The Gombe-MI Pocket Reference Guide

A Technical Specification Summary

COLLABORATORS			
	TITLE : The Gombe-MI Pocket Reference Guide		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Karl O. Pinc, Elizabeth V. Lonsdorf, PHD., and Carson M. Murray, PhD.	August 25, 2018	

Contents

List of Figures

1 Gombe-MI Summarized

1.1 Main Table Overview

Demography Related	
Table	One row for each
ADOPTIONS	time a chimpanzee is adopted
BIOGRAPHY_DATA	chimpanzee
CHIMPIDS	chimpanzee id; usually 1 row per chimpanzee but 2 rows per twin
COMMUNITY_MEMBERSHIP	tenure of an individual in a community
TWINS	individual who is a twin

Follow Mechanics Related	
Table	One row for each
FIELDFOLLOWS	mother infant data packet collected by any one researcher in the field
FOLLOWPARTS	mother infant follow for every focal participant
FOLLOWS	mother infant follow
INTERVALS	observation time interval of every follow
Sheets	template sheet used in the field or, when the details of what was on which sheet are not available, one row per follow

Follow Event Related	
Table	One row for each
EATS	part of every kind of food eaten during a eating behavioral event
EVENTS	non-social behavior and one row for each social behavior/social partner combination of each focal recorded for each interval
FISHED	every kind of food fished for during a fishing behavioral event
PARTS	individual, focal or not, involved in every recorded behavior, excluding baboons but including unrecognized or otherwise anonymous individuals

Follow Event Related	
Table	One row for each
PLAYS	play activity occurring during a play event
RESTS	resting posture assumed during a resting event
RIDES	riding posture assumed during a riding event
VOCS	kind of vocalization made during a vocalization behavioral event

Follow Distance Related	
Table	One row for each
DISTANCEPARTS	individual who's distance to another individual was recorded
DISTANCES	individual who's distance to another individual was recorded

Group Related	
Table	One row for each
GROUPMEMBERS	group member identified per automated group survey analysis
GROUPS	automated group survey analysis
RAW_GROUPMEMBERS	group member identified per group survey
RAW_GROUPS	group survey

1.2 The Warning Sub-System Tables

Table	One row for each
INTEGRITY_QUERIES	query used to discover data integrity problems
INTEGRITY_WARNINGS	data integrity problem discovered by the warning sub-system

1.3 Support Table Overview

Demography Related			
Table	Id Column	Related Column(s)	One entry for every possible choice of...
BDDISTS	BDDist	BIOGRAPHY_DATA.BDDist	birth date probability distribution
DEPARTTYPES	Departtype	BIOGRAPHY_DATA.Departtype	means by which individual's depart from the study population
ENTRYTYPES	Entrytype	BIOGRAPHY_DATA.Entrytype	means by which individual's enter into the study population

Follow Mechanics Related			
Table	Id Column	Related Column(s)	One entry for every possible choice of...
BEHAVIOR_TRANS	Old_Behavior	None	behavior code written in the field plus one row for every column on the field data entry sheet which represents specific behaviors
DISTANCE_TRANS	Old_Distance	None	distance code written in the field plus one row for every column on the field data entry sheet which represents a specific inter-focal distance
FOLLOWROLES	Role	FOLLOWPARTS.Role, DISTANCEPARTS.Role	role among follow focals (mother, infant, sibling, twin1, etc.)
FOLLOWTYPES	Type	Sheets.Followtype	type of data entry sheet
FOODPART_TRANS	Old_Foodpart	None	food part code written in the field
GC_TRANS	Mark	None	mark written in the field onto the group composition portion of the data collection sheets, in the cells of the columns which record chimpanzee or baboon presence
PEOPLE	Person	FIELDFOLLOWS.Collector, FIELDFOLLOWS.Enterer	person who collects data in the field or who enters data in the labs
PLAY_TRANS	Old_Play	None	play code written in the field
VOC_TRANS	Old_Voc	None	vocalization code written in the field

Follow Event Related			
Table	Id Column	Related Column(s)	One entry for every possible choice of...
BEHAVIORS	Behavior	EVENTS.Behavior, BEHAVIOR_TRANS.New_Behavior	kind of behavior
FISHING_RESULTS	Result	FISHED.Result	outcome of a fishing attempt
FOODKINDS	Foodkind	EATS.Foodkind, BEHAVIOR_TRANS.Fishkind_Val	kind of food eaten by chimpanzees
FOODPARTS	Foodpart	EATS.Foodpart, FOODPART_TRANS.New_Foodpart, BEHAVIOR_TRANS.Fishpart_Val	part (plant part, animal organ, etc) of the different kinds of foods eaten by chimpanzees
POSITIONS	Position	EVENTS.Position, BEHAVIOR_TRANS.Position_Val	designated region within the forest canopy
PLAY_TYPES	Play	PLAYS.Play, PLAY_TRANS.New_Play, BEHAVIOR_TRANS.Play_Val	play activity
RESTING_POSTURES	Rest	RESTS.Rest, BEHAVIOR_TRANS.Rest_Val	posture when resting
RIDING_POSTURES	Rides	RIDES.Ride, BEHAVIOR_TRANS.Ride_Val	posture when riding
VOCALIZATIONS	Vocalization	VOCS.Vocalization, VOC_TRANS.New_Voc, BEHAVIOR_TRANS.Vocalization_Val	class of chimpanzee vocalization defined in the data collection protocols

Follow Distance Related			
Table	Id Column	Related Column(s)	One entry for every possible choice of...
DISTANCECODES	Distance	DISTANCES.Distance, DISTANCE_TRANS.New_Distance	distance category

Group Related			
Table	Id Column	Related Column(s)	One entry for every possible choice of...
GCSTATUSES	GCStatus	FIELDFOLLOWS.GCStatus	group composition status
GM_ORIGINS	Origin	RAW_GROUPMEMBERS.Origin, GC_TRANS.Origin	source of data for an individual's group membership

1.4 The Warning Sub-System Support Tables

Table	Id Column	Related Column(s)	One entry for every possible choice of...
IQTYPES	IQType	INTEGRITY_QUERIES.Type	kind of problem with data integrity
WARNING_REMARKS	WRID	INTEGRITY_WARNINGS.Category	remark which might apply to more than one instance of questionable database integrity

1.5 An Overview Of The Views

Demography Related View	One row for each	Purpose	Tables/Views used
BIOGRAPHY	BIOGRAPHY_DATA row	Provides an interface into the master biography table imported from Access which is invariant, and slightly easier to use.	BIOGRAPHY_DATA, TWINS
BIOGRAPHY_UPLOAD	BIOGRAPHY_DATA row	Provides an upload interface into the master biography table enabling easy import from Access.	BIOGRAPHY_DATA

Demography Related View	One row for each	Purpose	Tables/Views used
CHIMPS	BIOGRAPHY_DATA row plus one row for each twin	Provides an interface into the master biography table with one extra row for each twin, which is a duplicate of the twin's BIOGRAPHY_DATA row but used when the twins are not distinguish-able.	BIOGRAPHY_DATA, TWINS CHIMPIDS
COMMUNITY_MEMBERSHIP_UPLOAD	COMMUNITY_MEMBERSHIP row	Provides an upload interface into the master commu-nity_membership table enabling easy import from Access.	COMMUNITY_MEMBERSH

Follow Mechanics Related View	One row for each	Purpose	Tables/Views used
DATASHEET_EVENTS	EVENTS row per any related EATS, FISHED, PLAYS, RESTS, RIDES, or VOCS rows.	To resemble the event portion of the data entry sheets and provide a mechanism for updating event related data in the database.	FOLLOWS, INTERVALS, EVENTS, PARTS, EATS, FISHED, PLAYS, RESTS, RIDES, VOCS
DATASHEETS	EVENTS row per any related EATS, FISHED, PLAYS, RESTS, RIDES, or VOCS rows plus one row per DISTANCES row	To produce a report resembling the data entry sheets.	FOLLOWS, INTERVALS, EVENTS, PARTS, DISTANCES, DISTANCEPARTS, EATS, FISHED, PLAYS, RESTS, RIDES, VOCS
FOCAL_FOLLOWS	FOLLOWS row, per each possible paring of infant and sibling (i.e. one row per follow when there are no twins, two rows per follow when either the infants or siblings are twins, and 4 rows per follow when both the infants and siblings are twins)	To provide a simple mechanism for deleting and creating follows.	FOLLOWS, FOLLOWPARTS (3 times)
FOLLOW_DURATIONS	FOLLOWS row	Computes various totals of time spent observing during the follow.	FOLLOWS
FOLLOW_INTERVALS	INTERVALS row per INTERVALS row		FOLLOWS, FOLLOWPARTS (3 times), INTERVALS

Follow Mechanics Related View	One row for each	Purpose	Tables/Views used
UPLOAD_BEHAVIORS	FOLLOWS row	This view returns no rows. It is used only to upload follow data into Gombe-MI.	FOLLOWS, FOLLOWPARTS INTERVALS, EVENTS, PARTS, DISTANCES, DISTANCEPARTS

Follow Event Related View	One row for each	Purpose	Tables/Views used
ACTOR_ACTEES	EVENTS row where there are 2 chimpanzees involved in an actor/actee behavioral event, events involving baboons are not included.	Provides a straightforward way to analyze directed social behaviors from the perspective of who is initiating and who receiving the behavior.	EVENTS, PARTS (twice, once for each chimpanzee involved in the behavior), FOLLOWROLES (twice, once for each chimpanzee involved in the behavior)
ACTOR_ACTEES_W_BABOONS	EVENTS row where there are 2 animals involved in an actor/actee behavioral event, including events involving baboons.	Provides a straightforward way to analyze directed social behaviors from the perspective of who is initiating and who receiving the behavior.	EVENTS, PARTS (twice, once for each animal involved in the behavior), FOLLOWROLES (up to twice, once for each chimpanzee involved in the behavior)

Follow Event Related View	One row for each	Purpose	Tables/Views used
ASOCIAL_EVENTS	EVENTS row having a related PARTS row with a Part value of S	Provides a direct way to analyze asocial behavioral events.	EVENTS, PARTS, FOLLOWROLES
EVENT_BOUTS	FOLLOWPARTS row per INTERVALS row, per EVENTS.Behavior value having a related PARTS row that shows the given focal involved in the given behavior -- one row per follow, per focal, per interval in which the focal is involved in a behavior (excluding behavioral events which involve a baboon), per code for the behavior involved in.	Provides per-focal, per-minute analysis for contiguous minutes of per behavior-code activity.	FOLLOWPARTS, FOLLOWROLES, INTERVALS, EVENTS, PARTS, CHIMPIDS
EVENT_BOUT_AGGS	FOLLOWPARTS row, per EVENTS.Behavior value having a related PARTS row that shows the given focal involved in the given behavior -- one row per follow, per focal in which the focal is involved in a behavior (excluding behavioral events which involve a baboon), per code for the behavior involved in.	Provides per-focal analysis for contiguous minutes of per behavior-code activity.	FOLLOWPARTS, FOLLOWROLES, INTERVALS, EVENTS, PARTS, CHIMPIDS

Follow Event Related View	One row for each	Purpose	Tables/Views used
FAMILY_EVENTS	EVENTS row	Provides a way to review recorded behaviors in a way similar to the way data is recorded in the field.	EVENTS, PARTS (four times, once for each of the mother, infant, sibling, and other non-focal individual who may be involved in the behavior)
FOCAL_EVENTS	EVENTS row *per focal individual involved in the behavioral event*	Provides a direct way to analyze behaviors when interested in the function (mom, infant, sib, non-focal) of the chimpanzees involved in the behavior.	EVENTS, PARTS (once or twice, once for the focal and again if there is another individual), FOLLOWROLES (once or twice, once for the focal and again if there is another individual)
FOCAL_SOCIAL_EVENTS	EVENTS row *per individual involved in the behavioral event when the individual is a focal in the follow* where there are 2 chimpanzees involved in the exhibited behavior	Provides a direct way to analyze social behaviors when interested in the function (mom, infant, sib, non-focal) of the chimpanzees involved in the behavior.	EVENTS, PARTS (twice, once for the focal and once for the other individual), FOLLOWROLES (twice, once for the focal and once for the other individual)

Follow Event Related View	One row for each	Purpose	Tables/Views used
NOOBS	FOLLOWS row, per FOLLOWPARTS row, per minute from 06:00 through 20:00, inclusive, but only when there is no behavioral data on the focal.	Computes minutes in each follow's day during which each focal has no data on behavior.	FOLLOWPARTS, CHIMPIDS INTERVALS, EVENTS, PARTS
SOCIAL_BOUTS	FOLLOWPARTS row per INTERVALS row, per EVENTS.Behavior value having a related PARTS row that shows the given focal involved in the given behavior with a social partner, per AnimID of the social partner -- one row per follow, per focal, per social partner, per interval in which the focal is involved in a social behavior (excluding behavioral events which involve a baboon), per code for the behavior involved in.	Provides per-focal, per-social-partner, per-minute analysis for contiguous minutes of per behavior-code activity.	FOLLOWPARTS, FOLLOWROLES, INTERVALS, EVENTS, PARTS, CHIMPIDS
SOCIAL_BOUT_AGGS	FOLLOWPARTS row per EVENTS.Behavior value having a related PARTS row that shows the given focal involved in the given behavior with a social partner, per AnimID of the social partner -- one row per follow, per focal, per social partner (excluding behavioral events which involve a baboon), per code for the behavior involved in.	Provides per-focal, per social partner analysis for contiguous minutes of per behavior-code activity.	FOLLOWPARTS, FOLLOWROLES, INTERVALS, EVENTS, PARTS, CHIMPIDS

Follow Event Related			
View	One row for each	Purpose	Tables/Views used
SOCIAL_EVENTS	EVENTS row, per chimpanzee, where there are 2 chimpanzees (no baboons) involved in the exhibited behavior. So, 2 rows per EVENTS row for those events, involving chimpanzees, that are social in nature.	Provides a direct way to analyze chimp social behaviors regardless of whether a chimpanzee is a focal in the follow.	EVENTS, PARTS (twice, once for each chimpanzee involved in the behavior), FOLLOWROLES (twice, once for each chimpanzee involved in the behavior)
SOCIAL_EVENTS_W_BABOONS	EVENTS row, per chimpanzee, where there are 2 animals involved in the exhibited behavior, baboons included. So, 2 rows per EVENTS row for those events, involving animals, that are social in nature.	Provides a direct way to analyze social behaviors regardless of whether a chimpanzee is a focal in the follow.	EVENTS, PARTS (twice, once for each chimpanzee involved in the behavior), FOLLOWROLES (twice, once for each chimpanzee involved in the behavior)

Follow Distance Related			
View	One row for each	Purpose	Tables/Views used
DISTS	DISTANCES row per related DISTANCEPARTS row. So, 2 rows per DISTANCES row.	Provides a direct way to analyze chimp inter-familial distances using either ChimpID or role the chimp plays in the follow.	DISTANCES, DISTANCEPARTS (twice, once for each chimpanzee involved).

Follow Distance Related View	One row for each	Purpose	Tables/Views used
FAMILY_DISTS	DISTANCES row	Provides a direct way to analyze chimp inter-familial distances based on the role (mom, infant, sibling) the chimp plays in the follow, and a way to maintain the underlying data.	DISTANCES, DISTANCEPARTS (three times, once for each possible role played in the follow).

