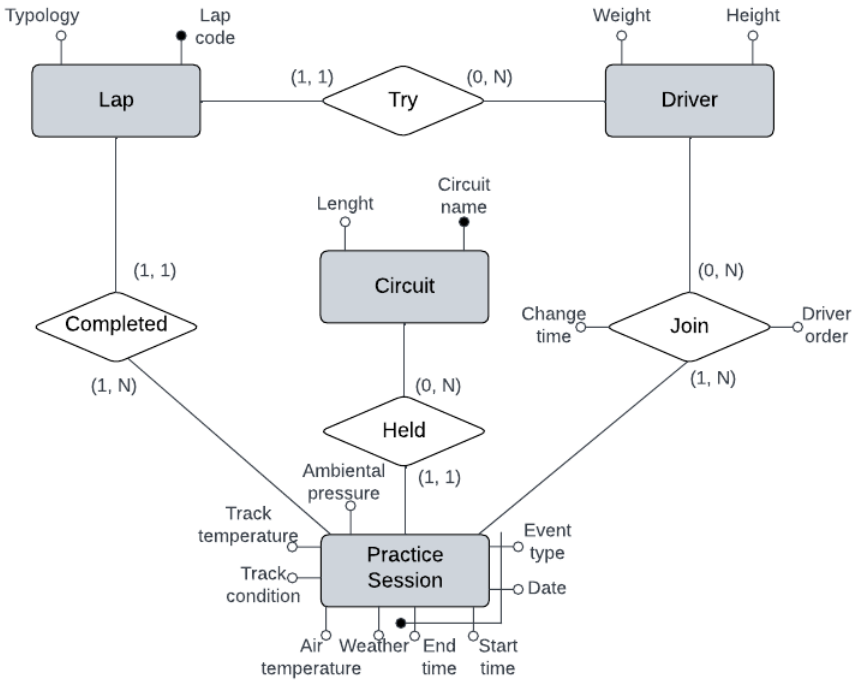

Figure 2.14: Completamento concetto sessione

Da come si può vedere sull'entità **Driver** non viene definito nessun identificatore primario. Ciò è voluto perché lo sviluppo verrà illustrato in una sezione successiva.

2.4 Modifica concetto Giro

Anche al concetto di **Giro**, centrale in questo database, deve essere attuata una modifica.

Per prima sono state aggiunte più entità tra cui troviamo **Comment**, che serve a tenere traccia dei commenti fatti dal pilota o dal team su un determinato giro, e **Driver Error**, che invece conserva al suo interno tutti gli errori dei piloti che hanno portato alla rottura o al danneggiamento del veicolo.

In definitiva, lo sviluppo del concetto **Giro** risulta il seguente.

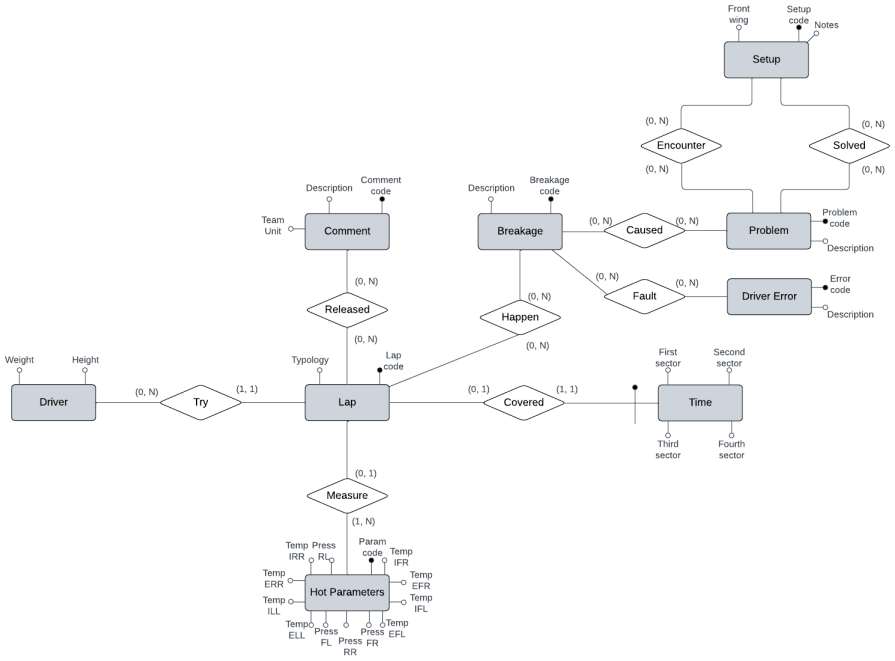

Figure 2.15: Modifica concetto giro

2.5 Diagramma iniziale completo

Una volta che tutte le parti sono state modificate si può ricostruire un diagramma di base che consenta di visualizzare[1] il contenuto che il database conterrà nella maniera più chiara possibile.

