# FINAL EXAM REVIEW

CS121: Introduction to Relational Database Systems

Fall 2018 – Lecture 27

# Final Exam Overview



- Unlimited time, multiple sittings
  - Open book, notes, MySQL database, etc. (the usual)
- Primary topics: everything in the last half of the term
  - DB schema design and Entity-Relationship Model
  - Functional/multivalued dependencies, normal forms
  - Also SQL DDL, DML, stored routines, hierarchies, etc.
- Questions will generally take this form:
  - "Design a database to model such-and-such a system."
    - Create an E-R diagram for the database
    - Translate to relational model and DDL
    - Write some queries and/or stored routines against your schema
  - Functional/multivalued dependency problems as well

# Final Exam Admin Notes



- Final exam will be available this afternoon

- **Due next Friday, December 14 at 5:00 pm**

- Solution sets for all assignments except HW7 are available
  - HW7 solutions will be available over the weekend

# Entity-Relationship Model



- Diagramming system for specifying DB schemas
  - Can map an E-R diagram to the relational model
- Entity-sets (a.k.a. strong entity-sets)
  - "Things" that can be uniquely represented
  - Can have a set of attributes; <u>must</u> have a primary key
- Relationship-sets
  - Associations between two or more entity-sets
  - Can have descriptive attributes
  - Relationships in a relationship-set are uniquely identified by the participating entities, *not* the descriptive attributes
  - Primary key of relationship depends on mapping cardinality of the relationship-set

# Entity-Relationship Model (2)



- Weak entity-sets
  - Don't have a primary key; have a discriminator instead
  - Must be associated with a strong entity-set via an identifying relationship
  - Diagrams must indicate both weak entity-set and the identifying relationship(s)
- Generalization/specialization of entity-sets
  - Subclass entity-sets inherit attributes and relationships of superclass entity-sets
- Schema design problems will likely involve most or all of these things in one way or another

# E-R Model Guidelines



- You should know:
  - How to properly diagram each of these things
  - Various constraints that can be applied, what they mean, and how to diagram them
  - How to map each E-R concept to the relational model
    - Including rules for primary keys, candidate keys, etc.
- Final exam problem will require familiarity with all of these points
- Make sure you are familiar with the various E-R design issues, so you don't make those mistakes!

# E-R Model Attributes



- Attributes can be:
  - Simple or composite
  - Single-valued or multivalued
  - Base or derived
- Attributes are listed in the entity-set's rectangle
  - Components of composite attributes are indented
  - Multivalued attributes are enclosed with { }
  - Derived attributes have a trailing ()
- Entity-set primary key attributes are underlined
- Weak entity-set partial key has dashed underline
- Relationship-set descriptive attributes aren't a key!

# Example Entity-Set



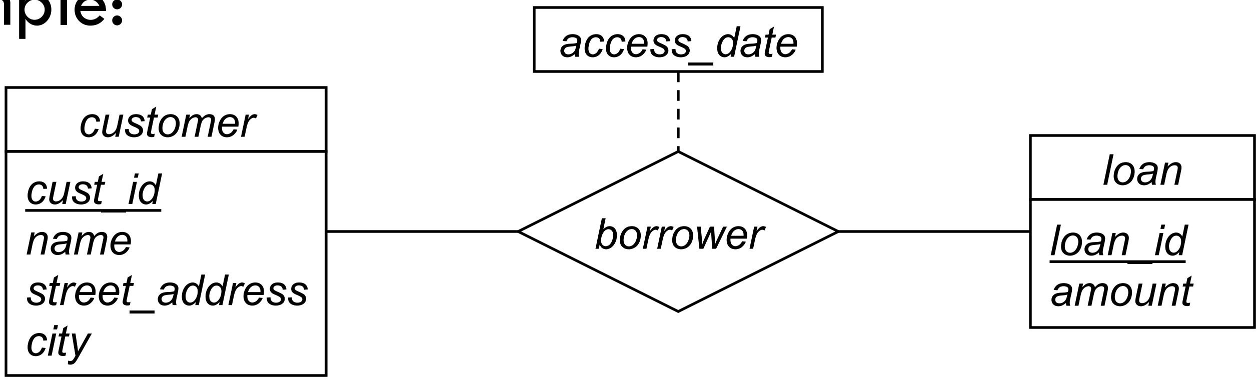
- *customer* entity-set
- Primary key:
  - *cust_id*
- Composite attributes:
  - *name, address*
- Multivalued attribute:
  - *phone_number*
- Derived attribute:
  - *age*

| *customer* |
|---|
| <u>*cust_id*</u><br>*name*<br>    *first_name*<br>    *middle_initial*<br>    *last_name*<br>*address*<br>    *street*<br>    *city*<br>    *state*<br>    *zip_code*<br>{ *phone_number* }<br>*birth_date*<br>*age* () |

# Example Relationship-Set



- Relationships are identified *only* by participating entities
  - Different relationships can have same value for a descriptive attribute
- Example:




  - A given pair of *customer* and *loan* entities can only have one relationship between them via the *borrower* relationship-set

# E-R Model Constraints

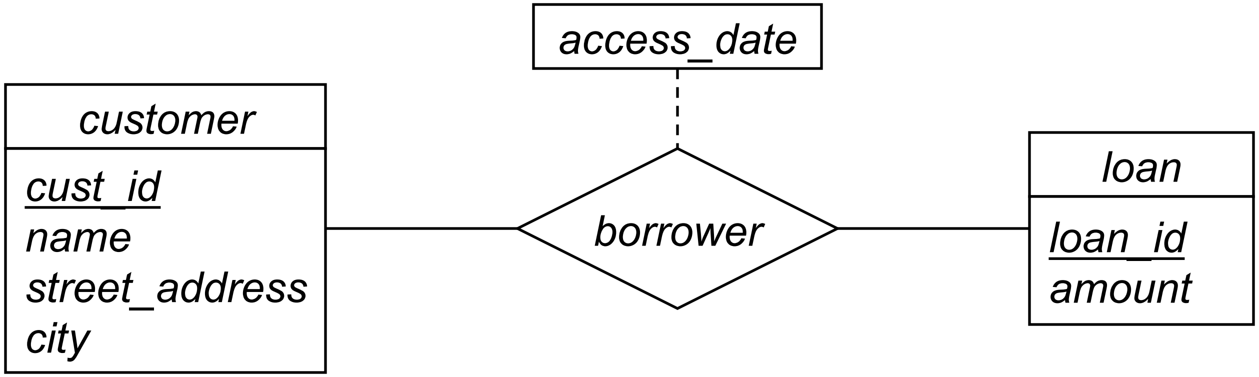


- E-R model can represent several constraints:
  - Mapping cardinalities
  - Key constraints in entity-sets
  - Participation constraints
- Make sure you know when and how to apply these constraints
- Mapping cardinalities:
  - "How many other entities can be associated with an entity, via a particular relationship set?"
  - Choose mapping cardinality based on the rules of the enterprise being modeled

