ENTITY-RELATIONSHIP MODEL II

CS121: Introduction to Relational Database Systems
Fall 2014 – Lecture 15

Last Lecture

- Began to explore the Entity-Relationship Model
  - A visual representation of database schemas
  - Can represent entities and relationships
  - Can represent constraints in the schema
- Last time, left off with mapping cardinalities

Entity-Set Keys

- Entities in an entity-set must be uniquely distinguishable using their values
  - Entity-set: each entity is unique
- E-R model also includes the notion of keys:
  - Superkey: a set of one or more attributes that can uniquely identify an entity
  - Candidate key: a *minimal* superkey
  - Primary key: a candidate key chosen by DB designer as the primary means of accessing entities
- Keys are a property of the entity-set
  - They apply to *all* entities in the entity-set

Choosing Candidate Keys

- Candidate keys constrain the values of the key attributes
  - No two entities can have the same values for those attributes
  - Need to ensure that database can actually represent all expected circumstances
- Simple example: *customer* entity-set
  - Using customer name as a candidate key is bad design: different customers can have the same name

Choosing Primary Keys

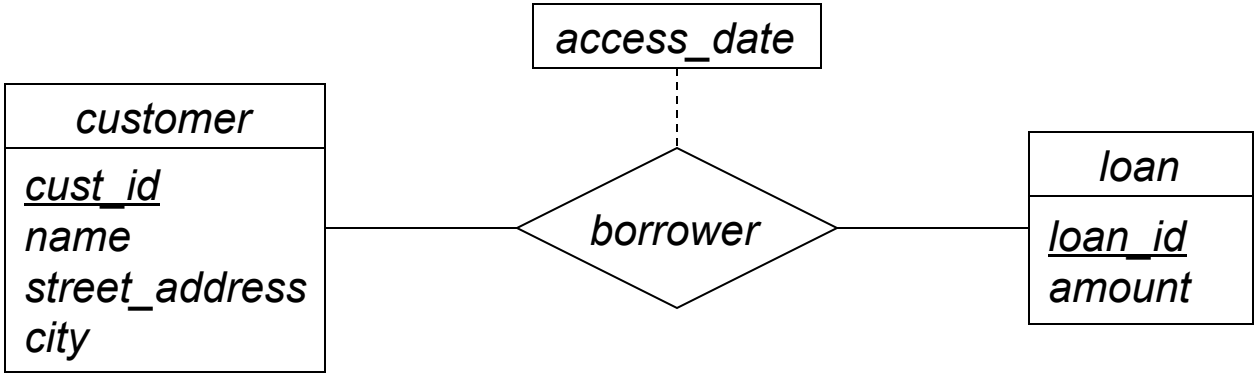
- An entity-set may have multiple candidate keys
- The primary key is the candidate key most often used to reference entities in the set
  - In logical/physical design, primary key values will be used to represent relationships
  - External systems may also use primary key values to reference entities in the database
- The primary key attributes should *never* change!
  - If ever, it should be *extremely* rare.

Choosing Keys: Performance

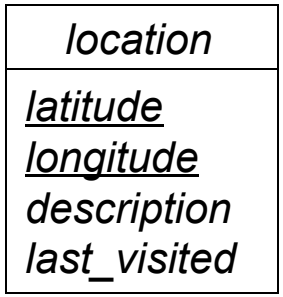
- Large, complicated, or multiple-attribute keys are generally slower
  - Use smaller, single-attribute keys
    - (You can always generate them…)
  - Use faster, fixed-size types
    - e.g. **`INT`** or **`BIGINT`**
- Especially true for primary keys!
  - Values used in both database and in access code
  - Use something small and simple, if possible

Diagramming Primary Keys

- In an entity-set diagram, all attributes in the primary key have an underlined name



- Another example: a geocache *location* entity-set



Keys and Relationship-Sets

- Need to be able to distinguish between individual relationships in a relationship-set as well
  - Relationships aren't distinguished by their descriptive attributes
  - (They might not even have descriptive attributes)
- Relationships are identified by the *entities* participating in the relationship
  - Specific relationship instances are uniquely identified by the primary keys of the participating entities

