Associazione Erpetologica Allevatori

6 GIUGNO

Progetto di Basi di Dati

Autore: Viol Alessandro IN0500349

Sommario

Descrizione del problema

Si vuole realizzare una base di dati per un'ipotetica Associazione Erpetologica[1] Allevatori che si occupa di allevare e vendere serpenti, oltre che di accogliere serpenti feriti o non autoctoni. Vogliamo quindi rappresentare le informazioni dei vari allevamenti dell'Associazione, dei relativi serpenti e dei nostri clienti. Per aiutare l'acquisto consapevole da parte dei potenziali clienti, si vogliono anche rappresentare le informazioni relative alla specie dei serpenti in vendita e a come prendersene cura. Inoltre, anche per supportare gli allevatori, si rappresentano invece le informazioni relative alle visite mediche, all'alimentazione e alla brumazione[2] dei serpenti.

Descrizione dei requisiti

Per gli allevamenti, ci interessa permanere le informazioni relative al loro nome, partita IVA, e-mail, indirizzo, numeri di telefono e, se presente, l'indirizzo del loro sito.

Per quanto riguarda i clienti, vogliamo rappresentare codice fiscale, nome, cognome, e-mail e numero di telefono. Per ogni serpente acquistato dal cliente viene memorizzata anche la data di acquisto.

Dei serpenti ci interessa conoscere il nome, il sesso, la data di nascita, e, se il serpente è in vendita o in adozione, un prezzo, un URL che rimandi al suo certificato Cites[3] e un URL che rimandi a una sua foto. Si vuole anche rappresentare lo stato del serpente, ossia se questo è in vendita o in adozione, se è già stato venduto o adottato, se è morto[4] o se semplicemente non è in vendita o viene usato come esemplare riproduttore.

Se tutti i serpenti acquistati e adottati da un cliente sono morti, si cancellano le informazioni relative al cliente.

Due serpenti dello stesso allevamento non possono avere lo stesso nome.

Poiché un serpente viene accoppiato con più serpenti ogni anno e non è facile capire il padre di una covata, oltre a rappresentare i legami di parentela tra serpenti vogliamo tenere traccia anche dei tentativi di accoppiamento con relativa data.

Se a un serpente viene imposto uno stato di brumazione, questo deve essere registrato, rappresentando la data di inizio della pre-brumazione, la data di inizio e di fine della brumazione e la data di fine post-brumazione.

Deve essere possibile gestire il caso in cui un serpente venga rimosso dal periodo di brumazione a causa di problemi di salute.

Vengono mantenuti i dati delle brumazioni dei soli serpenti vivi e di proprietà dell'allevamento.

---

[1] L'erpetologia è la branca della zoologia che si occupa dei rettili.
[2] Processo simile al letargo che interessa i rettili. In cattività i serpenti non vanno autonomamente in brumazione, né ne hanno la necessità a meno di prepararsi alla stagione degli accoppiamenti. Tuttavia, un esemplare abituato alla brumazione subisce un forte stress nel caso non venga più brumato e può anche decidere di rifiutare il cibo portandolo eventualmente alla morte.
[3] Il certificato Cites è una certificazione simile al Pedigree di cani e gatti obbligatoria per la vendita di rettili ma non per la loro adozione.
[4] Nell'allevamento di rettili è fondamentale tenere traccia della genetica dei propri animali. Perciò, anche se morti, è di interesse permanere i loro dati. Analogamente, è di interesse mantenere i dati anche nel caso di vendita o adozione, poiché non è da escludere che un allevamento possa chiedere di riacquistare un particolare serpente per rimpiazzare un riproduttore.

Per ogni serpente presente negli allevamenti, si vuole redirigere una lista di visite di controllo, che possono essere eseguite sia dall'allevamento che da una clinica esterna in caso di problemi di salute del serpente.

Delle visite si vogliono rappresentare dimensioni e peso dell'animale, la data, il motivo del controllo, l'eventuale prescrizione di farmaci o terapie e il tempo di somministrazione della stessa.

È richiesto di effettuare almeno due visite all'anno per ogni serpente, oltre che una alla nascita e una alla morte. Al momento della morte di un serpente in allevamento, è previsto che un operatore cancelli tutte le visite salvo quelle che si ritiene contengano informazioni su problematiche che potrebbero essere ereditate dalla prole. Se il serpente non ha prole, si può invece procedere con l'eliminazione diretta di tutte le sue visite.

Per le cliniche, si rappresenta il nome, la partita IVA, i numeri di telefono e l'indirizzo.

Per ogni serpente presente negli allevamenti, si tiene un elenco dei pasti[5], che registra la data, il tipo di somministrazione[6], il cibo somministrato e se il serpente ha accettato il pasto.

Nel registro pasti vogliamo mantenere solo l'ultimo anno di pasti e solo dei serpenti vivi e ancora di proprietà degli allevamenti.

Un serpente può essere messo in vendita solo se ha consumato con successo almeno due pasti.

Dei cibi somministrabili ai serpenti si rappresenta il nome, la dimensione e il peso.

Per i serpenti riproduttori femmina degli allevamenti, è di interesse rappresentare i dati dei concepimenti. Per ogni concepimento si rappresenta pertanto la data, il numero totale della progenie e dei morti[7]. Se il serpente è oviparo, si memorizza anche il numero di uova non fecondate (slugs).

Per ogni covata vi è un'incubazione, di cui rappresentiamo la temperatura, l'umidità e le date di inizio e di fine.

Ogni serpente viene classificato in una specie, di cui si rappresentano nome scientifico e comune, l'età massima, l'aspettativa media di vita, l'alimentazione, il periodo del giorno di attività[8], il meccanismo di difesa[9], il temperamento[10], una valutazione sulla difficoltà di accudimento, se è possibile coabitare più esemplari della stessa specie nello stesso terrario, se sono ovovivipari[11], qual è il range di uova o cuccioli che producono e quanto tempo di incubazione necessitano.

Si suppone che, quando un serpente di un cliente raggiunge l'età dell'aspettativa di vita della specie, questo non sia più di interesse per gli allevatori. Viene pertanto indicato come morto.

Ogni specie viene divisa in tre fascie di età[12]. Per ognuna si rappresenta l'intervallo di età e la lunghezza e il peso medi.

---

[5] La principale causa di morte dei serpenti è il rifiuto del cibo a causa dello stress. Per un cliente è fondamentale sapere se il serpente che intende acquistare soffre di disturbi alimentari, in quanto possono essere difficili da correggere per un neofita. Dal punto di vista degli allevatori, tenere un registro alimentare è essenziale, in quanto un serpente che mangia regolarmente consuma un pasto ogni 1,5-3 settimane e può rifiutare il cibo per più di un mese.

[6] Scongelato, pre-ucciso o vivo. La difficoltà nel reperire il cibo vivo e la difficoltà nel far cambiare abitudini alimentari al serpente rendono questa informazione molto importante.

[7] È abbastanza comune per i serpenti non riuscire a superare i primi giorni di vita o non svilupparsi. Stimiamo che la percentuale che non ce la fa sia del 10%.

[8] I serpenti sono attivi in diverse fasi del giorno in base alla loro specie. Si distinguono in notturni, diurni o crepuscolari.

[9] I serpenti di una specie, se intimiditi, tendono a reagire principalmente in uno solo dei seguenti modi: mordendo, mordendo e avvelenando (in Italia i serpenti velenosi commerciabili non hanno un veleno pericoloso per l'uomo, a meno di allergie), effettuando tanatosi, intimidendo, rilasciando forti odori (musk) o avviluppandosi sulla minaccia.

[10] Facilità con cui si lasciano approcciare.

[11] Ci sono serpenti ovovivipari, ossia che non depongono uova, come ad esempio il garter snake.

[12] Baby, sub-adulti e adulti.

In natura, alcune specie di serpente vanno in brumazione, perciò si vogliono rappresentare le informazioni relative al mese di inizio e fine della brumazione, la temperatura di brumazione e quante settimane di acclimatamento sono necessarie[13].

Vi sono vari substrati[14] che possono essere adatti alla brumazione di un serpente, ognuno con delle sue proprietà.

Per ogni specie viene utilizzato un diverso tipo di terrario.

Dei terrari, rappresentiamo le informazioni sul metodo di riscaldamento[15], la temperatura della zona calda e della zona fredda, l'umidità, la tipologia di contenitore[16], se necessitano di piccole piscine, appigli da scalare o zone in cui scavare.

La dimensione minima e consigliata dei terrari e delle tane ivi inserite dipende dalla fascia di età della specie che viene ospitata.

Come per la brumazione, ogni terrario utilizza uno o più substrati.

Ogni specie possiede uno o più morphs[17] e può avere delle patologie associate ad esso[18]. L'aspetto di un serpente può essere dettato da uno o più morph. Un serpente può avere una certa percentuale di probabilità di essere "het" per un particolare morph[19].