# Mapping Cardinalities



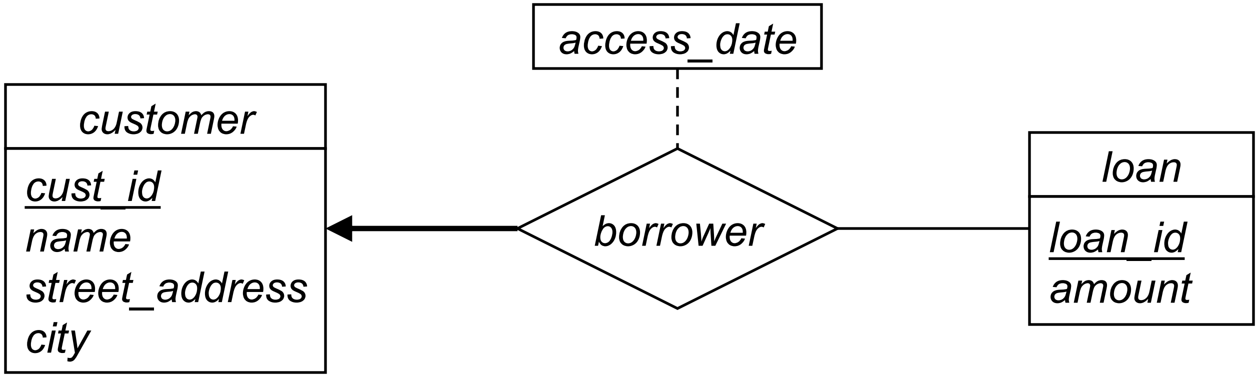
- In relationship-set diagrams:
  - arrow towards entity-set represents "one"
  - line with no arrow represents "many"
  - arrow is *always* towards the entity-set
- Example: many-to-many mapping
  - The way that most banks work…



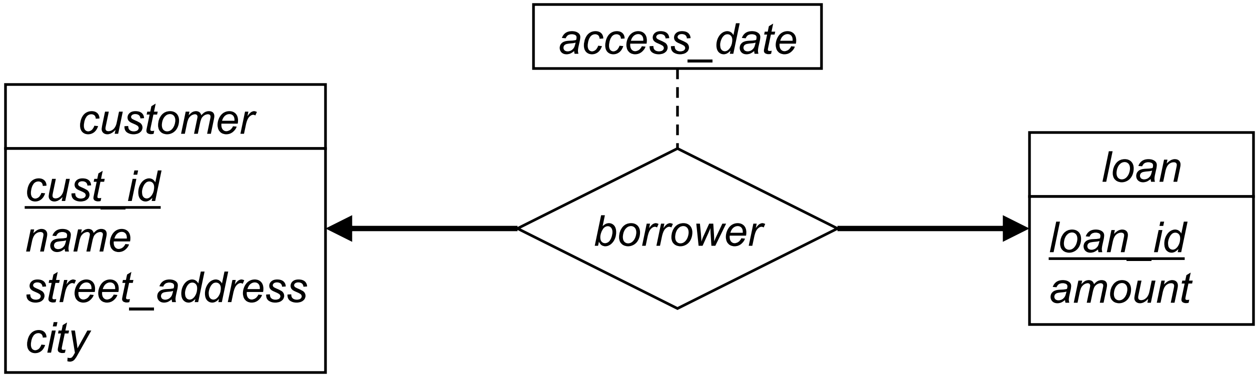
# Mapping Cardinalities (2)



- One-to-many mapping:




- One-to-one mapping:



# Relationship-Set Primary Keys



- Relationship-set *R*, involving entity-sets *A* and *B*
- If mapping is many-to-many, primary key is:
  *primary_key*(*A*) ∪ *primary_key*(*B*)
- If mapping is one-to-many, *primary_key*(*B*) is primary key of relationship-set
- If mapping is many-to-one, *primary_key*(*A*) is primary key of relationship-set
- If mapping is one-to-one, use *primary_key*(*A*) or *primary_key*(*B*) for primary key
  - Enforce <u>both</u> as candidate keys in the implementation schema!

# Participation Constraints

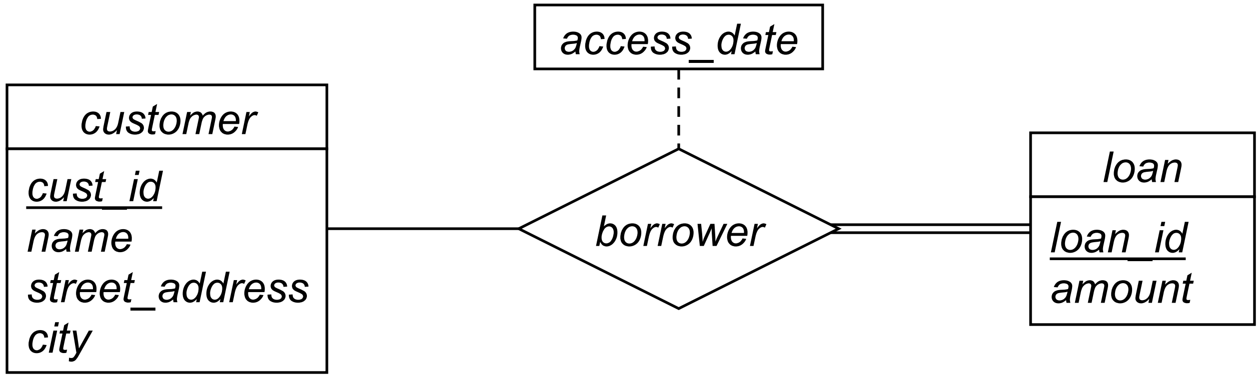


- Given entity-set *E*, relationship-set *R*
- If every entity in *E* participates in at least one relationship in *R*, then:
  - *E*'s participation in *R* is total
- If only some entities in *E* participate in relationships in *R*, then:
  - *E*'s participation in *R* is partial
- Use total participation when enterprise requires all entities to participate in at least one relationship

