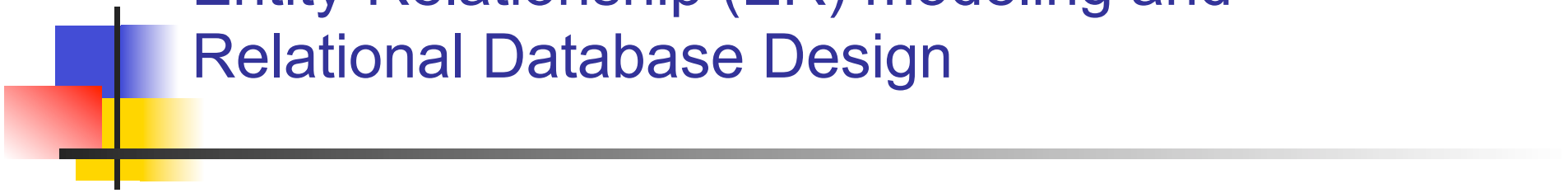


Entity-Relationship (ER) modeling and Relational Database Design

Fernando J. Pineda

140.636

Programmers-For-Hire Inc: A one-table database

MID	Name	Skills	Firm	Loc
1	Joh Smith	Access, DB2, FoxPro	ABC	AL
2	Dave Jones	dBASE, Clipper	MCI	FL
3	Mike Beach		IBM	DE
4	Jerry Miller	DB2, Oracle	MCI	FL
5	Ben Stuart	Oracle, Sybase	AIC	NE
6	Fred Flint	Informix	ABC	AL
7	Joe Blow		RUN	IA
8	Greg Brown	Access, MS SQLserver	XYZ	NY
9	Doug Hope		IBM	DE

Problems due to bad database design

- Some queries require special programming
  - The Skills attribute has more than one value in each row
  - We need to *parse* the values of the skills attribute when we search for a particular skill
- If we insert a new employee at IBM, but incorrectly enter MD as the location, then how do we determine where IBM actually resides? (***insertion anomaly***)
- If Greg Brown quits, and we delete him from the table, we lose the fact that XYZ was in our organization. (***deletion anomaly***)
- If the IBM office moves from DE to MA, we need to scan every row and update the IBM entries. If we miss just one, we don't know where IBM is located (***update anomaly***)

Data Modeling

- The purpose of data modeling is to impose a formal structure on data. The formal structure is known as a **data model**.
- Most databases support only one data model.
- The **relational data model** is just one such representation.
- An **entity-relationship diagram** (**ER diagram**) is a DBMS-independent technique for representing the structure of a relational data model.

ER modeling terminology

- **Entity: a table**
  - Something about which we store data, e.g. a protein, a customer, a citation, etc..
- **Attribute: a column in a table**
  - Entities are characterized by their attributes, e.g. a customer entity is described by a customer_id, first name, last name, street, city street, zipcode and telephone number.
- **Instances (of an entity): a row in a table**
  - Entities in a relational database are *represented* by a row of attributes.
  - A row of attributes is an instance of an entity.
  - Often we simply say *entity* when we mean *instance of an entity*.

