WIKIPEDIA

Cardinality (data modeling)

In database design, the **cardinality** or fundamental principle of one data aspect with respect to another is a critical feature. The relationship of one to the other must be precise and exact between each other in order to explain how each aspect links together.

In the relational model, tables can be related as any of "one-to-many", "many-to-many" "one-to-zero-or-one", etc.. This is said to be the **cardinality** of a given table in relation to another.

For example, consider a database designed to keep track of hospital records. Such a database could have many tables like:

- a *doctor* table with information about physicians;
- a *patient* table for medical subjects undergoing treatment;
- and a *department* table with an entry for each division of a hospital.

In that model:

- There is a **many-to-many** relationship between the records in the doctor table and records in the patient table because doctors have many patients, and a patient could have several doctors;
- There is a **one-to-many** relationship between the department table and the doctor table because each doctor may work for only one department, but one department could have many doctors.

A "one-to-one" relationship is mostly used to split a table in two in order to provide information concisely and make it more understandable. In the hospital example, such a relationship could be used to keep apart doctors' own unique professional information from administrative details.

In data modeling, collections of data elements are grouped into "data tables" which contain groups of data field names called "database attributes". Tables are linked by "key fields". A "primary key" assigns a field to its "special order table". For example, the "Doctor Last Name" field might be assigned as a primary key of the Doctor table with all people having same last name organized alphabetically according to the first three letters of their first name. A table can also have a **foreign key** which indicates that field is linked to the primary key of another table.

A complex data model can involve hundreds of related tables. A renowned computer scientist, Edgar F. Codd, created a systematic method to decompose and organize relational databases. Codd's steps for organizing database tables and their keys is called database normalization, which avoids certain hidden database design errors (**delete anomalies** or **update anomalies**). In real life the process of database normalization ends up breaking tables into a larger number of smaller tables.

In real world, data modeling is critical because as the data grows voluminous, tables linked by keys must be used to speed up programmed retrieval of data. If a data model is poorly crafted, even a computer applications system with just a million records will give the end-users unacceptable response time delays. For this reason, data modeling is a keystone in the skills needed by a modern software developer.

Contents

Database modeling techniques

The entity–relationship model proposes a technique that produces entity–relationship diagrams (ERDs), which can be employed to capture information about data model entity types, relationships and cardinality. A Crow's foot shows a **one-to-many** relationship. Alternatively a single line represents a one-to-one relationship.

Application program modeling approaches

In the object-oriented application programming paradigm, which is decidedly marginal to database structure design, **UML class diagrams** may be used for object modeling (which should subsequently be mapped to a proper data modeling method). In that case, object relationships are modeled using UML associations, and multiplicity is used on those associations to denote **cardinality**. Here are some examples:

Relationship	Example	left	right
one-to-one	person ←→ id card	**1**	**1**
one-to-many *(optional on one side)*	person ←→ driving license	**1**	**0..1** or **?**
many-to-one	person ←→ birth place	**1..*** or **+**	**1**
many-to-many *(optional on both sides)*	person ←→ book	**0..*** or *	**0..*** or *

See also

- Arity

External links

- UML multiplicity as data model cardinality (http://www.agiledata.org/essays/umlDataModelingProfile.html#Relationships) - http://www.agiledata.org

---