# Diagramming Participation



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



# Weak Entity-Sets

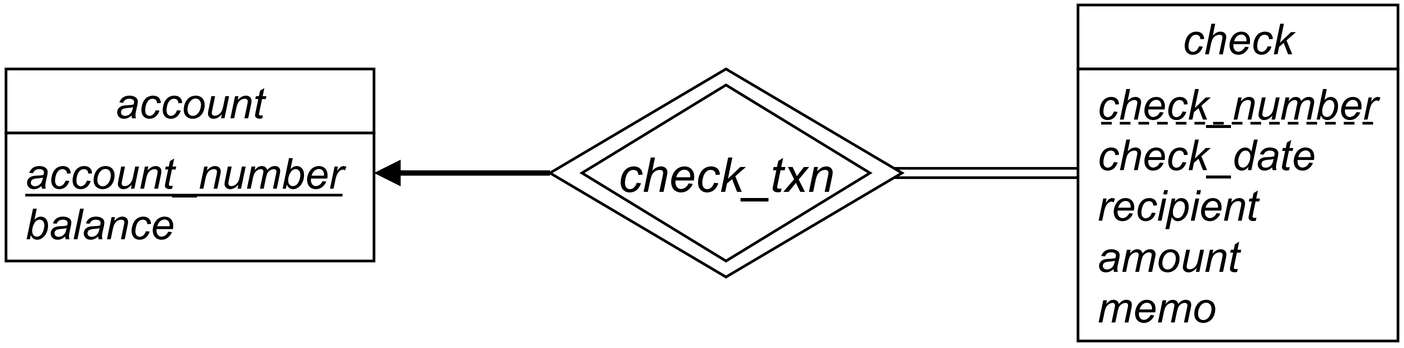


- Weak entity-sets don't have a primary key
  - *Must* be associated with an identifying entity-set
  - Association called the identifying relationship
  - If you use weak entity-sets, make sure you also include both of these things!
- <u>Every</u> weak entity is associated with an identifying entity
  - Weak entity's participation in relationship-set is <u>total</u>
- Weak entities have a discriminator (partial key)
  - Need to distinguish between the weak entities
  - Weak entity-set's primary key is partial key combined with identifying entity-set's primary key

# Diagramming Weak Entity-Sets



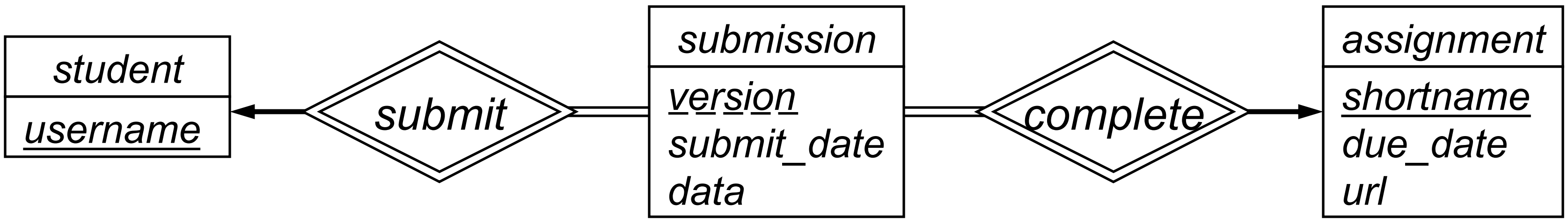
- In E-R model, can only tell that an entity-set is weak if it has a discriminator instead of a primary key
  - Discriminator attributes have a dashed underline
- Identifying relationship to owning entity-set indicated with a double diamond
  - One-to-many mapping
  - Total participation on weak entity side



# Weak Entity-Set Variations



- Can run into interesting variations:
  - A strong entity-set that owns several weak entity-sets
  - A weak entity-set that has multiple identifying entity-sets
- Example:




  - Other (possibly better) ways of modeling this too, e.g. make submission a strong entity-set with its own ID
- Don't forget: weak entity-sets can also have their own non-identifying relationship-sets, etc.

# Conversion to Relation Schemas



- Converting strong entity-sets is simple
  - Create a relation schema for each entity-set
  - Primary key of entity-set is primary key of relation schema
- Components of compound attributes are included directly in the schema
  - Relational model requires atomic attributes
- Multivalued attributes require a second relation
  - Includes primary key of entity-set, and "single-valued" version of attribute
- Derived attributes normally require a view
  - Must compute the attribute's value

# Schema Conversion Example



- *customer* entity-set:

| *customer* |
|---|
| *cust_id*<br>*name*<br>*address*<br>    *street*<br>    *city*<br>    *state*<br>    *zip_code*<br>{ *email* } |

- Maps to schema:

