Database Fundamentals

Robert J. Robbins

Johns Hopkins University
rrobbins@gdb.org

What is a Database?

General:

- A database is any collection of related data.

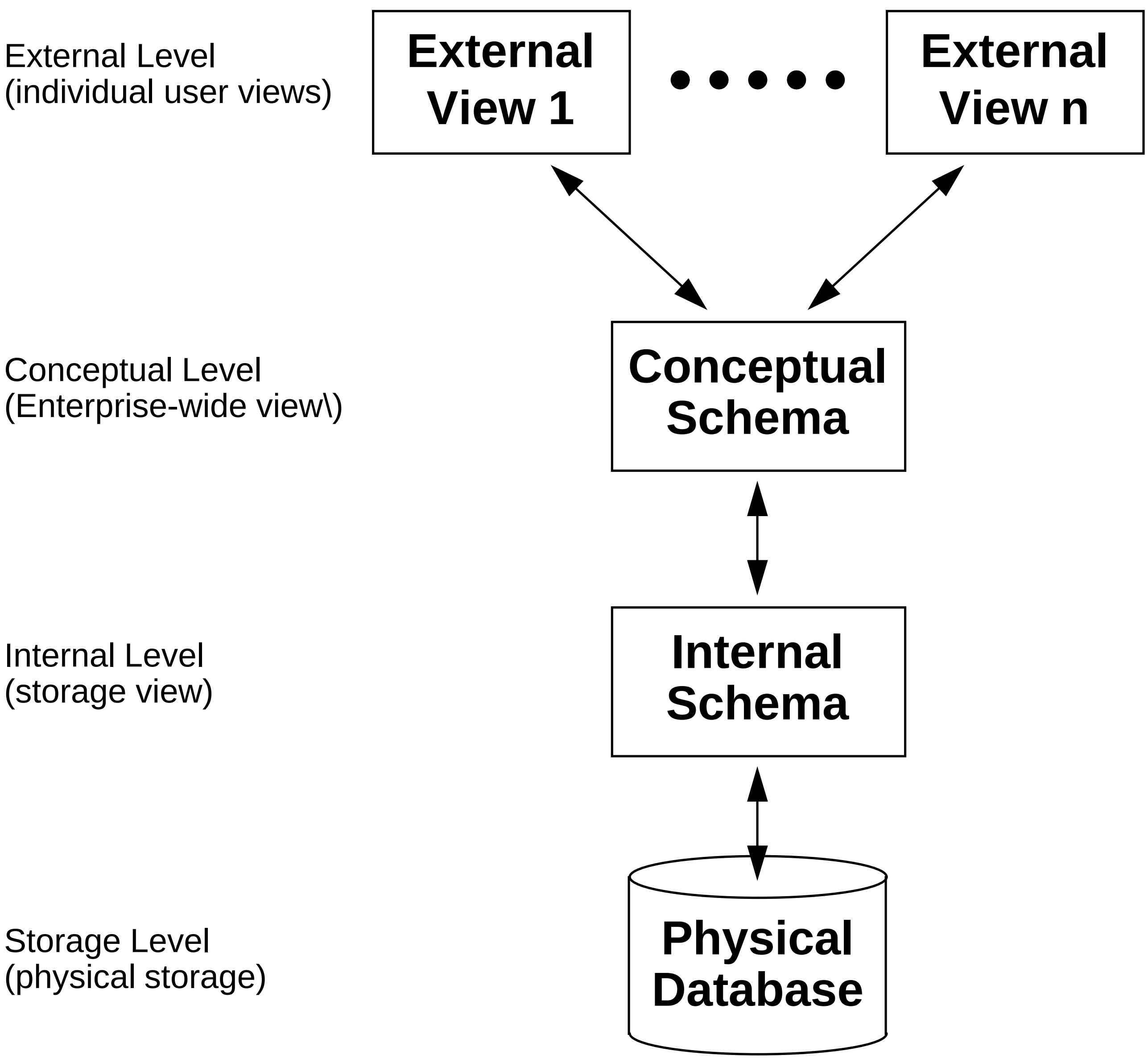
Restrictive:

- A database is a persistent, logically coherent collection of inherently meaningful data, relevant to some aspects of the real world.

The portion of the real world relevant to the database is sometimes referred to as the **universe of discourse** or as the **database miniworld**. Whatever it is called, it must be well understood by the designers of the database.

What is a Database Management System?

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. According to the ANSI/SPARC DBMS Report (1977), a DBMS should be envisioned as a multi-layered system:



What Does a DBMS Do?

Database management systems provide several functions in addition to simple file management:

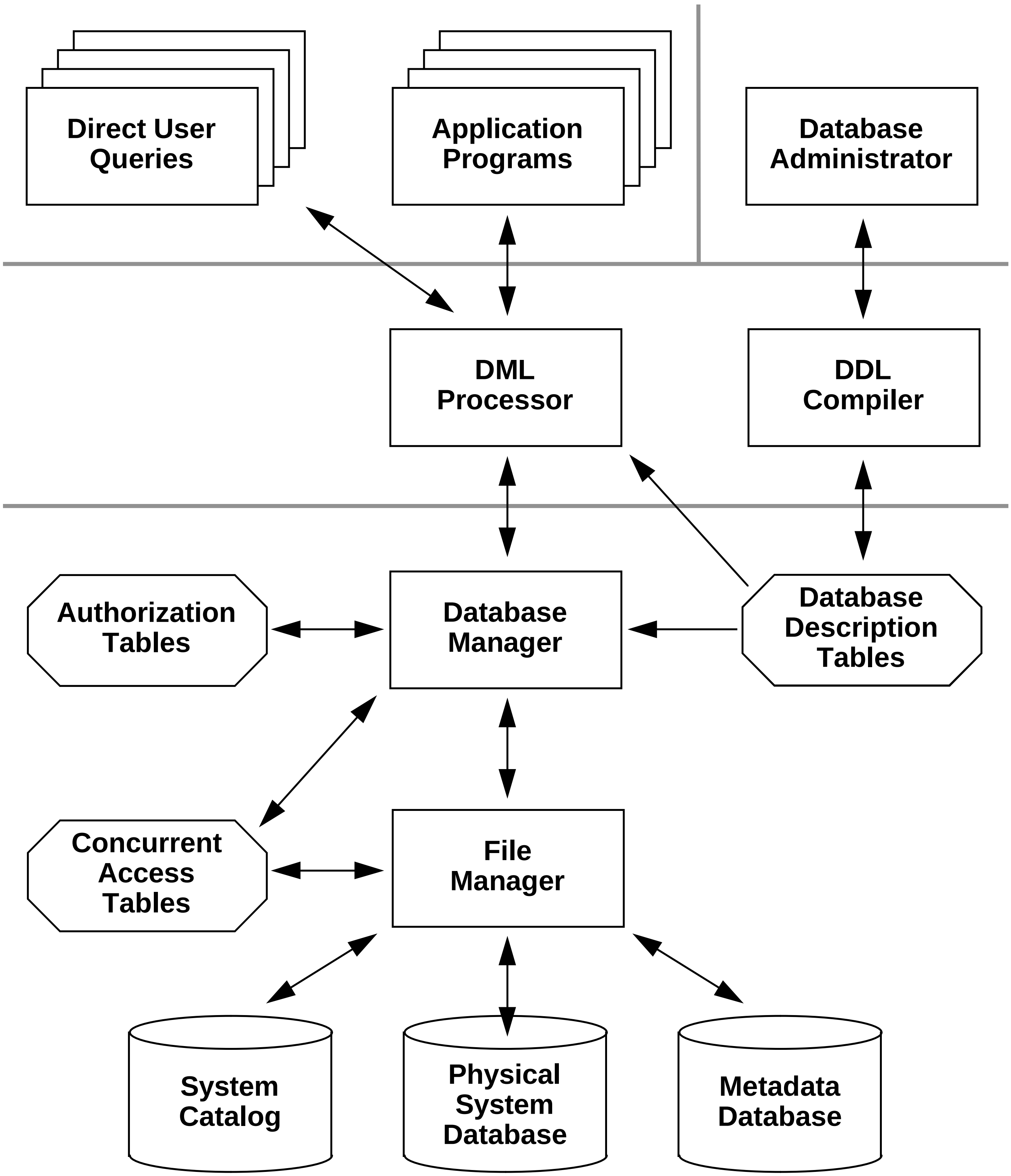
- **allow concurrency**
- **control security**
- **maintain data integrity**
- **provide for backup and recovery**
- **control redundancy**
- **allow data independence**
- **provide non-procedural query language**
- **perform automatic query optimization**

