Professor Bear :: The Relational Model

Bear

The Relational Model

The relational model (RM) for database management is an approach to managing data using a structure and language consistent with first-order predicate logic, first described in 1969 by Edgar F. Codd. In the relational model of a database, all data is represented in terms of tuples, grouped into relations. A database organized in terms of the relational model is a relational database.

In the relational model, related records are linked together with a "key". The purpose of the relational model is to provide a declarative method for specifying data and queries: users directly state what information the database contains and what information they want from it, and let the database management system software take care of describing data structures for storing the data and retrieval procedures for answering queries.

Tuples

A tuple is a finite ordered list of elements. In mathematics, an n-tuple is a sequence (or ordered list) of n elements, where n is a non-negative integer. There is only one 0-tuple, an empty sequence. An n-tuple is defined inductively using the construction of an ordered pair.

Note that a row in a table is a tuple. Tuples are *ordered* sets of known values with names. Thus, the following while having the same values are different tuples.

```
(x=1, y=2, z=3)
(z=3, y=2, x=1)
(y=2, z=3, x=1)
```

Tuples, sets with the same order would be of the following form:

```
(x=1, y=2, z=3)
(x=1, y=3, z=5)
(x=1, y=9, z=3)
```

Relations

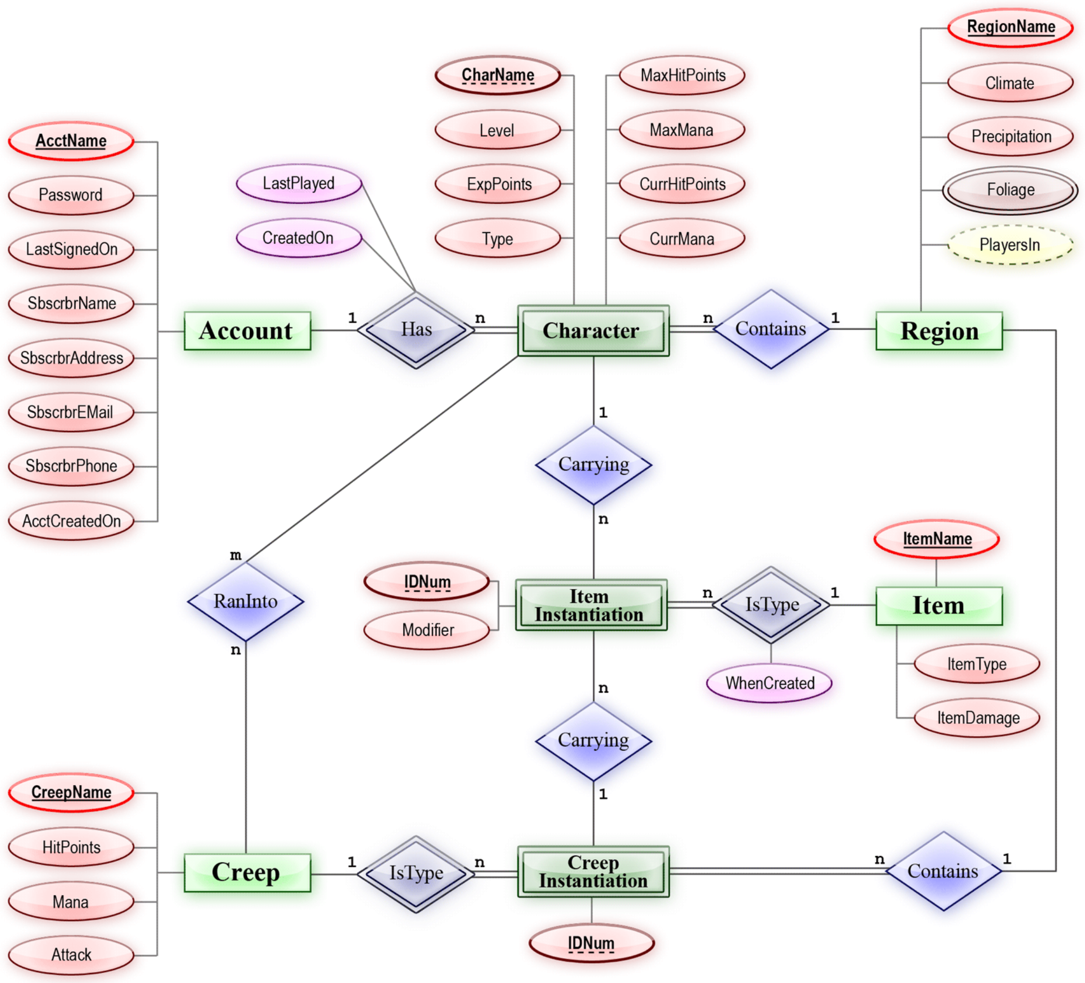
In relational database theory, a relation, as originally defined by E. F. Codd, is a set of tuples (A, B, ..., dn), where each element dj is a member of Dj, a data domain. In relational databases, tables are relations (in mathematical meaning). Relations are sets of tuples. Thus table row in relational database is tuple in relation.

