Relational Database Design

Module 6: Basic Normalization (Part 1)

Hugo Kornelis
hugo@perFact.info

pluralsight
hardcore developer training

Outline

- **Normalization**
- **Functional dependencies**
- **Basic normal forms**
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
- **Finding functional dependencies**

Why normalize?

- **Non-atomic values**
  - Complex code required
  - Performance impact

- **Redundancy**
  - Same fact stored multiple times
  - Storage space wasted
  - Performance impact
  - Possibility of conflicting data
  - Derived facts: special case of redundancy

Redundancy: Misconceptions

- Repeating a <u>value</u> is not redundant

FriendName	HairColor
Jack	Red
Jane	Red

Duplicates?

FriendName	HairColorID
Jack	3
Jane	3

HairColorID	Color
1	Black
2	Blond e
3	Red
4	Brown

Jack has red hair.
Jane has red hair.

No duplicates!

Redundancy: Misconceptions

- Repeating a <u>value</u> is not redundant

FriendName	HairColor
Jack	Red
Jane	Red

No duplicates!

FriendName	HairColorID
Jack	3
Jane	3

HairColorID	Color
1	Black
2	Blond e
3	Red
4	Brown

Redundancy: Misconceptions

- **Not all redundancy is bad**
  - Redundancy can help performance
  - Derived data may be impossible to derive again later
  - Derived data may be too expensive to derive every time

- **Uncontrolled redundancy *IS* bad!**
  - Mark duplicated data as such
  - Mark derived data as such
  - Prevent inconsistent data

Why normalize?

- **Non-atomic values**
  - Complex code required
  - Performance impact
- **Redundancy**
  - Same fact stored multiple times
  - Storage space wasted
  - Performance impact
  - Possibility of conflicting data
  - Derived facts: special case of redundancy
- **Modification anomalies**
  - Design causes modifications to have unwanted side effects

Modification anomalies

Tournament	Player Name	Player Phone
2012 Christmas Tournament	Dave	801-555-0124
2013 Midsummer Tournament	Dave	801-555-0124
…	…	…
Tournament		
…	…	…

How to normalize?

- **Steps**
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Elementary Key Normal Form (EKNF)
  - Boyce-Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)
  - Fifth Normal Form (5NF)
  - Domain/Key Normal Form (DKNF)
  - Sixth Normal Form (6NF)
- **Normal forms apply to table**
- **Normal form of database = lowest normal form of all its tables**

When to normalize?

- **Most common**
  - Convert Entity Relationship model to relational tables
  - Normalize relational tables
  - **Disadvantage:** Changes must be ported back to ER model

- **Alternative**
  - Normalize Entity Relationship model
  - Convert normalized ER model to relational tables
  - **Disadvantage:** Normalization is a bit more complicated
    - Normalize "every object that will eventually become a table"
    - For IDEF1X:
      - every entity type
      - every many-to-many relationship

Functional dependencies

- **Mathematical <u>function</u> ($f(x) \rightarrow y$)**
  - For every value of *x*:
    - Exactly one value of *y* can be computed, or
    - The value of *y* is not defined

- **<u>Functional</u> dependency (attribute *A* → attribute *B*)**
  - For every value of *A*:
    - Exactly one value of *B* can be determined, or
    - There is no value of *B*
  - "Attribute B is functionally dependent on attribute A".
  - "Attribute B functionally depends on attribute A".
  - "Attribute B depends on attribute A".

Determinant

Dependent attribute

Dave

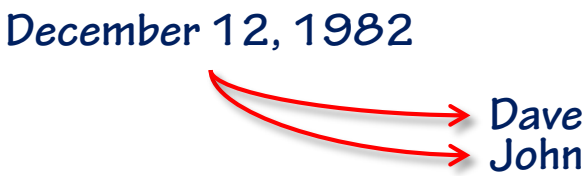
December 12, 1982

Megan

No birthdate on file

Properties of functional dependencies

- **Can be mutual**
  - Most are not!



Properties of functional dependencies

- **Can be mutual**
  - Most are not!

- **Can be on a combination of two or more attributes**

Dave

December 12, 1982

June 5, 1959

Properties of functional dependencies

- **Can be mutual**
  - Most are not!
- **Can be on a combination of two or more attributes**
- **Depend on "Universe of Discourse"**
  - Beware when making assumptions!
- **If *X* depends on *Y*, it also depends on each superset of *Y***

Dave, C league

December 12, 1982

Properties of functional dependencies

- **Can be mutual**
  - Most are not!
- **Can be on a combination of two or more attributes**
- **Depend on "Universe of Discourse"**
  - Beware when making assumptions!
- **If *X* depends on *Y*, it also depends on each superset of *Y***
  - Dependency on two or more attributes can sometimes be reduced!