Viene ricordato nuovamente che il concetto dei Piloti verrà sviluppato in una sezione successiva e di conseguenza non ha ancora presente un identificatore.

Inoltre, l'identificatore presente all'interno dell'entità **Sensor**, corrisponde all'identificatore presente in ogni entità figlia e cioè la grandezza fisica di riferimento che il sensore misura.

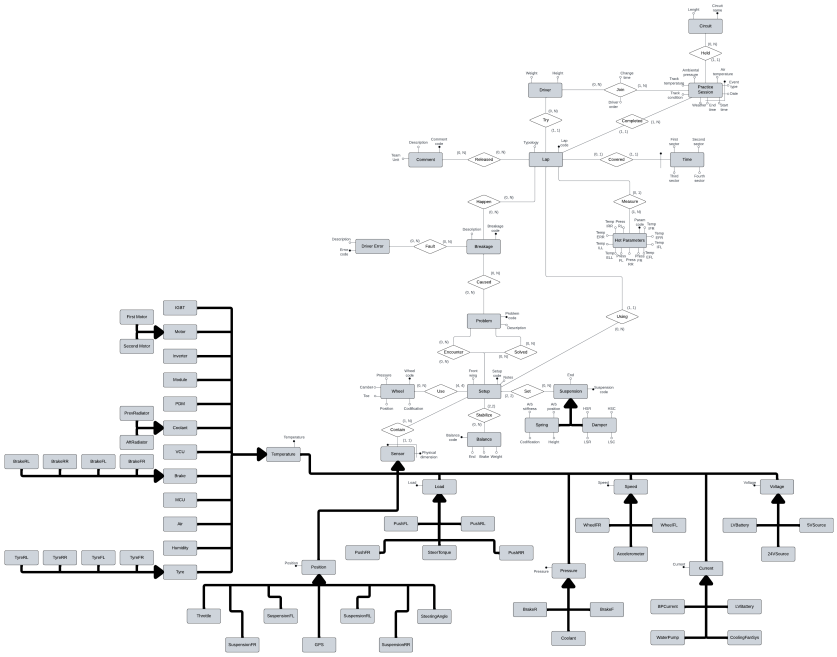

Figure 2.16: Diagramma iniziale completo

Alla fine di questo capitolo verranno spiegate tutte le entità e relazioni presenti, e, in caso di necessità, anche alcuni attributi.

[1] Per visualizzare lo schema è necessario ingrandire il pdf.

2.6 Eliminazione gerarchie

2.6.1 Sensori

In questa sezione vengono analizzate tutte le varie gerarchie presenti nel modello creato per eliminarle in seguito.

In primo piano si studiano quelle associate ai vari sensori della monoposto.

L'entità madre di tutti i sensori risulta **Sensor**. Nonostante questa venga eliminata per questioni legate al mantenimento di tutti i tipi di sensori presenti nella monoposto, per questioni di lettura del diagramma stesso viene mantenuta. Ciò che cambia rispetto al caso in cui **Sensor** rappresenta l'entità madre è che da qui in poi essa assume un connotato generale e indica con esattezza ognuna delle varie entità che sono presenti alla fine di ogni singola generalizzazione.

Detto in altro parole, tutte le entità da cui parte almeno una generalizzazione vengono eliminate e si mantengono tutte quelle che sono presenti nei rami finali. Ad esempio, per quanto riguarda i sensori atti a misurare le temperatura, vengono mantenute le entità **IGBT**, **BrakeFR**, **BrakeFL** etc..

L'identificatore diventa dunque quello già presente all'interno dell'entità **Sensor**, che ricordo rappresenti tutte le varie singole entità, che contiene la grandezza fisica che il sensore misura, unito ad un nuovo identificatore che però risulta esterno su **Setup** per garantire che ogni misura dei trasduttori sia collegata al setup di riferimento.