In mathematics (more specifically, in set theory and logic), a relation is a property that assigns truth values to combinations (k-tuples) of k individuals. Typically, the property describes a possible connection between the components of a k-tuple. For a given set of k-tuples, a truth value is assigned to each k-tuple according to whether the property does or does not hold.

Thus a set of tuples would be the rows of a table in the elational model.

Entity Relationship Model (ERM)

An entity–relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

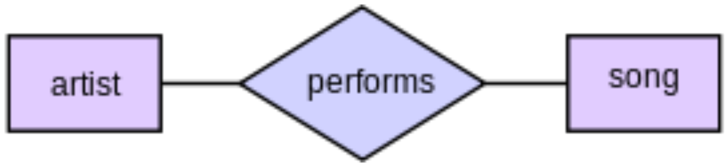


entity–relationship model (ER model)

Entities

An *Entity* is a person, a place, an object, an event, or a concept (often corresponds to a row in a table). All entities in an entity set have the same set of attributes.

- Each entity set has a key.
- Each attribute has a domain.



Two related entities

An *Entity Type* is a collection of entities (often corresponds to a table)

An Entity...

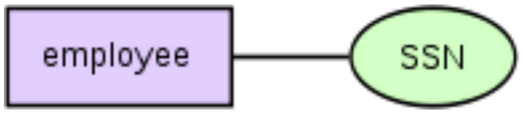
SHOULD BE:
* An object that will have many instances in the database
* An object that will be composed of multiple attributes * An object that we are trying to model

SHOULD NOT BE:
* A user of the database system
* An output of the database system ((e.g., a report)

Attributes

Attributes - Properties or characteristics of an entity or relationship type (often corresponds to a field in a table)

An *attribute* of an entity can have a value from a value set (domain)



An entity with an attribute

An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.

Attributes have a *domain*; the set of permitted values for each attribute. ((e.g. human age from 0 to 120 years)

Attribute types: Simple attributes Composite attributes. Single-valued attributes. Multi-valued attributes (Can be broken in to meaningful parts) ((e.g. phonenumbers or dates) Derived attributes (Can be computed from other attributes) ((e.g. age, given date of birth)

Classifications of attributes:
* Required versus Optional Attributes
* Simple versus Composite Attribute
* Single-Valued versus Multivalued Attribute
* Stored versus Derived Attributes
* Identifier Attributes

Keys

A key attribute of an entity type is one whose value uniquely identifies an entity of that type.

A combination of attributes may form a composite key. If there is no applicable value for an attribute that attribute is set to a null value. When you have your preliminary entities and attributes defined, you can start thinking about keys.

There are several types of keys:
* Primary Keys
* Candidate Keys
* Natural Keys
* Composite Keys
* Surrogate Keys
* Relational Keys

Primary Keys

A primary key uniquely identifies a row of data. A primary key must be unique for every row (that is, it can never repeat in the table that will result from the entity). For instance, a student ID can uniquely identify an individual student and the data associated with him or her. Every entity should have a primary key.