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

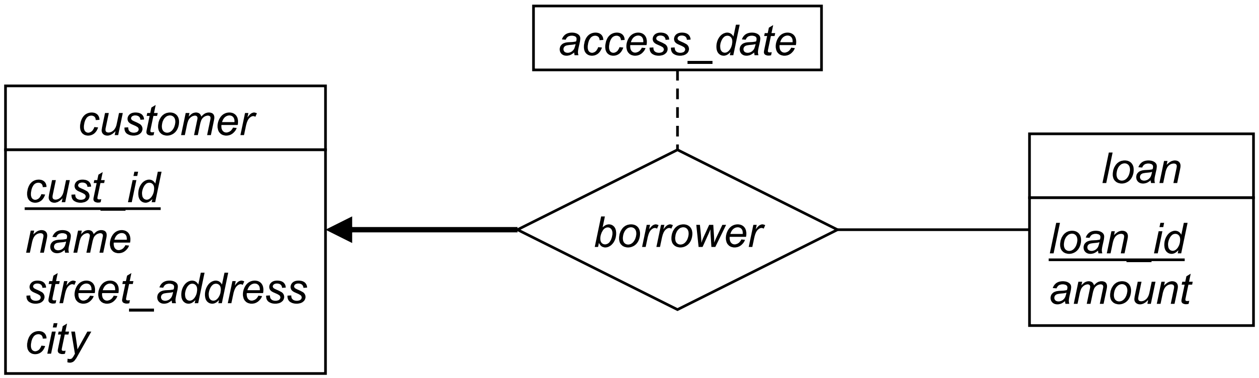
  *customer_emails*(*cust_id*, *email*)

- Primary-key attributes come first in attribute lists!

# Schema Conversion Example (2)



- Bank loans:




- Maps to schema:

*customer*(<u>*cust_id*</u>, *name*, *street_address*, *city*)

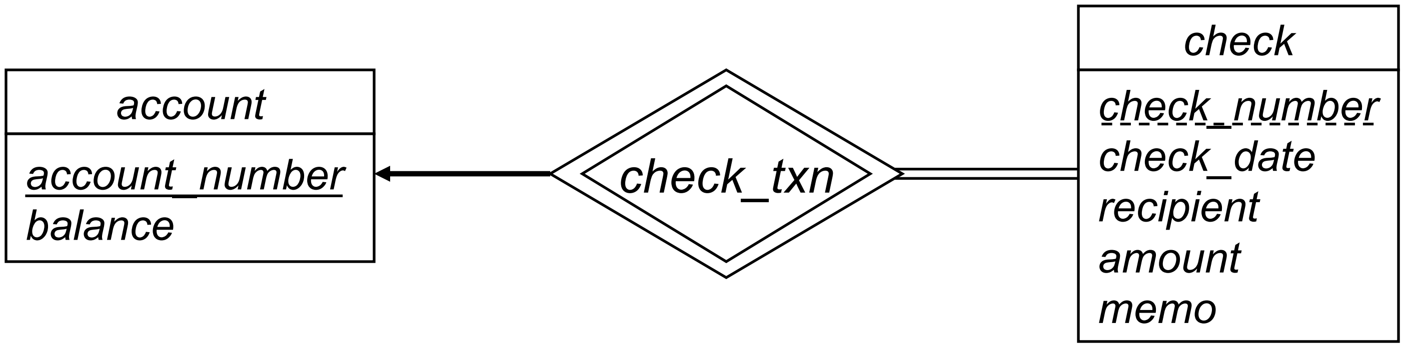
*loan*(<u>*loan_id*</u>, *amount*)

*borrower*(<u>*loan_id*</u>, *cust_id*, *access_date*)

# Schema Conversion Example (3)



- Checking accounts:




- Maps to schema:

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

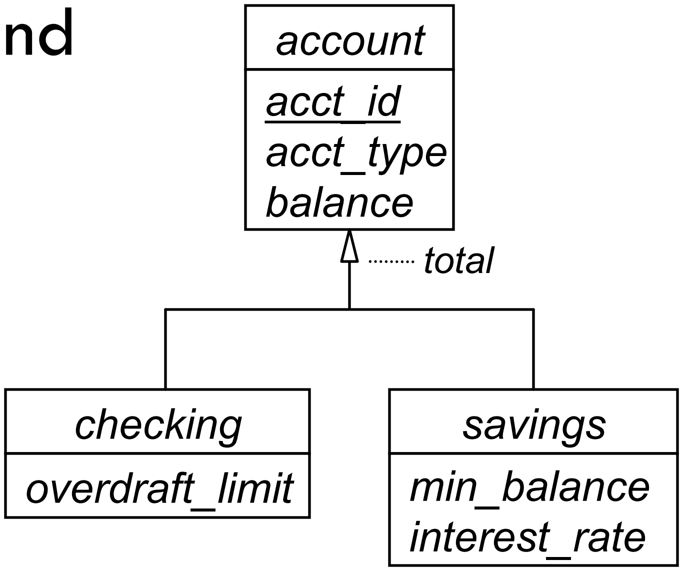
  *check*(*account_number*, *check_number*, *check_date*, *recipient*, *amount*, *memo*)

  - No schema for identifying relationship!

# Generalization and Specialization



- Use generalization when multiple entity-sets represent similar concepts
- Example: checking and savings accounts

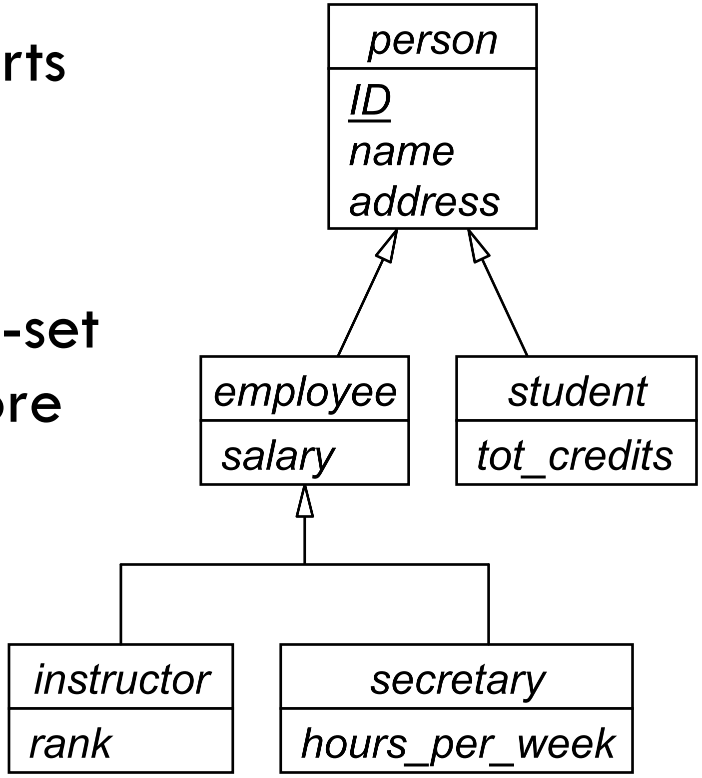



- Attributes <u>and relationships</u> are inherited
  - Subclass entity-sets can also have own relationships

