Modeling the Hierarchical Nature of Data

An Entity Relationship Approach

Introductions

- Richard Broersma
  - LAPUG Leader & PostgreSQL Enthusiast
  - Systems Integrator / Industrial Applications Developer
    Mangan Incorporated
- Mangan is a nationwide engineering and automation firm that serves multiple industries:
  - Petrochemical, Oil & Nat. Gas Pipelines, Oil & Nat. Gas Production
  - Chemical Production
  - Bio-Pharmaceutical Production
  - Solar Energy Production

Preview

1) Controversy
   - Hierarchical data in Relational Databases
2) Perceptions
   - Reality and Data modeling
3) Conceptual Designs
   - Entity Relationship Diagrams (ERD)
   - Modeling the Concept of Hierarchal Data
4) Physical Designs
   - Implementations of conceptual ERD Designs

Controversy

In RDBMS, R is for
Relational!

What's all this Hierarchal
Nonsense?

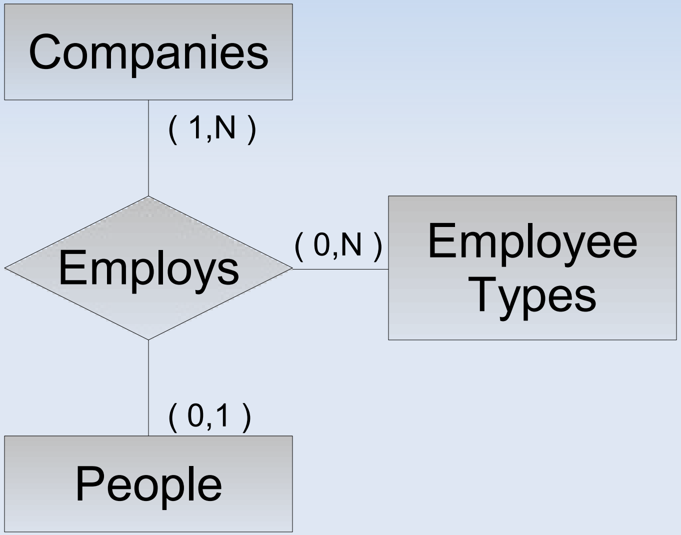
Controversy

- Network and Hierarchical database are ”things of the past.”
- Relational databases should be implemented using entities and relationships described in relational theory.
- Should Hierarchical modeling be avoided?

Perceptions

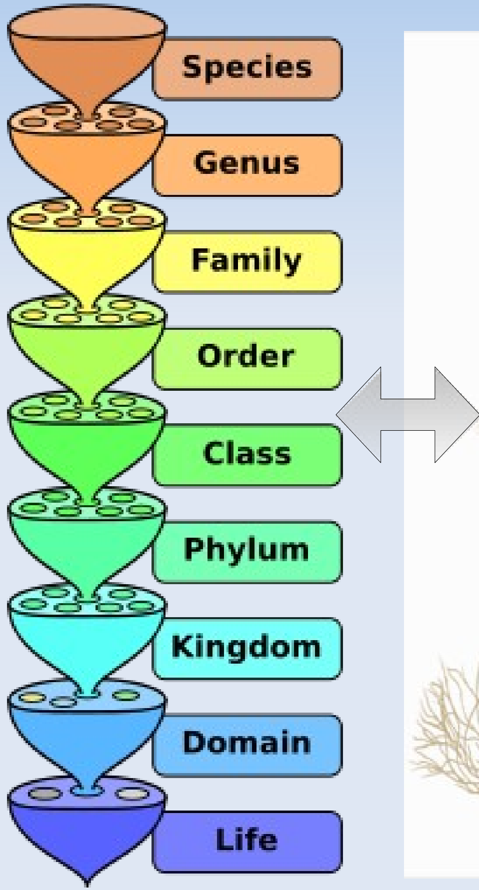
Basics of ERD Modeling

- Thinking in terms of data modeling
  - Entities
  - Relationships
  - Entity types
- Implementation using 3 Normal forms

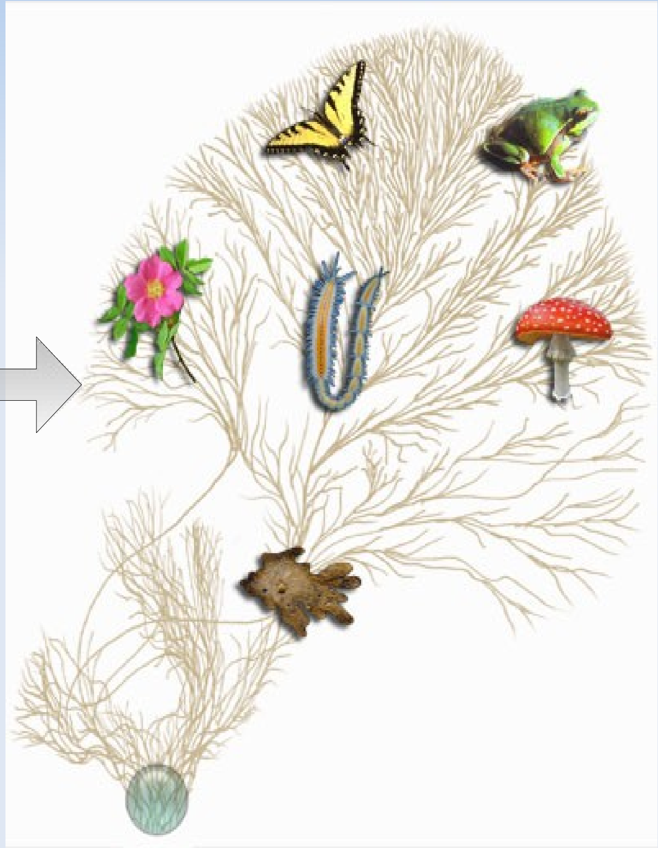


Perceptions of Reality

- Our Perceptions
  - Entities & Relationships
  - **Classifications**
- Taxonomy
  - How do the attributes of similar type of entities differ