Candidate Keys

A candidate key is an attribute or attributes of an entity that have the potential to become a primary key. Candidate keys are not actual keys, but are a list of attributes that should be considered when choosing the primary key.

Natural Keys

There are basically two ways of making keys: natural and surrogate. Natural keys are keys formed by using an attribute that "naturally" belongs to the entity, such as a student ID or a phone number.

Composite Keys

Composite keys are keys composed of more than one attribute. For example, to get a unique designation of a course section, it is necessary to combine the quarter, the year, and the item number. Composite keys are one key made out of many parts.

Surrogate Keys

Surrogate keys are keys that have no meaning. Often they are just integers incremented row by row. They can also be things such as time stamps of auto-generated IDs.

Relational Keys

Relational Keys

Superkey - An attribute, or set of attributes, that uniquely identifies a tuple within a relation.

Candidate Key - Superkey (K) such that no proper subset is a superkey within the relation. In each tuple of R, values of K uniquely identify that tuple (uniqueness). No proper subset of K has the uniqueness property (irreducibility).

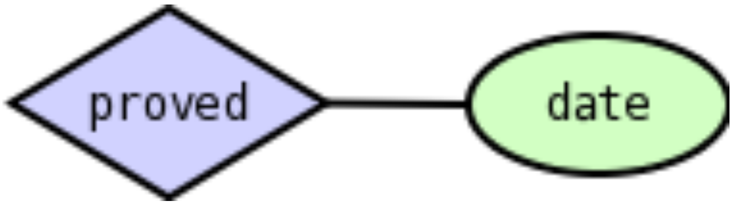
Primary Key - Candidate key selected to identify tuples uniquely within relation.

Alternate Keys - Candidate keys that are not selected to be primary key.

Foreign Key - Attribute, or set of attributes, within one relation that matches candidate key of some (possibly same) relation.

Relationship Sets

A *relationship* captures how entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. Examples: an owns relationship between a company and a computer, a supervises relationship between an employee and a department, a performs relationship between an artist and a song, a proves relationship between a mathematician and a conjecture.