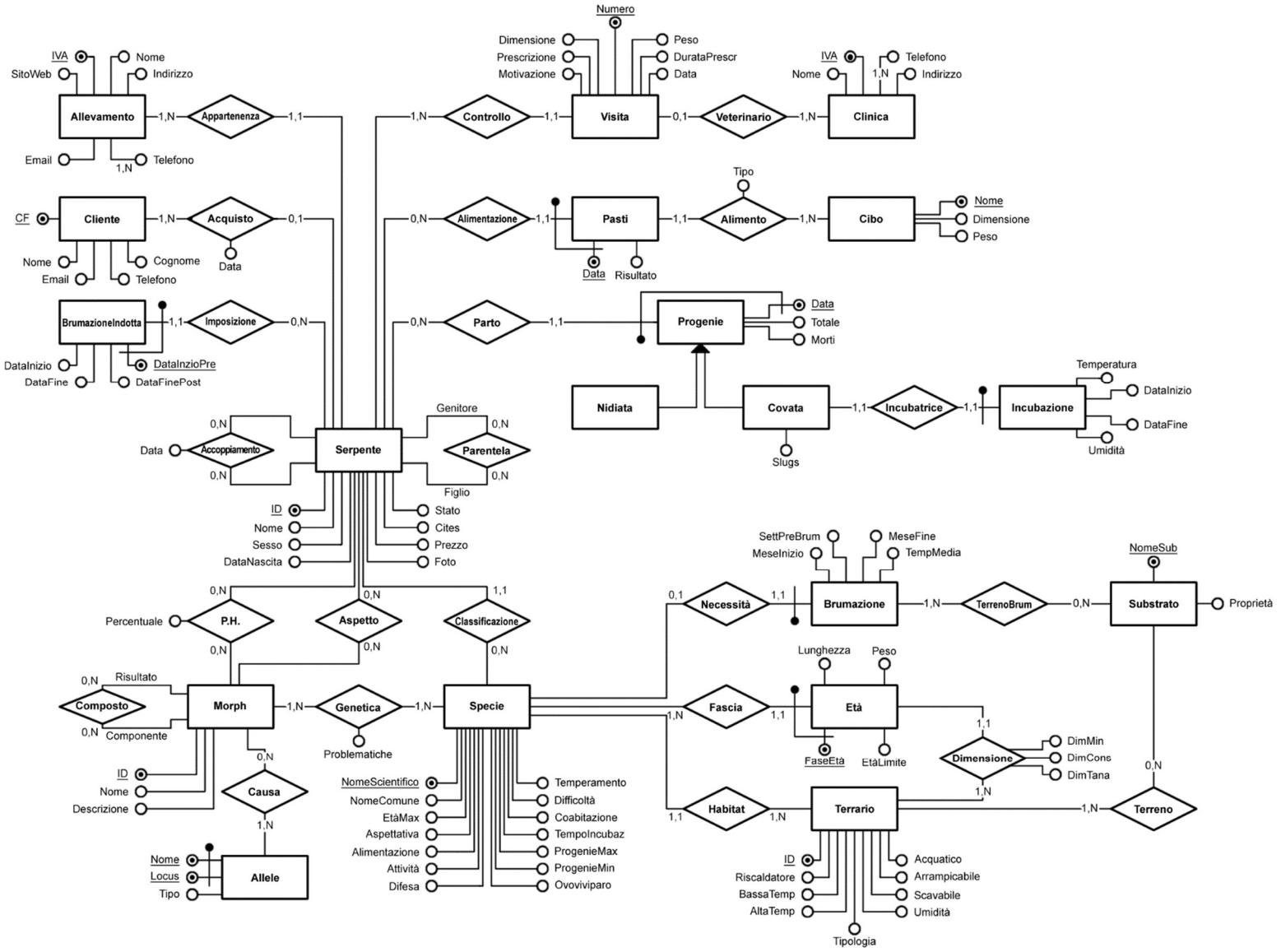
Di ogni morph rappresentiamo il nome e la sua descrizione.

Un morph può essere causato da combinazioni di due alleli o da combinazioni di più morph. Esistono morph che non seguono le leggi mendeliane e sono ottenibili solo tramite anomalie e accoppiamento selettivo di esemplari unici.

Per gli alleli che causano dei morph, rappresentiamo il nome, il locus e il tipo[20] di allele.

[13] Mediamente, la brumazione è preceduta da tre settimane di digiuno e riduzione delle temperature ed è seguita da altrettante settimane di digiuno con un graduale aumento delle temperature.
[14] Materiale posto all'interno dei terrari per emulare il terreno dell'habitat di una specie.
[15] Lampada riscaldante, tappetino riscaldante o entrambi.
[16] Plastica opaca, trasparente o entrambi. Dipende principalmente dal periodo di attività del serpente e dal metodo di riscaldamento.
[17] Caratteristica esteriore del serpente dettata da fattori genetici. Assieme alla specie e all'età, sono il fattore che detta il prezzo del serpente. La scelta e la previsione dei morph è la base dell'attività di un allevatore di rettili.
[18] Una specie può presentare dei morph che hanno un'alta possibilità di generare problemi di natura genetica sulla prole.
[19] Un serpente si dice che è het per un morph quando sappiamo che il serpente possiede un solo allele recessivo. Se un serpente è het per un morph, vuol dire che questo morph non è visibile ma che la progenie del serpente, nelle condizioni giuste, potrebbe palesarlo. Se non si ha la certezza che un serpente sia het per un morph, si parla di possible het. Nel caso un serpente sia possible het, si fa una stima della probabilità che sia het usando le leggi di Mendel.
[20] Dominante, codominante e recessivo. Un allele causa la comparsa di almeno un morph. Nel caso sia codominante almeno due.

Schema Concettuale



Dizionario dei dati

Entità

ENTITÀ	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORI
Allele	Alleli dei morph delle specie	Tipo	Nome, Locus
Allevamento	Strutture di allevamento dell'Associazione	Nome, SitoWeb, Indirizzo, Email, Telefono	IVA
Brumazione	Fasi di brumazione delle specie	SettPreBrum, MeseInizio, MeseFine, TempMedia	
BrumazioneIndotta	Registro brumazioni dei serpenti degli allevamenti	DataInizio, DataFine, DataFinePost	DataInizioPre
Cibo	Cibi adatti ai serpenti	Dimensione, Peso	Nome
Cliente	Persona che ha acquistato o adottato un serpente degli allevamenti	Nome, Cognome, Telefono, Email	CF
Clinica	Struttura responsabile di visite veterinarie	Nome, Telefono, Indirizzo	IVA
Covata	Uova deposte	Slugs	
Età	Dimensioni e pesi di specie secondo fasi di età	Lunghezza, Peso, EtàLimite	FaseEtà
Incubazione	Incubazioni delle uova dei serpenti allevati	Temperatura, DataInizio, DataFine, Umidità	
Morph	Morph delle specie	Nome, Descrizione	ID
Nidiata	Serpenti partoriti		
Pasti	Registro dei pasti dei serpenti allevati	Risultato	Data
Progenie	Insieme delle progenie dei	Totale, Morti	Data
Serpente	Serpente che è o che è stato di proprietà di un allevamento dell'Associazione	Nome, Sesso, DataNascita, Stato, Cites, Prezzo, Foto	ID
Specie	Specie di cui fanno parte i serpenti	NomeComune, EtàMax, Aspettativa, Alimentazione, Attività, Difesa, Temperamento, Difficoltà, Coabitazione, TempoIncubaz, ProgenieMax, ProgenieMin, Ovoviviparo	NomeScientifico
Substrato	Terreno dei Terrari	Proprietà	Nome
Terrario	Habitat artificiale dove vengono mantenuti i serpenti	Riscaldatore, BassaTemp, AltaTemp, Umidità, Tipologia, Acquatico, Arrampicabile, Scavabile	ID
Visita	Registro visite veterinarie e di controllo	Dimensione, Peso, Prescrizione, DurataPrescr, Motivazione, Data	Numero

Relazioni

RELAZIONE	DESCRIZIONE	COMPONENTI	ATTRIBUTI
Accoppiamento	Accoppiamenti fra serpenti	Serpente	Data
Acquisto	Acquisti o adozioni di serpenti da parte dei clienti	Serpente, Cliente	Data
Alimentazione	Pasti consumati dal serpente	Serpente, Pasti	
Alimento	Cibo che compone un pasto e modalità di somministrazione	Pasti, Cibo	Tipo
Appartenenza	Allevamento di appartenenza del serpente	Serpente, Allevamento	
Aspetto	Morphs del serpente	Serpente, Morph	
Causa	Alleli che causano il morph	Morph, Allele	
Classificazione	Specie di appartenenza del serpente	Serpente, Specie	
Composto	Morph ottenuto per composizione di morphs	Morph	
Controllo	Visite di controllo e mediche del serpente	Serpente, Visita	
Dimensione	Dimensioni relative al terrario per fasce di età della specie	Terrario, Età	DimMin, DimCons, DimTana
Fascia	Fasce di età della specie	Specie, Età	
Genetica	Morphs della specie e possibili problematiche	Specie, Morph	Problematiche
Habitat	Terrario per ospitare serpenti di una specie	Specie, Terrario	
Imposizione	Brumazioni artificiali di un serpente	Serpente, BrumazioneIndotta	
Incubatrice	Incubazione di una covata	Covata, Incubazione	
Necessità	Brumazione naturale della specie	Specie, Brumazione	
P.H.	Probabilità che il serpente sia het per un morph	Serpente, Morph	Percentuale
Parentela	Legami di parentela fra serpenti	Serpente	
Parto	Progenie di un serpente	Serpente, Progenie	
Terreno	Terreni adatti a un terrario	Terrario, Substrato	
TerrenoBrum	Terreni adatti per la brumazione di una specie	Brumazione, Substrato	
Veterinario	Clinica che si occupa di alcune visite mediche	Visita, Clinica	

Vincoli non esprimibili

- Due serpenti vivi e non venduti o adottati dello stesso allevamento non possono avere lo stesso nome.
- Se tutti i serpenti adottati o acquistati da un cliente sono morti, si cancella il cliente.
- Se un serpente è in vendita, in adozione o è stato venduto o adottato, allora il prezzo, la foto e il Cites sono obbligatori.
- Sono interessato a tenere il registro dei pasti, delle brumazioni e della progenie dei soli serpenti vivi di proprietà degli allevamenti.
- I sessi dei serpenti in accoppiamento e dei genitori in parentela devono essere opposti.
- Un serpente figlio non può avere più di due serpenti genitori.
- In BrumazioneIndotta, Brumazione e Incubazione, i le date finali non devono precedere quelle iniziali.
- In BrumazioneIndotta rappresentiamo solamente le brumazioni dei serpenti vivi e di proprietà dell'allevamento.
- I serpenti di fascia baby non possono essere brumati.
- È possibile interrompere la brumazione di un serpente.
- I serpenti vivi di proprietà dell'allevamento devono eseguire almeno due visite all'anno.
- Alla nascita e alla morte di un serpente dell'allevamento deve essere eseguita una visita.
- Alla morte di un serpente è necessario eliminare manualmente alcune visite.
- Nel registro pasti vogliamo mantenere solo l'ultimo anno di pasti e solo dei serpenti vivi e ancora di proprietà degli allevamenti.
- Un serpente può essere messo in vendita solo se ha consumato con successo almeno due pasti.
- La progenie di un serpente deve essere una covata o una nidiata in base all'attributo ovoviviparo di Specie.
- Quando l'età di un serpente di proprietà di un cliente raggiunge l'aspettativa di vita della specie, viene considerato morto.
- Una specie ha sempre tre fascie di età.
- La temperatura della zona calda dei terrari deve essere maggiore della temperatura della zona fredda.
- La dimensione minima dei terrari deve essere minore di quella consigliata.
- Un morph è causato da non più di due alleli.
- Associamo gli alleli ai soli morph non composti.
- Il nome comune di una specie deve essere diverso dal nome comune delle altre specie.
- Un morph composto ha come componenti morph semplici.