Who Interacts with a DBMS?

Many different individuals are involved with a database management system over its life:

- **systems analysts**
- **database designers**
- **database administrators**
- **application developers**
- **users**

Components of a Database System



Relational Database Model

What is a relational database?

- **a database that treats all of its data as a collection of relations**

What is a relation?

- **a kind of set**
- **a subset of a Cartesian product**
- **an unordered set of ordered tuples**

Basic Set Concepts

SET	**any collection of distinct entities of any sort.**
examples	A = { 1,2,3,4,5,6 } B = { H,T } C = { R,B } D = { Grant, Sherman, Lee }
CARTESIAN PRODUCT	**a set of ordered pairs, produced by combining each element of one set with each element of another set.**
example	B x C = { <H,R>,<H,B>,<T,R>,<T,B> }
Note:	Cartesian products may be generated by multiplying any number of sets together. The actual number of sets involved in a particular case is said to be the “*degree*” or “*arity*” of that Cartesian product.
RELATION	**a subset of a Cartesian product**
example	Q = { <H,R>,<H,B> }
Note:	Relations may be of any degree (arity).

Basic Set Concepts

A set is usually indicated by including a comma-delimited list of the names its members within a pair of wavy brackets:

```
R = { 1,2,3,4,5,6 }

G = { Marshall, Eisenhower, Bradley }
```

The members of a set are *unordered*. Two sets are considered equivalent if and only if they contain exactly the same members, without regard for the order in which the members are listed.

```
R = { 1,2,3,4,5,6 }

  = { 3,2,1,6,4,5 }


G = { Marshall, Eisenhower, Bradley }

  = { Bradley, Marshall, Eisenhower }
```

Basic Set Concepts

An *ordered* double (or triple or quadruple or n-tuple) is usually indicated by including a comma-delimited list of the names its members within a pair of pointed brackets:

```
S = < 2,4 >

C = < Marshall, Eisenhower, Bradley >
```

Order must be maintained in ordered n-tuples. Two tuples are considered different if they contain the same members in a different order.

```
S = < 2,4 > ≠ < 4,2 >

C = < Marshall, Eisenhower, Bradley >

  ≠ < Bradley, Eisenhower, Marshall >
```

A set may consist of an unordered collection of ordered tuples. For example, we could imagine the set of all ordered pairs of integers, such that the first element is the square root of the second element.

```
R = { <1,1>,< 2,4 >,<3,9> ... }
```

As this ellipsis indicates, sets can be infinite in size. However, sets that are actually represented in a database must be finite.

Basic Set Concepts

LET R be the set of possible outcomes when rolling a single red die.

R = { 1,2,3,4,5,6 }

LET B be the set of possible outcomes when rolling a single blue die.

B = { 1,2,3,4,5,6 }

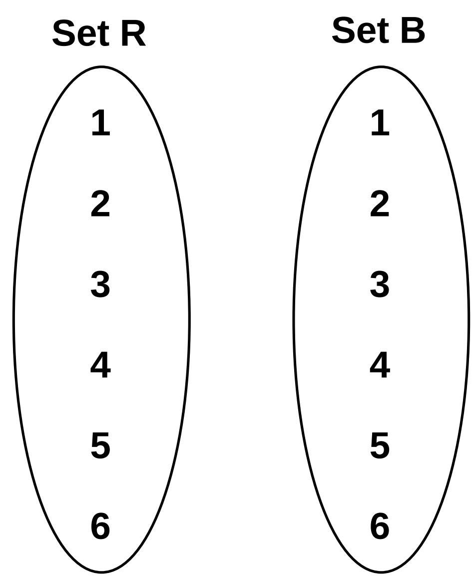
THEN The Cartesian product R x B gives the set of outcomes when the two dice are rolled together:

R x B: {

<1,1>	<3,1>	<5,1>
<1,2>	<3,2>	<5,2>
<1,3>	<3,3>	<5,3>
<1,4>	<3,4>	<5,4>
<1,5>	<3,5>	<5,5>
<1,6>	<3,6>	<5,6>
<2,1>	<4,1>	<6,1>
<2,2>	<4,2>	<6,2>
<2,3>	<4,3>	<6,3>
<2,4>	<4,4>	<6,4>
<2,5>	<4,5>	<6,5>
<2,6>	<4,6>	<6,6> }

Relation: Subset of a Cartesian Product

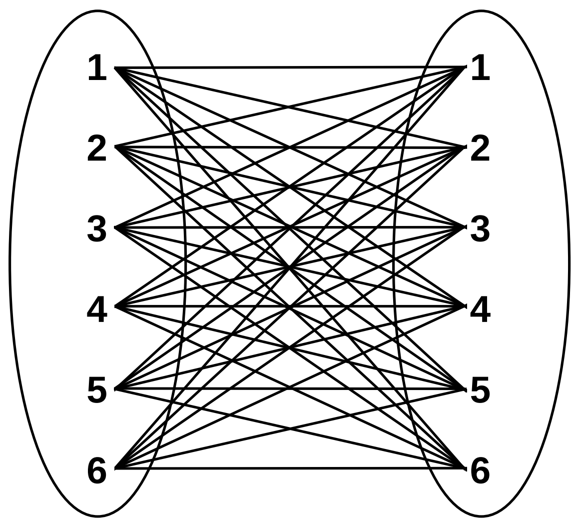
A Cartesian product of two sets can be generated by combining every member of one set with every member of the other set. This results in a complete set of ordered pairs, consisting of every possible combination of one member of the first set combined with one member of the second set. The number of elements in a Cartesian product is equal to M x N, where M and N give the number of members in each set.



Starting two sets.