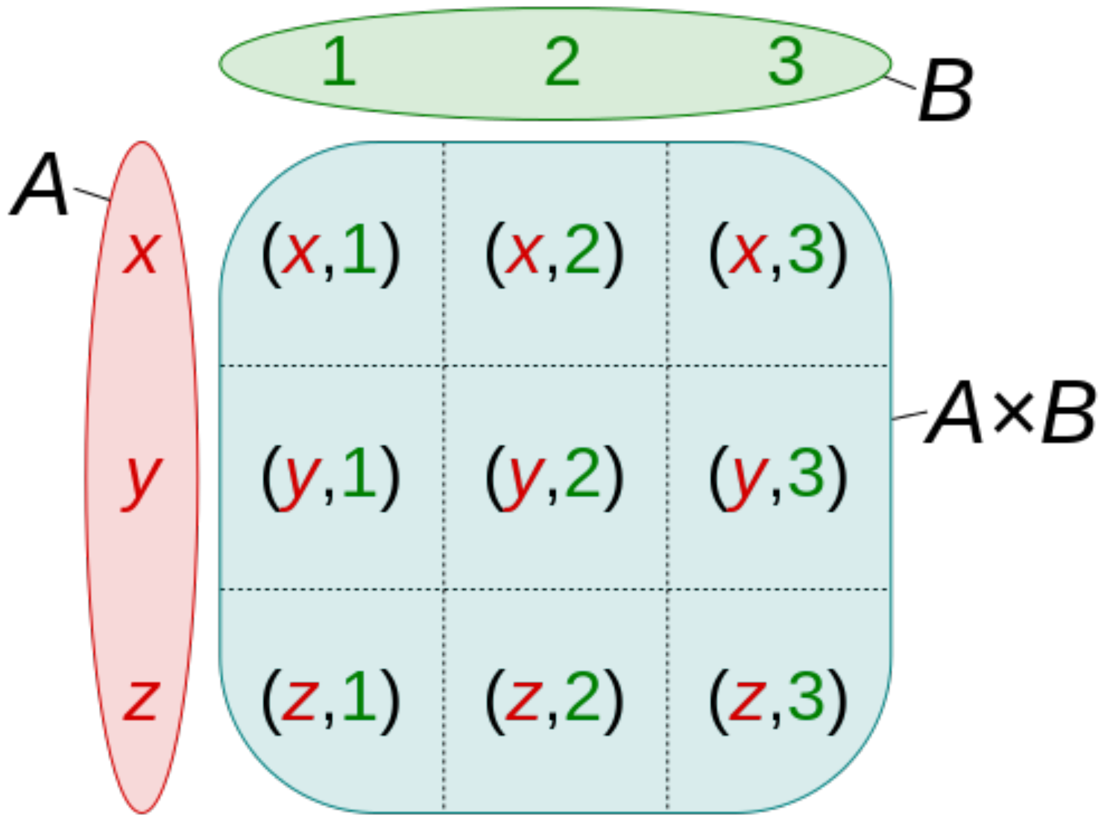


A relationship with an attribute

Relationship instance–link between entities (corresponds to primary key–foreign key equivalencies in related tables) Relationship type–category of relationship…link between entity types

Mathematical Definition of a Relation

In mathematics, a Cartesian product is a mathematical operation that returns a set (or product set or simply product) from multiple sets. That is, for sets A and B, the Cartesian product A × B is the set of all ordered pairs (a, b) where a ∈ A and b ∈ B. A table can be created by taking the Cartesian product of a set of rows and a set of columns. If the Cartesian product rows × columns is taken, the cells of the table contain ordered pairs of the form (row value, column value).



Cartesian product

Cartesian product $A{\times}B$ of the sets $A = \{x, y, z\}$ and $B = \{1,2,3\}$.

Consider two sets, A & B, where $A = \{x, y, z\}$ and $B = \{1,2,3\}$. Cartesian product, $A{\times}B$, is set of all ordered pairs, where first element is member of A and second element is member of B. $A{\times}B$ =
$\{(x, 1), (x, 2), (x, 3), (y, 1), (y, 2), (y, 3), (z, 1), (z, 2), (z, 3)\}$

Alternative way is to find all combinations of elements with first from A and second from B.

Any subset of Cartesian product is a relation; e.g.

$$R = \{(x, 1), (x, 2), (x, 3)\}$$

A relation specifies which pairs belong to a subset using some condition for selection;
(e.g. second element is 1:

$$R = \{(x, y) \quad | \quad x \in A, y \in B, \quad and \quad y = 1\}$$

first element is always twice the second:

$$S = \{(x, y) \quad | x \in A, y \in B, \quad and \quad x = 2y\}$$

Note that a relation may create an empty set ∅

Integrity Constraints

Intergrity means something like 'be right' and consistent. A constraint means something like limitation or restriction. The data in a database must 'be right' and consistent. To ensure this one often add limitations or restrictions to prevent bad data from existing in a database. More isn't always better.

There are several kinds of constraints: domain integrity, entity integrity, referential integrity, and other constraints.

Integrity Constraints

Null - Represents value for an attribute that is currently unknown or not applicable for tuple.

Null deals with incomplete or exceptional data. Null represents the absence of a value and is not the same as zero or spaces, which are values.

Entity Integrity - In a base relation, no attribute of a primary key can be null.

Domain Integrity - Domain integrity specifies that all columns in a relational database must be declared upon a defined domain and all of the values for each column must be a member if its respective domain.

Referential Integrity - Referential integrity is a property of data which, when satisfied, requires every value of one attribute (column) of a relation (table) to exist as a value of another attribute (column) in a different (or the same) relation. If foreign key exists in a relation, either foreign key value must match a candidate key value of some tuple in its home relation or foreign key value must be wholly null.