2 The Gombe-MI ER Diagrams

2.1 Gombe-MI Entity Relationship Diagrams

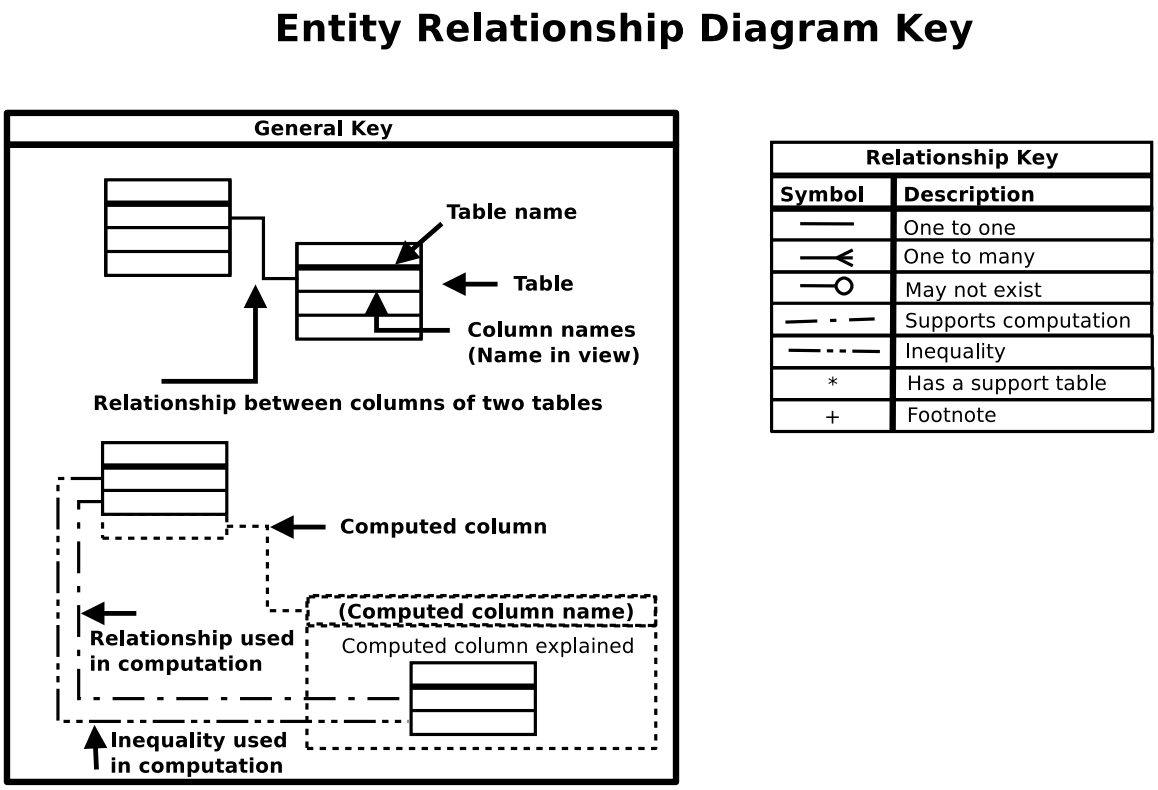


Figure 1: Key to the Gombe-MI Entity Relationship Diagrams

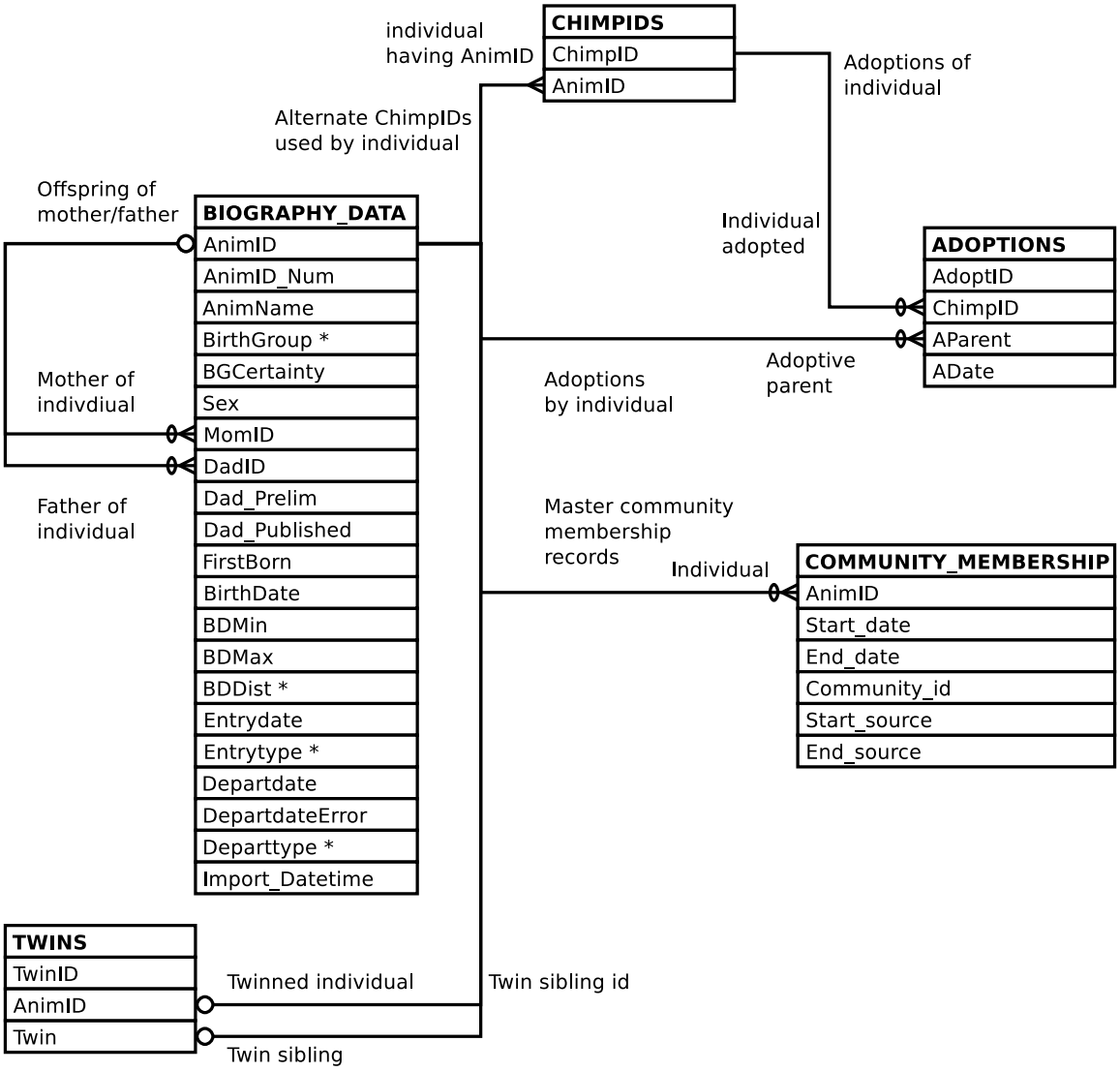


Figure 2: Gombe-MI Demography Entity Relationship Diagram

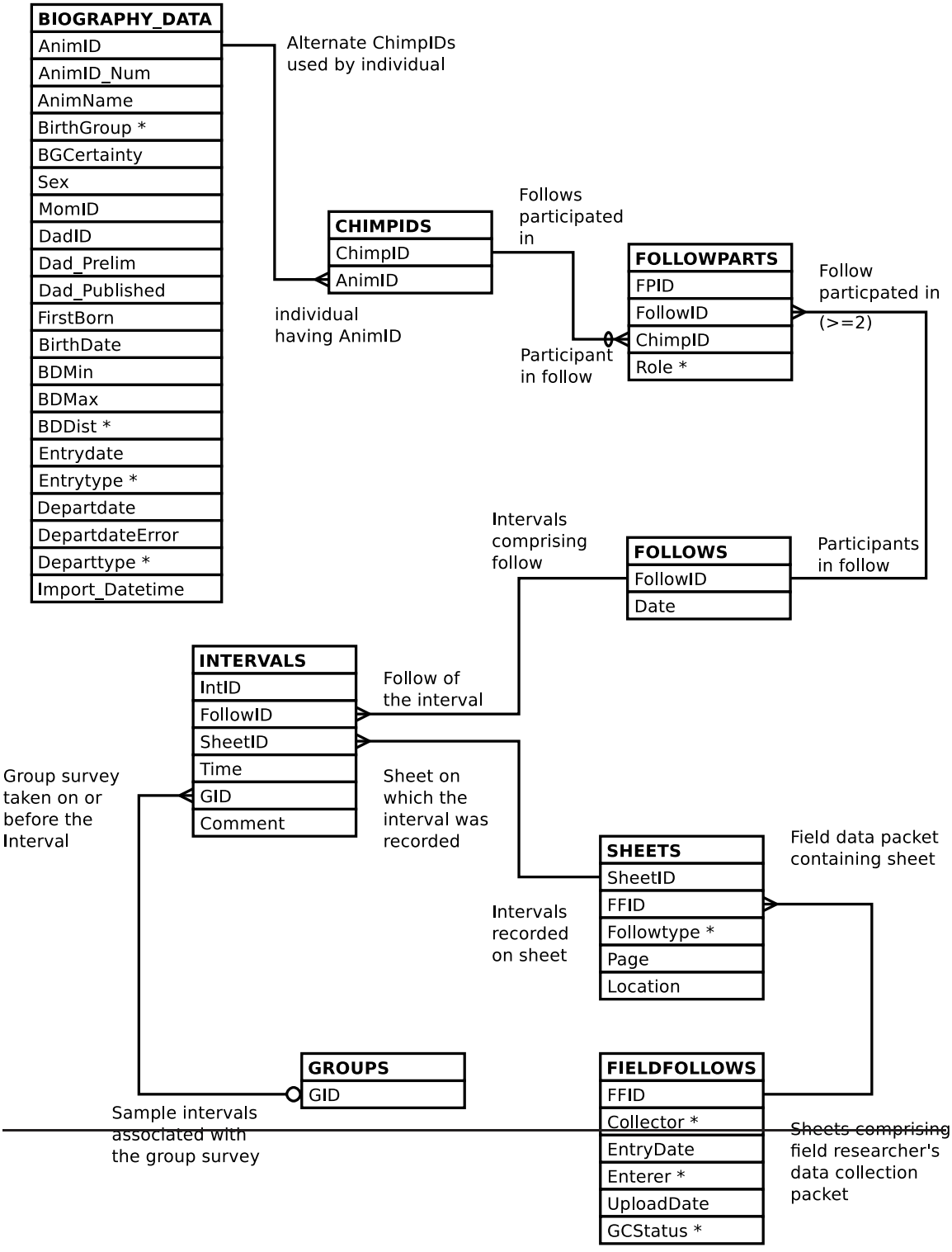


Figure 3: Gombe-MI Essentials of a Follow Entity Relationship Diagram

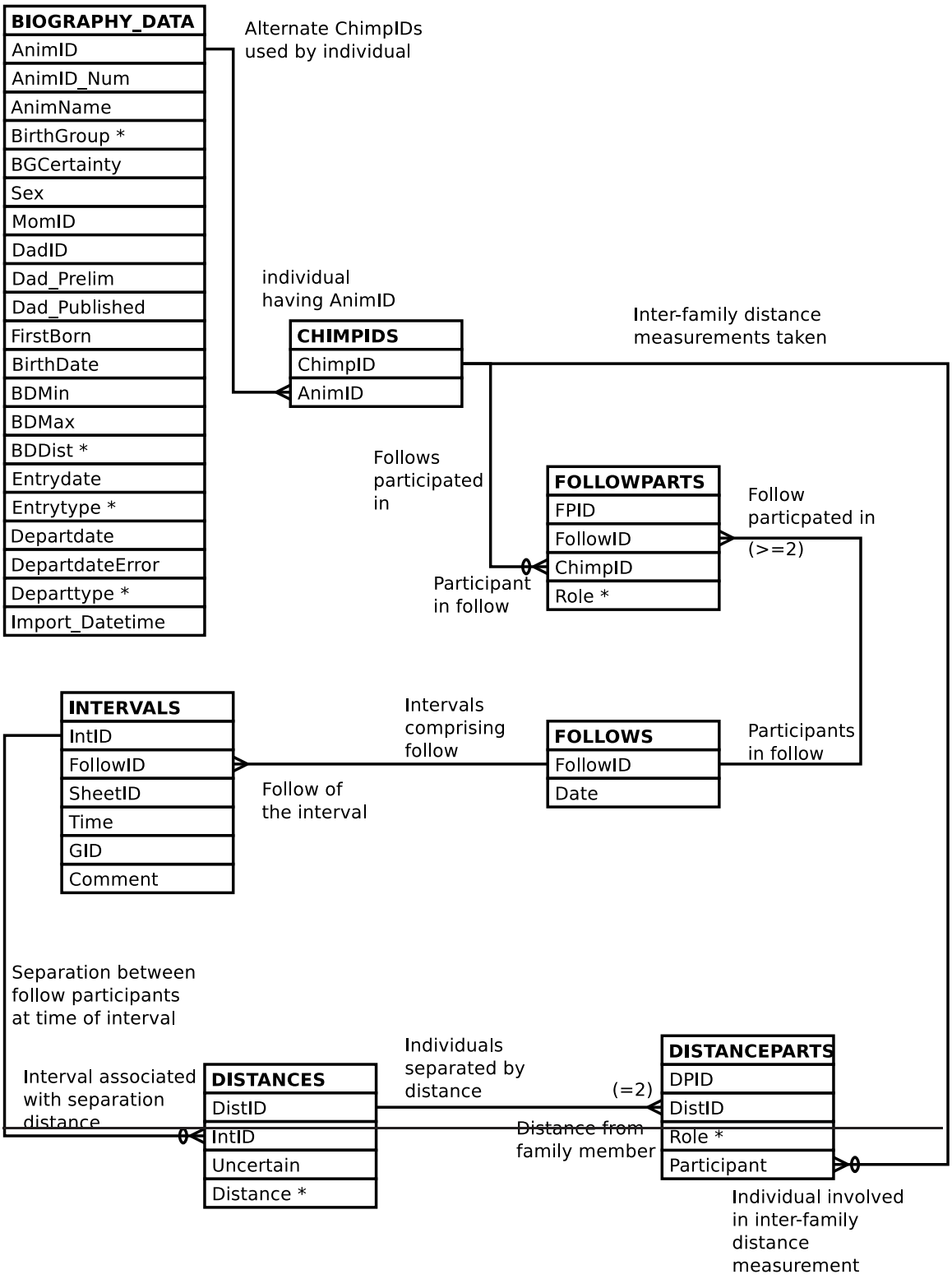
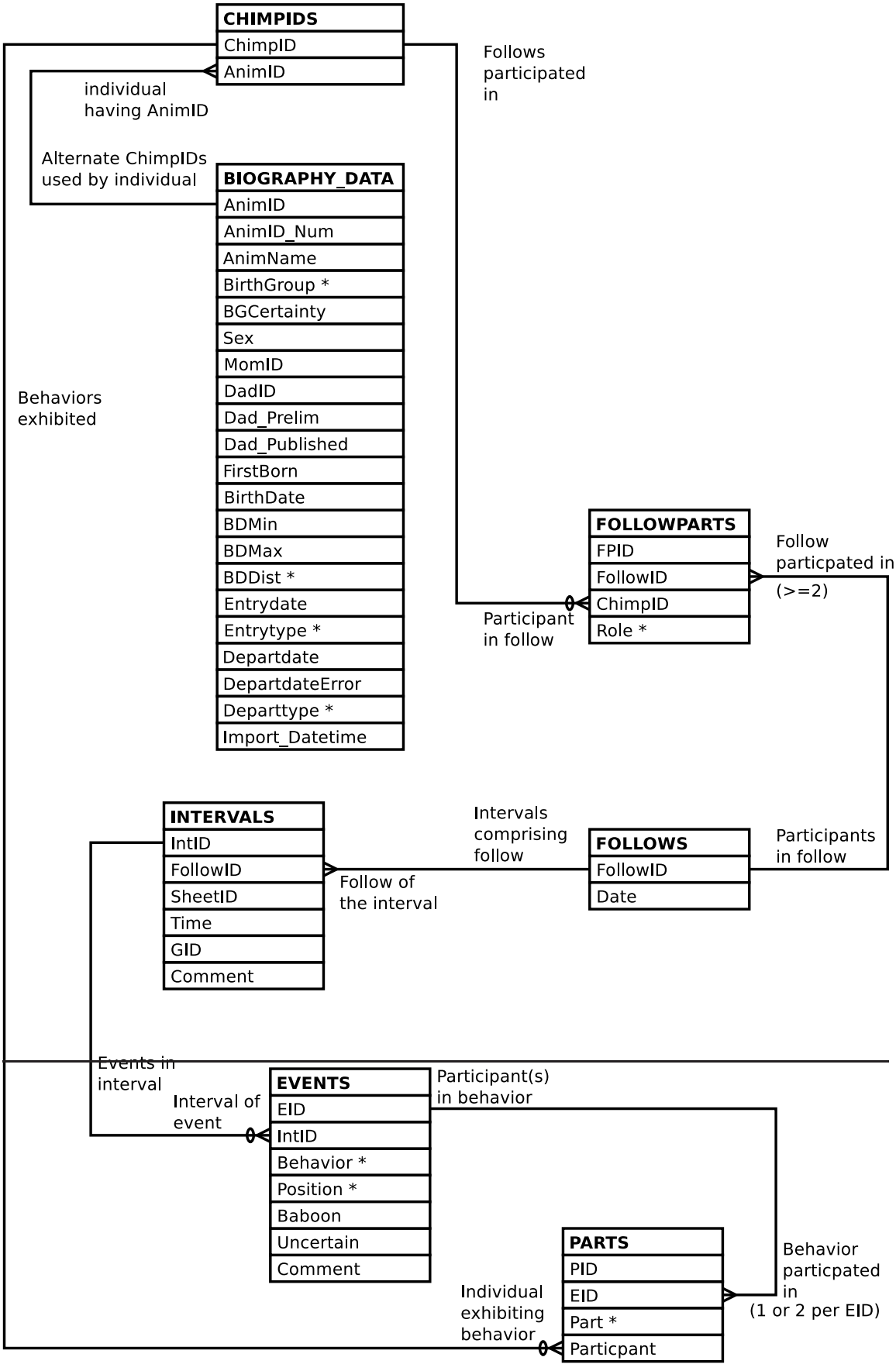
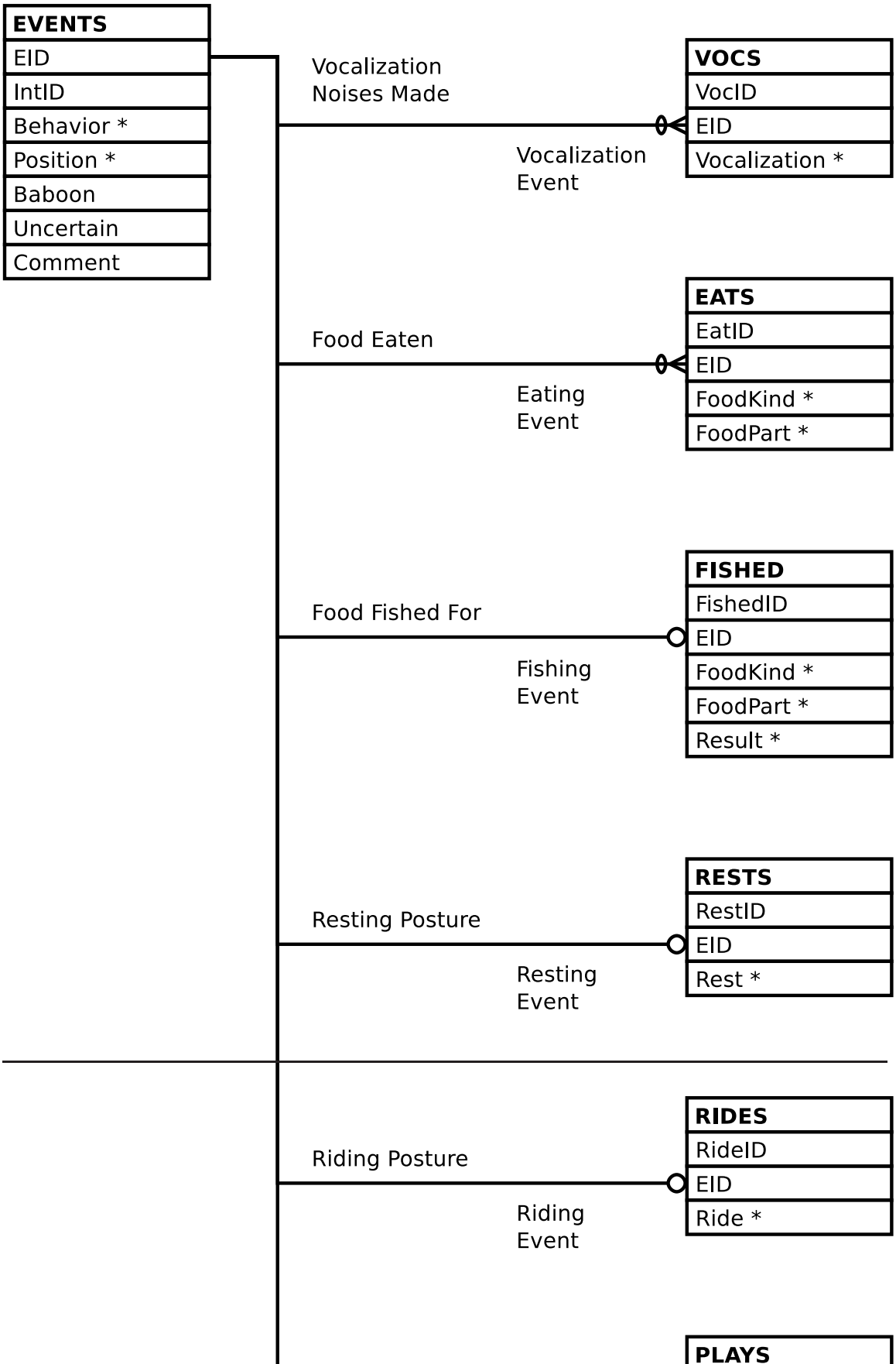


Figure 4: Gombe-MI Follow Distance Entity Relationship Diagram


CHIMPIDS
ChimpID
AnimID
individual
having AnimID
Alternate ChimpIDs
used by individual
BIOGRAPHY_DATA
AnimID
AnimID_Num
AnimName
BirthGroup *
BGCertainty
Sex
MomID
DadID
Dad_Prelim
Dad_Published
FirstBorn
BirthDate
BDMin
BDMax
BDDist *
Entrydate
Entrytype *
Departdate
DepartdateError
Departtype *
Import_Datetime
Behaviors
exhibited
Follows
participated
in
FOLLOWPARTS
FPID
FollowID
ChimpID
Role *
Participant
in follow
Follow
particpated in
(>=2)
INTERVALS
IntID
FollowID
SheetID
Time
GID
Comment
Follow of
the interval
Intervals
comprising
follow
FOLLOWS
FollowID
Date
Participants
in follow
Events in
interval
Interval of
event
EVENTS
EID
IntID
Behavior *
Position *
Baboon
Uncertain
Comment
Participant(s)
in behavior
PARTS
PID
EID
Part *
Particpant
Individual
exhibiting
behavior
Behavior
particpated
in
(1 or 2 per EID)


EVENTS
EID
IntID
Behavior *
Position *
Baboon
Uncertain
Comment
Vocalization
Noises Made
Vocalization
Event
VOCS
VocID
EID
Vocalization *
Food Eaten
Eating
Event
EATS
EatID
EID
FoodKind *
FoodPart *
Food Fished For
Fishing
Event
FISHED
FishedID
EID
FoodKind *
FoodPart *
Result *
Resting Posture
Resting
Event
RESTS
RestID
EID
Rest *
Riding Posture
Riding
Event
RIDES
RideID
EID
Ride *
PLAYS