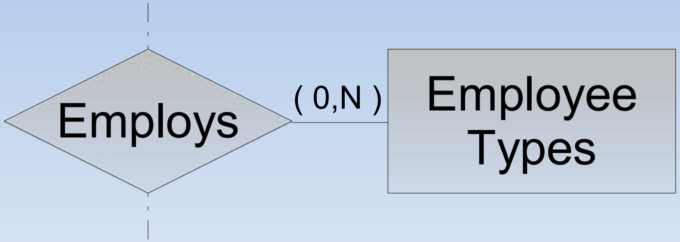
Layers of Hierarchical Abstraction



Physical Hierarchical Representation

Short-fall of Entity Types

- Employee Types table
  - Can't express attribute similarities or differences of similar types.
  - Can't define relationships between people of related types



Employee Types	
Electrician	Does Electrical Work
Engineer	Does Engineering
Manager	Manages others
President	Presides over a company
Welder	Welds

Employs		
ABC	TED	Welder
ABC	SANDY	President
ACME	RON	Electrician
ACME	JILL	Engineer
BP	DAVE	Manager

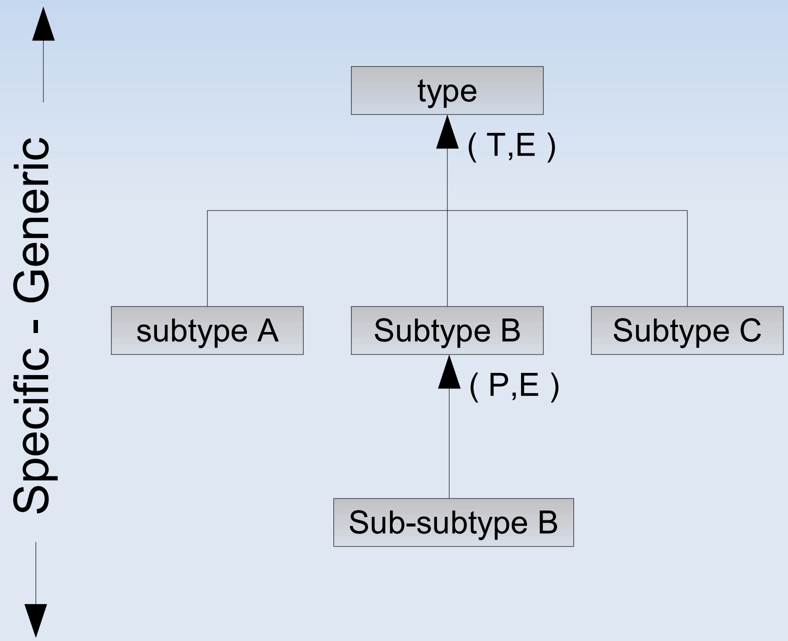
Conclusion

- If we need to know about the:
  - Extended Attributes of Entity Types
  - Relationships between Entity Types
- Then hierarchical data modeling must be implemented

Conceptual Designs

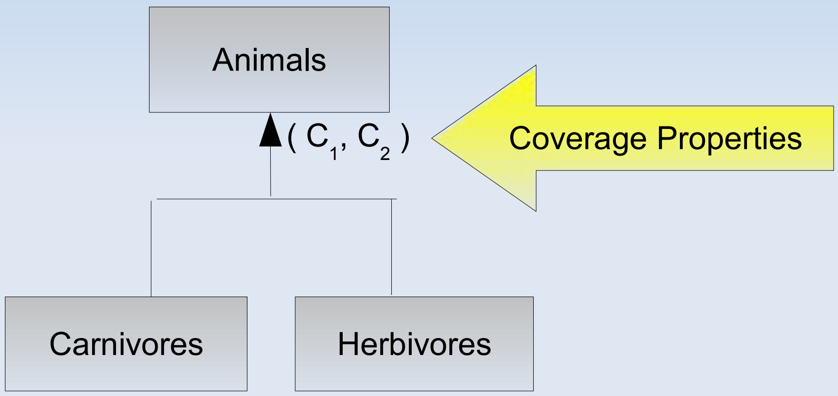
- ERD Provides Generalization Hierarchy Model
  - Defines hierarchical constraints for hierarchical mapping.
  - Grouping of similar entity types.
  - Similarities and differences are defined.
  - Relationships can be created between entities of any (sub)type.