Structural Constraints

Structural constraints of limitations of a relationship type. Most typically this involves constraint on the *cardinality ratio* or the *participation* type.

The *cardinality ratio* limits the number of relationship instances an entity can participate in, eg. 1:1, 1:N, M:N

Participation constraint: If each entity of an entity type is required to participate in some instance of a relationship type, then that participation is total; otherwise, it is partial.

Cardinality Constraints

Cardinality Constraints

We express cardinality constraints by drawing either a directed line (®), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set. (e.g.: One-to-one relationship: A customer is associated with at most one loan via the relationship borrower A loan is associated with at most one customer via borrower

One-to-One Each entity in the relationship will have exactly one related entity One-to-Many An entity on one side of the relationship can have many related entities, but an entity on the other side will have a maximum of one related entity Many-to-Many Entities on both sides of the relationship can have many related entities on the other side

Degree of a Relationship Set

Degree of a relationship set refers to number of entity sets that participate in a relationship set.

Relationship sets that involve two entity sets are binary (or degree two). Generally, most relationship sets in a database system are binary.

Relationship sets may involve more than two entity sets.

Relationships between more than two entity sets are rare. Most relationships are binary.

Mapping Cardinalities

Cardinalities express the number of entities to which another entity can be associated via a relationship set.

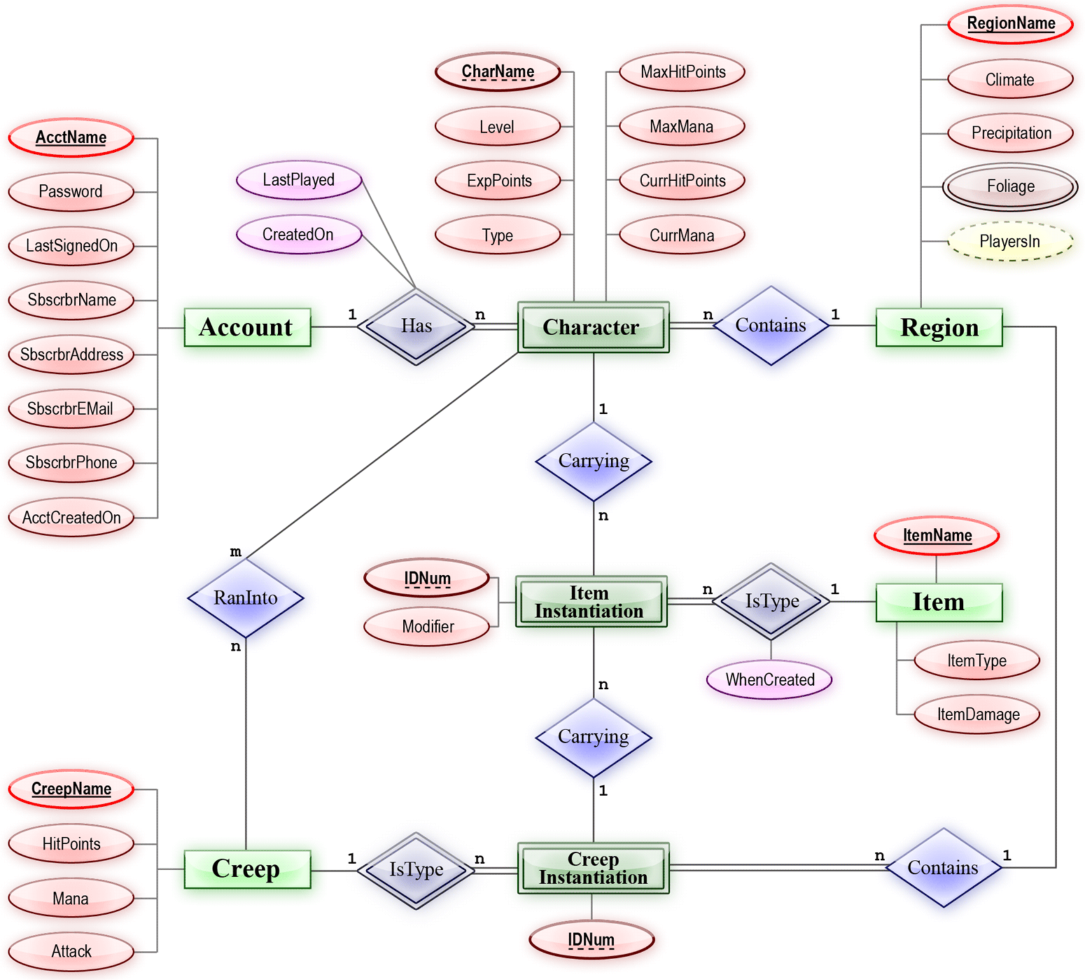
For a binary relationship set the mapping cardinality must be one of the following types:

