DB Design

- Ch 2. The Entity-Relationship Model
- Ch 3.5 Translating ER model to relational schema
- Ch 19 Normalization
- Ch 25.2 Multi-dimensional data model

Ch 2. The Entity-Relationship Model

Sang-Won Lee

http://icc.skku.ac.kr/~swlee

SKKU VLDB Lab. & SOS

(http://vldb.skku.ac.kr/)

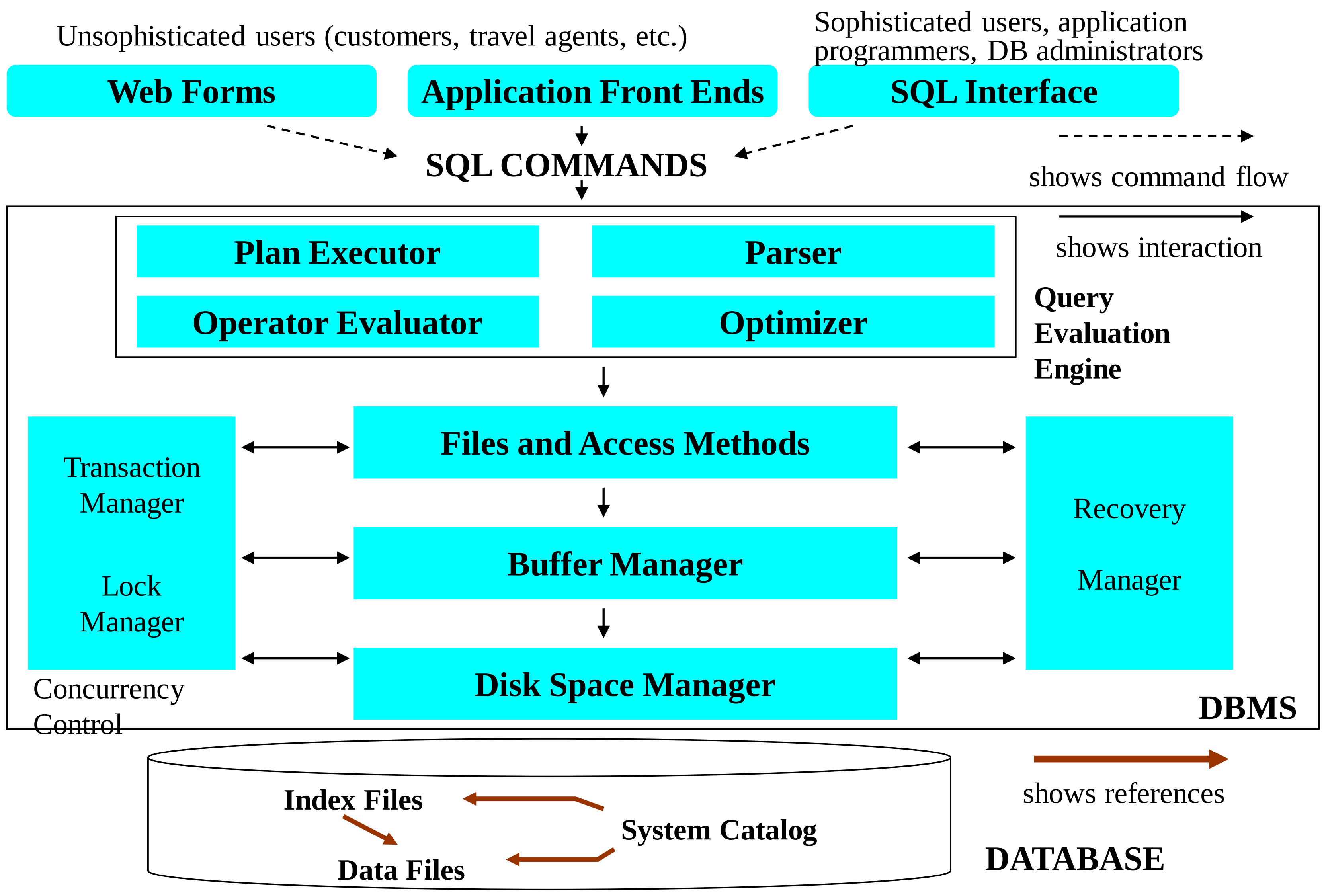


Figure 1.3 Architecture of a DBMS

ER Data Model

- Peter Chen, "The Entity-Relationship Model - Toward a Unified View of Data", ACM TODS 76,

- ER model **vs.** Relational model / normalization theory (Chap 19)
  - No ER-DBMS (← no query language, RDBMS fever)
  - But, ER model has been successful in logical database schema design
    - ✓ C.f. Normalization theory (ch 19, by Codd): Too difficult for DBAs to understand;
  - ER (high level lang.) vs. Normalization theory (machine lang.)

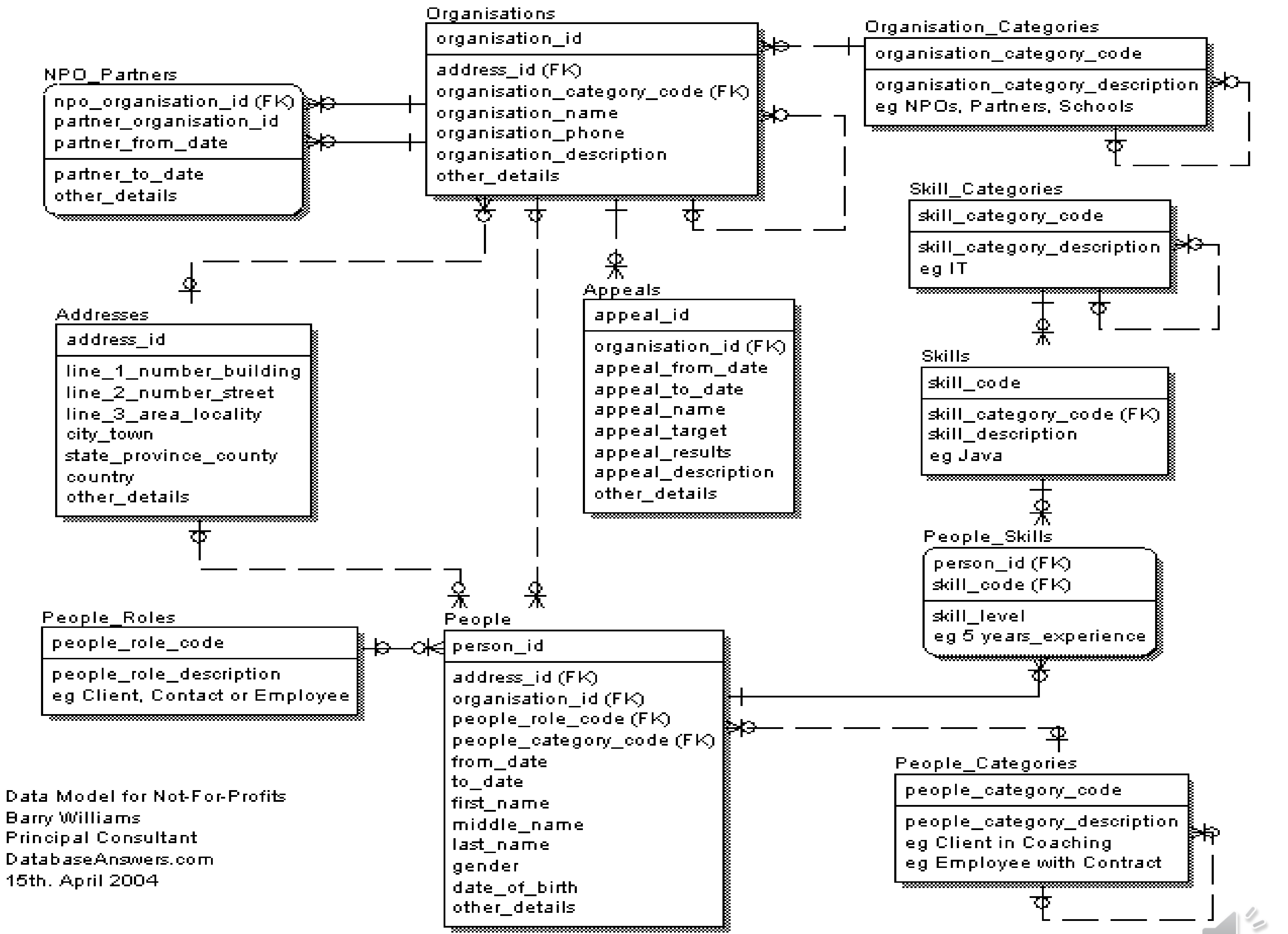
* Source: Stonebraker et. al. "What goes around comes around?"

Overview of Database Design

1. Requirement analysis: via informal discussion with user groups
2. **Conceptual design**: ER Diagram vs. UML
   - Commercial tools: ERWin, Oracle Designer, IBM Rational Rose
   - What kind of info.: Entity, Relationship, Constraints
3. **Logical design**: RDB Schema
   - Tables A(<u>a</u>,b,c), B(<u>d,e</u>,f), C(….), …..
   - Constraints: PK/FK/Check/Assertion …
4. **Physical design**
   - Create Table (a,b,c) .. physical property;
   - <u>Index, block size, partitioning, disk allocation</u>, ….