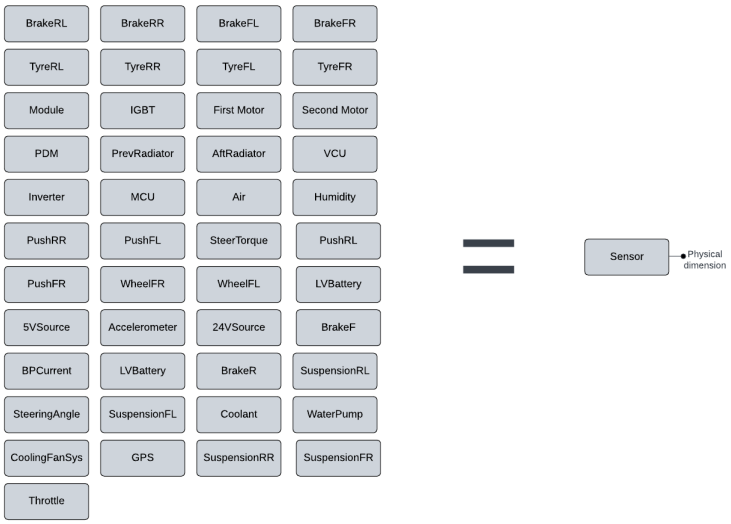

Figure 2.17: Eliminazione gerarchia sensori

Allora si può rimodellare la parte corrispettiva del diagramma E/R in questa maniera.

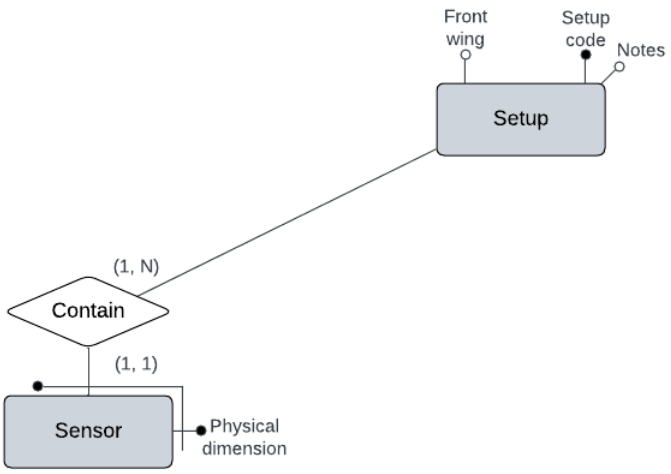

Figure 2.18: Ristrutturazione telemetria

2.6.2 Sospensioni

Per quanto riguarda invece la generalizzazione tra **Suspension** e **Spring** con **Damper** si decide di eliminare, anche in questo caso, l'entità madre per mantenere separati le 2 parti meccaniche perché sono di primaria importanza durante lo sviluppo della macchina.

Allora l'attributo End di **Suspension** viene ripartito su entrambe le entità figlie, così come l'identificatore.

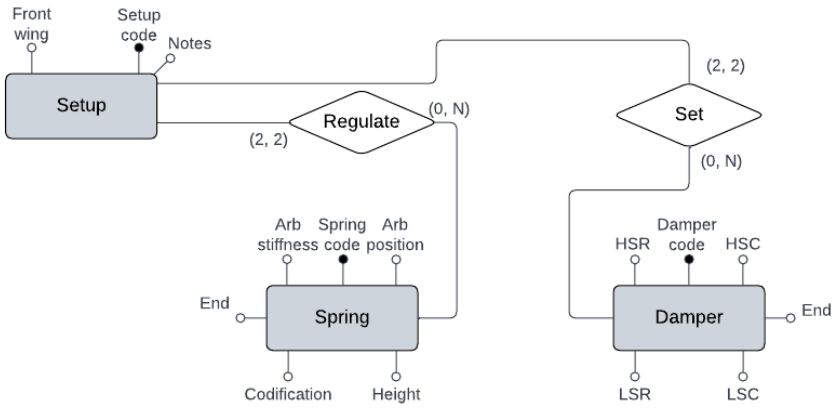

Figure 2.19: Eliminazione gerarchia sospensioni

2.7 Sviluppo pilota

In questa sezione viene sviluppata l'entità **Driver**. Dato che ogni pilota è un membro del team, per distinguere i 2 ruoli del gruppo, viene introdotta una relazione in cui, dal lato del pilota, è presente un identificatore esterno che consente di salvare all'interno dell'entità **Driver** la matricola del membro relativo.

