CS 186 Section 6: Advanced SQL, Data Modeling

Vikram Sreekanti

Data Modeling

ER Model

- Entity: A real-world object described by a set of attributes.
  - Entity Set (box): A collection of similar entities. (i.e., all Students.)
    - All entities in a set have the same attributes.
    - Entity sets have a primary key (underlined).
- Relationship: Association between two or more entities.
  - Relationship Set (diamond): collection of similar relationships.
  - Describe the interactions between entities.
  - Relationships can also have attributes.

Constraints

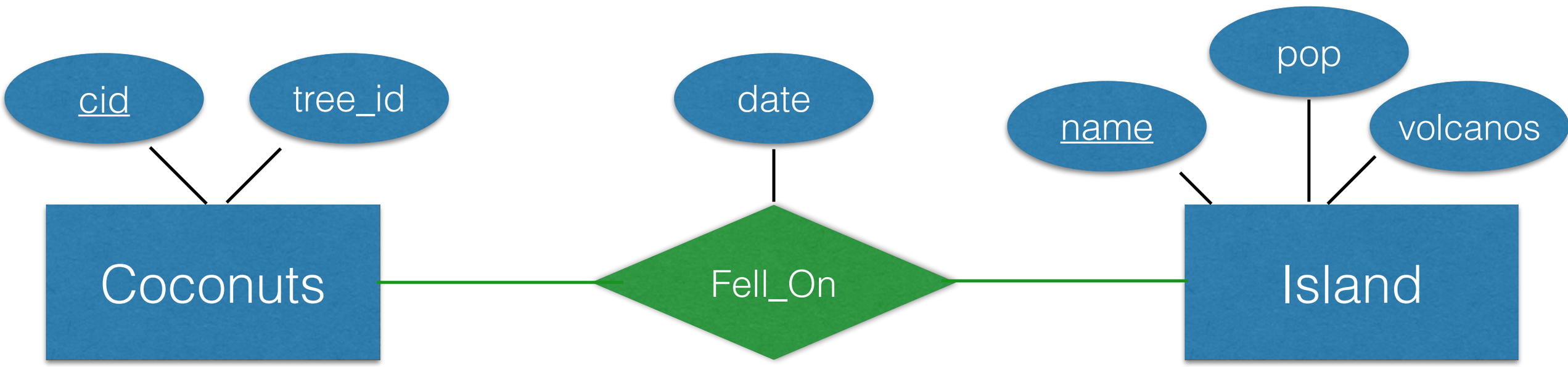
- An arrow denotes a key constraint. This means the key is there at most once.

- A bold line denotes a participation constraint. This means “at least one”.

Constraints Explained Better

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

ER Diagrams: Quick Example

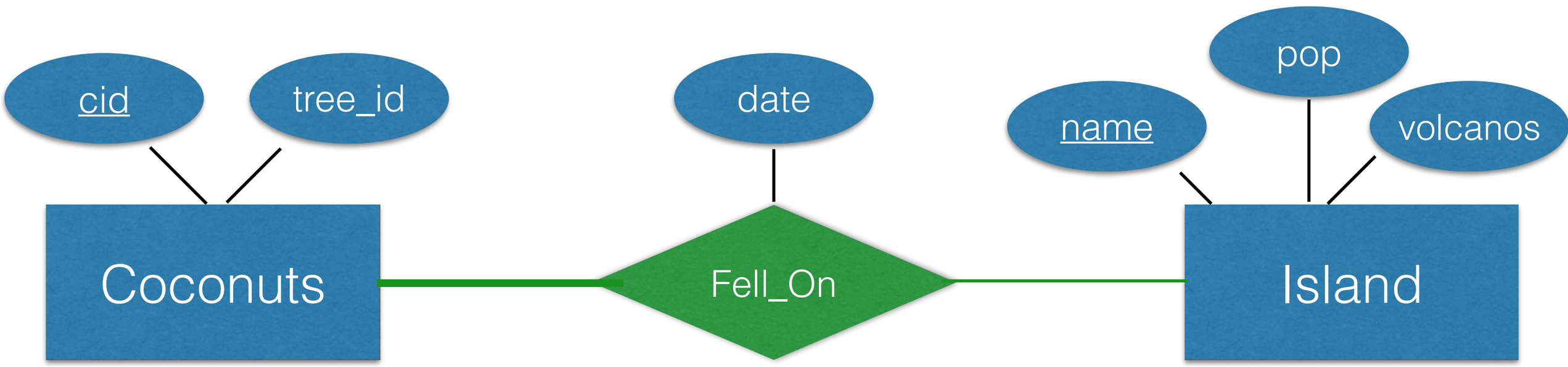


"Coconut participates in the Fell_On relationship 0 or more times."

???

"A coconut can fall on an island zero or more times."

