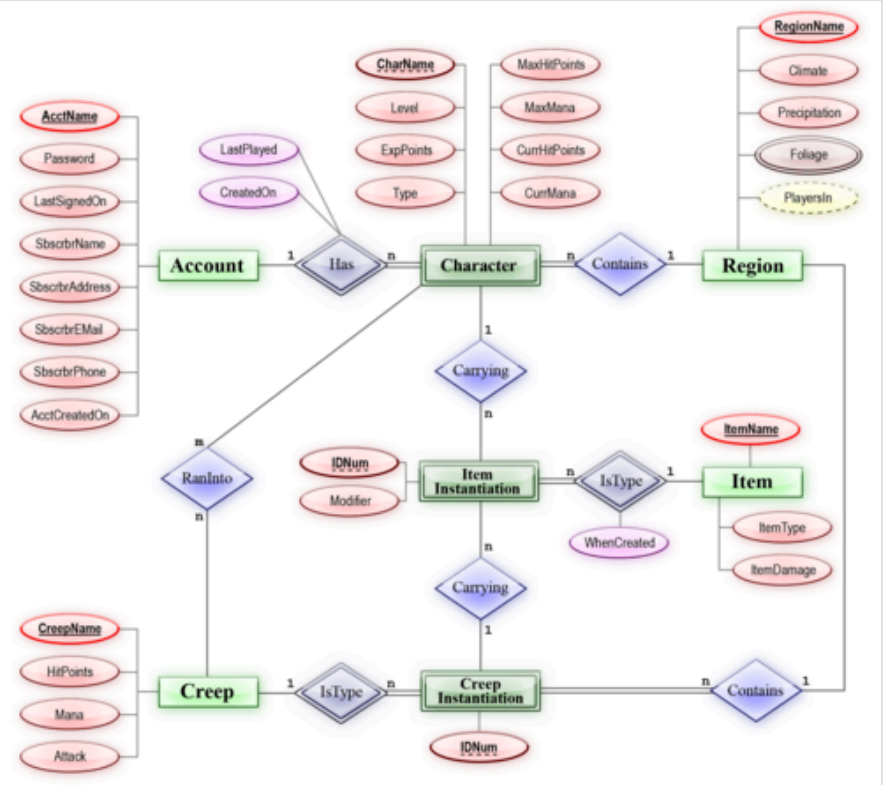
Entity–relationship model

An **entity–relationship model** (or **ER model**) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between entities (instances of those entity types).



An entity–attribute-relationship diagram for an MMORPG using Chen's notation

In software engineering, an ER model is commonly formed to represent things a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model,[1] that defines a data or information structure that can be implemented in a database, typically a relational database.

Entity–relationship modeling was developed for database and design by Peter Chen and published in a 1976 paper,[2] with variants of the idea existing previously.[3] Today it is commonly used for teaching students the basics of database structure. Some ER models show super and subtype entities connected by generalization-specialization relationships,[4] and an ER model can also be used to specify domain-specific ontologies.

Introduction

An ER model usually results from systematic analysis to define and describe the data created and needed by processes in a business area. Typically, it represents records of entities and events monitored and directed by business processes, rather than the processes themselves. It is usually drawn in a graphical form as boxes (*entities*) that are connected by lines (*relationships*) which express the associations and dependencies between entities. It can also be expressed in a verbal form, for example: *one building may be divided into zero or more apartments, but one apartment can only be located in one building.*[5]

Entities may be defined not only by relationships, but also by additional properties (*attributes*), which include identifiers called "primary keys". Diagrams created to represent attributes as well as entities and relationships may be called entity-attribute-relationship diagrams, rather than entity–relationship

models.[6]

An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity.

There is a tradition for ER/data models to be built at two or three levels of abstraction. The conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three schema approach to software engineering.

