Database Design

October 24, 2008

Outline

E-R diagrams

- Represent logical structure simply, clearly
  - **Rectangles**: entity sets
  - **Ellipses**: attributes
  - **Diamonds**: relationship sets
  - **Lines**: linking elements
  - **Double ellipse**: multi-valued attributes
  - **Dashed ellipse**: derived attributes
  - **Double lines**: total participation

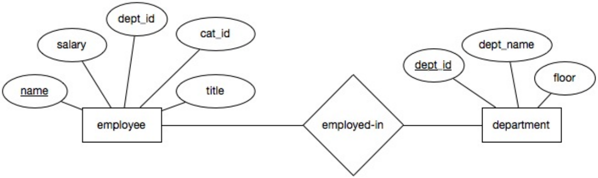


Figure: Entity-Relationship diagram

Cardinality Representation

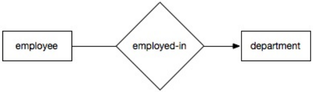


Figure: Many to one

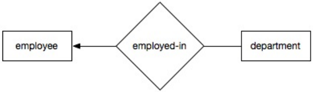


Figure: One to many

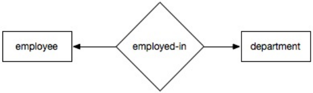


Figure: One to one

Attribute for relationship set

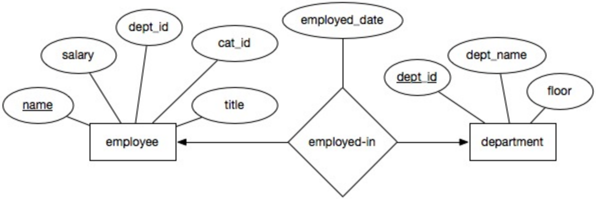


Figure: Attribute attached to a relationship set

Possible attribute types

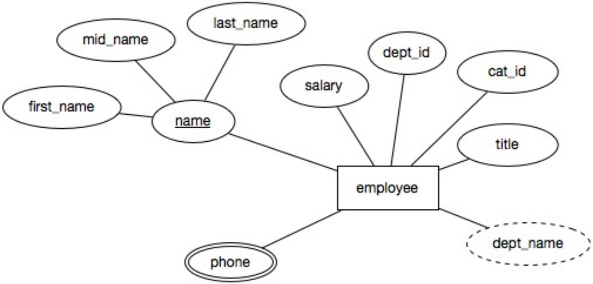


Figure: Composite, multi–valued, and derived attributes

Specifying roles

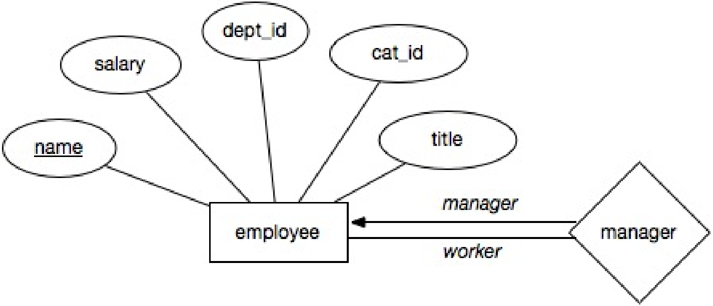


Figure: Role indicators

Total participation

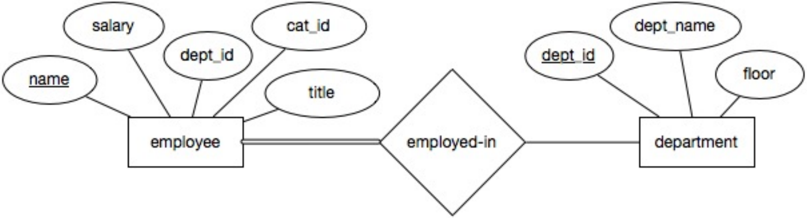


Figure: Total participation of employee entity set

Specifying cardinality limits

- Use numerical range for precise specification of cardinality
- *min* . . . *max*
- $1 \ldots * \implies$ double line (total participation)