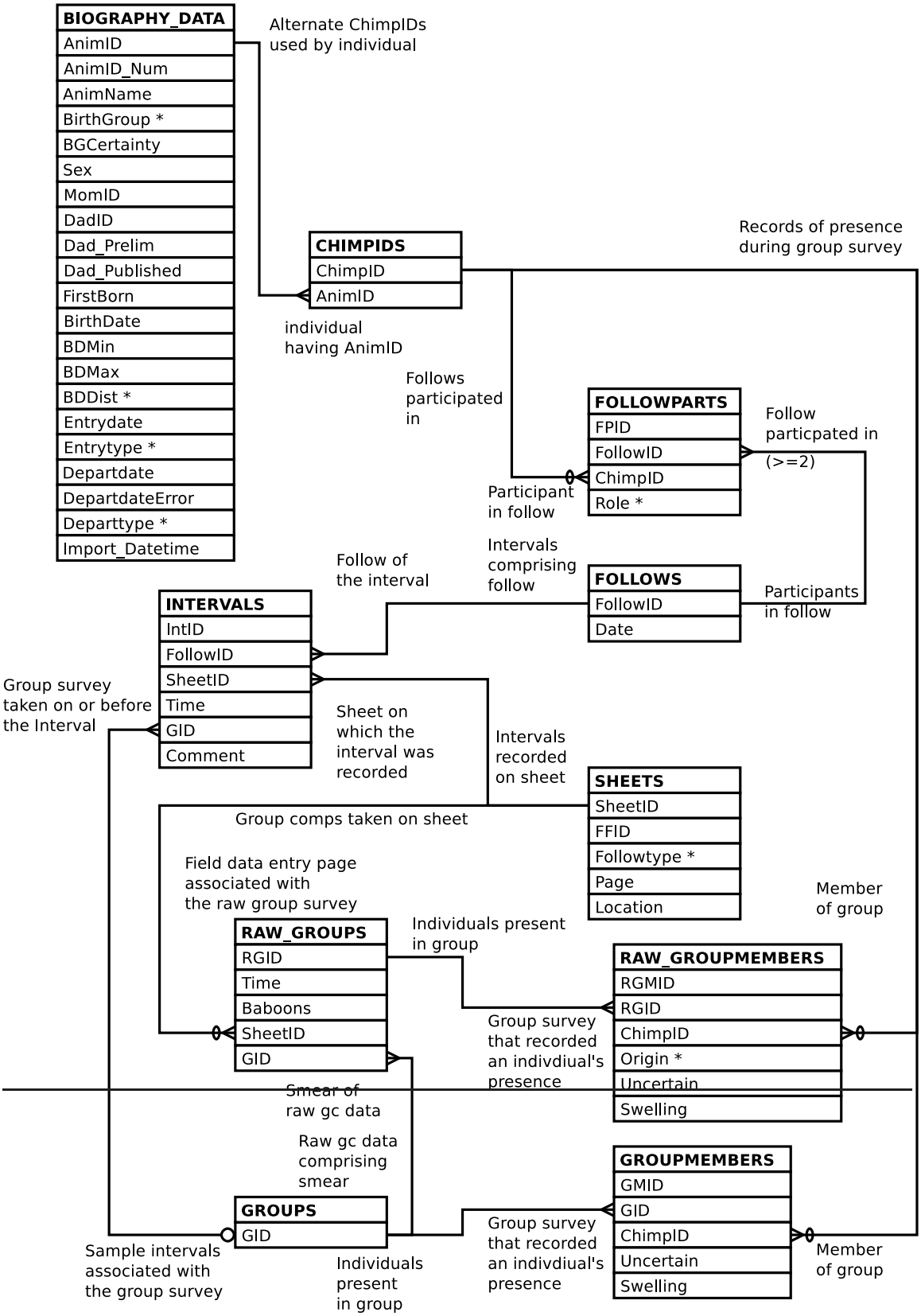


Figure 7: Gombe-MI Group Membership Entity Relationship Diagram

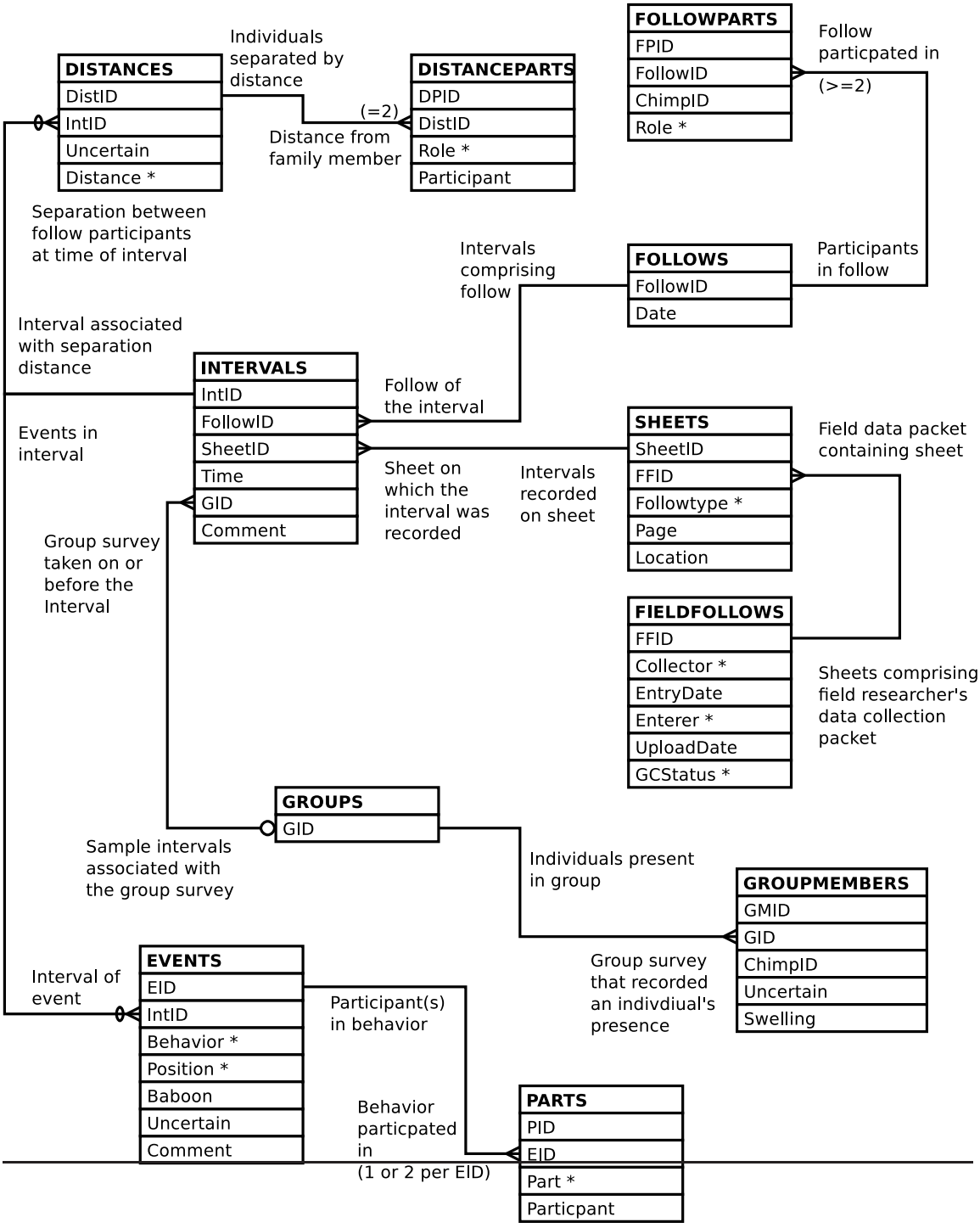


Figure 8: Gombe-MI Follow Detailed Entity Relationship Diagram

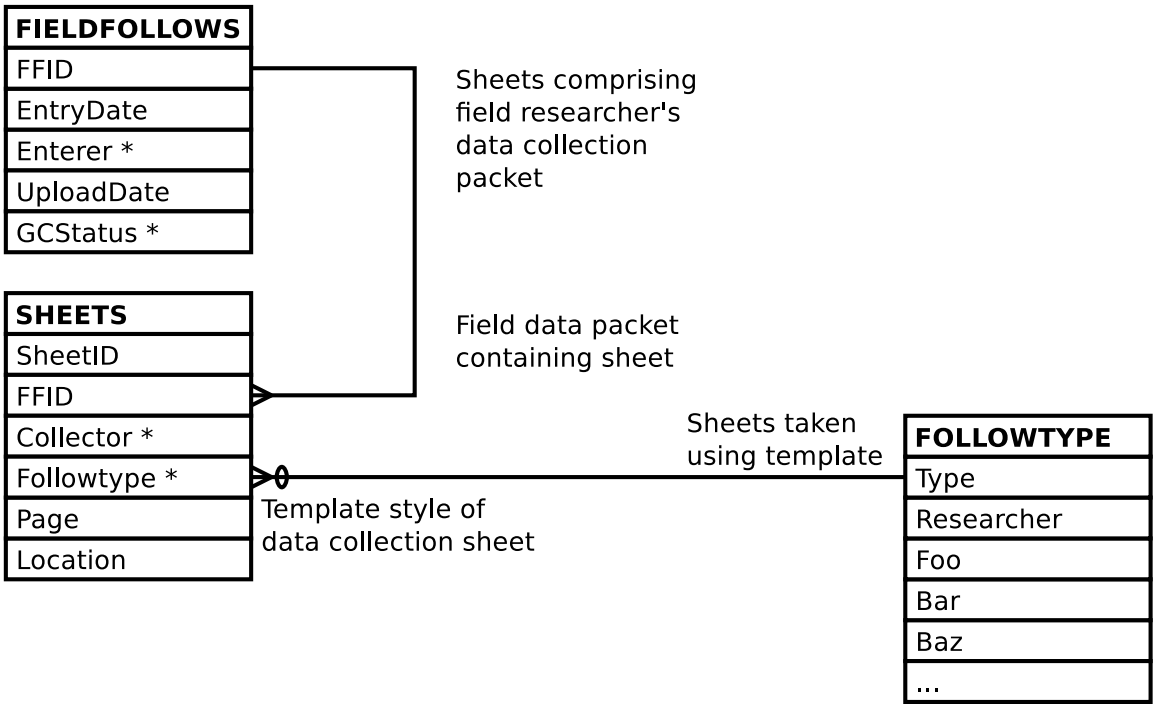


Figure 9: Gombe-MI Data Collection Templates Entity Relationship Diagram

2.2 Warning Sub-System ER Diagrams

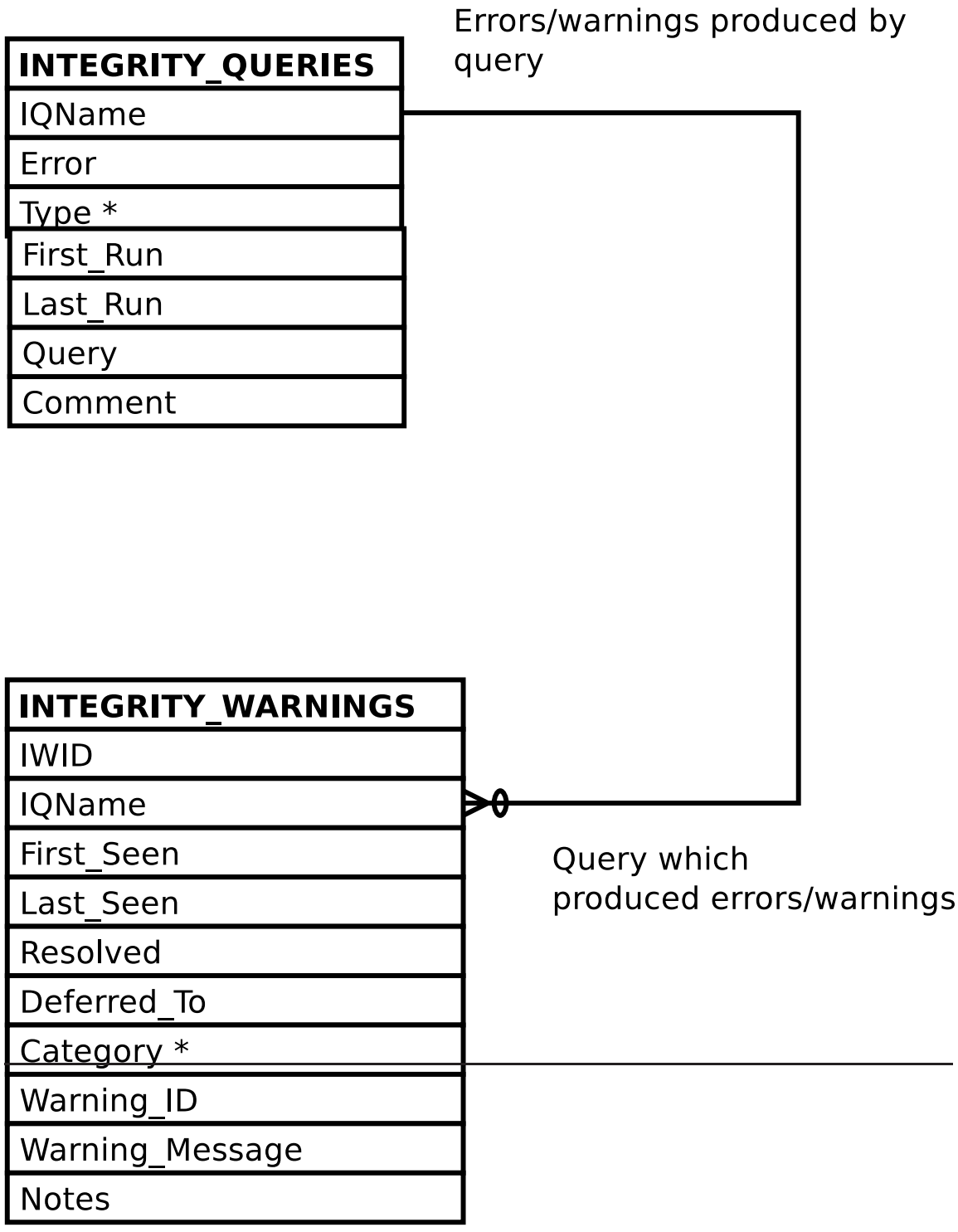


Figure 10: Warning Sub-System Entity Relationship Diagram

3 The Gombe-MI Views

For information on the operations (INSERT, UPDATE, DELETE) allowed by each view and their actions on the underlying tables see The Gombe-MI Views of The Gombe-MI Reference Manual.

3.1 Demography Related Views

3.1.1 The BIOGRAPHY View

```
SELECT biography_data.animid AS animid
     , biography_data.animid_num AS animid_num
     , biography_data.animname AS animname
     , biography_data.birthgroup AS birthgroup
     , biography_data.bgcertainty AS bgcertainty
     , biography_data.sex AS sex
     , biography_data.momid AS momid
     , biography_data.dadid AS dadid
     , biography_data.dad_prelim AS dad_prelim
     , biography_data.dad_published AS dad_published
     , biography_data.firstborn AS firstborn
     , biography_data.birthdate AS birthdate
     , biography_data.bdmin AS bdmin
     , biography_data.bdmax AS bdmax
     , biography_data.bddist AS bddist
     , biography_data.entrydate AS entrydate
     , biography_data.entrytype AS entrytype
     , biography_data.departdate AS departdate
     , biography_data.departdateerror AS departdateerror
     , biography_data.departtype AS departtype
     , biography_data.import_datetime AS import_datetime
     , twins.twin AS twin
  FROM biography_data
       LEFT OUTER JOIN twins ON (twins.animid = biography_data. ↩
          animid);
```

Figure 11: Query Defining the BIOGRAPHY View

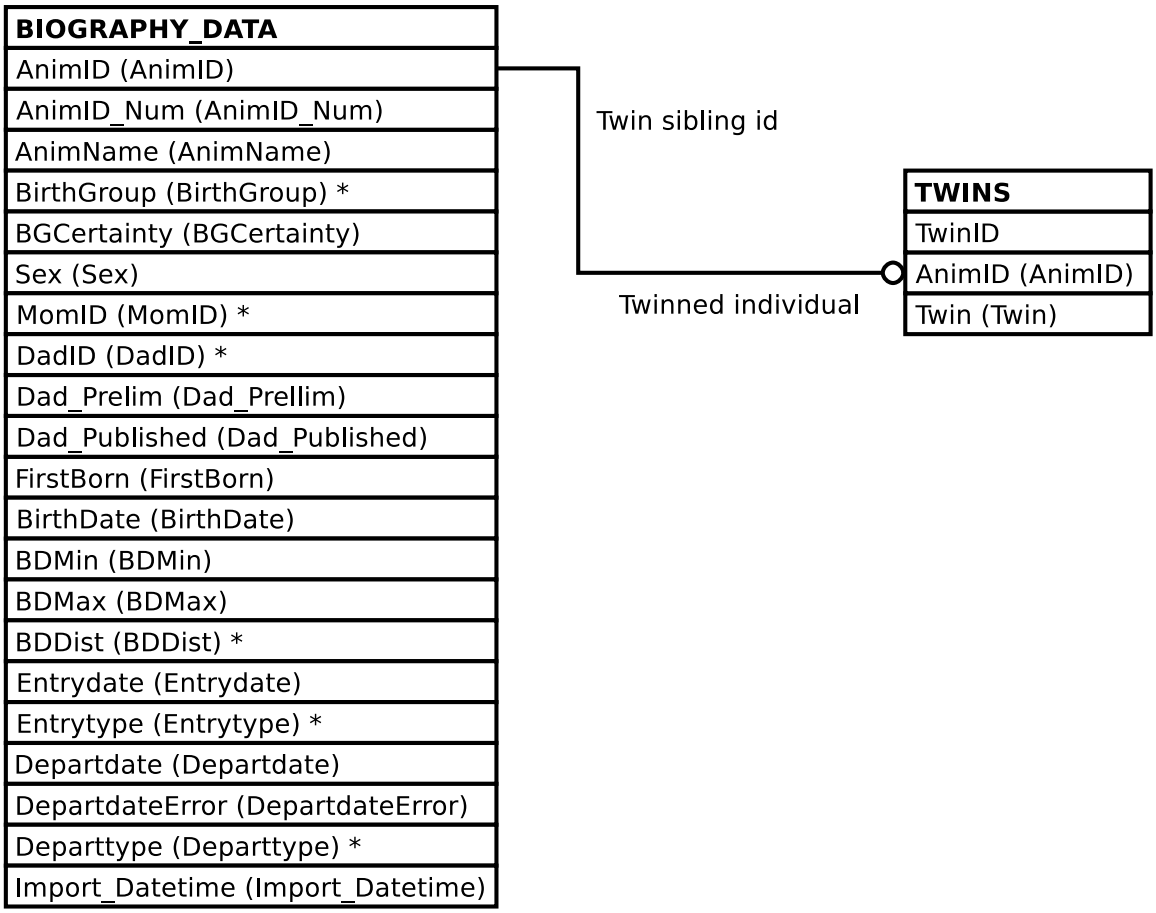


Figure 12: Entity Relationship Diagram of the BIOGRAPHY View

3.1.2 The BIOGRAPHY_UPLOAD View

```
SELECT biography_data.animid::TEXT AS "B_AnimID"
     , biography_data.animid_num::TEXT AS "B_AnimID_num"
     , biography_data.animname::TEXT AS "B_AnimName"
     , biography_data.birthgroup::TEXT AS "B_BirthGroup"
     , biography_data.bgcertainty::TEXT AS "B_BGCertainty"
     , biography_data.sex::TEXT AS "B_Sex"
     , biography_data.momid::TEXT AS "B_MomID"
     , CASE
         WHEN biography_data.dad_prelim THEN
           (biography_data.dadid || 'gmi_dadid_prelim')::TEXT
         ELSE
           biography_data.dadid::TEXT
       END AS "B_DadID"
     , biography_data.dad_published AS "B_DadID_publication_info"
     , biography_data.firstborn::TEXT AS "B_FirstBorn"
     , biography_data.birthdate AS "B_Birthdate"
     , biography_data.bdmin AS "B_BDMin"
     , biography_data.bdmax AS "B_BDMax"
     , biography_data.bddist::TEXT AS "B_BDDist"
     , biography_data.entrydate AS "B_Entrydate"
     , biography_data.entrytype::TEXT AS "B_Entrytype"
     , biography_data.departdate AS "B_Departdate"
     , biography_data.departdateerror AS "B_DepartdateError"
     , biography_data.departtype::TEXT AS "B_Departtype"
  FROM biography_data;
```

Figure 13: Query Defining the BIOGRAPHY_UPLOAD View

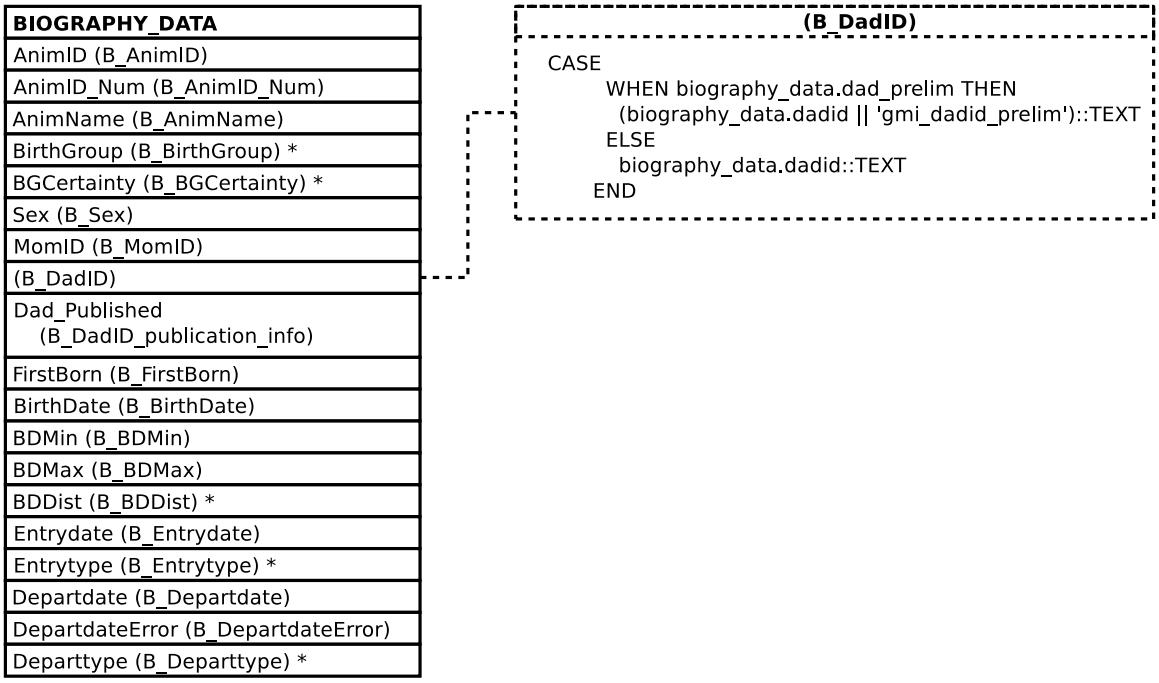


Figure 14: Entity Relationship Diagram of the BIOGRAPHY_UPLOAD View

3.1.3 The CHIMPS View

```
SELECT chimpids.chimpid AS chimpid
     , biography_data.animid AS animid
     , biography_data.animid_num AS animid_num
     , biography_data.animname AS animname
     , biography_data.birthgroup AS birthgroup
     , biography_data.bgcertainty AS bgcertainty
     , biography_data.sex AS sex
     , biography_data.momid AS momid
     , biography_data.dadid AS dadid
     , biography_data.dad_prelim AS dad_prelim
     , biography_data.dad_published AS dad_published
     , biography_data.firstborn AS firstborn
     , biography_data.birthdate AS birthdate
     , biography_data.bdmin AS bdmin
     , biography_data.bdmax AS bdmax
     , biography_data.bddist AS bddist
     , biography_data.entrydate AS entrydate
     , biography_data.entrytype AS entrytype
     , biography_data.departdate AS departdate
     , biography_data.departdateerror AS departdateerror
     , biography_data.departtype AS departtype
     , biography_data.import_datetime AS import_datetime
     , twins.twin AS twin
  FROM biography_data
       JOIN chimpids ON (chimpids.animid = biography_data.animid)
       LEFT OUTER JOIN twins ON (twins.animid = biography_data. ↩
          animid);
```