Generalization Hierarchy



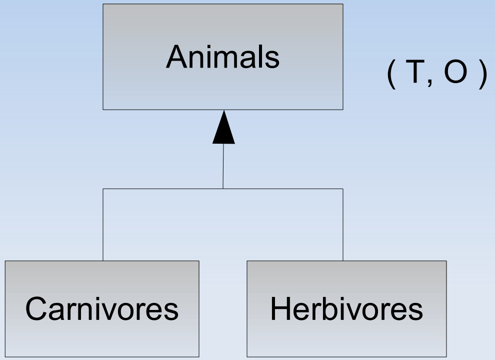
ERD - Hierarchical Constraints

- C_1 Property
  - {T} Total Coverage
  - {P} Partial Coverage
- C_2 Property
  - {E} Exclusive Coverage
  - {O} Overlapping Coverage



ERD - Entity Type Groupings

- Entity types having equal attributes are grouped together.
- Similarities and differences are defined.



Animals		
id	**animalcode**	**weight**
Bear-1	bear	500
Sheep-2	sheep	100
Wolf-3	wolf	120
Deer-4	deer	240
Puma-5	puma	200

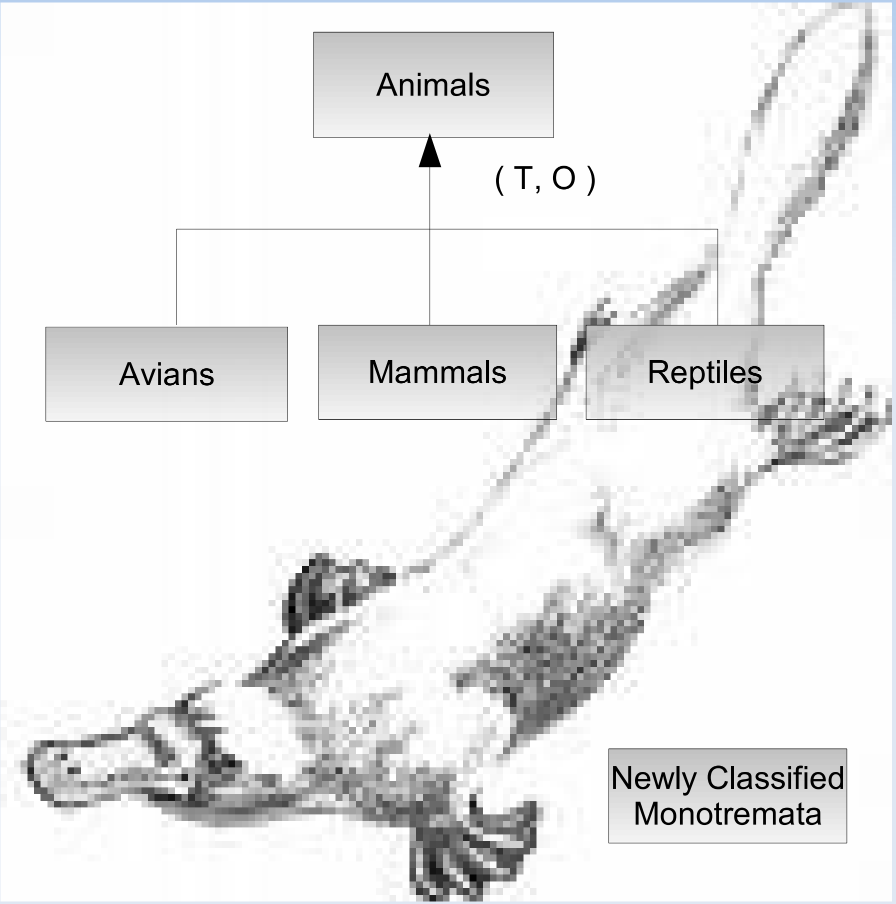
Carnivores			
id	**animalcode**	**weight**	**favoritePrey**
Bear-1	bear	500	salmon
Wolf-3	wolf	120	sheep
Puma-5	puma	200	deer

Herbivores			
id	**animalcode**	**weight**	**favoriteVegi**
Bear-1	bear	500	berries
Sheep-2	sheep	100	grass
Deer-5	deer	240	grass

Omnivores (implied by Overlapping)				
id	**animalcode**	**weight**	**favoritePrey**	**favoriteVegi**
Bear-1	bear	500	salmon	berries

ERD - Entity type Groupings

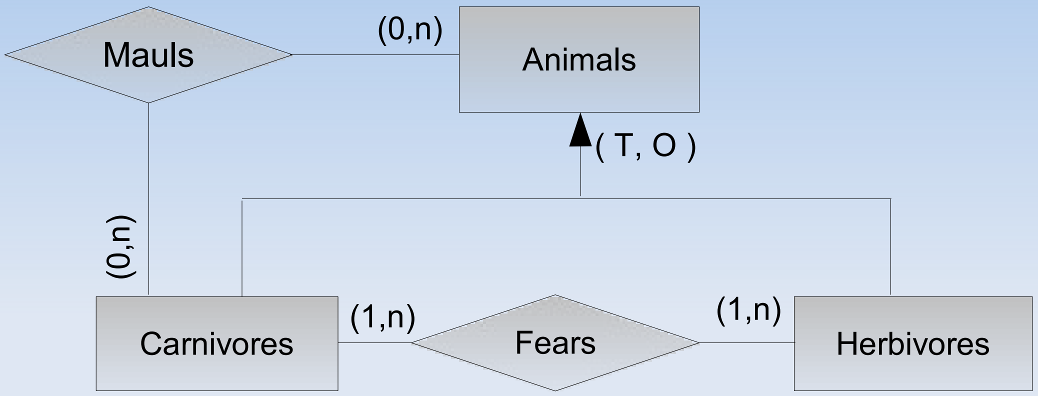
- Beware of the “platypus”!
  - Valid criticisms of G/H exist.
  - Some entities will map to most of the hierarchical attributes but not all.
  - Careful consideration required to minimize platypus affect.*



*If practical, G/H redesign can eliminate most “Platypuses”.