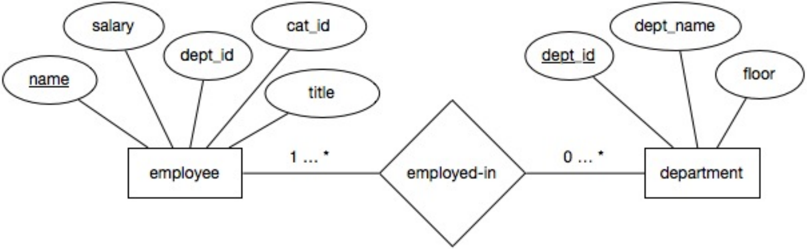


Figure: Cardinality limits on the relationship set

Outline

Attributes vs. Entity Sets

- Can use either in situations

Attributes vs. Entity Sets

- Can use either in situations

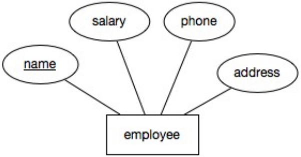


Figure: Phone as an attribute

Phone as a new relation

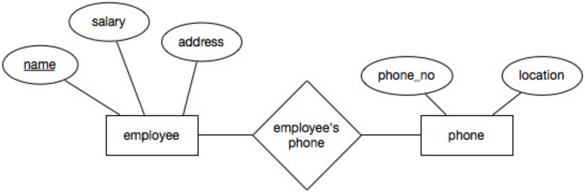


Figure: Phone as an entity

Phone as a new relation

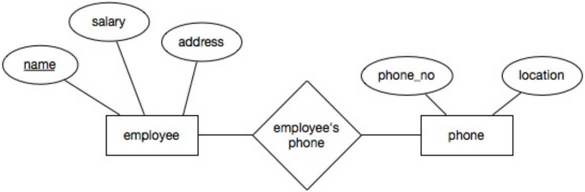


Figure: Phone as an entity

- If graduating to an entity:
  - remove *phone* from *employee*'s attribute list
  - Add entity set *phone* with attributes *phone_no* & *location*
  - Add relationship set *employee's phone* between the relations

Differences between the approaches

- Multiple values
  - If attribute $\Longrightarrow$ only 1 phone no.
    - (unless multi–valued)
- Main difference: entity set approach is *more general*
  - separate entity allows more information
  - Also, > 1 employee can share 1 phone

Entity Sets vs. Relationship Sets

- An object may be represented as either

Entity Sets vs. Relationship Sets

- An object may be represented as either
- Consider a *project* object

Entity Sets vs. Relationship Sets

- An object may be represented as either
- Consider a *project* object
- Easily modeled as an entity set

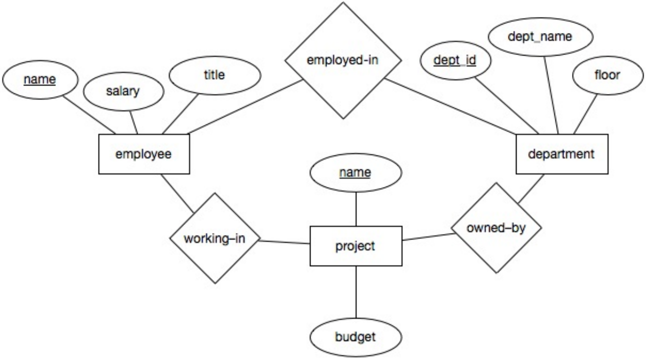


Figure: project modeled as an entity set

Modeling project as a relationship set

- May be modeled as:

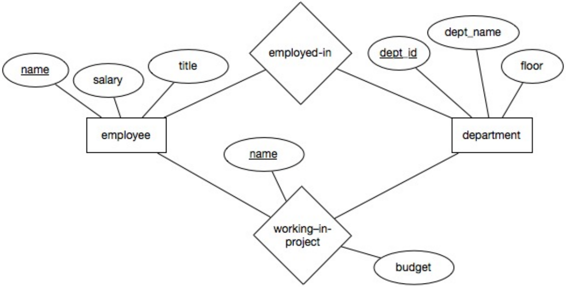


Figure: project modeled as a relationship set

Modeling project as a relationship set

- May be modeled as:

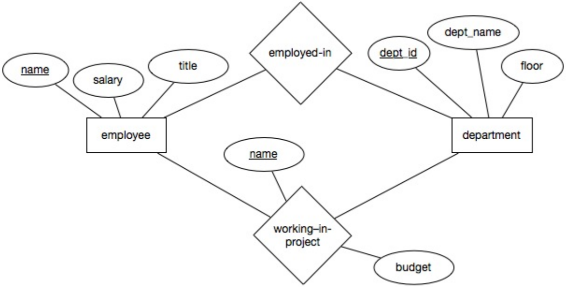


Figure: project modeled as a relationship set

- Works for strict 1-to-1 mapping
- What happens for two employees working on same project?
  - Or for one project shared by two departments

