# Converting E-R Diagrams to Relational Model

Dr. M. Brindha
Assistant Professor
Department of CSE
NIT, Trichy-15

# E-R Diagrams

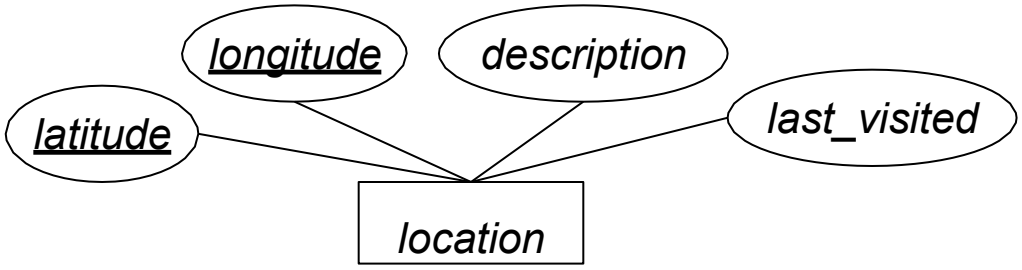
- Need to convert E-R model diagrams to an iimplementation schema
- Easy to map E-R diagrams to relational model, and then to SQL
  - Significant overlap between E-R model and relational model
  - Biggest difference is E-R composite/multivalued attributes, vs. relational model atomic attributes
- Three components of conversion process:
  - Specify schema of relation itself
  - Specify primary key on the relation
  - Specify any foreign key references to other relations

# Strong Entity-Sets

- Strong entity-set *E* with attributes $a_1$, $a_2$, …, $a_n$
  - Assume simple, single-valued attributes for now
- Create a relational schema with same name *E*, and same attributes $a_1$, $a_2$, …, $a_n$
- Primary key of relational schema is same as primary key of entity-set
  - No foreign key references for strong entity-sets
- Every entity in *E* represented by a tuple in corresponding relation

# Entity-Set Examples

- **Geocache location E-R diagram:**

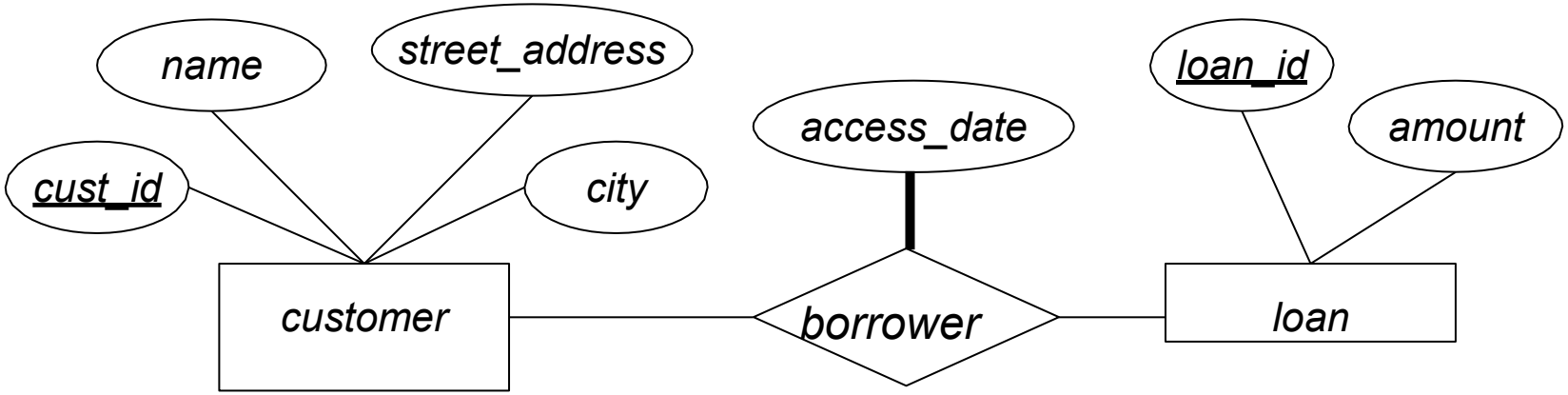



  - **Entity-set named *location***
- **Convert to relation schema:**

  ***location(latitude, longitude, description, last_visited)***

# Entity-Set Examples (2)

- E-R diagram for customers and loans:




- Convert *customer* and *loan* entity-sets:

  *customer*(*cust_id*, *name*, *street_address*, *city*)

  *loan*(*loan_id*, *amount*)

# Relationship-Sets

- Relationship-set *R*
  - Assume all participating entity-sets are strong entity-sets, for now
  - $a_1, a_2, \ldots, a_m$ is the union of all participating entity-sets' primary key attributes
  - $b_1, b_2, \ldots, b_n$ are descriptive attributes on *R* (if any)
- Relational schema for *R* is:
  - $\{a_1, a_2, \ldots, a_m\}$ ( $\{b_1, b_2, \ldots, b_n\}$
- $\{a_1, a_2, \ldots, a_m\}$ is a superkey, but not necessarily a candidate key
  - Primary key of *R* depends on *R*'s mapping cardinality

# Relationship-Set Primary Keys

- For binary relationship-sets:
  - e.g. between strong entity-sets *A* and *B*
  - If many-to-many mapping, union of all entity-set primary keys becomes primary key of relationship-set
    - *primary_key*(*A*) ∪ *primary_key*(*B*)
  - If one-to-one mapping, either entity-set's primary key is acceptable
    - *primary_key*(*A*), or *primary_key*(*B*)
    - Should enforce candidate key constraint for each!

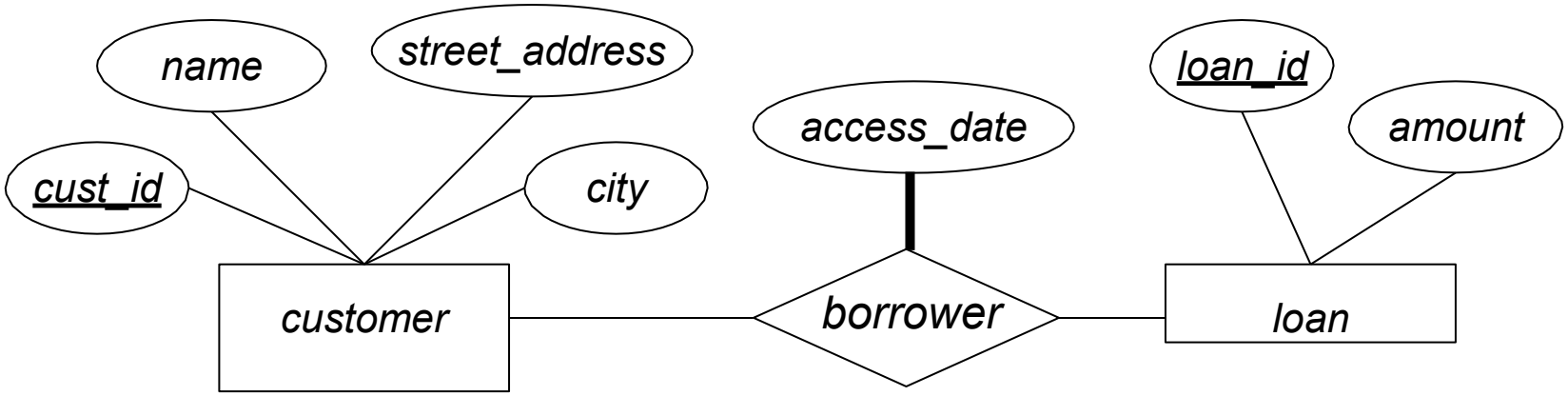
# Relationship-Set Primary Keys (2)

- For many-to-one or one-to-many mappings:
  - e.g. between strong entity-sets *A* and B
  - Primary key of entity-set on "many" side is primary key of relationship
- Example: relationship *R* between *A* and *B*
  - One-to-many mapping, with *B* on "many" side
  - Schema contains *primary_key*(*A*)U*primary_key*(*B*), plus any descriptive attributes on *R*
  - *primary_key*(*B*) is primary key of *R*

# Relationship-Set Foreign Keys

- Relationship-sets associate entities in entity-sets
  - Need foreign key constraints on relation schema for *R*
- For each entity-set $E_i$ participating in *R*:
  - Relation schema for *R* has a foreign-key constraint on $E_i$ relation, for *primary_key*($E_i$) attributes
- Relation schema notation doesn't provide a mechanism for indicating foreign key constraints
  - Don't forget about foreign keys and candidate keys!
  - Can specify both foreign key constraints, and candidate keys, in SQL DDL