ERD – Entity type Relationships

- Complex Relationships are possible between sub types



Fears	
carnivoreid	**herbivoreid**
Deer-4	Wolf-3
Deer-4	Puma-5
Deer-4	Bear-1
Sheep-2	Wolf-3
Sheep-2	Puma-5
Bear-1	Bear-6

Maulings		
carnivoreid	**animalid**	**Mauling-date**
Bear-1	Wolf-3	01/15/08
Bear-1	Deer-4	07/12/07
Wolf-3	Sheep-2	09/22/07

Physical Designs

- There are 5 physical designs (that I know of) for implementing conceptual Generalization Hierarchies
- Each physical design varies in the features that the conceptual Generalization Hierarchy Model defines

Physical Designs

- Entity-Attribute-Value Table (EAV)
  ~Relational purists favorite
- Nullable Attributes Table (NA)
  ~Happens overtime
- Vertical Disjunctive Table Partitioning (VDP)
  ~My favorite
- Horizontal Disjunctive Table Partitioning (HDP)
  ~PostgreSQL Table inheritance
- (NA–EAV) Hybrid Table
  ~Worst Design Ever – know it to avoid it

Implementation Features List

Model VS. Features	EAV	NA	VDP	HDP	EAV-NA
Total Coverage	A	E	A	A	A
Partial Coverage	A	E	E	A	A
Exclusive Coverage	A	E	E	E	A
Overlapping Coverage	A	E	A	E	A
Supports Grouping			S	S	
Supports Relations			S	S	

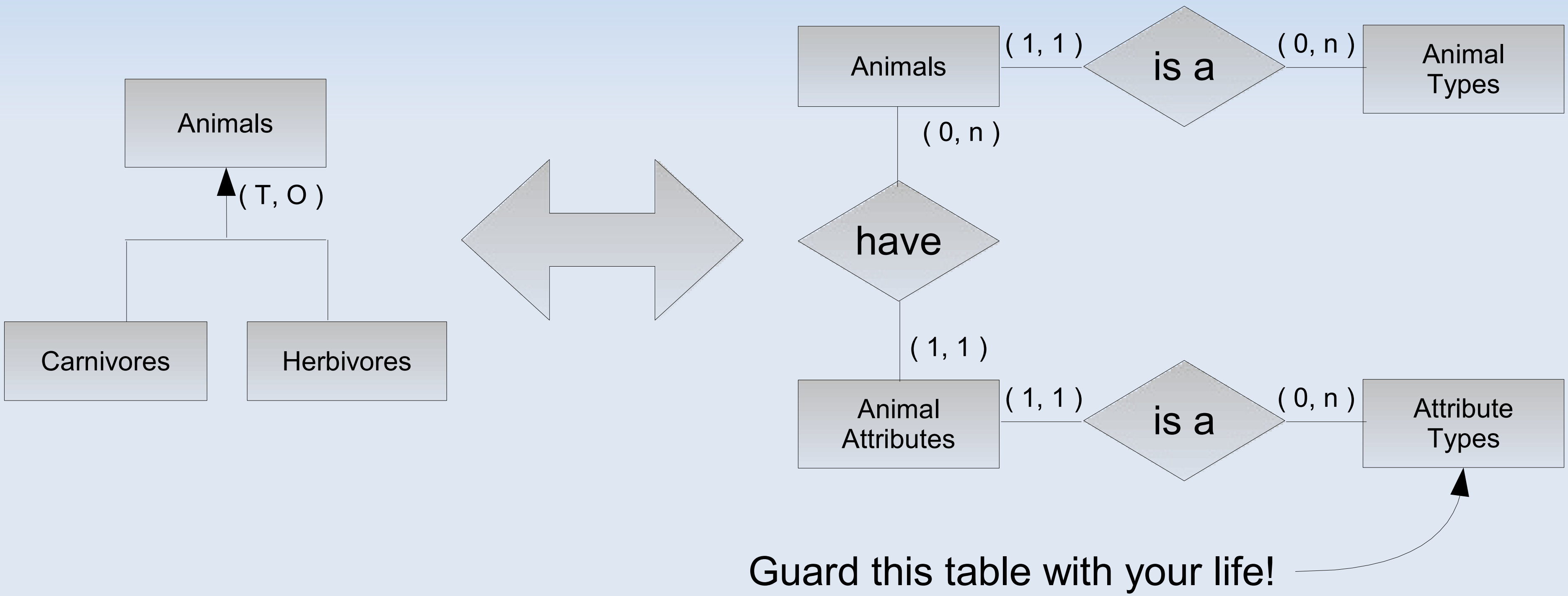
Legend
E = Enforced (DDL or DRI)
A = Allows (Client Enforced)
S = Supports

Good Design Guidelines

- Always include an entity type column associated with the primary key throughout the hierarchy
  - This is still the best way to identify the type of hierarchical entity or hierarchical relationship
  - CHECK Constraints can then be implemented based on the entity types (Hierarchical Foreign Keys can be used also.)
  - This will prevent data corruption at the RDBMS tier that could be caused by application bugs or lazy users.