Conceptual data model

This is the highest level ER model in that it contains the least granular detail but establishes the overall scope of what is to be included within the model set. The conceptual ER model normally defines master reference data entities that are commonly used by the organization. Developing an enterprise-wide conceptual ER model is useful to support documenting the data architecture for an organization.
A conceptual ER model may be used as the foundation for one or more *logical data models* (see below). The purpose of the conceptual ER model is then to establish structural metadata commonality for the master data entities between the set of logical ER models. The conceptual data model may be used to form commonality relationships between ER models as a basis for data model integration.

Logical data model

A logical ER model does not require a conceptual ER model, especially if the scope of the logical ER model includes only the development of a distinct information system. The logical ER model contains more detail than the conceptual ER model. In addition to master data entities, operational and transactional data entities are now defined. The details of each data entity are developed and the relationships between these data entities are established. The logical ER model is however developed independently of the specific database management system into which it can be implemented.

Physical data model

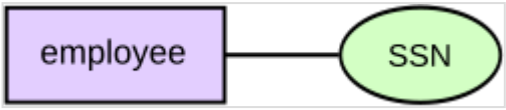
One or more physical ER models may be developed from each logical ER model. The physical ER model is normally developed to be instantiated as a database. Therefore, each physical ER model must contain enough detail to produce a database and each physical ER model is technology dependent since each database management system is somewhat different.
The physical model is normally instantiated in the structural metadata of a database management system as relational database objects such as database tables, database indexes such as unique key indexes, and database constraints such as a foreign key constraint or a commonality constraint. The ER model is also normally used to design modifications to the relational database objects and to maintain the structural metadata of the database.

The first stage of information system design uses these models during the requirements analysis to describe information needs or the type of information that is to be stored in a database. The data modeling technique can be used to describe any ontology (i.e. an overview and classifications of used terms and their relationships) for a certain area of interest. In the case of the design of an information system that is based on a database, the conceptual data model is, at a later stage (usually called logical design), mapped to a logical data model, such as the relational model. This in turn is mapped to a physical model during physical design. Sometimes, both of these phases are referred to as "physical design."

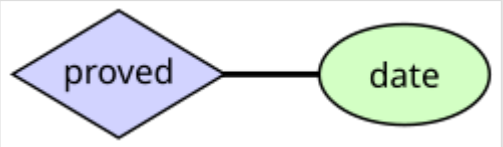
Components

An *entity* may be defined as a thing that is capable of an independent existence that can be uniquely identified, and is capable of storing data.[7] An entity is an abstraction from the complexities of a domain. When we speak of an entity, we normally speak of some aspect of the real world that can be distinguished from other aspects of the real world.[8]

Two related entities



An entity with an attribute



A relationship with an attribute

An entity is a thing that exists either physically or logically. An entity may be a physical object such as a house or a car (they exist physically), an event such as a house sale or a car service, or a concept such as a customer transaction or order (they exist logically—as a concept). Although the term entity is the one most commonly used, following Chen, entities and entity-types should be distinguished. An entity-type is a category. An entity, strictly speaking, is an instance of a given entity-type. There are usually many instances of an entity-type. Because the term entity-type is somewhat cumbersome, most people tend to use the term entity as a synonym.

Primary key

Entities can be thought of as nouns.[9] Examples include a computer, an employee, a song, or a mathematical theorem.

A *relationship* captures how entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns.[9] Examples include 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, and a *proves* relationship between a mathematician and a conjecture.

The model's linguistic aspect described above is used in the declarative database query language ERROL, which mimics natural language constructs. ERROL's semantics and implementation are based on reshaped relational algebra (RRA), a relational algebra that is adapted to the entity–relationship model and captures its linguistic aspect.

Entities and relationships can both have attributes. For example, an *employee* entity might have a *Social Security Number* (SSN) attribute, while a *proved* relationship may have a *date* attribute.

All entities except weak entities must have a minimal set of uniquely identifying attributes that may be used as a unique/primary key.

Entity-relationship diagrams (ERDs) do not show single entities or single instances of relations. Rather, they show entity sets (all entities of the same entity type) and relationship sets (all relationships of the same relationship type). For example, a particular *song* is an entity, the collection of all songs in a database is an entity set, the *eaten* relationship between a child and his lunch is a single relationship, and

the set of all such child-lunch relationships in a database is a relationship set. In other words, a relationship set corresponds to a relation in mathematics, while a relationship corresponds to a member of the relation.