Keys and Relationship-Sets (2)

- Given:
  - R is a relationship-set with no descriptive attributes
  - Entity-sets $E_1, E_2, \ldots, E_n$ participate in R
  - $primary\_key(E_i)$ denotes set of attributes in E_i that represent the primary key of E_i
- A relationship instance in R is identified by

  $primary\_key(E_1) \cup primary\_key(E_2) \cup \ldots \cup primary\_key(E_n)$
  - This is a superkey
  - Is it a candidate key?
    - Depends on the *mapping cardinality* of the relationship set!

Keys and Relationship-Sets (3)

- If R also has descriptive attributes $\{a_1, a_2, \ldots\}$, a relationship instance is described by:

  $primary\_key(E_1) \cup primary\_key(E_2) \cup \ldots \cup primary\_key(E_n) \cup \{a_1, a_2, \ldots\}$

  - Not a minimal superkey!
  - By definition, there can only be one relationship between $\{E_1, E_2, \ldots, E_n\}$ in the relationship-set
    - i.e. the descriptive attributes do not identify specific relationships!
- Thus, just as before, this is also a superkey:

  $primary\_key(E_1) \cup primary\_key(E_2) \cup \ldots \cup primary\_key(E_n)$

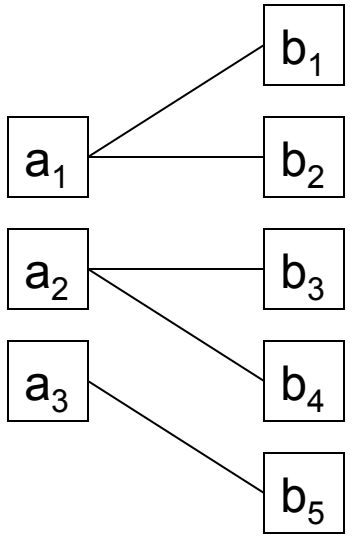
Relationship-Set Primary Keys

- What is the primary key for a binary relationship-set?
  - Must also be a candidate key
  - Depends on the mapping cardinalities
- Relationship-set *R*, involving entity-sets *A* and *B*
  - If mapping is many-to-many, primary key is:

    $primary\_key(A) \cup primary\_key(B)$
  - Any given entity's primary-key values can appear multiple times in *R*
  - We need both entity-sets' primary key attributes to uniquely identify relationship instances

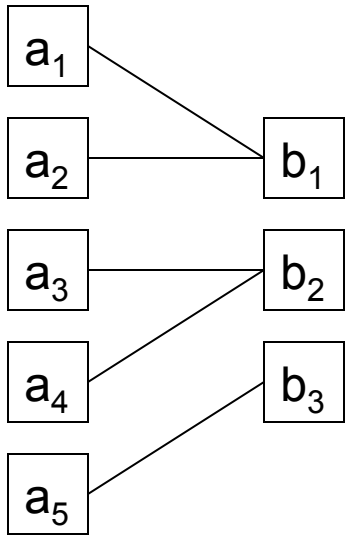
Relationship-Set Primary Keys (2)

- Relationship-set *R*, involving entity-sets *A* and *B*
  - Individual relationships are described by *primary_key*(*A*) ∪ *primary_key*(*B*)
- If mapping is one-to-many:
  - Entities in *B* associated with *at most* one entity in *A*
  - A given *primary_key*(*A*) value can appear in multiple relationships
  - Each value of *primary_key*(*B*) can appear only once
  - Relationships in *R* are uniquely identified by *primary_key*(*B*)
  - *primary_key*(*B*) is primary key of relationship-set



Relationship-Set Primary Keys (3)

- Relationship-set *R*, involving entity-sets *A* and *B*
- Many-to-one is exactly the opposite of one-to-many
  - *primary_key*(*A*) uniquely identifies relationships in *R*