Graficamente il modello risulta il seguente.

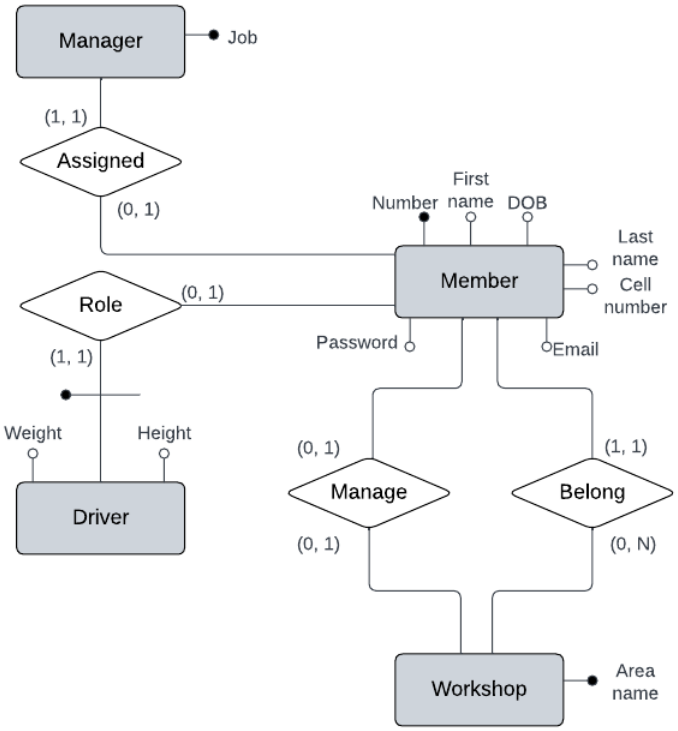

Figure 2.20: Sviluppo pilota

Come si può vedere dal modello, ogni membro è verificato da un username, che corrisponde alla propria matricola, e una password. In questo modo quando si utilizza un qualsiasi programma che utilizza dati presenti nel database sviluppato in questo documento, in base alla propria posizione gerarchica e al reparto di appartenenza si potranno effettuare e visualizzare interfacce e contenuti diversi.

2.8 Modello logico finale

Dopo aver eliminato le gerarchie nella sezione precedente e aver visto che il modello risulta in forma normale, si possono unire tutte le parti sviluppate singolarmente per ottenere il diagramma completo che serve a definire il database finale.