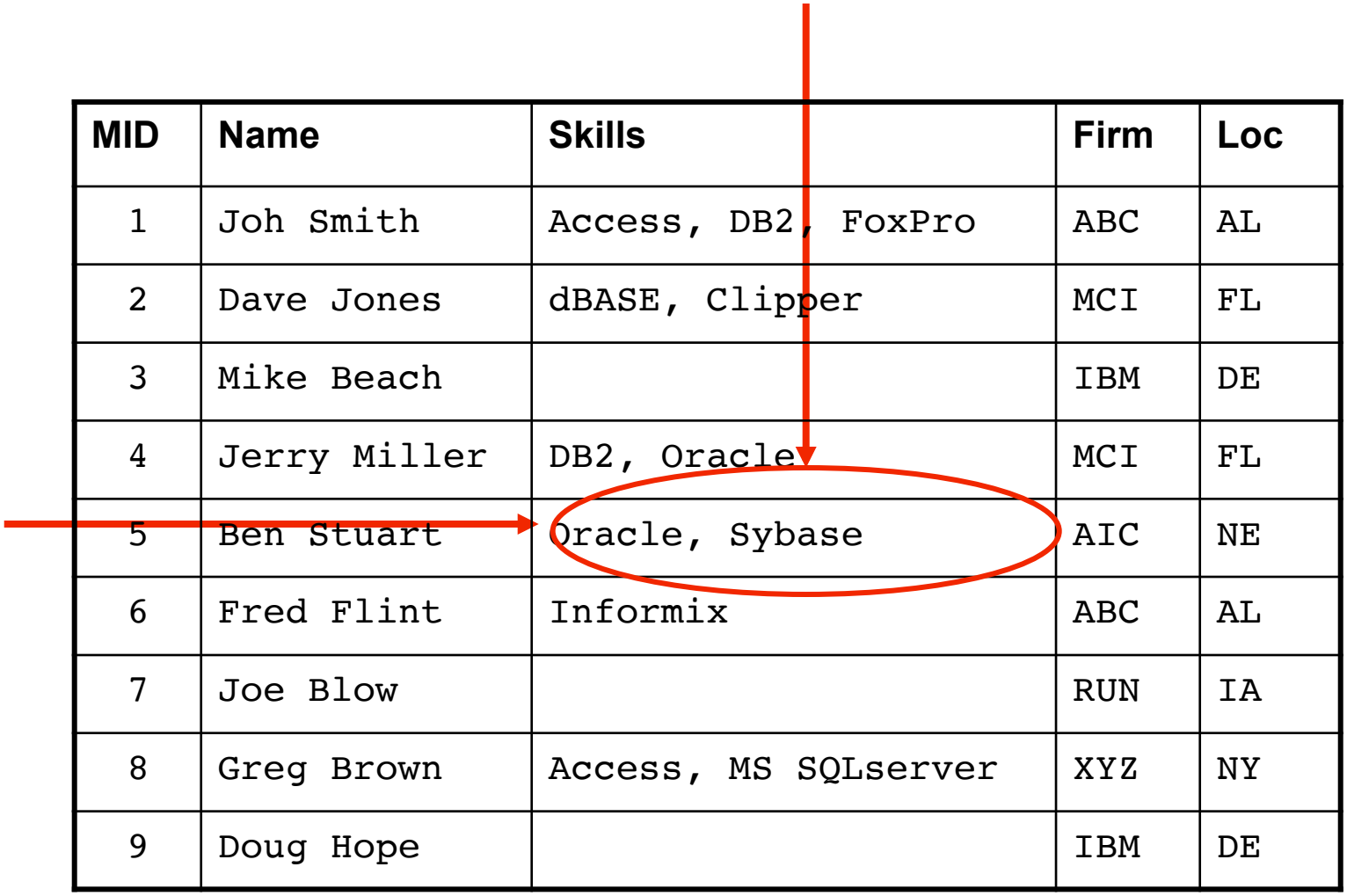
Entity Identifiers

- We put data in a database so we can ultimately retrieve it!
- To retrieve specific data we need a means of distinguishing one instance of an entity from another instance.
- Each instance of an entity must have at least one unique attribute (the **entity identifier**)

Entity Identifiers & keys

- Superkeys
  - A set of columns in a database that together serve to uniquely identify a row
- key
  - A minimal superkey
- foreign keys
  - a column in a table that is the key of another table (used for joining tables together).

Entity+identifer+attribute uniquely retrieves entity data



MID	Name	Skills	Firm	Loc
1	Joh Smith	Access, DB2, FoxPro	ABC	AL
2	Dave Jones	dBASE, Clipper	MCI	FL
3	Mike Beach		IBM	DE
4	Jerry Miller	DB2, Oracle	MCI	FL
5	Ben Stuart	Oracle, Sybase	AIC	NE
6	Fred Flint	Informix	ABC	AL
7	Joe Blow		RUN	IA
8	Greg Brown	Access, MS SQLserver	XYZ	NY
9	Doug Hope		IBM	DE

Choosing entity identifiers

- Bad ways of identifying instances (e.g. of people)
  - Social security number
  - Phone_number
  - First_name+Last_name+social
- Best practice,
  - unique meaningless numbers make the best entity identifiers.
  - Sometimes there is a natural or previously assigned meaningless number, e.g. an order number.
  - Sometimes concatenated entity identifiers are appropriate
  - AUTO_INCREMENT in MySQL

Single vs multi-valued attributes

ID	Parent	children	Birthdays
1	Fernando	Katy, Daniel	12/21/93 03/26/95
2	Rosie	Lucky,Two,Yeller, Little-Squirt,Furzy	01/06/2004

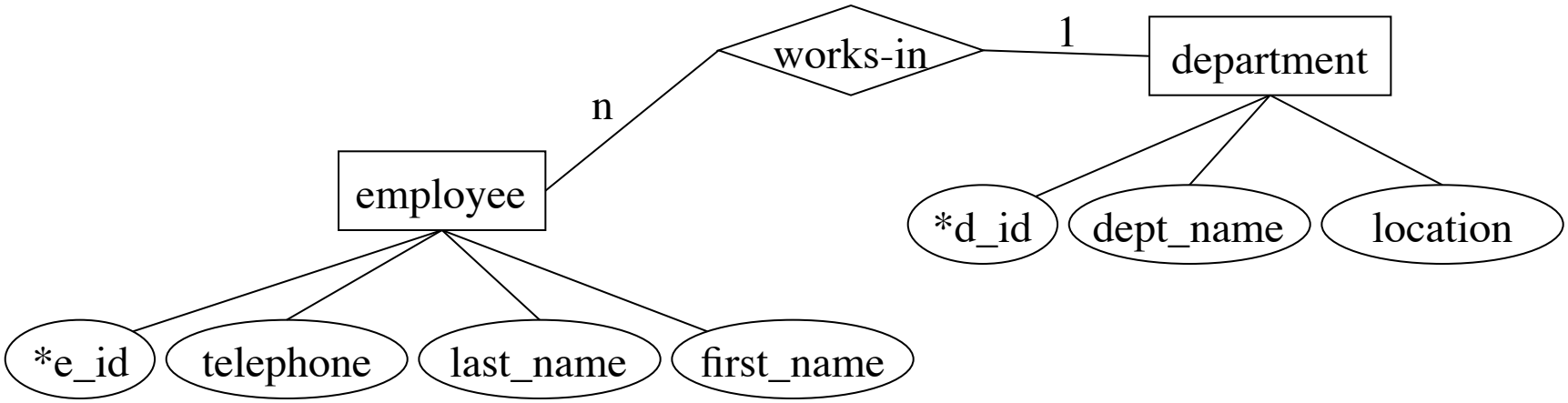
- Multi-valued attributes cause problems:
  - What is meaning of the data?
  - Slows down searching
  - How many values can be stored? Unnecessarily restricts amount of data that can be stored.
- In general, if you encounter multi-valued data, it's a hint that it's time to create a new instance (row).