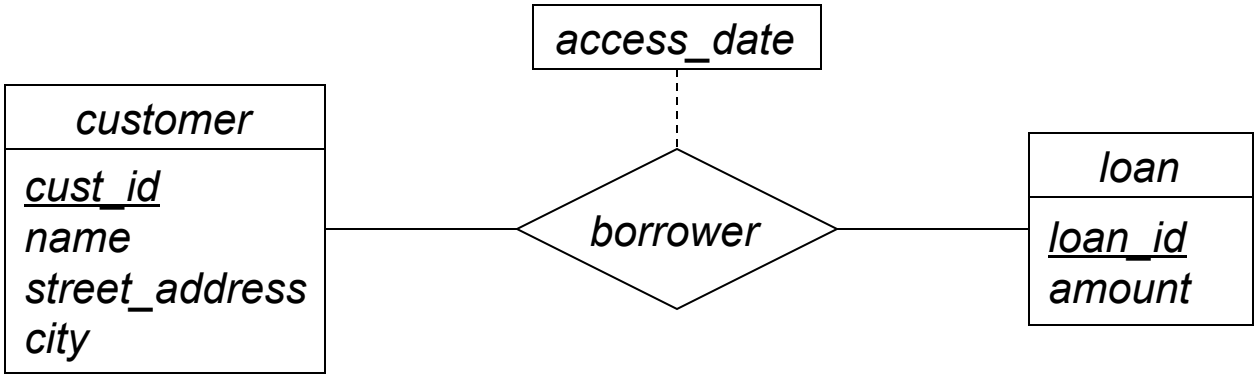


Relationship-Set Primary Keys (4)

- Relationship-set *R*, involving entity-sets *A* and *B*
- If mapping is one-to-one:
  - Entities in *A* associated with *at most* one entity in *B*
  - Entities in *B* associated with *at most* one entity in *A*
  - Each entity's key-value can appear only once in *R*
  - Either entity-set's primary key can be primary key of *R*
- For one-to-one mapping, *primary_key*(*A*) and *primary_key*(*B*) are both candidate keys
  - Make sure to enforce both candidate keys in the implementation schema!

Example

- What is the primary key for *borrower* ?



- *borrower* is a many-to-many mapping
  - Relationship instances are described by (*cust_id*, *loan_id*, *access_date*)
  - Primary key for relationship-set is (*cust_id*, *loan_id*)

Participation Constraints

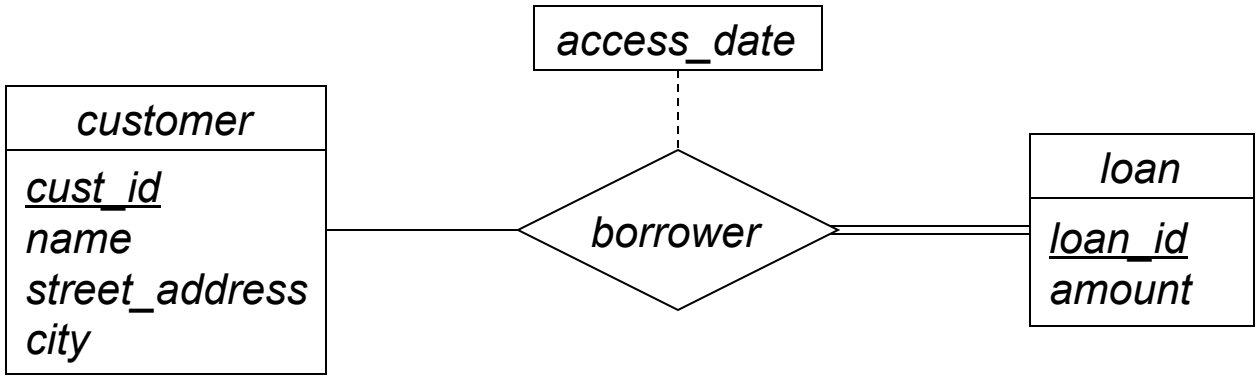
- Given entity-set *E*, relationship-set *R*
  - How many entities in *E* participate in *R* ?
  - In other words, what is minimum number of relationships that each entity in *E must* participate in?
- If every entity in *E* participates in at least one relationship in *R*, then:
  - *E*'s participation in *R* is total
- If only some entities in *E* participate in relationships in *R*, then:
  - *E*'s participation in *R* is partial