Ristrutturazione schema E/R

Tabella dei Volumi

ENTITÀ	VOLUME
Allele	1000
Allevamento	25
Brumazione	30
BrumazioneIndotta	600
Cibo	50
Cliente	16200
Clinica	75
Covata	1200
Età	150
Incubazione	1200
Morph	8000
Nidiata	300
Pasti	24960
Progenie	1500
Serpente	23500
Specie	50
Substrato	15
Terrario	50
Visita	12000

RELAZIONE	VOLUME
Accoppiamento	3000
Acquisto	20250
Alimentazione	24960
Alimento	24960
Appartenenza	23500
Aspetto	35250
Causa	1200
Classificazione	23500
Composto	20400
Controllo	12000
Dimensione	150
Fascia	150
Genetica	50000
Habitat	50
Imposizione	600
Incubatrice	1200
Necessità	30
P.H.	35250
Parentela	30550
Parto	1500
Terreno	100
TerrenoBrum	100
Veterinario	9600

Stime effettuate per l'ottenimento dei volumi:

- Si suppone che i volumi siano riferiti a 4 anni di operatività di un'associazione di 25 allevamenti.
- Si stima una media di 40 serpenti per allevamento, di cui 15 femmine riproduttrici.
- Si considera che un serpente riproduttore produca una prole di 15 serpenti all'anno. Si suppone che il 20% dei serpenti sia ovoviviparo e che un serpente consumi una media di 0,48 pasti alla settimana. Per ogni serpente si suppone di conoscere solo il 30% delle volte il padre, mentre la madre è sempre nota.
- Stimando che il 90% dei serpenti prodotti ogni anno venga venduto e che un cliente possieda in media 1,25 serpenti, si ha un totale di 16200 clienti serviti nei 4 anni.
- Si suppone un numero medio di 3 visite annue per serpente, di cui l'80% effettuate da una clinica. Si considera che un allevamento faccia riferimento in media a tre cliniche.
- Si assume che un serpente abbia 1,5 morphs in media e che sia possible het per altrettanti. Assunto che ci siano 1000 alleli che causano morph semplici, si suppone che un allele sia causa di 1,2 morph semplici. Si suppone che ogni morph composto sia composto da una media di tre morph. Si stima che una specie possieda 1000 morph.
- Si suppone che ogni terrario e ogni brumazione utilizzi in media due substrati.

Eliminazione delle generalizzazioni

Nello schema E/R proposto è presente la generalizzazione di Nidiata e Covata con l'entità Progenie. La generalizzazione è totale esclusiva e si prevede che gli accessi alle entità figlie siano concorrenti all'entità padre ma indipendenti tra di loro. Pertanto, la scelta preferibile è quella di accorpare l'entità padre nelle figlie.

Partizionamento e accorpamento di entità e relazioni

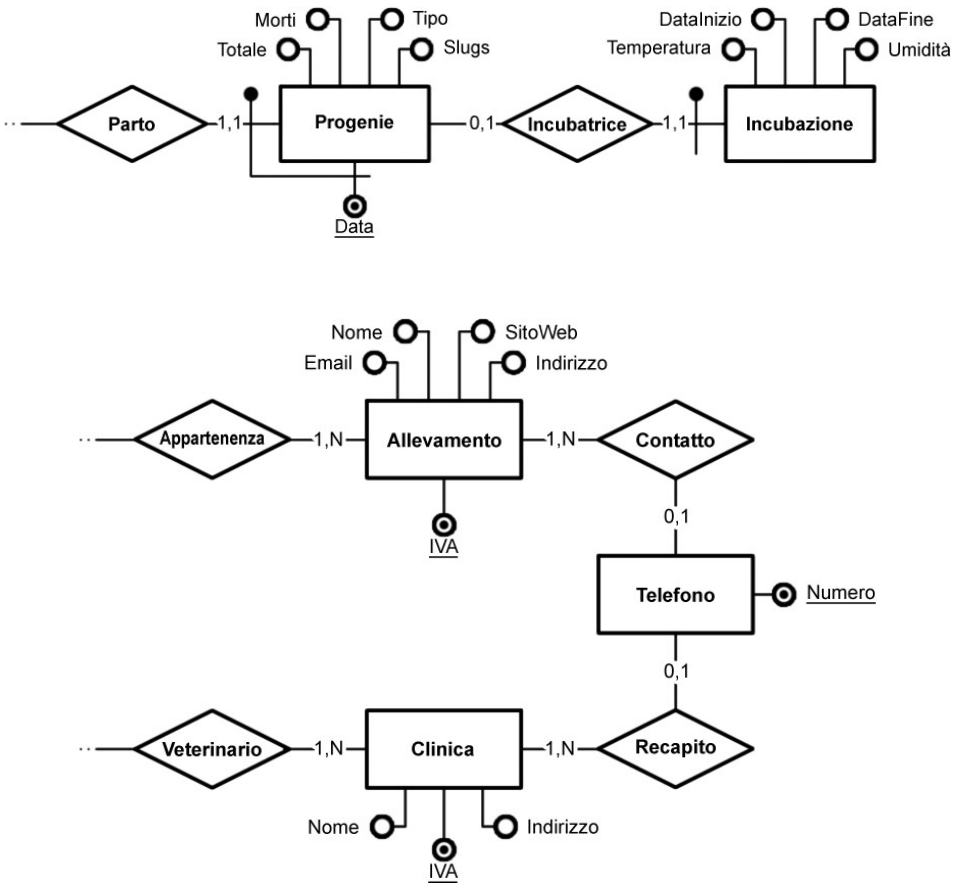
Precedentemente si è divisa la generalizzazione in due entità figlie. Tuttavia, poiché quella che era l'entità figlia Nidiata non aveva attributi propri o associazioni, si è deciso di accorpare le entità Nidiata e Covata per semplicità. Di fatto, in questo modo, è come se avessimo accorpato le entità figlie nel padre nel passaggio di eliminazione delle generalizzazioni.

Sono presenti degli attributi multivalore Telefono sia nell'entità Allevamento che nell'entità Clinica. Pertanto, è stata aggiunta un'entità Telefono associata sia ad Allevamento che a Clinica.

Scelta degli identificatori primari

Per ragioni di semplicità, gli identificatori sono già stati rappresentati nel modello E/R. Gli identificatori interni sono stati rappresentati con un pallino nero e sottolineando l'attributo. Gli identificatori esterni e composti sono rappresentati tramite un segmento che termina con un pallino nero e che interseca gli identificatori interni o i rami delle associazioni.

Risultato ristrutturazione



Modello Logico

Schema Finale

Di seguito è riportato l'elenco delle intestazioni delle tabelle che compongono lo schema finale. In grassetto è indicato il nome della tabella, mentre tra parentesi sono indicati gli attributi. Gli attributi sottolineati rappresentano le chiavi primarie, gli attributi in azzurro rappresentano gli attributi con vincolo di integrità referenziale:

Allevamento (IVA, Nome, SitoWeb, Indirizzo, Email)

Cliente (CF, Nome, Cognome, Email, Telefono)

Serpente (ID, Nome, Sesso, DataNascita, Stato, Cites, Prezzo, Foto, IVAAllevamento, CFCliente,

DataAcquisto, NomeSpecie)

BrumazioneIndotta (DataInizioPre, IDSerpente, DataInizio, DataFine, DataFinePost)

Clinica (IVA, Nome, Indirizzo)

Telefono (Numero, IVAAllevamento, IVAClinica)

Visita (Numero, Dimensione, Peso, Prescrizione, DurataPrescrizione, Motivazione, Data, IDSerpente, IVAClinica)

Cibo (Nome, Dimensione, Peso)

Pasti (Data, IDSerpente, Risultato, NomeCibo, Tipo)

Progenie (Data, IDSerpente, tipo, totale, morti, slugs)

Incubazione (DataParto, IDSerpente, DataInizio, DataFine, Temperatura, Umidita)

Morph (ID, Nome, Descrizione)

Allele (Nome, Locus, Tipo)

Terrario (ID, Riscaldatore, BassaTemp, AltaTemp, Tipologia, Acquatico, Arrampicabile, Scavabile, Umidita)

Specie (NomeScientifico, NomeComune, EtaMax, Aspettativa, Alimentazione, Attivita, Difesa, Temperamento, Difficolta, Coabitazione, TempoIncubaz, ProgenieMax, ProgenieMin, Ovoviviparo, IDTerrario)

Brumazione (NomeSpecie, SettPreBrum, MeseInizio, MeseFine, TempMedia)

SubStrato (NomeSub, Proprietà)

Eta (FaseEta, NomeSpecie, Lunghezza, Peso, EtaLimite, IDTerrario, DimMinTerrario, DimConsTerrario, DimTana)

Accoppiamento (IDSerpenteM, IDSerpenteF, Data)

Parentela (IDGenitore, IDFiglio)

PH (IDSerpente, IDMorph, Percentuale)

Aspetto (IDSerpente, IDMorph)

Composto (IDComponente, IDRisultato)

Causa (IDMorph, NomeAllele, LocusAllele)

Genetica (IDMorph, NomeSpecie, Problematiche)

TerrenoBrum (NomeSpecie, NomeSubstrato)

Terreno (IDTerrario, NomeSubstrato)

DDL:

Segue l'elenco dei comandi DDL per la creazione delle tabelle:

Allevamento (IVA, Nome, SitoWeb, Indirizzo, Email)

```
01 |CREATE TABLE Allevamento(
02 |IVA BIGINT(11) UNSIGNED,
03 |Nome VARCHAR(150) NOT NULL,
04 |SitoWeb VARCHAR(63) DEFAULT NULL,
05 |Indirizzo VARCHAR(200) NOT NULL,
06 |Email VARCHAR(150) NOT NULL,
07 |
08 |PRIMARY KEY (IVA)
09 |);
```

Cliente (CF, Nome, Cognome, Email, Telefono)

```
01 |CREATE TABLE Cliente(
02 |CF CHAR(16),
03 |Nome VARCHAR(100) NOT NULL,
04 |Cognome VARCHAR(100) NOT NULL,
05 |Email VARCHAR(150) NOT NULL,
06 |Telefono BIGINT(11) UNSIGNED NOT NULL,
07 |
08 |PRIMARY KEY (CF)
09 |);
```

Serpente (ID, Nome, Sesso, DataNascita, Stato, Cites, Prezzo, Foto, IVAAllevamento, CFCliente,