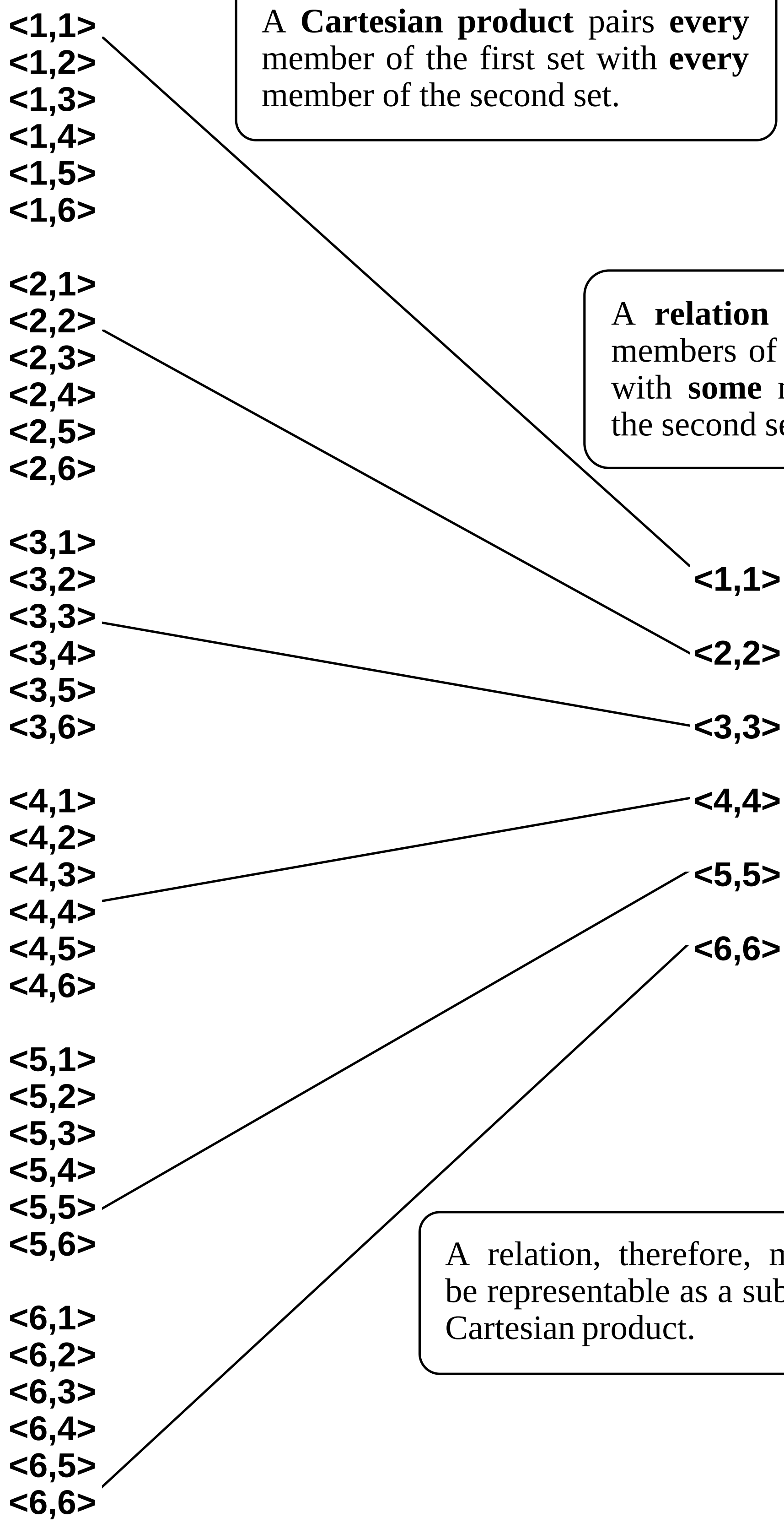
<1,1>	**<3,1>**	**<5,1>**
<1,2>	**<3,2>**	**<5,2>**
<1,3>	**<3,3>**	**<5,3>**
<1,4>	**<3,4>**	**<5,4>**
<1,5>	**<3,5>**	**<5,5>**
<1,6>	**<3,6>**	**<5,6>**
<2,1>	**<4,1>**	**<6,1>**
<2,2>	**<4,2>**	**<6,2>**
<2,3>	**<4,3>**	**<6,3>**
<2,4>	**<4,4>**	**<6,4>**
<2,5>	**<4,5>**	**<6,5>**
<2,6>	**<4,6>**	**<6,6>**

A Cartesian product of two sets, shown as a list of ordered pairs.



A Cartesian product of two sets, shown as a connection diagram, with each member of the first set connected to each member of the other set.

Relation: Subset of a Cartesian Product



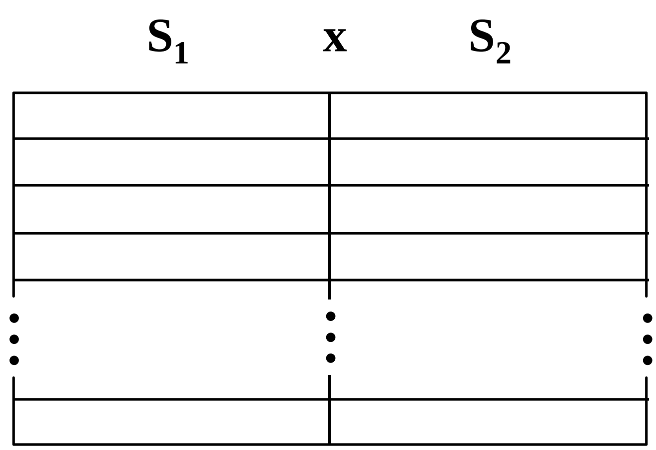
A **Cartesian product** pairs **every** member of the first set with **every** member of the second set.

A **relation** pairs **some** members of the first set with **some** members of the second set.

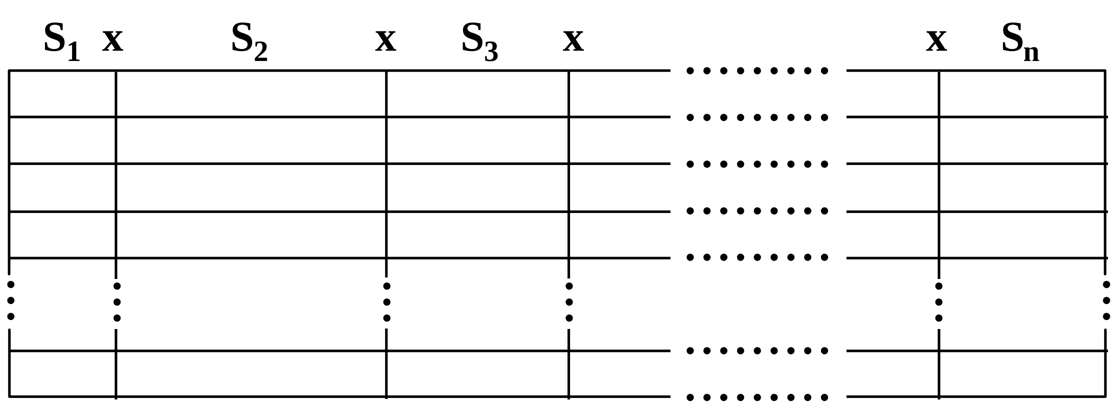
A relation, therefore, must always be representable as a subset of some Cartesian product.

Relation: Set of Ordered Tuples

A binary relation is a set of ordered doubles, with one element a member of the first set and one element a member of the second set. Generally, we could represent a set of ordered doubles as below. S_1 is the first set and S_2 the second.