ER diagram Styles

- Special case of Associative Networks
  - http://web.cs.mun.ca/~ulf/pld/assoc.html
- Chen Style (Classical ER modeling approach)
  - Chen P. "The Entity-Relationship Model-Toward a Unified view of Data" ACM Trans on Database systems vol. 1 ,no. 1 Mar 1976)
- Information Engineering Style (Classical ER modeling approach)
  - Martin, J. and McClure, C., *Diagramming Techniques for Analysts and Programmers* (publ: Prentice Hall 1985)
  - C. Finkelstein: *An introduction to information engineering: From strategic planning to information systems*. Addison Wesley (1989).
  - http://www.inconcept.com/JCM/April2000/halpin.html
- Object oriented Styles (Modern approaches)
  - UML -- essentially ER + methods
  - ORM

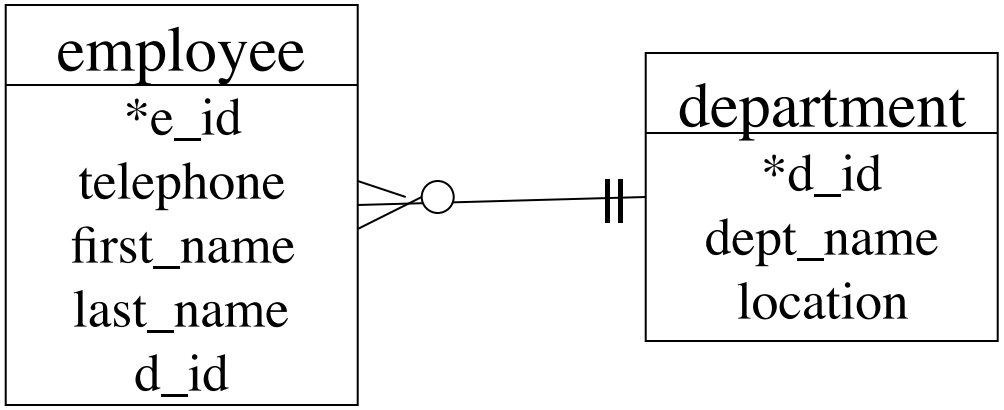
Chen Style

- **Box** Entity
- **Ellipse/Rounded box** Attribute
- **Asterisk** Entity instance identifier
- **Arrow/line** Relationship
- **Diamond** makes relationships explicit



Information Engineering Style

- **Box** Entity
  - Entity name labels the box
  - Attributes are in the lower section of the box
  - Instance identifier has an asterisk
- **Line** Indicates Relationship
- **Terminators** Indicate multiplicity of the relationship
  - || One and only one
  - 0| Zero or one
  - >| One or more
  - >0 Zero, one or more



Domains

- Each attribute has a **domain** which is the set of values the attribute is allowed to take.
- A domain can be small, e.g.
  - TRUE, FALSE
  - Dates
  - zip codes (different from integers since they can start with zero)
- A DBMS enforces a domain via **Domain constraints**, e.g.
  - Date domain constraints reject illegal dates
  - Time domain constraints enforce 24 hours, 60 minutes and 60 seconds.
- Practical domain constraints
  - Use **data types** allowed by particular DBMS

Examples of Practical Domains (e.g. in MySQL)

- **CHAR:** Fixed-length string text, often up to 256 chars
- **VARCHAR:** Variable-length string, often up to 256 chars
- **INT:** Integer, whose size depends on OS
- **DECIMAL(m,n):** decimal with m characters and n digits to the right of the decimal point
- **DATE:** a date
- **TIME:** a time
- **DATETIME**: The combination of a date and time
- **BOOLEAN**: a logical value (TRUE, FALSE)
- **BLOB**: Binary Large Object for anything binary

