Welcome to CS 186, Section 6!

TA: Bryan Munar
OH: Mondays 11-12pm and Thursdays 2:30-3:30pm (651 Soda)
DISC: Tuesdays 11-12am (136 Barrows) and Wednesdays 10-11am (130 Wheeler)

Announcements and Such

- **Project/HW 3** due on Monday!

- How was the midterm? And how'd you do? **Come to OH if you wanna discuss anything**

- Any inconsistencies with score? **Ask for a regrade (when scores are released!)**

Discussion 6: Relational Algebra, Entity-Relationship Diagrams, and Functional Dependencies

Overview:

1. Relational Algebra
2. Worksheet exercises
3. Entity-Relationship Diagrams
4. Worksheet Exercises
5. Functional Dependencies
6. Worksheet Exercises

(A majority of the slides are from Michelle and lecture!)

Relational Algebra

IMPORTANT

Relational Algebra

- Input and output: Relation instances (tables)
- Has set semantics
  - No duplicate tuples in a relation
- Useful for representing semantics of execution plans in a DBMS (more later!)

Relational Algebra

Operation	Symbol	Explanation
Selection	σ	Selects rows
Projection	π	Selects columns
Union	$\cup$	Tuples in r1 or r2
Intersection	$\cap$	Tuples in r1 and r2
Cross-product	$\times$	Combines two relations
Join	$\bowtie$	Conditional cross-product
Difference	$-$	Tuples in r2 not in r1

Selection

- Select rows
- Example: $\sigma_{gpa > 3.5}(R)$

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6

Selection

- Select rows
- Example: $\sigma_{gpa > 3.5}(R)$

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6

Projection

- Select columns
- Example: $\pi_{name, sid}(R)$

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6

Projection

- Select columns
- Example: $\pi_{name, sid}(R)$

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6

Selection & Projection

- Example: $\pi_{name, sid}(\sigma_{gpa > 3.5}(R))$

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6

Selection & Projection

- Example: $\pi_{name, sid}(\sigma_{gpa > 3.5}(R))$

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6

Union

- Set union between two relations
- Example: $\sigma_{sid < 3}(R) \cup \sigma_{sid\%2 == 0}(R)$

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6

Union

- Set union between two relations
- Example: $\sigma_{sid < 3}(R) \cup \sigma_{sid\%2 == 0}(R)$

name	sid	gpa
Bob	1	3.7
Ron	2	1.2

$\cup$

name	sid	gpa
Ron	2	1.2
Al	4	4.0

Union

- Set union between two relations
- Example: $\sigma_{sid < 3}(R) \cup \sigma_{sid\%2 == 0}(R)$

name	sid	gpa
Bob	1	3.7
Ron	2	1.2
Al	4	4.0

Intersection

- Set intersection between two relations
- Example: $\sigma_{sid < 3}(R) \cap \sigma_{sid\%2 == 0}(R)$

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6

Intersection

- Set intersection between two relations
- Example: $\sigma_{sid < 3}(R) \cap \sigma_{sid\%2 == 0}(R)$

name	sid	gpa
Bob	1	3.7
Ron	2	1.2

$\cap$

name	sid	gpa
Ron	2	1.2
Al	4	4.0

Intersection

- Set intersection between two relations
- Example: $\sigma_{sid < 3}(R) \cap \sigma_{sid\%2 == 0}(R)$

name	sid	gpa
Ron	2	1.2

Cross Product

- Takes all rows from A and combines with all rows in B
- Example: $\pi_{name}(R) \times \pi_{gpa}(R)$

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2

Cross Product

- Takes all rows from A and combines with all rows in B
- Example: $\pi_{name}(R) \times \pi_{gpa}(R)$

name
Bob
Sue
Ron

$\times$

gpa
3.7
2.9
1.2

Cross Product

- Takes all rows from A and combines with all rows in B
- Example: $\pi_{name}(R) \times \pi_{gpa}(R)$

name
Bob
Sue
Ron

×

gpa
3.7
2.9
1.2

=

name	gpa
Bob	3.7
Bob	2.9
Bob	1.2
Sue	3.7
Sue	2.9
Sue	1.2
Ron	3.7
Ron	2.9
Ron	1.2

Join

- Joins A and B based on some column
- Example: $\pi_{name,sid}(R) \bowtie \pi_{name,gpa}(R)$

name	sid
Bob	1
Sue	3
Ron	2

$\bowtie$

name	gpa
Bob	3.7
Sue	2.9
Ron	1.2

Join

- Joins A and B based on some column
- Example: $\pi_{name,sid}(R) \bowtie \pi_{name,gpa}(R)$

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2

Difference

- Takes rows in A that are not in B
- Example: $\sigma_{gpa > 3.5}(R) - \sigma_{sid\%2==0}(R)$

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6

Difference

- Takes rows in A that are not in B
- Example: $\sigma_{gpa > 3.5}(R)$ - $\sigma_{sid\%2==0}(R)$

name	sid	gpa
Bob	1	3.7
Al	4	4.0
Sally	5	3.6

-

name	sid	gpa
Ron	2	1.2
Al	4	4.0

Difference

- Takes rows in A that are not in B
- Example: $\sigma_{gpa > 3.5}(R) - \sigma_{sid\%2==0}(R)$

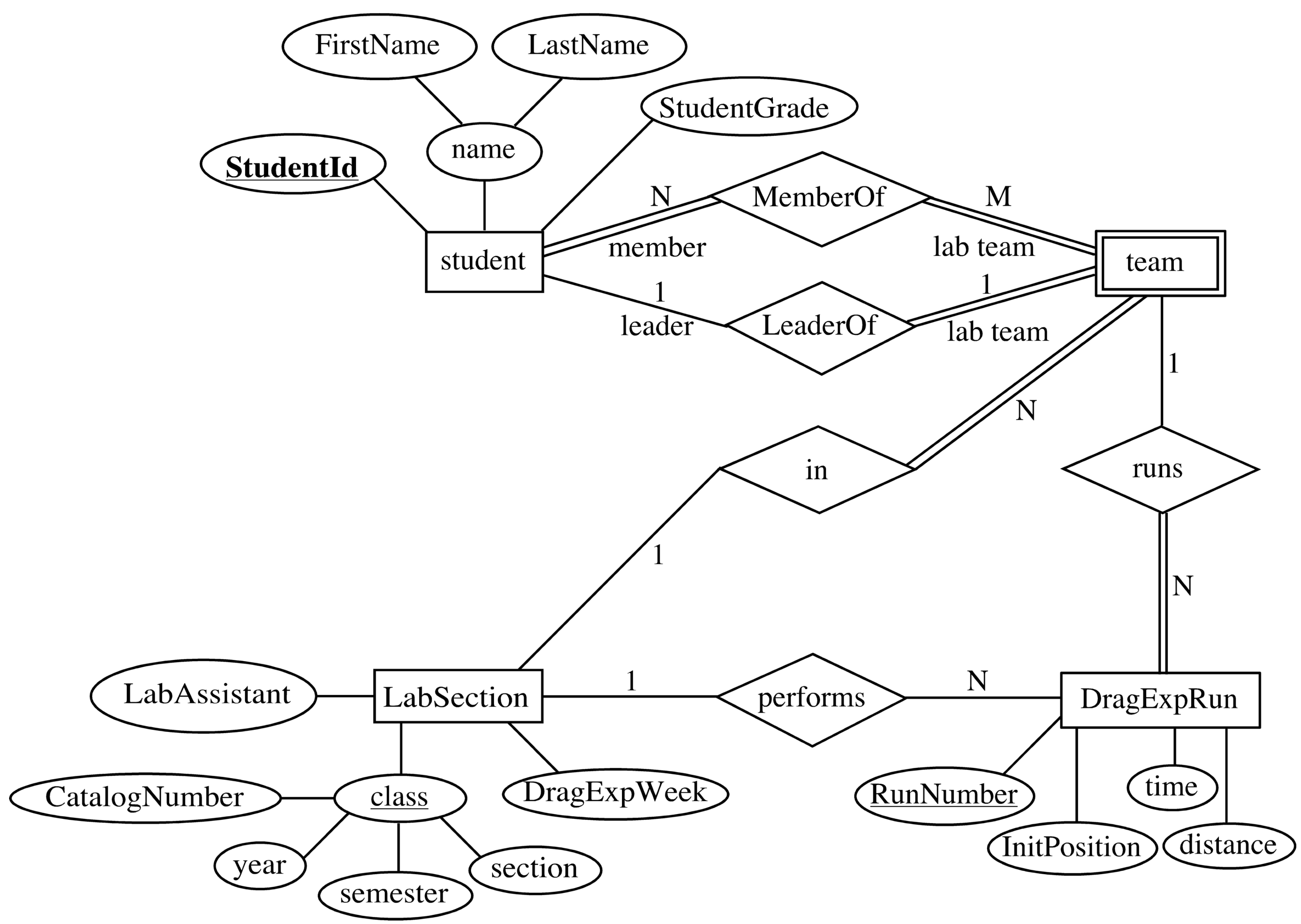
name	sid	gpa
Bob	1	3.7
Sally	5	3.6

Do first page of worksheet!