Dave → December 12, 1982

Properties of functional dependencies

- **Can be mutual**
  - Most are not!
- **Can be on a combination of two or more attributes**
- **Depend on "Universe of Discourse"**
  - Beware when making assumptions!
- **If *X* depends on *Y*, it also depends on each superset of *Y***
  - Dependency on two or more attributes can sometimes be reduced!
  - *Full* dependency: Functional dependency that cannot be reduced
- **Every attribute depends on itself (and on each superset of itself)**

Dave → Dave

Properties of functional dependencies

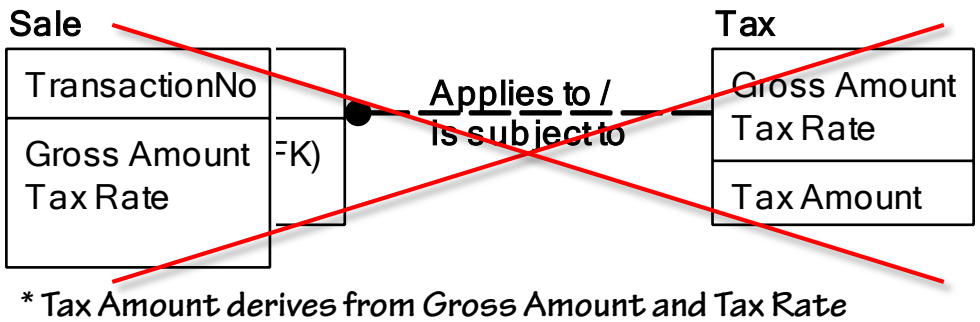
- **Can be mutual**
  - Most are not!
- **Can be on a combination of two or more attributes**
- **Depend on "Universe of Discourse"**
  - Beware when making assumptions!
- **If *X* depends on *Y*, it also depends on each superset of *Y***
  - Dependency on two or more attributes can sometimes be reduced!
  - *Full* dependency: Functional dependency that cannot be reduced
- **Every attribute depends on itself (and on each superset of itself)**
  - *Trivial* dependency

Functional dependencies and normalization

- **Normalization uses functional dependencies that are:**
  - Non-trivial
  - Full
- **How to find all functional dependencies?**
  - *Most* are obvious
  - But what about the rest?
- **Guaranteed method**
  - Combines finding functional dependencies with normalization
  - Tedious and time-consuming; use only when needed

Functional dependencies and derived attributes

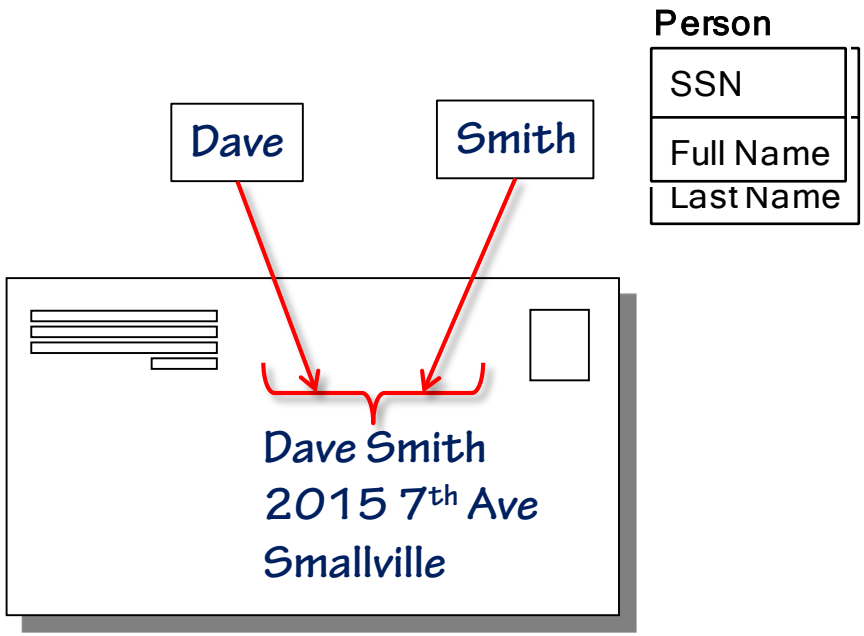
- Derived attributes may show up as functionally dependent
- These dependencies are different from "normal" dependencies