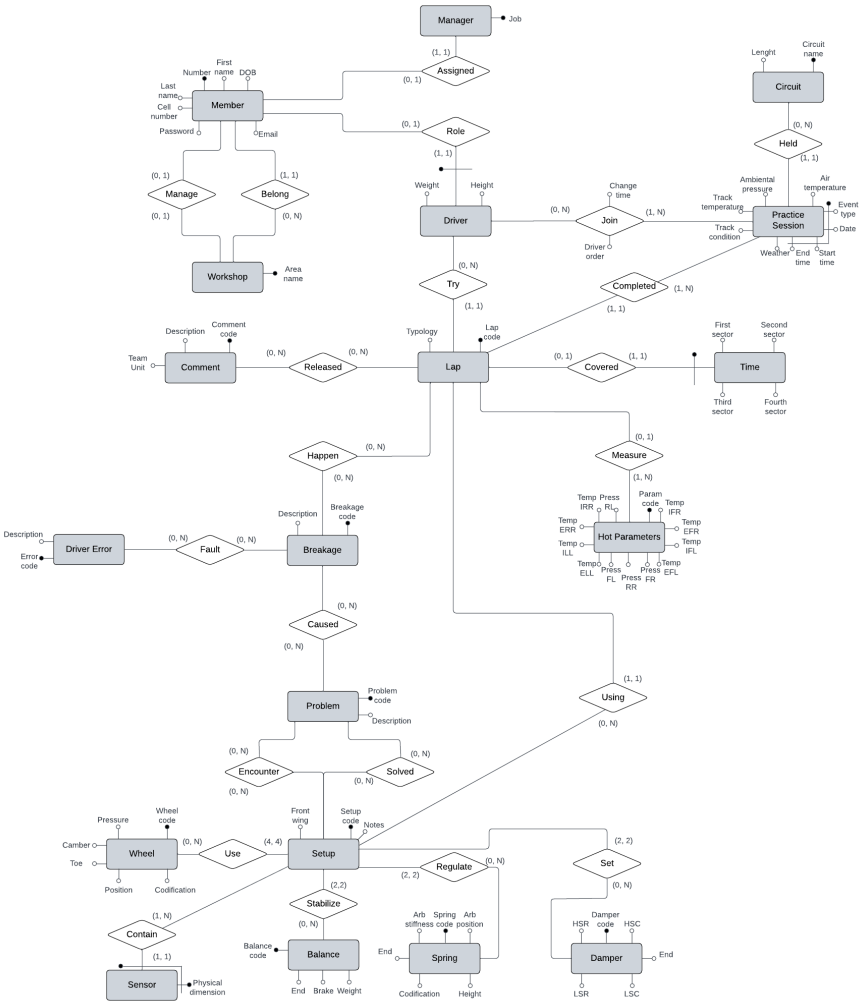

Figure 2.21: Modello logico finale completo

2.9 Glossario entità e relazioni

Balance	Entità che rappresenta il bilanciamento sia del peso sia del freno della monoposto. Anche in questo caso l'attributo End serve a specificare se ci si sta riferendo alla parte frontale o alla parte del retro della monoposto.
Breakage	Entità che rappresenta l'insieme di tutte le rotture e danni che possono essere stati subiti dalla monoposto.
Circuit	Entità che rappresenta l'insieme di tutti i vari circuiti su può essere svolta una sessione.
Comment	Entità che rappresenta l'insieme di tutti i commenti fatti su un determinato giro. L'attributo Team unit serve a specificare se il commento è stato fatto dal team o dal pilota.
Damper	Entità che rappresenta l'insieme di tutte le possibili combinazioni dei vari parametri degli ammortizzatori. L'attributo End serve a specificare se ci si sta riferendo alla parte frontale o alla parte del retro della monoposto.
Driver Error	Entità che rappresenta l'insieme di tutti gli errori del pilota che possono aver determinato una rottura o un danno alla monoposto.
Driver	Entità che rappresenta l'insieme di tutti i piloti del team.
Hot Parameters	Entità che rappresenta l'insieme di tutte le misurazioni di parametri delle gomme fatte a caldo durante i giri in cui la monoposto rientra ai box.

Join	Relazione che mette in associazione il pilota con le sessioni in cui scende in pista. L'attributo Driver order serve a determinare l'ordine dei vari piloti che hanno provato la monoposto. L'attributo Time change invece viene settato diverso dal valore nullo solamente nel momento in cui la sessione consiste in una prova di Endurance e il numero dei piloti che provano la monoposto sono più di due. Infatti in quella situazione viene salvato il tempo con cui due piloti si danno il cambio.
Lap	Entità che rappresenta l'insieme di tutti i giri che sono stati provati dai vari piloti. L'attributo Typology determina che tipo di giro è stato provato.
Manager	Entità che rappresenta il team leader e il direttore tecnico del team del polimarche. L'attributo Job serve proprio a determinare l'incarico del membro.
Member	Entità che rappresenta l'insieme dei vari membri del team che sono identificati tramite la matricola dell'università. L'attributo DOB si riferisce alla data di nascita. Ad ogni occorrenza viene associato anche il reparto di appartenenza del membro.
Practice Session	Entità che rappresenta l'insieme di tutte le sessioni in cui è stata provata la monoposto.
Problem	Entità che rappresenta l'insieme dei problemi che potrebbero presentarsi sui vari setup a contrario dell'attributo Notes di Setup che contiene dei semplici commenti.
Sensor	Entità che rappresenta tutti i singoli sensori presenti sulla monoposto. All'interno di ogni entità distinta vengono salvati i dati ottenuti tramite la centralina di comando presente nella monoposto.
Setup	Entità che rappresenta l'insieme di tutti i setup provati sulla monoposto. L'attributo Notes serve ad aggiungere dei commenti da parte del team sul singolo setup.