ER Diagrams: Quick Example

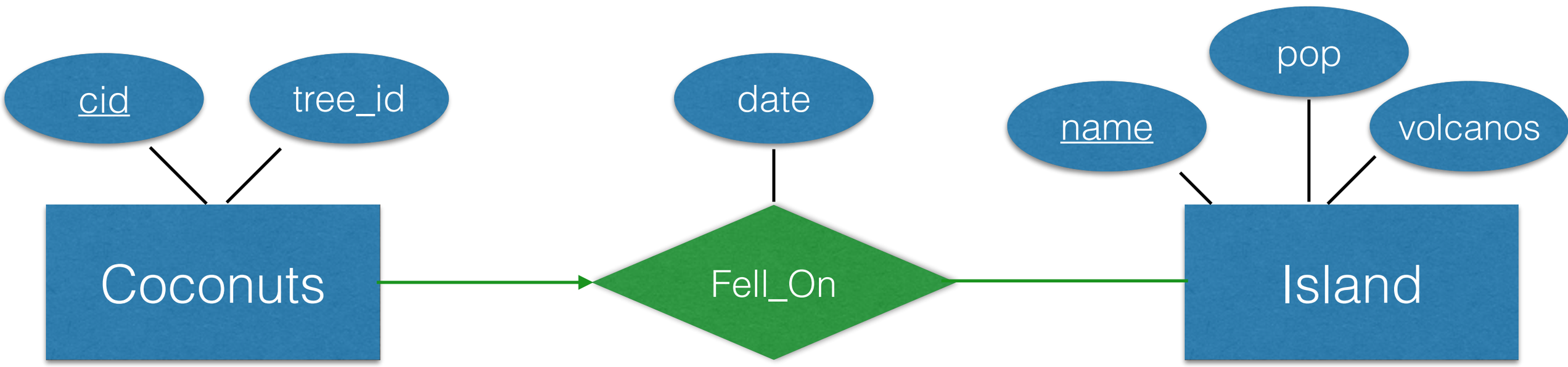


"Coconut participates in the Fell_On relationship 1 or more times."

???

"A coconut must fall on an island at least once."

ER Diagrams: Quick Example

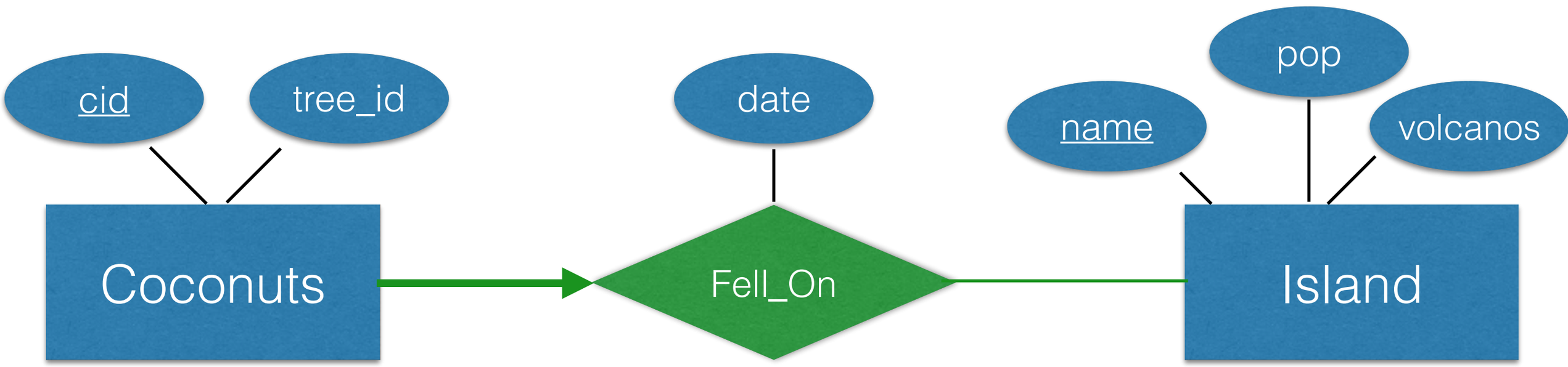


"Coconut participates in the Fell_On relationship 0 or 1 times."

???

"A coconut can fall on an island at least 0 or 1 times."