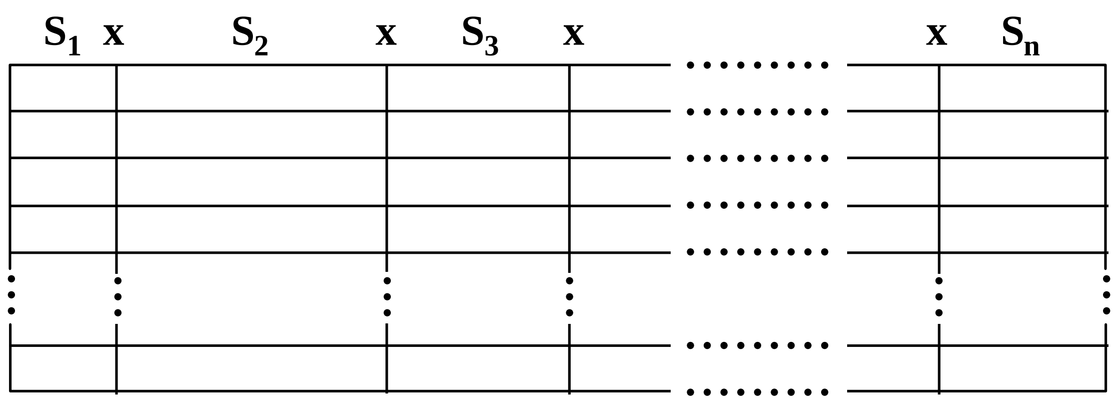


By adding sets, relations can be extended to include ordered triples, ordered quadruples or, in general, any ordered n-tuple, as below. A relation with n participating sets is said to be of **degree n** or to possess **arity n**.



Relations as a Database

An n-ary relation (i.e., a subset of a Cartesian product of n sets) could be be represented in a computer system as an n-column tabular file, with one member from the first set named in the first column of each record and one member of the second set in the second column, etc.

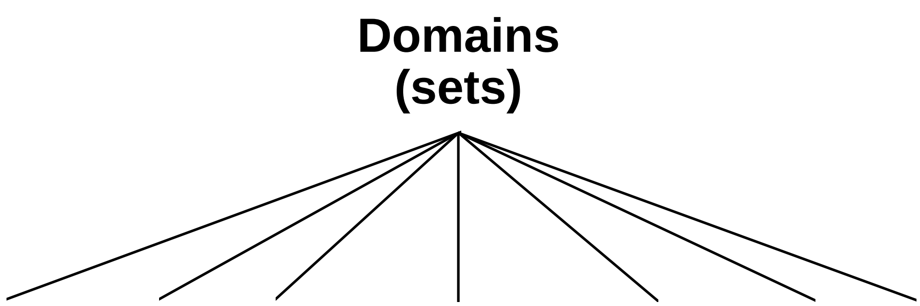


Codd recognized that many of the files used in computerized information systems were very similar in structured to tabularized relations.

Smith	Robert	L.	1154 Elm Street	Glendale	MD	21200
Smith	Judy	F.	1154 Elm Street	Glendale	MD	21200
Jones	Greg	G.	765 Cedar Lane	Towson	MD	21232
Harris	Lloyd	K.	2323 Maple Dr	Towson	MD	21232
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ziegler	Fred	K.	7272 Cherry Ln.	Baltimore	MD	21208

Relations as a Database

The business data file resembles a relation in a number of ways. The tabular file itself corresponds to a relation. Each column, or attribute, in the file corresponds to a particular set and all of the values from a particular column come from the same domain, or set. Each row, or record, in the file corresponds to a tuple



Name-L	Name-F	MI	address	city	state	zip
Smith	Robert	L.	1154 Elm Street	Glendale	MD	21200
Smith	Judy	F.	1154 Elm Street	Glendale	MD	21200
Jones	Greg	G.	765 Cedar Lane	Towson	MD	21232
Harris	Lloyd	K.	2323 Maple Dr	Towson	MD	21232
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ziegler	Fred	K.	7272 Cherry Ln.	Baltimore	MD	21208

If such a file is to be genuinely interchangeable with a relation, certain contraints must be met:

- every tuple must be unique
- every attribute within a tuple must be single-valued
- in in all tuples, the values for the same attribute must come from the same domain or set
- no attributes should be null

Relations as a Database

An essential attribute of a relation is that every tuple must be unique. This means that the values present in some individual attribute (or set of attributes) must always provide enough information to allow a unique identification of every tuple in the relation. In a relational database, these identifying values are known as **key values** or just as the **key**.

Sometimes more than one key could be defined for given table. For example, in the table below (which represents, perhaps, a patient record file), several columns might serve as a key. Either patient number (assigned by the hospital) or social security number (brought with the patient) are possibilities. In addition, one might argue that the combination of last name, address, and birth date could collectively serve as a key.

Any attribute or set of attributes that might possibly serve as a key is known as a **candidate key.** Keys that involve only one attribute are known as **simple keys**. Keys that involve more than one attribute are **composite keys**.

patient #	SS #	Last Name	address	birth date
P-64122	123-45-6789	Smith	123 Main Street	10 MAY 44
P-75642	001-32-6873	Pedersen	1700 Cedar Barn Way	31 MAR 59
P-70875	444-44-5555	Wilson	1321 North South St	7 AUG 90
P-79543	555-12-1212	Grant	808 Farragut Avenue	1 DEC 66
⋮	⋮	⋮	⋮	⋮
P-71536	888-88-8888	MacPherson	1617 Pennsylvania Ave	11 APR 60

In designing a database, one of the candidate keys for each relation must be chosen to be the **primary key** for that table. Choosing primary keys is a crucial task in database design. If keys need to be redesignated, the entire system may have to be redone. Primary keys can never be null and should never be changed. Sometimes none of the candidate keys for a relation are likely to remain stable over time. Then, an arbitrary identifier might be created to serve as a primary key. Such arbitrary keys are also known as **surrogate keys**.

Relations as a Database

A binary relation (i.e., a subset of a Cartesian product of two sets) could be be represented in a computer system as two-column tabular file, with one member from the first set named in the first column of each record and one member of the second set in the second column. For example, a binary relation could be used to provide unique three-letter identifiers for academic departments. Additional relations could be used to give more information about individual departments or individual faculty members.

ZOL	Zoology
PSD	Political Science
CPS	Computer Science
HIS	History
⋮	⋮
ACC	Accounting

ZOL	Zoology	Room 203	Natural Science Bldg	355 4640
CPS	Computer Science	Room 714A	Wells Hall	355 5210
BSP	Biological Science	Room 141	Natural Science Bldg	353 4610
CEM	Chemistry	Room 320	Chemistry Bldg	355 9175
⋮	⋮	⋮	⋮	⋮
PSD	Political Science	Room 303	South Kedzie Hall	355 6590

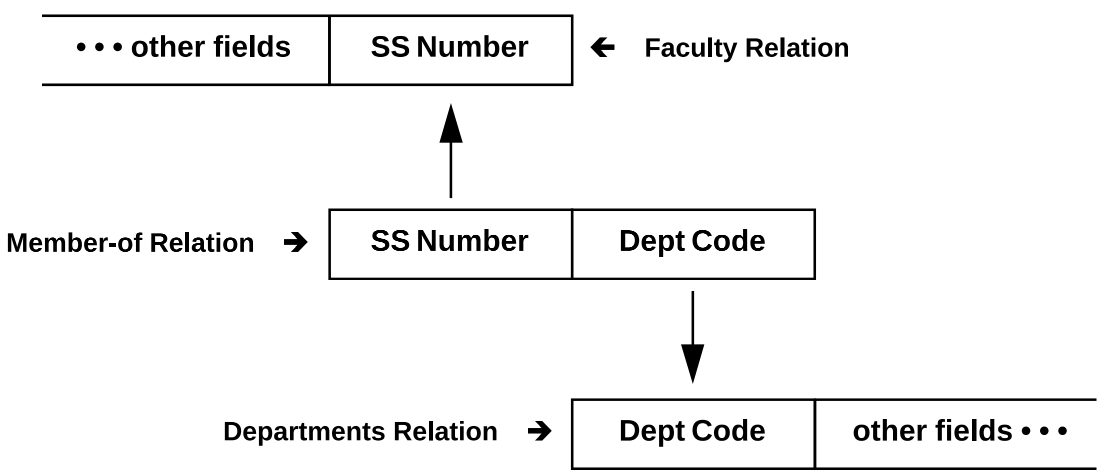
999-99-9999	Johnson	William	F.	1533 Raleigh Dr.	Baltimore	MD	21211
888-88-8888	Johnson	William	F.	2842 Colony Ave.	Baltimore	MD	21201
777-77-7777	Brown	James	G.	99 W. East St.	Towson	MD	21232
666-66-6666	Brown	Gwen	K.	99 W. East St.	Towson	MD	21232
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
111-11-1111	Ziegler	Samual	L.	7272 Cherry Ln.	Baltimore	MD	21208