Entity Relationship Diagrams

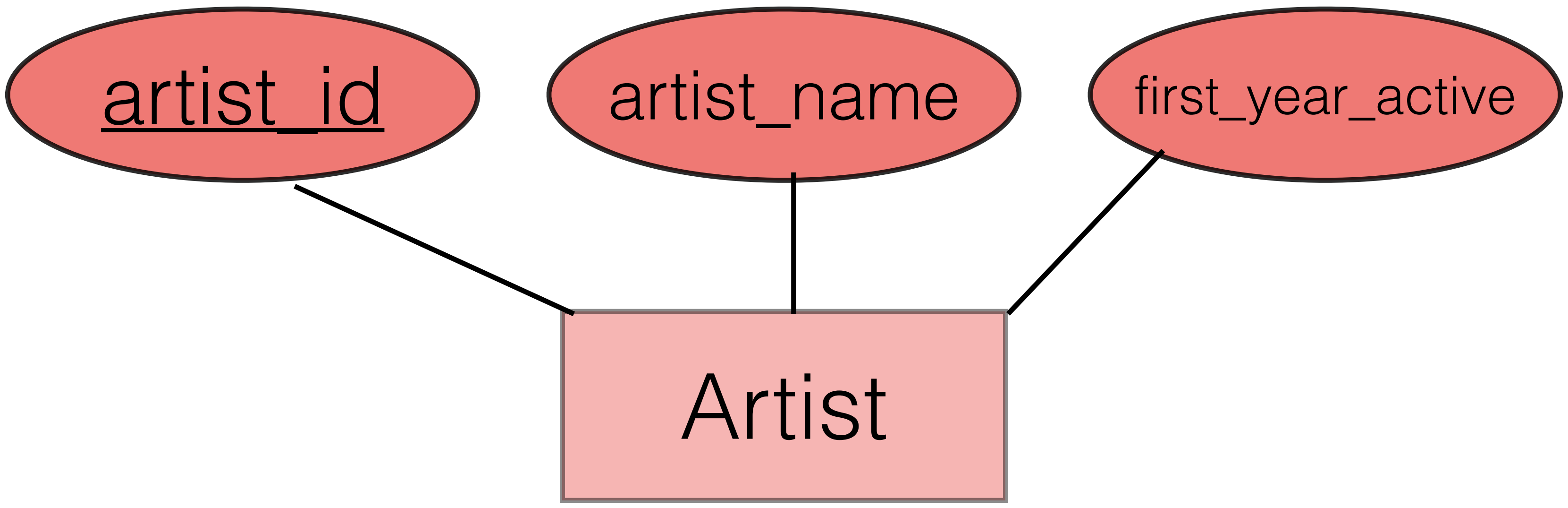
Motivation

- Visualize data schema



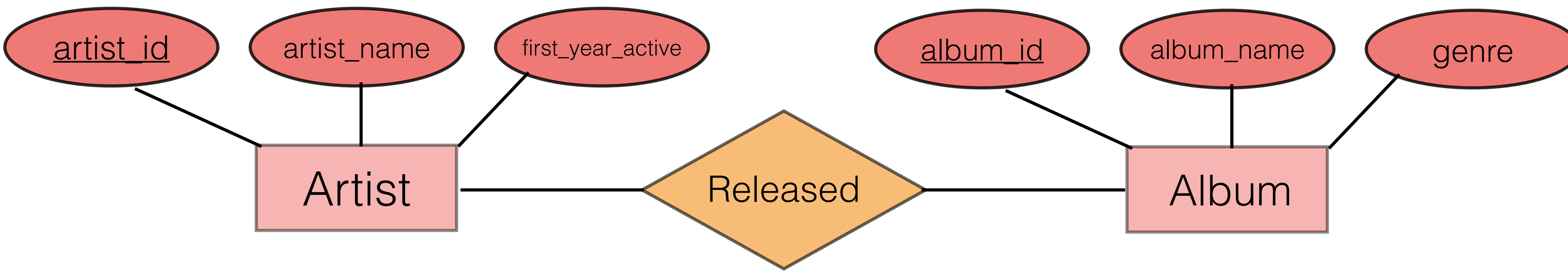
Entities

- Entity: "thing"
- Attribute: Property of the entity
  - Primary key underlined



Relationships

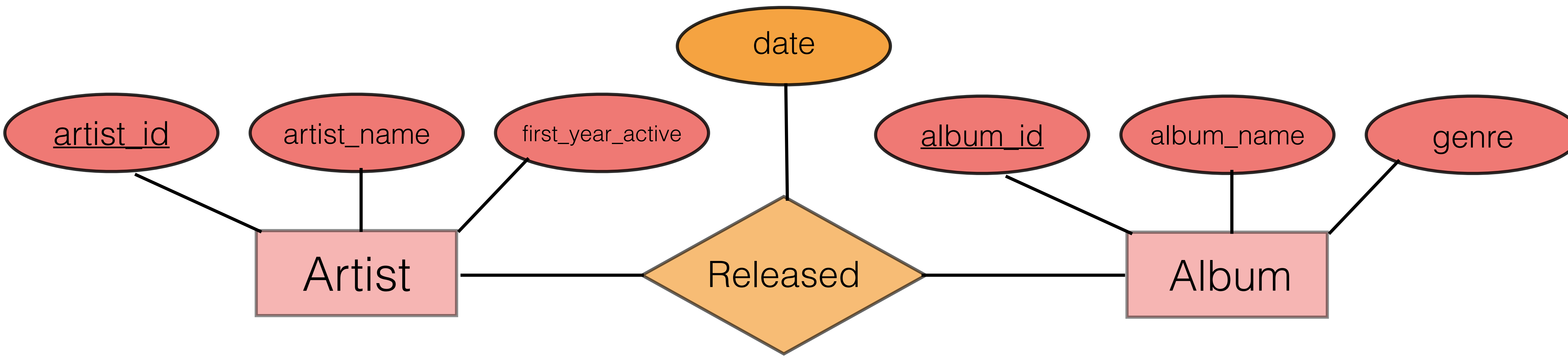
- How two entities interact



Artist 4 released album 2.

Relationships

- How two entities interact
  - Interactions can have attributes



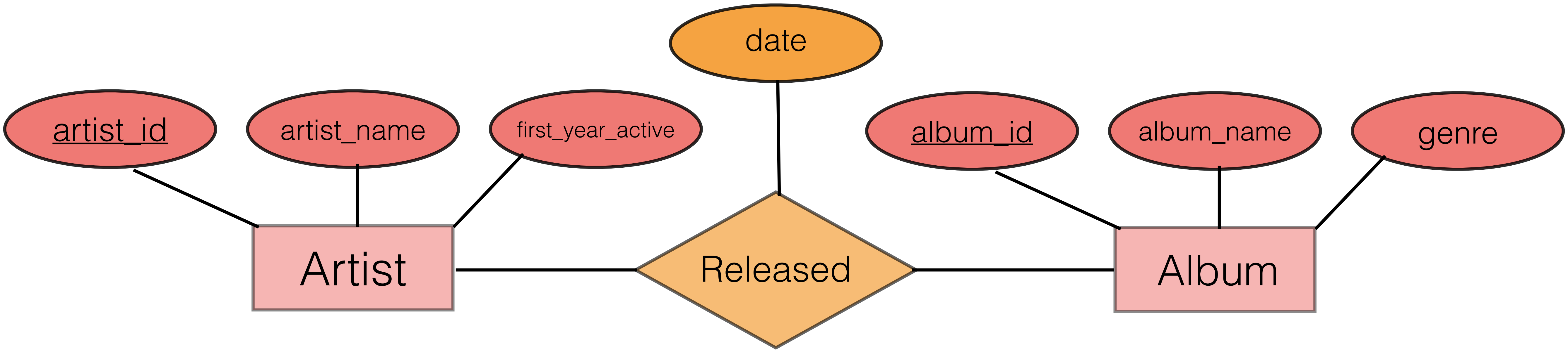
Artist 4 released album 2 on February 27, 2015.