DataAcquisto, NomeSpecie)

```
01 |CREATE TABLE Serpente(
02 |ID INT UNSIGNED AUTO_INCREMENT,
03 |Nome VARCHAR(70) NOT NULL,
04 |Sesso CHAR(1) NOT NULL,
05 |DataNascita DATE NOT NULL,
06 |Stato ENUM("In vendita", "Venduto", "In adozione", "Adottato", "Non in vendita", "Morto", "Riproduttore")
    DEFAULT "Non in vendita" NOT NULL,
07 |Cites VARCHAR(150) UNIQUE DEFAULT NULL,
08 |Prezzo SMALLINT UNSIGNED DEFAULT NULL,
09 |Foto VARCHAR(150) UNIQUE DEFAULT NULL,
10 |IVAAllevamento BIGINT(11) UNSIGNED NOT NULL,
11 |CFCliente CHAR(16) DEFAULT NULL,
12 |DataAcquisto DATE DEFAULT NULL,
13 |NomeSpecie VARCHAR(100) NOT NULL,
14 |
15 |PRIMARY KEY (ID),
16 |CONSTRAINT FKSerpente_IVAAllevamento FOREIGN KEY (IVAAllevamento) REFERENCES Allevamento(IVA)
17 |ON DELETE CASCADE ON UPDATE CASCADE,
18 |CONSTRAINT FKSerpente_CFCliente FOREIGN KEY (CFCliente) REFERENCES Cliente(CF)
19 |ON DELETE SET NULL ON UPDATE CASCADE,
20 |CONSTRAINT FKSerpente_NomeSpecie FOREIGN KEY (NomeSpecie) REFERENCES Specie(NomeScientifico)
21 |ON DELETE CASCADE ON UPDATE CASCADE);
```

BrumazioneIndotta (DataInizioPre, **IDSerpente**, DataInizio, DataFine, DataFinePost)

```
01 |CREATE TABLE BrumazioneIndotta(
02 |DataInizioPre DATE NOT NULL,
03 |IDSerpente INT UNSIGNED NOT NULL,
04 |DataInizio DATE DEFAULT NULL,
05 |DataFine DATE DEFAULT NULL,
06 |DataFinePost DATE DEFAULT NULL,
07 |
08 |PRIMARY KEY (DataInizioPre, IDSerpente),
09 |CONSTRAINT FKBrumazioneIndotta_IDSerpente FOREIGN KEY (IDSerpente) REFERENCES Serpente(ID)
10 |ON DELETE CASCADE ON UPDATE CASCADE
11 |);
```

Clinica (IVA, Nome, Indirizzo)

```
01 |CREATE TABLE Clinica(
02 |IVA BIGINT(11) UNSIGNED,
03 |Nome VARCHAR(150) NOT NULL,
04 |Indirizzo VARCHAR(200) NOT NULL,
05 |
06 |PRIMARY KEY (IVA)
07 |);
```

Telefono (Numero, IVAAllevamento, IVAClinica)

```
01 |CREATE TABLE Telefono(
02 |Numero BIGINT(11) UNSIGNED,
03 |IVAAllevamento BIGINT(11) UNSIGNED DEFAULT NULL,
04 |IVAClinica BIGINT(11) UNSIGNED DEFAULT NULL,
05 |
06 |PRIMARY KEY (Numero),
07 |CONSTRAINT FKTelefono_IVAAllevamento FOREIGN KEY (IVAAllevamento) REFERENCES Allevamento(IVA)
08 |ON DELETE CASCADE ON UPDATE CASCADE,
09 |CONSTRAINT FKTelefono_IVAClinica FOREIGN KEY (IVAClinica) REFERENCES Clinica(IVA)
10 |ON DELETE CASCADE ON UPDATE CASCADE
11 |);
```

Visita (Numero, Dimensione, Peso, Prescrizione, DurataPrescrizione, Motivazione, Data, IDSerpente, IVAClinica)

```
01 |CREATE TABLE Visita(
02 |Numero INT UNSIGNED AUTO_INCREMENT,
03 |Dimensione SMALLINT UNSIGNED NOT NULL,
04 |Peso TINYINT UNSIGNED NOT NULL,
05 |Prescrizione VARCHAR(4000) DEFAULT NULL,
06 |DurataPrescrizione VARCHAR(20) DEFAULT NULL,
07 |Motivazione VARCHAR(300) NOT NULL,
08 |Data DATE NOT NULL,
09 |IDSerpente INT UNSIGNED NOT NULL,
10 |IVAClinica BIGINT(11) UNSIGNED DEFAULT NULL,
11 |
12 |PRIMARY KEY (Numero),
13 |CONSTRAINT FKVisita_IDSerpente FOREIGN KEY (IDSerpente) REFERENCES Serpente(ID)
14 |ON DELETE CASCADE ON UPDATE CASCADE,
15 |CONSTRAINT FKVisita_IVAClinica FOREIGN KEY (IVAClinica) REFERENCES Clinica(IVA)
16 |ON DELETE SET NULL ON UPDATE CASCADE
17 |);
```

Cibo (Nome, Dimensione, Peso)

```
01 |CREATE TABLE Cibo(
02 |Nome VARCHAR(60),
03 |Dimensione TINYINT UNSIGNED NOT NULL,
04 |Peso TINYINT UNSIGNED NOT NULL,
05 |
06 |PRIMARY KEY (Nome)
07 |);
```

Pasti (Data, IDSerpente, Risultato, NomeCibo, Tipo)

```
01 |CREATE TABLE Pasti(
02 |Data DATE NOT NULL,
03 |IDSerpente INT UNSIGNED NOT NULL,
04 |Risultato ENUM(“Consumato”, “Rifiutato”, “Rigurgitato”) DEFAULT “Consumato” NOT NULL,
05 |Tipo ENUM(“Scongelato”, “Pre-ucciso”, “Vivo”) DEFAULT “Scongelato” NOT NULL,
06 |NomeCibo VARCHAR(60) NOT NULL,
07 |
08 |PRIMARY KEY (Data, IDSerpente),
09 |CONSTRAINT FKPasti_IDSerpente FOREIGN KEY (IDSerpente) REFERENCES Serpente(ID)
10 |ON DELETE CASCADE ON UPDATE CASCADE,
11 |CONSTRAINT FKPasti_NomeCibo FOREIGN KEY (NomeCibo) REFERENCES Cibo(Nome)
12 |ON DELETE RESTRICT ON UPDATE CASCADE
13 |);
```

Progenie (Data, IDSerpente, tipo, totale, morti, slugs)

```
01 |CREATE TABLE Progenie(
02 |Data DATE NOT NULL,
03 |IDSerpente INT UNSIGNED NOT NULL,
04 |Tipo ENUM("Covata", "Nidiata") DEFAULT "Covata" NOT NULL,
05 |Totale TINYINT UNSIGNED NOT NULL,
06 |Morti TINYINT UNSIGNED NOT NULL,
07 |Slugs TINYINT UNSIGNED DEFAULT 0,
08 |
09 |PRIMARY KEY (Data, IDSerpente),
10 |CONSTRAINT FKProgenie_IDSerpente FOREIGN KEY (IDSerpente) REFERENCES Serpente(ID)
11 |ON DELETE CASCADE ON UPDATE CASCADE
12 |);
```

Incubazione (DataParto, IDSerpente, DataInizio, DataFine, Temperatura, Umidita)

```
01 |CREATE TABLE Incubazione(
02 |DataParto DATE NOT NULL,
03 |IDSerpente INT UNSIGNED NOT NULL,
04 |DataInizio DATE NOT NULL,
05 |DataFine DATE DEFAULT NULL,
06 |Temperatura TINYINT UNSIGNED NOT NULL,
07 |Umidita TINYINT UNSIGNED NOT NULL,
08 |
09 |PRIMARY KEY (DataParto, IDSerpente),
10 |CONSTRAINT FKIncubazione_Progenie FOREIGN KEY (DataParto, IDSerpente)
11 |REFERENCES Progenie(Data, IDSerpente)
12 |ON DELETE CASCADE ON UPDATE CASCADE
13 |);
```

Morph (ID, Nome, Descrizione)

```
01 |CREATE TABLE Morph(
02 |ID INT UNSIGNED AUTO_INCREMENT,
03 |Nome VARCHAR(100) NOT NULL,
04 |Descrizione VARCHAR(1000) NOT NULL,
05 |
06 |PRIMARY KEY(ID)
07 |);
```

Allele (Nome, Locus, Tipo)

```
01 |CREATE TABLE Allele(
02 |Nome VARCHAR(100) NOT NULL,
03 |Locus VARCHAR(100) NOT NULL,
04 |Tipo ENUM("Dominante", "Codominante", "Recessivo") NOT NULL,
05 |
06 |PRIMARY KEY (Nome, Locus)
07 |);
```

Terrario (ID, Riscaldatore, BassaTemp, AltaTemp, Tipologia, Acquatico, Arrampicabile, Scavabile, Umidita)

```
01 |CREATE TABLE Terrario(
01 |ID INT UNSIGNED AUTO_INCREMENT,
02 |Riscaldatore ENUM("Tappetino", "Lampada", "Entrambe") DEFAULT "Tappetino" NOT NULL,
03 |AltaTemp TINYINT NOT NULL,
04 |BassaTemp TINYINT NOT NULL,
05 |Tipologia ENUM("Opaco", "Trasparente","Entrambi") DEFAULT "Entrambi" NOT NULL,
06 |Acquatico BOOL DEFAULT FALSE NOT NULL,
07 |Arrampicabile BOOL DEFAULT FALSE NOT NULL,
08 |Scavabile BOOL DEFAULT FALSE NOT NULL,
09 |Umidita TINYINT NOT NULL,
10 |
11 |PRIMARY KEY (ID)
12 |);
```