Modeling project as a relationship set

- May be modeled as:

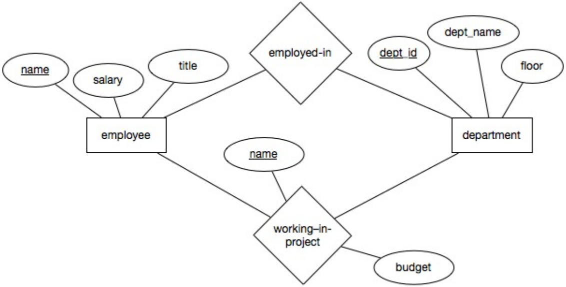


Figure: project modeled as a relationship set

- Works for strict 1-to-1 mapping
- What happens for two employees working on same project?
  - Or for one project shared by two departments
- Issues:
  - Duplication $\Longrightarrow$ storage wastage
  - Updates need to update twice; inconsistencies

Solutions

- **Normalization theory**
- Model verbs as relationship sets; nouns as entity sets

Outline

Specialization

- Subgrouping of entity sets
    - *Person* → *Employee*, *Customer*
- Specialization: defining subgroupings

Specialization

- Subgrouping of entity sets
  - *Person → Employee, Customer*
- Specialization: defining subgroupings

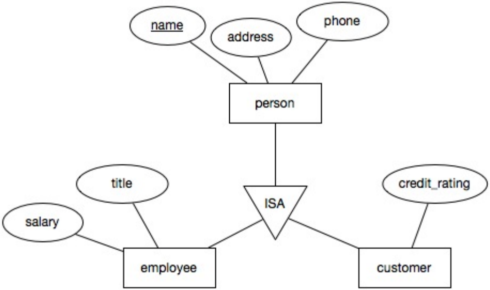


Figure: Specialization on Person set

- *Higher* and *lower* entity sets
  - **superclass**, **subclass**
- Attribute inheritance

Another Specialization

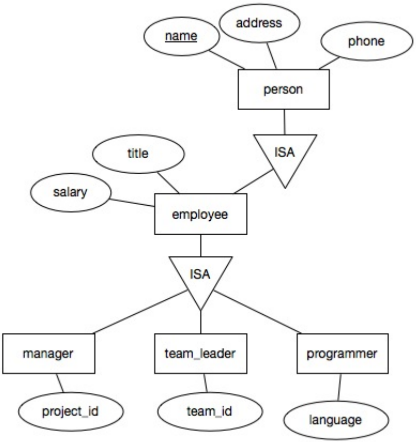


Figure: Specialization on Employee set

- **Bottom-up** approach: *Generalization*

Aggregation

- For modeling relationship between relationships

Aggregation

- For modeling relationship between relationships
- For e.g., *manager* related to all entity sets in a relationship
    - Quaternary: $(manager, employee, project, department)$

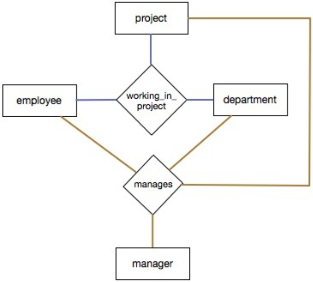


Figure: Tertiary & Quaternary Relationship Sets

- Duplication of values

An alternative

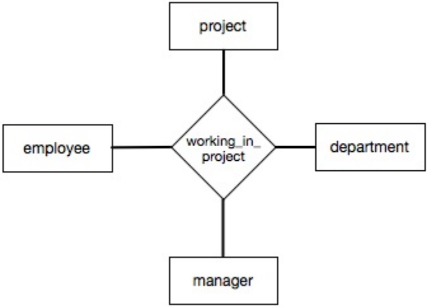


Figure: An alternative

An alternative

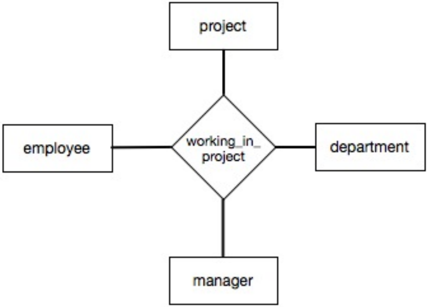


Figure: An alternative

- But, a (*employee*, *project*, *department*) may not have a *manager* assigned

Another alternative

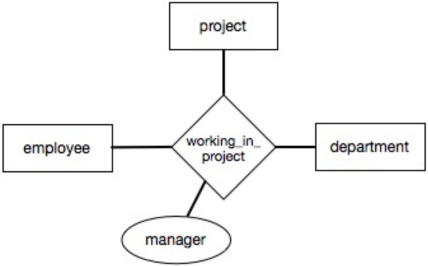


Figure: Another alternative with manager as an attribute

- Only if *manager* is a single value

Aggregation

- **Aggregation**: Relationships are treated as entities
- *working–in–project*(*employee*, *project*, *department*) → relationship set + entity
- *manages* → relationship set