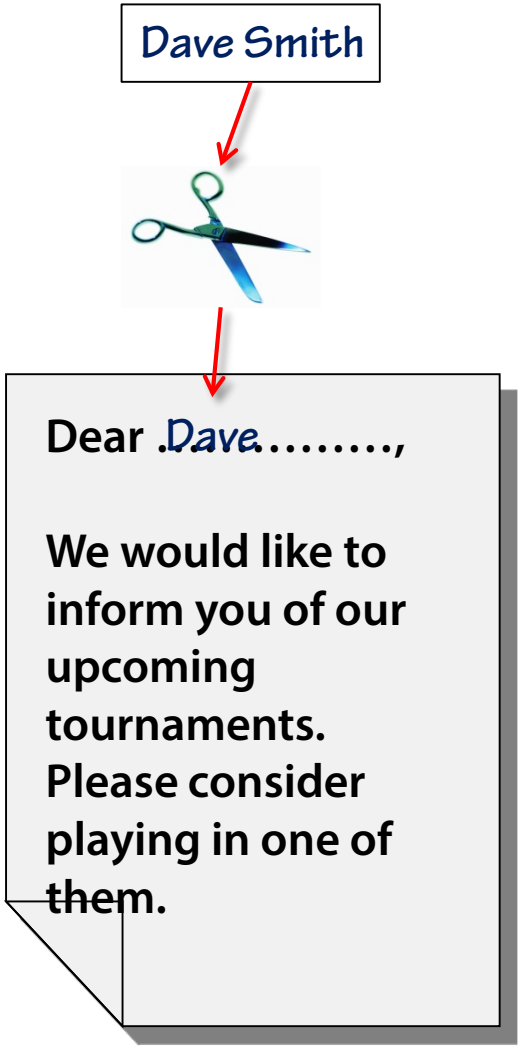


{Gross Amount, Tax Rate} → Tax Amount

First Normal Form

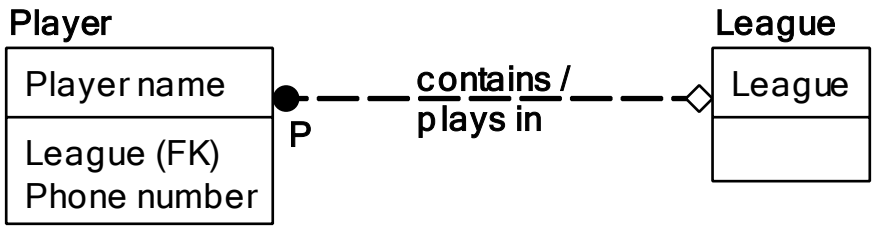
- **Requirements for First Normal Form (1NF):**
  - Table must have a key
    - (that all attributes depend on)
  - Every column stores *atomic* values
    - Not composite
      - **DEPENDS ON HOW DATA IS USED!**





First Normal Form

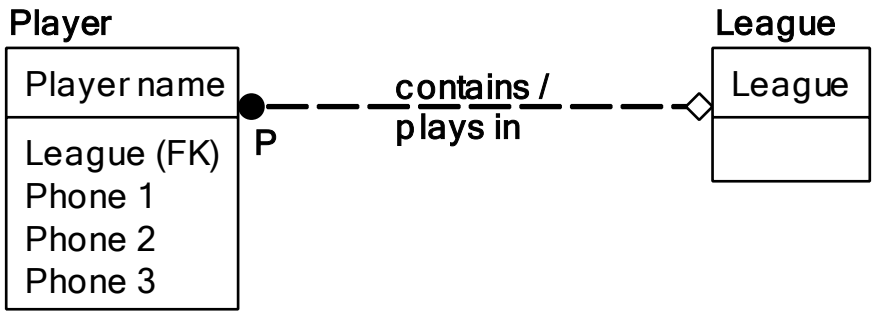
- **Requirements for First Normal Form (1NF):**
  - Table must have a key
    - (that all attributes depend on)
  - Every column stores *atomic* values
    - Not composite
      - **DEPENDS ON HOW DATA IS USED!**
    - Not a repeating group
      - **DEPENDS ON HOW DATA IS USED!**



Player name	League	Phone number
Dave	C	801-555-0124, 801-555-9505, 801-555-3009
Mary	B	801-555-0125

First Normal Form

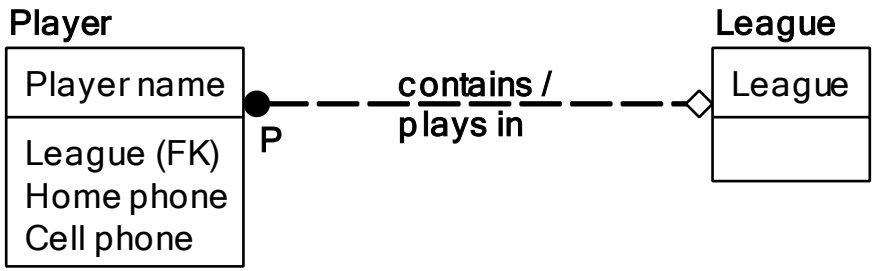
- **Requirements for First Normal Form (1NF):**
  - Table must have a key
    - (that all attributes depend on)
  - Every column stores *atomic* values
    - Not composite
      - **DEPENDS ON HOW DATA IS USED!**
    - Not a repeating group
      - **DEPENDS ON HOW DATA IS USED!**



Player name	League	Phone 1	Phone 2	Phone 3
Dave	C	801-555-0124	801-555-9505	801-555-3009
Mary	B	801-555-0125		