Relations as a Database

Yet another relation could be used to show what faculty were members of what departments. Notice that faculty member 999-99-9999 is a member of more than one department and that, even on this short list, the department of zoology has two members given.

999-99-9999	ZOL
888-88-8888	PSD
7777-77-7777	CPS
666-66-6666	ZOL
⋮	⋮
999-99-9999	BSP

Relations of this sort, that combine identifiers from two other relations, provide the “glue” that holds a relational database together.



Whenever the values in an attribute column in one table “point to” primary keys in another (or the same) table, the attribute column is said to be a **foreign key**. Columns containing foreign keys are subject to an **integrity constraint**: any value present as a foreign key must also be present as a primary key.

Relational Database Operators

Data models consist of data structures and permitted operations on those data structures. Part of Codd's genius was to recognize that many of the standard set operators that can take relations as operands map nicely to real data manipulation problems:

- **Cartesian product**
- **union**
- **intersection**
- **difference**

Codd devised some additional operators to provide extra manipulatory power:

- **select**
- **project**
- **join**
- **divide**

The operators have now been extended to include more useful manipulations:

- **outer join**
- **outer union**

Relational Database Normal Forms

Considerable study has been made of the properties of relations as they affect the behavior of relational databases. The results of these studies are captured in the definition of **normal forms**.

First Normal Form:

- A relation is in first normal form (**1NF**) if and only if all underlying domains contain atomic values only.

Second Normal Form:

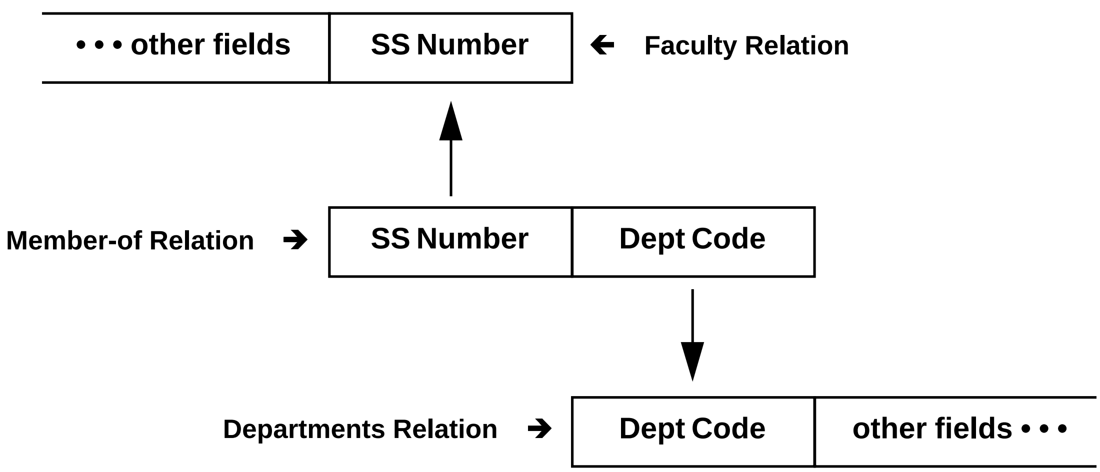
- A relation is in second normal form (**2NF)** if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key.

Third Normal Form:

- A relation is in third normal form (**3NF**) if and only if it is in 2 NF and the non-key attributes are mutually independent.

What is the E-R Data Model?

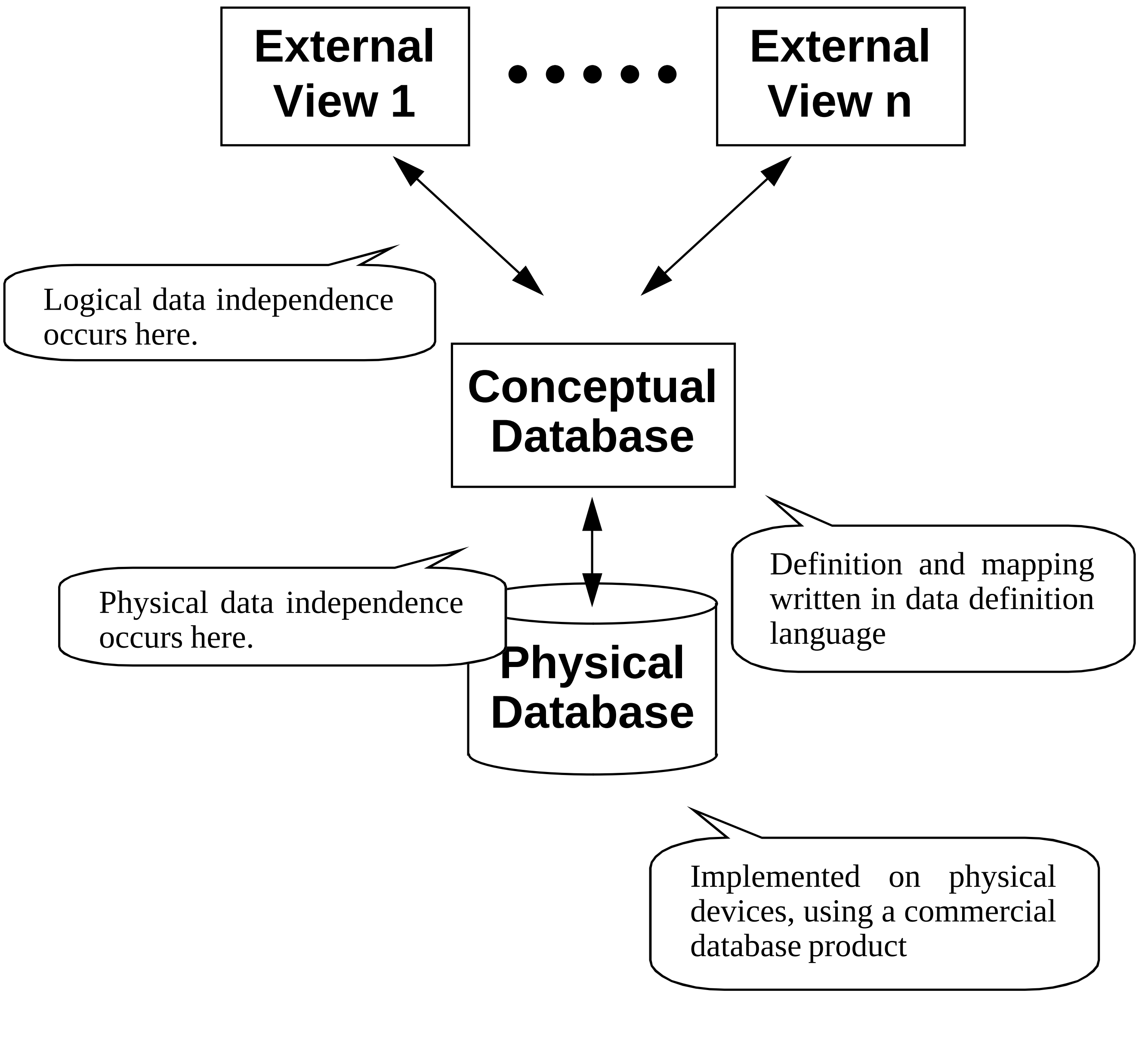
The Entity-Relationship (E-R) data model is a semantically rich model that can be mapped to a relational system.