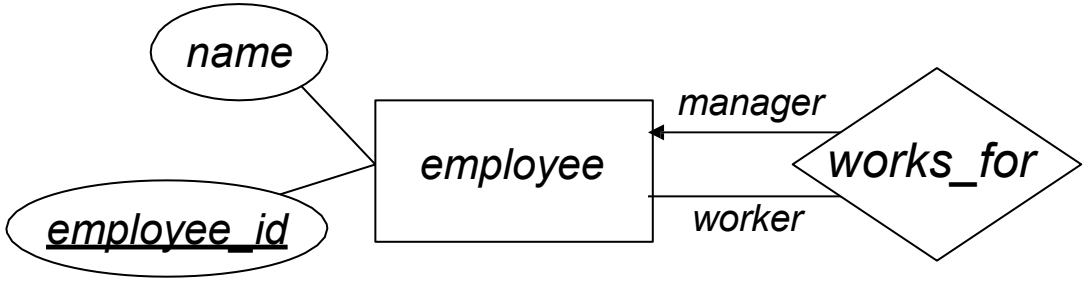
# Relationship-Set Example




- **Relation schema for *borrower*:**
  - **Primary key of *customer* is *cust_id***
  - **Primary key of *loan* is *loan_id***
  - **Descriptive attribute *access_date***
  - ***borrower* mapping cardinality is many-to-many**

    ***borrower*(*cust_id*, *loan_id*, *access_date*)**

# Relationship-Set Example (2)




- Relation schema for *employee* entity-set:

  *employee*(*employee_id*, *name*)

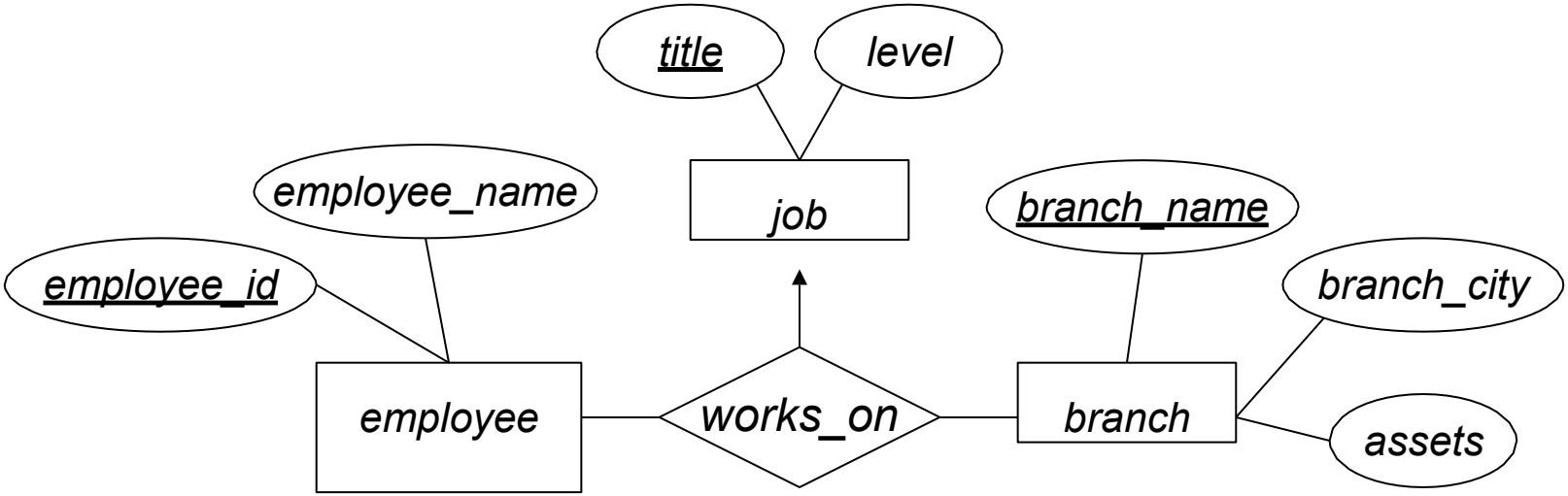
- Relation schema for *works_for*:
  - One-to-many mapping from *manager* to *worker*
  - "Many" side is used for primary key