Figure 15: Query Defining the CHIMPS View

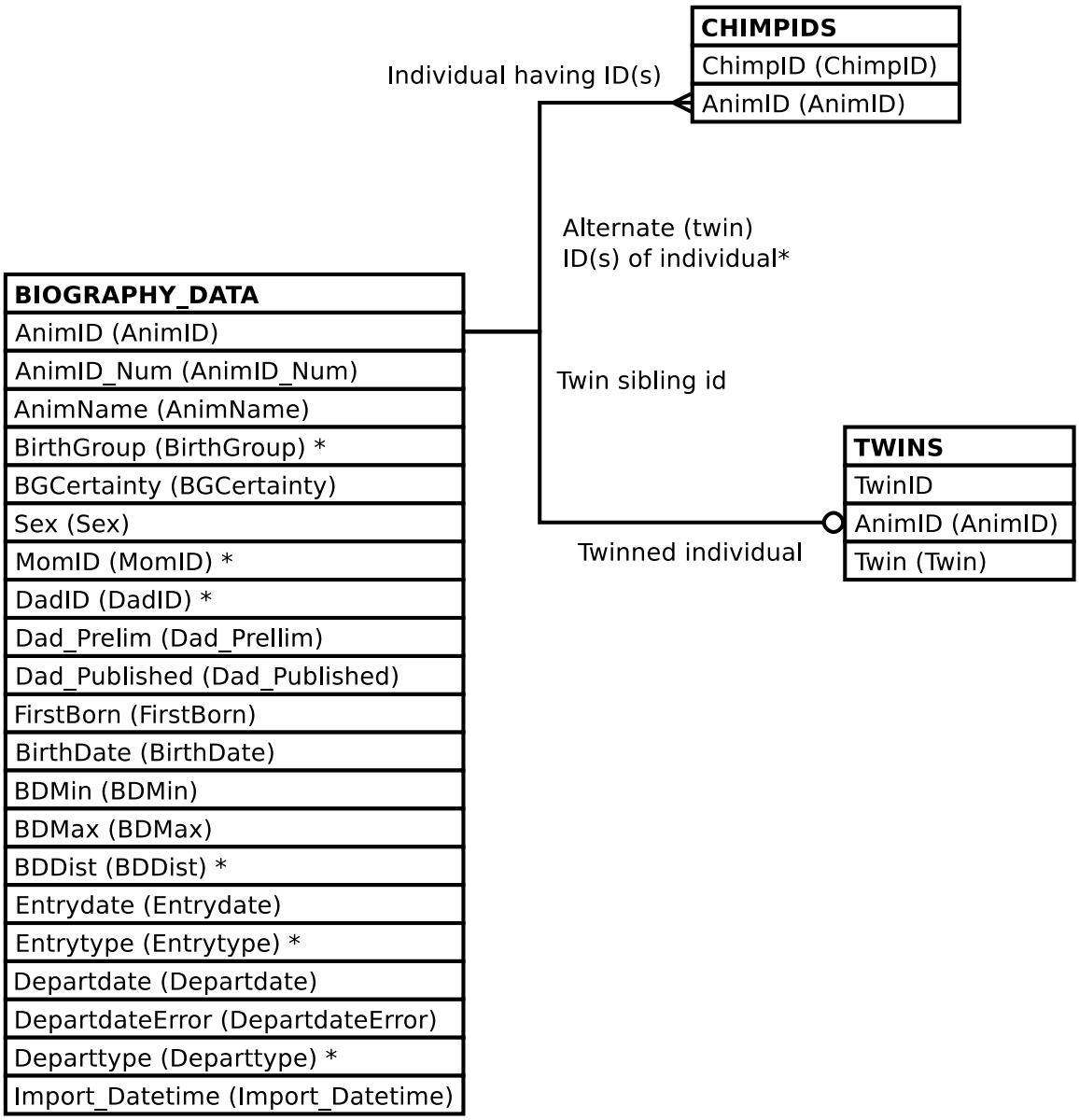


Figure 16: Entity Relationship Diagram of the CHIMPS View

3.1.4 The COMMUNITY_MEMBERSHIP_UPLOAD View

```
SELECT community_membership.animid::TEXT AS cm_b_animid
     , community_membership.start_date AS cm_start_date
     , community_membership.end_date AS cm_end_date
     , community_membership.community_id::TEXT AS cm_cl_community_id
     , community_membership.start_source AS cm_start_source
     , community_membership.end_source AS cm_end_source
  FROM community_membership;
```

Figure 17: Query Defining the COMMUNITY_MEMBERSHIP_UPLOAD View

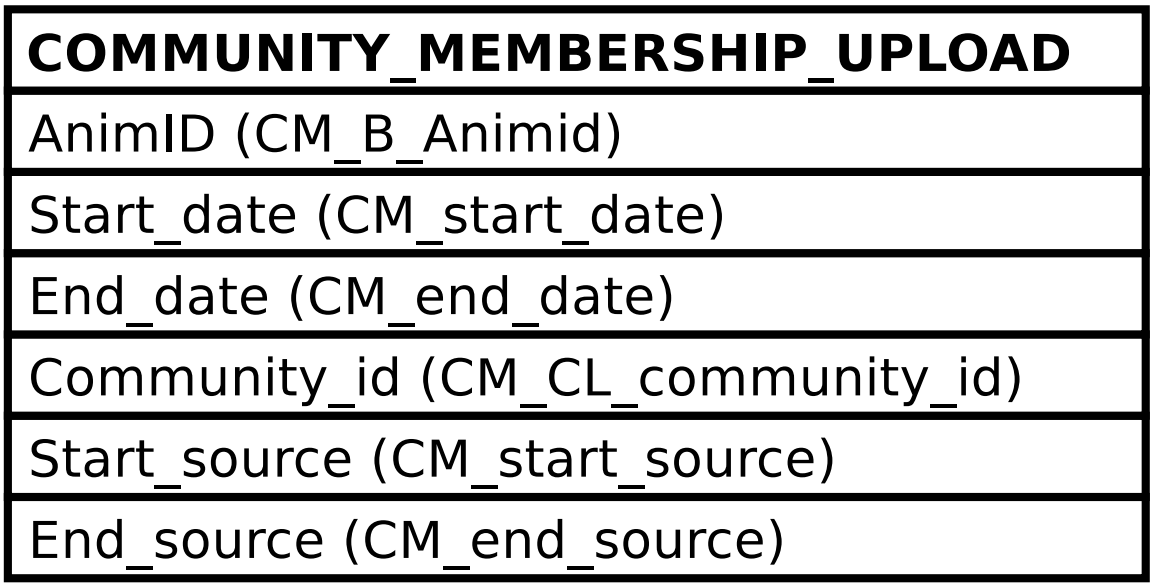


Figure 18: Entity Relationship Diagram of the COMMUNITY_MEMBERSHIP_UPLOAD View

3.2 Follow Mechanics Related Views

3.2.1 The DATASHEETS View

```
SELECT *
  FROM
    (SELECT datasheet_events.followid AS followid
          , datasheet_events.intid AS intid
          , datasheet_events.date AS date
          , datasheet_events.time AS time
          , datasheet_events.mom AS mom
          , datasheet_events.infant AS infant
          , datasheet_events.infant2 AS infant2
          , datasheet_events.sib AS sib
          , datasheet_events.sib2 AS sib2
          , datasheet_events.eid AS eid
          , datasheet_events.behavior AS e_behavior
          , datasheet_events.position AS e_position
          , datasheet_events.uncertain AS e_uncertain
          , datasheet_events.e_mom AS e_mom
          , datasheet_events.e_mompart AS e_mompart
          , datasheet_events.e_mom_foodpart AS e_mom_foodpart
          , datasheet_events.e_mom_foodkind AS e_mom_foodkind
          , datasheet_events.e_mom_fishpart AS e_mom_fishpart
          , datasheet_events.e_mom_fishkind AS e_mom_fishkind
          , datasheet_events.e_mom_fishresult AS e_mom_fishresult
          , datasheet_events.e_mom_voc AS e_mom_voc
          , datasheet_events.e_mom_play AS e_mom_play
          , datasheet_events.e_mom_rest AS e_mom_rest
          , datasheet_events.e_infant AS e_infant
          , datasheet_events.e_infantpart AS e_infantpart
          , datasheet_events.e_infant_foodpart AS e_infant_foodpart
          , datasheet_events.e_infant_foodkind AS e_infant_foodkind
          , datasheet_events.e_infant_fishpart AS e_infant_fishpart
          , datasheet_events.e_infant_fishkind AS e_infant_fishkind
          , datasheet_events.e_infant_fishresult AS  ↩
             e_infant_fishresult
          , datasheet_events.e_infant_voc AS e_infant_voc
          , datasheet_events.e_infant_play AS e_infant_play
          , datasheet_events.e_infant_ride AS e_infant_ride
          , datasheet_events.e_infant_rest AS e_infant_rest
          , datasheet_events.e_sib as e_sib
          , datasheet_events.e_sibpart AS e_sibpart
          , datasheet_events.e_sib_foodpart AS e_sib_foodpart
          , datasheet_events.e_sib_foodkind AS e_sib_foodkind
          , datasheet_events.e_sib_fishpart AS e_sib_fishpart
          , datasheet_events.e_sib_fishkind AS e_sib_fishkind
          , datasheet_events.e_sib_fishresult AS e_sib_fishresult
          , datasheet_events.e_sib_voc AS e_sib_voc
          , datasheet_events.e_sib_play AS e_sib_play
          , datasheet_events.e_sib_ride AS e_sib_ride
          , datasheet_events.e_sib_rest AS e_sib_rest
          , datasheet_events.e_nonfocal AS e_nonfocal
          , datasheet_events.e_nonfocalpart as e_nonfocalpart
          , datasheet_events.e_baboon AS e_baboon
```

```
  , NULL AS d_distid
  , NULL AS d_uncertain
  , NULL AS d_mom
  , NULL AS d_infant
  , NULL AS d_sib
  , NULL AS d_distance
  , datasheet_events.i_comment AS i_comment
  , datasheet_events.e_comment AS e_comment
FROM datasheet_events
```

Figure 20: Query Defining the DATASHEETS View: Part II

```
UNION
SELECT family_dists.followid AS followid
     , family_dists.intid AS intid
     , family_dists.date AS date
     , family_dists.time AS time
     , family_dists.mom AS mom
     , family_dists.infant AS infant
     , family_dists.infant2 AS infant2
     , family_dists.sib AS sib
     , family_dists.sib2 AS sib2
     , NULL AS eid
     , NULL AS e_behavior
     , NULL AS e_position
     , NULL AS e_uncertain
     , NULL AS e_mom
     , NULL AS e_mompart
     , NULL AS e_mom_foodpart
     , NULL AS e_mom_foodkind
     , NULL AS e_mom_fishpart
     , NULL AS e_mom_fishkind
     , NULL AS e_mom_fishresult
     , NULL AS e_mom_voc
     , NULL AS e_mom_play
     , NULL AS e_mom_rest
     , NULL AS e_infant
     , NULL AS e_infantpart
     , NULL AS e_infant_foodpart
     , NULL AS e_infant_foodkind
     , NULL AS e_infant_fishpart
     , NULL AS e_infant_fishkind
     , NULL AS e_infant_fishresult
     , NULL AS e_infant_voc
     , NULL AS e_infant_play
     , NULL AS e_infant_ride
     , NULL AS e_infant_rest
     , NULL AS e_sib
     , NULL AS e_sibpart
     , NULL AS e_sib_foodpart
     , NULL AS e_sib_foodkind
     , NULL AS e_sib_fishpart
     , NULL AS e_sib_fishkind
     , NULL AS e_sib_fishresult
     , NULL AS e_sib_voc
     , NULL AS e_sib_play
     , NULL AS e_sib_ride
     , NULL AS e_sib_rest
     , NULL AS e_nonfocal
     , NULL AS e_nonfocalpart
     , NULL AS e_baboon
     , NULL AS e_baboonpart
```

Figure 21: Query Defining the DATASHEETS View: Part III

```
          , family_dists.distid AS d_distid
          , family_dists.uncertain AS d_uncertain
          , family_dists.d_mom AS d_mom
          , family_dists.d_infant AS d_infant
          , family_dists.d_sib AS d_sib
          , family_dists.distance AS d_distance
          , family_dists.i_comment AS i_comment
          , NULL AS e_comment
       FROM family_dists
  ) AS q
ORDER BY q.date
       , q.time
       , q.e_behavior
       , q.e_mom IS NULL
       , q.e_infant IS NULL
       , q.e_nonfocal
       , q.d_mom IS NULL
       , q.d_infant IS NULL;
```

Figure 22: Query Defining the DATASHEETS View: Part IV

The DATASHEETS view ER diagram intentionally omitted due to query size.

Figure 23: Entity Relationship Diagram of the DATASHEETS View

3.2.2 The UPLOAD_BEHAVIORS View

```
SELECT NULL::TEXT AS followtype
     , NULL::DATE AS entrydate
     , NULL::TEXT as entered_by
     , NULL::TEXT AS researcher
     , NULL::DATE AS date
     , NULL::INT AS page
     , NULL::TEXT AS location
     , NULL::TEXT AS mother
     , NULL::TEXT AS infant
     , NULL::TEXT AS sibling
     , NULL::TEXT AS time_on_sheet
     , NULL::TIME AS time_converted_mil
     , NULL::TEXT AS time_type
     , NULL::TEXT AS bad_observation
     , NULL::TEXT AS rain
     , NULL::TEXT AS ma_behavior
     , NULL::TEXT AS ma_foodpart
     , NULL::TEXT AS ma_foodkind
     , NULL::TEXT AS ma_fishpart
     , NULL::TEXT AS ma_fishkind
     , NULL::BOOLEAN AS ma_gp_combined
     , NULL::TEXT AS ma_groomingplay
     , NULL::TEXT AS ma_grooming
     , NULL::TEXT AS ma_play
     , NULL::TEXT AS ma_groomed_by
     , NULL::TEXT AS ma_voc
```

Figure 24: Query Defining the UPLOAD_BEHAVIORS View: Part I

```
    , NULL::TEXT AS inf_behavior
    , NULL::TEXT AS inf_foodpart
    , NULL::TEXT AS inf_foodkind
    , NULL::TEXT AS inf_fishpart
    , NULL::TEXT AS inf_fishkind
    , NULL::TEXT AS inf_distance_to_ma
    , NULL::TEXT AS inf_nurse
    , NULL::TEXT AS inf_travel
    , NULL::TEXT AS inf_eat
    , NULL::TEXT AS inf_solitary_play
    , NULL::TEXT AS inf_rest
    , NULL::TEXT AS inf_distance_1
    , NULL::TEXT AS inf_distance_2
    , NULL::TEXT AS inf_distance_3
    , NULL::TEXT AS inf_distance_4
    , NULL::TEXT AS inf_distance_5
    , NULL::TEXT AS inf_distance_over_5m
    , NULL::TEXT AS inf_ride_on_belly
    , NULL::TEXT AS inf_ride_on_back
    , NULL::TEXT AS inf_ride_on_unk
    , NULL::TEXT AS inf_dangle
    , NULL::TEXT AS inf_rejection
    , NULL::TEXT AS inf_touching_another
    , NULL::TEXT AS inf_grooming
    , NULL::TEXT AS inf_being_groomed
    , NULL::TEXT AS inf_social_play
    , NULL::TEXT AS inf_display
    , NULL::TEXT AS inf_cry
    , NULL::TEXT AS inf_voc
    , NULL::TEXT AS inf_beg
    , NULL::TEXT AS inf_assr
    , NULL::TEXT AS inf_sex_related
    , NULL::TEXT AS inf_tooluse
    , NULL::TEXT AS inf_directaggression
    , NULL::TEXT AS sib_bad_observation
    , NULL::TEXT AS sib_behavior
    , NULL::TEXT AS sib_foodpart
    , NULL::TEXT AS sib_foodkind
    , NULL::TEXT AS sib_fishpart
    , NULL::TEXT AS sib_fishkind
    , NULL::TEXT AS sib_play
    , NULL::TEXT AS sib_grooming
    , NULL::TEXT AS sib_groomed_by
    , NULL::TEXT AS sib_ride
    , NULL::TEXT AS sib_distance_to_ma
    , NULL::TEXT AS sib_distance_to_infant
    , NULL::TEXT AS sib_voc
    , NULL::TEXT AS comments
WHERE _raise_gombemi_exception(
        'Cannot select UPLOAD_BEHAVIORS'
        || ': The only use of the UPLOAD_BEHAVIORS view is to ↩
           insert'
        || ' new data into the 1-minute interval related portion ↩
           of'
        || ' Gombe-MI');
```

The UPLOAD_BEHAVIORS view is used only to insert follow data into Gombe-MI. Since it cannot be queried and the semantics are complicated it has no ER diagram.

Figure 26: Entity Relationship Diagram of the UPLOAD_BEHAVIORS View

3.3 Follow Event Related Views

3.3.1 The ACTOR_ACTEES View

```
SELECT events.eid AS eid
     , events.intid AS intid
     , actors.pid AS actorpid
     , actees.pid AS acteepid
     , events.behavior AS behavior
     , events.position AS position
     , events.uncertain AS uncertain
     , followrole(intervals.followid, actors.participant) AS  ↩
        actorrole
     , actors.participant AS actor
     , followrole(intervals.followid, actees.participant) AS  ↩
        acteerole
     , actees.participant AS actee
     , events.comment AS comment
  FROM events
         JOIN intervals
           ON (intervals.intid = events.intid)
         JOIN parts AS actors
           ON (actors.eid = events.eid)
         JOIN parts AS actees
           ON (actees.eid = events.eid)
  WHERE actors.part = 'gmi_actor'
        AND actees.part = 'gmi_actee';
```