ER Diagrams: Quick Example

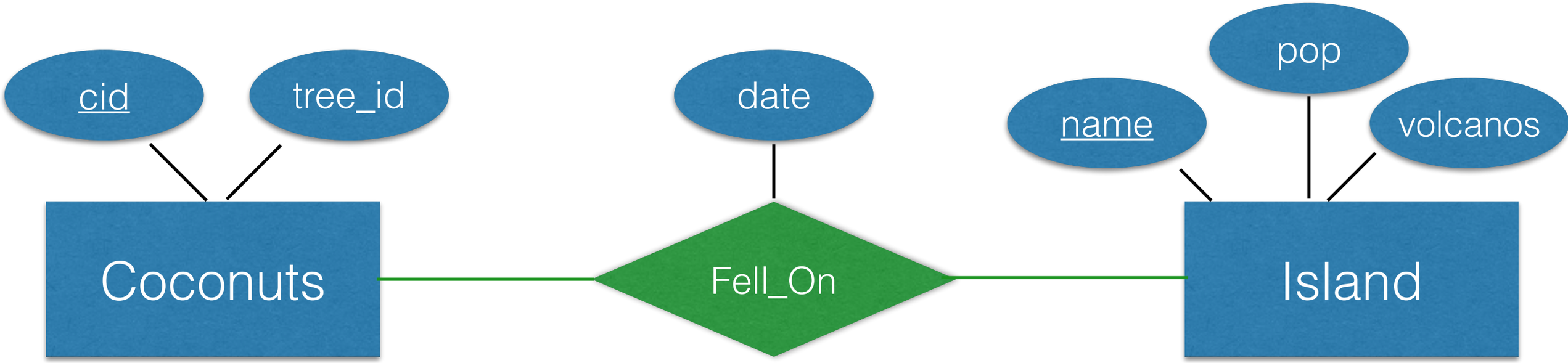


"Coconut participates in the Fell_On relationship1 time."

???

"A coconut falls on an island exactly once."

ER Diagrams: Quick Exercise



What constraint do we want from Coconuts?

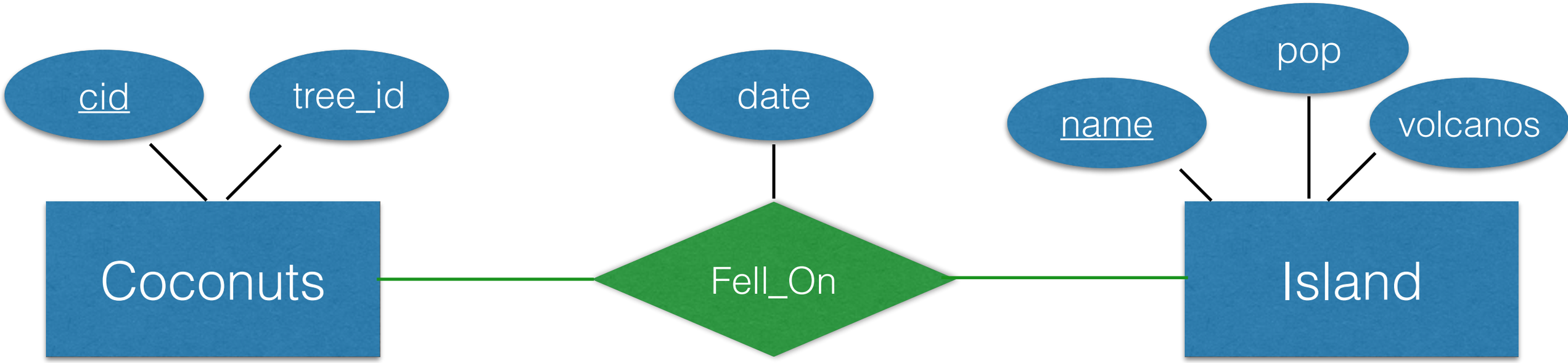
A. A coconut falls 0 or more times.

B. A coconut falls 1 or more times

C. A coconut falls 0 or 1 times.

D. A coconut falls exactly once.

ER Diagrams: Quick Exercise



What constraint do we want from Coconuts?

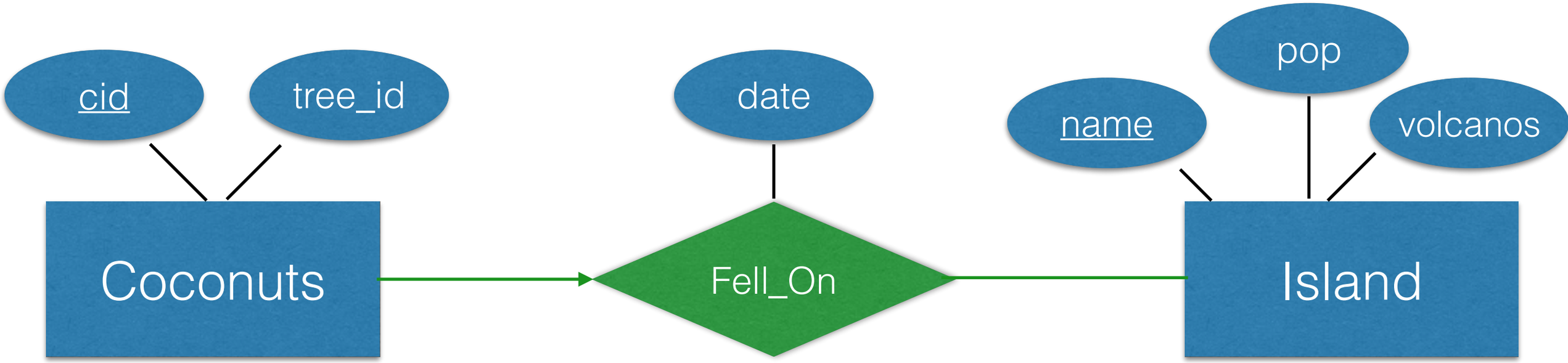
A. A coconut falls 0 or more times.