# Specialization Constraints



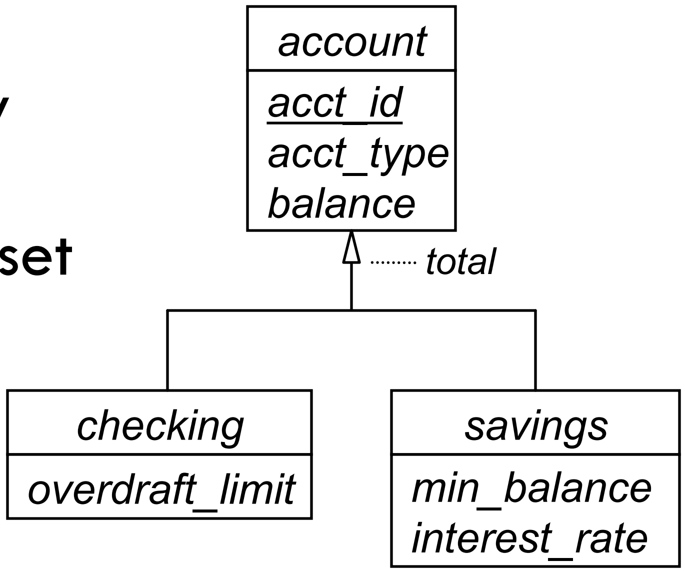
- Disjointness constraint, a.k.a. disjoint specialization:
  - Every entity in superclass entity-set can be a member of at most one subclass entity-set
  - One arrow split into multiple parts shows disjoint specialization
- Overlapping specialization:
  - An entity in the superclass entity-set can be a member of zero or more subclass entity-sets
  - Multiple separate arrows show overlapping specialization



# Specialization Constraints (2)



- Completeness constraint:
  - Total specialization: every entity in superclass entity-set must be a member of some subclass entity-set
  - Partial specialization is default
  - Show total specialization with "total" annotation on arrow

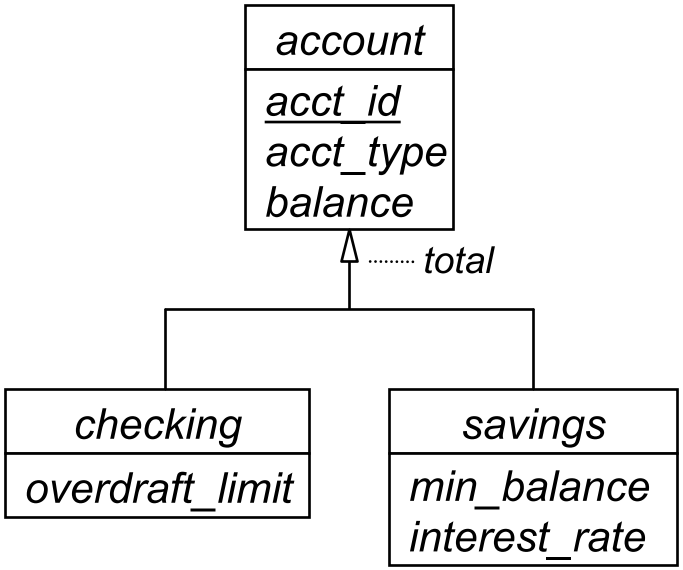



- Membership constraint:
  - What makes an entity a member of a subclass?
  - Attribute-defined vs. user-defined specialization

# Generalization Example



- Checking and savings accounts:




- One possible mapping to relation schemas:

  *account*(*<u>acct_id</u>*, *acct_type*, *balance*)

  *checking*(*<u>acct_id</u>*, *overdraft_limit*)

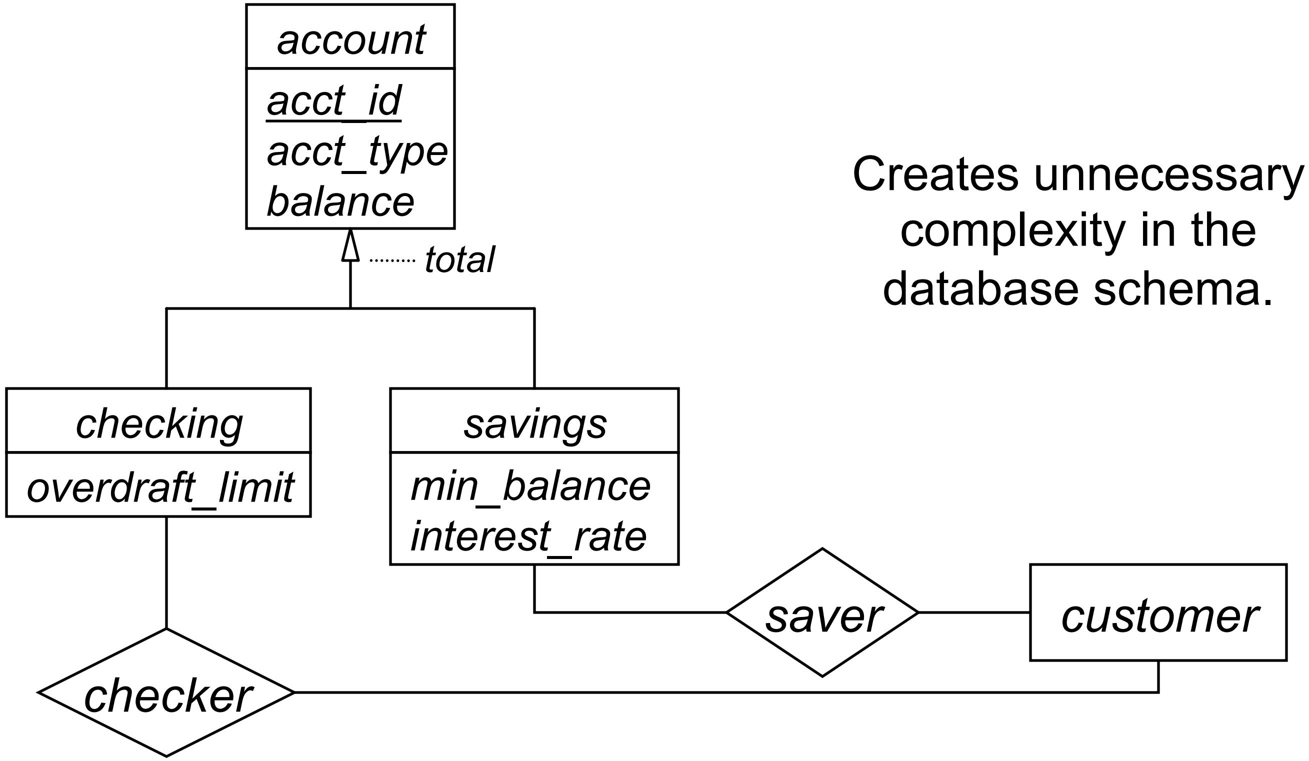
  *savings*(*<u>acct_id</u>*, *min_balance*, *interest_rate*)