Spring	Entità che rappresenta l'insieme di tutte le possibili combinazioni dei vari parametri delle molle. L'attributo End serve a specificare se ci si sta riferendo alla parte frontale o alla parte del retro della monoposto.
Time	Entità che rappresenta l'insieme di tutte le tempistiche dei giri che sono stati provati. Vengono salvati i tempi solo per alcune determinate tipologie di giri.
Wheel	Entità che rappresenta l'insieme delle gomme utilizzate sulla monoposto con i proprio parametri. In particolare l'attributo Position specifica in che parte della monoposto quella gomma è presente.
Workshop	Entità che rappresenta l'insieme dei vari reparti del team. E' importante anche dire che all'interno di quest'entità viene salvato anche il magazzino nonostante non rappresenti un reparto effettivo. Ad ogni reparto, tranne che al Magazzino, viene associato un membro che rappresenta il caporeparto di un determinato reparto.

2.10 Traduzione del modello

Concetto	**Traduzione**
Balance	Balance(Balance_code, End, Brake, Weight)
Spring	Spring(Spring_code, End, Codification, Height, Arb_stiffness, Arb_position)
Damper	Damper(Damper_code, HSR, HSC, LSR, LSC, End)
Wheel	Wheel(Wheel_code, Pressure, Camber, Toe, Position, Codification)
Setup	Setup(Setup_code, Front_wing, Notes, **Spring**.Front, **Spring**.Rear, **Damper**.Front, **Damper**.Rear, **Wheel**.Front_right, **Wheel**.Front_left, **Wheel**.Rear_right, **Wheel**.Front_left)
IGBT	IGBT(**Setup**.Setup_code, Temperature)
FirstMotor	FirstMotor(**Setup**.Setup_code, Temperature)
SecondMotor	SecondMotor(**Setup**.Setup_code, Temperature)
Inverter	Inverter(**Setup**.Setup_code, Temperature)
Module	Module(**Setup**.Setup_code, Temperature)
PDM	PDM(**Setup**.Setup_code, Temperature)
VCU	VCU(**Setup**.Setup_code, Temperature)
MCU	MCU(**Setup**.Setup_code, Temperature)
PrevRadiator	PrevRadiator(**Setup**.Setup_code, Temperature)
AftRadiator	AftRadiator(**Setup**.Setup_code, Temperature)
BrakeRL	BrakeRL(**Setup**.Setup_code, Temperature)
BrakeRR	BrakeRR(**Setup**.Setup_code, Temperature)
BrakeFL	BrakeFL(**Setup**.Setup_code, Temperature)
BrakeFR	BrakeFR(**Setup**.Setup_code, Temperature)
Air	Air(**Setup**.Setup_code, Temperature)

Humidity	Humidity(**Setup**.Setup_code, Temperature)
TyreRL	TyreRL(**Setup**.Setup_code, Temperature)
TyreRR	TyreRR(**Setup**.Setup_code, Temperature)
TyreFL	TyreFL(**Setup**.Setup_code, Temperature)
TyreFR	TyreFR(**Setup**.Setup_code, Temperature)
Throttle	Throttle(**Setup**.Setup_code, Position)
SuspensionRL	SuspensionRL(**Setup**.Setup_code, Position)
SuspensionRR	SuspensionRR(**Setup**.Setup_code, Position)
SuspensionFL	SuspensionFL(**Setup**.Setup_code, Position)
SuspensionFR	SuspensionFR(**Setup**.Setup_code, Position)
SteeringAngle	SteeringAngle(**Setup**.Setup_code, Position)
GPS	GPS(??)
PushRL	PushRL(**Setup**.Setup_code, Load)
PushRR	PushRR(**Setup**.Setup_code, Load)
PushFL	PushFL(**Setup**.Setup_code, Load)
PushFR	PushFR(**Setup**.Setup_code, Load)
SteerTorque	SteerTorque(**Setup**.Setup_code, Load)
BrakeFront	BrakeFront(**Setup**.Setup_code, Pressure)
BrakeRear	BrakeRear(**Setup**.Setup_code, Pressure)
Coolant	Coolant(**Setup**.Setup_code, Pressure)
WheelFrontRight	WheelFrontRight(**Setup**.Setup_code, Speed)
WheelFrontLeft	WheelFrontLeft(**Setup**.Setup_code, Speed)
Accelerometer	Accelerometer(**Setup**.Setup_code, Speed)
BPCurrent	BPCurrent(**Setup**.Setup_code, Current)
WaterPump	WaterPump(**Setup**.Setup_code, Current)