Constraints

- Make relationship lines meaningful
  - Participation constraint (Partial/Total)
    - Total participation: participates at least once
  - Key/Non-key constraint
    - Key: Participates at most once

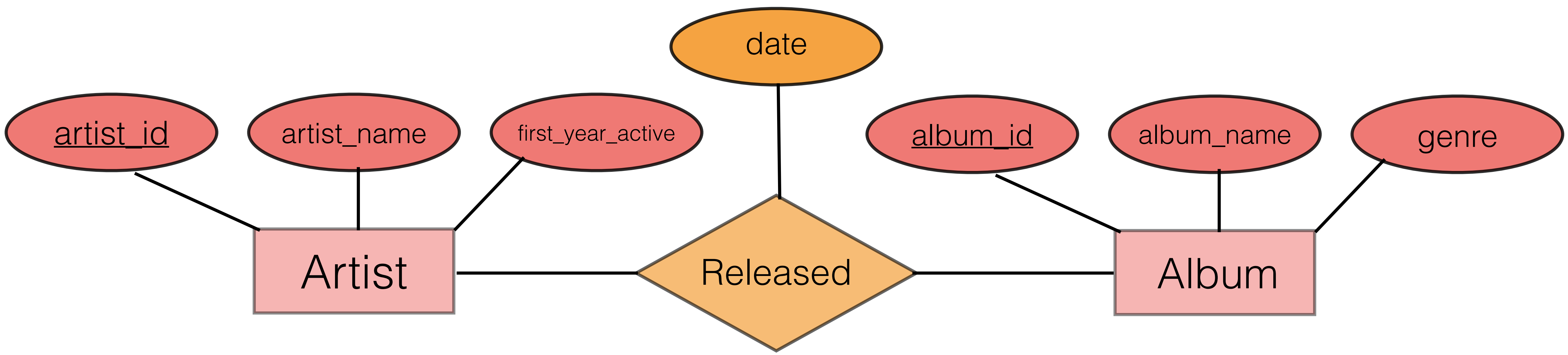
	Partial Participation	Total Participation
Non-Key	0 or More	1 or More
Key	0 or 1	Exactly 1

Constraints



Non-Key constraint with partial participation

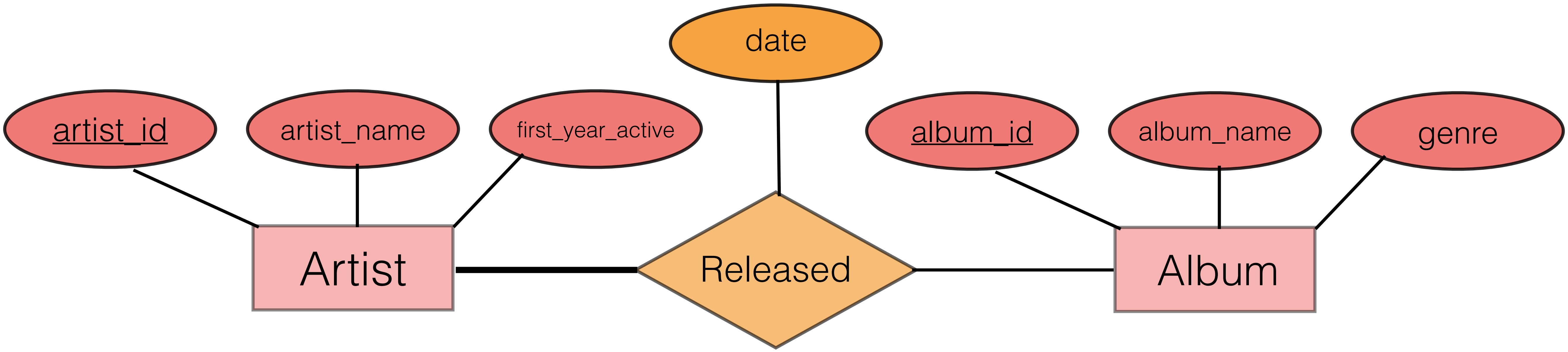
Constraints



Non-Key constraint with partial participation

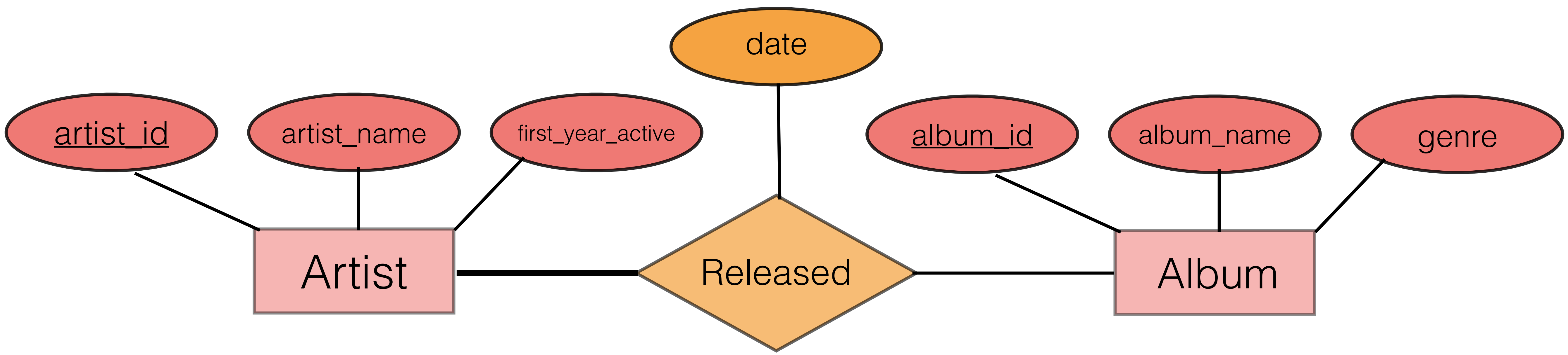
An artist can release an album zero or more times.

Constraints



Non-Key constraint with total participation

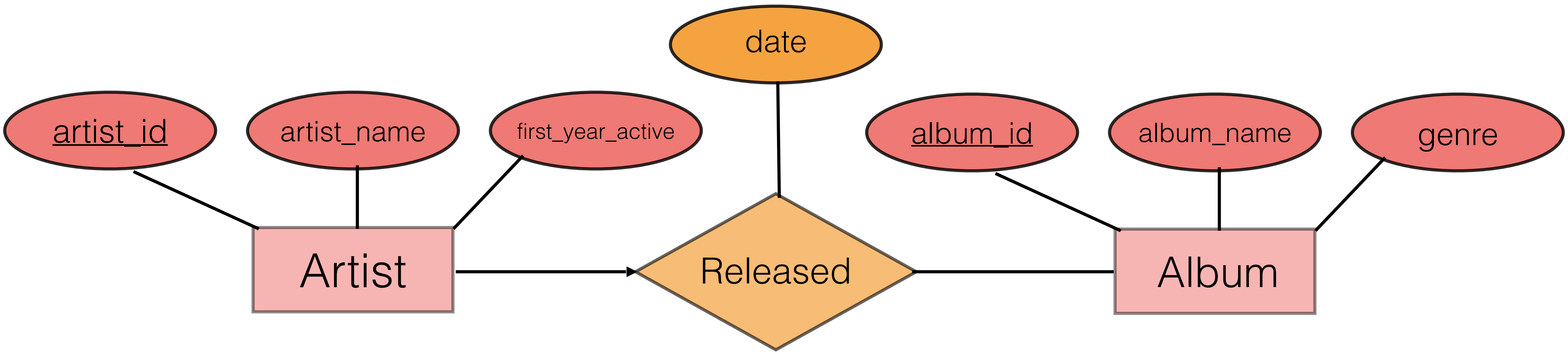
Constraints



Non-Key constraint with total participation

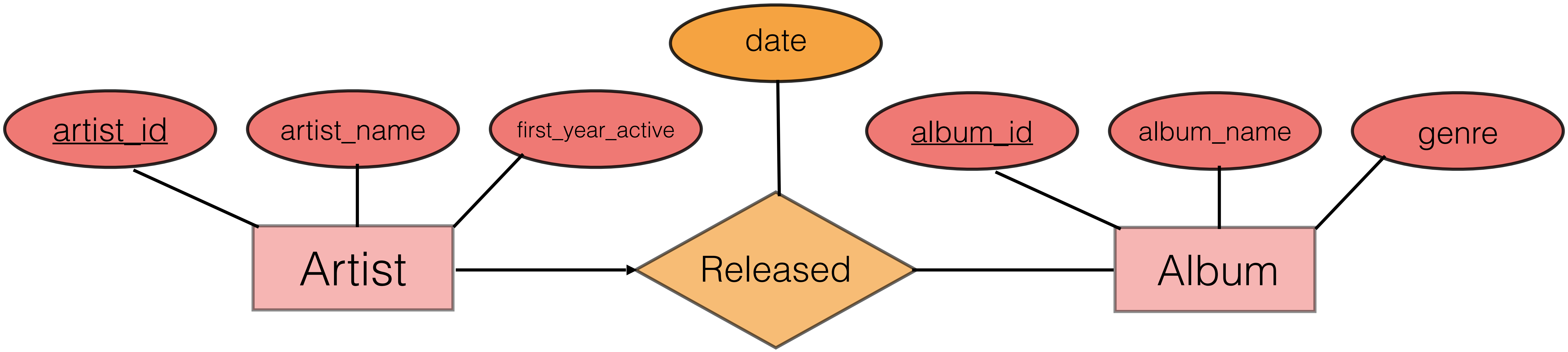
An artist can release an album one or more times.