Participation Constraints (2)

- Example: *borrower* relationship between *customer* and *loan*
- A customer might not have a bank loan
  - Could have a bank account instead
  - Could be a new customer
  - Participation of *customer* in *borrower* is partial
- Every loan definitely has at least one customer
  - Doesn't make any sense not to!
  - Participation of *loan* in *borrower* is total

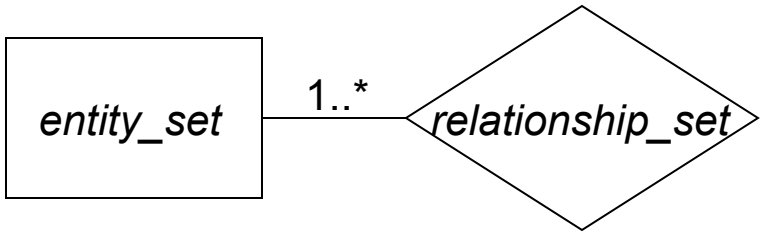
Diagramming Participation

- Can indicate participation constraints in entity-relationship diagrams
  - Partial participation shown with a single line
  - Total participation shown with a double line



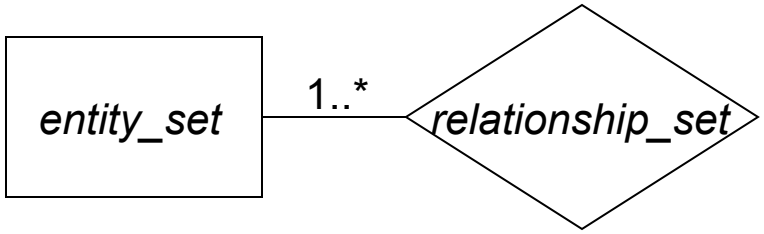
Numerical Constraints

- Can also state numerical participation constraints
  - Specifies how many different relationship instances each entity in the entity-set can participate in
  - Indicated on link between entity and relationship
- Form: lower..upper
  - * means "unlimited"
  - 1..* = one or more
  - 0..3 = between zero and three, inclusive
  - etc.



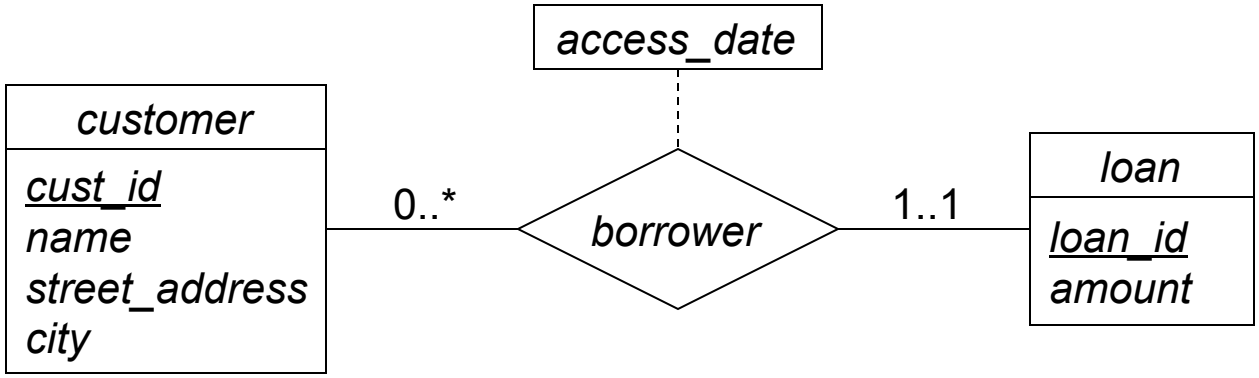
Numerical Constraints (2)

- Can also state mapping constraints with numerical participation constraints
- Total participation:
  - Lower bound at least 1
- Partial participation:
  - Lower bound is 0