Overview of Database Design(2)

- Conceptual design: (ER Model is used.)
  - What are the *entities* and *relationships* in the enterprise?
  - What information about entities and relationships should we store in the database? - *attribute*
  - What are the *integrity constraints* or *business rules* that hold?
  - A database `schema' in the ER Model can be represented pictorially (*ER diagrams*).
  - Can map an ER diagram into a relational schema.
    - ✓ semi-automatic!



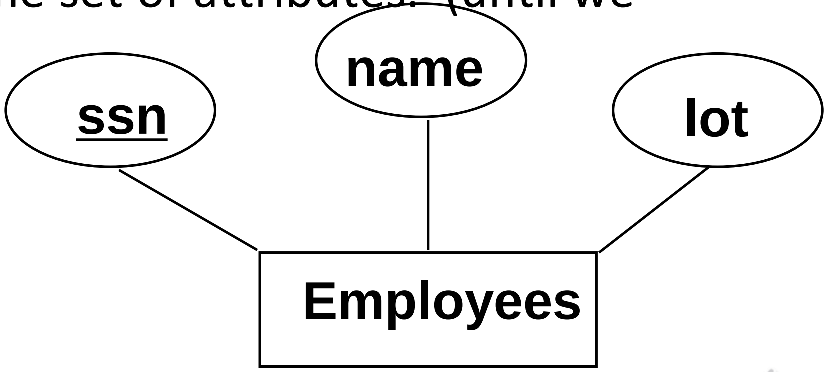
Organisations
organisation_id
address_id (FK)
organisation_category_code (FK)
organisation_name
organisation_phone
organisation_description
other_details
Organisation_Categories
organisation_category_code
organisation_category_description
eg NPOs, Partners, Schools
NPO_Partners
npo_organisation_id (FK)
partner_organisation_id
partner_from_date
partner_to_date
other_details
Skill_Categories
skill_category_code
skill_category_description
eg IT
Addresses
address_id
line_1_number_building
line_2_number_street
line_3_area_locality
city_town
state_province_county
country
other_details
Appeals
appeal_id
organisation_id (FK)
appeal_from_date
appeal_to_date
appeal_name
appeal_target
appeal_results
appeal_description
other_details
Skills
skill_code
skill_category_code (FK)
skill_description
eg Java
People_Skills
person_id (FK)
skill_code (FK)
skill_level
eg 5 years_experience
People_Roles
people_role_code
people_role_description
eg Client, Contact or Employee
People
person_id
address_id (FK)
organisation_id (FK)
people_role_code (FK)
people_category_code (FK)
from_date
to_date
first_name
middle_name
last_name
gender
date_of_birth
other_details
People_Categories
people_category_code
people_category_description
eg Client in Coaching
eg Employee with Contract
Data Model for Not-For-Profits
Barry Williams
Principal Consultant
DatabaseAnswers.com
15th. April 2004

2.2 Entities

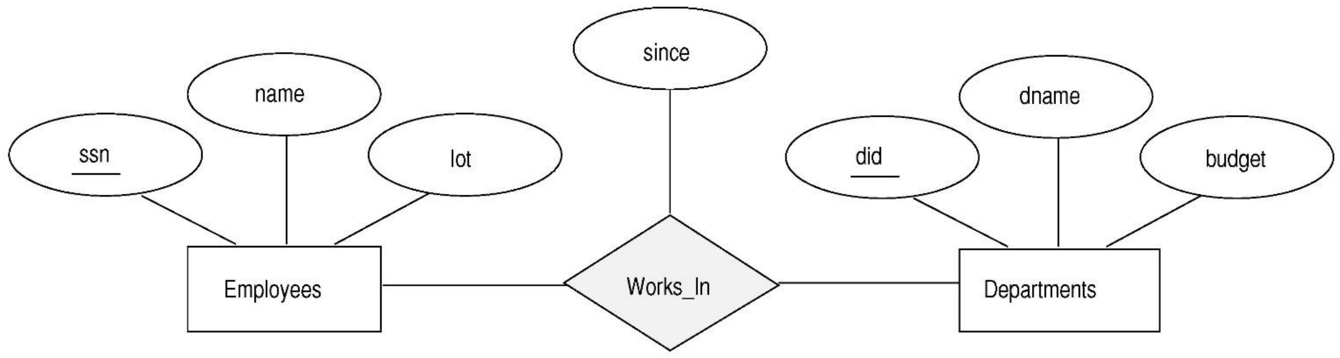
- *Entity:* Real-world object distinguishable from other objects.

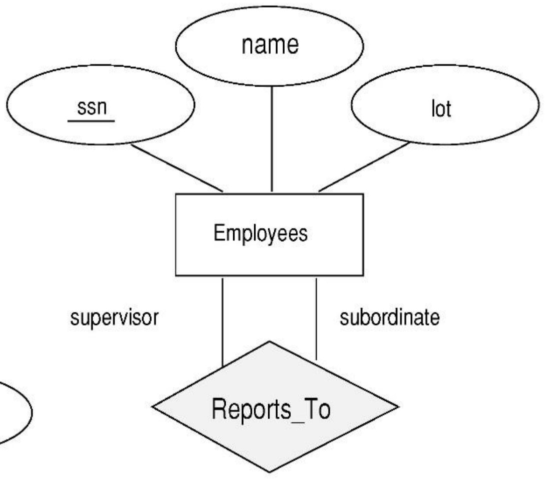
- An entity is described (in DB) using a set of *attributes*.

- *Entity Set*: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes. (until we consider ISA hierarchies, anyway!)
  - Each entity set has a *key*.
  - Each attribute has a *domain*.



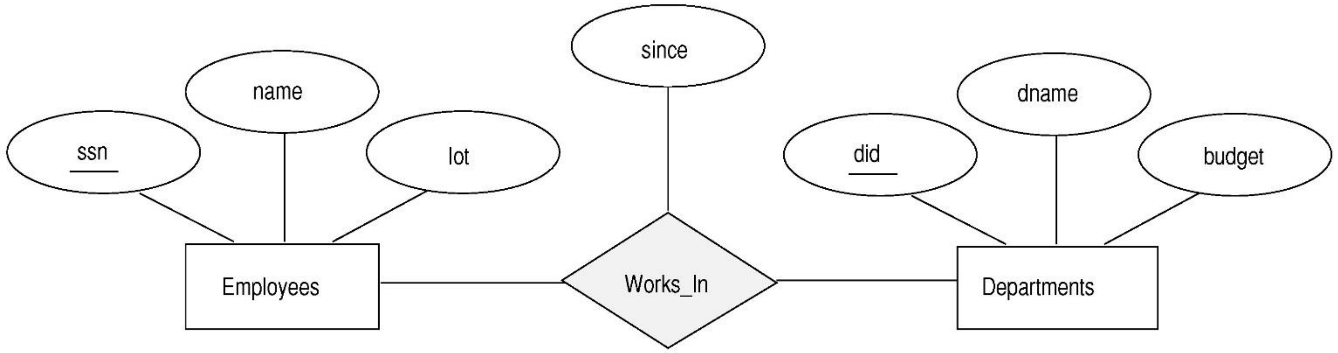
2.3 Relationships

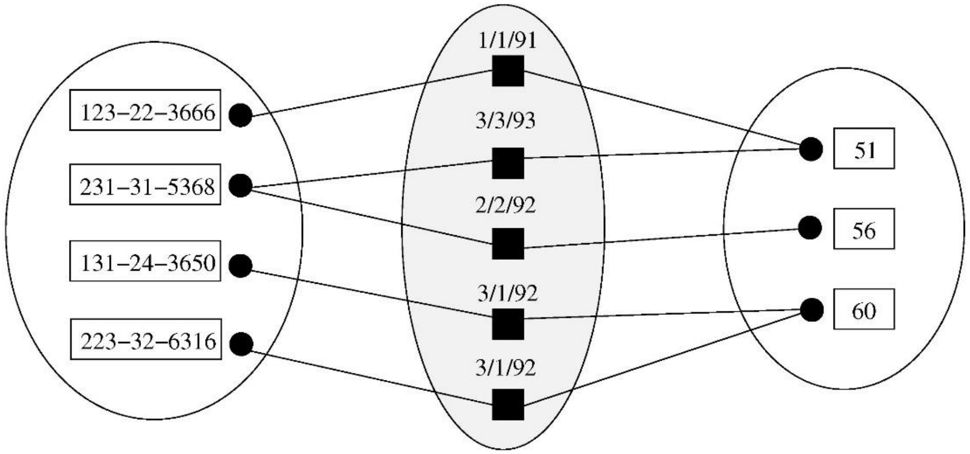




- *Relationship*: Association among two or more entities. E.g., Attishoo works in Pharmacy department.
  - A relationship can have **descriptive attributes**
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets E1 ... En; each relationship in R involves entities e1 in E1, ..., en in En; e.g. binary, ternary
  - Same entity set could participate in different relationship sets, or in different “roles” in same relationship set (with separate role indicators).: works_in vs. reports_to