Constraints



Key constraint with partial participation

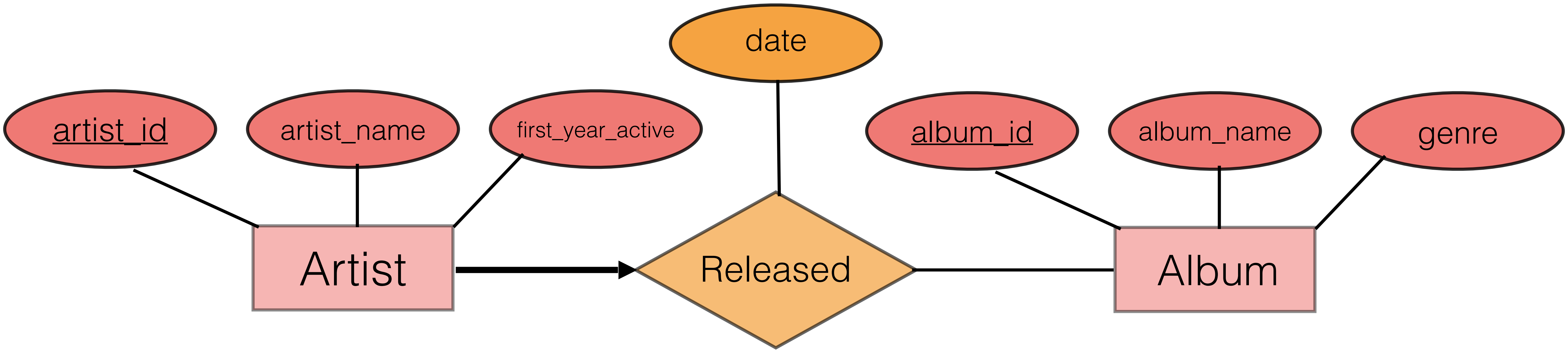
Constraints



Key constraint with partial participation

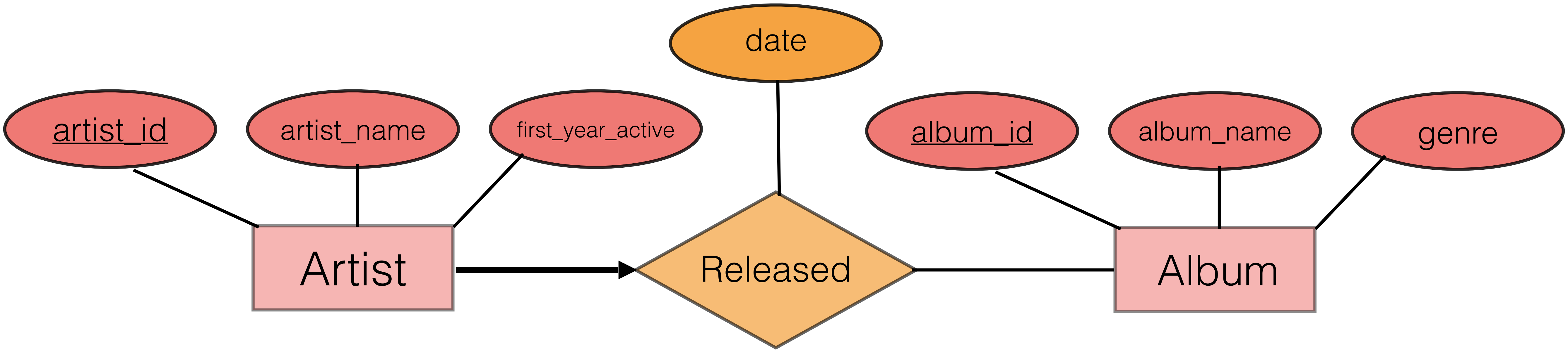
An artist can release an album zero or one times.

Constraints



Key constraint with total participation

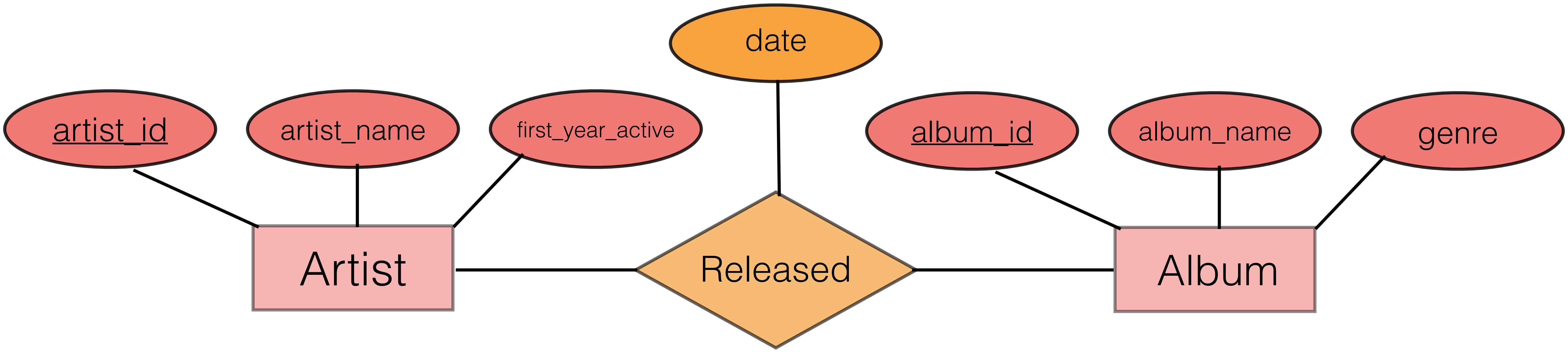
Constraints



Key constraint with total participation

An artist can release exactly one album.

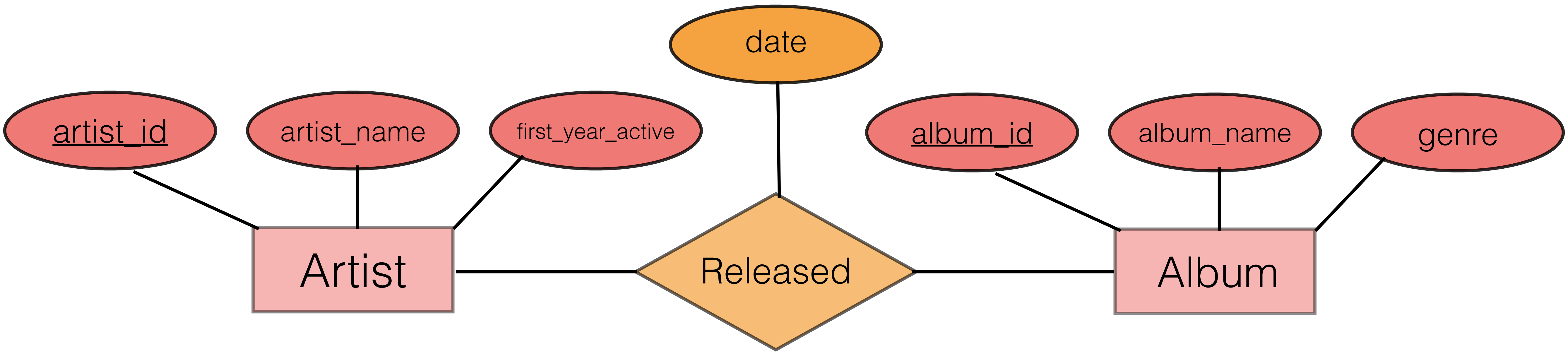
We want...