Numerical Constraint Example

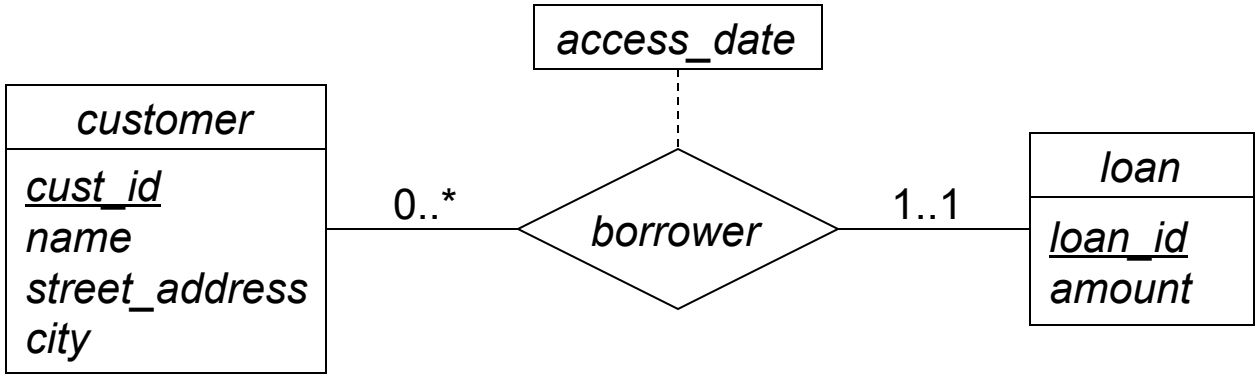
- What does this mean?



  - Each *customer* entity may participate in zero or more relationships in this relationship-set
    - *A customer can have zero or more loans.*
  - Each *loan* entity must participate in *exactly one* relationship (no more, no less) in this relationship-set
    - *Each loan must be owned by exactly one customer.*

Numerical Constraint Example (2)

- What is the mapping cardinality of *borrower* ?



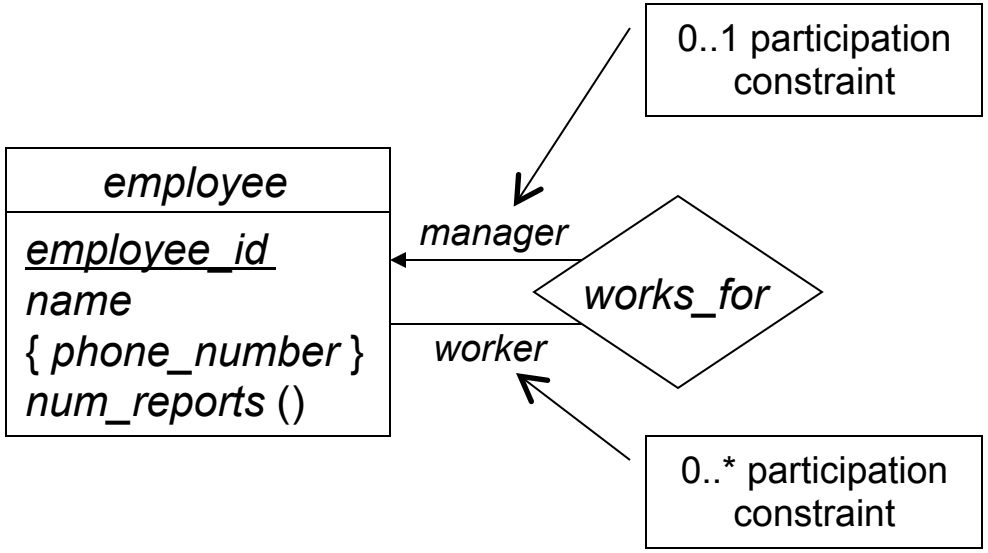
- From last slide:
  - *A customer can have zero or more loans*
  - *Each loan must be owned by exactly one customer.*
- This is a one-to-many mapping from *customer* to *loan*