First Normal Form

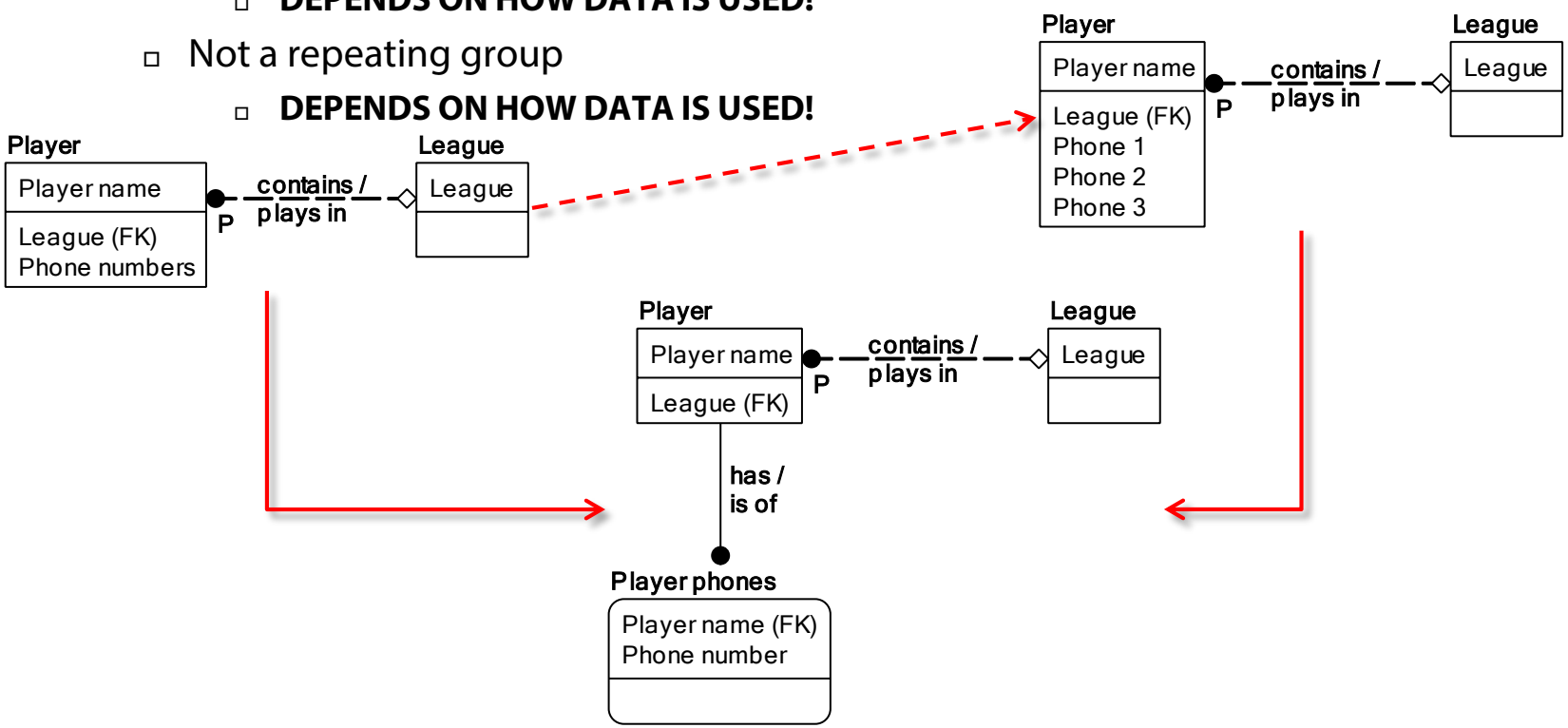
- **Requirements for First Normal Form (1NF):**
  - Table must have a key
    - (that all attributes depend on)
  - Every column stores *atomic* values
    - Not composite
      - **DEPENDS ON HOW DATA IS USED!**
    - Not a repeating group
      - **DEPENDS ON HOW DATA IS USED!**



Player name	League	Home phone	Cell phone
Dave	C	801-555-0124	801-555-9505
Mary	B	801-555-0125	

First Normal Form

- **Requirements for First Normal Form (1NF):**
  - Table must have a key
    - (that all attributes depend on)
  - Every column stores *atomic* values
    - Not composite
      - **DEPENDS ON HOW DATA IS USED!**
    - Not a repeating group
      - **DEPENDS ON HOW DATA IS USED!**



First Normal Form

- **Requirements for First Normal Form (1NF):**
  - Table must have a key
    - (that all attributes depend on)
  - Every column stores *atomic* values
    - Not composite
      - **DEPENDS ON HOW DATA IS USED!**
    - Not a repeating group
      - **DEPENDS ON HOW DATA IS USED!**

- **Candidate keys**
  - Attribute that can be used to identify individual rows
  - Combination of attributes that can be used to identify individual rows
    - Has to be minimal (not containing all attributes of another candidate key)
    - Sometimes also used for candidate keys not (yet) known to be minimal
  - Do not choose primary key yet!
    - (IDEF1X unfortunately forces a –preliminary!- choice)

Finding candidate keys

- **Use functional dependencies**
  - An attribute is candidate key if all other attributes depend on it
  - A combination of attributes is candidate key is:
    - It is not a superset of another candidate key
    - All other attributes depend on the combination or a subset of the combination

Example

A
B (AK1.1) C (AK1.2) D E F

A → B
A → C
A → D
A → E
A → F

B → E

D → F

{B, C} → A
{B, C} → D
{B, C} → F

Finding candidate keys

- **Use functional dependencies**
  - An attribute is candidate key if all other attributes depend on it
  - A combination of attributes is candidate key is:
    - It is not a superset of another candidate key
    - All other attributes depend on the combination or a subset of the combination
  - Multiple candidate keys are always mutually dependent

Example

A
B (AK1.1) C (AK1.2) D E F

$A \rightarrow B$
$A \rightarrow C$
$A \rightarrow D$
$A \rightarrow E$
$A \rightarrow F$

$B \rightarrow E$

$D \rightarrow F$

$\{B, C\} \rightarrow A$
$\{B, C\} \rightarrow D$
$\{B, C\} \rightarrow F$

Finding functional dependencies, part 1

- **Find functional dependencies for every object that becomes a table**
  - IDEF1X:
    - Entity types
    - Many to many relationships
- **Based on table with column for each attribute**
  - IDEF1X:
    - Entity types: key and non-key attributes (including foreign key attributes)
    - Many to many relationships: key attributes of connected entity types
- **First candidate key**
  - IDEF1X
    - Entity types: key attributes
    - Many to many relationships: key attributes of connected entity types

Finding functional dependencies, part 1

- **Mark all candidate keys**
  - Underline attribute names
  - Arrow over attribute names

Tablename (Column1, Column2, Column3, Column4, Column5)