Figure 27: Query Defining the ACTOR_ACTEES View

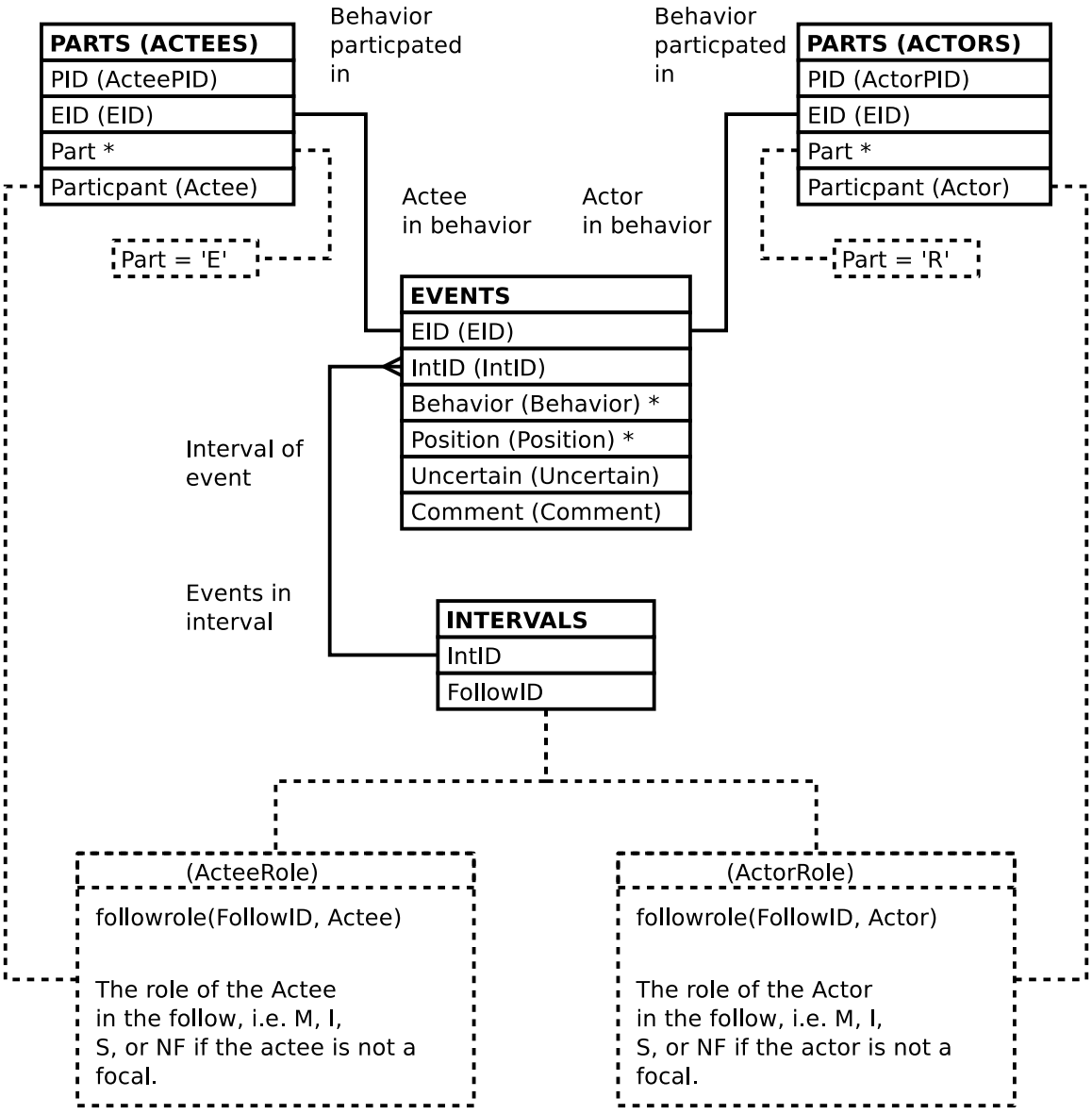


Figure 28: Entity Relationship Diagram of the ACTOR_ACTEES View

3.3.2 The ACTOR_ACTEES_W_BABOONS View

```
SELECT events.eid AS eid
     , events.intid AS intid
     , actors.pid AS actorpid
     , actees.pid AS acteepid
     , events.behavior AS behavior
     , events.position AS position
     , events.uncertain AS uncertain
     , followrole(intervals.followid, actors.participant) AS  ↩
        actorrole
     , FALSE AS baboonactor
     , actors.participant AS actor
     , followrole(intervals.followid, actees.participant) AS  ↩
        acteerole
     , FALSE AS baboonactee
     , actees.participant AS actee
     , events.comment AS comment
  FROM events
         JOIN intervals
           ON (intervals.intid = events.intid)
         JOIN parts AS actors
           ON (actors.eid = events.eid)
         JOIN parts AS actees
           ON (actees.eid = events.eid)
  WHERE actors.part = 'gmi_actor'
        AND actees.part = 'gmi_actee'
UNION
SELECT events.eid AS eid
     , events.intid AS intid
     , actors.pid AS actorpid
     , NULL AS acteepid
     , events.behavior AS behavior
     , events.position AS position
     , events.uncertain AS uncertain
     , followrole(intervals.followid, actors.participant) AS  ↩
        actorrole
     , FALSE AS baboonactor
     , actors.participant AS actor
     , 'NF' AS acteerole
     , TRUE AS baboonactee
     , NULL::VARCHAR(gmi_animid_len) AS actee
     , events.comment AS comment
  FROM events
         JOIN intervals
           ON (intervals.intid = events.intid)
         JOIN parts AS actors
           ON (actors.eid = events.eid)
  WHERE actors.part = 'gmi_actor'
        AND events.baboon
```

Figure 29: Query Defining the ACTOR_ACTEES_W_BABOONS View: Part I

```
UNION
SELECT events.eid AS eid
     , events.intid AS intid
     , NULL AS actorpid
     , actees.pid AS acteepid
     , events.behavior AS behavior
     , events.position AS position
     , events.uncertain AS uncertain
     , 'NF' AS actorrole
     , TRUE AS baboonactor
     , NULL::VARCHAR(gmi_animid_len) AS actor
     , followrole(intervals.followid, actees.participant) AS  ↩
        acteerole
     , FALSE AS baboonactee
     , actees.participant AS actee
     , events.comment AS comment
  FROM events
         JOIN intervals
           ON (intervals.intid = events.intid)
         JOIN parts AS actees
           ON (actees.eid = events.eid)
  WHERE actees.part = 'gmi_actee'
        AND events.baboon;
```

Figure 30: Query Defining the ACTOR_ACTEES_W_BABOONS View: Part II

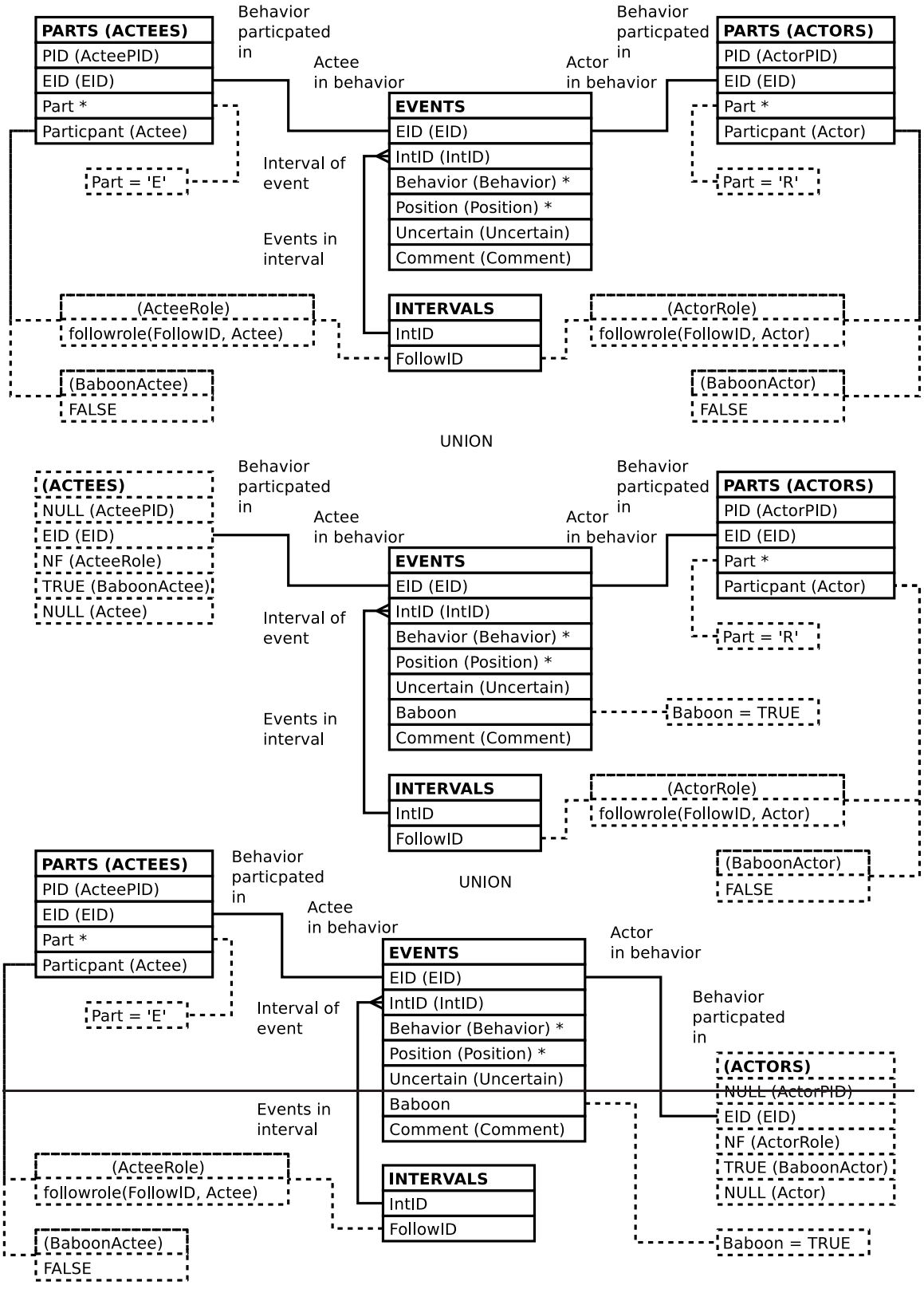


Figure 31: Entity Relationship Diagram of the ACTOR_ACTEES_W_BABOONS View

3.3.3 The ASOCIAL_EVENTS View

```
SELECT events.eid AS eid
     , events.intid AS intid
     , parts.pid AS pid
     , events.behavior AS behavior
     , events.position AS position
     , events.uncertain AS uncertain
     , followparts.role AS role
     , parts.participant AS participant
     , events.comment AS comment
  FROM events
         JOIN intervals
           ON (intervals.intid = events.intid)
         JOIN parts
           ON (parts.eid = events.eid)
         JOIN followparts
           ON (followparts.followid = intervals.followid)
         JOIN chimpids AS p_ids
           ON (p_ids.chimpid = parts.participant)
         JOIN chimpids AS fp_ids
           ON (fp_ids.chimpid = followparts.chimpid)
  WHERE parts.part = 'gmi_self'
        AND fp_ids.animid = p_ids.animid;
```

Figure 32: Query Defining the ASOCIAL_EVENTS View

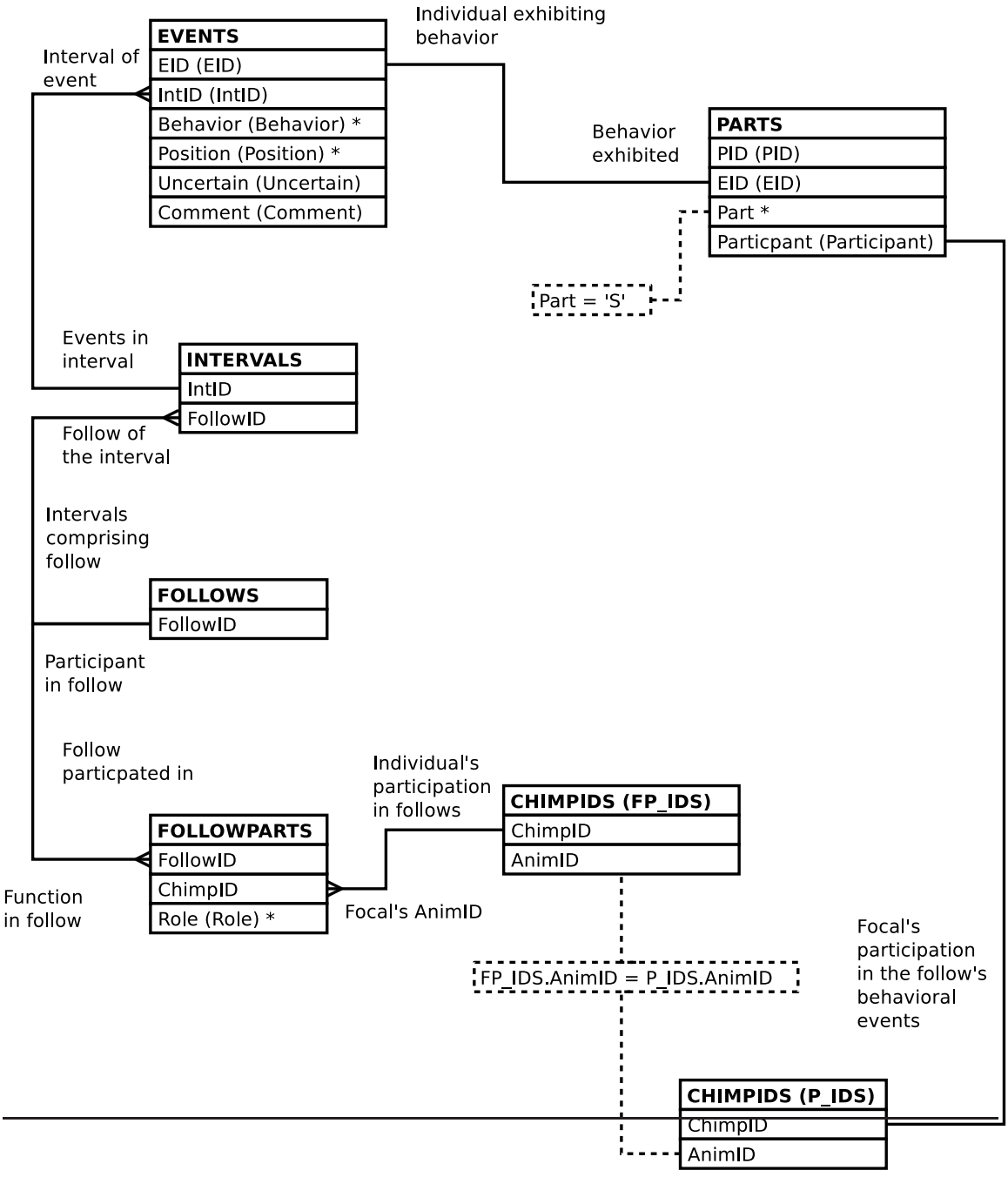


Figure 33: Entity Relationship Diagram of the ASOCIAL_EVENTS View

3.3.4 The EVENT_BOUTS View

```
-- Note that we don't worry about leap seconds given
-- the time values in the db.

WITH detail AS
     -- Compute mins_in and mins_out
     (SELECT bouts.followid AS followid
           , bouts.intid AS intid
           , bouts.function AS function
           , bouts.animid AS animid
           , bouts.role AS role
           , bouts.chimpid AS chimpid
           , bouts.bout AS bout
           , bouts.behavior AS behavior
           , bouts.uncertain AS uncertain
           , bouts.time AS time
           , (EXTRACT(EPOCH FROM bouts.time)
                - EXTRACT(EPOCH FROM FIRST_VALUE(bouts.time) OVER  ↩
                   bout_w)
             ) / 60
             + 1
             AS mins_in
           , (EXTRACT(EPOCH FROM LAST_VALUE(bouts.time)
                                 OVER (bout_w RANGE BETWEEN  ↩
                                    CURRENT ROW
                                              AND UNBOUNDED  ↩
                                                 FOLLOWING))
                - EXTRACT(EPOCH FROM bouts.time)
             ) / 60
             AS mins_out
        FROM
          -- Compute bout number by summing transition ticks
          (SELECT ticks.followid AS followid
                , ticks.intid AS intid
                , ticks.function AS function
                , ticks.animid AS animid
                , ticks.role AS role
                , ticks.chimpid AS chimpid
                , ticks.behavior AS behavior
                , ticks.uncertain AS uncertain
                , ticks.time AS time
                , SUM(ticks.bouttick)
                    OVER (PARTITION BY ticks.followid
                                     , ticks.behavior
                                     , ticks.animid
                          ORDER BY ticks.time
                                 , ticks.bouttick DESC)
                  AS bout
```

Figure 34: Query Defining the EVENT_BOUTS View: Part I

```
FROM
  -- Distinguish one bout from another
  (SELECT intervals.followid AS followid
        , intervals.intid AS intid
        , followroles.function AS function
        , part_cids.animid AS animid
        , followparts.role AS role
        , followparts.chimpid AS chimpid
        , events.behavior AS behavior
        , BOOL_AND(events.uncertain) AS uncertain
        , intervals.time AS time
        , CASE
            WHEN LAG(MIN(intervals.time)) OVER tick_w
                   = MIN(intervals.time) - '1 minute':: ↩
                      INTERVAL
              THEN 0
            ELSE
              1
          END
          AS bouttick
     FROM intervals
            JOIN events
                 ON (events.intid = intervals.intid)
            JOIN parts
                 ON (parts.eid = events.eid)
            JOIN chimpids AS part_cids
                 ON (part_cids.chimpid = parts. ↩
                    participant)
            JOIN chimpids AS fp_cids
                 ON (fp_cids.animid = part_cids.animid)
            JOIN followparts
                 ON (followparts.followid = intervals. ↩
                    followid
                     AND followparts.chimpid = fp_cids. ↩
                        chimpid)
            JOIN followroles
                 ON (followroles.role = followparts. ↩
                    role)
     WHERE NOT(events.baboon)
     GROUP BY intervals.followid
        , intervals.intid
        , part_cids.animid
        , events.behavior
        -- The values that are unique per above anyway
        , followroles.function
        , followparts.role
        , followparts.chimpid
        , intervals.time
     WINDOW tick_w AS (PARTITION BY intervals.followid
                                  , events.behavior
                                  , part_cids.animid
                       ORDER BY intervals.time)
  ) AS ticks
) AS bouts
WINDOW bout_w AS (PARTITION BY bouts.followid
```

```
-- Distribute aggregated bout info over the bout's minutes
SELECT detail.*
     , summary.prior_gap AS prior_gap
     , summary.gap_noobs AS gap_noobs
     , summary.left_censored AS left_censored
     , summary.right_censored AS right_censored
     , summary.day_start AS day_start
     , summary.day_end AS day_end
  FROM
    detail
      JOIN
      (-- Aggregate per-bout information
       SELECT bout_ends.followid AS followid
            , bout_ends.animid AS animid
            , bout_ends.behavior AS behavior
            , bout_ends.bout AS bout
            , CASE
                WHEN bout_ends.bout = 1 THEN
                  NULL
                ELSE
                  SUM(bout_ends.prior_gap)
              END
              AS prior_gap
            , CASE
                WHEN bout_ends.bout = 1 THEN
                  NULL
                ELSE
                  SUM(bout_ends.prior_gap)
                    -- Subtract number of minutes the focal had
                    -- any behaviors during the gap.
                    - (SELECT COUNT(DISTINCT intervals.intid)
                         FROM intervals
                                JOIN events
                                     ON (events.intid = intervals. ↩
                                         intid)
                                JOIN parts
                                     ON (parts.eid = events.eid)
                                JOIN chimpids
                                     ON (chimpids.chimpid
                                          = parts.participant)
                        WHERE intervals.followid = bout_ends. ↩
                            followid
                           AND MIN(bout_ends.time) > intervals ↩
                               .time
                           AND intervals.time
                               >= MIN(bout_ends.time)
                                    - (SUM(bout_ends.prior_gap ↩
                                        )
                                       || ' minutes')::  ↩
                                           INTERVAL
                           AND chimpids.animid = bout_ends. ↩
                               animid)
              END
              AS gap_noobs
            , BOOL_OR(bout_ends.left_censored) AS left_censored
```

```
, CASE
    WHEN bout_ends.bout = 1 THEN
      NOT EXISTS
            -- Behavioral data on the focal before the ↩
                bout
            (SELECT 1
               FROM intervals
                      JOIN events
                           ON (events.intid =  ↩
                              intervals.intid)
                      JOIN parts
                           ON (parts.eid = events.eid)
                      JOIN chimpids
                           ON (chimpids.chimpid
                               = parts.participant)
               WHERE intervals.followid = bout_ends. ↩
                  followid
                     AND intervals.time < MIN( ↩
                        bout_ends.time)
                     AND chimpids.animid = bout_ends. ↩
                        animid
               LIMIT 1)
    ELSE
      FALSE
  END
  AS day_start
, CASE
    WHEN bout_ends.bout = LAST_VALUE(bout_ends.bout)
                            OVER (PARTITION BY
                                    bout_ends.followid
                                  , bout_ends.animid
                                  , bout_ends.behavior
                                  ORDER BY bout_ends. ↩
                                     bout
                                  RANGE BETWEEN  ↩
                                     CURRENT ROW
                                        AND UNBOUNDED  ↩
                                           FOLLOWING)
      THEN
      NOT EXISTS
            -- Behavioral data on the focal after the  ↩
               bout
            (SELECT 1
               FROM intervals
                      JOIN events
                           ON (events.intid =  ↩
                              intervals.intid)
                      JOIN parts
                           ON (parts.eid = events.eid)
                      JOIN chimpids
                           ON (chimpids.chimpid
                               = parts.participant)
               WHERE intervals.followid = bout_ends. ↩
                  followid
                     AND intervals.time
```

```
FROM
  -- Examine the bout's endpoints to compute prior_gap,  ↩
     and
  -- left and right censoring.
  (SELECT detail.followid
        , detail.animid
        , detail.behavior
        , detail.bout
        , detail.mins_in AS mins_in
        , detail.mins_out AS mins_out
        , CASE
            WHEN detail.mins_in = 1 THEN
              (EXTRACT(EPOCH FROM detail.time)
                 - EXTRACT(EPOCH FROM
                           LAG(detail.time)
                             OVER (PARTITION BY
                                     detail.followid
                                   , detail.animid
                                   , detail.behavior
                                   ORDER BY detail.time))
              ) / 60 - 1
            ELSE
              0
          END
          AS prior_gap
        , CASE
            WHEN detail.mins_in = 1 THEN
              NOT EXISTS
                (SELECT 1
                   FROM intervals AS i
                          JOIN events AS e
                               ON (e.intid = i.intid)
                          JOIN parts AS p
                               ON (p.eid = e.eid)
                          JOIN chimpids AS p_cids
                               ON (p_cids.chimpid = p. ↩
                                  participant)
                   WHERE i.followid = detail.followid
                         AND i.time = detail.time
                                      - '1 minute':: ↩
                                         INTERVAL
                         AND p_cids.animid = detail. ↩
                            animid
                         LIMIT 1)
            ELSE
              FALSE
            END
          AS left_censored
```

Figure 38: Query Defining the EVENT_BOUTS View: Part V

```
          , CASE
              WHEN detail.mins_out = 0 THEN
                NOT EXISTS
                  (SELECT 1
                     FROM intervals AS i
                            JOIN events AS e
                                 ON (e.intid = i.intid)
                            JOIN parts AS p
                                 ON (p.eid = e.eid)
                            JOIN chimpids AS p_cids
                                 ON (p_cids.chimpid = p. ↩
                                    participant)
                     WHERE i.followid = detail.followid
                           AND i.time = detail.time
                                        + '1 minute':: ↩
                                           INTERVAL
                           AND p_cids.animid = detail. ↩
                              animid
                           LIMIT 1)
              ELSE
                FALSE
              END
            AS right_censored
          , detail.time AS time
       FROM detail
       WHERE detail.mins_in = 1
             OR detail.mins_out = 0
   ) AS bout_ends
 GROUP BY bout_ends.followid
        , bout_ends.animid
        , bout_ends.behavior
        , bout_ends.bout)
AS summary
ON (summary.followid = detail.followid
    AND summary.animid = detail.animid
    AND summary.behavior = detail.behavior
    AND summary.bout = detail.bout);
```

Figure 39: Query Defining the EVENT_BOUTS View: Part VI

ER diagram intentionally omitted.

