Updates

The database maintains a current database state.

Updates to the data:

Write

1) add a tuple
2) delete a tuple
3) modify an attribute in a tuple

⟺ Read Query

Updates to the data happen very frequently.

Updates to the schema: relatively rare. Rather painful. Why?

Q: Schema vs. Instances

- Think of it as columns vs. rows

Think of an example, answer following:

For Schema:

- When do you determine a schema?
- How often do you change your mind?

For Instance:

- When do you determine an instance?
- How often do you change your mind?

Gut? Vision?

CS Hall of Fame

Behind the Scene: Database Turing Awards?

1966 A.J. Perlis
1967 Maurice V. Wilkes
1968 Richard Hamming
1969 Marvin Minsky
1970 J.H. Wilkinson
1971 John McCarthy
1972 E.W. Dijkstra
1973 Charles W. Bachman
1974 Donald E. Knuth
1975 Allen Newell
1975 Herbert A. Simon
1976 Michael O. Rabin
1977 John Backus

Ted Codd

Network DB

1978 Robert W. Floyd
1979 Kenneth E. Iverson
1980 C. Antony R. Hoare
1981 Edgar F. Codd
1982 Stephen A. Cook
1983 Ken Thompson
1983 Dennis M. Ritchie
1984 Niklaus Wirth
1985 Richard M. Karp
1986 John Hopcroft
1986 Robert Tarjan
1987 John Cocke

1988 Ivan Sutherland
1989 William (Velvel) Kahan
1990 Fernando J. Corbato'
1991 Robin Milner
1992 Butler W. Lampson
1993 Juris Hartmanis
1993 Richard E. Stearns
1994 Edward Feigenbaum
1994 Raj Reddy
1995 Manuel Blum
1996 Amir Pnueli
1997 Douglas Engelbart
1998 James Gray
1999 Frederick P. Brooks, Jr.
2000 Andrew Chi-Chih Yao

Behind the Scene: It's all about modeling

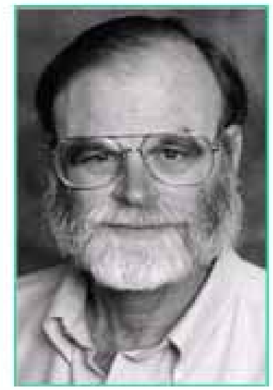
- 1973 Charles W. Bachman
- 1981 Edgar F. Codd
- 1998 James Gray

Who's who?

What have they contributed?

And we certainly need more!

A



B

C

16

Defining a Database Schema in SQL

Defining a Database Schema

- A database schema comprises declarations for the relations ("tables") of the database.
- Many other kinds of elements may also appear in the database schema, including views, indexes, and triggers, which we'll introduce later.

SQL Querying

Data Manipulation.

Declaring a Relation

- Simplest form is:

    CREATE TABLE <name> (

        <list of elements>

    );

- And you may remove a relation from the database schema by:

    DROP TABLE <name>;

Elements of Table Declarations

- The principal element is a pair consisting of an attribute and a type.
- The most common types are:
  - INT or INTEGER (synonyms).
  - REAL or FLOAT (synonyms).
  - CHAR(n) = fixed-length string of n characters.
  - VARCHAR(n) = variable-length string of up to n characters.

Example: Create Table

```
CREATE TABLE Sells (
    bar      CHAR(20),
    beer          VARCHAR(20),
    price         REAL
);
```

Dates and Times

- DATE and TIME are types in SQL.
- The form of a date value is:

  DATE ‘yyyy-mm-dd’

  – Example: DATE ‘2002-09-30’ for Sept. 30, 2002.

Times as Values

- The form of a time value is:

  TIME ‘hh:mm:ss’

with an optional decimal point and fractions of a second following.

  – Example: TIME ’15:30:02.5’ = two and a half seconds after 3:30PM.

Declaring Keys

- An attribute or list of attributes may be declared PRIMARY KEY or UNIQUE.
- These each say the attribute(s) so declared functionally determine all the attributes of the relation schema.
- There are a few distinctions to be mentioned later.

Declaring Single-Attribute Keys

- Place PRIMARY KEY or UNIQUE after the type in the declaration of the attribute.
- Example:

```
CREATE TABLE Beers (
    name    CHAR(20) UNIQUE,
    manf    CHAR(20)
);
```

Declaring Multiattribute Keys

- A key declaration can also be another element in the list of elements of a CREATE TABLE statement.
- This form is essential if the key consists of more than one attribute.
  - May be used even for one-attribute keys.