Certain cardinality constraints on relationship sets may be indicated as well.

Guiding rules for mapping natural language descriptions into ER diagrams[10]

English grammar structure	ER structure
Common noun	Entity type
Proper noun	Entity
Transitive verb	Relationship type
Intransitive verb	Attribute type
Adjective	Attribute for entity
Adverb	Attribute for relationship

Physical views show how data is actually stored.

Relationships, roles, and cardinalities

Chen's original paper gives an example of a relationship and its roles. He describes a relationship "marriage" and its two roles, "husband" and "wife".

A person plays the role of husband in a marriage (relationship) and another person plays the role of wife in the (same) marriage. These words are nouns.

Chen's terminology has also been applied to earlier ideas. The lines, arrows, and crow's feet of some diagrams owes more to the earlier Bachman diagrams than to Chen's relationship diagrams.

Another common extension to Chen's model is to "name" relationships and roles as verbs or phrases.

Role naming

It has also become prevalent to name roles with phrases such as *is the owner of* and *is owned by*. Correct nouns in this case are *owner* and *possession*. Thus, *person plays the role of owner* and *car plays the role of possession* rather than *person plays the role of*, *is the owner of*, etc.

Using nouns has direct benefit when generating physical implementations from semantic models. When a *person* has two relationships with *car* it is possible to generate names such as *owner_person* and *driver_person*, which are immediately meaningful.[11]

Cardinalities

Modifications to the original specification can be beneficial. Chen described look-across cardinalities. As an aside, the Barker–Ellis notation, used in Oracle Designer, uses same-side for minimum cardinality (analogous to optionality) and role, but look-across for maximum cardinality (the crow's foot).

Research by Merise, Elmasri & Navathe and others has shown there is a preference for same-side for roles and both minimum and maximum cardinalities,[12][13][14] and researchers (Feinerer, Dullea et al.) have shown that this is more coherent when applied to n-ary relationships of order greater than 2.[15][16]

Dullea et al. states: "A 'look across' notation such as used in the UML does not effectively represent the semantics of participation constraints imposed on relationships where the degree is higher than binary."

Feinerer says: "Problems arise if we operate under the look-across semantics as used for UML associations. Hartmann[17] investigates this situation and shows how and why different transformations fail." *(Although the "reduction" mentioned is spurious as the two diagrams 3.4 and 3.5 are in fact the same)* and also "As we will see on the next few pages, the look-across interpretation introduces several difficulties that prevent the extension of simple mechanisms from binary to n-ary associations."

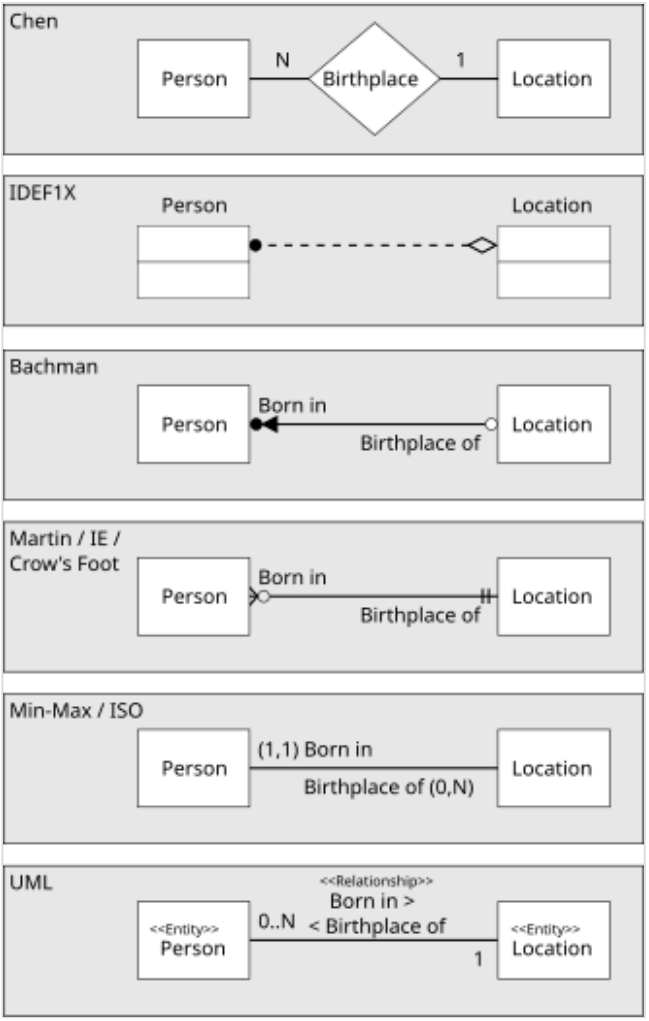
Chen's notation for entity–relationship modeling uses rectangles to represent entity sets, and diamonds to represent relationships appropriate for first-class objects: they can have attributes and relationships of their own. If an entity set participates in a relationship set, they are connected with a line.