The three files represented above are all relations in the formal sense. Chen (1976) noted that different relations may play different roles in a database and that being able to recognize and document those roles is a key part of database design. The "faculty" and the "department" relations above both store information about particular real-world entities. The "member-of" relation, on the other hand, stores information about specific relationships involving individual pairs of real-world entities.

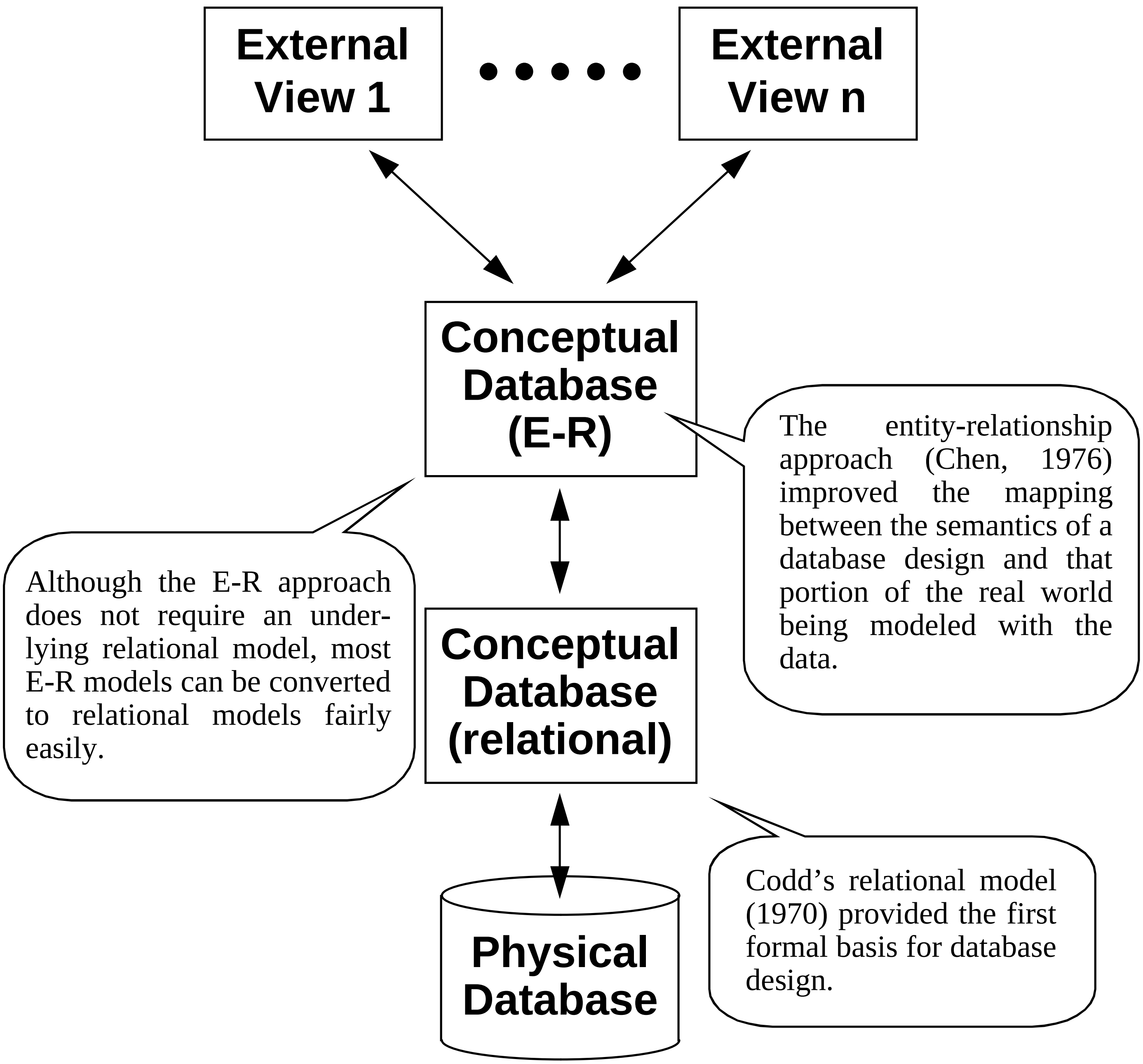
The E-R Data Model

Different needs for access and use of the database can be supported through different user views