Example: Multiattribute Key

- The bar and beer together are the key for Sells:

```
CREATE TABLE Sells (
        bar         CHAR(20),
        beer        VARCHAR(20),
        price       REAL,
        PRIMARY KEY (bar, beer)
);
```

PRIMARY KEY Versus UNIQUE

- The SQL standard allows DBMS implementers to make their own distinctions between PRIMARY KEY and UNIQUE.
  - Example: some DBMS might automatically create an *index* (data structure to speed search) in response to PRIMARY KEY, but not UNIQUE.

Q: What does MySQL do for UNI ^ PRI ?

Required Distinctions

Standard

- However, standard SQL requires these distinctions:
  1. There can be only one PRIMARY KEY for a relation, but several UNIQUE attributes.
  2. No attribute of a PRIMARY KEY can ever be NULL in any tuple. But attributes declared UNIQUE may have NULL's, and there may be several tuples with NULL.

Other Declarations for Attributes

- Two other declarations we can make for an attribute are:
  1. NOT NULL means that the value for this attribute may never be NULL.
  2. DEFAULT <value> says that if there is no specific value known for this attribute's component in some tuple, use the stated <value>.

Example: Default Values

```
CREATE TABLE Drinkers (
   name CHAR(30) PRIMARY KEY,
   addr CHAR(50)
      DEFAULT '123 Sesame St.',
   phone CHAR(16)
);
```

Effect of Defaults -- 1

- Suppose we insert the fact that Sally is a drinker, but we know neither her address nor her phone.
- An INSERT with a partial list of attributes makes the insertion possible:

```
INSERT INTO Drinkers(name)
VALUES('Sally');
```

addr

= '123 S. St'

Effect of Defaults -- 2

- But what tuple appears in Drinkers?

name	addr	phone
‘Sally’	‘123 Sesame St’	NULL

- If we had declared phone NOT NULL, this insertion would have been rejected.

Adding Attributes

- We may change a relation schema by adding a new attribute ("column") by:

  ALTER TABLE <name> ADD

  <attribute declaration>;

- Example:

```
ALTER TABLE Bars ADD
phone CHAR(16)DEFAULT 'unlisted';
```

Deleting Attributes

- Remove an attribute from a relation schema by:

  good student in CS411 won't do.

  ALTER TABLE <name>
  DROP <attribute>;

- Example: we don't really need the license attribute for bars:

```
ALTER TABLE Bars DROP license;
```

Peter Chen Edgar Codd

ER Model → Relational Model

+ close to R.W. + Algebra to manipulate relations.