- Be familiar with other mappings, and their tradeoffs

# Generalization and Relationships



- If all subclass entity-sets have a relationship with a particular entity-set:
  - e.g. all accounts are associated with customers
  - Don't create a separate relationship for each subclass entity-set!

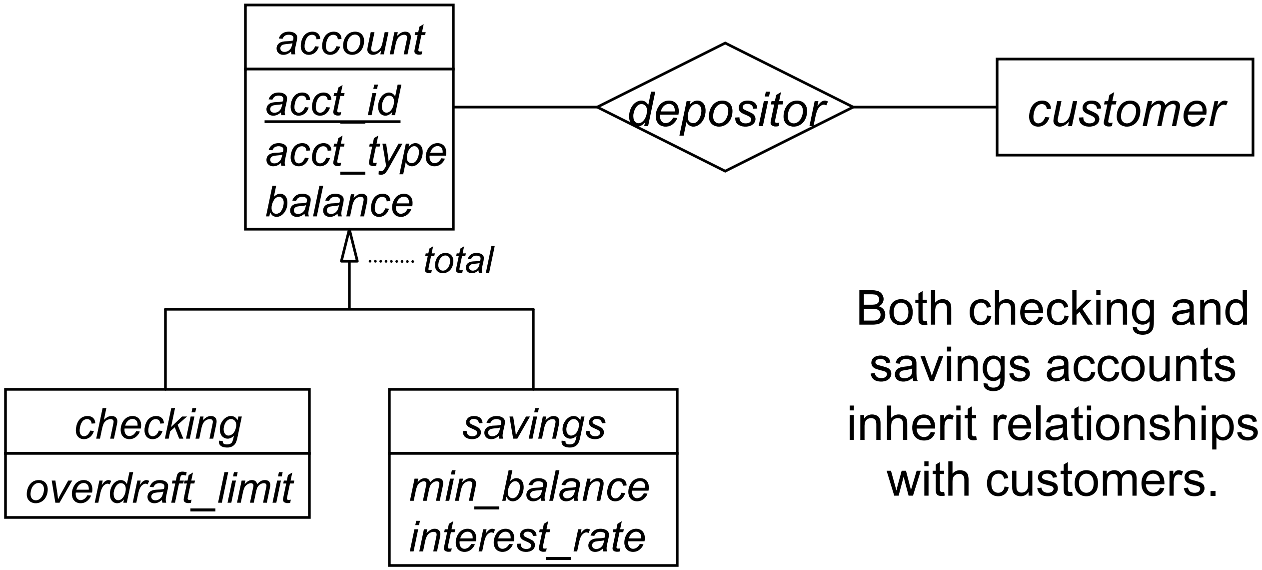



Creates unnecessary complexity in the database schema.

# Generalization, Relationships (2)



- If <u>all</u> subclass entity-sets have a relationship with a particular entity-set:
  - Create a relationship with superclass entity-set
  - Subclass entity-sets inherit this relationship

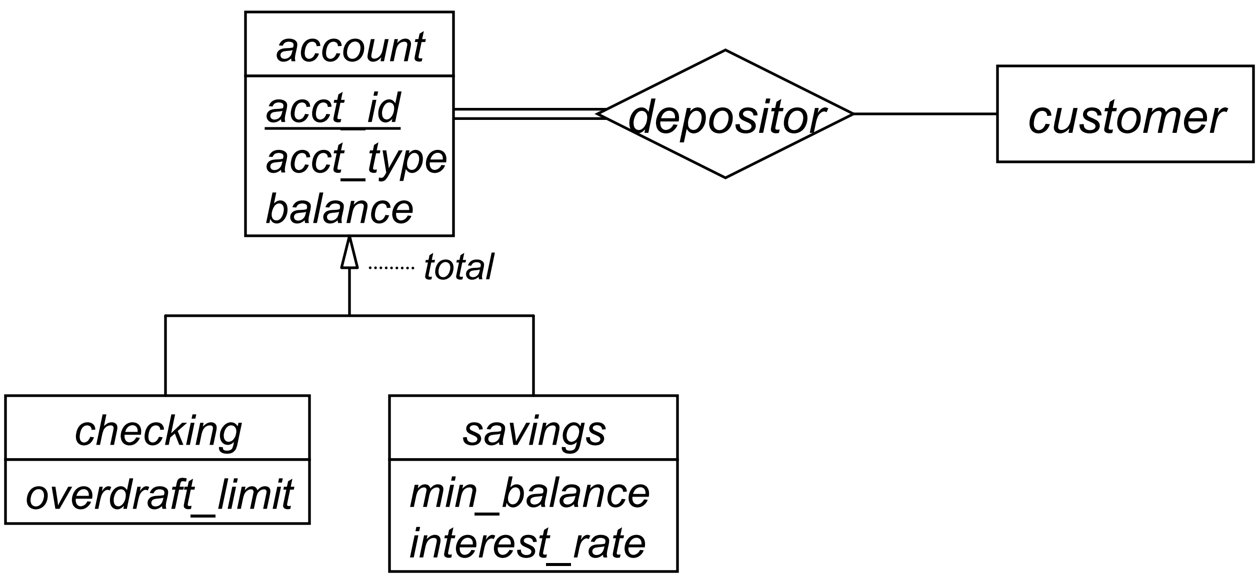



Both checking and savings accounts inherit relationships with customers.