Diagramming Roles

- Entities have roles in relationships
  - An entity's role indicates the entity's function in the relationship
  - e.g. role of customer in *borrower* relationship-set is that they own the loan
- Sometimes roles are ambiguous
  - e.g. when the same kind of entity is involved in a relationship multiple times
- Example: *works_for* relationship
  - Relationship is between two *employee* entities
  - One is the *manager*; the other is the *worker*

Diagramming Roles (2)

- If roles need to be indicated, put labels on the lines connecting entity to relationship



- *works_for* relationship-set is one-to-many from managers to workers

Weak Entity-Sets

- Sometimes an entity-set doesn't have distinguishing attributes
  - Can't define a primary key for the entity-set!
  - Called a weak entity-set
- Example:
  - Checking accounts have a unique account number
  - Checks have a check number
    - Unique for a given account, but not across all accounts!
    - Number only makes sense in context of a particular account
  - Want to store check transactions in the database

Weak Entity-Sets (2)

- Weak entity-sets *must* be associated with another (strong) entity-set
  - Called the identifying entity-set, or owner entity-set
  - The identifying entity-set owns the weak entity-set
  - Association called the identifying relationship
- Every weak entity *must* be associated with an identifying entity
  - Weak entity's participation in relationship-set is total
  - The weak entity-set is existence dependent on the identifying entity-set
  - If the identifying entity is removed, its weak entities should also cease to exist
  - *(this is where cascade-deletes may be appropriate…)*

Weak Entity-Set Keys

- Weak entity-sets don't have a primary key
  - Still need to distinguish between weak entities associated with a particular strong entity
- Weak entities have a discriminator
  - A set of attributes that distinguishes between weak entities associated with a strong entity
  - Also known as a partial key
- Checking account example:
  - The check number is the discriminator for check transactions

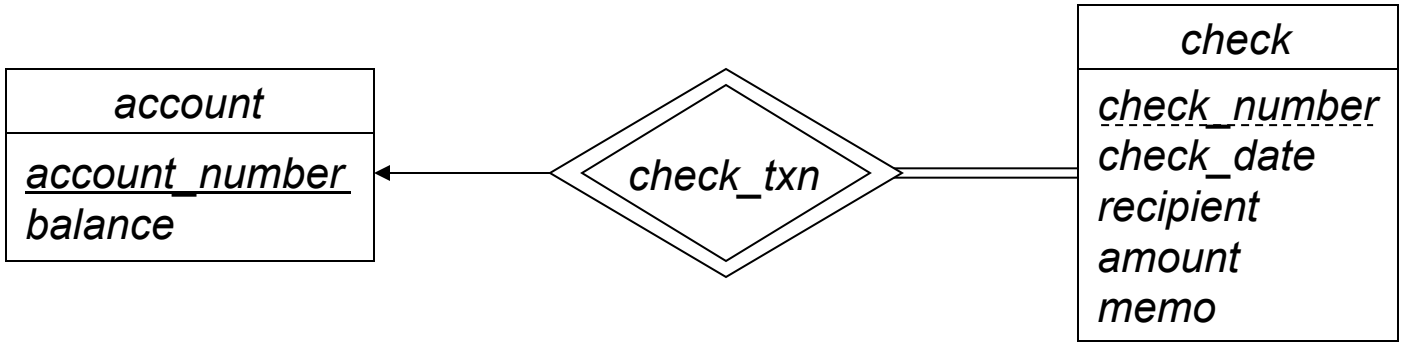
Weak Entity-Set Keys (2)

- Using discriminator, can define a primary key for weak entity-sets
- For a weak entity-set W, and an identifying entity-set S, primary key of W is:

  $primary\_key(S) \cup discriminator(W)$
- Checking account example:
  - *account_number* is primary key for checking accounts
  - *check_number* is discriminator (partial key) for checks
  - Primary key for check transactions would be (*account_number*, *check_number*)