    *works_for*(*employee_id*, *manager_id*)

# N-ary Relationship Primary Keys

- **For degree > 2 relationship-sets:**
  - If no arrows ("many-to-many" mapping), relationship-set primary key is union of all participating entity-sets' primary keys
  - If one arrow ("one-to-many" mapping), relationship-set primary key is union of primary keys of entity-sets without an arrow
  - Don't allow more than one arrow for relationship-sets with degree > 2

# N-ary Relationship-Set Example




- **Entity-set schemas:**

  *job*(*title*, *level*)

  *employee*(*employee_id*, *employee_name*)

  *branch*(*branch_name*, *branch_city*, *assets*)

- **Relationship-set schema:**
  - **Primary key includes entity-sets on non-arrow links**

    *works_on*(*employee_id*, *branch_name*, *title*)

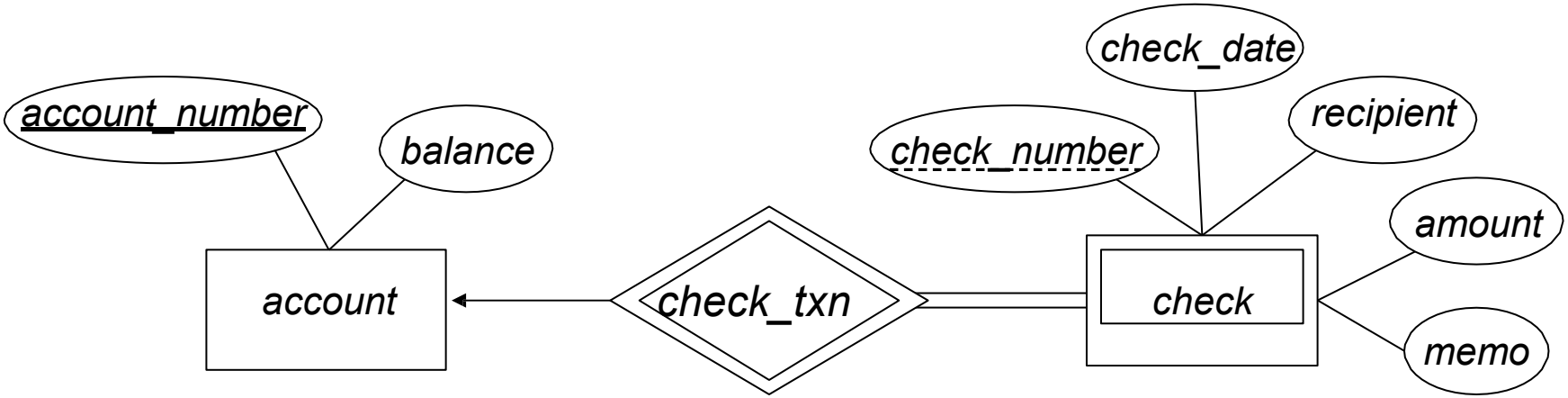
# Weak Entity-Sets

- Weak entity-sets depend on at least one strong entity-set
  - Identifying entity-set, or owner entity-set
  - Relationship between the two called the identifying relationship
- Weak entity-set *A* owned by strong entity-set *B*
  - Attributes of *A* are $\{a_1, a_2, \ldots, a_m\}$
  - *primary_key*(*B*) = $\{b_1, b_2, \ldots, b_n\}$
  - Relational schema for *A*: $\{a_1, a_2, \ldots, a_m\} \cup \{b_1, b_2, \ldots, b_n\}$
  - Primary key of *A* is *discriminator*(*A*)U*primary_key*(*B*)
  - *A* has foreign key constraint on *primary_key*(*B*), to *B*

# Identifying Relationship?

- Identifying relationship is many-to-one, with no descriptive attributes
- Relational schema for weak entity-set includes primary key for strong entity-set
  - Foreign key constraint imposed, too
- No need to create relational schema for identifying relationship
  - Would be redundant to weak entity-set's relational schema!