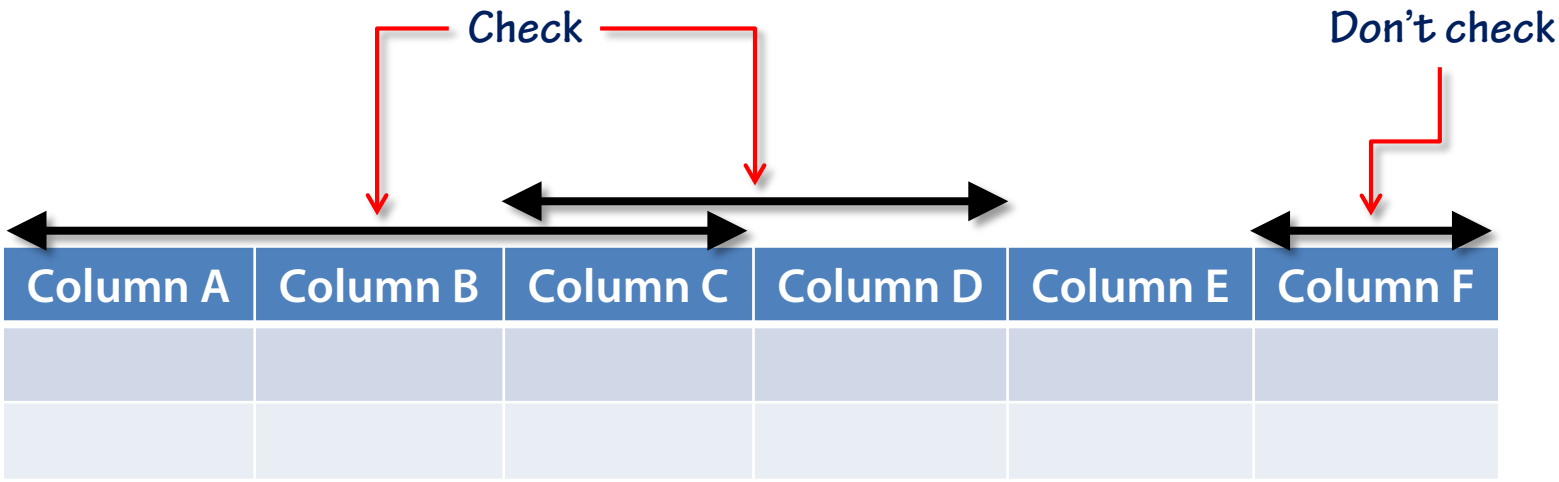
Column1	Column2	Column3	Column4	Column5

Finding functional dependencies, part 1

- **Mark all candidate keys**
  - Underline attribute names
  - Arrow over attribute names
- **All other columns depend on candidate key**
  - These dependencies are implied
  - No need to spell them out

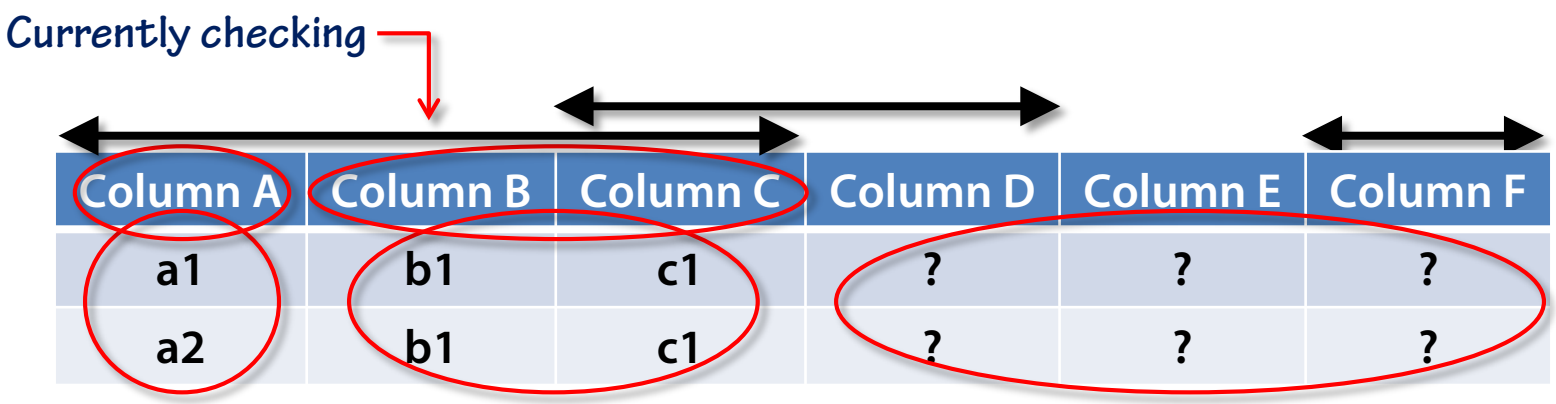
Finding functional dependencies, part 1

- **Verify that candidate keys are minimal**
  - Single-column key is always minimal
  - Procedure executed for each *composite* (multi-column) key



Finding functional dependencies, part 1

- **Verify that candidate keys are minimal**
  - Single-column key is always minimal
  - Procedure executed for each *composite* (multi-column) key
  - Pattern to be populated:
    - One key column different
    - Other key columns identical
    - Satisfy known business rules
    - Add extra rows as needed



Finding functional dependencies, part 1

- **Verify that candidate keys are minimal**
  - Find existing example that includes the desired pattern
  - Create new example that includes the desired pattern
    - Represent in familiar notation for subject matter expert
    - Check if valid
      - If invalid: because of pattern, or for another reason?