See the RDBMS documentation for other data types

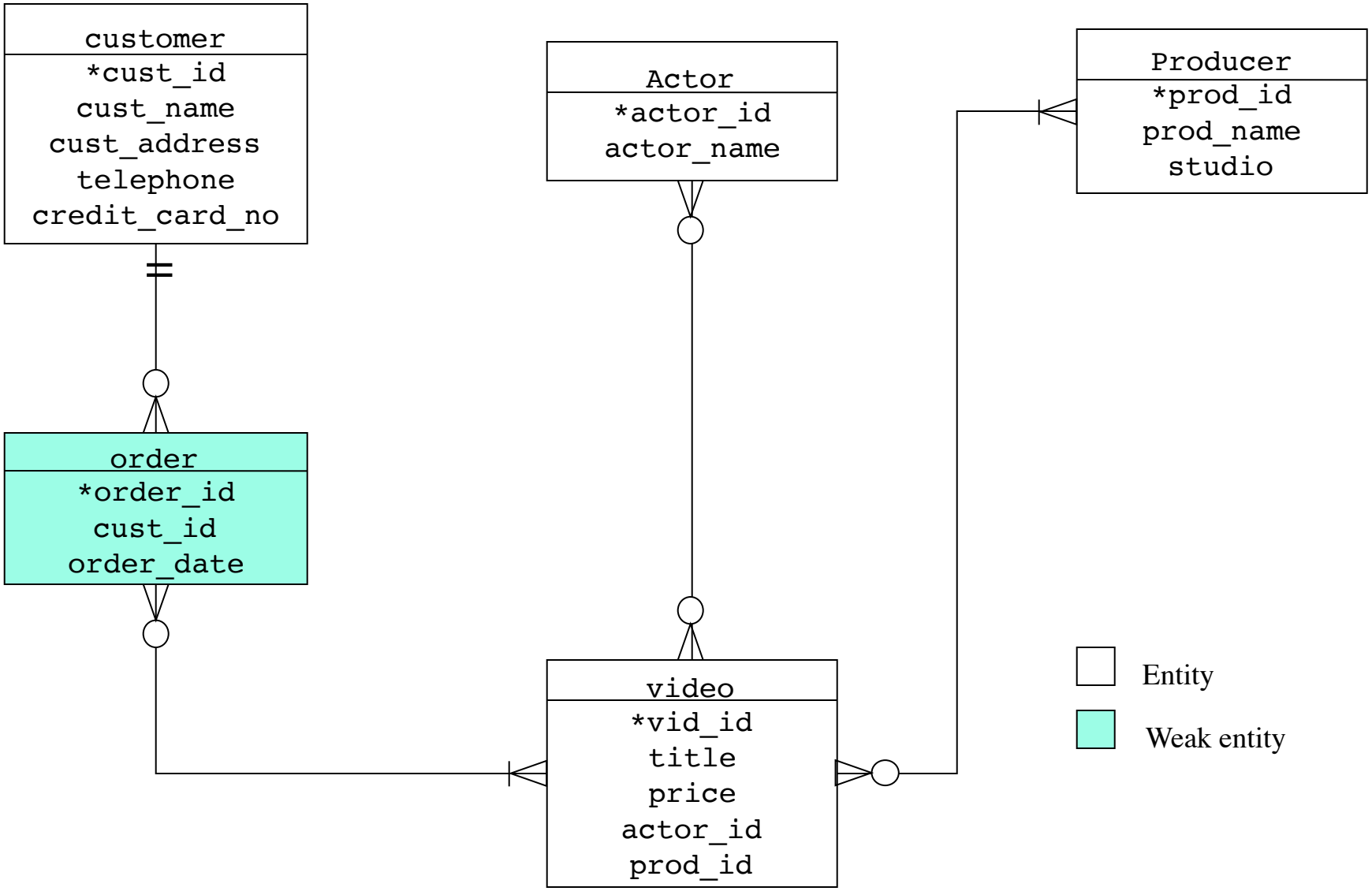
Basic Data Relationships

- Three basic types of relationships
  - **one-to-one** (e.g. people and social security numbers)
  - **one-to-many** (e.g.species and tissues)
  - **many-to-many** (e.g. students and classes)
- Relationships are between particular instances of entities
  - ER diagrams show *possible* relationships between instances.
  - No requirement that every instance of every entity have every relationship.

Schemas

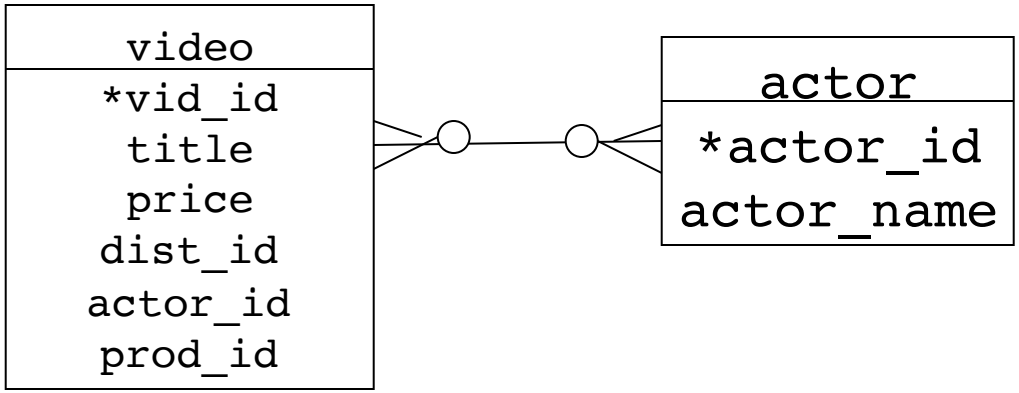
- A complete entity-relation (ER) diagram represents the overall logical plan of a relational database.
- The relational database schema specifies how to implement the ER diagram in a real database.

Video store (first draft)



Many-to-many relationship

- Actors can be in zero or more videos and a given video can have zero or more actors



Many-to-many relationships are problematic

- Actors can be in more than one video and a given video can have multiple actors.
- One (terrible) way to handle this is to use multiple-valued attributes?

Actor

actor_id	actor_name
30	Billy Crystal
27	Meg Ryan
1077	Tom Hanks

Video

vid_id	title	vid_actors
10578	When Harry met Sally	301,27
2901	Sleepless in Seatle	1077, 27

But we know that multivalued attributes cause problems

Many-to-many relationships are problematic

- We could eliminate multi-valued attributes by creating a new row for each actor in the video table.