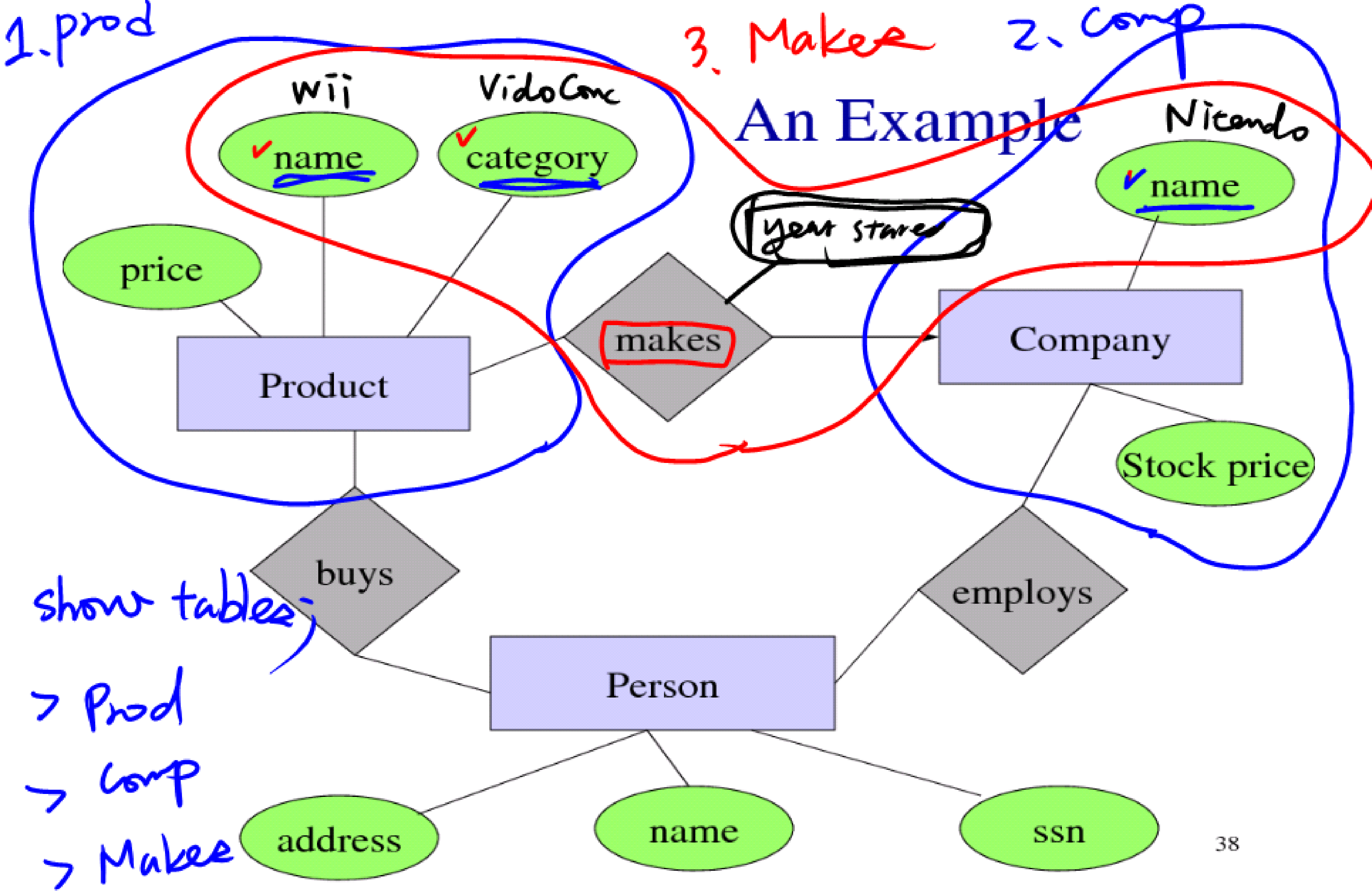
X

Translating ER Diagram to Rel. Design

Table

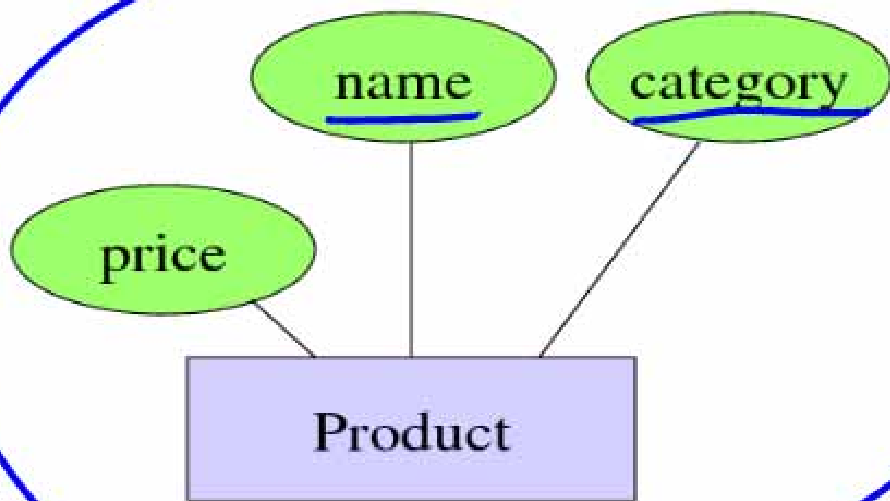
- Basic cases
  - E – entity set E = relation with attributes of E
  - R – relationship R = relation with attributes being keys of related entity sets + attributes of R
- Special cases
  - combining two relations
  - translating weak entity sets
  - translating is-a relationships and subclasses

An Example


1. prod
3. Makes
2. comp
Wii
VidoConc
Nitendo
name
category
price
Product
makes
year started
name
Company
Stock price
buys
employs
Person
address
name
ssn
show tables;
> Prod
> comp
> Makes