# Weak Entity-Set Example



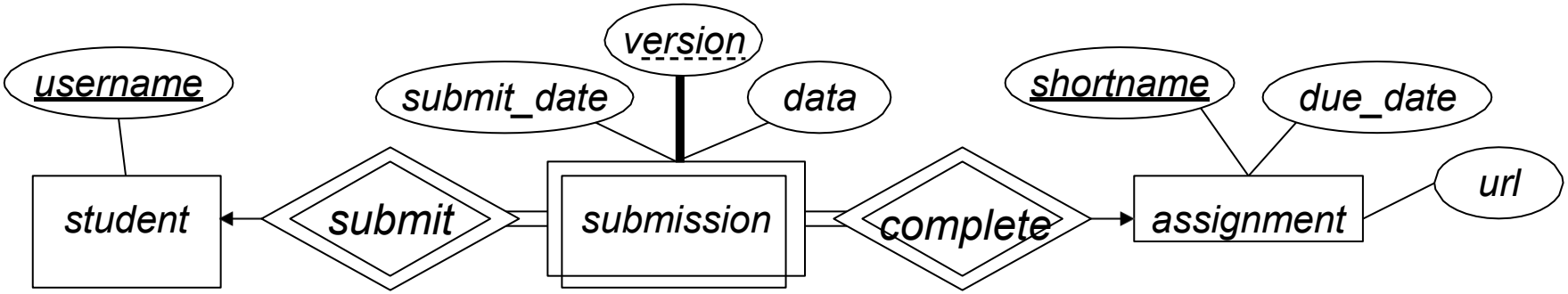

- ***account* schema:**

  ***account*(*account_number*, *balance*)**

- ***check* schema:**
  - **Discriminator is *check_number***
  - **Primary key for *check* is:**
    **(*account_number*, *check_number*)**

  ***check*(*account_number*, *check_number*,**
  ***check_date*, *recipient*, *amount*, *memo*)**

# Weak Entity-Set Example (2)




- Schemas for strong entity-sets: *student*(*username*) *assignment* (*shortname*, *due_date*, *url*)
- Schema for *submission* weak entity-set:
  - Discriminator is *version*
  - Both *student* and *assignment* are owners!

  *submission*(*username*, *shortname*, *version*, *submit_date*, *data*)

# Schema Combination

- Relationship between weak entity-set and strong entity-set doesn't need represented separately
  - Many-to-one relationship
  - Weak entity-set has total participation
  - Weak entity-set's schema includes representation of identifying relationship
- Can apply technique to other relationship-sets with many-to-one mapping
  - Entity-sets *A* and *B*, with relationship-set *AB*
  - Many-to-one mapping
  - *A*'s participation in *AB* is total

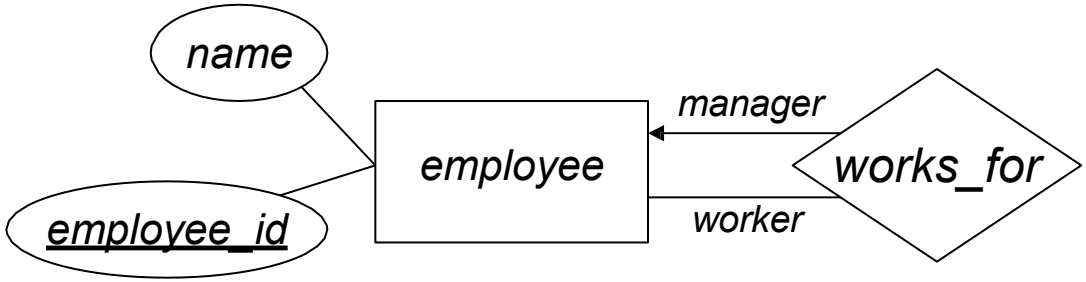
# Schema Combination (2)

- Entity-sets *A* and *B*, relationship-set *AB*
  - Many-to-one mapping
  - *A*'s participation in *AB* is total
- Generates relation schemas *A*, *B*, *AB*
  - Primary key of *AB* is *primary_key*(*A*)
    - (*A* is on "many" side of mapping)
  - *AB* has foreign key constraints on both *A* and *B*
- Combine *A* and *AB* relation schemas
  - Primary key of combined schema still *primary_key*(*A*)
  - Only need one foreign-key constraint, to *B*

# Schema Combination (3)

- If *A*'s participation in *AB* is partial, can still combine schemas
  - Need to store *null* values for *primary_key*(*B*) attributes when an entity in *A* maps to no entity in *B*
- If *AB* is one-to-one mapping:
  - Can also combine schemas in this case
  - Could incorporate *AB* into schema for *A*, or schema for *B*
  - When relationship-set is combined into an entity-set, the entity-set's primary key *doesn't change!*