Actor

actor_id	actor_name
30	Billy Crystal
27	Meg Ryan
1077	Tom Hanks

Video

vid_id	title	vid_actors
10578	When Harry met Sally	301
10578	When Harry met Sally	27
2901	Sleepless in Seatle	1077
2901	Sleepless in Seatle	27

But then our vid_id is not unique

Many-to-many relationships are problematic

- We could create a unique key in the table by combining vid_id and actor_id into a primary key

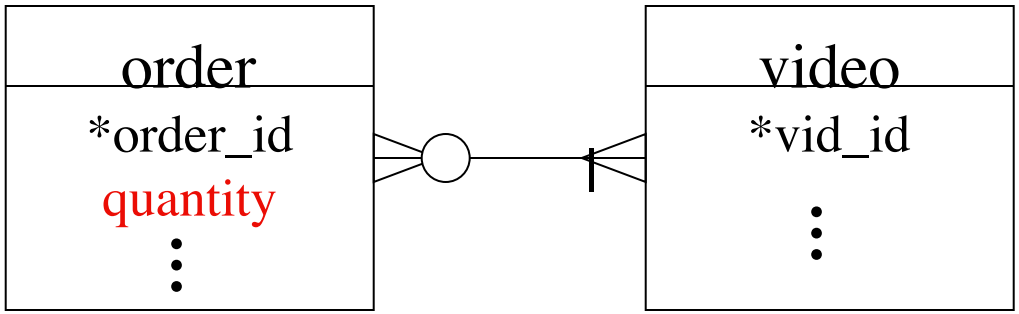
Video

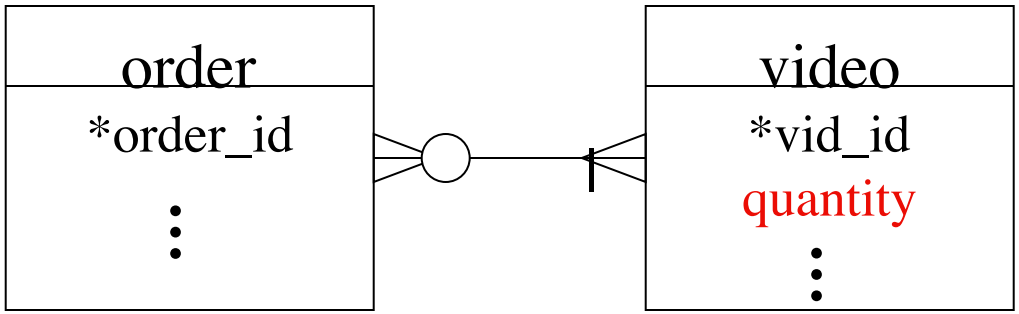
vid_id	title	actor_id
10578	When Harry met Sally	301
10578	When Harry met Sally	27
2901	Sleepless in Seatle	1077
2901	Sleepless in Seatle	27

But now we have lost videos as an entity

Many-to-many attributes are problematic

- A more subtle problem.
- An order can have many videos and a video can be in many orders.
- Customers can order more than one copy of a video.
- Where do we put the *quantity* attribute?

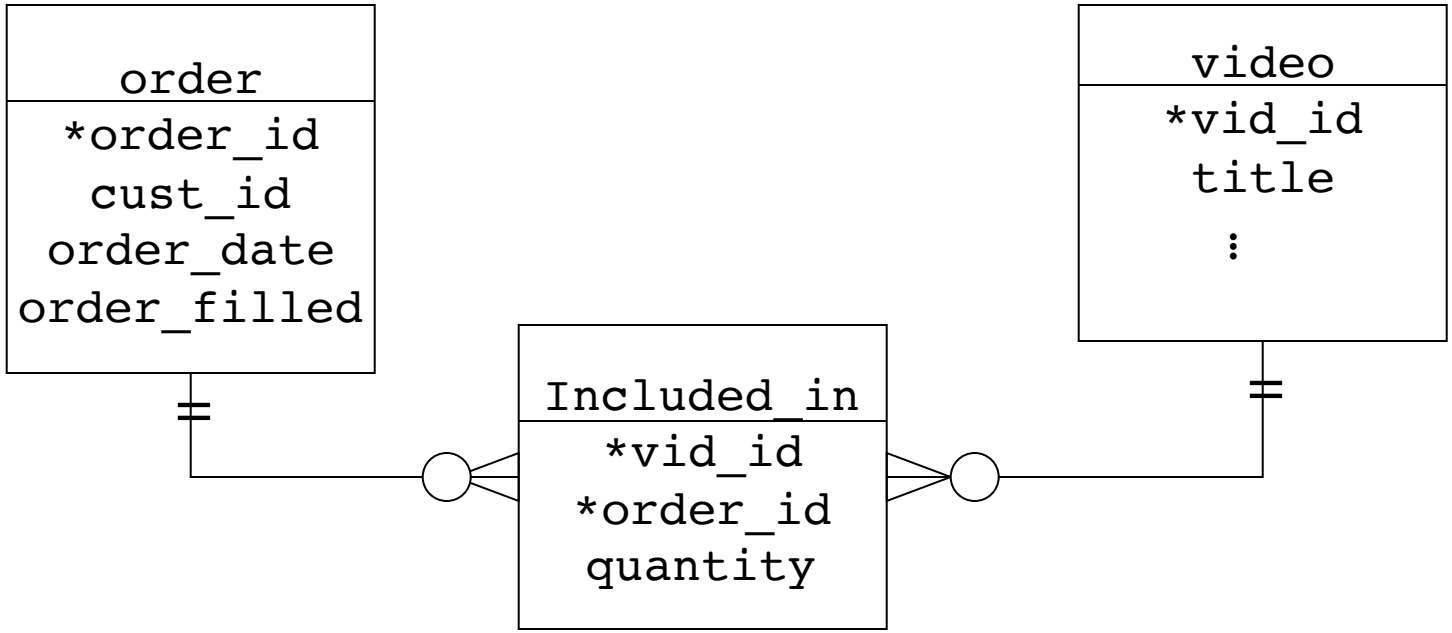




- Neither is correct since quantity applies to the relationship between the entities rather than either of the entities.
- quantity is an example of **relationship data**