Currently checking

Column A	Column B	Column C	Column D	Column E	Column F
a1	b1	c1	?	?	?
a2	b1	c1	?	?	?

Finding functional dependencies, part 1

- **Verify that candidate keys are minimal**
  - Impossible to create **valid** example with the required pattern?
    - Column with difference depends on column(s) with no difference
      - Can be a normal functional dependency, or a derivation rule
    - Implies: all columns in table depend on column(s) with no difference
    - Implies: candidate key was not minimal
      - Replace with candidate key on column(s) with no difference

Currently checking

Column A	Column B	Column C	Column D	Column E	Column F
a1	b1	c1	?	?	?
a2	b1	c1	?	?	?

Finding functional dependencies, part 1

- **Verify that candidate keys are minimal**
  - Always test all columns of original candidate key

Currently checking

Column A	Column B	Column C	Column D	Column E	Column F
a1	b1	c1	?	?	?
a2	b1	c1	?	?	?

Finding functional dependencies, part 1

- **Verify that candidate keys are minimal**
  - Always test all columns of original candidate key

Currently checking

Column A	Column B	Column C	Column D	Column E	Column F
a1	b1	c1	?	?	?
a1	b2	c1	?	?	?

Finding functional dependencies, part 1

- **Verify that candidate keys are minimal**
  - Always test all columns of original candidate key
  - Use a fixed order to ensure you don't miss one
  - No new functional dependencies found?
    - Original candidate key was minimal!
  - New functional dependencies found?
    - Original candidate key replaced by new key(s)
    - New key(s) should be tested too!
- **Verify *all* composite candidate keys**

Currently checking

Test this one next!

Column A	Column B	Column C	Column D	Column E	Column F
a1	b1	c1	?	?	?
a1	b1	c2	?	?	?

Finding functional dependencies, part 1

- **Artificial entity type**
  - Original candidate key often not minimal
- **Normal entity type**
  - Original candidate key usually minimal
- **Many-to-many relationship**
  - Original candidate key almost always minimal
  - When not, revisit design

Finding functional dependencies, part 2

- **Look for missing candidate keys**
  - All attributes depend on candidate key
    - No two rows with same value for candidate key attributes
  - Valid population with same values in combination of attributes?
    - These attributes and their combinations are not candidate keys
  - No such population possible?
    - This attribute or combination of attributes is a candidate key

Finding functional dependencies, part 2

- **Look for missing single-column candidate keys**
  - Pattern to be populated:
    - One non-key column identical
    - Other columns irrelevant
    - Satisfy known business rules
      - Difference required in at least one column of every candidate key
    - Add extra rows as needed

Column A	Column B	Column C	Column D	Column E	Column F
?	?	?	?	e1	?
?	?	?	?	e1	?

Finding functional dependencies, part 2

- **Look for missing single-column candidate keys**
  - **Valid** example with required pattern found or created?
    - Tested column is not a candidate key
  - Impossible to create **valid** example with the required pattern?
    - Tested column ***is*** a candidate key

Column A	Column B	Column C	Column D	Column E	Column F
?	?	?	?	e1	?
?	?	?	?	e1	?

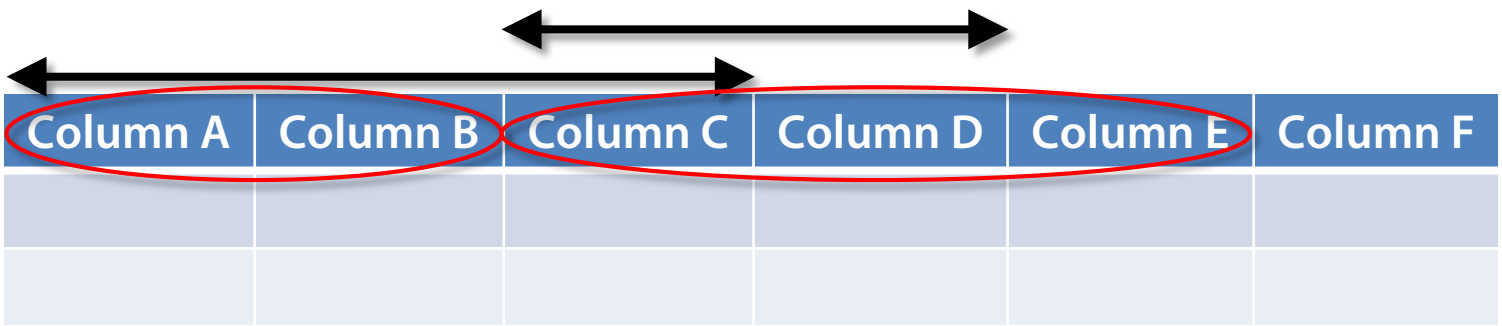
Finding functional dependencies, part 2

- **Look for missing single-column candidate keys**
  - Always test all non-key attributes
  - Use a fixed order to ensure you don't miss one