Basic Cases

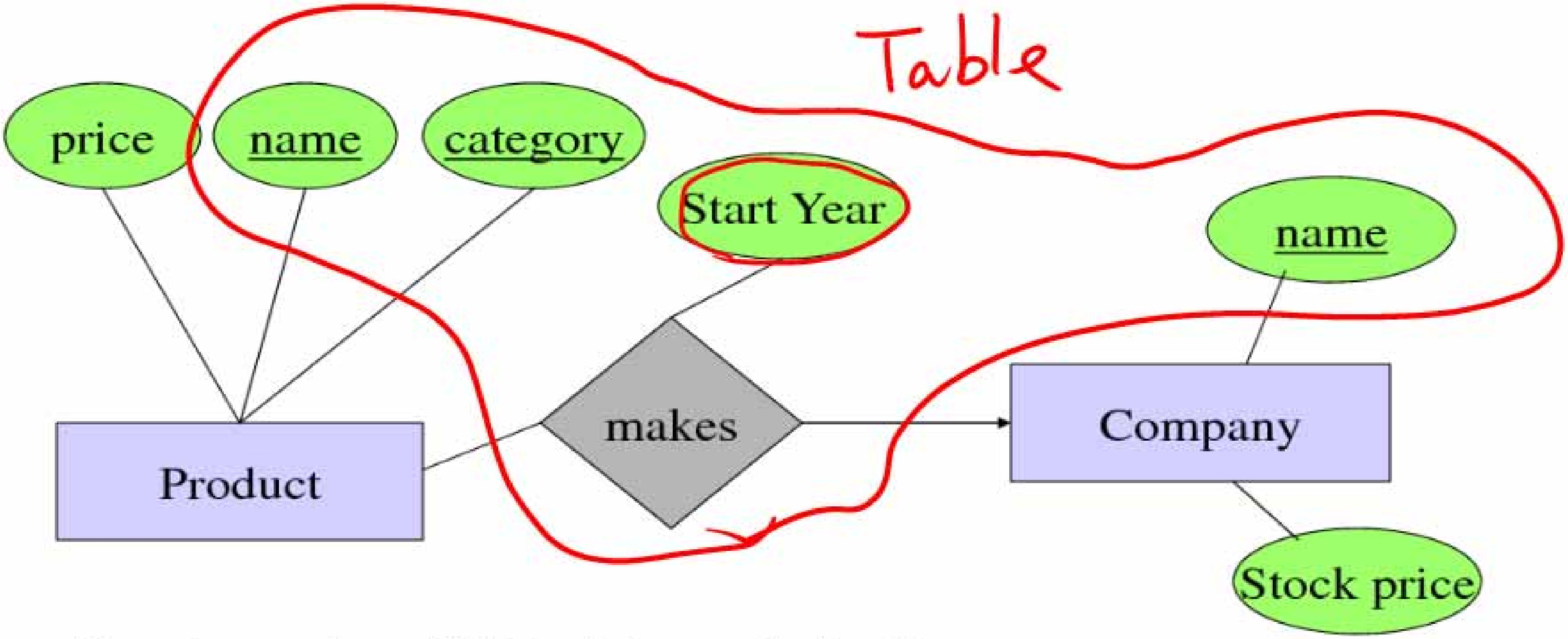
Entity Set to Relation



Product:

Name	Category	Price
gizmo	gadgets	$19.99

Q: Relationship → Relation?