Specie (NomeScientifico, NomeComune, EtaMax, Aspettativa, Alimentazione, Attivita, Difesa, Temperamento, Difficolta, Coabitazione, TempoIncubaz, ProgenieMax, ProgenieMin, Ovoviviparo, IDTerrario)

```
01 |CREATE TABLE Specie(
02 |NomeScientifico VARCHAR(100),
03 |NomeComune VARCHAR(100) NOT NULL UNIQUE,
04 |EtaMax TINYINT UNSIGNED NOT NULL,
05 |Aspettativa TINYINT UNSIGNED NOT NULL,
06 |Alimentazione ENUM("Roditori", "Piccoli Mammiferi", "Pesci", "Uova", "Vermi") DEFAULT "Roditori" NOT NULL,
07 |Attivita ENUM("Diurno", "Notturno", "Crepuscolare") DEFAULT "Crepuscolare" NOT NULL,
08 |Difesa ENUM("Morso", "Veleno", "Tanatosi", "Intimidazione", "Musk", "Avviluppamento") DEFAULT "Morso" NOT
    NULL,
09 |Temperamento ENUM("Schivo", "Aggressivo", "Calmo", "Attivo") DEFAULT "Calmo" NOT NULL,
10 |Difficolta ENUM("Principiante", "Intermedio", "Esperto") DEFAULT "Intermedio" NOT NULL,
11 |Coabitazione BOOL DEFAULT FALSE NOT NULL,
12 |TempoIncubaz TINYINT UNSIGNED DEFAULT NULL,
13 |ProgenieMax TINYINT UNSIGNED NOT NULL,
14 |ProgenieMin TINYINT UNSIGNED NOT NULL,
15 |Ovoviviparo BOOL DEFAULT FALSE NOT NULL,
16 |IDTerrario INT UNSIGNED NOT NULL,
17 |
18 |PRIMARY KEY (NomeScientifico),
19 |CONSTRAINT FKSpecie_IDTerrario FOREIGN KEY (IDTerrario) REFERENCES Terrario(ID)
20 |ON DELETE RESTRICT ON UPDATE CASCADE
21 |);
```

Brumazione (NomeSpecie, SettPreBrum, MeseInizio, MeseFine, TempMedia)

```
01 |CREATE TABLE Brumazione(
02 |NomeSpecie VARCHAR(100),
03 |SettPreBrum TINYINT UNSIGNED NOT NULL,
04 |MeseInizio ENUM("Gen", "Feb", "Mar", "Apr", "Mag", "Giu", "Lug", "Ago", "Set", "Ott", "Nov", "Dic") NOT NULL,
05 |MeseFine ENUM("Gen", "Feb", "Mar", "Apr", "Mag", "Giu", "Lug", "Ago", "Set", "Ott", "Nov", "Dic") NOT NULL,
06 |TempMedia TINYINT UNSIGNED NOT NULL,
07 |
08 |PRIMARY KEY (NomeSpecie),
09 |CONSTRAINT FKBrumazione_NomeSpecie FOREIGN KEY (NomeSpecie) REFERENCES Specie(NomeScientifico)
10 |ON DELETE CASCADE ON UPDATE CASCADE);
```

SubStrato (NomeSub, Proprietà)

```
01 |CREATE TABLE Substrato(
02 |NomeSub VARCHAR(100),
03 |Proprietà VARCHAR(400) NOT NULL,
04 |
05 |PRIMARY KEY (NomeSub)
06 |);
```

Eta (FaseEta, NomeSpecie, Lunghezza, Peso, EtaLimite, IDTerrario, DimMinTerrario, DimConsTerrario, DimTana)

```
01 |CREATE TABLE Eta(
02 |FaseEta ENUM("Baby", "Sub-adulto", "Adulto") NOT NULL,
03 |NomeSpecie VARCHAR(100) NOT NULL,
04 |Lunghezza TINYINT UNSIGNED NOT NULL,
05 |Peso TINYINT UNSIGNED NOT NULL,
06 |EtaLimite TINYINT UNSIGNED,
07 |IDTerrario INT UNSIGNED NOT NULL,
08 |DimMinTerrario VARCHAR(7) NOT NULL,
09 |DimConsTerrario VARCHAR(7) NOT NULL,
10 |DimTana VARCHAR(7) NOT NULL,
11 |
12 |PRIMARY KEY (FaseEta, NomeSpecie),
13 |CONSTRAINT FKEta_NomeSpecie FOREIGN KEY (NomeSpecie) REFERENCES Specie(NomeScientifico)
14 |ON DELETE CASCADE ON UPDATE CASCADE,
15 |CONSTRAINT FKEta_IDTerrario FOREIGN KEY (IDTerrario) REFERENCES Terrario(ID)
16 |ON DELETE RESTRICT ON UPDATE CASCADE
17 |);
```

Accoppiamento (IDSerpenteM, IDSerpenteF, Data)

```
01 |CREATE TABLE Accoppiamento(
02 |IDSerpenteM INT UNSIGNED NOT NULL,
03 |IDSerpenteF INT UNSIGNED NOT NULL,
04 |Data DATE,
05 |
06 |PRIMARY KEY (IDSerpenteM, IDSerpenteF),
07 |CONSTRAINT FKAccoppiamento_IDSerpenteM FOREIGN KEY (IDSerpenteM) REFERENCES Serpente(ID)
08 |ON DELETE CASCADE ON UPDATE CASCADE,
09 |CONSTRAINT FKAccoppiamento_IDSerpenteF FOREIGN KEY (IDSerpenteF) REFERENCES Serpente(ID)
10 |ON DELETE CASCADE ON UPDATE CASCADE
11 |);
```

Parentela (IDGenitore, IDFiglio)

```
CREATE TABLE Parentela(
IDGenitore INT UNSIGNED NOT NULL,
IDFiglio INT UNSIGNED NOT NULL,

PRIMARY KEY (IDGenitore, IDFiglio),
CONSTRAINT FKParentela_IDGenitore FOREIGN KEY (IDGenitore) REFERENCES Serpente(ID)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT FKParentela_IDFiglio FOREIGN KEY (IDFiglio) REFERENCES Serpente(ID)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

PH (IDSerpente, IDMorph, Percentuale)

```
CREATE TABLE PH(
IDSerpente INT UNSIGNED NOT NULL,
IDMorph INT UNSIGNED NOT NULL,
Percentuale TINYINT NOT NULL,

PRIMARY KEY (IDSerpente, IDMorph),
CONSTRAINT FKPH_IDSerpente FOREIGN KEY (IDSerpente) REFERENCES Serpente(ID)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT FKPH_IDMorph FOREIGN KEY (IDMorph) REFERENCES Morph(ID)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

Aspetto (IDSerpente, IDMorph)

```
CREATE TABLE Aspetto(
IDSerpente INT UNSIGNED NOT NULL,
IDMorph INT UNSIGNED NOT NULL,

PRIMARY KEY (IDSerpente, IDMorph),
CONSTRAINT FKAspetto_IDSerpente FOREIGN KEY (IDSerpente) REFERENCES Serpente(ID)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT FKAspetto_IDMorph FOREIGN KEY (IDMorph) REFERENCES Morph(ID)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

Composto (IDComponente, IDRisultato)

```
CREATE TABLE Composto(
IDComponente INT UNSIGNED NOT NULL,
IDRisultato INT UNSIGNED NOT NULL,

PRIMARY KEY (IDComponente, IDRisultato),
CONSTRAINT FKComposto_IDComponente FOREIGN KEY (IDComponente) REFERENCES Morph(ID)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT FKComposto_IDRisultato FOREIGN KEY (IDRisultato) REFERENCES Morph(ID)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

Causa (IDMorph, NomeAllele, LocusAllele)

```
01 |CREATE TABLE Causa(
02 |IDMorph INT UNSIGNED NOT NULL,
03 |NomeAllele VARCHAR(100) NOT NULL,
04 |LocusAllele VARCHAR(100) NOT NULL,
05 |
06 |PRIMARY KEY (IDMorph, NomeAllele, LocusAllele),
07 |CONSTRAINT FKCausa_IDMorph FOREIGN KEY (IDMorph) REFERENCES Morph(ID)
08 |ON DELETE CASCADE ON UPDATE CASCADE,
09 |CONSTRAINT FKCausa_Allele FOREIGN KEY (NomeAllele, LocusAllele) REFERENCES Allele(Nome, Locus)
10 |ON DELETE CASCADE ON UPDATE CASCADE
11 |);
```

Genetica (IDMorph, NomeSpecie, Problematiche)

```
01 |CREATE TABLE Genetica(
02 |IDMorph INT UNSIGNED NOT NULL,
03 |NomeSpecie VARCHAR(100) NOT NULL,
04 |Problematiche VARCHAR(1000),
05 |
06 |PRIMARY KEY (IDMorph, NomeSpecie),
07 |CONSTRAINT FKGenetica_IDMorph FOREIGN KEY (IDMorph) REFERENCES Morph(ID)
08 |ON DELETE CASCADE ON UPDATE CASCADE,
09 |CONSTRAINT FKGenetica_NomeSpecie FOREIGN KEY (NomeSpecie) REFERENCES Specie(NomeScientifico)
10 |ON DELETE CASCADE ON UPDATE CASCADE
11 |);
```