Entity Attribute Value (EAV)

- Physical Implementation:



EAV Table - DDL

```
CREATE TABLE Animaltypes(
   animalcode       VARCHAR( 20 ) PRIMARY KEY,
   description      TEXT NOT NULL DEFAULT '' );

CREATE TABLE Animals (
   animal_id        VARCHAR( 20 ) PRIMARY KEY,
   animalcode       VARCHAR( 20 ) REFERENCES Animaltypes( animalcode )
                    ON UPDATE CASCADE,
   weight           NUMERIC( 7, 2) CHECK( weight > 0 ));


CREATE TABLE Attributetypes(
   attributecode   VARCHAR( 20 ) PRIMARY KEY,
   description     TEXT NOT NULL DEFAULT '' );

CREATE TABLE Animalattributes(
   animal_id       VARCHAR( 20 ) REFERENCES Animals( animal_id )
                   ON UPDATE CASCADE ON DELETE CASCADE,
   attribute       VARCHAR( 20 ) REFERENCES Attributetypes( attributecode )
                   ON UPDATE CASCADE,
   att_value       TEXT NOT NULL,

   PRIMARY KEY ( animal_id, attribute ));
```

EAV Table - Consideration

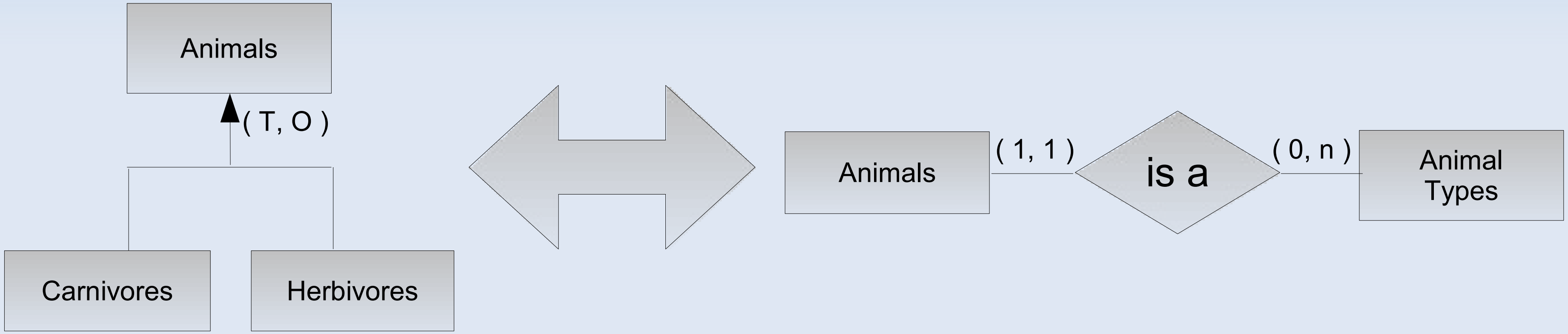
- Advantages:
- Provides a flexible mechanism to record the attributes associated with any entity.
- The flexibility eliminates the possibility of “platypuses”.
- This EAV design requires almost no consideration of the nature of the applicable hierarchical data and requires very little time to implement (cookie cutter).

EAV Table - Consideration

- Disadvantages:
- Users or Application logic becomes responsible to ensuring that all entities of a specific type will have the required associated attributes. (no DDL or DRI server constraints will work).
- The EAV table uses a TEXT (or VARCHAR) column for all attribute values regardless if Dates, Timestamps, Integers, Numerics or Booleans would be more appropriate.
- No Foreign Keys on Attribute Values: There isn't a way to prevent bad data-entry. For example nothing would prevent a user from entering 'I like peanut butter.' for the attribute value for Birth Date.

Null-able Attributes (NA) Table

- Physical Implementation:



(NA) Table - DDL

```
CREATE TABLE Animaltypes(
   animalcode       VARCHAR( 20 ) PRIMARY KEY,
   description      TEXT NOT NULL DEFAULT '' );

CREATE TABLE Animals (
   animal_id        VARCHAR( 20 ) PRIMARY KEY,
   animalcode       VARCHAR( 20 ) REFERENCES Animaltypes( animalcode )
                    ON UPDATE CASCADE,
   weight           NUMERIC( 7, 2) CHECK( weight > 0 ),

   favoriteprey     VARCHAR( 20 ) REFERENCES Animaltypes( animalcode )
                    ON UPDATE CASCADE
                    CHECK( CASE WHEN animalcode IN ('Bear', 'Wolf', 'Puma')
                                  THEN favoriteprey IS NOT NULL
                                WHEN animalcode IN ('Deer', 'Sheep' )
                                  THEN favoriteprey IS NULL END )

   favoritevegi     VARCHAR( 20 ) REFERENCES Vegitypes ( Vegicode )
                    ON UPDATE CASCADE
                    CHECK( CASE WHEN animalcode IN ('Bear', 'Deer', 'Sheep')
                                  THEN favoritevegi IS NOT NULL
                                WHEN animalcode IN ('Wolf', 'Pump' )
                                  THEN favoritevegi IS NULL END )
);
```