B. A coconut falls 1 or more times

C. A coconut falls 0 or 1 times.

D. A coconut falls exactly once.

ER Diagrams: Quick Exercise



What constraint do we want from Island?

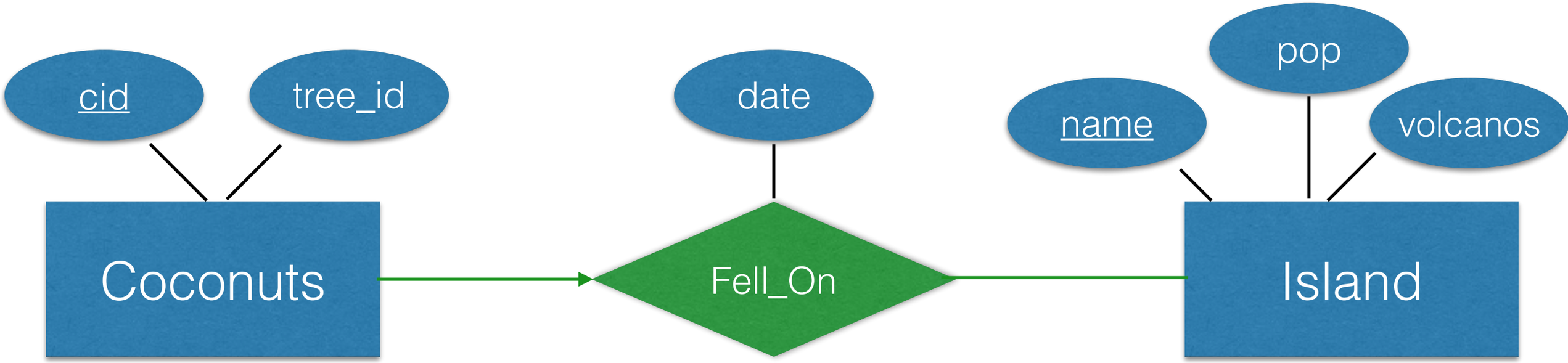
A. A coconut falls 0 or more times.

B. A coconut falls 1 or more times

C. A coconut falls 0 or 1 times.

D. A coconut falls exactly once.

ER Diagrams: Quick Exercise



What constraint do we want from Island?

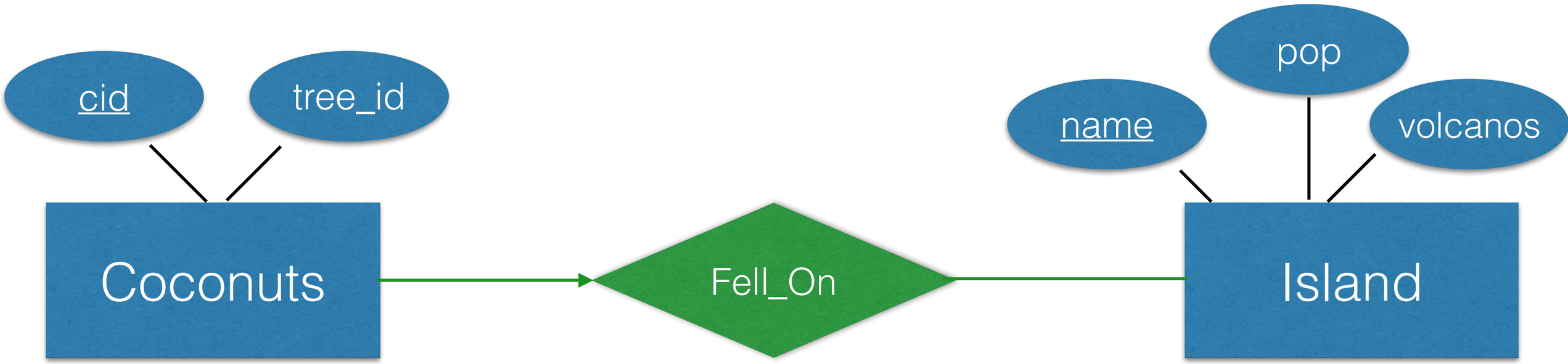
A. A coconut falls 0 or more times.