# Generalization, Relationships (3)



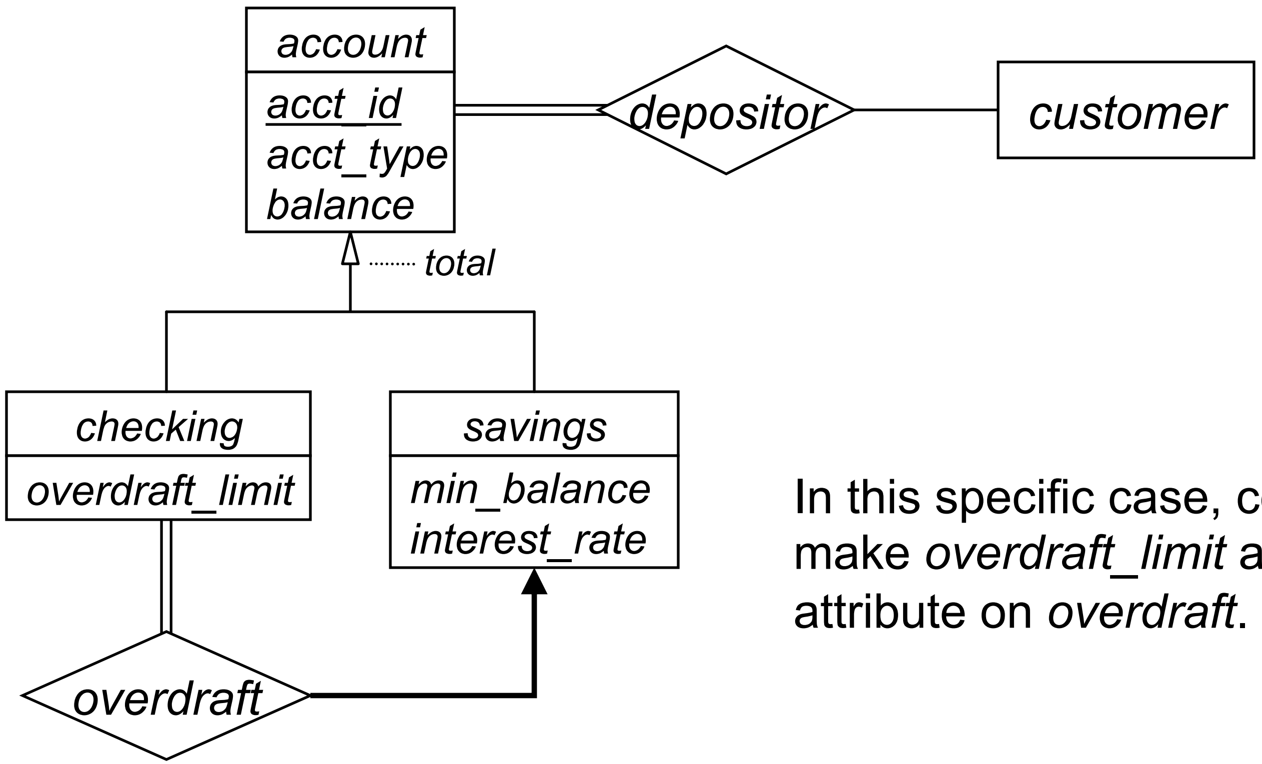
- Finally, ask yourself:
  - "What constraints should I enforce on *depositor* ?"
  - All accounts have to be associated with at least one customer
  - A customer may have zero or more accounts
  - *account* has total participation in *depositor*



# Generalization, Relationships (4)



- Subclass entity-sets can have their own relationships
  - e.g. associate every checking account with one specific "overdraft" savings account
  - What constraints on *overdraft* ?




In this specific case, could also make *overdraft_limit* a descriptive attribute on *overdraft*.

# Normal Forms



- Normal forms specify "good" patterns for database schemas
- First Normal Form (1NF)
  - All attributes must have atomic domains
  - Happens automatically in E-R to relational model conversion
- Second Normal Form (2NF) of historical interest
  - Don't need to know about it
- Higher normal forms use more formal concepts
  - Functional dependencies:  BCNF, 3NF
  - Multivalued dependencies:  4NF

# Normal Form Notes



- Make sure you can:
  - Identify and state functional dependencies and multivalued dependencies in a schema
  - Determine if a schema is in BCNF, 3NF, 4NF
  - Normalize a database schema
- Functional dependency requirements:
  - Apply rules of inference to functional dependencies
  - Compute the closure of an attribute-set
  - Compute $F_c$ from $F$, without any programs this time ☺
  - Identify extraneous attributes

# Functional Dependencies



- Given a relation schema $R$ with attribute-sets $\alpha, \beta \subseteq R$
  - The functional dependency $\alpha \rightarrow \beta$ holds on $r(R)$ if $\langle \forall\ t_1, t_2 \in r : t_1[\alpha] = t_2[\alpha] : t_1[\beta] = t_2[\beta] \rangle$
  - If $\alpha$ is the same, then $\beta$ must be the same too
- Trivial functional dependencies hold on all possible relation values
  - $\alpha \rightarrow \beta$ is trivial if $\beta \subseteq \alpha$
- A superkey functionally determines the schema
  - $K$ is a superkey if $K \rightarrow R$

# Inference Rules



- Armstrong's axioms:
  - Reflexivity rule:
    If $\alpha$ is a set of attributes and $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ holds.
  - Augmentation rule:
    If $\alpha \rightarrow \beta$ holds, and $\gamma$ is a set of attributes, then $\gamma\alpha \rightarrow \gamma\beta$ holds.
  - Transitivity rule:
    If $\alpha \rightarrow \beta$ holds, and $\beta \rightarrow \gamma$ holds, then $\alpha \rightarrow \gamma$ holds.
- Additional rules:
  - Union rule:
    If $\alpha \rightarrow \beta$ holds, and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds.
  - Decomposition rule:
    If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds.
  - Pseudotransitivity rule:
    If $\alpha \rightarrow \beta$ holds, and $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds.