Relationship Set and Its Instance - M:N



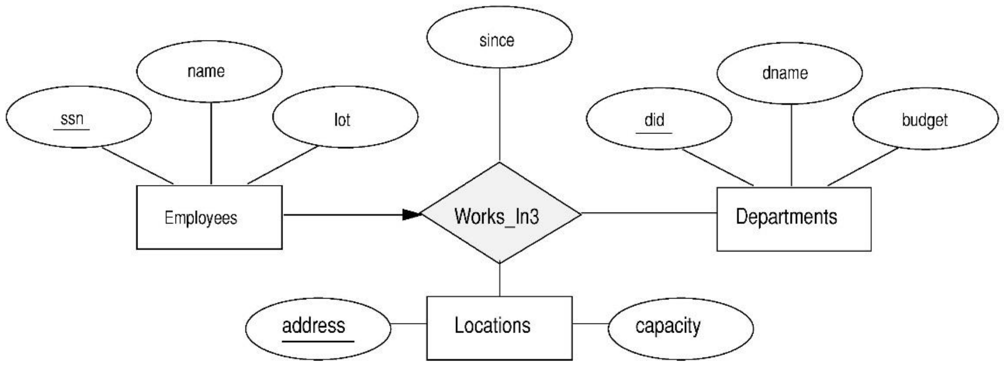


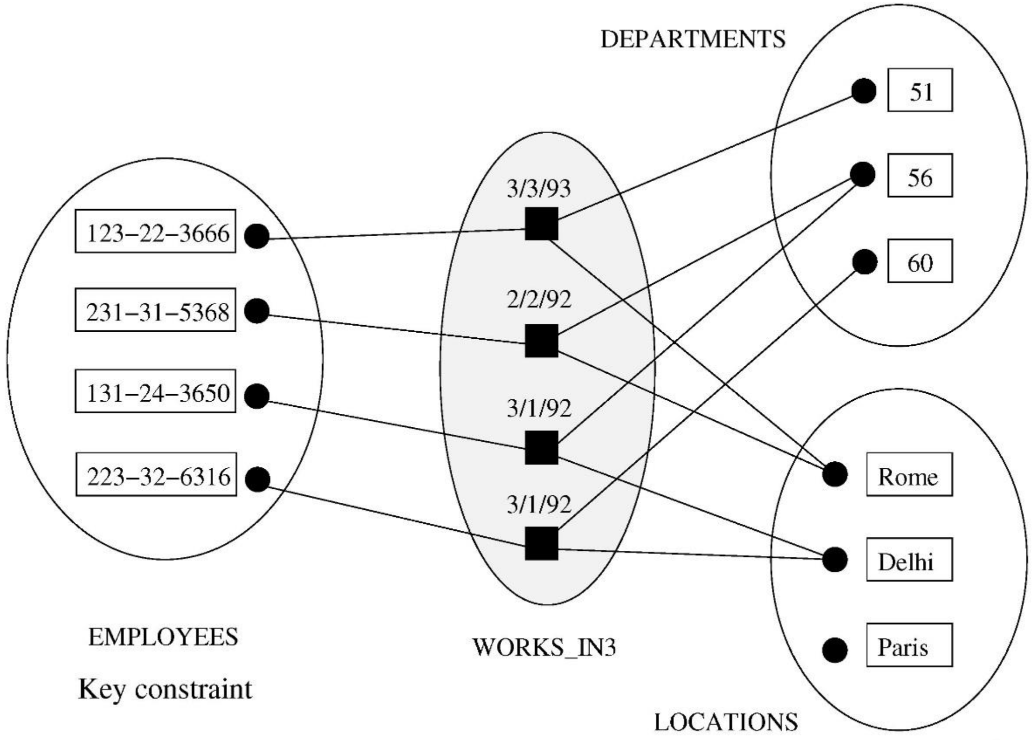
EMPLOYEES
Total participation

WORKS_IN
Many to Many

DEPARTMENTS
Total participation

Ternary Relationship Set



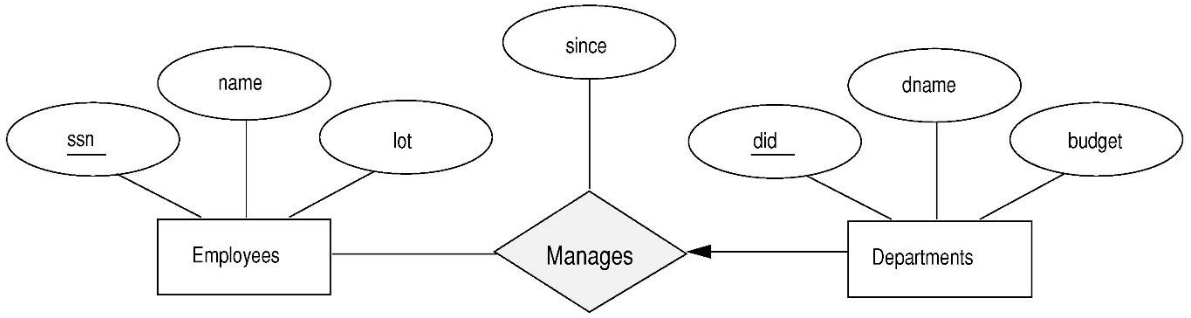


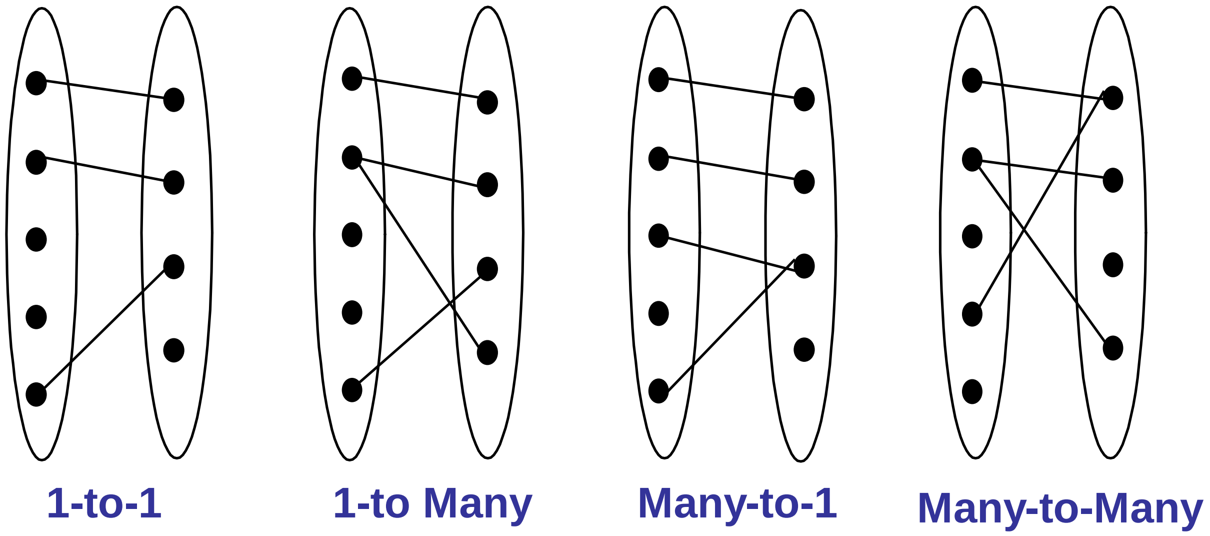
2.4.1 Key Constraints

- Consider Works_In: An employee can work in many departments; a dept can have many employees. (M:N)

- In contrast, each dept has at most one manager, according to the *key constraint* on Manages.