# Schema-Combination Example




- Manager to worker mapping is one-to-many
- Relation schemas were:

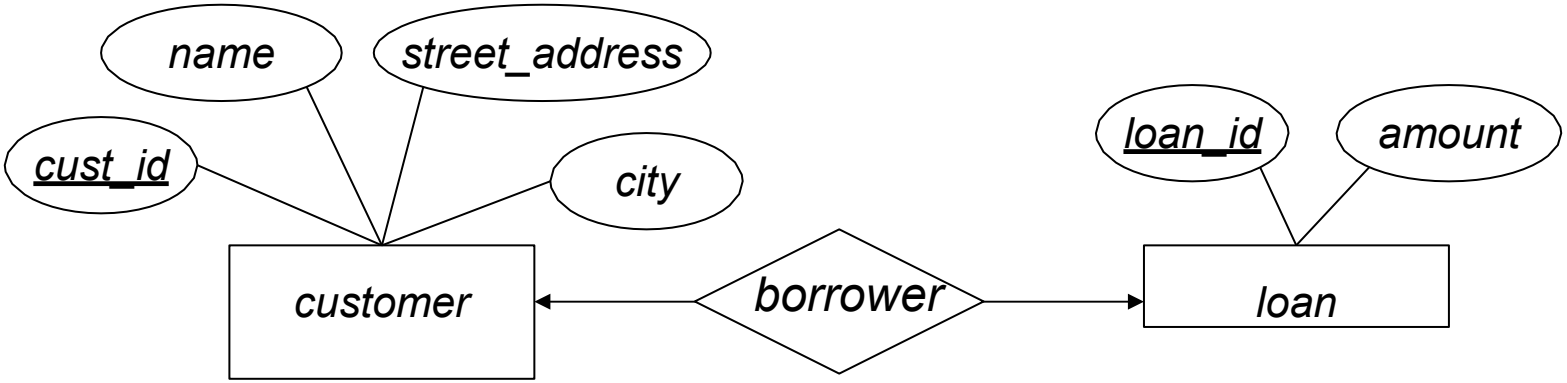
  *employee*(*employee_id*,*name*)

  *works_for*(*employee_id*, *manager_id*)
- Could combine into:

  *employee*(*employee_id*, *name*, *manager_id*)

  – Need to store *null* for employees with no manager

# Schema Combination Example (2)




- One-to-one mapping between customers, loans

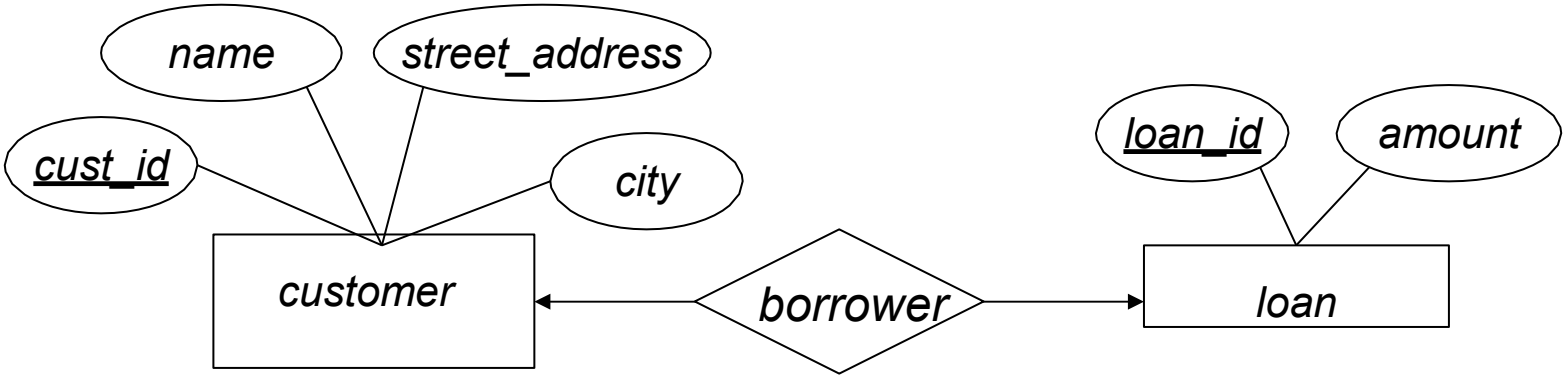
  *customer*(*cust_id*, *name*, *street_address*, *city*)

  *loan*(*loan_id*, *amount*)

  *borrower*(*cust_id*, *loan_id*)

  - *borrower* could also use *loan_id* for primary key
- Could combine *borrower* schema into *customer* or *loan* schema
  - Does it matter which one youchoose?