Attributes are drawn as ovals and connected with a line to exactly one entity or relationship set.

Cardinality constraints are expressed as follows:

- a double line indicates a *participation constraint*, totality, or surjectivity: all entities in the entity set must participate in *at least one* relationship in the relationship set;
- an arrow from an entity set to a relationship set indicates a key constraint, i.e. injectivity: each entity of the entity set can participate in *at most one* relationship in the relationship set;
- a thick line indicates both, i.e. bijectivity: each entity in the entity set is involved in *exactly one* relationship.
- an underlined name of an attribute indicates that it is a key: two different entities or relationships with this attribute always have different values for this attribute.

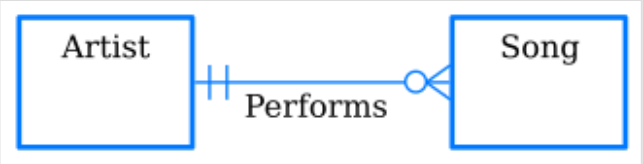
Attributes are often omitted as they can clutter up a diagram. Other diagram techniques often list entity attributes within the rectangles drawn for entity sets.



Various methods of representing the same one to many relationship. In each case, the diagram shows the relationship between a person and a place of birth: each person must have been born at one, and only one, location, but each location may have had zero or more people born at it.

Related diagramming convention techniques

- Bachman notation
- Barker's notation
- EXPRESS
- IDEF1X
- § Crow's foot notation (also Martin notation)
- (min, max)-notation of Jean-Raymond Abrial in 1974
- UML class diagrams
- Merise
- Object-role modeling



Two related entities shown using Crow's Foot notation. In this example, an optional relationship is shown between Artist and Song; the symbol composed of branching lines, closest to the song entity represents "zero, one, or many", whereas a song has "one and only one" Artist, emphasized by the symbol composed of parallel lines. The former is therefore read as, an Artist (can) perform(s) "zero, one, or many" song(s).

Crow's foot notation

Crow's foot notation, the beginning of which dates back to an article by Gordon Everest (1976),[18] is used in Barker's notation, Structured Systems Analysis and Design Method (SSADM), and information technology engineering. Crow's foot diagrams represent entities as boxes, and relationships as lines between the boxes. Different shapes at the ends of these lines represent the relative cardinality of the relationship.

Crow's foot notation was in use in ICL in 1978,[19] and was used in the consultancy practice CACI. Many of the consultants at CACI (including Richard Barker) came from ICL and subsequently moved to Oracle UK, where they developed the early versions of Oracle's CASE tools, introducing the notation to a wider audience.

With this notation, relationships cannot have attributes. Where necessary, relationships are promoted to entities in their own right: for example, if it is necessary to capture where and when an artist performed a song, a new entity "performance" is introduced (with attributes reflecting the time and place), and the relationship of an artist to a song becomes an indirect relationship via the performance (artist-performs-performance, performance-features-song).