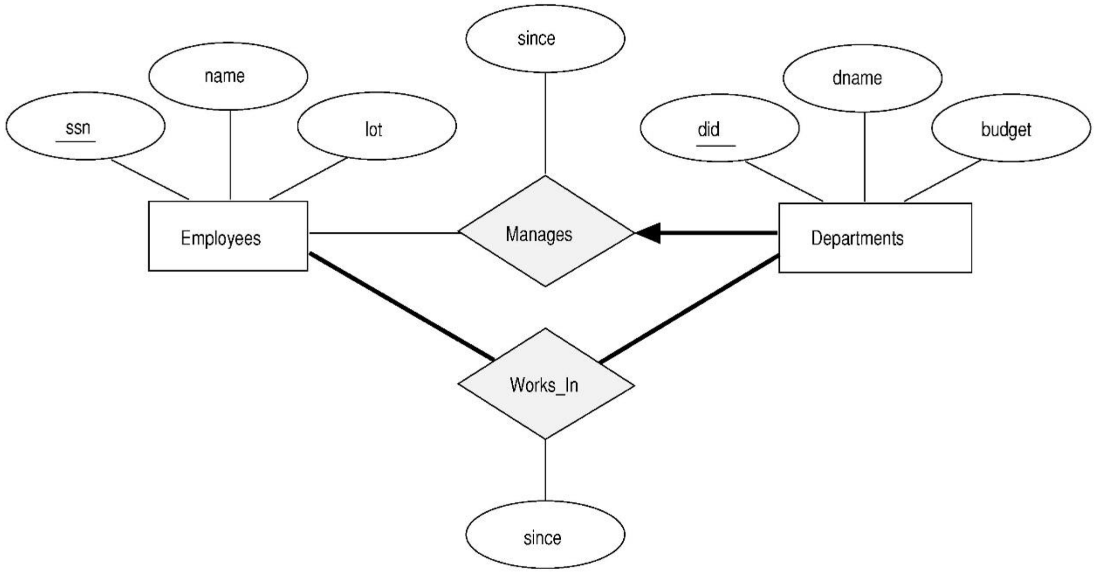
(M:1)



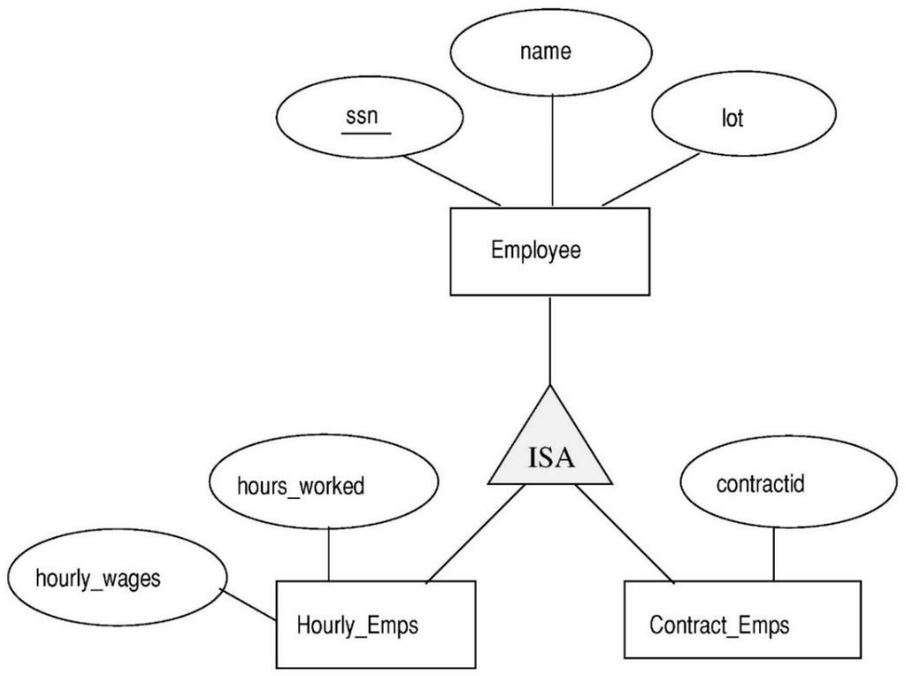


2.4.2 Participation Constraints

- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be ***total*** (vs. ***partial***).
    - ✓ Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)



2.4.4 Class Hierarchy



- Superclass vs. Subclass
- Specialization vs. Generalization
- Overlapping and covering constraint

2.5 Conceptual Design

- Several design alternatives
  - Entity vs. attribute
  - Entity vs. relationship
  - Binary relationship vs. ternary
  - Ternary vs. aggregation

Ch 3. Relational Data Model:

Section 3.5 Logical DB Design

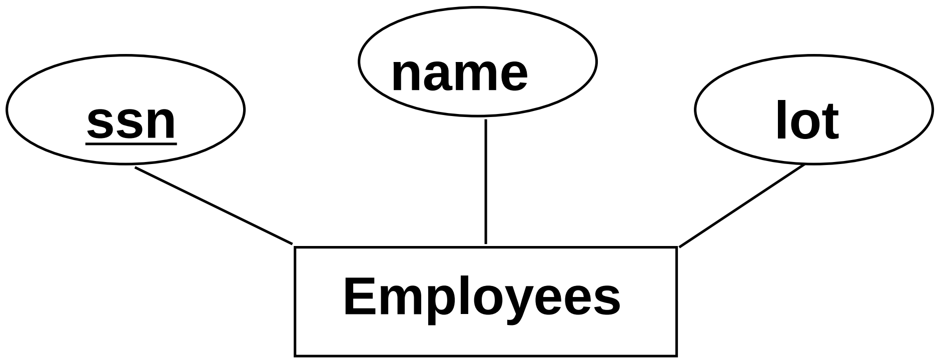
Sang-Won Lee

http://icc.skku.ac.kr/~swlee

SKKU VLDB Lab. & SOS

(http://vldb.skku.ac.kr/)

Entity Sets to Tables



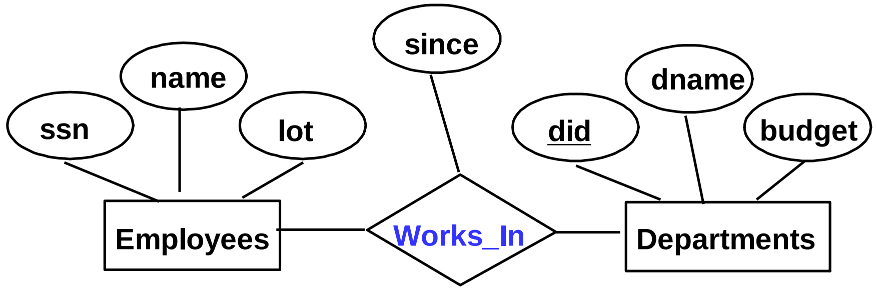
```
CREATE TABLE Employees
        ssn   CHAR(11),
        name  CHAR(20),
        lot   INTEGER,
        PRIMARY KEY  (ssn))
```

- More complex semantics:
  - e.g what about class hierarchy?

- Note: in RDB, field has atomic value!

Relationship Sets to Tables

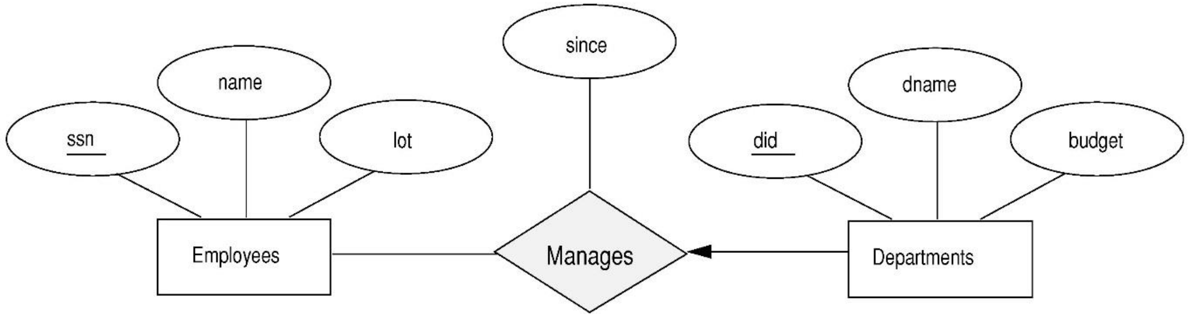
- In translating a relationship set to a relation, attributes of the relation must include:
  - Keys for each participating entity set (as foreign keys).
    - ✓ This set of attributes forms a *superkey* for the relation.
  - All descriptive attributes.

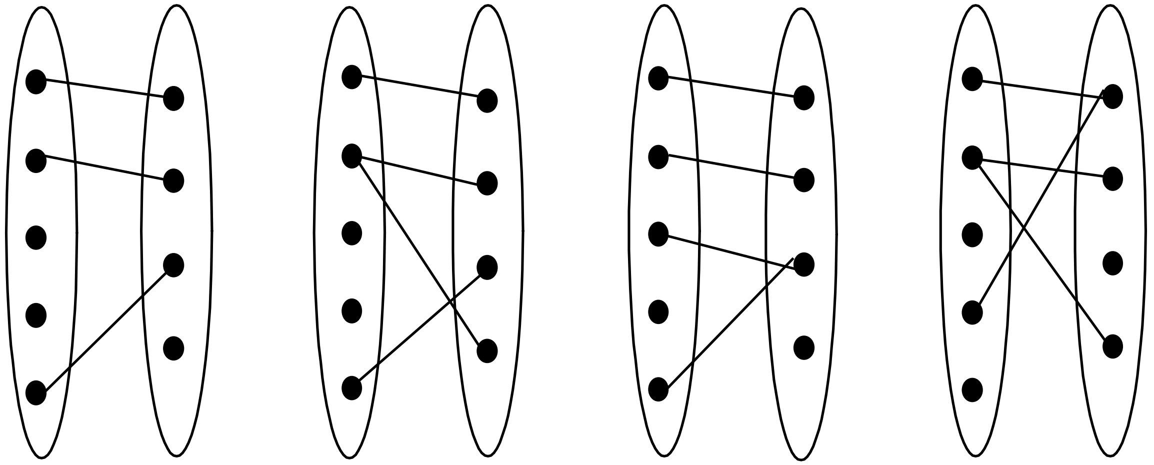


```
CREATE TABLE Works_In (
   ssn   CHAR(11),
   did   INTEGER,
   since   DATE, /* descriptive attr.*/
   PRIMARY KEY (ssn, did),
   FOREIGN KEY (ssn)
           REFERENCES Employees,
   FOREIGN KEY (did)
           REFERENCES Departments)
```

Review: Key Constraints

- Each dept has at most one manager, according to the *key constraint* on Manages.



