Mapping ER Diagrams to a relational database.

Binary relationships:

Three possible configurations:
1. One table..
2. Two tables..
3. Three tables..

1-1 relationships $R(AB) - S(CD)$

Typical relationship between entities is $((a,b),(c,d))$ is best represented by one table $RS(a,b,c,d)$

$n$-1 (or 1-$n$)

Relationship between entities looks like
$((a, b), (c1, d1))$
$((a, b), (c2, d2))$

If we were to represent this a single table of the form $RS(ABCD)$, the preceding relationship would have to be represented by 2 rows:
(a, b, c1, d1)
(a, b, c2, d2)
The correct solution is to use 2 tables, include the primary key of the "one" as an attribute in the "many"

R(A, B) and S(C,D,A)...A is called a foreign key. The relationship then requires the following rows
in R: (a,b) in S: (c1,d1,a) and (c2,d2,a)

many-many relationships expressed by the following relationships

- ((a1, b1) , (c1, d1))
- ((a1, b1), (c2, d2))
- ((a2, b2), (c1, d1))

generally require three tables, R and S and a third table containing nothing but the primary keys from R and S (the joining table)

R(AB), S(CD), RS(AC) with rows

- in R: (a1, b1) and (a2, b2)
- in S: (c1 ,d1) and (c2, d2) and
- in RS: (a1, c1),(a1, c2),(a2, c1)

Other considerations..
Weak entities...two tables using the foreign key from the strong partner or possibly one table.
Existence dependencies can usually be treated as weak entities are...but enforcement of the existence dependency cannot be implemented structurally..

ISA hierarchies again offer three possibilities...one table with a field or fields which differentiate subtypes or a separate table for each subtype such that the union of all subtables returns the entire superentity or both, ie a table which holds common information for all subtypes and specialized table for all or some of the subtypes...examples
Faculty has two subtypes Tenured and not...use one table with a field to distinguish them
Coaches can either be male or female, it is important to keep track of the maiden names of female coaches...use two tables with an extra field (Maiden Name) for female coaches
Airline Employees include pilots and cabin crew...use a central table for information common to all employees and specialized tables for pilots and cabin crew.

Relationships with Attributes: Use three tables, include attribute(s) with the joining table.
Normalization using the theory of Functional Dependencies

First, the need for normalization...redundancy.

Anomalies caused by redundancy:

The insertion anomaly...given the schema
EMP(name,ssno,b_date,address,dept_no,dept_name,dept_mgr)
1. To insert a new employee, we either have to know which department he/she
works for or use a null.
2. To insert new department, you'd have to create a dummy employee record.

In the "sports" database, this principle can be observed in an early version of the "teams"
table:
teams(code,type,wins,losses,coachln,coachfn,alma_mater,grad_year)
To create a team that doesn't yet have a coach, you need to insert dummy values.

This example can also serve to illustrate the deletion anomaly...

If a department has only one employee and you delete that employee the
department is deleted as well. Similarly, if a coach gets fired (i.e. he/she are no longer
associated to a teams) where does the coach's personal data go?

And the famous update anomaly.

If the dept manager changes, you have to change multiple records or else the data
will become inconsistent.

Notation: if we want to talk about the set of values that a tuple takes on a set of attributes
A, we will write t(A). So in relation colleges t(code, name, city, state) for the tuple that
corresponds to Yale University is

\{YALE, YALE UNIVERSITY, NEW HAVEN, CT\}

Defn. A set of attributes Y is said to be functionally dependent on a set of attributes X,
written X→Y if for all tuples t1 and t2,
If t1(X) = t2(X)
Then t1(Y) = t2(Y)

In English, this means that if tuple 1 and tuple 2 have the exact same value(s) on
X, then they have the exact same value(s) on Y. More important actually is the
contrapositive: if tuple 1 and tuple 2 disagree on Y then they MUST disagree on X

eg. at MSU it is true that course_no → no_credits
anti-eg. from the college table: name → state is NOT true.
Functional dependencies
    can be intrinsic like A → A
determined by the data like
    zip_code → city, state
imposed by the database designer
    team → Coach

Note that this last FD is actually a translation of a "business rule" i.e. A coach can coach only one team.