Column A	Column B	Column C	Column D	Column E	Column F
?	?	?	?	?	f1
?	?	?	?	?	f1

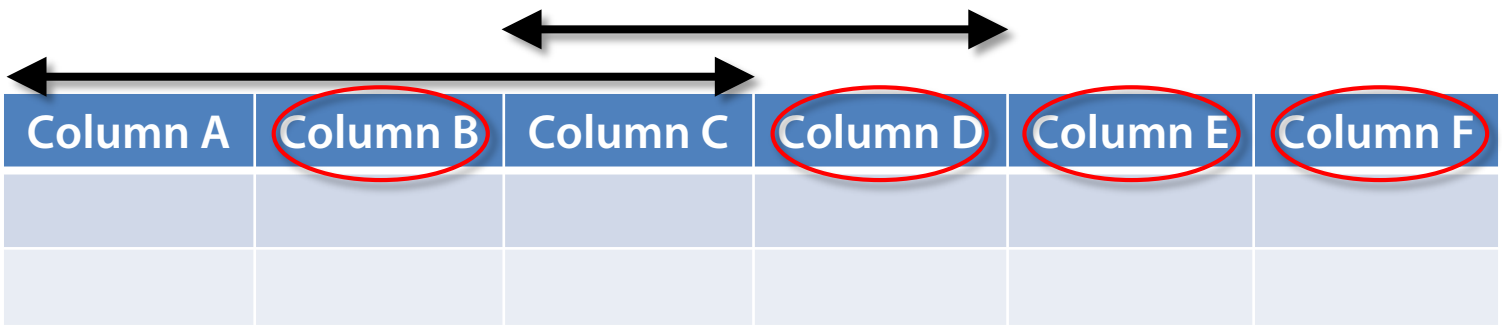
Finding functional dependencies, part 2

- **Look for missing multi-column candidate keys**
  - Check possible composite candidate keys
    - Skip supersets of smaller candidate keys
    - Skip subsets of larger candidate keys



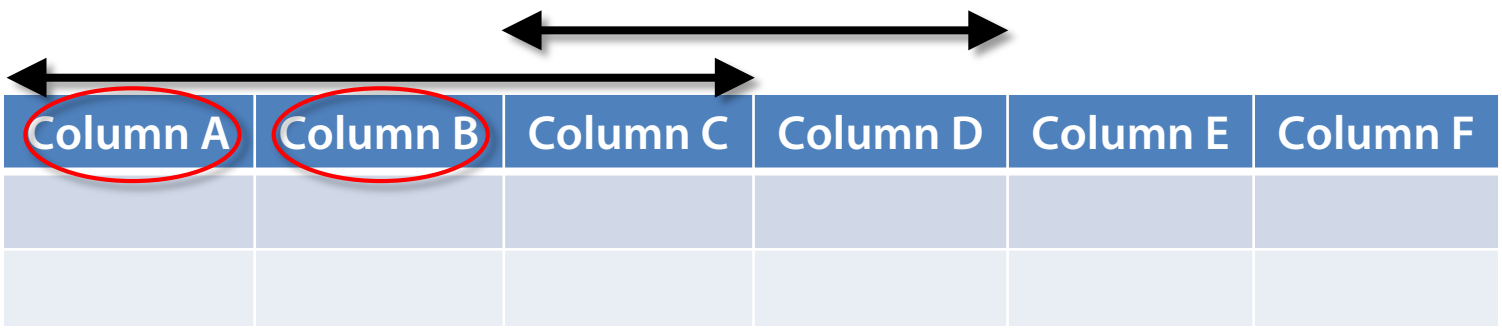
Finding functional dependencies, part 2

- **Look for missing multi-column candidate keys**
  - Check possible composite candidate keys
    - Skip supersets of smaller candidate keys
    - Skip subsets of larger candidate keys
    - Two column possibilities:
      - Two non-key columns
      - Non-key + part of multi-column key
      - Part of multi-column key + part of *other* multi-column key



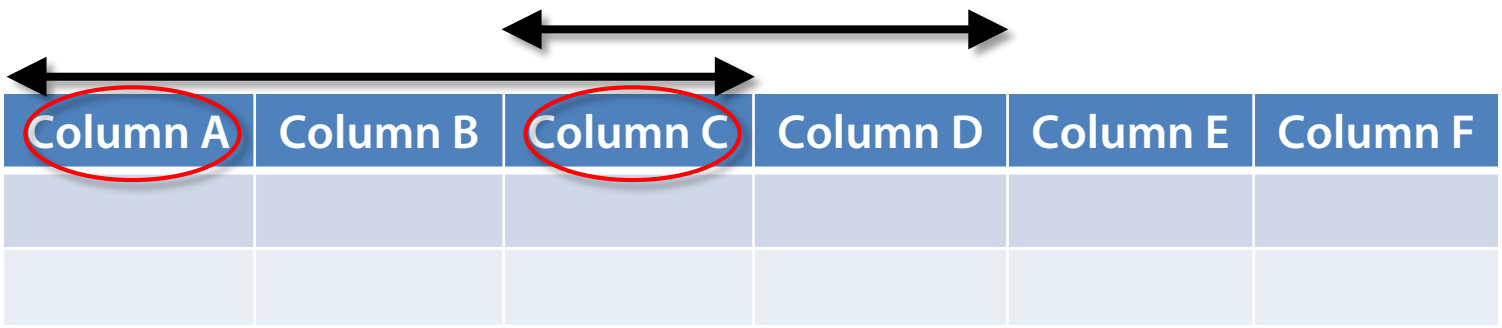
Finding functional dependencies, part 2

- **Look for missing multi-column candidate keys**
  - Check possible composite candidate keys
    - Work systematically, verify all combinations
      - Testing needed?
      - If yes: Test