LVBattery	LVBattery(**Setup**.Setup_code, Current)
CoolingFanSys	CoolingFanSys(**Setup**.Setup_code, Current)
LVBattery	LVBattery(**Setup**.Setup_code, Voltage)
5VSource	5VSource(**Setup**.Setup_code, Voltage)
24VSource	24VSource(**Setup**.Setup_code, Voltage)
Problem	Problem(Problem_code, Description)
Encounter	Encounter(**Problem**.Problem_code, **Setup**.Setup_code)
Solved	Solved(**Problem**.Problem_code, **Setup**.Setup_code)
Breakage	Breakage(Breakage_code, Description)
Caused	Caused(**Problem**.Problem_code, **Breakage**.Breakage_code)
DriverError	DriverError(Error_code, Description)
Fault	Fault(**DriverError**.Error_code, **Breakage**.Breakage_code)
Circuit	Circuit(Circuit_name, Lenght)
PracticeSessione	PracticeSession(Event_type, Date, Start_time, End_time, **Circuit**.Circuit_name, Weather, Track_condition, Track_temperature, Ambiental_pressure, Air_temperature)
Member	Member(Number, **Workshop**.Area_name, First_name, Last_name, DOB, Email, Password, Cell_number)
Workshop	Workshop(Area_name, **Member**.Number)
Manager	Manager(Job, **Member**.Number)
Driver	Driver(**Member**.Number, Weight, Height)
Lap	Lap(Lap_code, **Driver**.Number, **PracticeSession**.Event_type, **PracticeSession**.Date, **PracticeSession**.Start_time, **PracticeSession**.End_time, **Setup**.Setup_code, Typology)
Happen	Happen(**Lap**.Lap_code, **Breakage**.Breakage_code)
Join	Join(**Driver**.Number, **PracticeSession**.Event_type, **PracticeSession**.Date, **PracticeSession**.Start_time, **PracticeSession**.End_time, Change_time, Driver_order)

Comment	Comment(Comment_code, Description, Team_unit)
Released	Released(**Lap**.Lap_code, **Comment**.Comment_code)
Time	Time(**Lap**.Lap_code, First_sector, Second_sector, Third_sector, Fourth_sector)
HotParameters	HotParameters(Parameters_code, Temp_IFR, Temp_EFR, Temp_IFL, Temp_EFL, Press_FR, Press_RR, Press_FL, Press_RL, Temp_ELL, Temp_ILL, Temp_ERR, Temp_IRR)
Measure	Measure(**Lap**.Lap_code, **HotParameters**.Parameters_code)

2.11 Data types degli attributi

In questa sezione verranno stabiliti i tipi delle variabili usate per rappresentare i vari dati associati ai singoli attributi presenti nel modello logico completo.

Concetto	Traduzione
Balance	Balance_code → Int End → String Brake → Float Weight → Float
Spring	Spring_code → Int End → String Codification → String Height → Float Arb_stiffness → Float Arb_position → String
Damper	Damper_code → Int HSR → Float HSC → Float LSR → Float LSC → Float End → String
Wheel	Wheel_code → Int Pressure → Float Camber → String Toe → String Position → String Codification → String
Setup	Setup_code → Int Front_wing → String Notes → String **Spring**.Front → Int **Spring**.Rear → Int **Damper**.Front → Int **Damper**.Rear → Int **Wheel**.Front_right → Int **Wheel**.Front_left → Int **Wheel**.Rear_right → Int **Wheel**.Front_left → Int
IGBT	**Setup**.Setup_code → Int Temperature → Float
FirstMotor	**Setup**.Setup_code → Int Temperature → Float
SecondMotor	**Setup**.Setup_code → Int Temperature → Float
Inverter	**Setup**.Setup_code → Int Temperature → Float