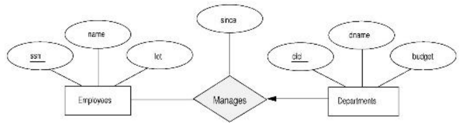


How to translate
Into relational model?

Translating ER Diagrams with Key Constraints

- Map relationship to a table:
  - Note that did is the key now!
  - Separate tables for Employees and Departments.

```
CREATE TABLE  Manages (
   ssn  CHAR(11), /* null-able*/
   did  INTEGER,
   since  DATE,
   PRIMARY KEY  (did),
   FOREIGN KEY (ssn) REFERENCES Employees,
   FOREIGN KEY (did) REFERENCES Departments)
```



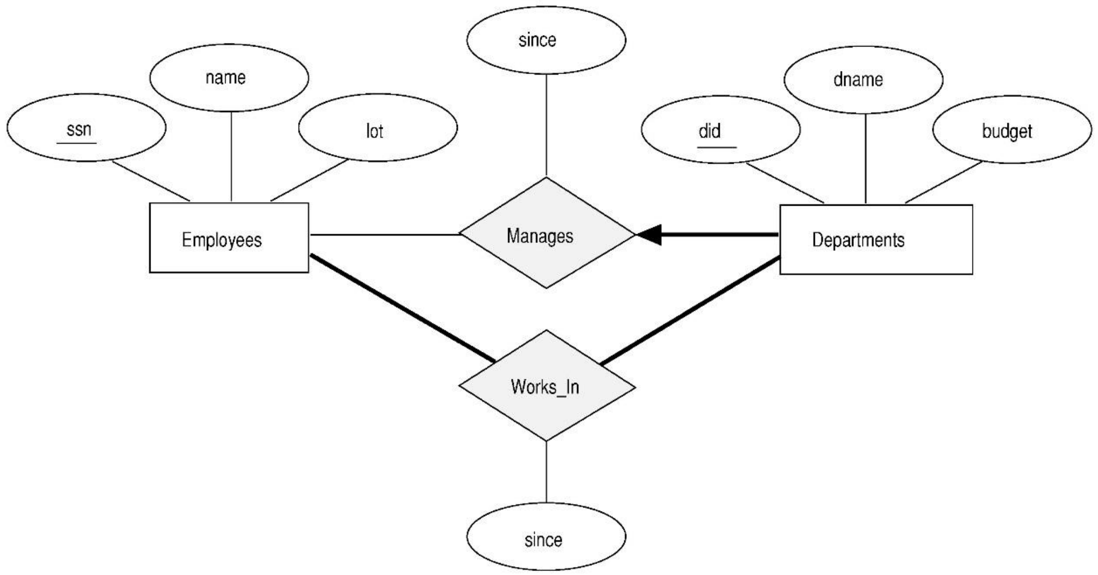
- Since each department has a unique manager, we could instead combine Manages and Departments.

```
CREATE TABLE  Dept_Mgr (
   did  INTEGER,
   dname  CHAR(20),
   budget  REAL,
   ssn  CHAR(11), /* null-able*/
   since  DATE,
   PRIMARY KEY  (did),
   FOREIGN KEY (ssn) REFERENCES Employees)
```

- But, both approaches **do not ensure** that a manager is appointed for each department. (total participation of each department)

Review: Participation Constraints

- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).
    - ✓ Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)



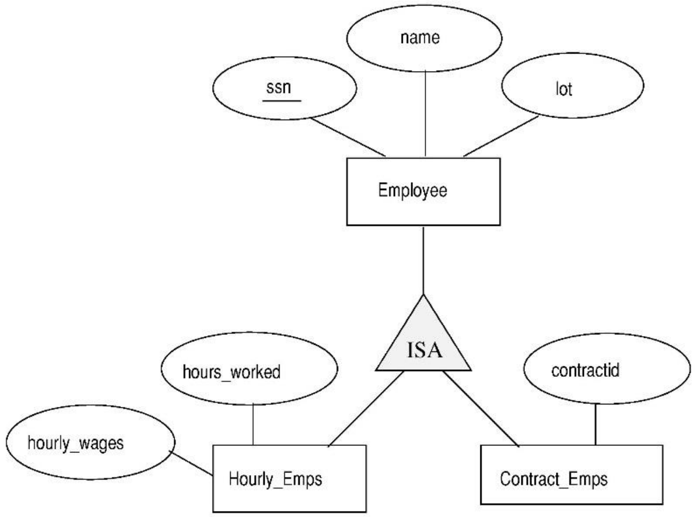
Participation Constraints in SQL

- We can capture participation constraints involving one entity set in a binary relationship

```
CREATE TABLE  Dept_Mgr (
   did  INTEGER,
   dname  CHAR(20),
   budget  REAL,
   ssn  CHAR(11) NOT NULL,
   since  DATE,
   PRIMARY KEY  (did),
   FOREIGN KEY  (ssn) REFERENCES Employees
      ON DELETE NO ACTION) /* what if CASCADE? */
```

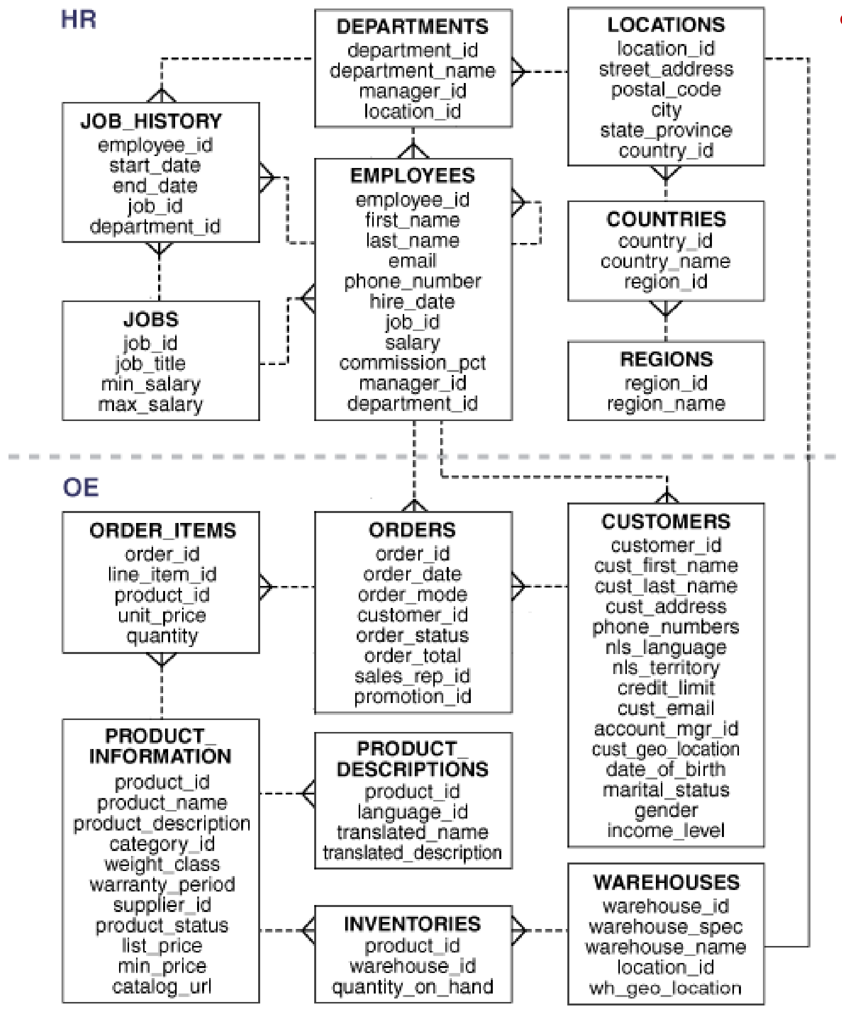
- But little else ☹ (without resorting to CHECK or ASSERTION).
  - e.g. how to express total participation in Work-In table?
  - Refer to 3.5.4 for details

3.5.5 Translating Class Hierarchy



- EMP, H_EMP (ssn, name, lot, h_wages, h_worked), C_EMP (…)
- EMP, H_EMP(ssn, h_wages,h_worked), C_EMP
- EMP (…, emp_type, h_wages, h_worked, contractid)
- Pros and Cons?