B. A coconut falls 1 or more times

C. A coconut falls 0 or 1 times.

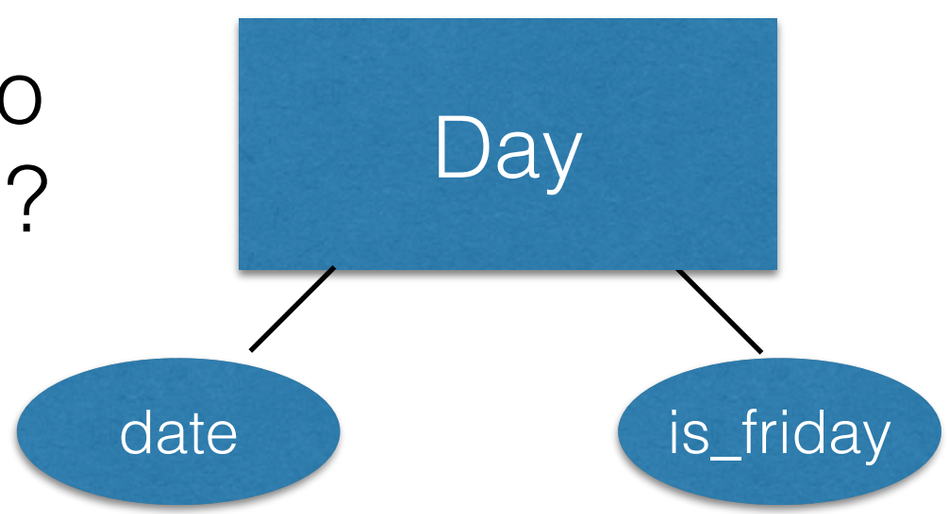
D. A coconut falls exactly once.

ER Diagrams: Ternary Relations

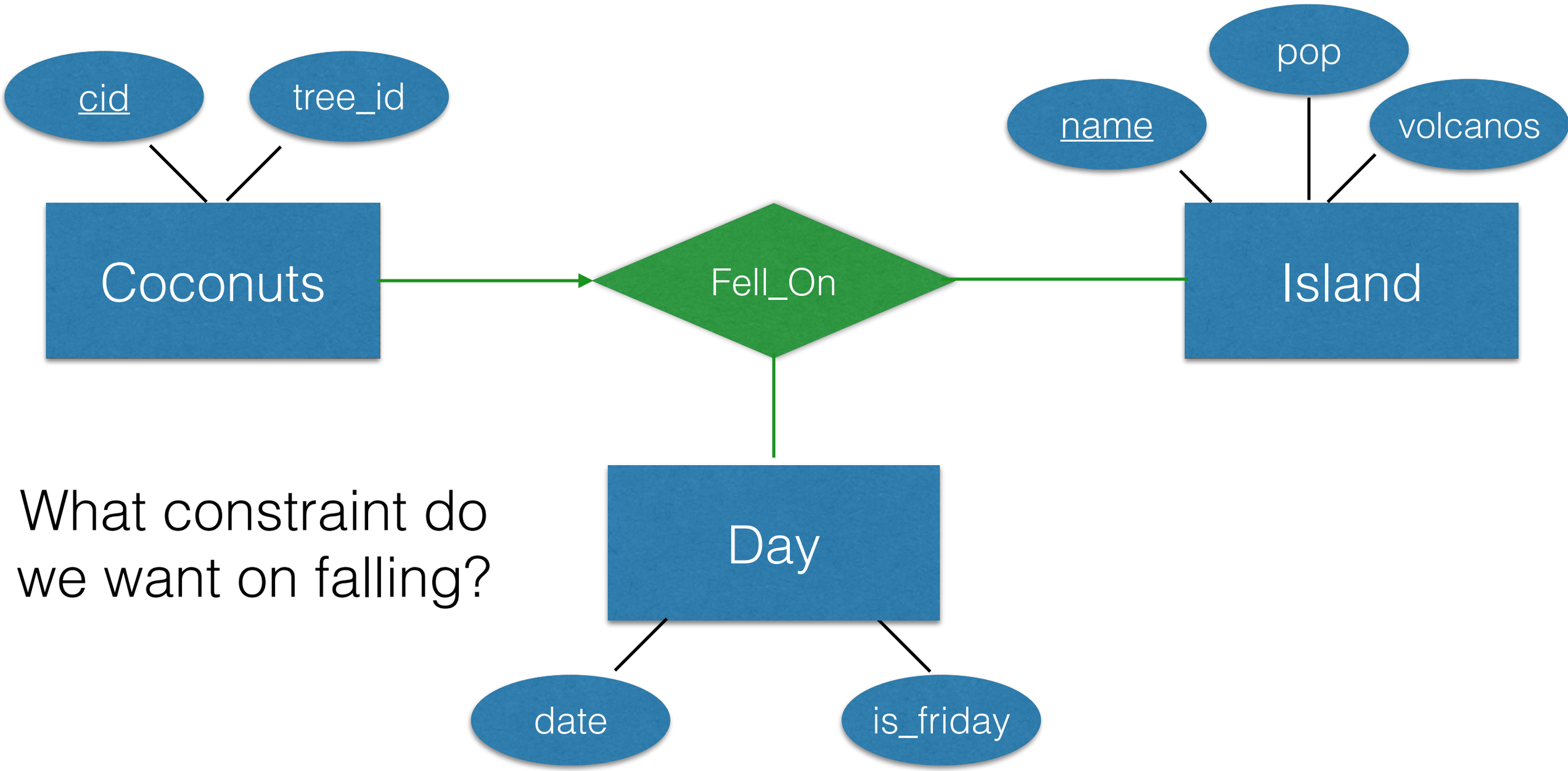


???