# Schema Combination Example (3)

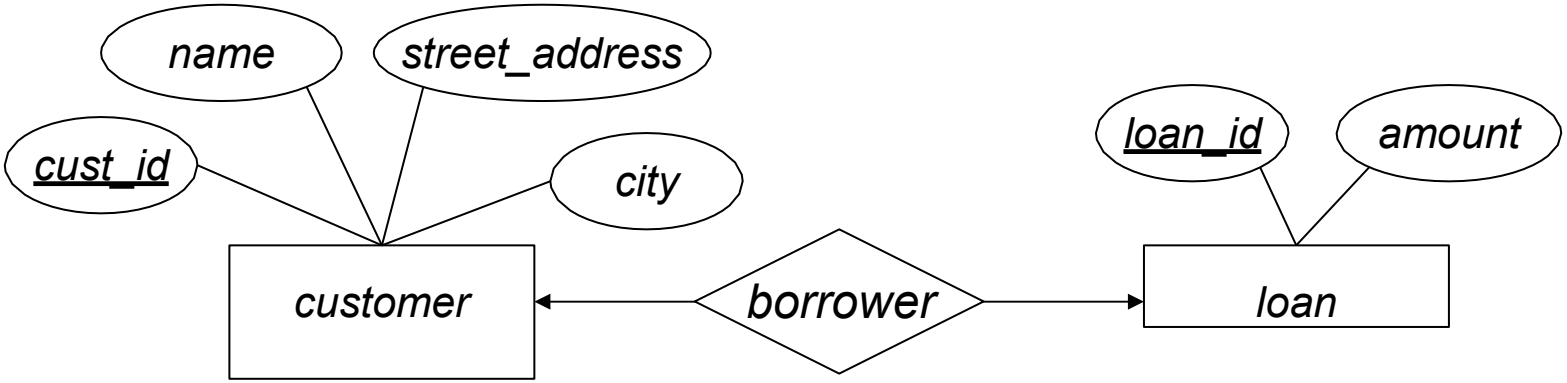



- Participation of *loan* in *borrower* will be total
  - Combining *borrower* into *customer* would require *null* values for customers without loans
- Better to combine *borrower* into *loan* schema

  *customer*(*cust_id*, *name*, *street_address*, *city*)

  *loan*(*loan_id*, *cust_id*, *amount*)
  - No *null* values!

# Schema Combination Example (4)




- **Schema:**
  - *customer(cust_id, name, street_address, city)*

    *loan(loan_id, cust_id, amount)*
- **What if, after a while, we wanted to change the mapping cardinality?**
  - **Change to schema would be significant**
  - **Would need to migrate existing data to new schema**

# Schema Combination Notes

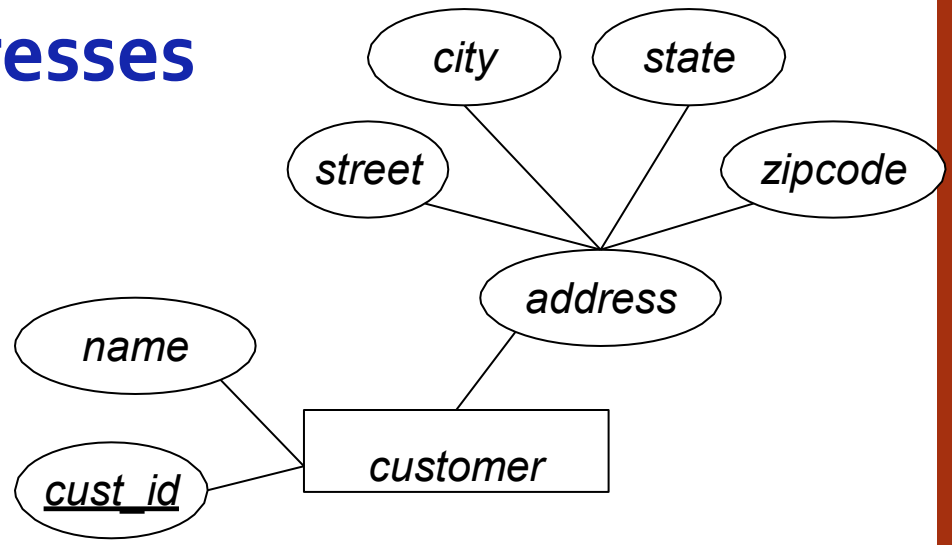
- Benefits of schema combination:
  - Eliminate a foreign-key constraint, and associated performance impact
    - Constraint enforcement
    - Extra join operations in queries
  - Reduce storage requirements
- Drawbacks of schema combination:
  - May necessitate use of *null* values
  - Makes it harder to change mapping cardinality constraints in the future