Three symbols are used to represent cardinality:

- the *ring* represents "zero"
- the *dash* represents "one"
- the *crow's foot* represents "many" or "infinite"

These symbols are used in pairs to represent the four types of cardinality that an entity may have in a relationship. The inner component of the notation represents the minimum, and the outer component represents the maximum.

- *ring* and *dash* → **minimum zero, maximum one (optional)**
- *dash* and *dash* → **minimum one, maximum one (mandatory)**
- *ring* and *crow's foot* → **minimum zero, maximum many (optional)**

- *dash* and *crow's foot* → **minimum one, maximum many (mandatory)**

Model usability issues

Users of a modeled database can encounter two well-known issues where the returned results differ from what the query author assumed. These are known as the fan trap and the chasm trap, and they can lead to inaccurate query results if not properly handled during the design of the Entity-Relationship Model (ER Model).

Both the fan trap and chasm trap underscore the importance of ensuring that ER models are not only technically correct but also fully and accurately reflect the real-world relationships they are designed to represent. Identifying and resolving these traps early in the design process helps avoid significant issues later, especially in complex databases intended for business intelligence or decision support.

Fan trap

The first issue is the fan trap. It occurs when a (master) table links to multiple tables in a one-to-many relationship. The issue derives its name from the visual appearance of the model when it is drawn in an entity–relationship diagram, as the linked tables 'fan out' from the master table. This type of model resembles a star schema, which is a common design in data warehouses. When attempting to calculate sums over aggregates using standard SQL queries based on the master table, the results can be unexpected and often incorrect due to the way relationships are structured. The miscalculation happens because SQL treats each relationship individually, which may result in double-counting or other inaccuracies. This issue is particularly common in decision support systems. To mitigate this, either the data model or the SQL query itself must be adjusted. Some database querying software designed for decision support includes built-in methods to detect and address fan traps.

Chasm trap

The second issue is the chasm trap. A chasm trap occurs when a model suggests the existence of a relationship between entity types, but the pathway between these entities is incomplete or missing in certain instances.

For example, imagine a database where a Building has one or more Rooms, and these Rooms hold zero or more Computers. One might expect to query the model to list all Computers in a Building. However, if a Computer is temporarily not assigned to a Room (perhaps under repair or stored elsewhere), it won't be included in the query results. The query would only return Computers currently assigned to Rooms, not all Computers in the Building. This reflects a flaw in the model, as it fails to account for Computers that are in the Building but not in a Room. To resolve this, an additional relationship directly linking the Building and Computers would be required.

In semantic modeling

Semantic model

A semantic model is a model of concepts and is sometimes called a "platform independent model". It is an intensional model. At least since Carnap, it is well known that:[20]

> *"...the full meaning of a concept is constituted by two aspects, its intension and its extension. The first part comprises the embedding of a concept in the world of concepts as a whole, i.e. the totality of all relations to other concepts. The second part establishes the referential meaning of the concept, i.e. its counterpart in the real or in a possible world".*

Extension model

An extensional model is one that maps to the elements of a particular methodology or technology, and is thus a "platform specific model". The UML specification explicitly states that associations in class models are extensional and this is in fact self-evident by considering the extensive array of additional "adornments" provided by the specification over and above those provided by any of the prior candidate "semantic modelling languages"."UML as a Data Modeling Notation, Part 2" (http://www.tdan.com/view-articles/8589)

Entity–relationship origins

Peter Chen, the father of ER modeling said in his seminal paper:

> "*The entity-relationship model adopts the more natural view that the real world consists of entities and relationships. It incorporates some of the important semantic information about the real world.*"[2]

In his original 1976 article Chen explicitly contrasts entity–relationship diagrams with record modelling techniques:

> "*The data structure diagram is a representation of the organization of records and is not an exact representation of entities and relationships.*"

Several other authors also support Chen's program:[21][22][23][24][25]

Philosophical alignment

Chen is in accord with philosophical traditions from the time of the Ancient Greek philosophers: Plato and Aristotle.[26] Plato himself associates knowledge with the apprehension of unchanging Forms (namely, archetypes or abstract representations of the many types of things, and properties) and their relationships to one another.