NA Table - Consideration

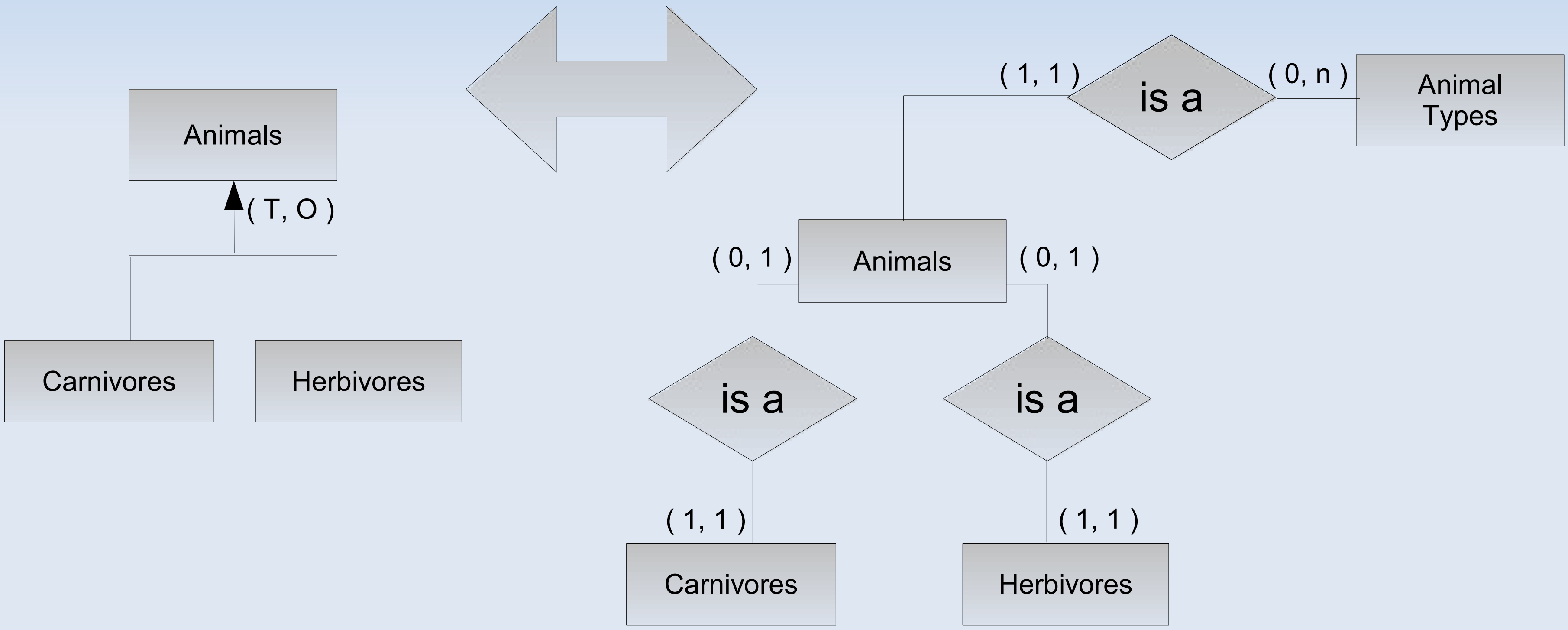
- Advantages:
- Provides a flexible mechanism to record the attributes associated with any entity.
- All attributes values can be constrained with foreign keys.
- Requires almost no consideration of the nature of the applicable hierarchical data. Hierarchical attributes are added via DDL as they are encounter during the life time of the application.

NA Table - Consideration

- Disadvantages:
  - Validating Hierarchical data integrity requires too many checks constraints. This can really hurt INSERT and UPDATE performance
  - Tuples in the table can get to be too big with many-many unused nulled columns.
  - The concept of null can get obscured. Does Null mean “Don't Know” or “Doesn't Apply”.

(VDP) Table

- Physical Implementation:



(VDP) Table - DDL

```
CREATE TABLE Animals (
   animal_id        VARCHAR( 20 ) UNIQUE NOT NULL,
   animalcode       VARCHAR( 20 ) REFERENCES Animaltypes( animalcode )
                    ON UPDATE CASCADE,
   weight           NUMERIC( 7, 2) CHECK( weight > 0 ),

   PRIMARY KEY ( animal_id, animalcode )
   --RI to handle denormalization of sub-tables );

CREATE TABLE Carnivores (
   animal_id        VARCHAR( 20 ) UNIQUE NOT NULL,

   animalcode       VARCHAR( 20 ) NOT NULL
                    CHECK( animalcode IN ( 'Bear', 'Wolf', 'Puma' )),

   favoriteprey     VARCHAR( 20 ) REFERENCES Animaltypes( animalcode )
                    ON UPDATE CASCADE,

   PRIMARY KEY ( animal_id, animalcode ),

   FOREIGN KEY ( animal_id, aminalcode )
    REFERENCES Animals( animal_id, animalcode )
     ON UPDATE CASCADE ON DELETE CASCADE,

   --RI to handle denormalization of animalcode );
```