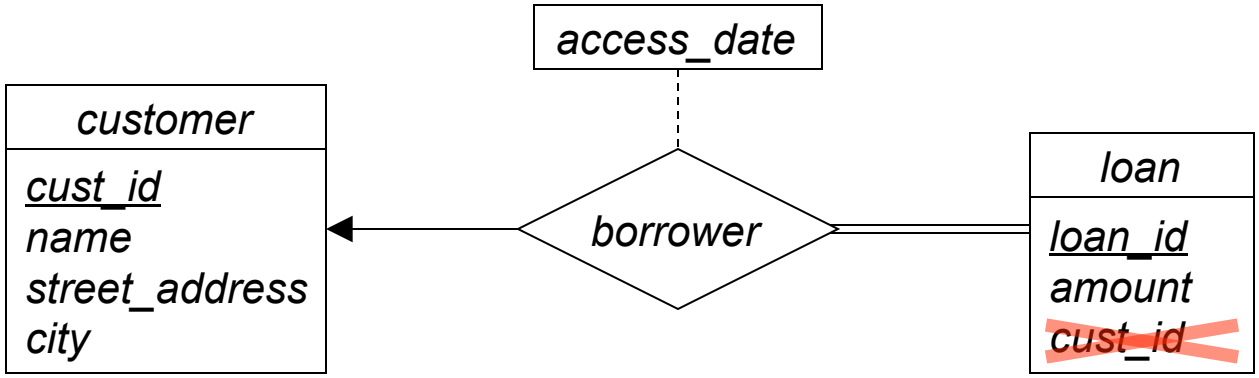
Diagramming Weak Entity-Sets

- Weak entity-sets drawn similarly to strong entity-sets
  - Difference: discriminator attributes are underlined with a dashed underline
- Identifying relationship to the owning entity-set is indicated with a double diamond
  - One-to-many mapping
  - Total participation on weak entity side



Common Attribute Mistakes

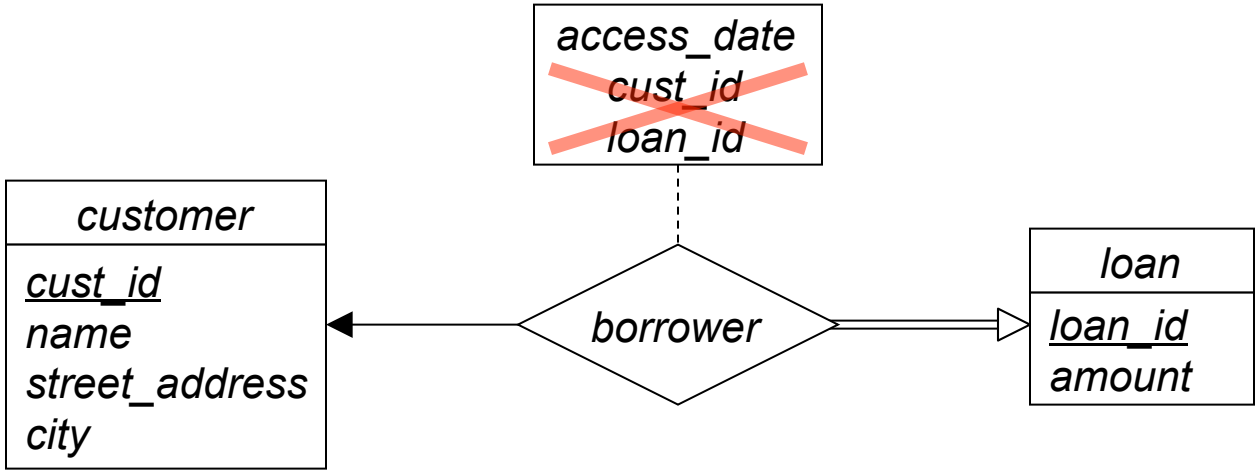
- Don't include entity-set primary key attributes on other entity-sets!
  - e.g. customers and loans, in a one-to-many mapping



- Even if every loan is owned by only one customer, this is still wrong
  - The association is recorded by the *relationship*, so specifying foreign key attributes on the entity-set is redundant

Common Attribute Mistakes (2)

- Don't include primary key attributes as descriptive attributes on relationship-set, either!
- This time, assume *borrower* is a 1:1 mapping
  - IDs used as descriptive attributes on *borrower*



- Again, this is implicit in the relationship