Aggregation

- **Aggregation**: Relationships are treated as entities
- *working–in–project*(*employee*, *project*, *department*) → relationship set + entity
- *manages* → relationship set

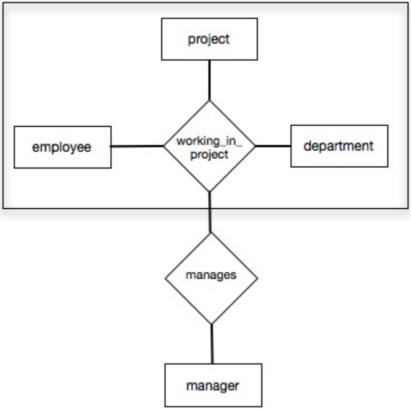


Figure: E-R diagram with aggregation

Need for weak entity sets

- Some sets have undefinable primary keys
- Consider *payment* entity set, related to *loan*
    - $payment(payment\_id, amount)$

Need for weak entity sets

- Some sets have undefinable primary keys
- Consider *payment* entity set, related to *loan*
  - $payment(payment\_id, amount)$

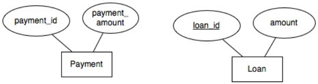


Figure: Payment entity set

Need for weak entity sets

- Some sets have undefinable primary keys
- Consider *payment* entity set, related to *loan*
  - $payment(payment\_id, amount)$

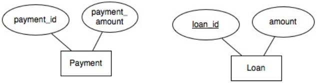


Figure: Payment entity set

- Entity in *payment* are not unique

Weak Entity Sets

- Weak Entity Sets → no primary keys
- *payment* is **existence dependent** on *loan*, the *identifying* set
- *loan* **owns** the weak set *payment*
- Each loan entity related to a set of payment entities
  - *payment_id* : **discriminator**
  - *(loan_id, payment_id)* : primary key for *payment*

Weak Entity Sets

- Weak Entity Sets → no primary keys
- *payment* is **existence dependent** on *loan*, the *identifying* set
- *loan* **owns** the weak set *payment*
- Each loan entity related to a set of payment entities
    - *payment_id* : **discriminator**
    - *(loan_id, payment_id)* : primary key for *payment*

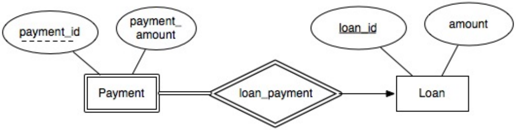


Figure: E–R diagram with a weak entity set

Outline

Gathering Data Requirements

- **Branches**: located in a *city*
- **Customers**: identified by *customer_id*
  - name, street, city
  - **accounts** and **loans**
  - associated with a **banker**
- **Employees**: idenitified by *employee_id*
  - name, phone no., dependent name
  - employee_id of the manager
  - start date
- Savings and checking **accounts**
  - Related to ≥ 1 customer
  - Unique account number
  - balance, last date of access by each customer
  - savings → interest rate; checking → overdrafts recorded
- **Loan**: associated with a branch
  - identified by unique loan_id
  - payment: amount, date, id

Initial Entity Sets

- *branch*: (*branch_name*, *branch_city*, *assets*)
- *customer*: (*customer_id*, *customer_name*, *customer_street*, *customer_city*) ... *banker_name* ?
- *employee*: *employee_id*, *employee_name*, *phone_no*, *salary*, *manager*
  - multi–valued *dependent_name*
  - base: *start_date*, *employment_length*
- *savings*, *account*: both have *account_number*, *balance*
  - *savings*: *interest_rate*
  - *checking*: *overdraft_amount*
- *loan*: *loan_number*, *amount*, *original_branch*
- *loan_payment*: weak entity set
  - *payment_number*, *payment_date*, *payment_amount*