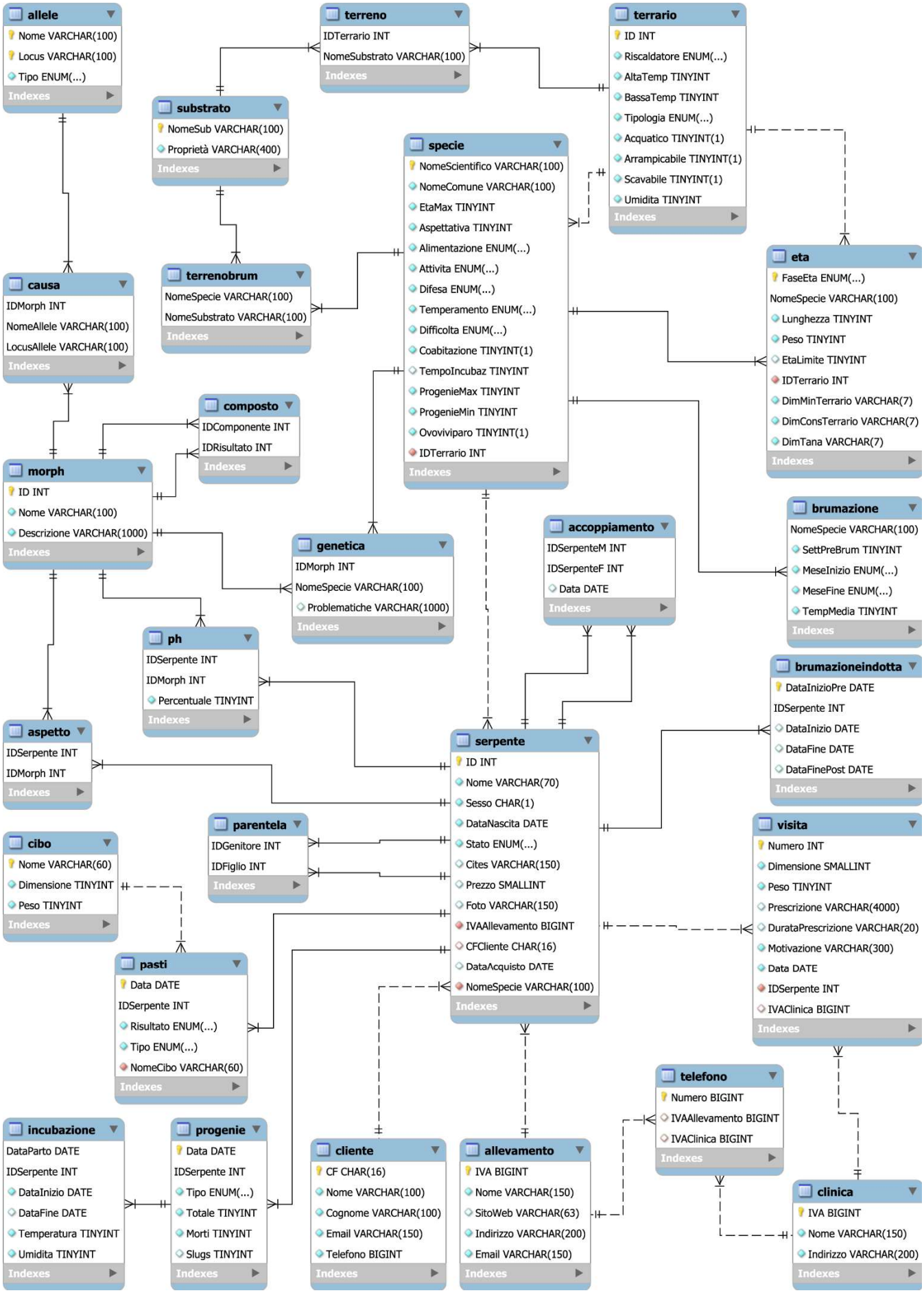
TerrenoBrum (NomeSpecie, NomeSubstrato)

```
01 |CREATE TABLE TerrenoBrum(
02 |NomeSpecie VARCHAR(100) NOT NULL,
03 |NomeSubstrato VARCHAR(100) NOT NULL,
04 |
05 |PRIMARY KEY (NomeSpecie, NomeSubstrato),
06 |CONSTRAINT FKTerrenoBrum_NomeSpecie FOREIGN KEY (NomeSpecie)
07 |REFERENCES Specie(NomeScientifico)
08 |ON DELETE CASCADE ON UPDATE CASCADE,
09 |CONSTRAINT FKTerrenoBrum_NomeSubstrato FOREIGN KEY (NomeSubstrato)
10 |REFERENCES Substrato(NomeSub)
11 |ON DELETE CASCADE ON UPDATE CASCADE);
```

Terreno (IDTerrario, NomeSubstrato)

```
01 |CREATE TABLE Terreno(
02 |IDTerrario INT UNSIGNED NOT NULL,
03 |NomeSubstrato VARCHAR(100) NOT NULL,
04 |
05 |PRIMARY KEY (IDTerrario, NomeSubstrato),
06 |CONSTRAINT FKTerreno_IDTerrario FOREIGN KEY (IDTerrario) REFERENCES Terrario(ID)
07 |ON DELETE CASCADE ON UPDATE CASCADE,
08 |CONSTRAINT FKTerreno_NomeSubstrato FOREIGN KEY (NomeSubstrato) REFERENCES Substrato(NomeSub)
09 |ON DELETE CASCADE ON UPDATE CASCADE);
```

Schema Logico



Operazioni sulla base di dati

Sono state implementate alcune stored procedures, alcuni trigger e una user defined function. Segue un'elenco delle più rilevanti. Queste operazioni sono anche disponibili nei file SQL allegato per una maggiore leggibilità.

User defined function

Questa UDF permette di verificare se un periodo di date è stato inserito nell'ordine corretto.

```
01 |DELIMITER $$
02 |CREATE FUNCTION DateCheck(
03 |firstDate DATE, lastDate DATE)
04 |RETURNS BOOL
05 |DETERMINISTIC
06 |BEGIN
07 |    IF YEAR(lastDate) - YEAR(firstDate) < 0 THEN
08 |            RETURN FALSE;
09 |    ELSEIF YEAR(lastDate) - YEAR(firstDate) = 0 THEN
10 |            IF MONTH(lastDate) - MONTH(firstDate) < 0 THEN
11 |                    RETURN FALSE;
12 |            ELSEIF MONTH(lastDate) - MONTH(firstDate) = 0 THEN
13 |                    IF DAY(lastDate) - DAY(firstDate) <= 0 THEN
14 |                            RETURN FALSE;
15 |                    END IF;
16 |            END IF;
17 |    END IF;
18 |    RETURN TRUE;
19 |END $$ DELIMITER ;
```

Stored procedures

Sono state inserite alcune stored procedures per il popolamento del database. Seppur non particolarmente interessanti, permettono di applicare alcuni dei vincoli non esprimibili. Molte di queste stored procedures fanno uso di triggers per mantenere l'integrità della base di dati.

Stored procedure per l'aggiunta di nuovi serpenti:

```
01 |DROP PROCEDURE IF EXISTS AggiungiSerpente;
02 |DELIMITER $$
03 |CREATE PROCEDURE AggiungiSerpente(
04 |IN nome VARCHAR(70),
05 |IN sesso CHAR(1),
06 |IN dataNascita DATE,
07 |IN stato ENUM('In vendita','Venduto','In adozione','Adottato','Non in vendita','Morto','Riproduttore'),
08 |IN cites VARCHAR(150),
09 |IN foto VARCHAR(150),
10 |IN ivaAllevamento BIGINT UNSIGNED,
11 |IN nomeSpecie VARCHAR(100))
12 |BEGIN
13 |    DECLARE EXIT HANDLER FOR SQLSTATE '45001'
14 |            SHOW ERRORS;
15 |
16 |    IF Stato IN ('Venduto','Adottato', 'In vendita', 'In adozione') THEN
17 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato scelto uno stato non valido. Non è
    possibile inserire un nuovo serpente in vendita';
18 |    END IF;
19 |
20 |    INSERT INTO Serpente
21 |            VALUES (NULL, nome, sesso, dataNascita, stato, cites, NULL, foto, ivaAllevamento, NULL, NULL,
    nomeSpecie);
22 |END $$ DELIMITER;
```

Stored Procedure per l'aggiunta di pasti. Implementa anche l'eliminazione automatica dei record più vecchi di un anno e impedisce di inserire record multipli per lo stesso serpente nell'arco della giornata.

```
DELIMITER $$
CREATE PROCEDURE AggiungiPasto(
IN dataPasto DATE,
IN idSerpente INT UNSIGNED,
IN risultato ENUM('Consumato','Rifiutato','Rigurgitato'),
IN tipo ENUM('Scongelato','Pre-ucciso','Vivo'),
IN nomeCibo VARCHAR(60))
BEGIN
    DECLARE EXIT HANDLER FOR SQLSTATE '45001'
            SHOW ERRORS;

    IF (SELECT COUNT(ID) FROM Serpente WHERE ID = idSerpente) = 0 THEN
            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Serpente selezionato non esistente.';
    END IF;

    IF (SELECT COUNT(dataPasto) FROM Pasti p WHERE p.IDSerpente = idSerpente AND p.Data = dataPasto) = 0 THEN
            INSERT INTO Pasti
                    VALUES (dataPasto, idSerpente, risultato, tipo, nomeCibo);
    ELSE
            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Questo serpente ha già consumato un pasto oggi';
    END IF;

    DELETE FROM Pasti WHERE ((DATEDIFF(NOW(), Data)) / 365) >= 1;
END $$ DELIMITER;
```

Il resto delle stored procedures implementare per l'inserimento dei dati sono banali. Se ne riporta solo una per rappresentarle tutte:

```
DELIMITER $$
CREATE PROCEDURE AggiungiTerrario(
riscaldatore enum('Tappetino','Lampada','Entrambe'),
altaTemp tinyint,
bassaTemp tinyint,
tipologia enum('Opaco','Trasparente','Entrambi'),
acquatico tinyint(1),
arrampicabile tinyint(1),
scavabile tinyint(1),
umidita tinyint)
BEGIN
    DECLARE EXIT HANDLER FOR SQLSTATE '45001'
            SHOW ERRORS;
    INSERT INTO Terrario
            VALUES (NULL, riscaldatore, altaTemp, bassaTemp, tipologia, acquatico, arrampicabile, scavabile,
    umidità);
END $$ DELIMITER;
```

Segue una lista di stored procedure più complesse per operare sui dati del database.

Questa stored procedure viene utilizzata per mettere in vendita un serpente effettuando prima dei controlli sulla legittimità dell'operazione.

```
01 |DELIMITER $$
02 |CREATE PROCEDURE MettiInVenditaSerpente(
03 |IN idSerpente INT UNSIGNED,
04 |IN nuovoStato ENUM('In vendita','Venduto','In adozione','Adottato','Non in vendita','Morto','Riproduttore'),
05 |IN cites VARCHAR(150),
06 |IN prezzo SMALLINT UNSIGNED,
07 |IN foto VARCHAR(150))
08 |BEGIN
09 |    DECLARE EXIT HANDLER FOR SQLSTATE '45001'
10 |            SHOW ERRORS;
11 |    IF (SELECT COUNT(ID) FROM Serpente WHERE ID = idSerpente) = 0 THEN
12 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Serpente selezionato non esistente.';
13 |    END IF;
14 |    IF nuovoStato IN ('Venduto','Adottato','Non in vendita','Morto','Riproduttore') THEN
15 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato scelto uno stato non valido.';
16 |    ELSEIF (SELECT COUNT(*) FROM Pasti p WHERE p.IDSerpente = idSerpente AND Risultato = 'Consumato') < 2
    THEN
17 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato scelto un serpente che non ha consumato
    abbastanza pasti.';
18 |    ELSEIF idSerpente IS NULL OR nuovoStato IS NULL OR prezzo IS NULL THEN
19 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato omesso almeno uno dei parametri. Tutti i
    parametri sono obbligatori';
20 |    END IF;
21 |    UPDATE serpente SET Stato = nuovoStato,
22 |                    Cites = cites,
23 |                    Prezzo = prezzo,
24 |                    Foto = foto
25 |            WHERE ID = idSerpente;
26 |END $$ DELIMITER;
```