(VDP) Table - DDL

```
CREATE TABLE Herbivores (
   animal_id        VARCHAR( 20 ) UNIQUE NOT NULL,

   animalcode       VARCHAR( 20 ) NOT NULL
                    CHECK( animalcode IN ( 'Deer', 'Sheep', 'Bear' )),

   favoriteprey     VARCHAR( 20 ) REFERENCES Animaltypes( animalcode )
                                  ON UPDATE CASCADE,

   PRIMARY KEY ( animal_id, animalcode ),

   FOREIGN KEY ( animal_id, aminalcode )
    REFERENCES Animals( animal_id, animalcode )
     ON UPDATE CASCADE ON DELETE CASCADE,

--RI to handle denormalization of animalcode );
```

VDP Table - Consideration

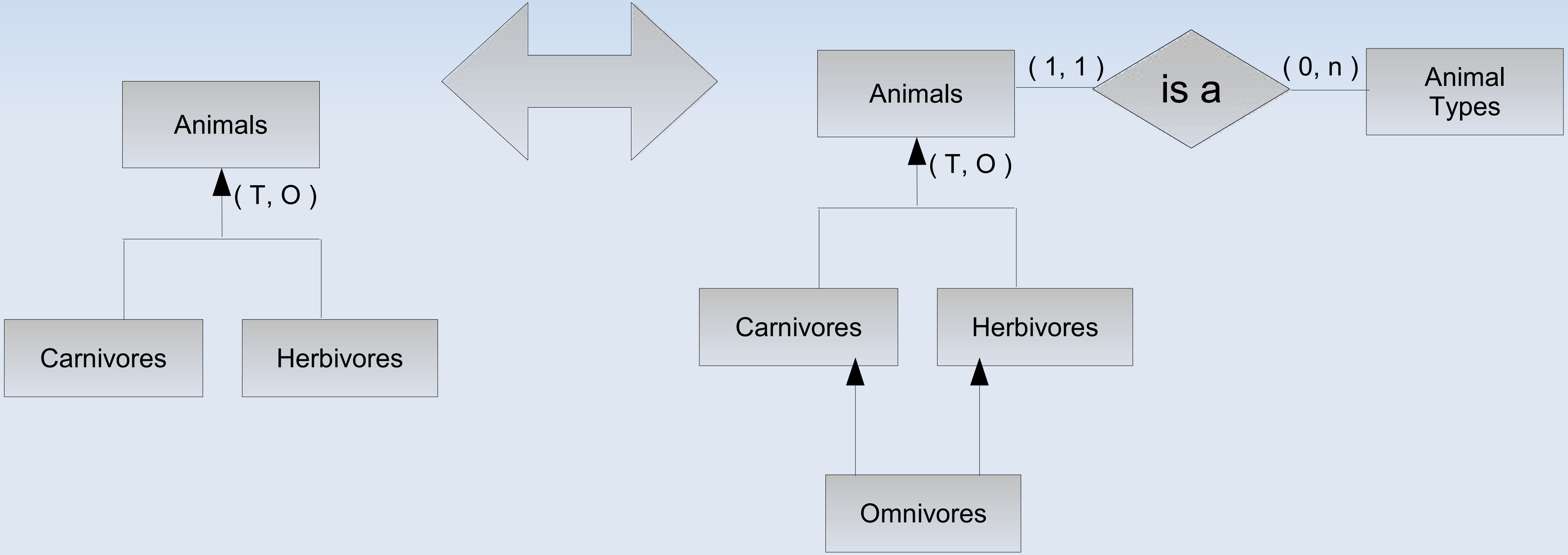
- Advantages:
  - All attributes values can be constrained with foreign keys.
  - Requires few Checks Constraints than the NA Table.

VDP Table - Consideration

- Disadvantages:
  - Checks only required for Entity type field, but too many check constraints can still hurt INSERT performance
  - Additional Application logic required to handle multiple INSERTs and UPDATEs to various (sub)type tables
  - Requires some denormalization to enforce data integrity. Referential Integrity handles this problem.
  - This design requires the designer to be well versed in the domain that is being modeled

(HDP) Table

- Physical Implementation:



(HDP) Table - DDL

```
CREATE TABLE Animals (
   animal_id        VARCHAR( 20 ) PRIMARY KEY,
   animalcode       VARCHAR( 20 ) REFERENCES Animaltypes( animalcode )
                    ON UPDATE CASCADE,
   weight           NUMERIC( 7, 2) CHECK( weight > 0 ));


CREATE TABLE Carnivores (
   favoriteprey     VARCHAR( 20 ))
INHERITS( Animals );


   ALTER TABLE Carnivores
ADD CONSTRAINT Cornivores_animalcode_check_iscarnivore
        CHECK( animalcode IN ( 'Bear', 'Wolf', 'Puma' ));
```

(HDP) Table - DDL

```
CREATE TABLE Herbivores (
   favoritevegi    VARCHAR( 20 ))
INHERITS( Animals );


   ALTER TABLE Herbivores
ADD CONSTRAINT Herbivores_animalcode_check_isHerbivore
        CHECK( animalcode IN ( 'Bear', 'Deer', 'Sheep' ));


CREATE TABLE Omnivores ()
INHERITS( Carnivores, Herbivores ); -- PostgreSQL also inherits Check Constraint
                                    -- The Overlapping checks will algebraically
                                    -- reduce to CHECK( animalcode = 'Bear' )

                             --  CarnivoreCodes ∩ HerbivoreCodes = OmnivoreCodes
```