E–R Diagram for entity sets

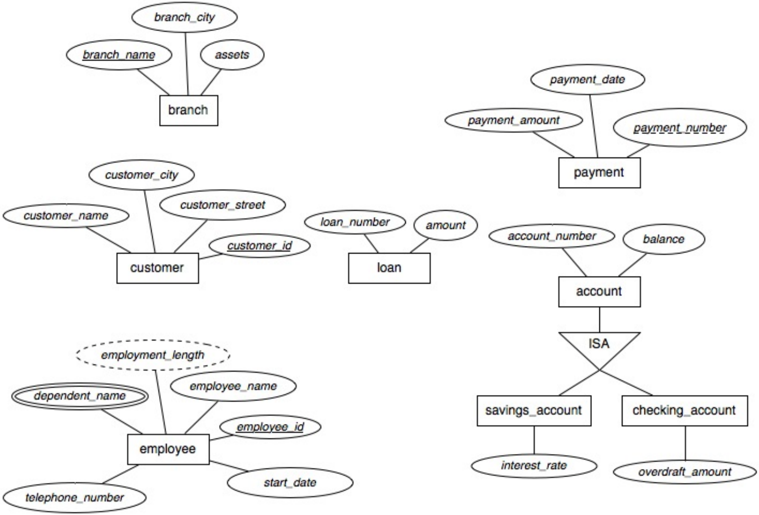
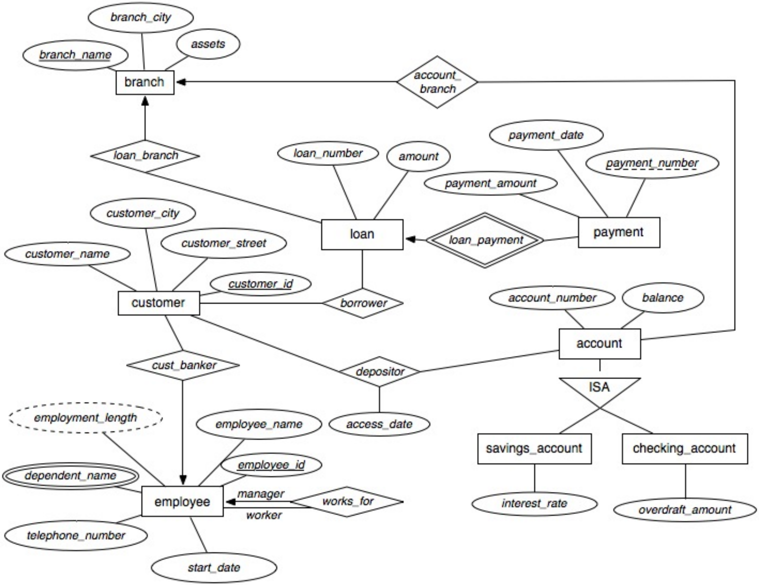


Figure: E–R Diagram for entity sets

Relationship sets

- *borrower*: *customer* and *loan*; many–to–many
- *loan_branch*: *loan* and *branch*; many–to–one
  - replaces the attribute *original_branch* of *loan*
- *loan_payment*: *loan* and *payment*; one–to–many
  - documents that loan payments are made
- *depositor*: *customer* and *account*; many–to–many
  - indicates that a customer owns an account
  - with attribute *access_date*
- *cust_banker*: *customer* and *employee*; many–to–one
  - the customer is advised by a bank employee
  - replaces attribute *banker_name* of *customer*
- *works_for*: between *employee*s; one–to–many
  - role indicators (*manager*, *worker*)
  - replaces *manager* attribute of *employee*

E–R diagram with Relationship Set



Outline

Representation of Strong Entity Sets

- Let E be entity set; descriptive attributes $a_1, a_2, \ldots, a_n$
- Represented by schema E_s with n attributes
- Each entity corresponds to tuple in E_s
    - will discuss multi–valued and composite attributes later
- *primary key* remains the same
- E.g., entity set *loan* becomes a schema

$$loan = (\underline{loan\_number}, amount)$$

Representation of Weak Entity Sets

- Let A be a weak entity set; attributes $a_1, a_2, \ldots, a_m$
- B be the owner strong entity set of A; **primary key** attributes $b_1, b_2, \ldots, b_n$
- $A_s \equiv a_1, a_2, \ldots, a_m \cup b_1, b_2, \ldots, b_n$
- $primary\_key(A_s) \equiv primary\_key(B) \ \cup discriminator(A)$
- Foreign key constraints for $b_1, b_2, \ldots, b_n$ in A_s
- $payment_s = (\underline{loan\_number, payment\_number}, payment\_date, payment\_amount)$