# Sets of Functional Dependencies



- A set $F$ of functional dependencies
- $F^+$ is closure of $F$
  - Contains all functional dependencies in $F$
  - Contains all functional dependencies that can be logically inferred from $F$, too
  - Use Armstrong's axioms to generate $F^+$ from $F$
- $F_c$ is canonical cover of $F$
  - $F$ logically implies $F_c$, and $F_c$ logically implies $F$
  - No functional dependency has extraneous attributes
  - All dependencies have unique left-hand side
- **Review how to test if an attribute is extraneous!**

# Boyce-Codd Normal Form



- Eliminates all redundancy that can be discovered using functional dependencies
- Given:
  - Relation schema *R*
  - Set of functional dependencies *F*
- *R* is in BCNF with respect to *F* if:
  - For all functional dependencies $\alpha \rightarrow \beta$ in $F^+$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:
    - $\alpha \rightarrow \beta$ is a trivial dependency
    - $\alpha$ is a superkey for *R*
- Is <u>not</u> dependency-preserving
  - Some dependencies in *F* may not be preserved

# Third Normal Form



- A dependency-preserving normal form
  - Also allows more redundant information than BCNF
- Given:
  - Relation schema *R*, set of functional dependencies *F*
- *R* is in 3NF with respect to *F* if:
  - For all functional dependencies $\alpha \rightarrow \beta$ in $F^+$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:
    - $\alpha \rightarrow \beta$ is a trivial dependency
    - $\alpha$ is a superkey for *R*
    - Each attribute *A* in $\beta - \alpha$ is contained in a candidate key for *R*
- Can generate a 3NF schema from $F_c$

# Multivalued Dependencies



- Functional dependencies cannot represent multivalued attributes
  - Can't use functional dependencies to generate normalized schemas including multivalued attributes
- Multivalued dependencies are a generalization of functional dependencies
  - Represented as $\alpha \twoheadrightarrow \beta$
- More complex than functional dependencies!
  - Real-world usage is usually very simple
- Fourth Normal Form
  - Takes multivalued dependencies into account

# Multivalued Dependencies (2)



- Multivalued dependency $\alpha \twoheadrightarrow \beta$ holds on $R$ if, in any legal relation $r(R)$:
  - For all pairs of tuples $t_1$ and $t_2$ in $r$ such that $t_1[\alpha] = t_2[\alpha]$
  - There also exists tuples $t_3$ and $t_4$ in $r$ such that:
    - $t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$
    - $t_1[\beta] = t_3[\beta]$ and $t_2[\beta] = t_4[\beta]$
    - $t_1[R - \beta] = t_4[R - \beta]$ and $t_2[R - \beta] = t_3[R - \beta]$
- Pictorially:

| | $\alpha$ | $\beta$ | $R - (\alpha \cup \beta)$ |
|---|---|---|---|
| $t_1$ | $a_1 \ldots a_i$ | $a_{i+1} \ldots a_j$ | $a_{j+1} \ldots a_n$ |
| $t_2$ | $a_1 \ldots a_i$ | $b_{i+1} \ldots b_j$ | $b_{j+1} \ldots b_n$ |
| $t_3$ | $a_1 \ldots a_i$ | $a_{i+1} \ldots a_j$ | $b_{j+1} \ldots b_n$ |
| $t_4$ | $a_1 \ldots a_i$ | $b_{i+1} \ldots b_j$ | $a_{j+1} \ldots a_n$ |

# Trivial Multivalued Dependencies



- $\alpha \twoheadrightarrow \beta$ is a trivial multivalued dependency on *R* if all relations *r*(*R*) satisfy the dependency
- Specifically, $\alpha \twoheadrightarrow \beta$ is trivial if $\beta \subseteq \alpha$, or if $\alpha \cup \beta = R$

- Note that a multivalued dependency's trivial-ness may depend on the schema!
  - $A \twoheadrightarrow B$ is trivial on $R_1(A, B)$, but it is not trivial on $R_2(A, B, C)$
  - A major difference between functional and multivalued dependencies!
  - For functional dependencies: $\alpha \rightarrow \beta$ is trivial only if $\beta \subseteq \alpha$

# Functional & Multivalued Dependencies



- Functional dependencies are also multivalued dependencies
  - If $\alpha \rightarrow \beta$, then $\alpha \twoheadrightarrow \beta$ too
  - Additional caveat: each value of $\alpha$ has at most one associated value for $\beta$
- Don't state functional dependencies as multivalued dependencies!
  - Much easier to reason about functional dependencies!

# Functional & Multivalued Dependencies (2)



- Given a relation $R_1(\alpha, \beta)$ with $\alpha \rightarrow \beta$ and $\alpha \cap \beta = \varnothing$
  - What is the key of $R_1$?
  - $R_1(\underline{\alpha}, \beta)$
- Given a relation $R_2(\alpha, \beta)$ with $\alpha \twoheadrightarrow \beta$ and $\alpha \cap \beta = \varnothing$
  - What is the key of $R_2$?
  - $R_2(\underline{\alpha, \beta})$ – i.e. all attributes $\alpha \cup \beta$ are part of the key of $R_2$
- This is why we don't state functional dependencies as multivalued dependencies

# Fourth Normal Form



- Given:
  - Relation schema *R*
  - Set of functional and multivalued dependencies *D*
- *R* is in 4NF with respect to *D* if:
  - For all multivalued dependencies $\alpha \twoheadrightarrow \beta$ in $D^+$, where $\alpha \in R$ and $\beta \in R$, at least one of the following holds:
    - $\alpha \twoheadrightarrow \beta$ is a trivial multivalued dependency
    - $\alpha$ is a superkey for *R*
  - Note: If $\alpha \rightarrow \beta$ then $\alpha \twoheadrightarrow \beta$
- A database design is in 4NF if all schemas in the design are in 4NF