Figure 40: Entity Relationship Diagram of the EVENT_BOUTS View

3.3.5 The EVENT_BOUT_AGGS View

```
SELECT event_bouts.followid AS followid
     , event_bouts.function AS function
     , event_bouts.animid AS animid
     , event_bouts.role AS role
     , event_bouts.chimpid AS chimpid
     , event_bouts.bout AS bout
     , event_bouts.behavior AS behavior
     , BOOL_OR(event_bouts.uncertain) AS uncertain
     , MIN(event_bouts.time) AS start_time
     , MAX(event_bouts.time) AS stop_time
     , event_bouts.prior_gap AS prior_gap
     , event_bouts.gap_noobs AS gap_noobs
     , event_bouts.left_censored AS left_censored
     , event_bouts.right_censored AS right_censored
     , event_bouts.day_start AS day_start
     , event_bouts.day_end AS day_end
  FROM event_bouts
  GROUP BY event_bouts.followid
         , event_bouts.animid
         , event_bouts.bout
         , event_bouts.behavior
         -- The values that are unique per above anyway
         , event_bouts.function
         , event_bouts.role
         , event_bouts.chimpid
         , event_bouts.prior_gap
         , event_bouts.gap_noobs
         , event_bouts.left_censored
         , event_bouts.right_censored
         , event_bouts.day_start
         , event_bouts.day_end;
```

Figure 41: Query Defining the EVENT_BOUT_AGGS View

ER diagram intentionally omitted.

Figure 42: Entity Relationship Diagram of the EVENT_BOUT_AGGS View

3.3.6 The FAMILY_EVENTS View

```
SELECT focal_follows.followid AS followid
     , intervals.intid AS intid
     , events.eid AS eid
     , moms.pid AS e_mompid
     , infants.pid AS e_infantpid
     , sibs.pid AS e_sibpid
     , nfs.pid AS e_nonfocalpid
     , focal_follows.date AS date
     , intervals.time AS time
     , focal_follows.mom AS mom
     , focal_follows.infant AS infant
     , focal_follows.infant2 AS infant2
     , focal_follows.sib AS sib
     , focal_follows.sib2 AS sib2
     , events.behavior AS behavior
     , events.position AS position
     , events.uncertain AS uncertain
     , moms.participant AS e_mom
     , moms.part AS e_mompart
     , infants.participant AS e_infant
     , infants.part AS e_infantpart
     , sibs.participant AS e_sib
     , sibs.part AS e_sibpart
     , nfs.participant AS e_nonfocal
     , nfs.part AS e_nonfocalpart
     , events.baboon AS e_baboon
     , CASE
         WHEN events.baboon THEN
           CASE
             WHEN moms.part = 'gmi_actor'
                  OR infants.part = 'gmi_actor'
                  OR sibs.part = 'gmi_actor' THEN
               'gmi_actee'
             WHEN moms.part = 'gmi_actee'
                  OR infants.part = 'gmi_actee'
                  OR sibs.part = 'gmi_actee' THEN
               'gmi_actor'
             ELSE
               'gmi_participant'
           END
         ELSE
           NULL
       END::parts_part AS e_baboonpart
     , events.comment AS e_comment
     , intervals.comment AS i_comment
```

Figure 43: Query Defining the FAMILY_EVENTS View: Part I

```
FROM focal_follows
       JOIN intervals
         ON (intervals.followid = focal_follows.followid)
       JOIN events
         ON (events.intid = intervals.intid)
       LEFT OUTER JOIN parts AS moms
                    ON (moms.eid = events.eid
                        AND followfunc(intervals.followid
                                       , moms.participant)
                            = 'gmi_mom')
       LEFT OUTER JOIN parts AS infants
                    ON (infants.eid = events.eid
                        AND followfunc(intervals.followid
                                       , infants.participant)
                            = 'gmi_infant')
       LEFT OUTER JOIN parts AS sibs
                    ON (sibs.eid = events.eid
                        AND followfunc(intervals.followid
                                       , sibs.participant)
                            = 'gmi_sib')
       LEFT OUTER JOIN parts AS nfs
                    ON (nfs.eid = events.eid
                        AND followfunc(intervals.followid
                                       , nfs.participant)
                            = 'gmi_nonfocal');
```

Figure 44: Query Defining the FAMILY_EVENTS View: Part II

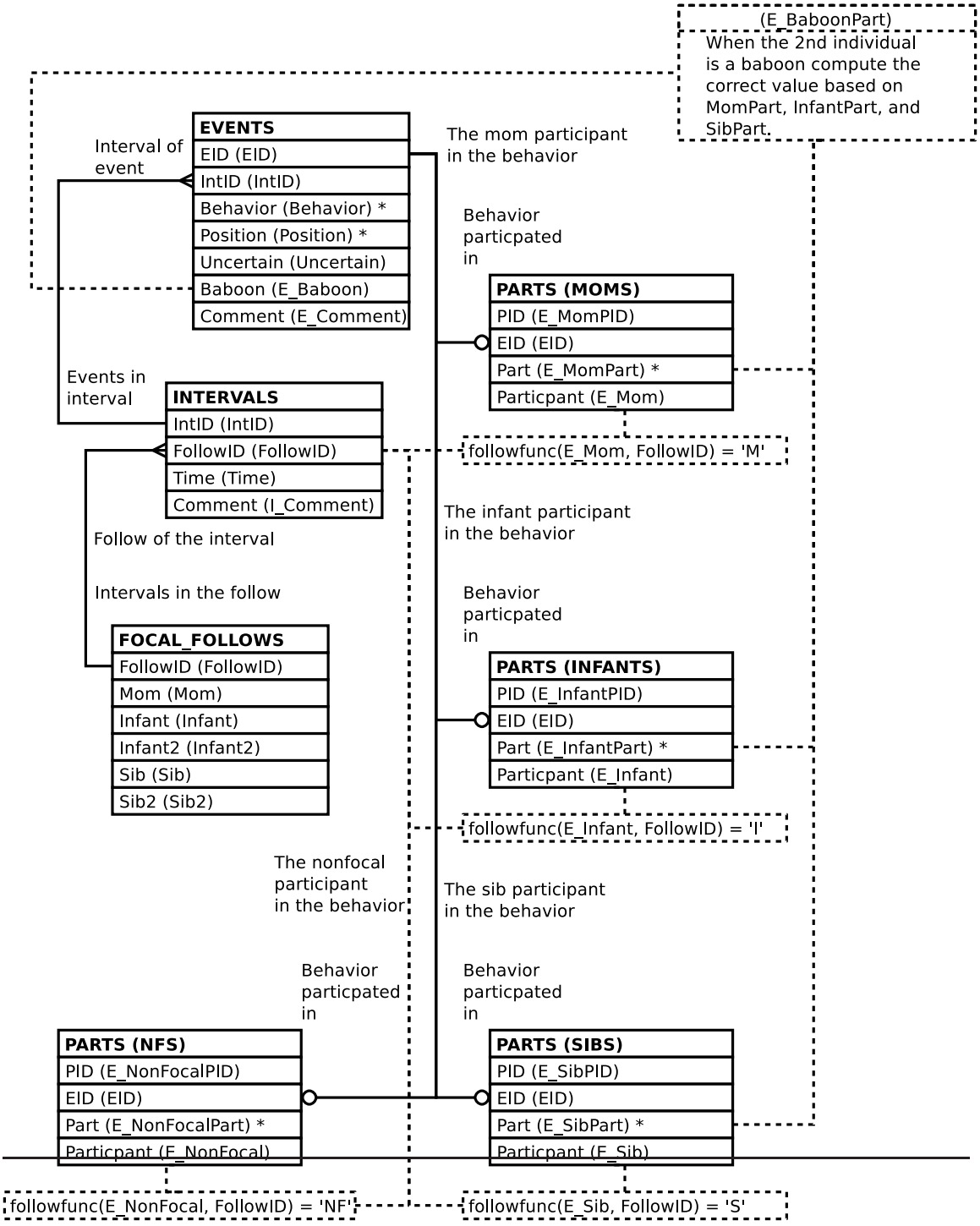


Figure 45: Entity Relationship Diagram of the FAMILY_EVENTS View

3.3.7 The FOCAL_EVENTS View

```
SELECT events.eid AS eid
     , events.intid AS intid
     , focals.pid AS focalpid
     , otherpart.pid AS otherpid
     , events.behavior AS behavior
     , events.position AS position
     , events.uncertain AS uncertain
     , followparts.role AS focalrole
     , focals.participant AS focal
     , focals.part AS focalpart
     , CASE
         WHEN events.baboon THEN
           'gmi_nonfocal'
         ELSE
           followrole(intervals.followid, otherpart.participant)
       END AS otherrole
     , otherpart.participant AS other
     , CASE
         WHEN events.baboon THEN
           CASE
             WHEN focals.part = 'gmi_actor' THEN
               'gmi_actee'
             WHEN focals.part = 'gmi_actee' THEN
               'gmi_actor'
             ELSE
               'gmi_participant'
           END::parts_part
         ELSE
           otherpart.part
       END AS otherpart
     , events.baboon AS baboon
     , events.comment AS comment
  FROM events
         JOIN intervals
           ON (intervals.intid = events.intid)
         JOIN followparts
           ON (followparts.followid = intervals.followid)
         JOIN chimpids AS fp_ids
           ON (fp_ids.chimpid = followparts.chimpid)
         JOIN parts AS focals
           ON (focals.eid = events.eid)
         JOIN chimpids AS focal_ids
           ON (focal_ids.chimpid = focals.participant)
         LEFT OUTER JOIN parts AS otherpart
                      ON (otherpart.eid = events.eid
                          AND focals.pid IS DISTINCT FROM otherpart. ↩
                             pid)
  WHERE focal_ids.animid = fp_ids.animid;
```

Figure 46: Query Defining the FOCAL_EVENTS View

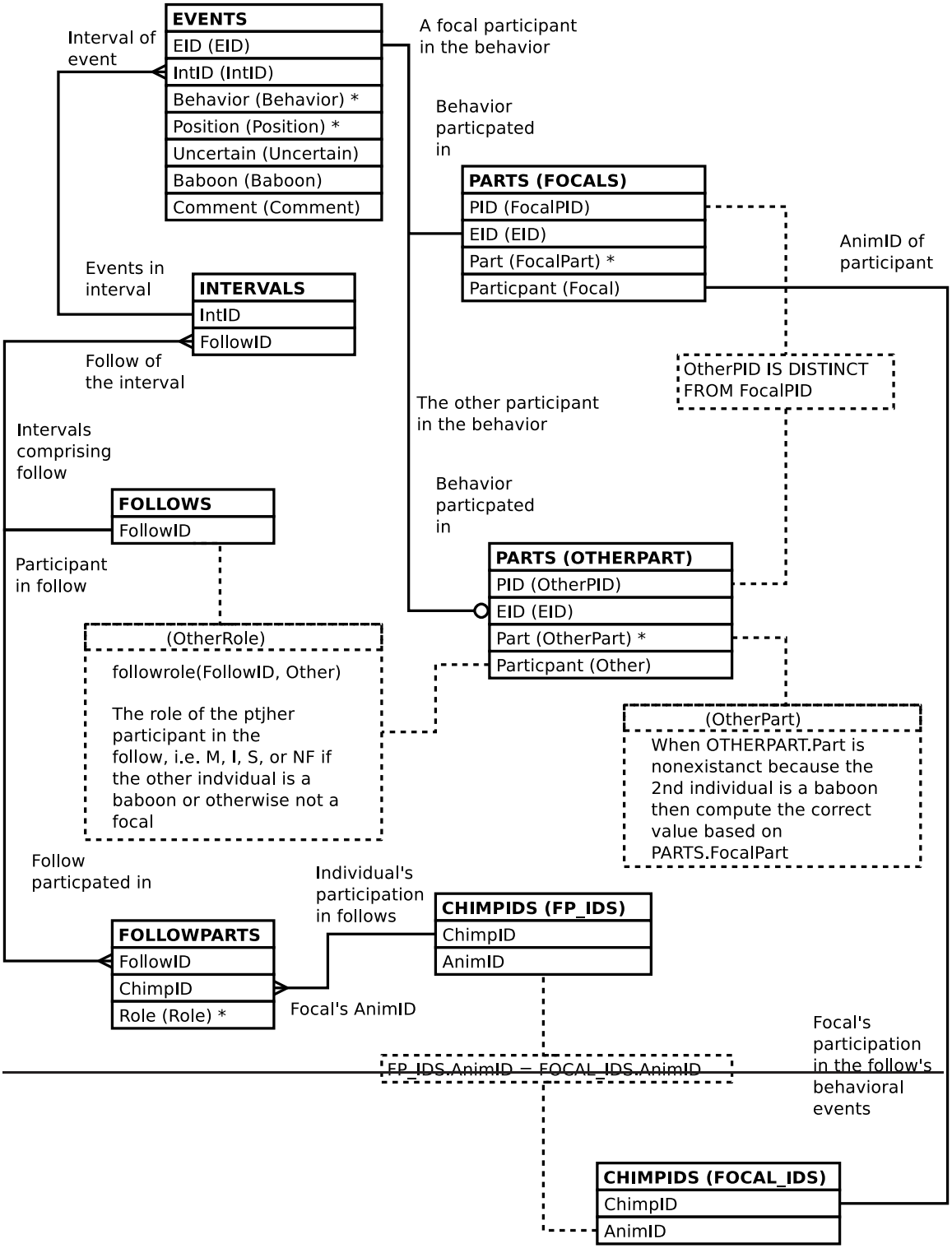


Figure 47: Entity Relationship Diagram of the FOCAL_EVENTS View

3.3.8 The FOCAL_SOCIAL_EVENTS View

```
SELECT events.eid AS eid
     , events.intid AS intid
     , focals.pid AS focalpid
     , otherpart.pid AS otherpid
     , events.behavior AS behavior
     , events.position AS position
     , events.uncertain AS uncertain
     , followparts.role AS focalrole
     , focals.participant AS focal
     , focals.part AS focalpart
     , followrole(intervals.followid, otherpart.participant) AS  ↩
        otherrole
     , otherpart.participant AS other
     , otherpart.part AS otherpart
     , events.comment AS comment
  FROM events
         JOIN intervals
           ON (intervals.intid = events.intid)
         JOIN followparts
           ON (followparts.followid = intervals.followid)
         JOIN chimpids AS fp_ids
           ON (fp_ids.chimpid = followparts.chimpid)
         JOIN parts AS focals
           ON (focals.eid = events.eid)
         JOIN chimpids AS focal_ids
           ON (focal_ids.chimpid = focals.participant)
         JOIN parts AS otherpart
           ON (otherpart.eid = events.eid)
  WHERE focal_ids.animid = fp_ids.animid
        AND focals.pid <> otherpart.pid;
```

Figure 48: Query Defining the FOCAL_SOCIAL_EVENTS View

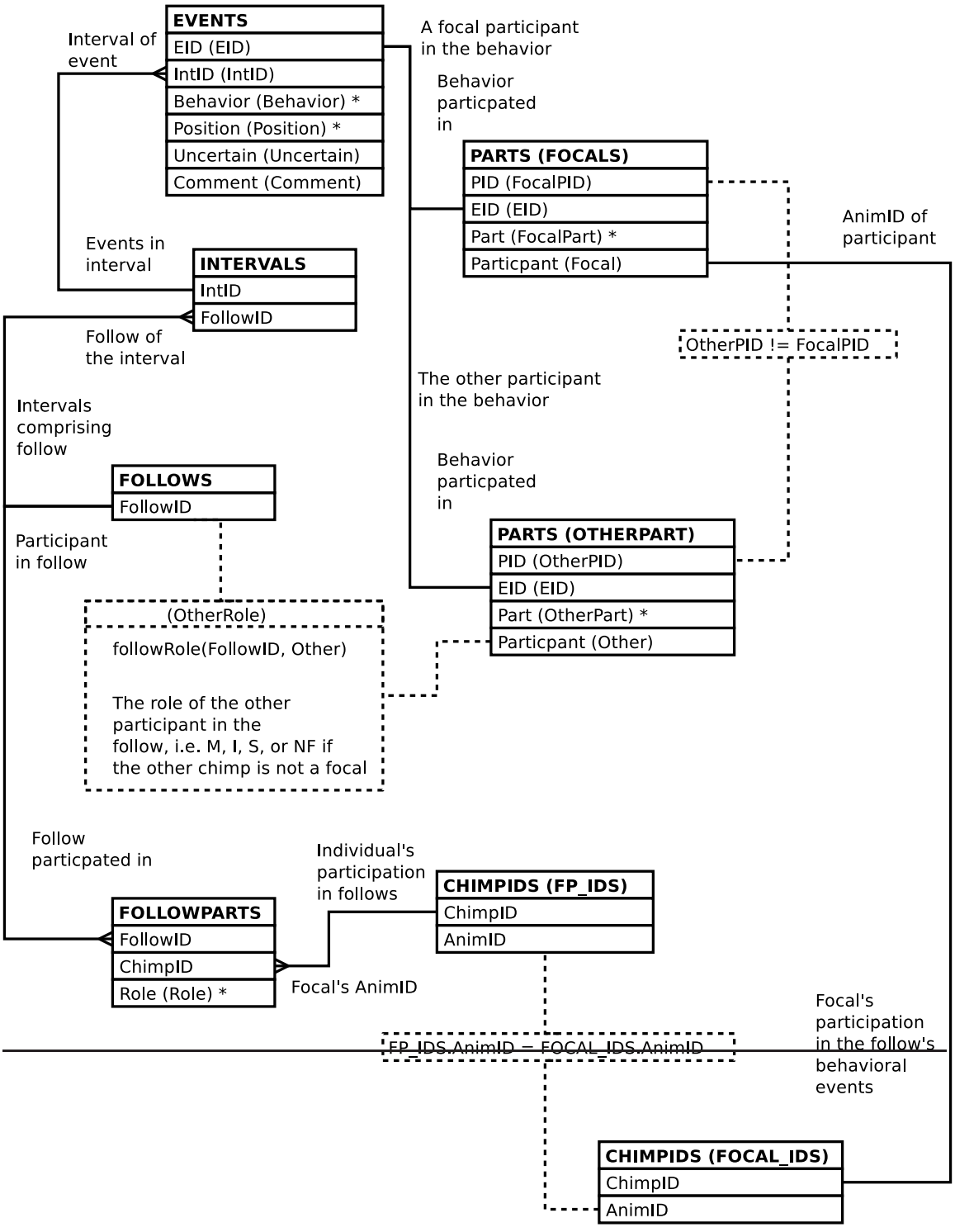


Figure 49: Entity Relationship Diagram of the FOCAL_SOCIAL_EVENTS View

3.3.9 The NOOBS View

```
SELECT followparts.followid AS followid
     , followparts.chimpid AS chimpid
     , followparts.role AS role
     , chimpids.animid AS animid
     , times.time AS time
  FROM followparts
         JOIN chimpids ON (chimpids.chimpid = followparts.chimpid)
         CROSS JOIN
           (SELECT GENERATE_SERIES('2000-01-01 gmi_day_start':: ↩
              TIMESTAMP
                                 , '2000-01-01 gmi_day_end':: ↩
                                    TIMESTAMP
                                 , '1 minute')::TIME
                    AS time
           ) AS times
  WHERE NOT EXISTS
          (SELECT 1
             FROM intervals
                  JOIN events ON (events.intid = intervals.intid)
                  JOIN parts ON (parts.eid = events.eid)
                  JOIN chimpids AS part_cids
                       ON (part_cids.chimpid = parts.participant)
            WHERE intervals.followid = followparts.followid
                  AND intervals.time = times.time
                  AND part_cids.animid = chimpids.animid
            LIMIT 1);
```