# Composite Attributes

- Relational model doesn't handle composite attributes
- When mapping E-R composite attributes to relation schema:
  - Each component attribute maps to a separate attribute in relation schema
  - In relation schema, simply can't refer to composite as a whole
  - (Can adjust this mapping for databases that support composite types)

# Composite Attribute Example

- Customers with addresses




- Each component of *address* becomes a separate attribute

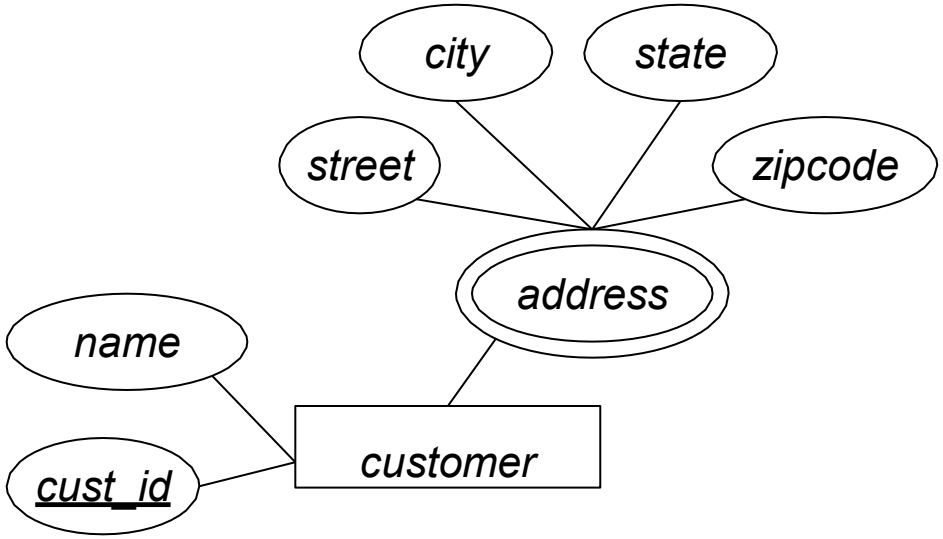
*customer(cust_id, name, street, city, state, zipcode)*

# Multivalued Attributes

- Multivalued attributes require a separate relation schema
  - No such thing as a multivalued attribute in relational model
- For multivalued attribute *M* in entity-set *E*
  - Create a relation schema *R* to store *M*, with attribute *A* corresponding to *M*
    - *A* is single-valued version of *M*
  - Attributes of *R* are: *A U primary_key*(*E*)
  - Primary key of *R* includes <u>all</u> attributes of *R*
    - Each value in *M* for entity *e* must be unique
  - Foreign key constraint from *R* to *E*, on *primary_key*(*E*) attributes

# Multivalued Attribute Example

- Customers with multiple addresses




- Create separate relation to store each address

  *customer*(*cust_id*, *name*)

  *cust_addrs*(*cust_id*, *street*, *city*, *state*, *zipcode*)

  – Large primary keys aren't ideal – tend to be costly

# Review

- Can map E-R model schemas to relational model schemas very easily
  - Mapping process is straightforward and unambiguous
- Some flexibility in optimizing relation schemas
  - Mapping cardinalities, etc.
- Some E-R concepts are more expensive
  - Multivalued attributes (especially composite ones)

# Thank You!!!