Non-Key constraint with partial participation

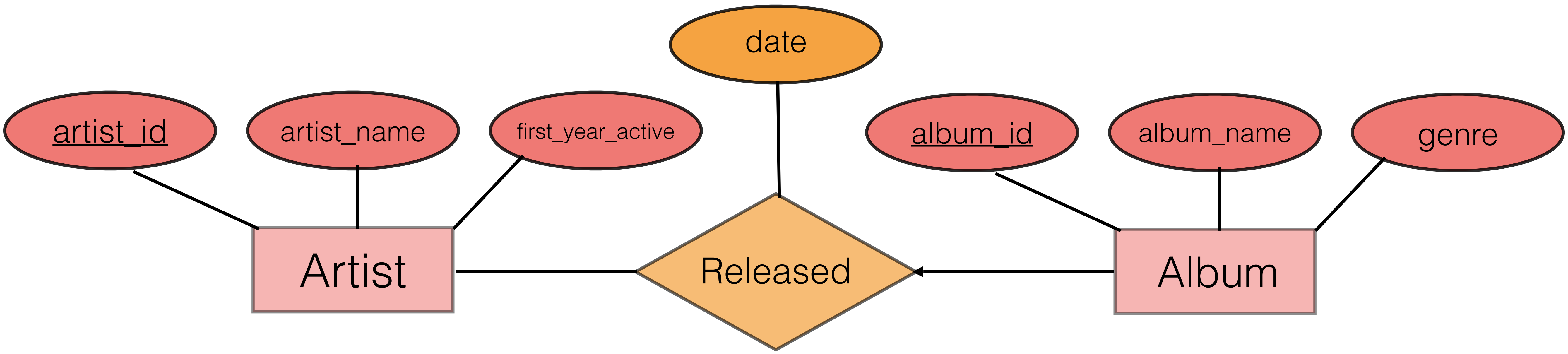
An artist can release an album zero or more times.

What constraint do we want from Album?



A. An album can be released 0 or more times.
B. An album can be released 1 or more times.
C. An album can be released 0 or 1 times.
D. An album is released exactly once.

What constraint do we want from Album?

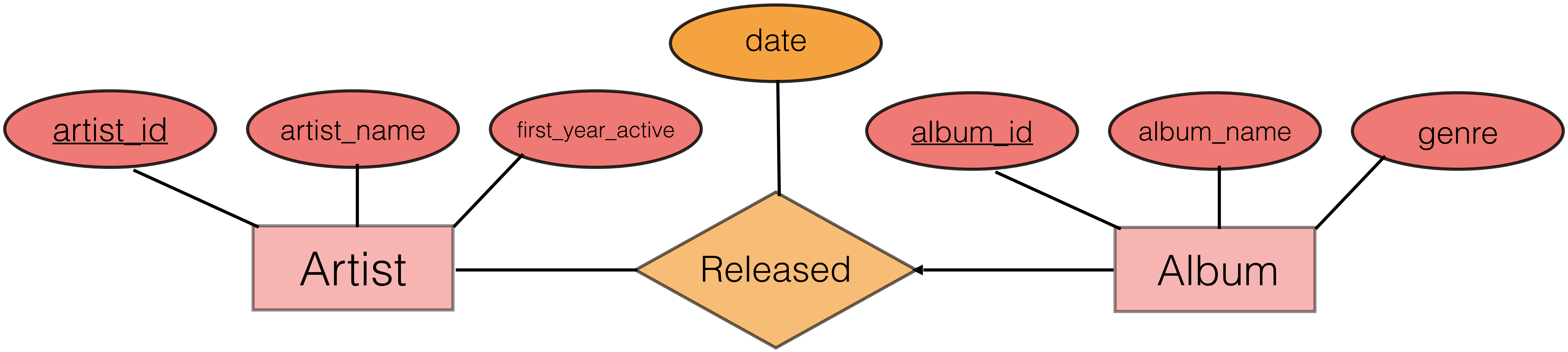


A. An album can be released 0 or more times.

B. An album can be released 1 or more times.

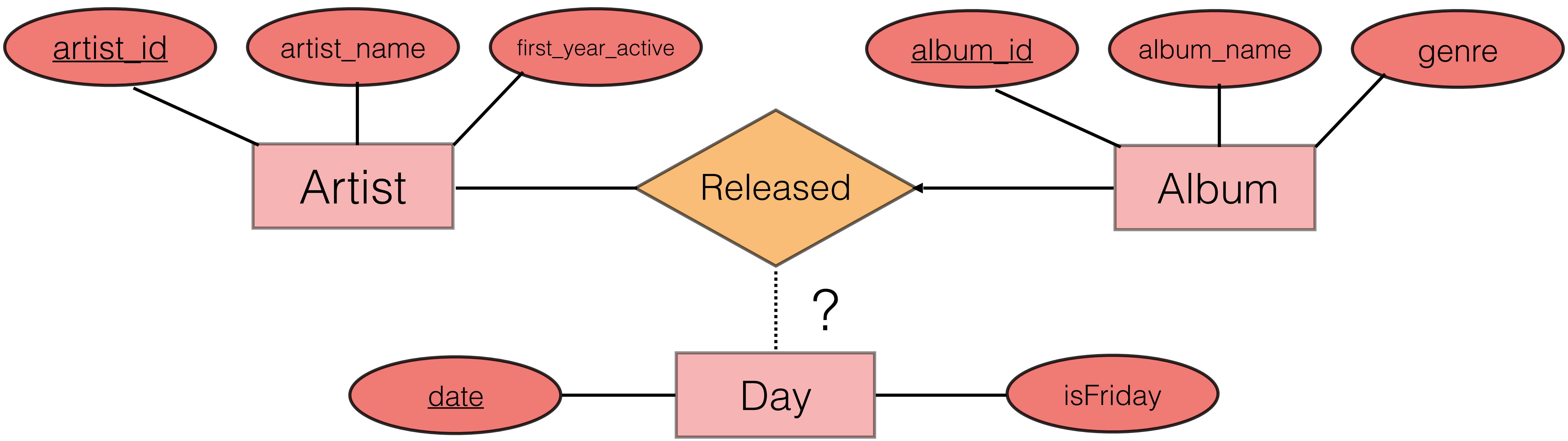
C. An album can be released 0 or 1 times.

D. An album is released exactly once.



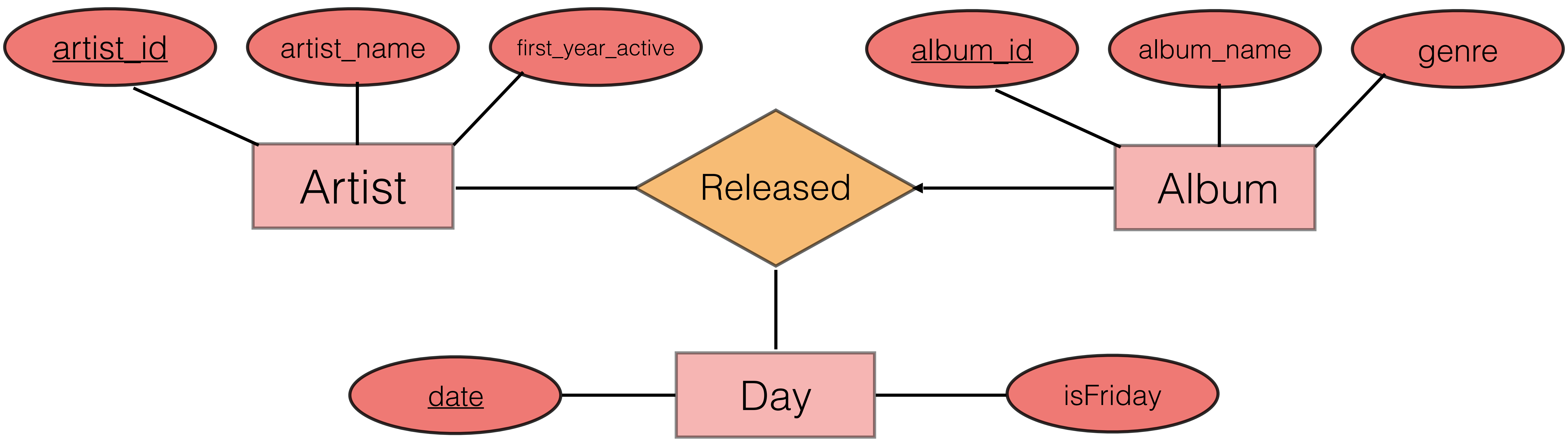
An artist may release zero or more albums.
An album may be released or unreleased.

Ternary Relations



An artist may release zero or more albums.
An album may be released or unreleased.
Releasing an album can occur ??? times a day.