Figure 50: Query Defining the NOOBS View

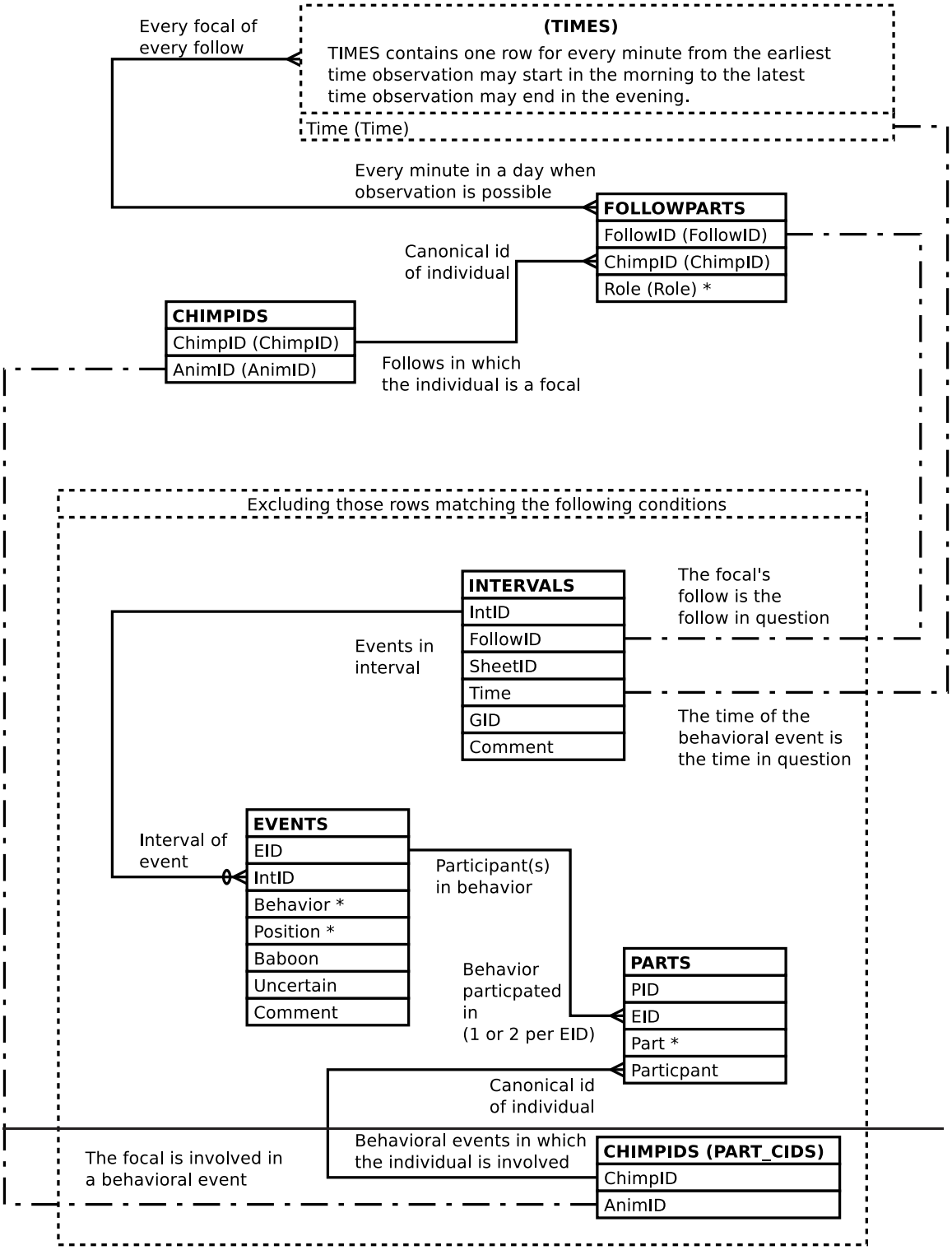


Figure 51: Entity Relationship Diagram of the NOOBS View

3.3.10 The SOCIAL_BOUTS View

```
-- Note that we don't worry about leap seconds given
-- the time values in the db.

WITH detail AS
     -- Compute mins_in and mins_out
     (SELECT bouts.followid AS followid
           , bouts.intid AS intid
           , bouts.function AS function
           , bouts.animid AS animid
           , bouts.role AS role
           , bouts.chimpid AS chimpid
           , bouts.other_animid AS other_animid
           , bouts.bout AS bout
           , bouts.behavior AS behavior
           , bouts.uncertain AS uncertain
           , bouts.time AS time
           , (EXTRACT(EPOCH FROM bouts.time)
                - EXTRACT(EPOCH FROM FIRST_VALUE(bouts.time) OVER  ↩
                   bout_w)
             ) / 60
             + 1
             AS mins_in
           , (EXTRACT(EPOCH FROM LAST_VALUE(bouts.time)
                                 OVER (bout_w RANGE BETWEEN  ↩
                                    CURRENT ROW
                                              AND UNBOUNDED  ↩
                                                 FOLLOWING))
                - EXTRACT(EPOCH FROM bouts.time)
             ) / 60
             AS mins_out
        FROM
          -- Compute bout number by summing transition ticks
          (SELECT ticks.followid AS followid
                , ticks.intid AS intid
                , ticks.function AS function
                , ticks.animid AS animid
                , ticks.role AS role
                , ticks.chimpid AS chimpid
                , ticks.other_animid AS other_animid
                , ticks.behavior AS behavior
                , ticks.uncertain AS uncertain
                , ticks.time AS time
                , SUM(ticks.bouttick)
                    OVER (PARTITION BY ticks.followid
                                     , ticks.behavior
                                     , ticks.animid
                                     , ticks.other_animid
                           ORDER BY ticks.time
                                  , ticks.bouttick DESC)
                  AS bout
```

```
FROM
  -- Distinguish one bout from another
  (SELECT intervals.followid AS followid
        , intervals.intid AS intid
        , followroles.function AS function
        , part_cids.animid AS animid
        , followparts.role AS role
        , followparts.chimpid AS chimpid
        , other_cids.animid AS other_animid
        , events.behavior AS behavior
        , BOOL_AND(events.uncertain) AS uncertain
        , intervals.time AS time
        , CASE
            WHEN LAG(MIN(intervals.time)) OVER tick_w
                   = MIN(intervals.time) - '1 minute':: ↩
                       INTERVAL
              THEN 0
            ELSE
              1
          END
          AS bouttick
```

Figure 53: Query Defining the SOCIAL_BOUTS View: Part II

```
                FROM intervals
                       JOIN events
                            ON (events.intid = intervals.intid)
                       JOIN parts
                            ON (parts.eid = events.eid)
                       JOIN chimpids AS part_cids
                            ON (part_cids.chimpid = parts. ↩
                               participant)
                       JOIN chimpids AS fp_cids
                            ON (fp_cids.animid = part_cids.animid)
                       JOIN followparts
                            ON (followparts.followid = intervals. ↩
                               followid
                                AND followparts.chimpid = fp_cids. ↩
                                   chimpid)
                       JOIN followroles
                            ON (followroles.role = followparts. ↩
                               role)
                       JOIN parts AS other_parts
                            ON (other_parts.eid = events.eid
                                AND other_parts.pid <> parts.pid)
                       JOIN chimpids AS other_cids
                            ON (other_cids.chimpid = other_parts. ↩
                               participant)
                WHERE NOT(events.baboon)
                GROUP BY intervals.followid
                   , intervals.intid
                   , part_cids.animid
                   , other_cids.animid
                   , events.behavior
                   -- The values that are unique per above anyway
                   , followroles.function
                   , followparts.role
                   , followparts.chimpid
                   , intervals.time
                WINDOW tick_w AS (PARTITION BY intervals.followid
                                            , events.behavior
                                            , part_cids.animid
                                            , other_cids.animid
                                  ORDER BY MIN(intervals.time))
            ) AS ticks
        ) AS bouts
    WINDOW bout_w AS (PARTITION BY bouts.followid
                                 , bouts.behavior
                                 , bouts.animid
                                 , bouts.other_animid
                                 , bouts.bout
                      ORDER BY bouts.time)
)
```

Figure 54: Query Defining the SOCIAL_BOUTS View: Part III

```
-- Distribute aggregated bout info over the bout's minutes
SELECT detail.followid AS followid
     , detail.intid AS intid
     , detail.function AS function
     , detail.animid AS animid
     , detail.role AS role
     , detail.chimpid AS chimpid
     , summary.other_function AS other_function
     , detail.other_animid AS other_animid
     , summary.other_role AS other_role
     , summary.other_chimpid AS other_chimpid
     , detail.bout AS bout
     , detail.behavior AS behavior
     , detail.uncertain AS uncertain
     , detail.time AS time
     , detail.mins_in AS mins_in
     , detail.mins_out AS mins_out
     , summary.prior_gap AS prior_gap
     , summary.gap_noobs AS gap_noobs
     , summary.left_censored AS left_censored
     , summary.right_censored AS right_censored
     , summary.day_start AS day_start
     , summary.day_end AS day_end
  FROM
    detail
      JOIN
      (-- Determine the role the focal's social partner has
       -- in the follow family group, if any.
       SELECT bouts.followid AS followid
            , bouts.animid AS animid
            , bouts.other_animid AS other_animid
            , other_followroles.function AS other_function
            , other_followparts.role AS other_role
            , other_followparts.chimpid AS other_chimpid
            , bouts.behavior AS behavior
            , bouts.bout AS bout
            , bouts.prior_gap AS prior_gap
            , bouts.gap_noobs AS gap_noobs
            , bouts.left_censored AS left_censored
            , bouts.right_censored AS right_censored
            , bouts.day_start AS day_start
            , bouts.day_end AS day_end
```

Figure 55: Query Defining the SOCIAL_BOUTS View: Part IV

```
FROM
  (-- Aggregate per-bout information
   SELECT bout_ends.followid AS followid
        , bout_ends.animid AS animid
        , bout_ends.other_animid AS other_animid
        , bout_ends.behavior AS behavior
        , bout_ends.bout AS bout
        , CASE
            WHEN bout_ends.bout = 1 THEN
              NULL
            ELSE
              SUM(bout_ends.prior_gap)
          END
          AS prior_gap
        , CASE
            WHEN bout_ends.bout = 1 THEN
              NULL
            ELSE
              SUM(bout_ends.prior_gap)
                -- Subtract number of minutes the focal  ↩
                   had
                -- any behaviors during the gap.
                - (SELECT COUNT(DISTINCT intervals.intid)
                     FROM intervals
                          JOIN events
                               ON (events.intid =  ↩
                                  intervals.intid)
                          JOIN parts
                               ON (parts.eid = events.eid)
                          JOIN chimpids
                               ON (chimpids.chimpid
                                   = parts.participant)
                    WHERE intervals.followid = bout_ends. ↩
                       followid
                          AND MIN(bout_ends.time)
                                > intervals.time
                          AND intervals.time
                                >= MIN(bout_ends.time)
                                   - (SUM(bout_ends. ↩
                                      prior_gap)
                                      || ' minutes':: ↩
                                         INTERVAL
                          AND chimpids.animid = bout_ends ↩
                              .animid)
          END
          AS gap_noobs
        , BOOL_OR(bout_ends.left_censored) AS  ↩
          left_censored
        , BOOL_OR(bout_ends.right_censored) AS  ↩
          right_censored
```

Figure 56: Query Defining the SOCIAL_BOUTS View: Part V

```
, CASE
    WHEN bout_ends.bout = 1 THEN
      NOT EXISTS
            -- Behavioral data on the focal before ↩
                the bout
            (SELECT 1
               FROM
                 intervals
                   JOIN events
                        ON (events.intid =  ↩
                            intervals.intid)
                   JOIN parts
                        ON (parts.eid = events.eid ↩
                            )
                   JOIN chimpids
                        ON (chimpids.chimpid
                             = parts.participant)
               WHERE intervals.followid =  ↩
                  bout_ends.followid
                    AND intervals.time
                          < MIN(bout_ends.time)
                    AND chimpids.animid =  ↩
                        bout_ends.animid
               LIMIT 1)
    ELSE
      FALSE
  END
  AS day_start
, CASE
    WHEN bout_ends.bout = LAST_VALUE(bout_ends. ↩
       bout)
                              OVER (PARTITION BY
                                      bout_ends. ↩
                                         followid
                                    , bout_ends. ↩
                                       animid
                                    , bout_ends. ↩
                                       behavior
                                    ORDER BY  ↩
                                       bout_ends. ↩
                                       bout
                                    RANGE
                                      BETWEEN  ↩
                                         CURRENT ROW
                                      AND UNBOUNDED  ↩
                                         FOLLOWING)
      THEN
      NOT EXISTS
        -- Behavioral data on the focal after the  ↩
           bout
        (SELECT 1
           FROM intervals
                  JOIN events
                       ON (events.intid =  ↩
                           intervals.intid)
```

```
FROM
  -- Examine the bout's endpoints to compute  ↩
     prior_gap, and
  -- left and right censoring.
  (SELECT detail.followid
        , detail.animid
        , detail.other_animid
        , detail.behavior
        , detail.bout
        , detail.mins_in AS mins_in
        , detail.mins_out AS mins_out
        , CASE
            WHEN detail.mins_in = 1 THEN
              (EXTRACT(EPOCH FROM detail.time)
                 - EXTRACT(EPOCH FROM
                           LAG(detail.time)
                             OVER (PARTITION BY
                                     detail.followid
                                   , detail.animid
                                   , detail. ↩
                                      other_animid
                                   , detail.behavior
                                   ORDER BY detail. ↩
                                      time))
              ) / 60 - 1
            ELSE
              0
          END
          AS prior_gap
        , CASE
            WHEN detail.mins_in = 1 THEN
              NOT EXISTS
                (SELECT 1
                   FROM
                     intervals AS i
                       JOIN events AS e
                            ON (e.intid = i.intid)
                       JOIN parts AS p
                            ON (p.eid = e.eid)
                       JOIN chimpids AS p_cids
                            ON (p_cids.chimpid = p. ↩
                               participant)
                  WHERE i.followid = detail.followid
                        AND i.time = detail.time
                                     - '1 minute':: ↩
                                        INTERVAL
                        AND p_cids.animid = detail. ↩
                           animid
                        LIMIT 1)
            ELSE
              FALSE
            END
          AS left_censored
```

```
                , CASE
                    WHEN detail.mins_out = 0 THEN
                      NOT EXISTS
                        (SELECT 1
                           FROM
                             intervals AS i
                               JOIN events AS e
                                    ON (e.intid = i.intid)
                               JOIN parts AS p
                                    ON (p.eid = e.eid)
                               JOIN chimpids AS p_cids
                                    ON (p_cids.chimpid = p. ↩
                                       participant)
                           WHERE i.followid = detail.followid
                                 AND i.time = detail.time
                                              + '1 minute':: ↩
                                                 INTERVAL
                               AND p_cids.animid = detail. ↩
                                   animid
                               LIMIT 1)
                    ELSE
                      FALSE
                    END
                  AS right_censored
                , detail.time AS time
            FROM detail
            WHERE detail.mins_in = 1
                  OR detail.mins_out = 0
        ) AS bout_ends
       GROUP BY bout_ends.followid
              , bout_ends.animid
              , bout_ends.other_animid
              , bout_ends.behavior
              , bout_ends.bout
    ) AS bouts
    JOIN chimpids AS other_chimpids
         ON (other_chimpids.animid = bouts.other_animid)
    LEFT OUTER JOIN followparts AS other_followparts
                    ON (other_followparts.followid = bouts. ↩
                       followid
                        AND other_followparts.chimpid
                            = other_chimpids.chimpid)
    LEFT OUTER JOIN followroles AS other_followroles
                    ON (other_followroles.role
                        = other_followparts.role)
) AS summary
  ON (summary.followid = detail.followid
      AND summary.animid = detail.animid
      AND summary.other_animid = detail.other_animid
      AND summary.behavior = detail.behavior
      AND summary.bout = detail.bout);
```

Figure 59: Query Defining the SOCIAL_BOUTS View: Part VIII

ER diagram intentionally omitted.

Figure 60: Entity Relationship Diagram of the SOCIAL_BOUTS View

3.3.11 The SOCIAL_BOUT_AGGS View

```
SELECT social_bouts.followid AS followid
     , social_bouts.function AS function
     , social_bouts.animid AS animid
     , social_bouts.role AS role
     , social_bouts.chimpid AS chimpid
     , social_bouts.other_function AS other_function
     , social_bouts.other_animid AS other_animid
     , social_bouts.other_role AS other_role
     , social_bouts.other_chimpid AS other_chimpid
     , social_bouts.bout AS bout
     , social_bouts.behavior AS behavior
     , BOOL_OR(social_bouts.uncertain) AS uncertain
     , MIN(social_bouts.time) AS start_time
     , MAX(social_bouts.time) AS stop_time
     , social_bouts.prior_gap AS prior_gap
     , social_bouts.gap_noobs AS gap_noobs
     , social_bouts.left_censored AS left_censored
     , social_bouts.right_censored AS right_censored
     , social_bouts.day_start AS day_start
     , social_bouts.day_end AS day_end
  FROM social_bouts
  GROUP BY social_bouts.followid
         , social_bouts.animid
         , social_bouts.other_animid
         , social_bouts.bout
         , social_bouts.behavior
         -- The values that are unique per above anyway
         , social_bouts.function
         , social_bouts.role
         , social_bouts.chimpid
         , social_bouts.other_function
         , social_bouts.other_role
         , social_bouts.other_chimpid
         , social_bouts.prior_gap
         , social_bouts.gap_noobs
         , social_bouts.left_censored
         , social_bouts.right_censored
         , social_bouts.day_start
         , social_bouts.day_end;
```

Figure 61: Query Defining the SOCIAL_BOUT_AGGS View

ER diagram intentionally omitted.

Figure 62: Entity Relationship Diagram of the SOCIAL_BOUT_AGGS View

3.3.12 The SOCIAL_EVENTS View

```
SELECT events.eid AS eid
     , events.intid AS intid
     , part1.pid AS part1pid
     , part2.pid AS part2pid
     , events.behavior AS behavior
     , events.position AS position
     , events.uncertain AS uncertain
     , followrole(intervals.followid, part1.participant) AS  ↩
        part1role
     , part1.participant AS participant1
     , part1.part AS part1
     , followrole(intervals.followid, part2.participant) AS  ↩
        part2role
     , part2.participant AS participant2
     , part2.part AS part2
     , events.comment AS comment
  FROM events
         JOIN intervals
           ON (intervals.intid = events.intid)
         JOIN parts AS part1
           ON (part1.eid = events.eid)
         JOIN parts AS part2
           ON (part2.eid = events.eid)
  WHERE part1.pid <> part2.pid;
```