More notation:
    If X & Y are sets of attributes, then XY will be used instead X ∪ Y. In particular if A and B are single attributes, then AB means {A} ∪ {B}

**Defn.** Let F be a set of FD's. We say that F logically implies the FD X → Y, written $F \models X \rightarrow Y$, if whenever all the members of F are true for some relation r, then so too is X → Y.
Eg. { X → A, A → Y } X → Y

**Defn.** $F^+$, the closure of F is the set of all FD'd logically implied by F

**Defn.** Given a set of FD’s F on the relation R, the set X is called a superkey of R relative to F if X → R is in $F^+$

**Defn.** X is candidate key if X is a superkey and no proper subset of X is a superkey. One candidate key is chosen by the D.B.A. to be a primary key. Eg. From the Colleges table, Code is a candidate key, and so is Name-City-State.

Calculating $F^+$ is a NP-complete problem, so we need another way.

Armstrong's Axioms:
The Inference Rules for Functional Dependencies.

1. Reflexivity: If X is a superset of Y, i.e. $X \supseteq Y$, then $X \rightarrow Y$
2. Augmentation: if $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any set Z
3. Transitivity: if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$
4. Additivity: (Sometimes called Union): if $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
5. Pseudotransitivity: if $X \rightarrow Y$ and $WY \rightarrow Z$, then $XW \rightarrow Z$.
6. Projectivity (Decomposition): if $X \rightarrow YZ$ then $X \rightarrow Y$ (and $X \rightarrow Z$)

**Defn.** Given a set of FD's we say that a FD X → Y can be derived from F, written $F \vdash X \rightarrow Y$ if there exists a sequence of FD's $F_1, F_2, \ldots, F_n$ such that each $F_i$ is either in F or can be derived from previous Fd's in the sequence using the derivation rules and $F_n$ is $X \rightarrow Y$
eg. Pseudotransitivity can be derived using only the first 3 rules...

\[
\begin{align*}
&\text{WY}\rightarrow Z \text{ (hypothesis)} \\
&\text{X}\rightarrow Y \text{ (hypothesis)} \\
&\text{XW}\rightarrow \text{WY} \text{ (augmentation)} \\
&\text{XW}\rightarrow Z \text{ (transitivity)}
\end{align*}
\]

**Theorem**: The rules of inference are sound (i.e. never inferring an invalid inference),

Pf...some other time

**Thrm**: The rules of inference are complete, i.e. if a FD, F, is logically implied by F, then
F can be derived from F (i.e the set of rules can be used to infer any valid inference).

Pf ..Some other place

**Conclusion**: Fortunately, is a mechanical way to calculate F+.
Unfortunately this method is also NP-Complete
Fortunately there's a kludge.

**Defn**: Given a set of attributes, X, X+ (the closure of X) is defined as the subset of attributes A₁, A₂, A₃..Aₙ s.t. X→Aᵢ

Extended example:

\[
\begin{align*}
\text{R=ABCDEFG} \\
F &= \{ \\
&\text{AB}\rightarrow \text{C} \quad \text{D}\rightarrow \text{EG} \\
&\text{C}\rightarrow \text{A} \quad \text{BE}\rightarrow \text{C} \\
&\text{BC}\rightarrow \text{D} \quad \text{CG}\rightarrow \text{BD} \\
&\text{ACD}\rightarrow \text{B} \quad \text{CE}\rightarrow \text{AG} \\
&\text{F}\rightarrow \text{A}
\}
\end{align*}
\]

Calculate AB+

\[
\begin{align*}
&\text{AB}\rightarrow \text{AB} \text{ reflexivity} \\
&\text{AB}\rightarrow \text{C} \text{ hypothesis} \\
&\text{AB}\rightarrow \text{ABC} \text{ additivity} \\
&\text{AB}\rightarrow \text{BC} \text{ projection} \\
&\text{BC}\rightarrow \text{D} \text{ hypothesis} \\
&\text{AB}\rightarrow \text{D} \text{ transitivity} \\
&\text{AB}\rightarrow \text{ABCD} \text{ additivity} \\
&\text{.} \\
&\text{.}
\end{align*}
\]
AB → ABCDEG

**NB:** ABF → ABCDEFG indicates that ABF is a superkey. In fact you could check that a candidate key is BF.

As an exercise try to prove that CB → ACBDEG