Limitations

- An ER model is primarily conceptual, an ontology that expresses predicates in a domain of knowledge.
- ER models are readily used to represent relational database structures (after Codd and Date) but not so often to represent other kinds of data structure (such as data warehouses and document stores)
- Some ER model notations include symbols to show super-sub-type relationships and mutual exclusion between relationships; some do not.
- An ER model does not show an entity's life history (how its attributes and/or relationships change over time in response to events). For many systems, such state changes are nontrivial and important enough to warrant explicit specification.
- Some researchers have extended ER modeling with constructs to represent state changes, an approach supported by the original author;[27] an example is Anchor Modeling.
- Others model state changes separately, using state transition diagrams or some other process modeling technique.
- Many other kinds of diagram are drawn to model other aspects of systems, including the 14 diagram types offered by UML.[28]
- Today, even where ER modeling could be useful, it is uncommon because many use tools that support similar kinds of model, notably class diagrams for OO programming and data models for relational database management systems. Some of these tools can generate code from diagrams and reverse-engineer diagrams from code.
- In a survey, Brodie and Liu[29] could not find a single instance of entity–relationship modeling inside a sample of ten Fortune 100 companies. Badia and Lemire[30] blame this lack of use on the lack of guidance but also on the lack of benefits, such as lack of support for data integration.
- The enhanced entity–relationship model (EER modeling) introduces several concepts not in ER modeling, but are closely related to object-oriented design, like is-a relationships.
- For modelling temporal databases, numerous ER extensions have been considered.[31] Similarly, the ER model was found unsuitable for multidimensional databases (used in OLAP applications); no dominant conceptual model has emerged in this field yet, although they generally revolve around the concept of OLAP cube (also known as *data cube* within the field).[32]

See also

- Associative entity – Term in relational and entity–relationship theory
- Concept map – Diagram showing relationships among concepts
- Database design – Designing how data is held in a database
- Data structure diagram
- Enhanced entity–relationship model – Extended version of the Entity-Relationship model for database design
- Enterprise architecture framework – Frame in which the architecture of a company is defined
- Entity Data Model – Open source object-relational mapping framework
- Value range structure diagrams
- Comparison of data modeling tools – Comparison of notable data modeling tools

- Knowledge graph – Type of knowledge base
- Ontology – Specification of a conceptualization
- Object-role modeling – Programming technique
- Three schema approach – Approach to building information systems
- Structured entity relationship model
- Schema-agnostic databases