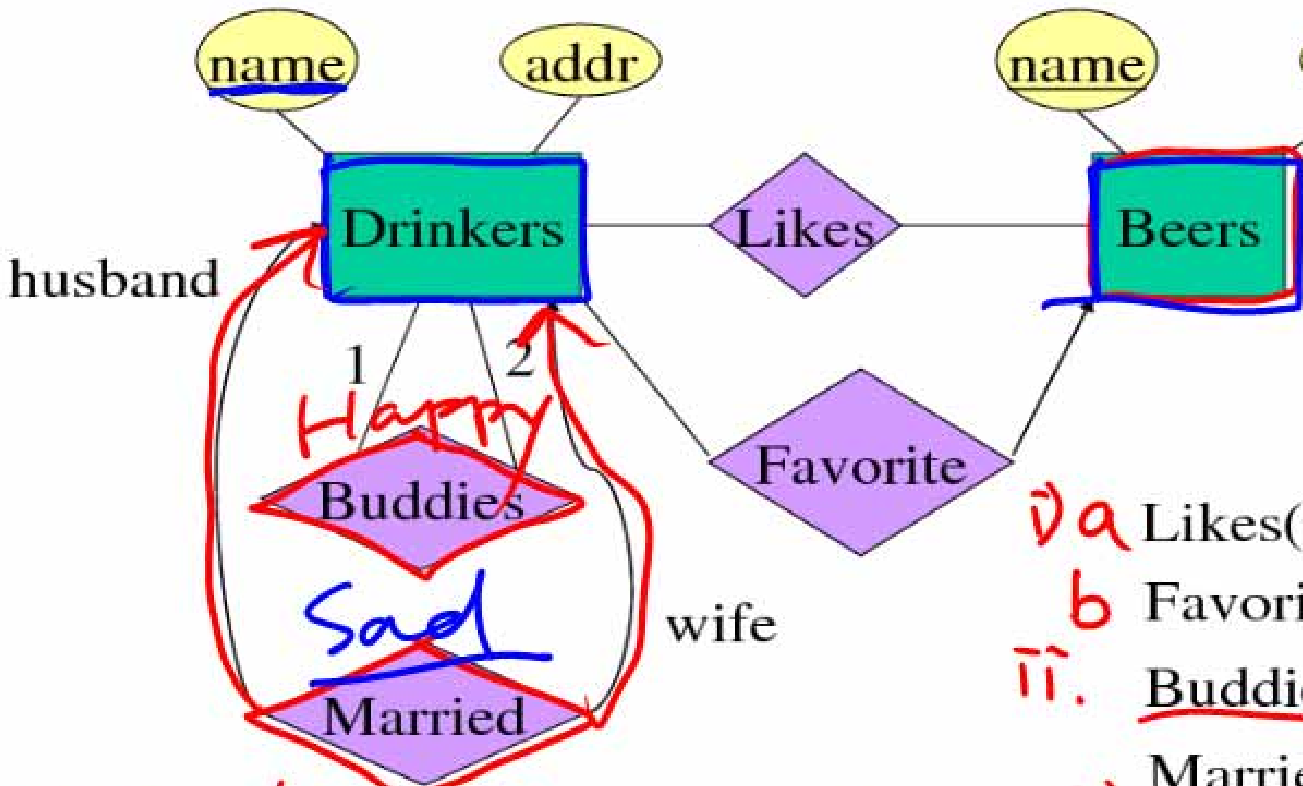


How to capture "Make" in a relation?

① key of prod ③ attr of makes

② key of comp

Relationship to Relation: Another Example



i) a Likes(drinker, beer)

b Favorite(drinker, beer)

ii. Buddies(name1, name2)

Married(husband, wife)

i) Diff relationships can exist between same set of entities

ii) a table can participate in multiple relationships

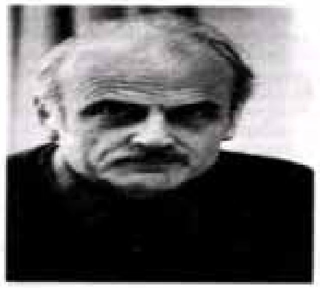
Cmt

Behind the Scene:
This was how relational DBMS started…

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation).

The 1981 ACM Turing Award Lecture

Delivered at ACM '81, Los Angeles, California, November 9, 1981



The 1981 ACM Turing Award was presented to Edgar F. Codd, an IBM Fellow of the San Jose Research Laboratory, by President Peter Denning on November 9, 1981 at the ACM Annual Conference in Los Angeles, California. It is the Association's foremost award for technical contributions to the computing community.

Codd was selected by the ACM General Technical Achievement Award Committee for his "fundamental and continuing contributions to the theory and practice of database management systems." The originator of the relational model for databases, Codd has made further important contributions in the development of relational algebra, relational calculus, and normalization of relations.

Edgar F. Codd joined IBM in 1949 to prepare programs for the Selective Sequence Electronic Calculator. Since then, his work in computing has encompassed logical design of computers (IBM 701 and Stretch), managing a computer center in Canada, heading the development of one of the first operating systems with a general multiprogramming capability, contributing to the logic of self-reproducing automata, developing high level techniques for software specification, creating and extending the relational approach to database management, and developing an English analyzing and synthesizing subsystem for casual users of relational databases. He is also the author of *Cellular Automata*, an early volume in the ACM Monograph Series.