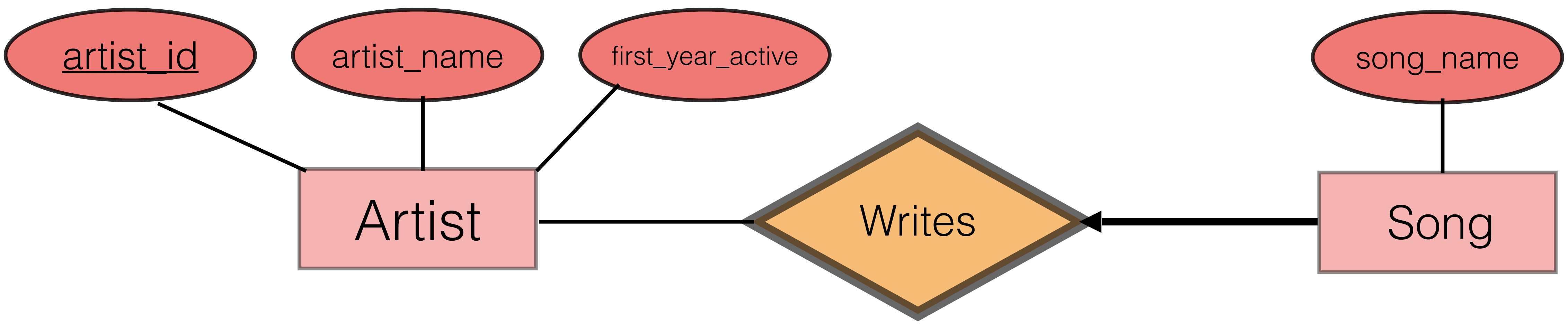
Ternary Relations



An artist may release zero or more albums.
An album may be released or unreleased.
Releasing an album can occur 0 or more times a day.

Weak Entities

- Weak entity can only be identified only when considering primary key of another (owner) entity.



- Song's key is actually (Artist.artist_id, Song.song_name)
- Can there be two songs with the same name?
  - How about by the same artist?
- Can a song exist without an artist?

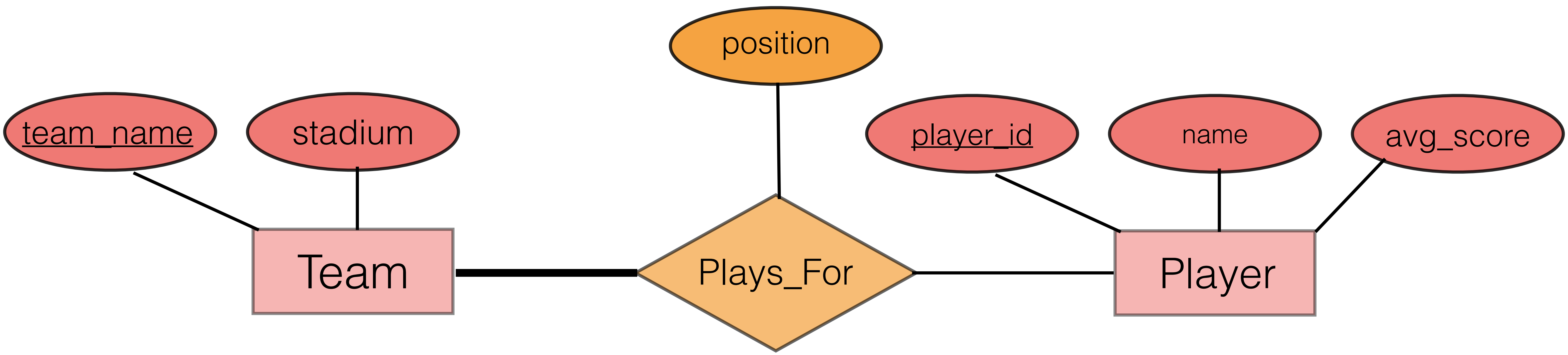
Do second page of worksheet!

Every Team in our database will have a team_id, a team_name and a stadium where they play their games. Each Player will have a player_id, name and their average score (This can be used for any sport!). Finally our database will contain who Plays_For which team and also the "position" that the player plays in.

- Assume that a player can play in more than one team (Yes, our league has different rules!) and that a team needs at least one player.

Every Team in our database will have a team_id, a team_name and a stadium where they play their games. Each Player will have a player_id, name and their average score (This can be used for any sport!). Finally our database will contain who Plays_For which team and also the “position” that the player plays in.

- Assume that a player can play in more than one team (Yes, our league has different rules!) and that a team needs at least one player.

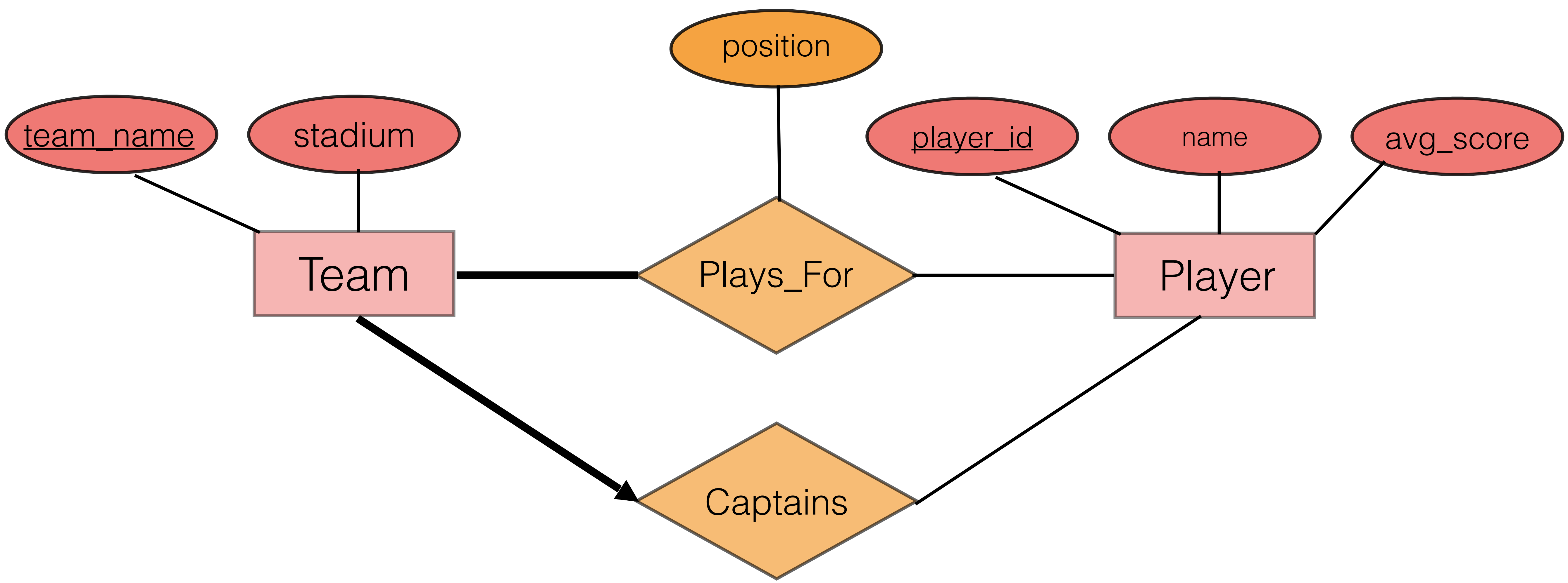


Every Team in our database will have a team_id, a team_name and a stadium where they play their games. Each Player will have a player_id, name and their average score (This can be used for any sport!). Finally our database will contain who Plays_For which team and also the "position" that the player plays in.

- Now let's say we want to also track who is the captain of every team. How will the ER diagram change from the previous case? Note: Every team needs exactly one captain!

Every Team in our database will have a team_id, a team_name and a stadium where they play their games. Each Player will have a player_id, name and their average score (This can be used for any sport!). Finally our database will contain who Plays_For which team and also the "position" that the player plays in.

- Now let's say we want to also track who is the captain of every team. How will the ER diagram change from the previous case? Note: Every team needs exactly one captain!