Representation of Relationship Sets

- For relationship set R, let $a_1, a_2, \ldots, a_n$ be the primary keys of all entity sets
- $b_1, b_2, \ldots, b_m$ be the descriptive attributes of R
- Form a new relation schema R_s with attributes

$$\{a_1, a_2, \ldots, a_n\} \cup \{b_1, b_2, \ldots, b_m\}$$

- Primary key is the same as that for R:
  - many-to-many: $primary\_key(E_1) \cup primary\_key(E_2)$
  - one-to-many: $primary\_key(E_2)$
  - many-to-one: $primary\_key(E_1)$
  - one-to-one: $primary\_key(E_1)$ or $primary\_key(E_2)$
- Create the necessary foreing key constraints
- For e.g., *borrower* involves
  - *customer* with primary key *customer_id*
  - *loan* with primary key $loan_number$
- $borrower_s$ schema $\equiv$ $(\underline{customer\_id, loan\_number})$
- Many–many relationship ⇒ both attributes are in primary key
- Foreign key constraints for both attributes

Redundancy of Some Relational Schemas

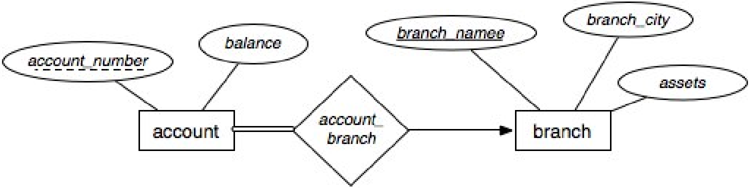
- Consider *loan_payment* relationship set
- $PK(loan) = loan\_number$, $PK(payment) = loan\_number, payment\_number$
- $\therefore$ $loan\_payment_s$ will have attributes $loan\_number, payment\_number$
- $\therefore$, duplication for $loan\_number, payment\_number$ values
- $\therefore$, *loan_payment* is redundant
- Usually the schema for a weak relationship set is redundant
  - not included in final relational DB design

Combination of Schemas

- Consider entity sets A, B; relationship set AB
- will produce corresponding 3 schemas
  - A_s, AB_s, B_s
- If AB is many–to–one; A **totally** participates:
  - Schemas A_s and AB_s can be combined

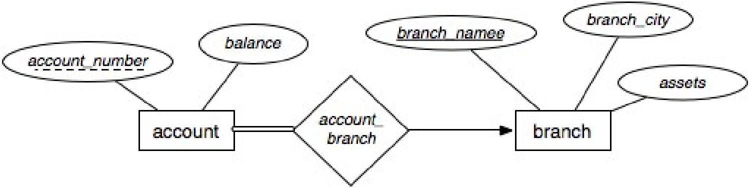
Example of Schema combination

- Consider:



Example of Schema combination

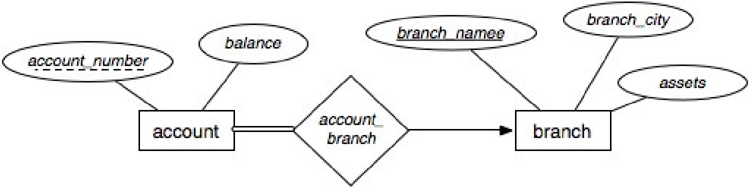
- Consider:



- Every *account* entity participates in *account_branch*
- Can combine *account* with *account_branch* Schemas:
  - *account* = (*account_number*, *balance*, *branch_number*)
  - *branch* = (*branch_name*, *branch_city*, *assets*)
- Primary key remains the same (*account_number*)
- Only one, the remaining (*branch_name*), foreign key contraint
- Why many–to–one?

Example of Schema combination

- Consider:



- Every *account* entity participates in *account_branch*
- Can combine *account* with *account_branch* Schemas:
  - *account* = (*account_number*, *balance*, *branch_number*)
  - *branch* = (*branch_name*, *branch_city*, *assets*)
- Primary key remains the same (*account_number*)
- Only one, the remaining (*branch_name*), foreign key contraint
- Why many–to–one?
  - One–to–one also (combine with A **or** B)

Composite and Multi–Valued Attributes

- Composite attribute is expanded into multiple attributes
    - original attribute is discarded
- New relation is created for a multi–valued attribute
- If *M* is multi–valued:
    - New relation *R* is created
    - Attributes
        1. *A*: same as M
        2. primary keys of M's entity set (act as foreign key)
    - Primary key → all attributes
    - Create foreign key via shared attribute