- One to one
- One to many
- Many to one
- Many to many

E-R Diagrams

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database.

- Entities, which are represented by rectangles.
- Relationships are represented by diamond shapes.
- Attributes are represented by ovals.



entity–relationship model (ER model)

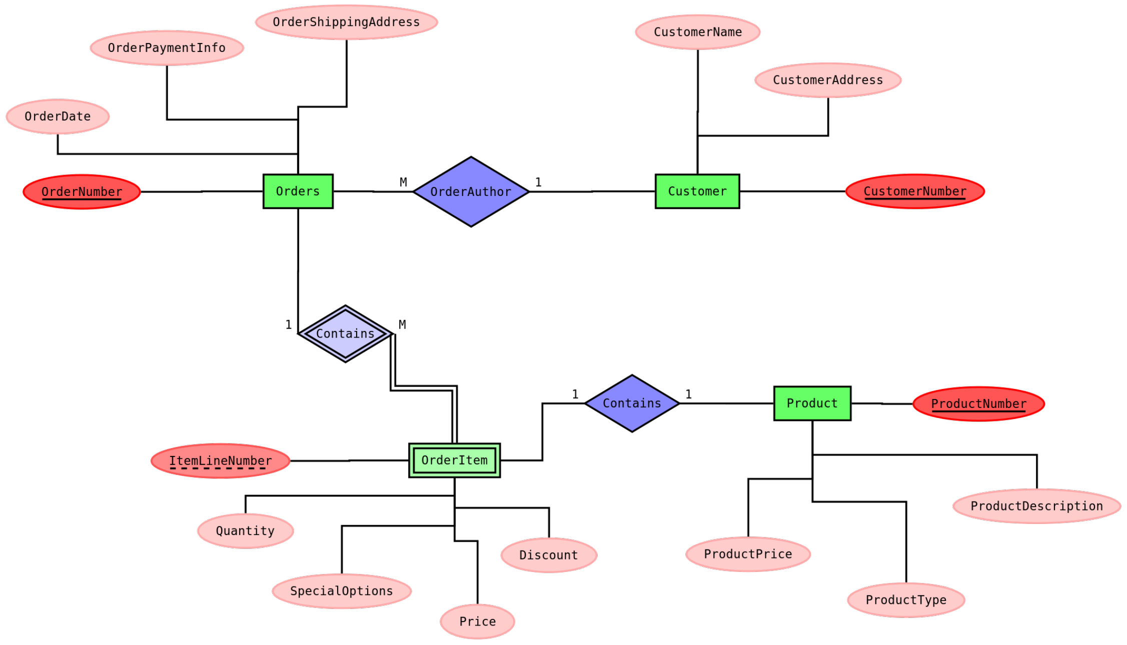
Roles

Roles - Entity sets of a relationship need not be distinct The labels “manager” and “worker” are called roles; they specify how employee entities interact via the works-for relationship set. Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles. Role labels are optional, and are used to clarify semantics of the relationship

Strong vs. Weak Entities, and Identifying Relationships

Strong entity - exists independently of other types of entities has its own unique identifier identifier underlined with single line.
Weak entity - dependent on a strong entity (identifying owner) cannot exist on its own does not have a unique identifier (only a partial identifier) entity box and partial identifier have double lines.