Eliminating many-to-many relations with *relationship entities (composite entities)*

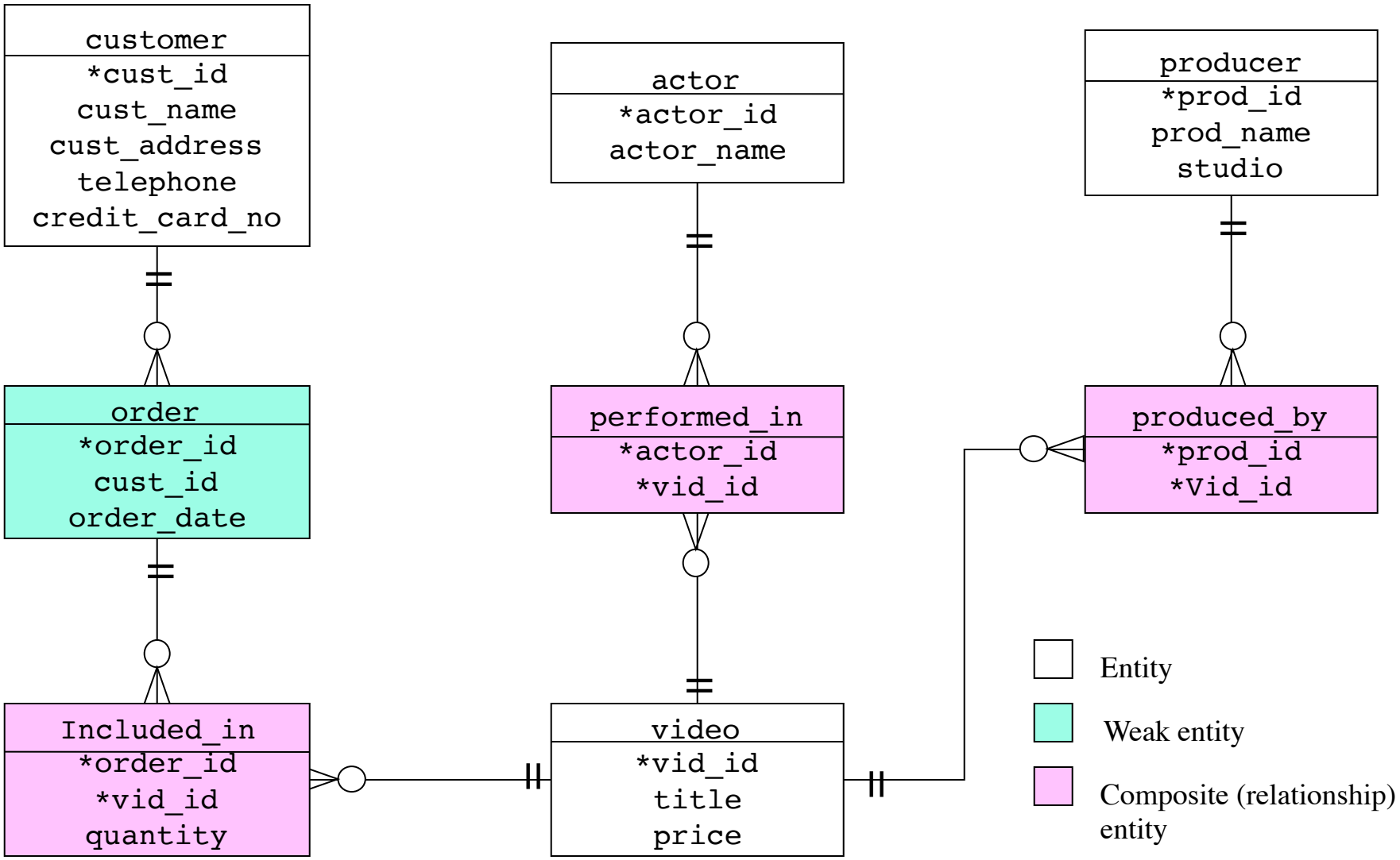
- Composite entities are entities that represent relationship data



- A single many-to-many relationship between order and video entities is replaced by two one-to-many relationships
- Note that the the identifier for the composite entity is composed by concatenating the entity identifiers of the two related entities

```
primary  key(vid_id,order_id)
```

Video store (second draft)



SQL statements (entities)

```
create table customer (
    customer_id           int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    customer_name         varchar(64),
    customer_address      varchar(64),
    customer_telephone    char(12),
    customer_credit_card  char(20),
) engine=InnoDB;

create table actor (
    actor_id     int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    actor_name    varchar(64)
) engine=InnoDB;

create table producer (
    prod_id     int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    prod_name     varchar(64)
) engine=InnoDB;

create table video (
    vid_id              int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    title               varchar(64),
    price               decimal(6,2)
) engine=InnoDB;
```

SQL statememements (weak entity)

```
create table order (
    order_id            int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    customer_id         int,
    order_date          DATE,
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
) engine=InnoDB;
```

Composite/Relationship (entities)