HDP Table - Consideration

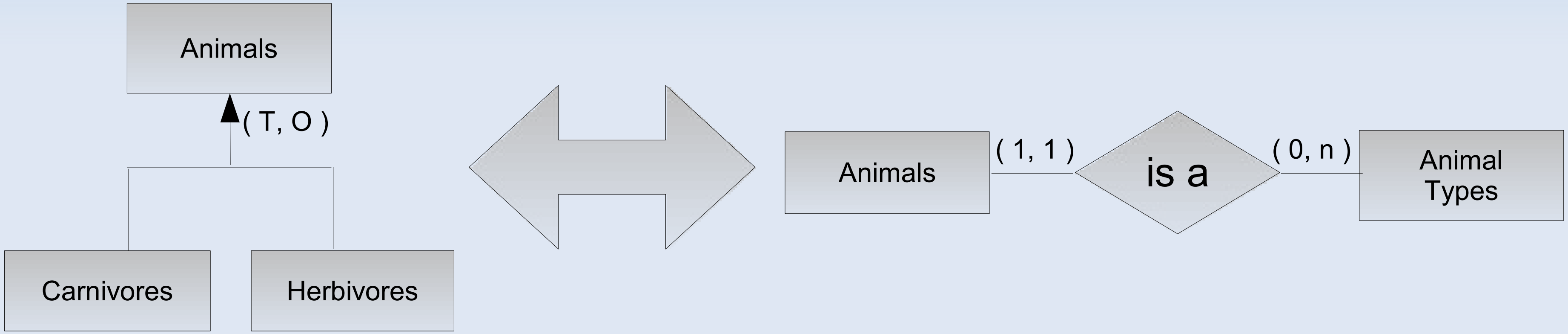
- Advantages:
  - All attributes values can be constrained with foreign keys. But you have to re-implement these Inherited foreign keys yourself.
  - Possible to allow for relationships only between hierarchical leaf entitles.
  - The application logic is simplified since all accesses to sub-entities are to a single table.

HDP Table - Consideration

- Disadvantages:
  - SLOW Sequential Scans are the only way to search the Root or Branch nodes of the hierarchy since scans on these tables are based on UNION ALL queries.
  - Uniqueness cannot be enforced across the hierarchy.
  - This design requires the designer to be well versed in the domain that is being modeled

(NA – EAV) Hybrid Table

- Physical Implementation:



(NA – EAV) Table - DDL

```
CREATE TABLE Animaltypes(
   animalcode       VARCHAR( 20 ) PRIMARY KEY,
   description      TEXT NOT NULL DEFAULT '' );

CREATE TABLE Animals (
   animal_id        VARCHAR( 20 ) PRIMARY KEY,
   animalcode       VARCHAR( 20 ) REFERENCES Animaltypes( animalcode )
                    ON UPDATE CASCADE,
   column1          VARCHAR( 255 ), --The application maps the attributes of each
   column2          VARCHAR( 255 ), --entity type to these intentionally vague
   column3          VARCHAR( 255 ), --columns.  Each entity type will have a unique
   column4          VARCHAR( 255 ), --mapping for column1 thru column100.
   column5          VARCHAR( 255 ),
   column6          VARCHAR( 255 ), --Unmapped columns not needed by an entity type
   column7          VARCHAR( 255 ), --may be treated as custom fields that the users
   column8          VARCHAR( 255 ), --may use any way they see fit.
 --   ...
   column100        VARCHAR( 255 )
);
```

NA – EAV Table - Consideration

- Advantages:
  - Provides a flexible mechanism to record the attributes associated with any entity.
  - The flexible mechanism eliminates the possibility of “platypuses”.

NA – EAV Table - Consideration

- Disadvantages:
  - These VARCHAR columns have no meaning. Each entity can map a column for a completely unrelated attribute.
  - The Application mapping becomes a major source of data corruption bugs if mapping isn't cleanly implemented or if entity type changes are required overtime.
  - If unmapped columns are exposed to the users as custom column, there is not way to ensure that various users will be consistent when implementing these columns.
  - Users or Application logic becomes responsible to ensuring that all entities of a specific type will have the required associated attributes. (no DDL server constraints will work)
  - The NAEAV table uses a VARCHAR column for all attribute values regardless if Dates, Timestamps, Integers, Numerics or Booleans would be more appropriate
  - No Foreign Keys on Attribute Columns: The isn't a way to prevent bad data-entry. For example nothing would prevent a user from entering 'I like peanut butter.' for the attribute value for Birthday
  - Table design concept is badly de-normalized.

Bibliography

Works Cited

Batini, Carol, Stefano Ceri, and Shamkant B. Navathe. Conceptual Database Design : An Entity-Relationship Approach. Boston: Benjamin Cummings Company, 1991.

Celko, Joe. Joe Celko's SQL for Smarties : Advanced SQL Programming. 3rd ed. Greensboro: Morgan Kaufmann, 2005.

Celko, Joe. Joe Celko's Thinking in Sets : Auxiliary, Temporal, and Virtual Tables in SQL. New York: Elsevier Science & Technology Books, 2008.

Celko, Joe. Joe Celko's Trees and Hierarchies in SQL for Smarties. Greensboro: Morgan Kaufmann, 2004.

Douglas, Korry. PostgreSQL. 2nd ed. Indianapolis: Sams, 2005.

Questions?

I have a question!
What's all this Hierarchal
Nonsense?