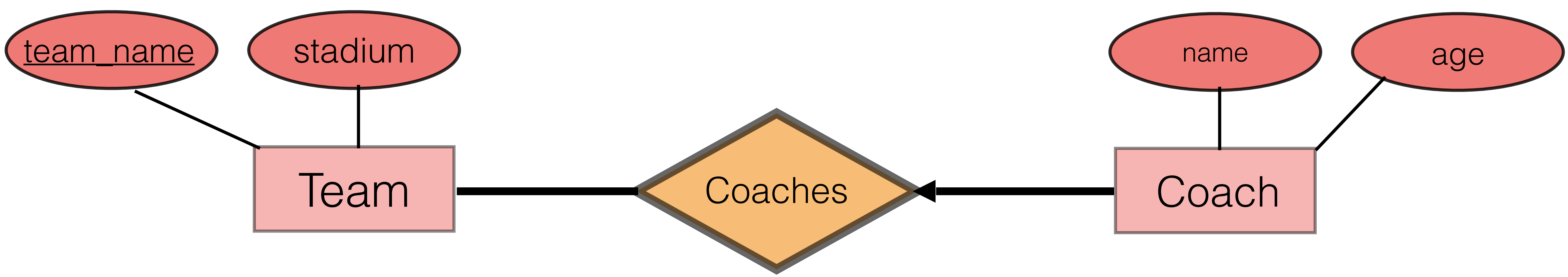


Every Team in our database will have a team_id, a team_name and a stadium where they play their games. Each Player will have a player_id, name and their average score (This can be used for any sport!). Finally our database will contain who Plays_For which team and also the "position" that the player plays in.

- Are there are any weak-entity relationships in either of our ER diagrams?

Every Team in our database will have a team_id, a team_name and a stadium where they play their games. Each Player will have a player_id, name and their average score (This can be used for any sport!). Finally our database will contain who Plays_For which team and also the "position" that the player plays in.

- Are there are any weak-entity relationships in either of our ER diagrams?
- No. A weak entity can be identified uniquely only by considering the primary key of another (owner) entity.
- Example of possible weak entity: Coaches



A team can have many coaches, but each coach exactly coaches one team.

Functional Dependencies

IMPORTANT

Functional Dependencies

- X → Y reads “X determines Y”
- Used to detect redundancies and refine schema

id	rating	wage
1	3	20
2	2	10
3	3	20
4	2	10

Functional Dependencies

- X → Y reads "X determines Y"
- Used to detect redundancies and refine schema

id	rating	wage
1	3	20
2	2	10
3	3	20
4	2	10

rating → wage

Functional Dependencies

- X → Y reads “X determines Y”
- Used to detect redundancies and refine schema

id	rating	wage
1	3	20
2	2	10
3	3	20
4	2	10

id → rating, wage

Functional Dependencies

- Key → All attributes of relation

id	rating	wage
1	3	20
2	2	10
3	3	20
4	2	10

id → rating, wage

Armstrong's Axioms

- Reflexivity: if $X \supseteq Y$, then $X \rightarrow Y$
  - Examples: $A \rightarrow A$, $AB \rightarrow A$
- Augmentation: if $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
- Transitivity: if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- Useful rules derived from Armstrong's Axioms:
  - Union: if $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - Decomposition: if $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

Armstrong's Axioms

- If $XA \rightarrow YA$, can you infer $X \rightarrow Y$?

Armstrong's Axioms

- If XA → YA, can you infer X → Y?
  - Trivial example: A → YA

Closures

- Functional dependency closure: F+
  - Set of all FDs implied by F, including trivial dependencies
  - Example: F = {A → B, B → C}
    - F+ = {A → B, B → C, A → C, A → A, A → AB, …}

Closures

- Attribute closure: X+
  - Given just X, what can we determine?
  - Example: F = {A → B, B → C}
    - A+ = ABC

Attribute Closures

- A methodical algorithm, given a set of FDs F:
  - Initialize X+ := X
  - Repeat until no change:
    - If U → V is in F such that U is in X+, add V to X+

- R = ABCDE
- F = {B → CD, D → E, B → A, E → C, AD → B }
- B+ = ?

Attribute Closures

- A methodical algorithm, given a set of FDs F:
  - Initialize X+ := X
  - Repeat until no change:
    - If U → V is in F such that U is in X+, add V to X+

- R = ABCDE
- F = {B → CD, D → E, B → A, E → C, AD → B }
- B+ = B

Attribute Closures

- A methodical algorithm, given a set of FDs F:
  - Initialize X+ := X
  - Repeat until no change:
    - If U → V is in F such that U is in X+, add V to X+

- R = ABCDE
- F = {B → CD, D → E, B → A, E → C, AD → B }
- B+ = BCD

Attribute Closures

- A methodical algorithm, given a set of FDs F:
  - Initialize X+ := X
  - Repeat until no change:
    - If U → V is in F such that U is in X+, add V to X+

- R = ABCDE
- F = {B → CD, D → E, B → A, E → C, AD → B }
- B+ = BCDE

Attribute Closures

- A methodical algorithm, given a set of FDs F:
  - Initialize X+ := X
  - Repeat until no change:
    - If U → V is in F such that U is in X+, add V to X+

- R = ABCDE
- F = {B → CD, D → E, B → A, E → C, AD → B }
- B+ = ABCDE

Attribute Closures

- A methodical algorithm, given a set of FDs F:
  - Initialize X+ := X
  - Repeat until no change:
    - If U → V is in F such that U is in X+, add V to X+

- R = ABCDE
- F = {B → CD, D → E, B → A, E → C, AD → B }
- B+ = ABCDE   B is a key of R!

Boyce-Codd Normal Form (BCNF)

- Motivation: Schema design is hard, want a way to ensure a reasonable design
- BCNF ≈ “reasonable schema”

Boyce-Codd Normal Form (BCNF)

- Definition: Relation R with FDs F is in BCNF if for all $X \rightarrow A$ in F+:
  - $X \rightarrow A$ is reflexive (a trivial FD) OR
  - X is a superkey for R
    - Superkey: Key that does not need to be minimal

BCNF Decomposition

- If $X \rightarrow A$ violates BCNF, decompose R into R - A and XA
  - Repeat as necessary

- R = ABCEG
- F = $\{AB \rightarrow C, AC \rightarrow B, BC \rightarrow G, E \rightarrow G\}$

BCNF Decomposition

- If $X \to A$ violates BCNF, decompose R into R - A and XA
  - Repeat as necessary

- R = ABCEG
- F = $\{AB \to C, AC \to B, BC \to G, E \to G\}$
- ABEG, ABC

BCNF Decomposition

- If $X \rightarrow A$ violates BCNF, decompose R into R - A and XA
  - Repeat as necessary

- R = ABCEG
- F = $\{AB \rightarrow C, AC \rightarrow B, BC \rightarrow G, E \rightarrow G\}$
- ABEG, ABC

BCNF Decomposition

- If $X \rightarrow A$ violates BCNF, decompose R into R - A and XA
  - Repeat as necessary

- R = ABCEG
- F = {$AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow G$, $E \rightarrow G$}
- ABEG, ABC

BCNF Decomposition

- If $X \rightarrow A$ violates BCNF, decompose R into R - A and XA
  - Repeat as necessary

- R = ABCEG
- F = {$AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow G$, $E \rightarrow G$}
- ABEG, ABC
- ABE, EG, ABC

BCNF Decomposition

- If $X \rightarrow A$ violates BCNF, decompose R into R - A and XA
  - Repeat as necessary

- R = ABCEG
- F = {$AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow G$, $E \rightarrow G$}
- ABEG, ABC
- ABE, EG, ABC

Decomposition Properties

- Lossless join: Can we reconstruct R?
  - Decomposing R into X and Y is lossless iff:
    - $X \cap Y \rightarrow X$, or
    - $X \cap Y \rightarrow Y$

Decomposition Properties

- Lossless join: Can we reconstruct R?
  - Decomposing R into X and Y is lossless iff:
    - $X \cap Y \rightarrow X$, or
    - $X \cap Y \rightarrow Y$
- Example:
  - ABC decomposed to AB, BC
  - FDs: $A \rightarrow B$, $C \rightarrow B$
  - This is lossy! $AB \cap BC = B \rightarrow B$

Decomposition Properties

- Lossless join: Can we reconstruct R?
  - Decomposing R into X and Y is lossless iff:
    - $X \cap Y \rightarrow X$, or
    - $X \cap Y \rightarrow Y$
- Example:
  - ABC decomposed to AC, BC
  - FDs: $A \rightarrow B$, $C \rightarrow A$
  - This is lossless! $AC \cap BC = C \rightarrow AC$

Decomposition Properties

- Dependency-preserving: Can we verify all FDs without joins?

- Example: ABE, EG, ABC
  - $F = \{AB \rightarrow C, AC \rightarrow B, BC \rightarrow G, E \rightarrow G\}$

Decomposition Properties

- Dependency-preserving: Can we verify all FDs without joins?

- Example: ABE, EG, ABC
  - F = {$AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow G$, $E \rightarrow G$}

Decomposition Properties

- Dependency-preserving: Can we verify all FDs without joins?