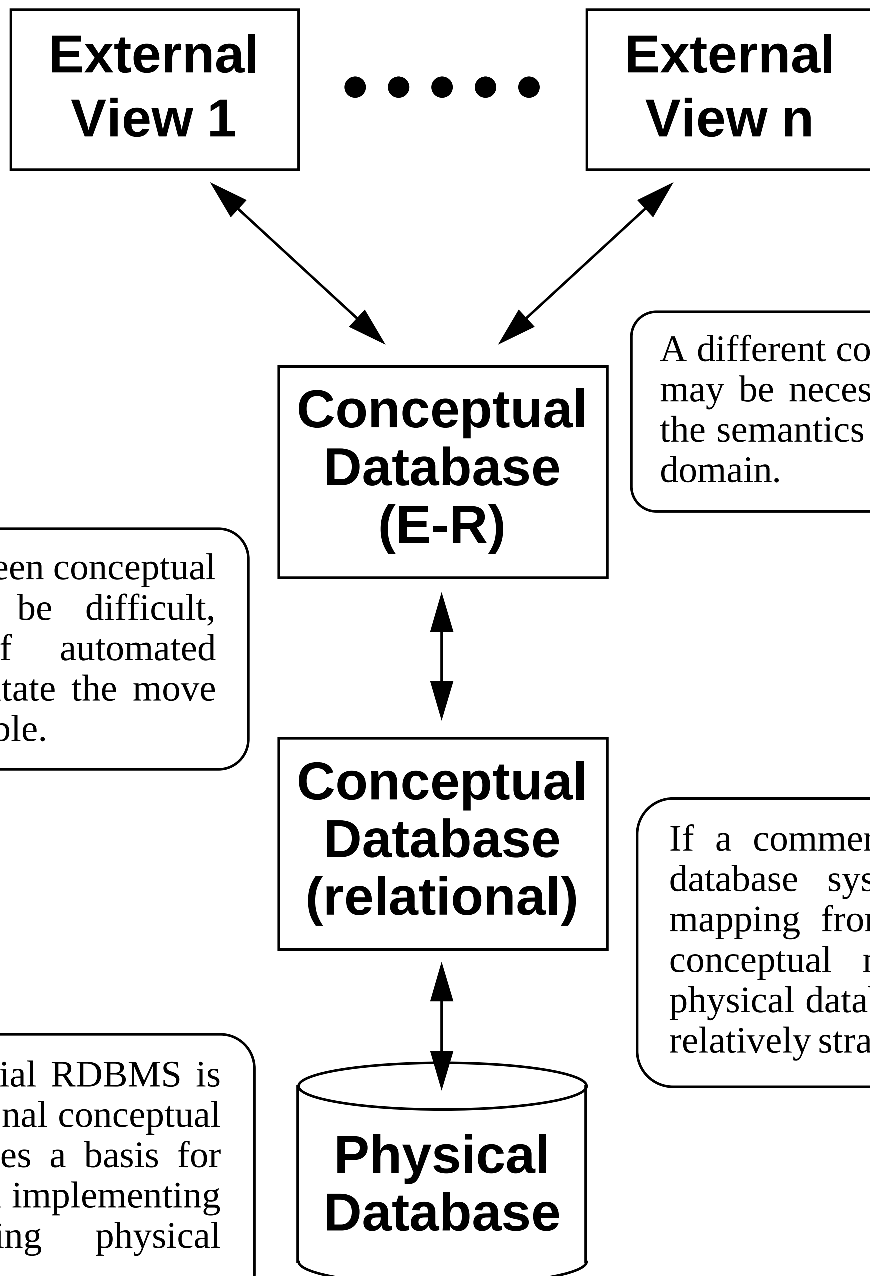
The E-R Data Model

Layers may be added to a conceptual design in order to increase the semantic richness available at the top design level.



The E-R Data Model

If layered conceptual models are used, the layering may be perceived differently by the system's users and developers. Users often see the database only in terms of the views that they employ. System analysts and designers may think primarily about the E-R schema, whereas the database administrator is likely to deal primarily with the relational schema and the physical system.



A different conceptual model may be necessary to capture the semantics of the database domain.

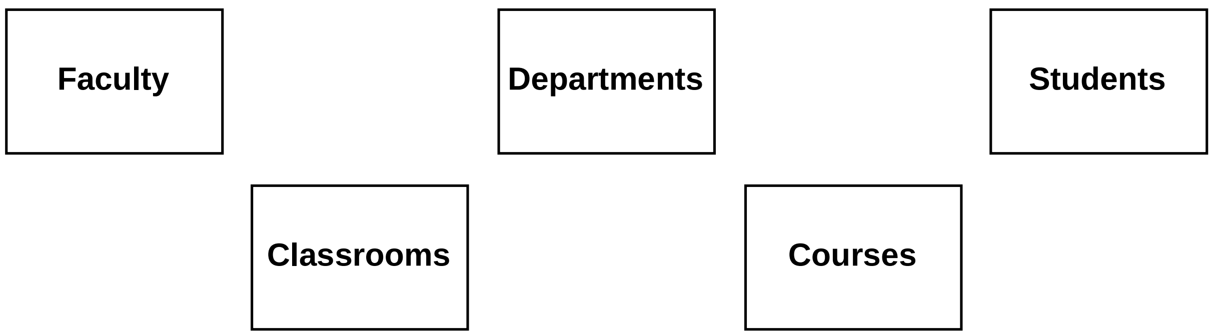
Moving between conceptual models can be difficult, especially if automated tools to facilitate the move are not available.

If a commercial relational database system is used, mapping from a relational conceptual model to the physical database should be relatively straightforward.

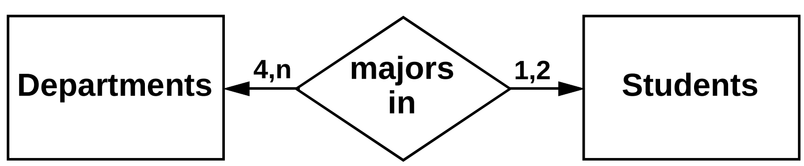
If a commercial RDBMS is used, a relational conceptual model provides a basis for designing and implementing an underlying physical database.

E-R Data Model: Graphical Conventions

Sets of real-world entities are represented with named rectangles:



Relationships between members of entity sets are represented with named diamonds that are connected to the rectangles of the participating entity sets with directed arcs:

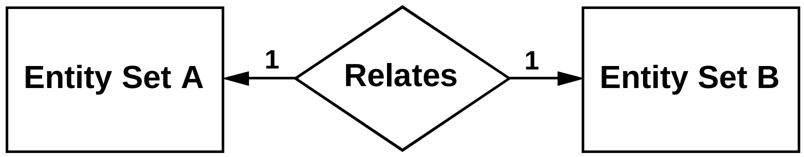


Arcs are drawn with an orientation that "points" from foreign keys to primary keys. The min:max **participation cardinality** can be indicated by placing pairs of numbers on each arc. Here, "4,n" means that every department is required to have at least four student majors, but can have many more; "1,2" means that each student is required to have at least one major and is permitted to have no more than two majors. Sometimes only the maximum participation cardinalities are shown.

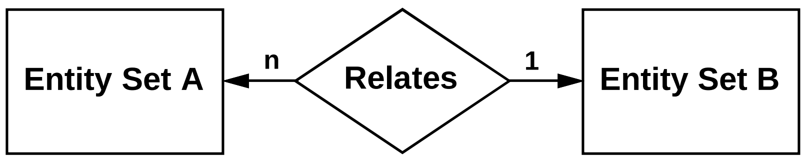
E-R Data Model: Graphical Conventions

Many different cardinalities are possible. Documenting the cardinalities is an essential part of database analysis and design.

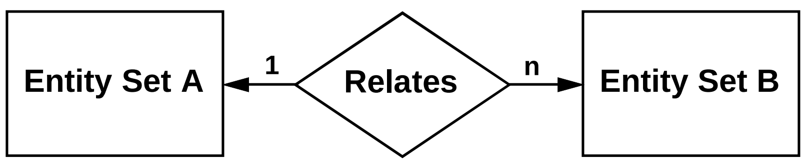
One-to-one:



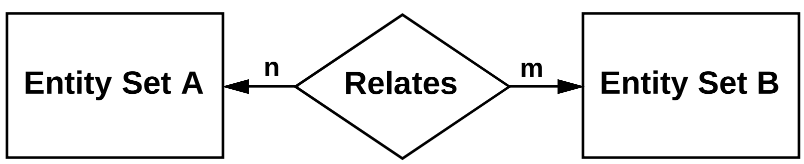
Many-to-one:



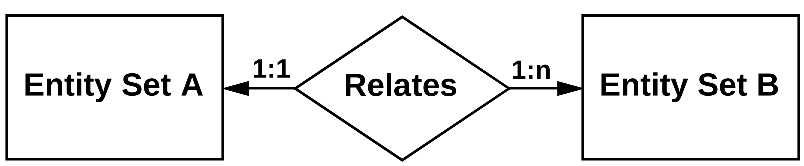
One-to-many:



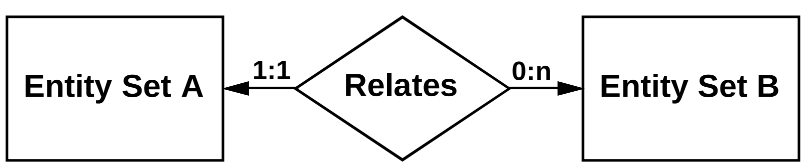
Many-to-many:



One-to-many:
(mandatory)

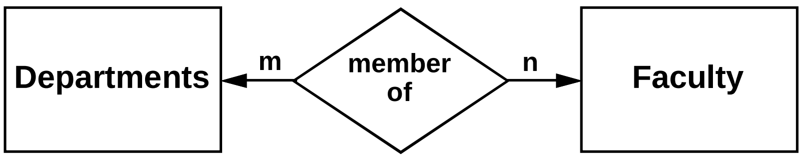


One-to-many:
(optional)