Schema Example: HR and OE Schema in Oracle



- How to use?
  - Sample schemas in Oracle
    - ✓ Description: 12c
    - ✓ Manual installation: see github
  - How to activate HR/OE user?

```
Conn / as sysdba;

ALTER USER hr/oe IDENTIFIED BY hr/oe ACCOUNT
UNLOCK;

GRANT CONNECT, RESOURCE to hr/oe;
```

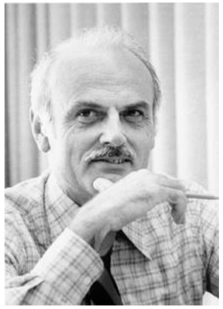
Ch 19. Schema Refinement and Normal Forms

Sang-Won Lee

http://icc.skku.ac.kr/~swlee

SKKU VLDB Lab. & SOS

(http://vldb.skku.ac.kr/)



19.1~2 The Evils of Redundancy

- *Redundancy* is a root cause of several problems in relational schemas:

ssn	*name*	*lot*	*rating*	*hourly_wages*	*hours_worked*
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Figure 19. 1

- Problems due to rating and hourly_wages
  - Redundant storage
  - *Update anomaly*: Can we change W in just the 1st tuple of SNLRWH?
  - *Insertion anomaly*: What if we want to insert an employee and don't know the hourly wage for his rating?
  - *Deletion anomaly*: If we delete all employees with rating 5, we lose the information about the wage for rating 5!

- Is the null value helpful? Not correct solution! Then, what solution?

19.1.2 Decomposition

- Will 2 smaller tables be better? Any anomaly?

ssn	*name*	*lot*	*rating*	*hours_worked*
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

rating	*hourly_wages*
8	10
5	7

Figure 19. 2

Decomposition of a Relation Scheme

Given relation R (*A1, ... An).*

- A *decomposition* of R consists of replacing R with two or more relations such that:
  - Each new relation scheme contains a subset of the attributes of R (and no attributes that do not appear in R), and
  - Every attribute of R appears as an attribute of one of the new relations.
- Intuitively, decomposing R means we will store instances of the relation schemes produced by the decomposition, instead of instances of R.

- E.g., SNLRWH → SNLRH and RW.
  - cf. With reasonable ER modeling, SNLRWH can be avoided

19.1.3 Problems with Decompositions

- Potential problems with decomposition
  1. Performance: Some queries become more expensive!
     - ✓ e.g., How much did sailor Joe earn? (salary = W*H)
  2. Information loss: Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!
     - ✓ Fortunately, not in the SNLRWH example.
  3. Dependency loss: Checking some dependencies may require joining the instances of the decomposed relations.
     - ✓ Fortunately, not in the SNLRWH example.
- Tradeoff: Must consider these issues vs. redundancy.

Decomposition

- Decomposition should be used judiciously:
  - Do we need to decompose a relation?
    - Normal forms and normalization
  - How to prevent the problems of decomposition (if any) ?
    - Lossless-join; dependency preserving; performance

- Normal forms: 1st, 2nd, BC(Boyce-Codd), 3rd, 4th, 5th
  - Powerful concept and design guideline based on the theory of functional dependency
  - *The concept of functional dependencies* is used to identify schemas with problems and to suggest refinements.
  - Refer to 19.3 – 5 for details

19.2 Functional Dependencies (FDs)

- A functional dependency X → Y holds over relation R if, for every allowable instance *r* of R (X and Y are *sets* of attributes.)
  - $t1 \in r$, $t2 \in r$, $\pi_X(t1) = \pi_X(t2)$ implies $\pi_Y(t1) = \pi_Y(t2)$
  - i.e., if t1.x = t2.x, then t1.Y = t2.Y
  - We say that X functionally determines Y

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

Figure 19. 3

- What FDs hold in Figure 19.3?
  - A → B?; D → C?; AB → C?; AB → CD?

Functional Dependencies (FDs)

- An FD is a statement about *all* allowable relations.
  - A kind of integrity constraint (IC) given by database designers! Where does it come from? real world semantic!
  - Given some allowable instance *r1* of R, we can check if it violates some FD *f* (*how to check in SQL?*), but we cannot tell if *f* holds over R!

- Primary key is a special kind of FD: K $\rightarrow$ R

Example: Constraints on Entity Set

Hourly_Emps (<u>*ssn*</u>, *name, lot, rating, hrly_wages*, *hrs_worked*)

- <u>*Notation*</u>: denote Hourly_Emps as SNLRWH
  - *Set* of attributes {S,N,L,R,W,H}.
  - Sometimes, we will refer to all attributes of a relation by using the relation name. (e.g., Hourly_Emps for SNLRWH)

- Some FDs on Hourly_Emps:
  - *ssn* is the key: S $\rightarrow$ SNLRWH
  - *rating* determines *hrly_wages*: R $\rightarrow$ W

19.3 Reasoning About FDs

- Given some FDs, we can usually infer additional FDs:
  - *ssn* → *did*, *did* → *lot* implies *ssn* → *lot*
- FD *f* is *implied by* a set of FDs *F* if *f* holds whenever all FDs in *F* hold.
  - F^+ = *closure of F* = set of all FDs implied by *F*.
- **Armstrong's Axioms** (X, Y, Z are sets of attributes):
  - *Reflexivity*: If X $\subseteq$ Y, then Y → X (trivial)
  - *Augmentation*: If X → Y, then XZ → YZ for any Z
  - *Transitivity*: If X → Y and Y → Z, then X → Z
- These axioms are ***sound*** and ***complete*** inference rules for FDs!

Reasoning About FDs (Contd.)

- Additional rules (derived from Armstrong axioms):
  - *Union*: If X → Y and X → Z, then X → YZ
  - *Decomposition*: If X → YZ, then X → Y and X → Z

- Example: Contracts(*cid,sid,jid,did,pid,qty,value*), and:
  - C is the key: C → CSJDPQV
  - Project purchases each part using single contract: JP → C
  - Dept purchases at most one part from a supplier: SD → P

    (* See Fig 2.19 and Ch 19.7.2;

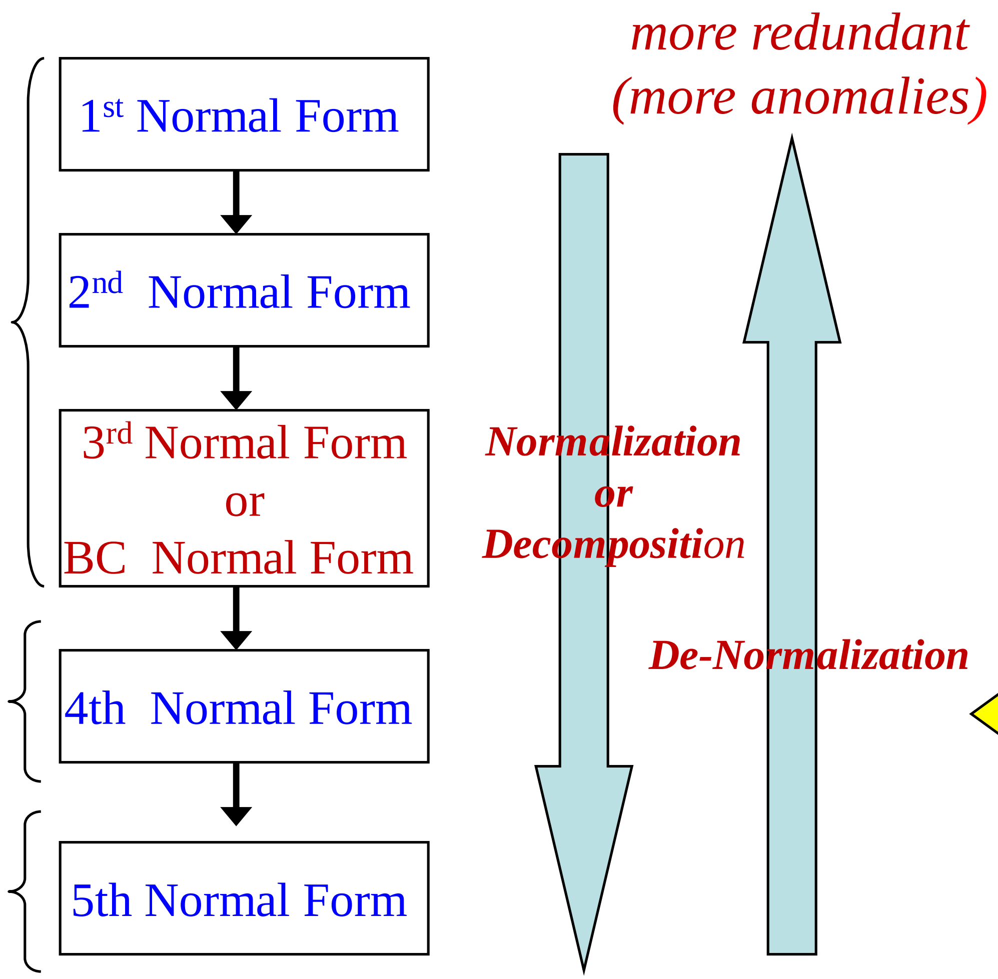
    FDs are used to represents complex semantics in relationships)
  - JP → C, C → CSJDPQV imply JP → CSJDPQV
  - SD → P implies SDJ → JP
  - SDJ → JP, JP → CSJDPQV imply SDJ → CSJDPQV

Can we express these constraints In ER model? No

Reasoning About FDs (Contd.)