Figure 63: Query Defining the SOCIAL_EVENTS View

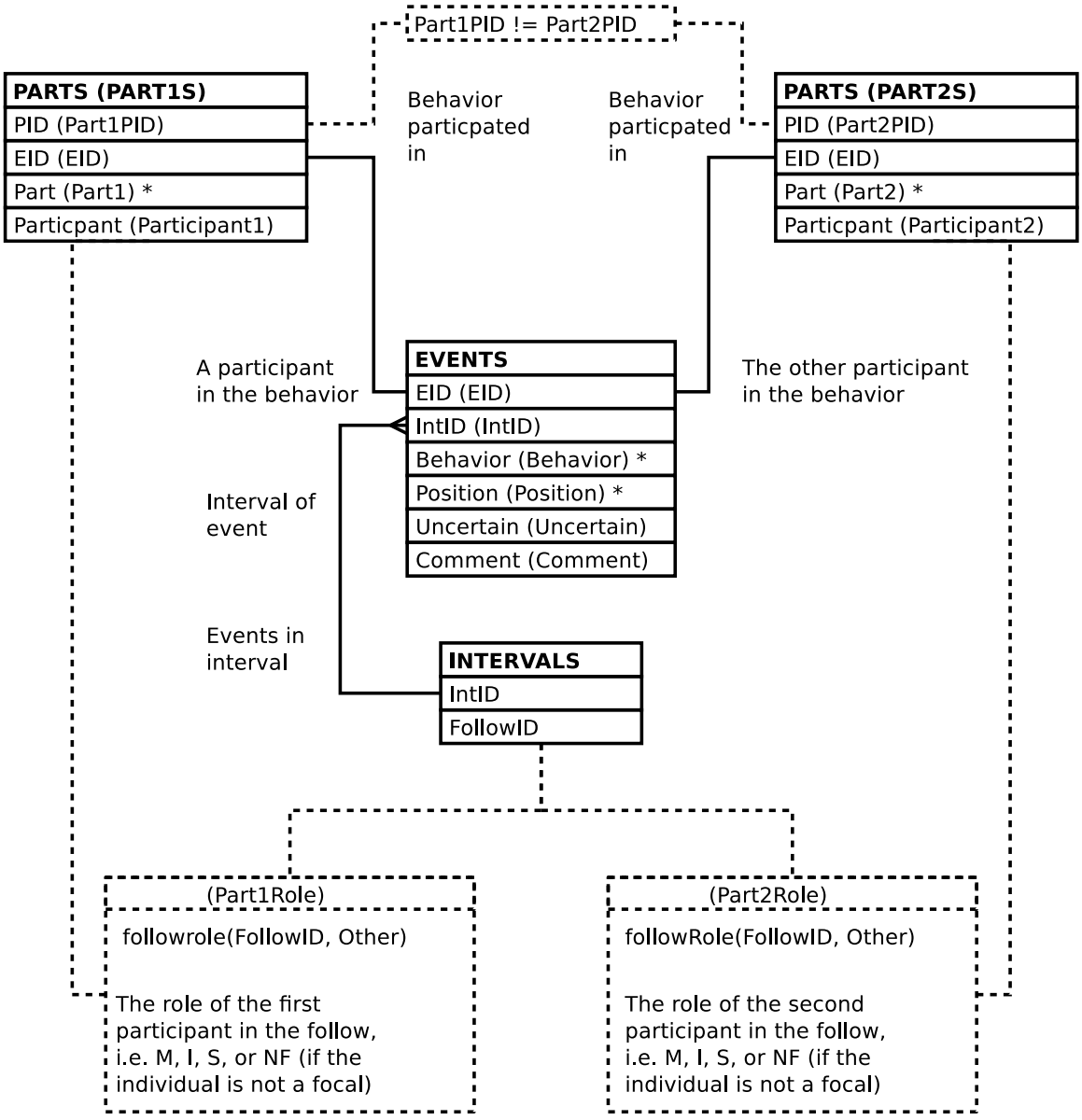


Figure 64: Entity Relationship Diagram of the SOCIAL_EVENTS View

3.3.13 The SOCIAL_EVENTS_W_BABOONS View

```
SELECT events.eid AS eid
     , events.intid AS intid
     , part1.pid AS part1pid
     , part2.pid AS part2pid
     , events.behavior AS behavior
     , events.position AS position
     , events.uncertain AS uncertain
     , followrole(intervals.followid, part1.participant) AS  ↩
        part1role
     , FALSE AS part1baboon
     , part1.participant AS participant1
     , part1.part AS part1
     , followrole(intervals.followid, part2.participant) AS  ↩
        part2role
     , FALSE AS part2baboon
     , part2.participant AS participant2
     , part2.part AS part2
     , events.comment AS comment
  FROM events
         JOIN intervals
           ON (intervals.intid = events.intid)
         JOIN parts AS part1
           ON (part1.eid = events.eid)
         JOIN parts AS part2
           ON (part2.eid = events.eid)
  WHERE part1.pid <> part2.pid
```

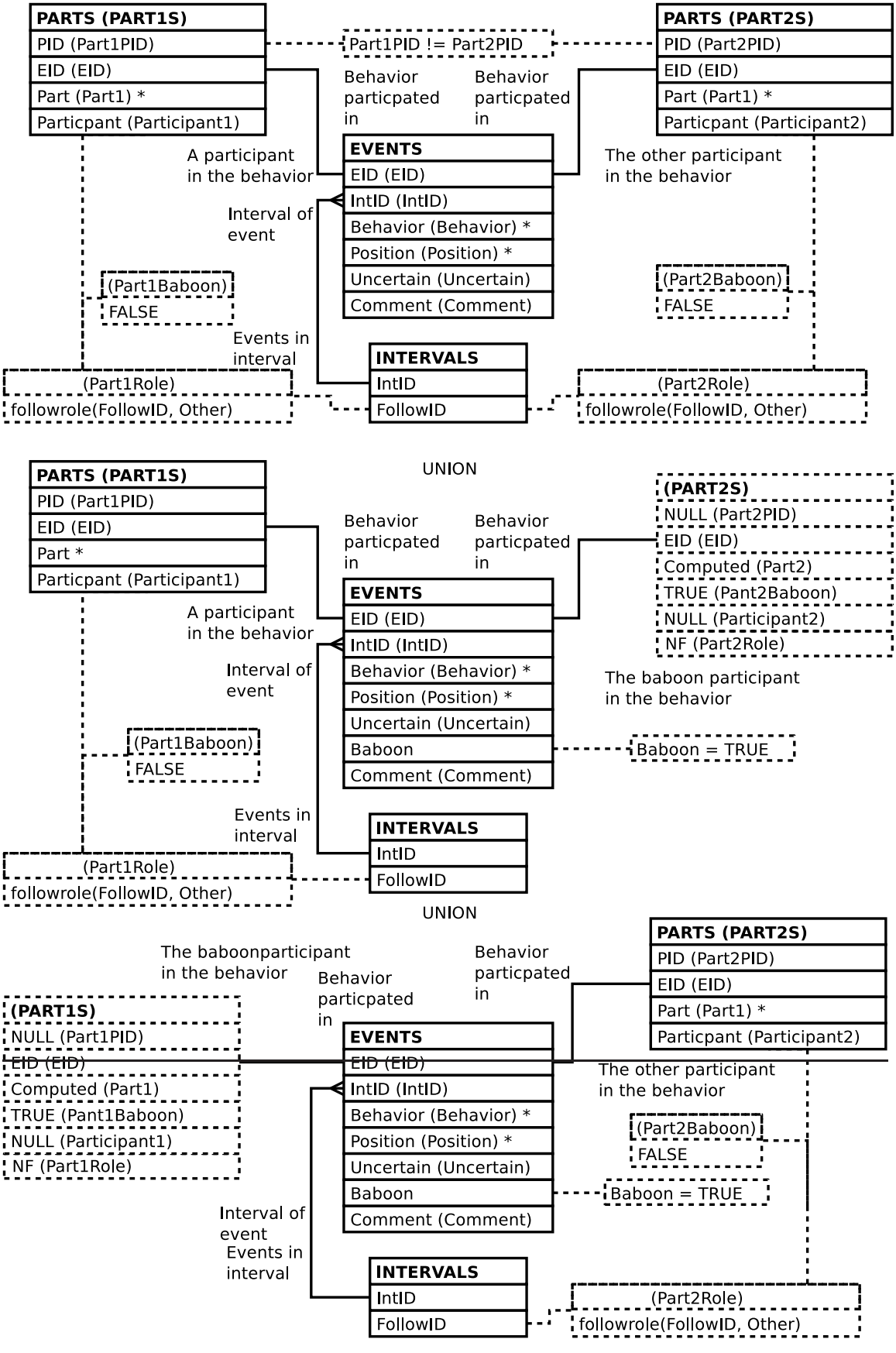
Figure 65: Query Defining the SOCIAL_EVENTS_W_BABOONS View: Part I

```
UNION
  SELECT events.eid AS eid
       , events.intid AS intid
       , part1.pid AS part1pid
       , NULL AS part2pid
       , events.behavior AS behavior
       , events.position AS position
       , events.uncertain AS uncertain
       , followrole(intervals.followid, part1.participant) AS  ↩
          part1role
       , FALSE AS part1baboon
       , part1.participant AS participant1
       , part1.part AS part1
       , 'NF' AS part2role
       , TRUE AS part2baboon
       , NULL::VARCHAR(gmi_animid_len) AS participant2
       , CASE
           WHEN part1.part = 'gmi_actor' THEN
             'gmi_actee'
           WHEN part1.part = 'gmi_actee' THEN
             'gmi_actor'
           ELSE
             'gmi_participant'
         END::parts_part
         AS part2
       , events.comment AS comment
  FROM events
         JOIN intervals
           ON (intervals.intid = events.intid)
         JOIN parts AS part1
           ON (part1.eid = events.eid)
  WHERE events.baboon
```

Figure 66: Query Defining the SOCIAL_EVENTS_W_BABOONS View: Part II

```
UNION
  SELECT events.eid AS eid
       , events.intid AS intid
       , NULL AS part1pid
       , part2.pid AS part2pid
       , events.behavior AS behavior
       , events.position AS position
       , events.uncertain AS uncertain
       , 'NF' AS part1role
       , TRUE AS part1baboon
       , NULL::VARCHAR(gmi_animid_len) AS participant1
       , CASE
           WHEN part2.part = 'gmi_actor' THEN
             'gmi_actee'
           WHEN part2.part = 'gmi_actee' THEN
             'gmi_actor'
           ELSE
             'gmi_participant'
         END::parts_part
         AS part1
       , followrole(intervals.followid, part2.participant) AS  ↩
          part2role
       , FALSE AS part2baboon
       , part2.participant AS participant2
       , part2.part AS part2
       , events.comment AS comment
  FROM events
         JOIN intervals
           ON (intervals.intid = events.intid)
         JOIN parts AS part2
           ON (part2.eid = events.eid)
  WHERE events.baboon;
```

Figure 67: Query Defining the SOCIAL_EVENTS_W_BABOONS View: Part III


PARTS (PART1S)
PID (Part1PID)
EID (EID)
Part (Part1) *
Particpant (Participant1)
Part1PID != Part2PID
PARTS (PART2S)
PID (Part2PID)
EID (EID)
Part (Part1) *
Particpant (Participant2)
Behavior particpated in
Behavior particpated in
A participant in the behavior
EVENTS
EID (EID)
IntID (IntID)
Behavior (Behavior) *
Position (Position) *
Uncertain (Uncertain)
Comment (Comment)
The other participant in the behavior
Interval of event
(Part1Baboon)
FALSE
(Part2Baboon)
FALSE
Events in interval
INTERVALS
IntID
FollowID
(Part1Role)
followrole(FollowID, Other)
(Part2Role)
followrole(FollowID, Other)
UNION
PARTS (PART1S)
PID (Part1PID)
EID (EID)
Part *
Particpant (Participant1)
(PART2S)
NULL (Part2PID)
EID (EID)
Computed (Part2)
TRUE (Pant2Baboon)
NULL (Participant2)
NF (Part2Role)
Behavior particpated in
Behavior particpated in
A participant in the behavior
EVENTS
EID (EID)
IntID (IntID)
Behavior (Behavior) *
Position (Position) *
Uncertain (Uncertain)
Baboon
Comment (Comment)
Interval of event
The baboon participant in the behavior
(Part1Baboon)
FALSE
Baboon = TRUE
Events in interval
INTERVALS
IntID
FollowID
(Part1Role)
followrole(FollowID, Other)
UNION
The baboonparticipant in the behavior
Behavior particpated in
Behavior particpated in
PARTS (PART2S)
PID (Part2PID)
EID (EID)
Part (Part1) *
Particpant (Participant2)
(PART1S)
NULL (Part1PID)
EID (EID)
Computed (Part1)
TRUE (Pant1Baboon)
NULL (Participant1)
NF (Part1Role)
EVENTS
EID (EID)
IntID (IntID)
Behavior (Behavior) *
Position (Position) *
Uncertain (Uncertain)
Baboon
Comment (Comment)
The other participant in the behavior
(Part2Baboon)
FALSE
Baboon = TRUE
Interval of event
Events in interval
INTERVALS
IntID
FollowID
(Part2Role)
followrole(FollowID, Other)

3.4 Follow Distance Related Views

3.4.1 The DISTS View

```
SELECT distances.distid AS distid
     , distances.intid AS intid
     , dpart1s.dpid AS dpart1pid
     , dpart2s.dpid AS dpart2pid
     , distances.uncertain AS uncertain
     , dpart1s.role AS dpart1role
     , dpart1s.participant AS participant1
     , dpart2s.role AS dpart2role
     , dpart2s.participant AS participant2
     , distances.distance AS distance
  FROM distances
         JOIN distanceparts AS dpart1s
           ON (dpart1s.distid = distances.distid)
         JOIN distanceparts AS dpart2s
           ON (dpart2s.distid = distances.distid)
  WHERE dpart1s.dpid <> dpart2s.dpid;
```

Figure 69: Query Defining the DISTS View

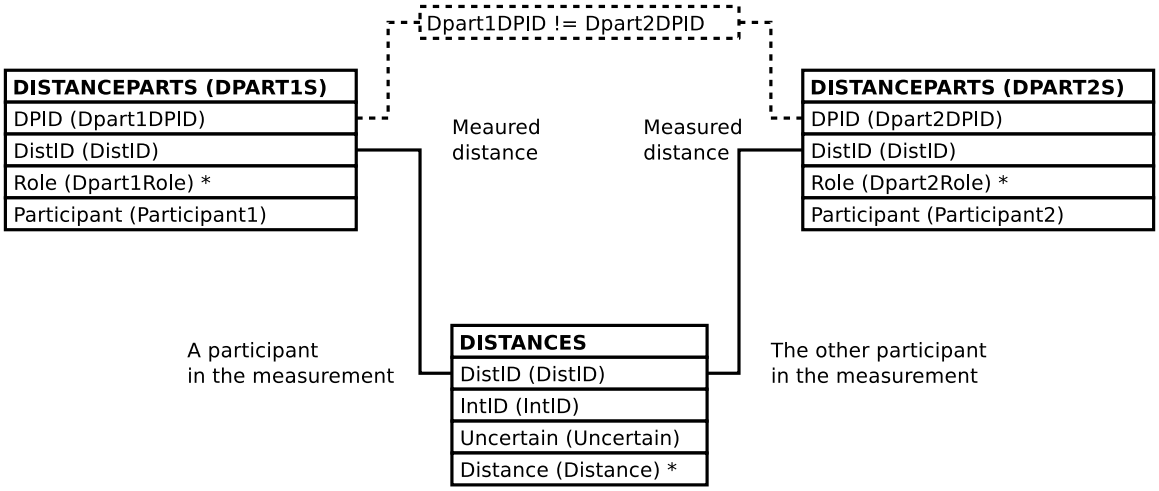


Figure 70: Entity Relationship Diagram of the DISTS View

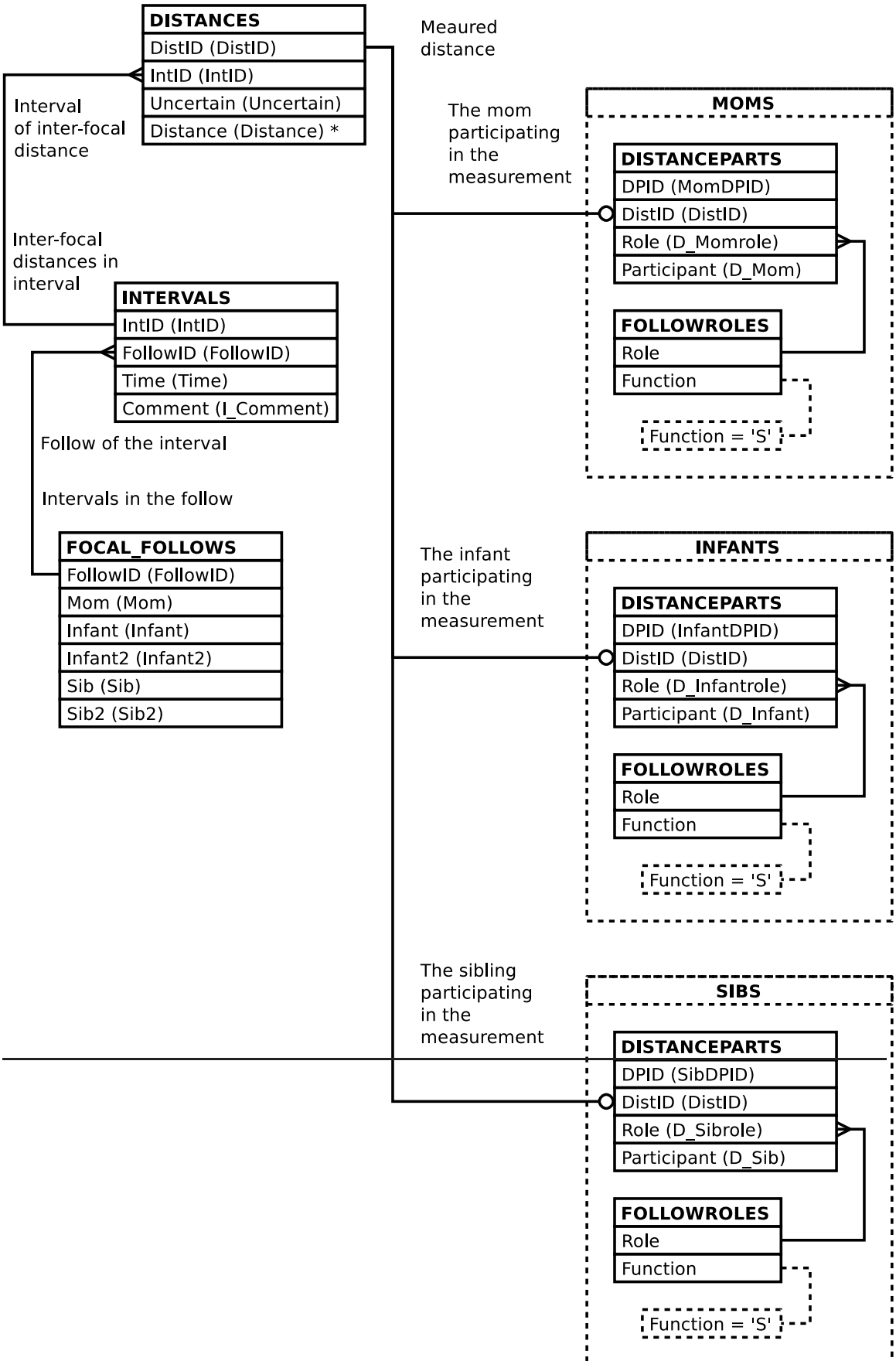
3.4.2 The FAMILY_DISTS View

```
SELECT focal_follows.followid AS followid
     , intervals.intid AS intid
     , distances.distid AS distid
     , moms.dpid AS d_momdpid
     , infants.dpid AS d_infantdpid
     , sibs.dpid AS d_sibdpid
     , focal_follows.date AS date
     , intervals.time AS time
     , focal_follows.mom AS mom
     , focal_follows.infant AS infant
     , focal_follows.infant2 AS infant2
     , focal_follows.sib AS sib
     , focal_follows.sib2 AS sib2
     , distances.uncertain AS uncertain
     , moms.participant AS d_mom
     , moms.role as d_momrole
     , infants.participant AS d_infant
     , infants.role AS d_infantrole
     , sibs.participant AS d_sib
     , sibs.role AS d_sibrole
     , distances.distance AS distance
     , intervals.comment AS i_comment
```

Figure 71: Query Defining the FAMILY_DISTS View: Part I

```
FROM focal_follows
       JOIN intervals
         ON (intervals.followid = focal_follows.followid)
       JOIN distances
         ON (distances.intid = intervals.intid)
       LEFT OUTER JOIN
         (SELECT distanceparts.dpid AS dpid
               , distanceparts.distid AS distid
               , distanceparts.participant AS participant
               , distanceparts.role AS role
               , followroles.function AS function
            FROM distanceparts
                   JOIN followroles
                     ON (followroles.role = distanceparts.role)
         ) AS moms
             ON (moms.distid = distances.distid
                 AND moms.function = 'gmi_mom')
       LEFT OUTER JOIN
         (SELECT distanceparts.dpid AS dpid
               , distanceparts.distid AS distid
               , distanceparts.participant AS participant
               , distanceparts.role AS role
               , followroles.function AS function
            FROM distanceparts
                   JOIN followroles
                     ON (followroles.role = distanceparts.role)
         ) AS infants
             ON (infants.distid = distances.distid
                 AND infants.function = 'gmi_infant')
       LEFT OUTER JOIN
         (SELECT distanceparts.dpid AS dpid
               , distanceparts.distid AS distid
               , distanceparts.participant AS participant
               , distanceparts.role AS role
               , followroles.function AS function
            FROM distanceparts
                   JOIN followroles
                     ON (followroles.role = distanceparts.role)
         ) AS sibs
             ON (sibs.distid = distances.distid
                 AND sibs.function = 'gmi_sib');
```

Figure 72: Query Defining the FAMILY_DISTS View: Part II


DISTANCES
DistID (DistID)
IntID (IntID)
Uncertain (Uncertain)
Distance (Distance) *
Meaured distance
Interval of inter-focal distance
Inter-focal distances in interval
INTERVALS
IntID (IntID)
FollowID (FollowID)
Time (Time)
Comment (I_Comment)
Follow of the interval
Intervals in the follow
FOCAL_FOLLOWS
FollowID (FollowID)
Mom (Mom)
Infant (Infant)
Infant2 (Infant2)
Sib (Sib)
Sib2 (Sib2)
The mom participating in the measurement
MOMS
DISTANCEPARTS
DPID (MomDPID)
DistID (DistID)
Role (D_Momrole)
Participant (D_Mom)
FOLLOWROLES
Role
Function
Function = 'S'
The infant participating in the measurement
INFANTS
DISTANCEPARTS
DPID (InfantDPID)
DistID (DistID)
Role (D_Infantrole)
Participant (D_Infant)
FOLLOWROLES
Role
Function
Function = 'S'
The sibling participating in the measurement
SIBS
DISTANCEPARTS
DPID (SibDPID)
DistID (DistID)
Role (D_Sibrole)
Participant (D_Sib)
FOLLOWROLES
Role
Function
Function = 'S'

3.5 Group Related Views