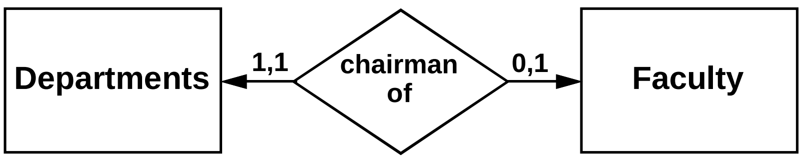


E-R Data Model: Examples

Faculty and departments entities could be related by a many-to-many "member-of" relationship:



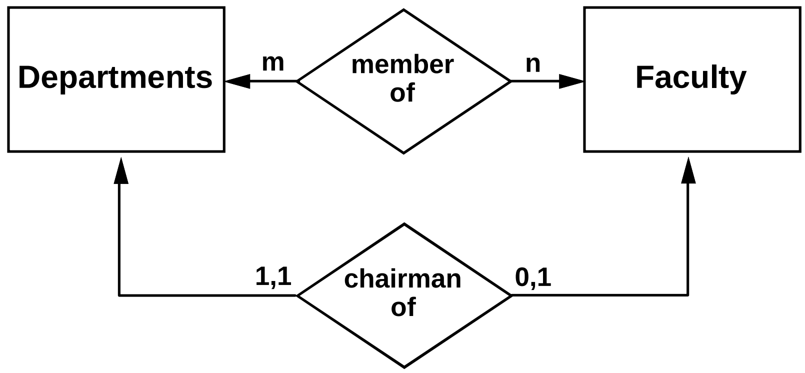
They could also be related by a one-to-one "chairman-of" relationship:



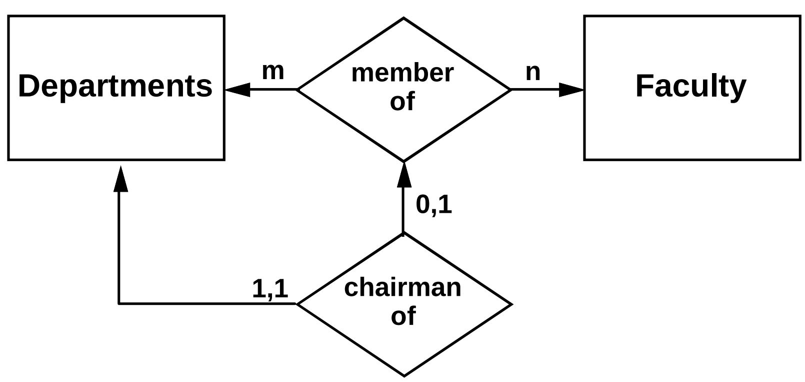
The "1,1" cardinality for departments means that every department must have one and only one chairman. The "0,1" cardinality for faculty means that not all faculty participate in the chairman-of relationship and that no faculty member may participate more than once. That is, not all faculty are chairmen and no one faculty member may serve as chairman of more than one department.

E-R Data Model: Graphical Conventions

Combining these two relationships into a single diagram, we would have:

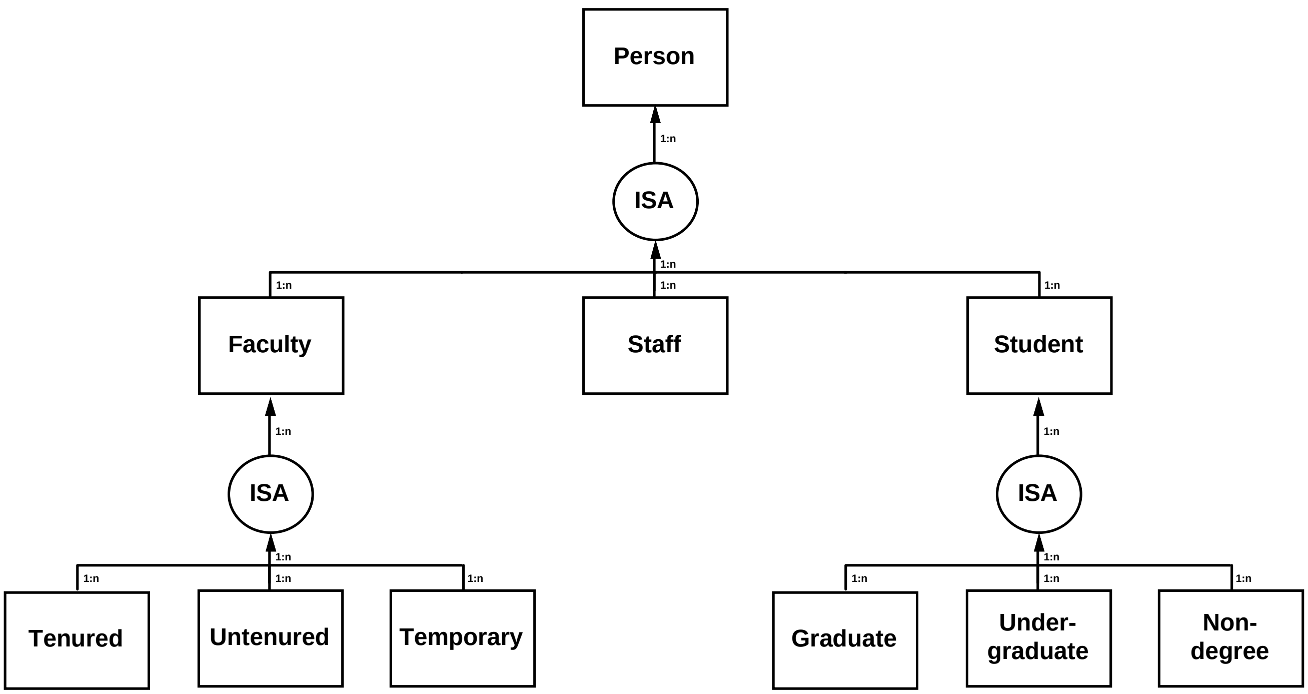


A database design derived from the figure above would allow a faculty member to chair a department of which he/she was not a member. To indicate an integrity constraint that requires membership in a department in order to chair the department, the E-R diagram would be modified as below:



E-R Data Model: Graphical Conventions

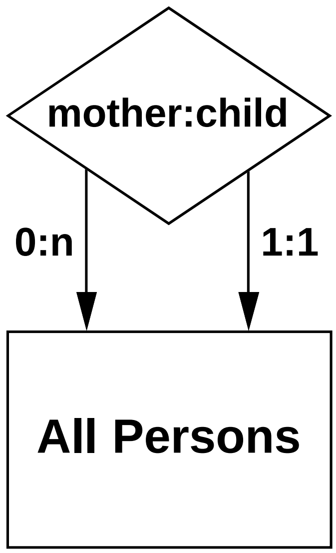
Class hierarchies ("ISA" hierarchies) can be indicated as below:



E-R Data Model: Graphical Conventions

Relationships may be recursive. Here, this E-R figure represents all possible mother-child relationships among all humans.

This cardinality indicates that not all persons participate in the relationship as mothers, but that those who do participate may participate one or more times.



This cardinality indicates that ***all*** persons participate in the relationship as child and that no child may have more than one mother.

Recursive relationships are particularly useful for representing any data structure that could also be represented as a directed graph. Entries in the entity table represent nodes of the graph and entries in the relationship table represent arcs.