Codd received his B.A. and M.A. in Mathematics from Oxford University in England, and his M.Sc. and Ph.D. in Computer and Communication Sciences from the University of Michigan. He is a Member of the National Academy of Engineering (USA) and a Fellow of the British Computer Society.

The ACM Turing Award is presented each year in commemoration of A. M. Turing, the English mathematician who made major contributions to the computing sciences.

Relational Database: A Practical Foundation for Productivity

E. F. Codd
IBM San Jose Research Laboratory

It is well known that the growth in demands from end users for new applications is outstripping the capability of data processing departments to implement the corresponding application programs. There are two complementary approaches to attacking this problem (and both approaches are needed): one is to put end users into direct touch with the information stored in computers; the other is to increase the productivity of data processing professionals in the development of application programs. It is less well known that a single technology, relational database management, provides a practical foundation for both approaches. It is explained why this is so.

While developing this productivity theme, it is noted that the time has come to draw a very sharp line between relational and non-relational database systems, so that the label "relational" will not be used in misleading ways. The key to drawing this line is something called a "relational processing capability."

CR Categories and Subject Descriptors: H.2.0 [**Database Management**]: General; H.2.1 [**Database Management**]: Logical Design–*data models*; H.2.4 [**Database Management**]: Systems

General Terms: Human Factors, Languages

Additional Key Words and Phrases: database, relational database, relational model, data structure, data manipulation, data integrity, productivity

Author's Present Address: E. F. Codd, IBM Research Laboratory, 5600 Cottle Road, San Jose, CA 95193.

Special Cases:

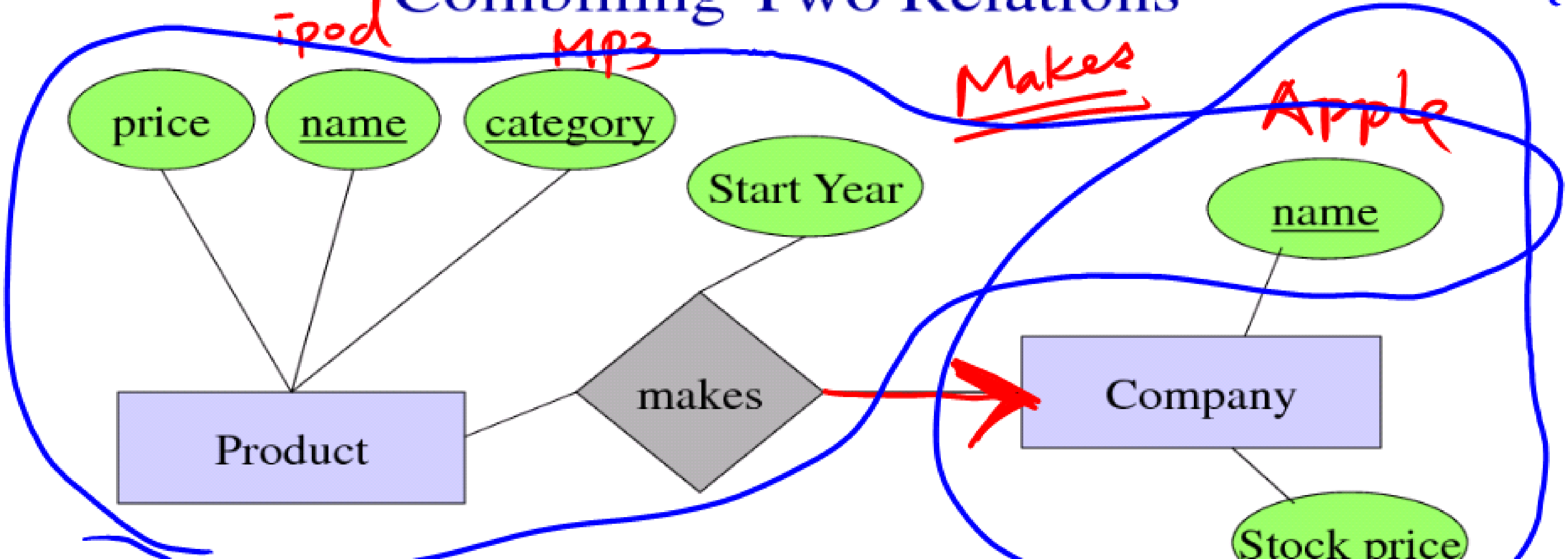
1) many one relations

2) weak entity sets

3) isa cases

price	name	cat	name	cat	comp
$150	ipod	MP3	ipod	MP3	apple
$200	Wii	VC.	Wii	V.C.	Nit