Questa stored procedure viene usata per finalizzare la vendita di un animale. Oltre a un controllo sulla correttezza dell'operazione, elimina anche i dati che a questo punto non saranno più di interesse per l'associazione.

```
01 |DELIMITER $$
02 |CREATE PROCEDURE VenditaSerpente(
03 |IN idSerpente INT UNSIGNED,
04 |IN nuovoStato ENUM('In vendita','Venduto','In adozione','Adottato','Non in vendita','Morto','Riproduttore'),
05 |IN cfCliente CHAR(16),
06 |IN dataAcquisto DATE)
07 |BEGIN
08 |    DECLARE EXIT HANDLER FOR SQLSTATE '45001'
09 |            SHOW ERRORS;
10 |    DECLARE CONTINUE HANDLER FOR 1175
11 |
12 |    IF (SELECT COUNT(ID) FROM Serpente WHERE ID = idSerpente) = 0 THEN
13 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Serpente selezionato non esistente.';
14 |    END IF;
15 |
16 |    IF nuovoStato IN ('In Vendita','In adozione','Non in vendita','Morto','Riproduttore') THEN
17 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato scelto uno stato non valido.';
18 |    ELSEIF idSerpente IS NULL OR nuovoStato IS NULL OR cfCliente IS NULL OR dataAcquisto IS NULL THEN
19 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato omesso almeno uno dei parametri. Tutti i
    parametri sono obbligatori';
20 |    END IF;
21 |
22 |    UPDATE serpente
23 |    SET Stato = nuovoStato,
24 |            CFCliente = cfCliente,
25 |            DataAcquisto = dataAcquisto
26 |    WHERE ID = idSerpente;
27 |
28 |    DELETE FROM Pasti WHERE IDSerpente = idSerpente;
29 |    DELETE FROM BrumazioneIndotta WHERE IDSerpente = idSerpente;
30 |    DELETE FROM Progenie WHERE IDSerpente = idSerpente;
31 |    DELETE FROM Incubazione WHERE IDSerpente = idSerpente;
32 |END $$ DELIMITER ;
```

Similmente alla stored procedure precedente, questa SP permette di registrare la morte di uno dei serpenti dell'allevamento, occupandosi anche di registrare un'ultima visita.

```
01 |DELIMITER $$
02 |CREATE PROCEDURE RegistraMorte(
03 |IN idSerpente INT UNSIGNED,
04 |IN dimensione SMALLINT UNSIGNED,
05 |IN peso TINYINT UNSIGNED,
06 |IN causaMorte VARCHAR(4000),
07 |IN dataMorte DATE,
08 |IN ivaClinica BIGINT UNSIGNED)
09 |BEGIN
10 |    DECLARE EXIT HANDLER FOR SQLSTATE '45001'
11 |            SHOW ERRORS;
12 |
13 |    DECLARE CONTINUE HANDLER FOR 1175
14 |
15 |    IF (SELECT COUNT(ID) FROM Serpente WHERE ID = idSerpente) = 0 THEN
16 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Serpente selezionato non esistente.';
17 |    END IF;
18 |
19 |    UPDATE serpente
20 |    SET Stato = 'Morto'
21 |    WHERE ID = idSerpente;
22 |
23 |    IF dataMorte IS NULL THEN
24 |            SET dataMorte = NOW();
25 |    END IF;
26 |
27 |    INSERT INTO Visita
28 |            VALUES (NULL, dimensione, peso, causaMorte, NULL, 'Morte', dataMorte, idSerpente, ivaClinica);
29 |
30 |    DELETE FROM Pasti WHERE IDSerpente = idSerpente;
31 |    DELETE FROM BrumazioneIndotta WHERE IDSerpente = idSerpente;
32 |    DELETE FROM Progenie WHERE IDSerpente = idSerpente;
33 |    DELETE FROM Incubazione WHERE IDSerpente = idSerpente;
34 |END $$ DELIMITER;
```

Questa stored procedure si occupa invece di identificare quali serpenti venduti hanno ormai superato l'aspettativa di vita della loro specie e ne modifica lo stato.

```
01 |DELIMITER $$
02 |CREATE PROCEDURE MorteAutomaticaSerpentiVenduti()
03 |BEGIN
04 |    DECLARE fine INTEGER DEFAULT 0;
05 |    DECLARE idSerpenti INT UNSIGNED;
06 |    DECLARE aspettativaSerpente TINYINT UNSIGNED;
07 |    DECLARE dataNascitaSerpente DATE;
08 |    DECLARE listaSerpenti CURSOR FOR SELECT ID FROM Serpente s WHERE s.CFCliente IS NOT NULL AND Stato <>
    'Morto';
09 |
10 |    DECLARE CONTINUE HANDLER FOR NOT FOUND
11 |            SET fine = 1;
12 |    DECLARE EXIT HANDLER FOR SQLSTATE '45001'
13 |            SHOW ERRORS;
14 |
15 |    OPEN listaSerpenti;
16 |
17 |    WHILE (fine = 0) DO
18 |            FETCH listaSerpenti INTO idSerpenti;
19 |        IF fine = 0 THEN
20 |                    SELECT sp.Aspettativa, se.DataNascita
21 |            INTO aspettativaSerpente, dataNascitaSerpente
22 |            FROM Specie sp INNER JOIN Serpente se ON sp.NomeScientifico = se.NomeSpecie
23 |            WHERE se.ID = idSerpenti AND Stato <> 'Morto';
24 |            IF (DATEDIFF(NOW(), dataNascitaSerpente)) / 365 >= aspettativaSerpente THEN
25 |                            UPDATE Serpente SET Stato = 'Morto' WHERE ID = idSerpenti;
26 |                    END IF;
27 |            END IF;
28 |    END WHILE;
29 |
30 |    CLOSE listaSerpenti;
31 |END $$ DELIMITER;
```

Questa procedure si occupa di eliminare dal database i serpenti morti e che non hanno figli, ossia che non sono più di interesse all'associazione.

```
01 |DELIMITER $$
02 |CREATE PROCEDURE CancellazioneAutomaticaSerpenti()
03 |BEGIN
04 |    DECLARE fine INTEGER DEFAULT 0;
05 |    DECLARE idSerpente INT UNSIGNED;
06 |    DECLARE totaleLegami TINYINT;
07 |    DECLARE listaSerpenti CURSOR FOR SELECT ID FROM Serpente WHERE Stato = 'Morto';
08 |
09 |    DECLARE EXIT HANDLER FOR SQLSTATE '45001'
10 |            SHOW ERRORS;
11 |    DECLARE CONTINUE HANDLER FOR NOT FOUND
12 |            SET fine = 1;
13 |
14 |    OPEN listaSerpenti;
15 |
16 |    WHILE (fine = 0) DO
17 |            FETCH listaSerpenti INTO idSerpente;
18 |        IF fine = 0 THEN
19 |                    SET totaleLegami = (SELECT COUNT(*) FROM Parentela p WHERE p.IDGenitore = idSerpente);
20 |            IF totaleLegami = 0 THEN
21 |                            DELETE FROM Serpente WHERE ID = idSerpente;
22 |                    END IF;
23 |            END IF;
24 |    END WHILE;
25 |    CLOSE listaSerpenti;
26 |END $$ DELIMITER;
```

Questa SP cancella i clienti i quali non hanno più serpenti in vita. All'inizio dell'esecuzione utilizza una delle stored procedure riportate in precedenza per aggiornare gli stati dei serpenti. A fine esecuzione, se il parametro specificato in input è TRUE, viene chiamata anche la SP precedente.

```
01 |DELIMITER $$
02 |CREATE PROCEDURE CancellazioneAutomaticaClienti(
03 |IN cancellaSerpenti BOOL)
04 |BEGIN
05 |    DECLARE fine INTEGER DEFAULT 0;
06 |    DECLARE cfClienti VARCHAR(16);
07 |    DECLARE totaleVivi TINYINT;
08 |    DECLARE listaClienti CURSOR FOR SELECT CF FROM Cliente;
09 |
10 |    DECLARE EXIT HANDLER FOR SQLSTATE '45001'
11 |            SHOW ERRORS;
12 |    DECLARE CONTINUE HANDLER FOR NOT FOUND
13 |            SET fine = 1;
14 |
15 |    CALL MorteAutomaticaSerpentiVenduti();
16 |
17 |    OPEN listaClienti;
18 |
19 |    WHILE (fine = 0) DO
20 |            FETCH listaClienti INTO cfClienti;
21 |        IF fine = 0 THEN
22 |                    SET totaleVivi = (SELECT COUNT(*) FROM serpente s WHERE s.CFCliente = cfClienti AND Stato
    <> 'Morto');
23 |            IF totaleVivi = 0 THEN
24 |                            DELETE FROM Cliente WHERE CF = cfClienti;
25 |                    END IF;
26 |            END IF;
27 |    END WHILE;
28 |
29 |    CLOSE listaClienti;
30 |
31 |    IF cancellaSerpenti THEN
32 |            CALL CancellazioneAutomaticaSerpenti();
33 |    END IF;
34 |END $$ DELIMITER ;
```

Questa stored procedure ritorna in due campi VARCHAR l'elenco dei morph e dei possible het posseduti dal serpente specificato

```
01 |DELIMITER $$
02 |CREATE PROCEDURE OttenimentoMorphEPH(
03 |IN idSerpenteRicerca INT UNSIGNED,
04 |INOUT morphSerpente VARCHAR(1500),
05 |INOUT phSerpente VARCHAR(1600))
06 |BEGIN
07 |    DECLARE fine INTEGER DEFAULT 0;
08 |    DECLARE temporaryId INT UNSIGNED;
09 |    DECLARE temporaryName VARCHAR(100) DEFAULT '';
10 |    DECLARE temporaryPercent TINYINT;
11 |    DECLARE listaMorph CURSOR FOR SELECT IDMorph FROM Aspetto WHERE IDSerpente = idSerpenteRicerca;
12 |    DECLARE listaPH CURSOR FOR SELECT IDMorph, Percentuale FROM PH WHERE IDSerpente = idSerpenteRicerca;
13 |
14 |    DECLARE EXIT HANDLER FOR SQLSTATE '45001'
15 |            SHOW ERRORS;
16 |    DECLARE CONTINUE HANDLER FOR NOT FOUND
17 |            SET fine = 1;
18 |
19 |    OPEN listaMorph;
20 |    OPEN listaPH;
21 |
22 |    WHILE (fine = 0) DO
23 |            FETCH listaMorph INTO temporaryID;
24 |        IF fine = 0 THEN
25 |                    SELECT Nome INTO temporaryName FROM Morph WHERE ID = temporaryID;
26 |            SET morphSerpente = CONCAT(temporaryName, ', ', morphSerpente);
27 |            END IF;
28 |    END WHILE;
29 |
30 |    SET fine = 0;
31 |    SET temporaryName = '';
32 |    CLOSE listaMorph;
33 |
34 |    WHILE (fine = 0) DO
35 |            FETCH listaPH INTO temporaryID, temporaryPercent;
36 |        IF fine = 0 THEN
37 |                    SELECT Nome INTO temporaryName FROM Morph WHERE ID = temporaryID;
38 |            SET phSerpente = CONCAT(temporaryPercent, '% ', temporaryName, ', ', phSerpente);
39 |            END IF;
40 |    END WHILE;
41 |
42 |    CLOSE listaPH;
43 |END $$ DELIMITER ;
```