What constraint do we want on falling?



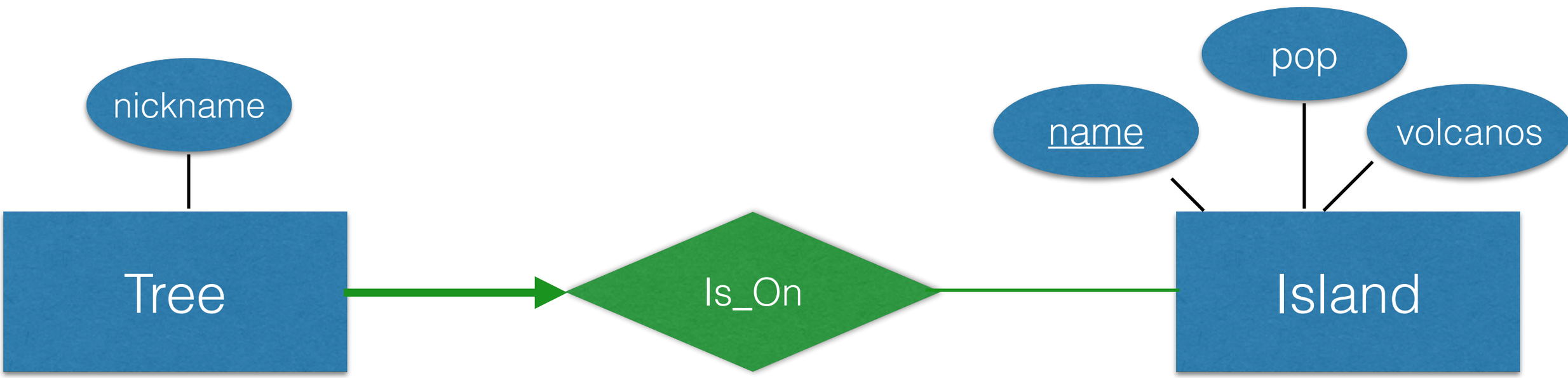
ER Diagrams: Ternary Relations



What constraint do we want on falling?

ER Diagrams: Weak Entities

- An entity that only makes sense in the context of a parent.



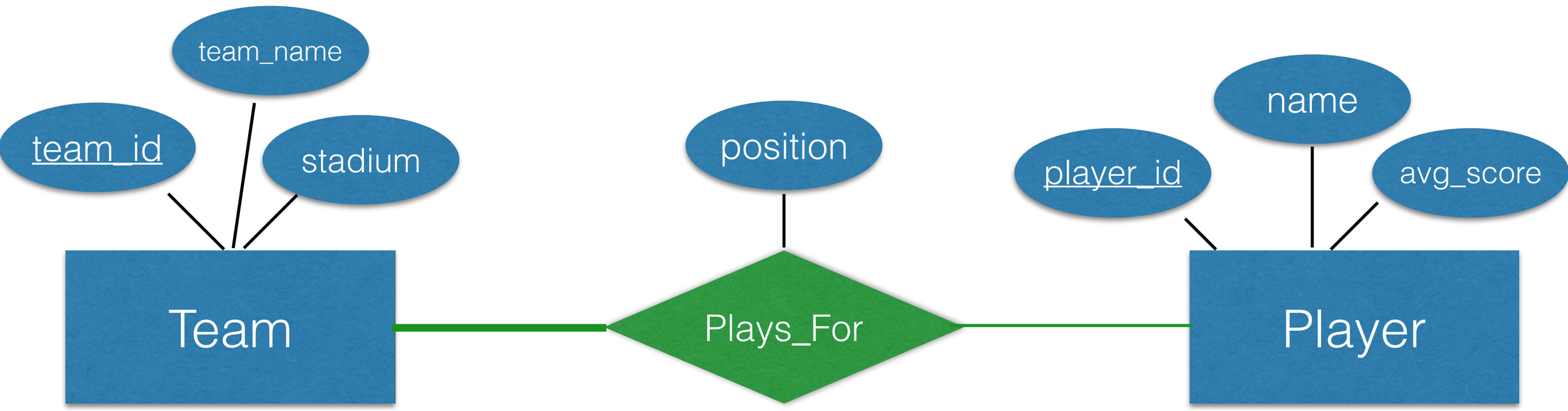
- A tree's key is (Island.name, Tree.nickname).
- There can be two trees with the same name, but not on the same island!
- A tree cannot exist without an island.