Module	**Setup**.Setup_code → Int Temperature → Float
PDM	**Setup**.Setup_code → Int Temperature → Float
VCU	**Setup**.Setup_code → Int Temperature → Float
MCU	**Setup**.Setup_code → Int Temperature → Float
PrevRadiator	**Setup**.Setup_code → Int Temperature → Float
AftRadiator	**Setup**.Setup_code → Int Temperature → Float
BrakeRL	**Setup**.Setup_code → Int Temperature → Float
BrakeRR	**Setup**.Setup_code → Int Temperature → Float
BrakeFL	**Setup**.Setup_code → Int Temperature → Float
BrakeFR	**Setup**.Setup_code → Int Temperature → Float
Air	**Setup**.Setup_code → Int Temperature → Float
Humidity	**Setup**.Setup_code → Int Temperature → Float
TyreRL	**Setup**.Setup_code → Int Temperature → Float
TyreRR	**Setup**.Setup_code → Int Temperature → Float
TyreFL	**Setup**.Setup_code → Int Temperature → Float
TyreFR	**Setup**.Setup_code → Int Temperature → Float
Throttle	**Setup**.Setup_code → Int Position → Float
SuspensionRL	**Setup**.Setup_code → Int Position → Float
SuspensionRR	**Setup**.Setup_code → Int Position → Float
SuspensionFL	**Setup**.Setup_code → Int Position → Float
SuspensionFR	**Setup**.Setup_code → Int Position → Float
SteeringAngle	**Setup**.Setup_code → Int Position → Float
GPS	????

PushRL	**Setup**.Setup_code → Int Load → Float
PushRR	**Setup**.Setup_code → Int Load → Float
PushFL	**Setup**.Setup_code → Int Load → Float
PushFR	**Setup**.Setup_code → Int Load → Float
SteerTorque	**Setup**.Setup_code → Int Load → Float
BrakeFront	**Setup**.Setup_code → Int Pressure → Float
BrakeRear	**Setup**.Setup_code → Int Pressure → Float
Coolant	**Setup**.Setup_code → Int Pressure → Float
WheelFrontRight	**Setup**.Setup_code → Int Speed → Float
WheelFrontLeft	**Setup**.Setup_code → Int Speed → Float
Accelerometer	**Setup**.Setup_code → Int Speed → Float
BPCurrent	**Setup**.Setup_code → Int Current → Float
WaterPump	**Setup**.Setup_code → Int Current → Float
LVBattery	**Setup**.Setup_code → Int Current → Float
CoolingFanSys	**Setup**.Setup_code → Int Current → Float
LVBattery	**Setup**.Setup_code → Int Voltage → Float
5VSource	**Setup**.Setup_code → Int Voltage → Float
24VSource	**Setup**.Setup_code → Int Voltage → Float
Problem	Problem_code → Int Description → String
Encounter	**Problem**.Problem_code → Int **Setup**.Setup_code → Int
Solved	**Problem**.Problem_code → Int **Setup**.Setup_code → Int
Breakage	Breakage_code → Int Description → String
Caused	**Problem**.Problem_code → Int **Breakage**.Breakage_code → Int

DriverError	Error_code → Int Description → String
Fault	**DriverError**.Error_code → Int **Breakage**.Breakage_code → Int
Circuit	Circuit_name → String Lenght → Float
PracticeSessione	Event_type → String Date → Date Start_time → Time End_time → End **Circuit**.Circuit_name → String Weather → String Track_condition → String Track_temperature → Float Ambiental_pressure → Float Air_temperature → Float
Member	Number → Bigint **Workshop**.Area_name → String First_name → String Last_name → String DOB → Date Email → String Password → String Cell_number → Bigint
Workshop	Area_name → String **Member**.Number → Bigint
Manager	Job → String **Member**.Number → Bigint
Driver	**Member**.Number → Bigint Weight → Float Height → Float
Lap	Lap_code → Int **Driver**.Number → Bigint **PracticeSession**.Event_type → String **PracticeSession**.Date → Date **PracticeSession**.Start_time → Time **PracticeSession**.End_time → Time **Setup**.Setup_code → Int Typology → String
Happen	**Lap**.Lap_code → Int **Breakage**.Breakage_code → Int
Join	**Driver**.Number → Bigint **PracticeSession**.Event_type → String **PracticeSession**.Date → Date **PracticeSession**.Start_time → Time **PracticeSession**.End_time → Time Change_time → Time Driver_order → Int

Comment	Comment_code → Int Description → String Team_unit → String
Released	**Lap**.Lap_code → Int **Comment**.Comment_code → Int
Time	**Lap**.Lap_code → Int First_sector → Time Second_sector → Time Third_sector → Time Fourth_sector → Time
HotParameters	Parameters_code → Int Temp_IFR → Float Temp_EFR → Float Temp_IFL → Float Temp_EFL → Float Press_FR → Float Press_RR → Float Press_FL → Float Press_RL → Float Temp_ELL → Float Temp_ILL → Float Temp_ERR → Float Temp_IRR → Float
Measure	**Lap**.Lap_code → Int **HotParameters**.Parameters_code → Int