- Example: ABE, EG, ABC
  - F = {AB → C, AC → B, BC → G, E → G}

Decomposition Properties

- Dependency-preserving: Can we verify all FDs without joins?

- Example: ABE, EG, ABC
  - $F = \{AB \rightarrow C, AC \rightarrow B, BC \rightarrow G, E \rightarrow G\}$
  - This dependency was not preserved!
  - We can fix this by adding BCG, but this may break BCNF.

Minimal Cover

- G for a set of FDs F
  - Closure of G = closure of F
  - Right hand side of each FD is G is a single attribute
- Implies lossless join and dependency preserving decomposition

- Example: A → B, ABCD → E, EF → GH, ACDF → EG

Minimal Cover

- G for a set of FDs F
  - Closure of G = closure of F
  - Right hand side of each FD is G is a single attribute
- Implies lossless join and dependency preserving decomposition

- Example: $A \rightarrow B$, $ABCD \rightarrow E$, $EF \rightarrow GH$, $ACDF \rightarrow EG$
- $A \rightarrow B$

Minimal Cover

- G for a set of FDs F
    - Closure of G = closure of F
    - Right hand side of each FD is G is a single attribute
- Implies lossless join and dependency preserving decomposition

- Example: A → B, ABCD → E, EF → GH, ACDF → EG
- A → B, ACD → E

Minimal Cover

- G for a set of FDs F
  - Closure of G = closure of F
  - Right hand side of each FD is G is a single attribute
- Implies lossless join and dependency preserving decomposition

- Example: $A \rightarrow B$, $ABCD \rightarrow E$, $EF \rightarrow GH$, $ACDF \rightarrow EG$
- $A \rightarrow B$, $ACD \rightarrow E$, $EF \rightarrow G$, $EF \rightarrow H$

Minimal Cover

- G for a set of FDs F
    - Closure of G = closure of F
    - Right hand side of each FD of G is a single attribute
- Implies lossless join and dependency preserving decomposition

- Example: A → B, ABCD → E, EF → GH, ACDF → EG
- A → B, ACD → E, EF → G, EF → H

Do last third page of worksheet!

Find the set of functional dependencies:

Flights(Flight_no, Date, fRom, To, Plane_id),

ForeignKey(Plane_id)

Planes(Plane_id, tYpe)

Seat(Seat_no, Plane_id, Legroom), ForeignKey(Plane_id)

Find the set of functional dependencies:

Flights(Flight_no, Date, fRom, To, Plane_id),
ForeignKey(Plane_id)

Planes(Plane_id, tYpe)

Seat(Seat_no, Plane_id, Legroom), ForeignKey(Plane_id)

- FD → RTP
- P → Y
- SP → L

Now consider the attribute set R = ABCDE and the FD set $F = \{AB \rightarrow C, A \rightarrow D, D \rightarrow E, AC \rightarrow B\}$. Compute the closure for the following attributes.

- A:
- AB:
- B:
- D:

Now consider the attribute set R = ABCDE and the FD set F = {AB → C, A → D, D → E, AC → B}. Compute the closure for the following attributes.

- A: ADE
- AB: ABCDE
- B: B
- D: DE

Decompose $R = ABCDEFG$ into BCNF, given the FD set:

$F = \{AB \rightarrow CD, C \rightarrow EF, G \rightarrow A, G \rightarrow F, CE \rightarrow F\}$.

Decompose R = ABCDEFG into BCNF, given the FD set:
F = {AB → CD, C → EF, G → A, G → F, CE → F}.

- ABEFG, ABCD

Decompose R = ABCDEFG into BCNF, given the FD set:
F = {AB → CD, C → EF, G → A, G → F, CE → F}.

- ABEFG, ABCD

Decompose R = ABCDEFG into BCNF, given the FD set:
F = {AB → CD, C → EF, G → A, G → F, CE → F}.

- ABEFG, ABCD
- BEFG, ABCD, AG

Decompose R = ABCDEFG into BCNF, given the FD set:
$F = \{AB \rightarrow CD, C \rightarrow EF, G \rightarrow A, G \rightarrow F, CE \rightarrow F\}$.

- ABEFG, ABCD
- BEFG, ABCD, AG
- BEG, FG, ABCD, AG

Decompose R = ABCDEFG into BCNF, given the FD set:
$F = \{AB \rightarrow CD, C \rightarrow EF, G \rightarrow A, G \rightarrow F, CE \rightarrow F\}$.

- ABEFG, ABCD
- BEFG, ABCD, AG
- BEG, FG, ABCD, AG

Does BEG, FG, ABCD, AG preserve dependencies?

$F = \{AB \rightarrow CD, C \rightarrow EF, G \rightarrow A, G \rightarrow F, CE \rightarrow F\}$.

Does BEG, FG, ABCD, AG preserve dependencies?
$F = \{AB \rightarrow CD, C \rightarrow EF, G \rightarrow A, G \rightarrow F, CE \rightarrow F\}$.

- No, $C \rightarrow EF$ and $CE \rightarrow F$ are not preserved.

Give a minimal cover for:

$F = \{AB \rightarrow CD,\ C \rightarrow EF,\ G \rightarrow A,\ G \rightarrow F,\ CE \rightarrow F\}.$

Give a minimal cover for:

$F = \{AB \rightarrow CD, C \rightarrow EF, G \rightarrow A, G \rightarrow F, CE \rightarrow F\}$.

- $AB \rightarrow C$
- $AB \rightarrow D$

Give a minimal cover for:

$F = \{AB \to CD,\ C \to EF,\ G \to A,\ G \to F,\ CE \to F\}$.

- $AB \to C$
- $AB \to D$
- $C \to F$
- $C \to E$

Give a minimal cover for:

$F = \{AB \rightarrow CD, C \rightarrow EF, G \rightarrow A, G \rightarrow F, CE \rightarrow F\}$.

- $AB \rightarrow C$
- $AB \rightarrow D$
- $C \rightarrow F$
- $C \rightarrow E$
- $G \rightarrow A$

Give a minimal cover for:

$F = \{AB \rightarrow CD, C \rightarrow EF, G \rightarrow A, G \rightarrow F, CE \rightarrow F\}$.

- $AB \rightarrow C$
- $AB \rightarrow D$
- $C \rightarrow F$
- $C \rightarrow E$
- $G \rightarrow A$
- $G \rightarrow F$

Give a minimal cover for:

$F = \{AB \rightarrow CD,\ C \rightarrow EF,\ G \rightarrow A,\ G \rightarrow F,\ CE \rightarrow F\}.$

- $AB \rightarrow C$
- $AB \rightarrow D$
- $C \rightarrow F$
- $C \rightarrow E$
- $G \rightarrow A$
- $G \rightarrow F$

Give a minimal cover for:
$F = \{AB \rightarrow CD, C \rightarrow EF, G \rightarrow A, G \rightarrow F, CE \rightarrow F\}$.

- $AB \rightarrow C$
- $AB \rightarrow D$
- $C \rightarrow F$
- $C \rightarrow E$
- $G \rightarrow A$
- $G \rightarrow F$

More lossless practice:

F = {AB -> CDE, BE -> X, A -> E}

Is ABC, BCDEX a lossless decomposition?

More lossless practice:
F = {AB -> CDE, BE -> X, A -> E}
Is ABC, BCDEX a lossless decomposition?

- No, it is lossy. ABC ∩ BCDEX = BC, which is not a superkey of ABC nor BCDEX.

R: ABCDE

Given FD $=\{AE \rightarrow BC, AC \rightarrow D, CD \rightarrow BE, D \rightarrow E\}$

Give three candidate keys.

R: ABCDE

Given FD ={AE → BC, AC → D, CD → BE, D → E}

Give three candidate keys.

- AE, AC and AD are candidate keys, as each of their attribute closures include all attributes and no subset of them is a super key by itself.

R: ABCDE

Given FD ={$AE \rightarrow BC$, $AC \rightarrow D$, $CD \rightarrow BE$, $D \rightarrow E$}

Is R already in BCNF?

R: ABCDE

Given FD ={AE → BC, AC → D, CD → BE, D → E}

Is R already in BCNF?

- No, because both CD → BE and D → E violate BCNF.

R: ABCD

Given FD $=\{A \rightarrow B, B \rightarrow D, C \rightarrow D\}$

Decomposed to AB, CD, AC. Is this lossless?

R: ABCD
Given FD = {A $\rightarrow$ B, B $\rightarrow$ D, C $\rightarrow$ D}
Decomposed to AB, CD, AC. Is this lossless?

- Yes, a lossless decomposition would be: ABC CD which is lossless because C is a key for CD and then a further decomposition of ABC into AB and AC which is lossless because A is a key for AB.