- Computing the closure of a set of FDs can be expensive. (Size of closure is *exponential* in # of attributes!)
- Typically, we just want to check if a given FD *X* → *Y* is F^+.
- An efficient check:
  - Compute <u>*attribute closure*</u> of X (denoted X^+) w.r.t *F:*
    - ✓ Set of all attributes A such that X → A is in F^+
    - ✓ There is a linear time algorithm to compute this. (Fig. 19.4)
  - Check if Y is in X^+
- Does F = {A → B, B → C, C D → E } imply A → E?
  - i.e, is A → E in the closure F^+ ? Equivalently, is E in A^+ ?

Big Picture of FD & Normalization


1st Normal Form
2nd Normal Form
3rd Normal Form or BC Normal Form
4th Normal Form
5th Normal Form
more redundant (more anomalies)
Normalization or Decomposition
De-Normalization

- When to decompose?
  - Normal forms and decomposition
- What problems of decomposition?
  - Information loss? dependency preserving? performance?

Theorical Tools & Concepts:

Functional Dependency(19.2),
Armstrong's Axiom(19.3),
Closure of FDs(19.3),
Lossless Join Decomposition(19.5),
Dependency Preserving(19.5),
Decomposition Algorithms(19.6),
Minimal Cover(19.6.2),
Multivalued Dependency(19.8),
Join Dependency(19.8)

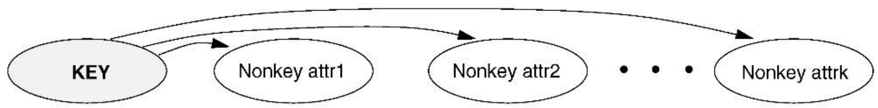
19.4 Normal Forms

- Returning to the issue of schema refinement, the first question is whether any refinement is needed!
- If **a relation** is in a certain *normal form* (BCNF, 3NF etc.), it is known that certain kinds of problems are avoided/minimized. This can be used to help us decide whether decomposing the relation will help.
- Role of FDs in detecting redundancy:
  - Consider a relation R with 3 attributes, ABC.
    - ✓ No FDs hold: There is no redundancy here.
    - ✓ Given A $\rightarrow$ B: Several tuples could have the same A value, and if so, they'll all have the same B value!

19.4.1 Boyce-Codd Normal Form (BCNF)

- R with FDs *F* is in BCNF if, for all X → A in F^+
  - $A \in X$ (called a *trivial* FD), or
  - X contains a key for R.

Figure 19. 5



- That is, R is in BCNF if the only non-trivial FDs that hold over R are key constraints.
  - No redundancy in R that can be detected using FDs alone.(cf. MVD, JD)
  - If we are shown two tuples that agree upon the X value, we can infer the A value in one tuple from the A value in the other.
  - If example relation is in BCNF, the 2 tuples must be identical (since X is a key). (Table in Fig 19.6 != BCNF)
  - *In ER terminology, each tuple can be taught of as an entity or relationship.*

X	Y	A
x	y_1	a
x	y_2	?

Figure 19.6

Third Normal Form (3NF)

- R with FDs *F* is in 3NF if, for all X → A in F^+
  - $A \in X$ (called a *trivial* FD), or
  - X contains a key for R, or
  - A is part of some key for R (Case 2 in Figure 19.8)
    (* less restricted than BCNF)
    (** Minimality of a key is crucial in third condition above!)
- If R is in BCNF, obviously in 3NF.
- If R is in 3NF, some redundancy is possible. It is a compromise, used when BCNF not achievable (e.g., no ``good'' decomposition, or performance considerations).
  - *Lossless-join, dependency-preserving decomposition of R into a collection of 3NF relations* ***always possible****.*

Normalization Theory vs. ER Modeling

- Problems with normalization theory
  - How do DBAs get an initial set of relations?
  - Too difficult for DBAs to understand

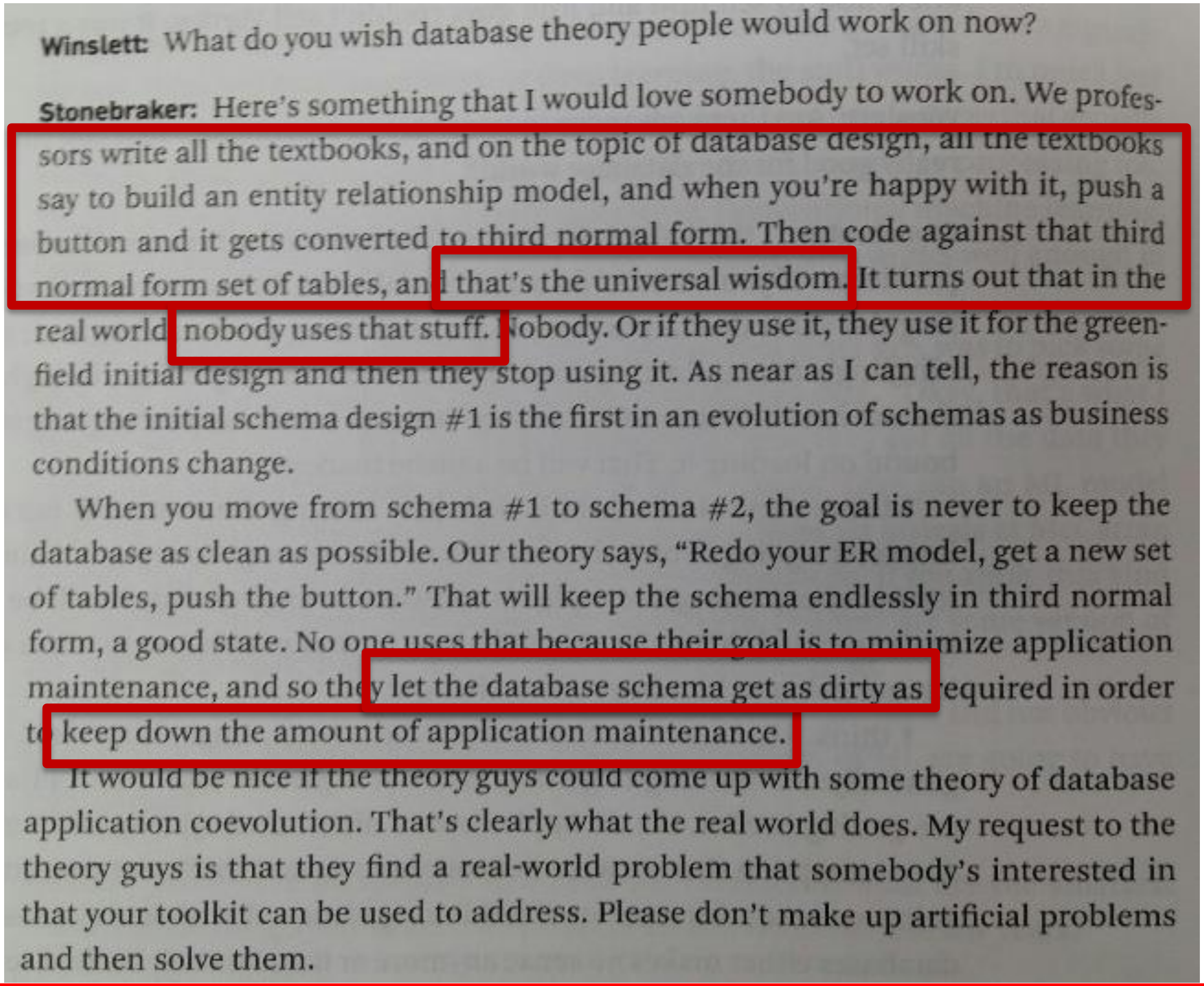
  ➔ DB design using normalization was “dead in the water”

- ER model
  - Methodology for constructing an initial ER diagram
  - Convert ER diagram into 3NF relations (automatically)

  * source: Stonebraker et al. “What goes around comes around?”

- ER (high level lang.) vs. Normalization theory (machine lang.)

Pragmatic Wisdom (Stonebraker)



Winslett: What do you wish database theory people would work on now?

Stonebraker: Here's something that I would love somebody to work on. We professors write all the textbooks, and on the topic of database design, all the textbooks say to build an entity relationship model, and when you're happy with it, push a button and it gets converted to third normal form. Then code against that third normal form set of tables, and that's the universal wisdom. It turns out that in the real world nobody uses that stuff. Nobody. Or if they use it, they use it for the green-field initial design and then they stop using it. As near as I can tell, the reason is that the initial schema design #1 is the first in an evolution of schemas as business conditions change.

When you move from schema #1 to schema #2, the goal is never to keep the database as clean as possible. Our theory says, "Redo your ER model, get a new set of tables, push the button." That will keep the schema endlessly in third normal form, a good state. No one uses that because their goal is to minimize application maintenance, and so they let the database schema get as dirty as required in order to keep down the amount of application maintenance.

It would be nice if the theory guys could come up with some theory of database application coevolution. That's clearly what the real world does. My request to the theory guys is that they find a real-world problem that somebody's interested in that your toolkit can be used to address. Please don't make up artificial problems and then solve them.