Consider a database that records customer orders, where an order is for one or more of the items that the enterprise sells. The database would contain a table identifying customers by a customer number (primary key); another identifying the products that can be sold by a product number (primary key); and it would contain a pair of tables describing orders.



eWeak entity example

One of the tables could be called Orders and it would have an order number (primary key) to identify this order uniquely, and would contain a customer number (foreign key) to identify who the products are being sold to, plus other information such as the date and time when the order was placed, how it will be paid for, where it is to be shipped to, and so on.

The other table could be called OrderItem; it would be identified by a compound key consisting of both the order number (foreign key) and an item line number; with other non-primary key attributes such as the product number (foreign key) that was ordered, the quantity, the price, any discount, any special options, and so on. There may be zero, one or many OrderItem entries corresponding to an Order entry, but no OrderItem entry can exist unless the corresponding Order entry exists. (The zero OrderItem case normally only applies transiently, when the order is first entered and before the first ordered item has been recorded.)

The OrderItem table stores weak entities precisely because an OrderItem has no meaning independent of the Order. Some might argue that an OrderItem does have some meaning on its own; it records that at some time not identified by the record, somebody not identified by the record ordered a certain quantity of a certain product. This information might be of some use on its own, but it is of limited use. For example, as soon as you want to find seasonal or geographical trends in the sales of the item, you need information from the related Order record.

An order would not exist without a product and a person to create the order, so it could be argued that an order would be described as a weak entity and that products ordered would be a multivalue attribute of the order.

Identifying relationship -links strong entities to weak entities.

Participation of an Entity Set in a Relationship Set

Participation of an Entity Set in a Relationship Set

Total participation (indicated by double line) - every entity in the entity set participates in at least one relationship in the relationship set (e.g. participation of loan in borrower is total every loan must have a customer associated to it via borrower)

Partial participation - some entities may not participate in any relationship in the relationship set (e.g. participation of customer in borrower is partial)

Design Issues

A database can be modeled as:
* A collection of entities
* Relationships among entities.

Example: person, tweet, company, event, film (a thing, a noun) Entities have attributes

Example: people have names and addresses
An entity set is a set of entities of the same type that share the same properties.

Example: set of all persons, companies, tweets

Use of entity sets vs. attributes. Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question.

Use of entity sets vs. relationship sets. Possible guideline is to designate a relationship set to describe an action that occurs between entities.

Binary versus n-ary relationship sets. Although it is possible to replace any nonbinary (n-ary, for n > 2) relationship set by a number of distinct binary relationship sets, a n-ary relationship set shows more clearly that several entities participate in a single relationship.

Entity vs. Attribute

Should address be an attribute of Employees or an entity (connected to Employees by a relationship)?

Depends upon the use we want to make of address information, and the semantics of the data:

If we have several addresses per employee, address must be an entity (since attributes cannot be set-valued).

If the structure (city, street, etc.) is important, (e.g., we want to retrieve employees in a given city, address must be modeled as an entity (since attribute values are atomic).

ER Model Example - Twitter

We want to model a Twitter user and a tweet? Entities? Entity Sets? entity attributes. (Until we consider ISA hierarchies, anyway!) Each entity set has a key. Each attribute has a domain.

Twitter API

dev.twitter.com

dev.twitter.com/rest/public

dev.twitter.com/rest/public/search

Twitter API JSON

dev.twitter.com/rest/reference/get/statuses/user_timeline

api.twitter.com/1.1/statuses/user_timeline.json

ER Model Example - Data Analytics Space

Data base of data science jobs, fellowships, learning resources, colleges and univerisities, algorithms, people, conferences, journals, buzz, trends.

Let's build this together! Right now!