Combining Two Relations



No need for Makes. Just modify Product:

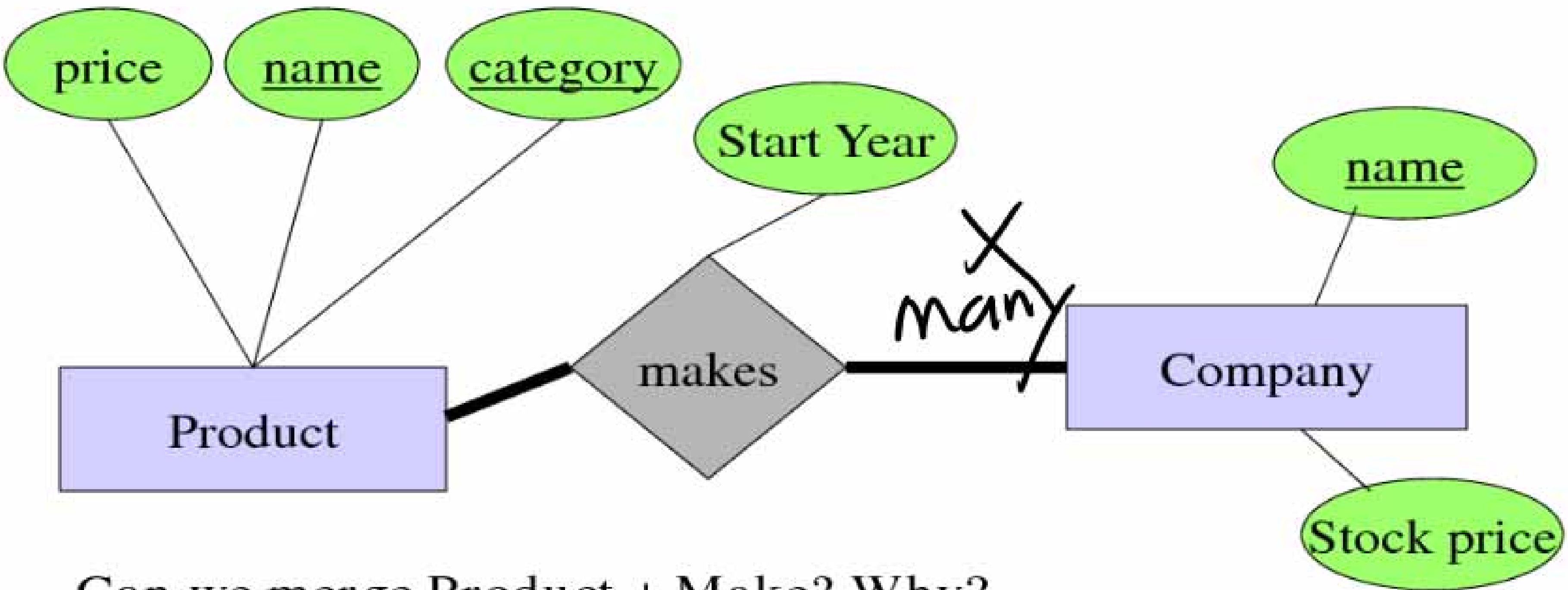
name	category	price	StartYear	companyName
gizmo	gadgets	19.99	1963	gizmoWorks

Combining Relations

- It is OK to combine the relation for an entity-set *E* with the relation *R* for a many-one relationship from *E* to another entity set.

- Example: Drinkers(name, addr) and Favorite(drinker, beer) combine to make Drinker1(name, addr, favoriteBeer).

Q: What happen if Many-to-Many?