References

1. Bagui & Earp 2022, p. 72, §4.2.1.
2. Chen, Peter (March 1976). "The Entity-Relationship Model - Toward a Unified View of Data". *ACM Transactions on Database Systems*. **1** (1): 9–36. CiteSeerX 10.1.1.523.6679 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.523.6679). doi:10.1145/320434.320440 (https://doi.org/10.1145%2F320434.320440). S2CID 52801746 (https://api.semanticscholar.org/CorpusID:52801746).
3. A.P.G. Brown, "Modelling a Real-World System and Designing a Schema to Represent It", in Douque and Nijssen (eds.), *Data Base Description*, North-Holland, 1975, ISBN 0-7204-2833-5.
4. "Lesson 5: Supertypes and Subtypes" (https://docs.microsoft.com/en-us/previous-versions/tn-archive/cc505839(v%3dtechnet.10)). *docs.microsoft.com*.
5. "Introduction of ER Model" (https://www.geeksforgeeks.org/dbms/introduction-of-er-model/). *GeeksforGeeks*. 2015-10-13. Retrieved 2026-01-05.
6. "What is Entity Relationship Diagram (ERD)?" (https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/;WWWSESSIONID=22BC6981693B8A61AABA219B83D88AD7.www1?utm_source). *www.visual-paradigm.com*. Retrieved 2026-01-05.
7. Bagui & Earp 2022, p. 73-74, §4.3.
8. Beynon-Davies, Paul (2004). *Database Systems*. Basingstoke, UK: Palgrave: Houndmills. ISBN 978-1403916013.
9. Bagui & Earp 2022, p. 112-116, §5.5.
10. "English, Chinese and ER diagrams" (http://bit.csc.lsu.edu/~chen/pdf/ER_C.pdf) by Peter Chen
11. "The Pangrammaticon: Emotion and Society" (http://pangrammaticon.blogspot.com/2013/01/emotion-and-society.html). January 3, 2013.
12. Hubert Tardieu, Arnold Rochfeld and René Colletti La methode MERISE: Principes et outils (Paperback - 1983)
13. Elmasri, Ramez, B. Shamkant, Navathe, Fundamentals of Database Systems, third ed., Addison-Wesley, Menlo Park, CA, USA, 2000.
14. Atzeni, Paolo; Chu, Wesley; Lu, Hongjun; Ling, Tok Wang; Zhou, Shuigeng (2004-10-27). *ER 2004 : 23rd International Conference on Conceptual Modeling, Shanghai, China, November 8-12, 2004* (https://books.google.com/books?id=odZK99osY1EC&q=genova&pg=PA52). Springer. ISBN 9783540237235.
15. "A Formal Treatment of UML Class Diagrams as an Efficient Method for Configuration Management 2007" (https://web.archive.org/web/20111006062336/http://publik.tuwien.ac.at/files/pub-inf_4582.pdf) (PDF). Archived from the original (http://publik.tuwien.ac.at/files/pub-inf_4582.pdf) (PDF) on 2011-10-06. Retrieved 2011-07-26.
16. "James Dullea, Il-Yeol Song, Ioanna Lamprou - An analysis of structural validity in entity-relationship modeling 2002" (https://web.archive.org/web/20090424030434/http://www.ischool.drexel.edu/faculty/song/publications/p_DKE_03_Validity.pdf) (PDF). Archived from the original (http://www.ischool.drexel.edu/faculty/song/publications/p_DKE_03_Validity.pdf) (PDF) on April 24, 2009.