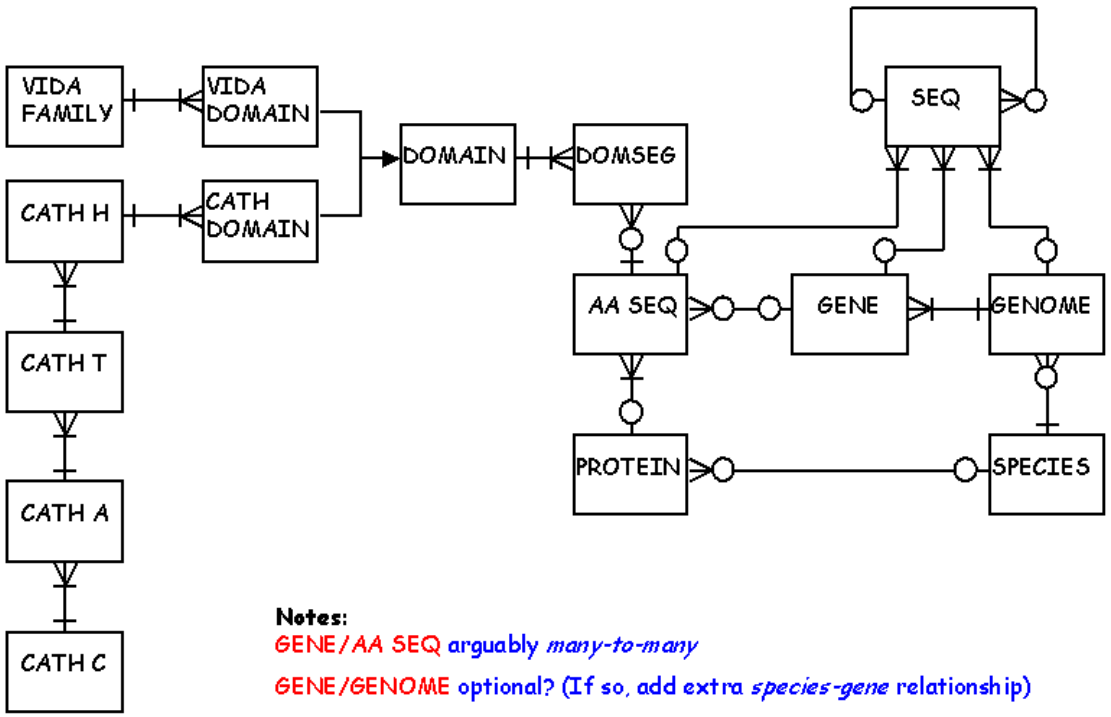
```
create table performed_in (
    vid_id       int NOT NULL,
    actor_id     int NOT NULL,
    FOREIGN KEY(actor_id) REFERENCES actor(actor_id),
    FOREIGN KEY(vid_id)   REFERENCES video(vid_id),
    PRIMARY KEY(actor_id, vid_id)
) engine= InnoDB;

create table produced_by (
    vid_id      int NOT NULL,
    prod_id     int NOT NULL,
    FOREIGN KEY(prod_id) REFERENCES producer(prod_id),
    FOREIGN KEY(vid_id)  REFERENCES video(vid_id),
    PRIMARY KEY(prod_id, vid_id)
) engine= InnoDB;

create table included_in(
    order_id           int NOT NULL,
    vid_id             int NOT NULL,
    quantity           int,
    FOREIGN KEY (order_id) REFERENCES order(order_id),
    FOREIGN KEY (vid_id)   REFERENCES video(vid_id),
    PRIMARY KEY (order_id,vid_id)
) engine=InnoDB;
```