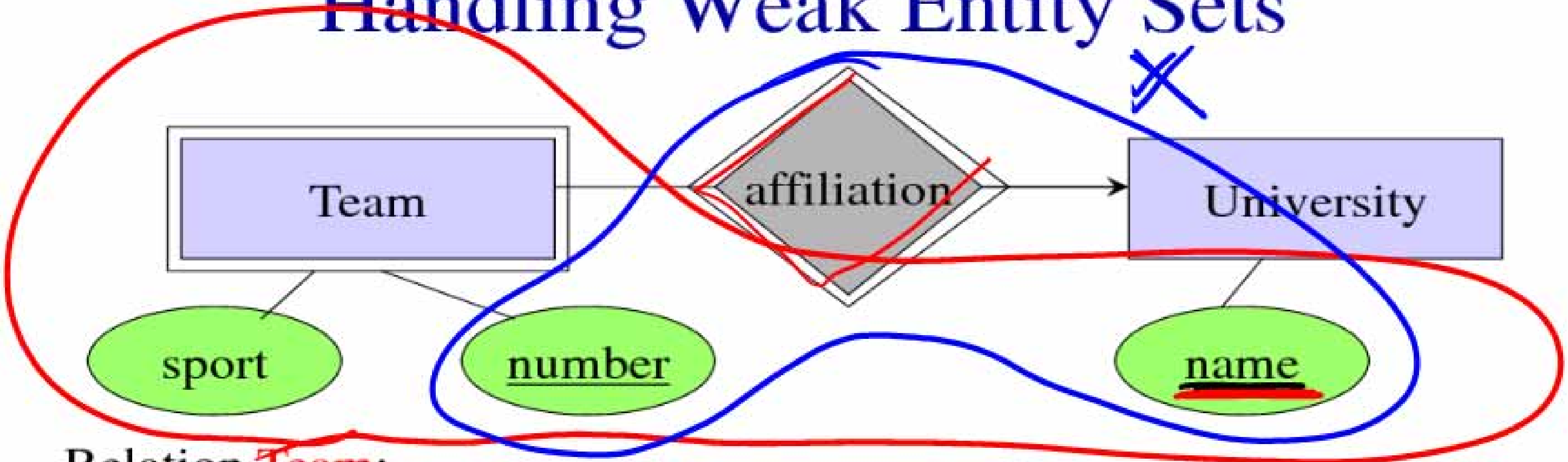
Can we merge Product + Make? Why?

price	name	cat	comp
$100	ipod	MP3	?

X set

name	cat	comp
ipod	MP3	Apple
ipod	MP3	Dell

Handling Weak Entity Sets



Relation Team:

Sport	Number	Affiliated University
mud wrestling	15	Montezuma State U.

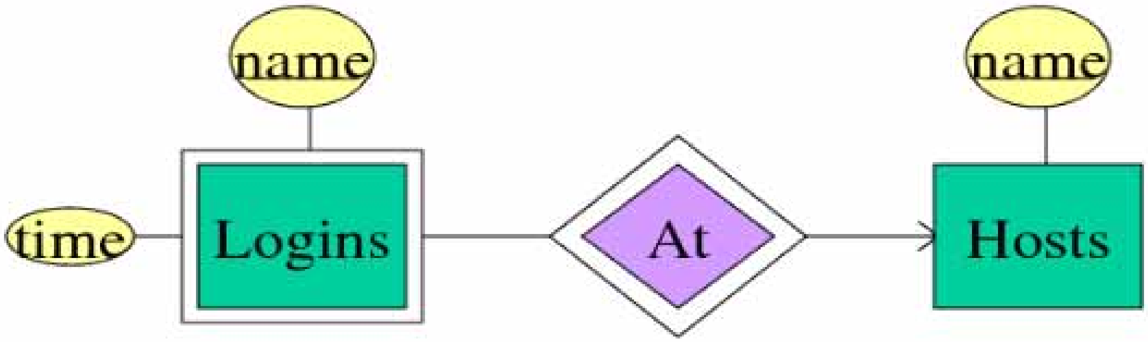
- need all the attributes that contribute to the key of Team
- don't need a separate relation for Affiliation. (why ?)

Handling Weak Entity Sets

- Relation for a weak entity set must include attributes for its complete key (including those belonging to other entity sets), as well as its own, nonkey attributes.

- A supporting (double-diamond) relationship is redundant and yields no relation.

Another Example



Hosts(hostName)
Logins(loginName, hostName, time)
~~At(loginName, hostName)~~

At becomes part of
Logins