ER Diagrams: Exercise

We want to store sports teams and their players in our database. So let's first make an ER diagram! 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 `avg_score`. 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 for more than team, and every team must have at least one player. Draw an ER diagram for this database.

ER Diagram: Exercise

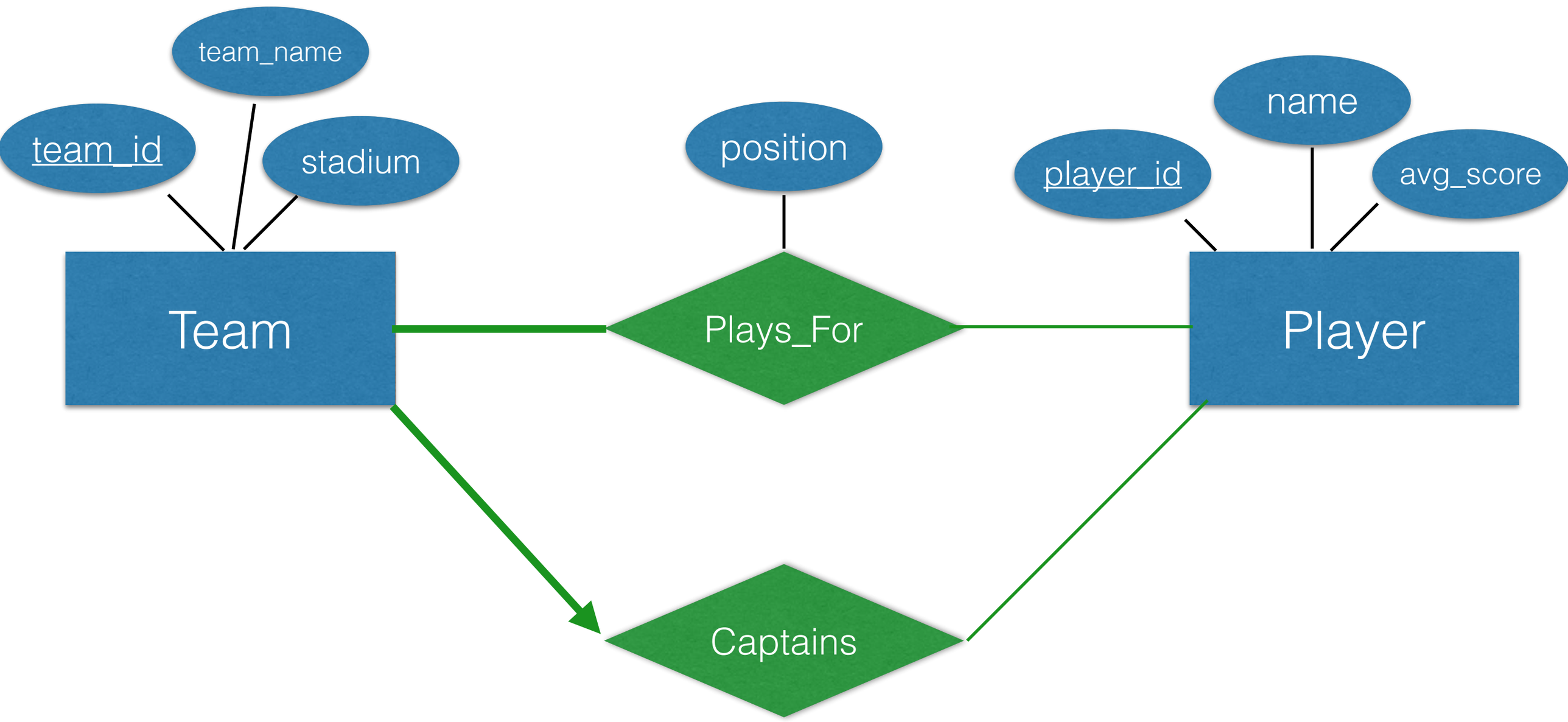


ER Diagrams: Exercise

We want to store sports teams and their players in our database. So let's first make an ER diagram! 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 `avg_score`. Finally our database will contain who `Plays_For` which team and also the `position` that the player plays in.

Now let's say we also want to have a `Captain` for each time. A `Captain` is a `Player` and every team must have one `Captain`. (The constraints from the previous problem apply.) Redraw the ER diagram.

ER Diagram: Exercise



Advanced SQL

I’m sure you guys are more than sick of SQL, so if you don’t want to hear this, feel free to leave.

Full SQL Query

```
SELECT [DISTINCT] <columns>
FROM <tables>
WHERE <predicate>
[GROUP BY <grouping column>
[HAVING <group filter>]
[ORDER BY <order>]
[LIMIT <limit>];
```

Combining Results of Queries

- `UNION` — combination of all the results of the first query and the second query.
- `INTERSECT` — tuples that are in both the first and the second query.
- `EXCEPT` — tuples that are in the first query but not in the second query.