Triggers

Sono stati sviluppati dei trigger sull'aggiornamento e l'inserimento dei dati nelle tabelle. I trigger riportati si occupano principalmente di assicurarsi che tutti i record presenti nelle tabelle siano sensati. La maggior parte sono banali, come ad esempio i seguenti.

```
01 |DELIMITER $$
02 |CREATE TRIGGER TR_AggiuntaTerrario
03 |BEFORE UPDATE ON Terrario
04 |FOR EACH ROW
05 |BEGIN
06 |    IF NEW.AltaTemp - NEW.BassaTemp < 0 THEN
07 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'La bassa temperatura è maggiore dell''alta
    temperatura.';
08 |    END IF;
09 |END $$ DELIMITER;
10 |
11 |DELIMITER $$
12 |CREATE TRIGGER TR_ModificaTerrario
13 |BEFORE UPDATE ON Terrario
14 |FOR EACH ROW
15 |BEGIN
16 |    IF NEW.AltaTemp - NEW.BassaTemp < 0 THEN
17 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'La bassa temperatura è maggiore dell''alta
    temperatura.';
18 |    END IF;
19 |END $$ DELIMITER;
```

I trigger successivi sono invece più complessi poiché riguardano la relazione Serpente dove vertono la maggior parte dei vincoli di progetto.

```
01 |DELIMITER $$
02 |CREATE TRIGGER TR_AggiuntaSerpente
03 |BEFORE INSERT ON Serpente
04 |FOR EACH ROW
05 |BEGIN
06 |    IF NEW.Nome IN (SELECT Nome FROM Serpente WHERE IVAAllevamento = NEW.IVAAllevamento AND Stato NOT IN
    ('Venduto', 'Adottato', 'Morto')) THEN
07 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Esiste già un serpente con questo nome in questo
    allevamento.';
08 |
09 |    ELSEIF NEW.Sesso NOT IN ('M', 'F') THEN
10 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'L''input inserito per il sesso del serpente non è né
    M né F.';
11 |
12 |    ELSEIF NEW.Stato IN ('In vendita','Venduto','In adozione','Adottato') THEN
13 |            IF NEW.Prezzo IS NULL THEN
14 |                    SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Non è stato indicato il prezzo per il
    serpente.';
15 |            ELSEIF NEW.Cites IS NULL THEN
16 |                    SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'URL della certificazione Cites mancante.';
17 |            ELSEIF NEW.Foto IS NULL THEN
18 |                    SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'URL della foto mancante.';
19 |
20 |            ELSEIF NEW.Stato IN ('Venduto', 'Adottato') THEN
21 |                    IF NEW.CFCliente IS NULL THEN
22 |                            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Codice fiscale acquirente mancante';
23 |                    ELSEIF NEW.DataAcquisto IS NULL THEN
24 |                            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Non è stata specificata la data di
    acquisto o adozione.';
25 |                    ELSEIF NEW.DataNascita IS NOT NULL AND NOT DateCheck(NEW.DataNascita, NEW.DataAcquisto)
    THEN
26 |                            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'L''intervallo di date inserito non è
    valido.';
27 |                    END IF;
28 |            END IF;
29 |
30 |    ELSEIF NEW.Stato <> 'Morto' THEN
31 |            IF NEW.Prezzo IS NOT NULL THEN
32 |                    SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato inserito il prezzo per un animale
    non in vendita o adozione.';
33 |            ELSEIF NEW.CFCliente IS NOT NULL THEN
34 |                    SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato inserito l''acquirente per un
    animale non in vendita o adozione.';
35 |            ELSEIF NEW.DataAcquisto IS NOT NULL THEN
36 |                    SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stata inserita la data di acquisto per un
    animale non in vendita o adozione.';
37 |            END IF;
38 |    END IF;
39 |END $$ DELIMITER ;
```

Il trigger seguente è ancora più complesso poiché necessita di verificare l'integrità delle modifiche apportate dalle stored procedure.

```
01 |DELIMITER $$
02 |CREATE TRIGGER TR_PreModificaSerpente
03 |BEFORE UPDATE ON Serpente
04 |FOR EACH ROW
05 |BEGIN
06 |    IF NEW.Nome IN (SELECT Nome FROM Serpente WHERE IVAAllevamento = NEW.IVAAllevamento AND Nome <> OLD.Nome
    AND Stato NOT IN ('Venduto', 'Adottato', 'Morto')) THEN
07 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Esiste già un serpente con questo nome in questo
    allevamento.';
08 |
09 |    ELSEIF NEW.Sesso NOT IN ('M', 'F') THEN
10 |            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'L''input inserito per il sesso del serpente non è né
    M né F.';
11 |    END IF;
12 |
13 |    IF NEW.Stato IN ('In vendita','Venduto','In adozione','Adottato') THEN
14 |            IF NEW.Cites IS NULL THEN
15 |                    IF OLD.Cites IS NOT NULL THEN
16 |                            SET NEW.Cites = OLD.Cites;
17 |                    ELSE
18 |                            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'URL della certificazione Cites
    mancante.';
19 |                    END IF;
20 |            END IF;
21 |
22 |            IF NEW.Foto IS NULL THEN
23 |                    IF OLD.Foto IS NOT NULL THEN
24 |                            SET NEW.Foto = OLD.Foto;
25 |                    ELSE
26 |                            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'URL della foto mancante.';
27 |                    END IF;
28 |            END IF;
29 |
30 |            IF NEW.Prezzo IS NULL THEN
31 |                    IF OLD.Prezzo IS NOT NULL THEN
32 |                            SET NEW.Prezzo = OLD.Prezzo;
33 |                    ELSE
34 |                            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Non è stato indicato il prezzo per il
    serpente.';
35 |                    END IF;
36 |            END IF;
37 |
38 |            IF NEW.Stato IN ('Venduto', 'Adottato') THEN
39 |                    IF OLD.Stato NOT IN ('In vendita', 'In adozione') THEN
40 |                            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato venduto o dato in adozione
    un serpente non in vendita o in adozione.';
41 |                    ELSEIF OLD.Stato = 'In vendita' AND NEW.Stato = 'Adottato' THEN
42 |                            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato adottato un serpente
    previsto per la vendita';
43 |                    ELSEIF OLD.Stato = 'In adozione' AND NEW.Stato = 'Venduto' THEN
44 |                            SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato venduto un serpente
    previsto per l''adottazione';
```

```
45 |                     ELSEIF NEW.CFCliente IS NULL THEN
46 |                             SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Codice fiscale acquirente mancante';
47 |                     ELSEIF NEW.DataAcquisto IS NULL THEN
48 |                             SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Non è stata specificata la data di
    acquisto o adozione.';
49 |                     ELSEIF NEW.DataNascita IS NOT NULL AND NOT DateCheck(NEW.DataNascita, NEW.DataAcquisto)
    THEN
50 |                             SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'L''intervallo di date inserito non è
    valido.';
51 |                     END IF;
52 |             END IF;
53 |
54 |     ELSEIF NEW.Stato <> 'Morto' THEN
55 |             IF NEW.Prezzo IS NOT NULL THEN
56 |                     SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato inserito il prezzo per un animale
    non in vendita o adozione.';
57 |             ELSEIF NEW.CFCliente IS NOT NULL THEN
58 |                     SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stato inserito l''acquirente per un
    animale non in vendita o adozione.';
59 |             ELSEIF NEW.DataAcquisto IS NOT NULL THEN
60 |                     SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'E'' stata inserita la data di acquisto per un
    animale non in vendita o adozione.';
61 |             END IF;
62 |     END IF;
63 |END $$ DELIMITER;
```

Visualizzazione dei dati del database

Sono state create delle semplicissime stored procedure di esempio per la visualizzazione di alcuni dati del database.

```
01 |DELIMITER $$
02 |CREATE PROCEDURE SelezionaSerpentePerStato(
03 |IN stato ENUM('In vendita','Venduto','In adozione','Adottato','Non in vendita','Morto','Riproduttore'))
04 |BEGIN
05 |     SELECT * FROM serpente s WHERE s.Stato = stato;
06 |END $$ DELIMITER;
07 |
08 |DELIMITER $$
09 |CREATE PROCEDURE VisualizzaSpecieSerpente(
10 |IN id INT UNSIGNED)
11 |BEGIN
12 |     IF (SELECT COUNT(ID) FROM Serpente WHERE ID = idSerpente) = 0 THEN
13 |             SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Serpente selezionato non esistente.';
14 |     END IF;
15 |
16 |     SELECT *
17 |     FROM serpente se INNER JOIN specie sp ON se.NomeSpecie = sp.NomeScientifico
18 |     WHERE se.ID = id;
19 |END $$ DELIMITER;
```

Infine, ecco una vista che idealmente rappresenta i dati che i clienti visualizzeranno in un ipotetico store online.

```
01 |CREATE VIEW StoreSerpenti AS(
02 |    SELECT se.ID, se.Nome, se.Sesso, se.DataNascita, se.Stato, se.Cites, se.Foto, se.Prezzo, a.Nome AS 'Nome
    Allevamento', sp.NomeComune AS 'Specie'
03 |    FROM Serpente se INNER JOIN Allevamento a ON se.IVAAllevamento = a.IVA
04 |            INNER JOIN Specie sp ON se.NomeSpecie = sp.NomeScientifico
05 |    WHERE Stato IN ('In vendita', 'In adozione')
06 |);
```