17. Hartmann, Sven. "Reasoning about participation constraints and Chen's constraints (http://crpit.com/confpapers/CRPITV17Hartmann.pdf) Archived (https://web.archive.org/web/20130510031920/http://crpit.com/confpapers/CRPITV17Hartmann.pdf) 2013-05-10 at the Wayback Machine". Proceedings of the 14th Australasian database conference-Volume 17. Australian Computer Society, Inc., 2003.
18. G. Everest, "BASIC DATA STRUCTURE MODELS EXPLAINED WITH A COMMON EXAMPLE", in Computing Systems 1976, Proceedings Fifth Texas Conference on Computing Systems, Austin, TX, 1976 October 18–19, pages 39-46. (Long Beach, CA: IEEE Computer Society Publications Office).
19. "Introduction to Data Analysis", ICL Training Publication T2384 Issue 2, November 1978
20. "The Role of Intensional and Extensional Interpretation in Semantic Representations" (https://www.researchgate.net/publication/249853469).
21. Kent in "Data and Reality" (http://www.bkent.net/Doc/darxrp.htm) :
    > "One thing we ought to have clear in our minds at the outset of a modelling endeavour is whether we are intent on describing a portion of "reality" (some human enterprise) or a data processing activity."
22. Abrial in "Data Semantics" : "... the so called "logical" definition and manipulation of data are still influenced (sometimes unconsciously) by the "physical" storage and retrieval mechanisms currently available on computer systems."
23. Stamper: "They pretend to describe entity types, but the vocabulary is from data processing: fields, data items, values. Naming rules don't reflect the conventions we use for naming people and things; they reflect instead techniques for locating records in files."
24. In Jackson's words: "The developer begins by creating a model of the reality with which the system is concerned, the reality that furnishes its [the system's] subject matter ..."
25. Elmasri, Navathe: "The ER model concepts are designed to be closer to the user's perception of data and are not meant to describe the way in which data will be stored in the computer."
26. Paolo Rocchi, *Janus-Faced Probability*, Springer, 2014, p. 62.
27. P. Chen. Suggested research directions for a new frontier: Active conceptual modeling (https://doi.org/10.1007%2F11901181_1). ER 2006, volume 4215 of Lecture Notes in Computer Science, pages 1–4. Springer Berlin / Heidelberg, 2006.
28. Carte, Traci A.; Jasperson, Jon (Sean); and Cornelius, Mark E. (2020) "Integrating ERD and UML Concepts When Teaching Data Modeling," Journal of Information Systems Education: Vol. 17 : Iss. 1, Article 9. (https://aisel.aisnet.org/jise/vol17/iss1/9)
29. The power and limits of relational technology in the age of information ecosystems (https://www.deri.ie/content/power-and-limits-relational-technology-age-information-ecosystems) Archived (https://web.archive.org/web/20160917100748/https://www.deri.ie/content/power-and-limits-relational-technology-age-information-ecosystems) 2016-09-17 at the Wayback Machine. On The Move Federated Conferences, 2010.
30. A. Badia and D. Lemire. A call to arms: revisiting database design (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.297.4916&rep=rep1&type=pdf). Citeseerx,
31. Gregersen, Heidi; Jensen, Christian S. (1999). "Temporal Entity-Relationship models—a survey". *IEEE Transactions on Knowledge and Data Engineering*. **11** (3): 464–497. Bibcode:1999ITKDE..11..464G (https://ui.adsabs.harvard.edu/abs/1999ITKDE..11..464G). CiteSeerX 10.1.1.1.2497 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.2497). doi:10.1109/69.774104 (https://doi.org/10.1109%2F69.774104).
32. RICCARDO TORLONE (2003). "Conceptual Multidimensional Models" (http://torlone.dia.uniroma3.it/pubs/idea03.pdf) (PDF). In Maurizio Rafanelli (ed.). *Multidimensional Databases: Problems and Solutions*. Idea Group Inc (IGI). ISBN 978-1-59140-053-0.

Further reading

- Chen, Peter (2002). "Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned" (http://bit.csc.lsu.edu/~chen/pdf/Chen_Pioneers.pdf) (PDF). *Software pioneers*. Springer-Verlag. pp. 296–310. ISBN 978-3-540-43081-0.
- Barker, Richard (1990). *CASE Method: Entity Relationship Modelling* (https://archive.org/details/casemethod00rich). Addison-Wesley. ISBN 978-0201416961.
- Barker, Richard (1990). *CASE Method: Tasks and Deliverables* (https://archive.org/details/casemethodtasksd00bark). Addison-Wesley. ISBN 978-0201416978.
- Mannila, Heikki; Räihä, Kari-Jouko (1992). *The Design of Relational Databases*. Addison-Wesley. ISBN 978-0201565232.
- Thalheim, Bernhard (2000). *Entity-Relationship Modeling: Foundations of Database Technology*. Springer. ISBN 978-3-540-65470-4.
- Bagui, Sikha; Earp, Richard Walsh (2022). *Database Design Using Entity-Relationship Diagrams*. Auerbach Publications. ISBN 978-1-032-01718-1.

External links

- "The Entity Relationship Model: Toward a Unified View of Data" (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.123.1085)
- Entity Relationship Modelling (https://web.archive.org/web/20160108012618/http://www.databasedesign.co.uk/bookdatabasesafirstcourse/chap3/chap3.htm)
- Logical Data Structures (LDSs) - Getting started (http://arquivo.pt/wayback/20160516171832/http://www.cems.uwe.ac.uk/~tdrewry/lds.htm) by Tony Drewry.
- Crow's Foot Notation (https://web.archive.org/web/20120125011338/http://www2.cs.uregina.ca/~bernatja/crowsfoot.html)
- Kinds of Data Models -- and How to Name Them (https://www.youtube.com/watch?v=PU7nKBNR1Vs) presentation by David Hay