SQL Example

Find **`sid`**s of sailors who've reserved a red or a green boat.

```
SELECT R.sid
FROM Boats B, Reserves R
WHERE R.bid=B.bid AND
      (B.color='red' or
       B.color='green');
```

SQL Example: `UNION`

Find `sid`s of sailors who've reserved a red or a green boat.

```
SELECT R.sid
FROM Boats B, Reserves R
WHERE R.bid=B.bid AND
      B.color='red'
UNION
SELECT R.sid
FROM Boats B, Reserves R
WHERE R.bid=B.bid AND
      B.color='green';
```

SQL Example

Find **`sid`**s of sailors who've reserved a red and a green boat.

```
SELECT R.sid
FROM Boats B, Reserves R
WHERE R.bid=B.bid AND
      (B.color='red' and
       B.color='green');
```

WTF?

SQL Example: `INTERSECT`

Find **`sid`**s of sailors who've reserved a red and a green boat.

```
SELECT R.sid
FROM Boats B, Reserves R
WHERE R.bid=B.bid AND
      B.color='red'
INTERSECT
SELECT R.sid
FROM Boats B, Reserves R
WHERE R.bid=B.bid AND
      B.color='green';
```

SQL Example: Self-Join

Find **`sid`**s of sailors who've reserved a red and a green boat using a self-join!

```
SELECT R.sid
FROM Boats B1, Reserves R1,
     Boats B2, Reserves R2
WHERE R1.sid=R2.sid
        AND R1.bid=B1.bid
        AND R2.bid=B2.bid
        AND(B1.color='red'
            AND B2.color='green');
```

SQL Example: EXCEPT

Find `sid`s of sailors who have not reserved at a boat.

```
SELECT S.sid
FROM Sailors S
EXCEPT
SELECT S.sid
FROM Sailors S, Reserves R
WHERE S.sid=R.sid;
```

SQL Example: `IN`

Find `sid`s of sailors who have reserved boat #102.

```
SELECT S.sid
FROM Sailors S
WHERE S.sid IN
    (SELECT R.sid
    FROM Reserves R
    WHERE R.bid=102;
```

SQL Example: `NOT IN`

Find `sid`s of sailors who have not reserved boat #102.

```
SELECT S.sid
FROM Sailors S
WHERE S.sid NOT IN
    (SELECT R.sid
    FROM Reserves R
    WHERE R.bid=102;
```

SQL Example: Correlated Subqueries

Find **`sid`**s of sailors who have not reserved boat #102.

```
SELECT S.sname
FROM Sailors S
WHERE EXISTS
    (SELECT *
    FROM Reserves R
    WHERE R.bid=102 AND S.sid=R.sid;
```

SQL Joins

- `INNER`: Return only rows that match.
- `LEFT`: Return rows that match plus all unmatched rows from the table on the left side of the join clause.
- `RIGHT`: Return rows that match plus all unmatched rows from the table on the right side of the join clause.
- `OUTER`: Return rows that match plus all unmatched rows from the table on both sides of the join clause.

SQL: Left Outer Join

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
95	Bob	3	63.5

sid	bid	day
22	101	10/10/2014
95	103	11/12/2014

s.sid	s.sname	r.bid
22	Dustin	101
31	Lubber	103
95	Bob	

SQL: Right Outer Join

sid	bid	day
22	101	10/10/2014
95	103	11/12/2014

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

r.sid	b.bid	b.name
22	101	Interlake
	102	Interlake
95	103	Clipper
	104	Marine

SQL: Full Outer Join

sid	bid	day
22	101	10/10/2014
95	103	11/12/2014

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Note that this is the same as the ROJ because `bid` is a foreign key in `Reserves`.

r.sid	b.bid	b.name
22	101	Interlake
	102	Interlake
95	103	Clipper
	104	Marine

SQL Exercises

```
Songs (song_id, song_name, album_id, weeks_in_top_40)
Artists(artist_id, artist_name, first_year_active)
Albums (album_id, album_name, artist_id, year_released, genre)
```

Find all the `album_ids` and `names` for every artist who became active 2000 or later. If an artist does not have any albums, you should still include the artist in your output.

```sql
SELECT Ar.artist_id, Ar.artist_name,
       Al.album_id, Al.album_name
FROM Artists AR
     LEFT OUTER JOIN Albums Al
ON Ar.artist_id = Al.artist_id
WHERE Ar.first_year_active >= 2000
```

SQL Exercises

```
Songs (song_id, song_name, album_id, weeks_in_top_40)
Artists(artist_id, artist_name, first_year_active)
Albums (album_id, album_name, artist_id, year_released, genre)
```

Find the `song_ids` and `names` for each song released that the artist of the album was active.

```sql
SELECT S.song_id, S.song_name
       Ar.artist_id, Ar.artist_name
FROM Song S INNER JOIN Albums Al
ON S.album_id=Al.album_id
INNER JOIN Song S
ON Al.artist_id=Ar.artist_id
AND Al.year_released=Ar.first_year_active
```