How to Create and Run a Stonebraker Startup— The Real Sto

Andy Palmer and Mike Stonebraker at the 2014 Turing Award Ce
Photo credit: Amy Palmer.

An Extraordinary Achievement. An Extraordina

Normalization vs. NoSQL

- E.g. Amazon user profile – **key-value store/document**
  - Extremely de-normalized scheme

- Trade-off between normalization vs. Key-value document
  - N small IO vs. 1 Big(?) IO
  - $ / Byte: redundancy ?
  - Delta change

Ch 25.2 Multi-dimensional Data Model

Sang-Won Lee

http://icc.skku.ac.kr/~swlee

SKKU VLDB Lab. & SOS

(http://vldb.skku.ac.kr/)

Multidimensional Data Model

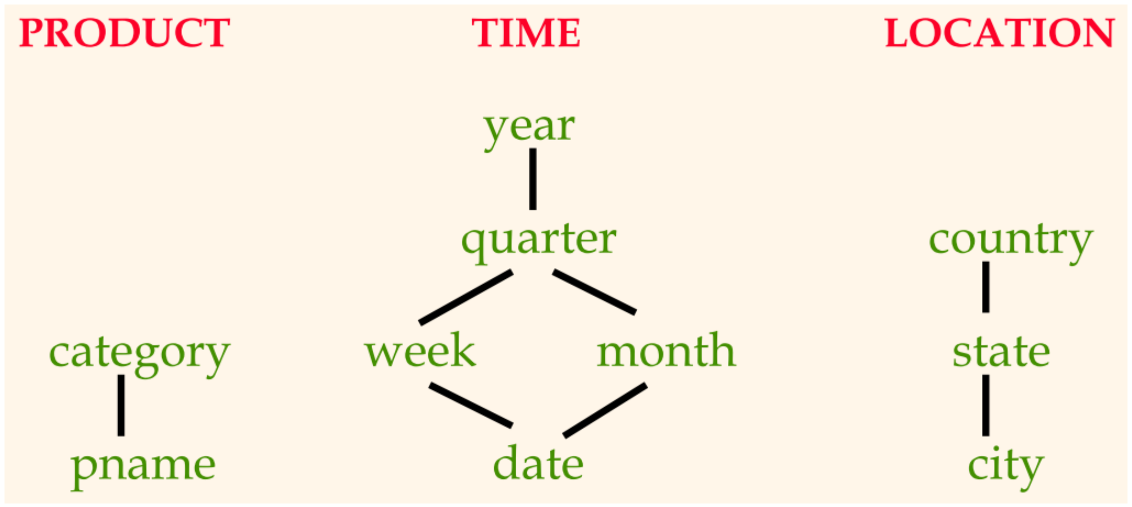
- **Fact table**: a set of tuples with numeric **measures** with **a set of dimensions**
  - E.g. measure: **Sales**
  - Dimensions:
    - ✓ **Product** (key: pid)
    - ✓ **Location** (key: locid)
    - ✓ **Time** (key: timeid)

PID	LOCID	TIMEID	SALES
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35

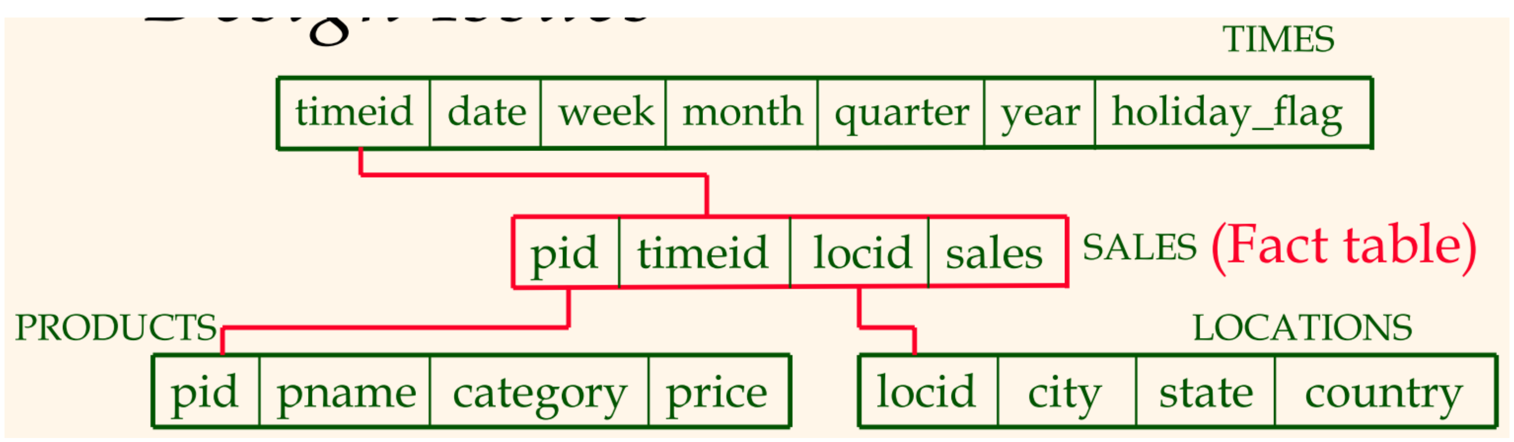
Dimension Hierarchy

- For each dimension, the set of values can be organized in a hierarchy
  - Each dimension is usually modeled with its dedicated dimension table(s) (see the example in next slide.)

- The user can write queries at any combination of levels in multiple dimensions



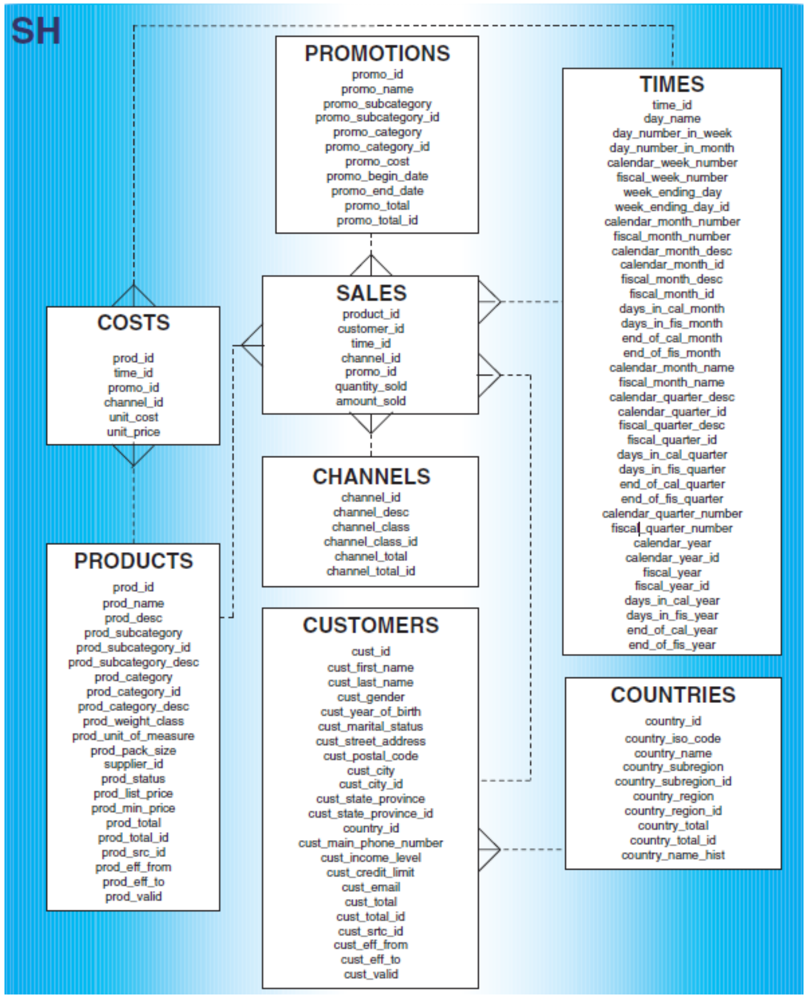
Star Schema



- Fact table in BCNF; dimension tables are de-normalized (WHY?)
  - Dimension tables are small; update/insert/delete are rare. So, anomalies due to de-normalizations are less important, instead important is the query performance.

- This kind of schema is very common in OLAP applications, and is called a **star** schema; computing the join of all these relations is called a **star join**.

Star Schema Example: SH Schema in Oracle



- How to use?
  - Sample schemas in Oracle
    - ✓ Description: 12c
    - ✓ Manual installation: see github
  - How to activate SH user?

```
Conn / as sysdba;
ALTER USER sh IDENTIFIED BY sh ACCOUNT UNLOCK;
GRANT CONNECT, RESOURCE to sh;
```

- Sample queries against SH schema
  - Chap 20 ~ 22 @ Oracle Data Warehousing Guide

DB Design - Conclusion

- Ch 2. The Entity-Relationship Model
- Ch 3.5 Translating ER model to relational schema
- Ch 19 Normalization
- Ch 25.2 Multi-dimensional data model