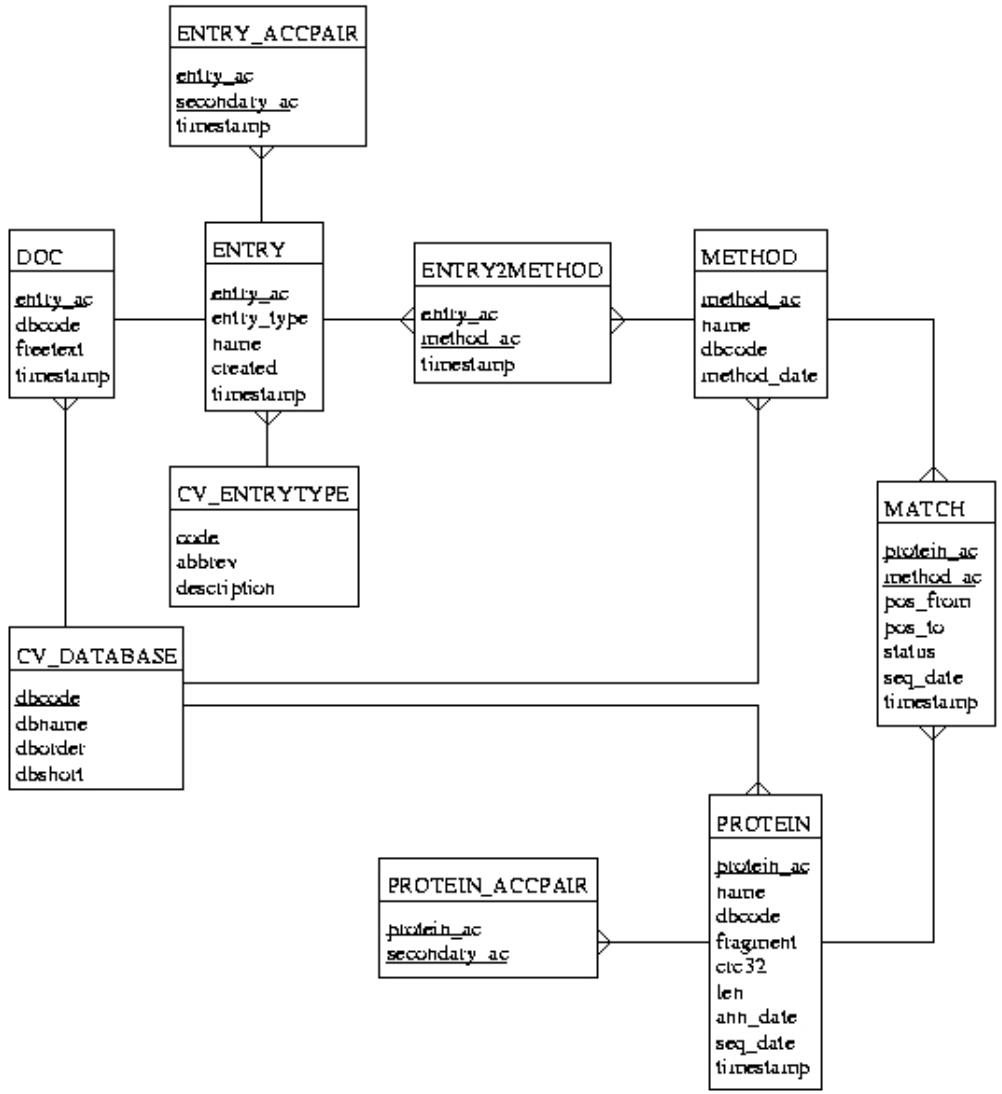
Example schemas

Plasmodium falciparum DB ER diagram



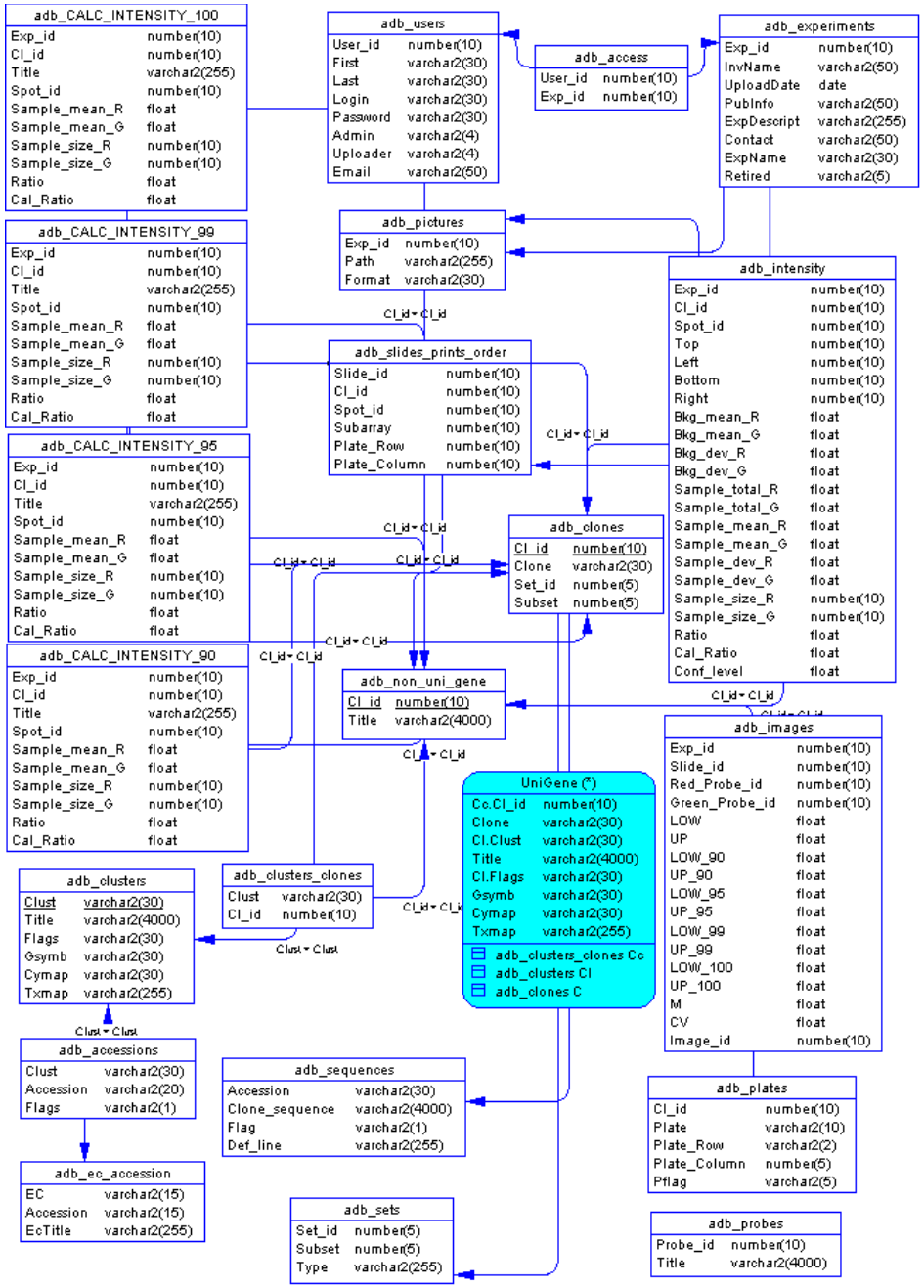
Swiss-Prot InterPro database schema

http://www.ebi.ac.uk/swissprot/Publications/mbd2.html



Protein families, domains and functional sites

Array Express Database Schema



End