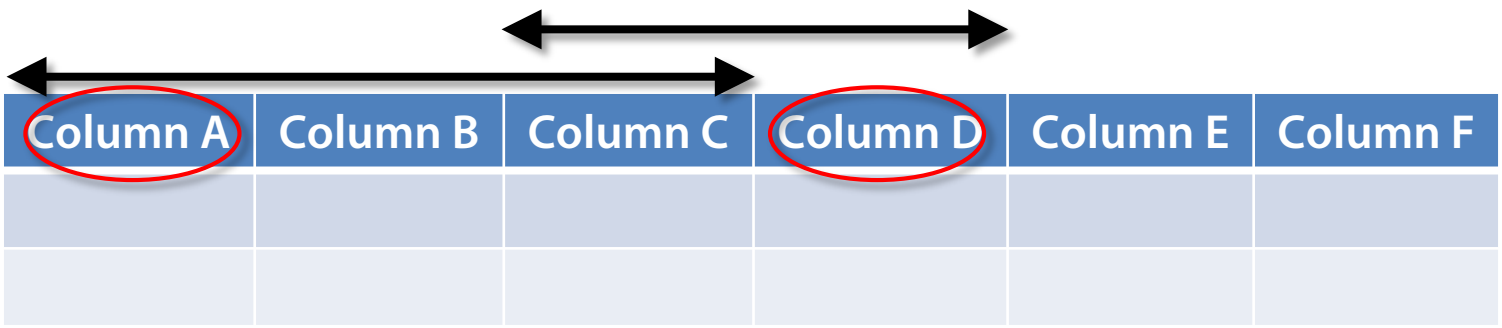
Finding functional dependencies, part 2

- **Look for missing multi-column candidate keys**
  - Check possible composite candidate keys
    - Work systematically, verify all combinations
      - Testing needed?
      - If yes: Test



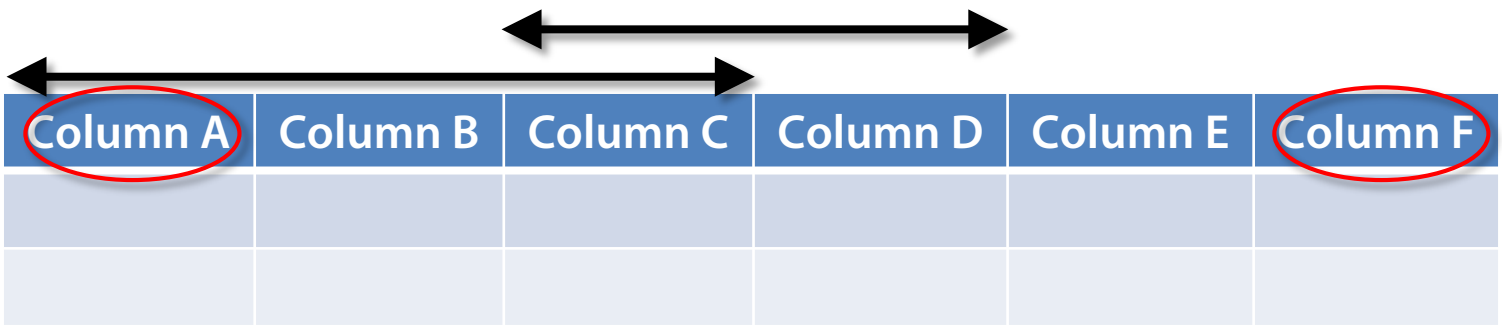
Finding functional dependencies, part 2

- **Look for missing multi-column candidate keys**
  - Check possible composite candidate keys
    - Work systematically, verify all combinations
      - Testing needed?
      - If yes: Test



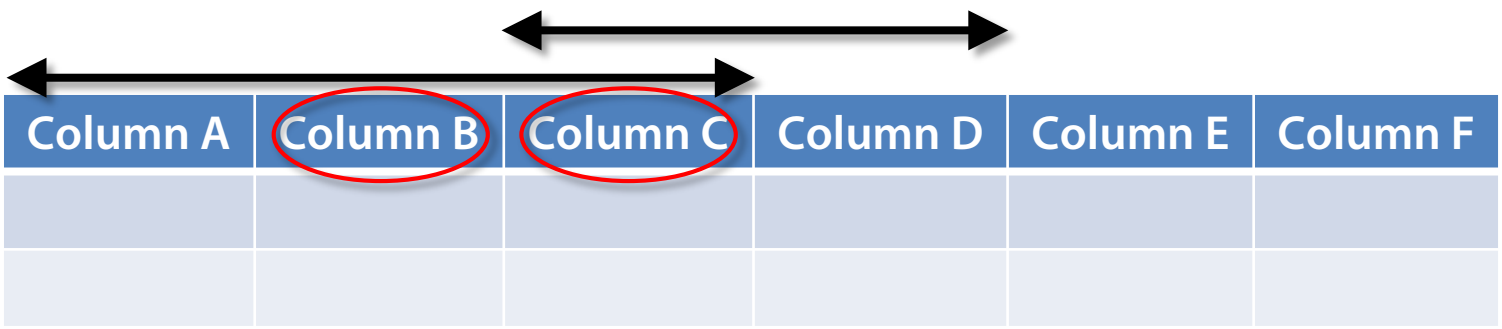
Finding functional dependencies, part 2

- **Look for missing multi-column candidate keys**
  - Check possible composite candidate keys
    - Work systematically, verify all combinations
      - Testing needed?
      - If yes: Test



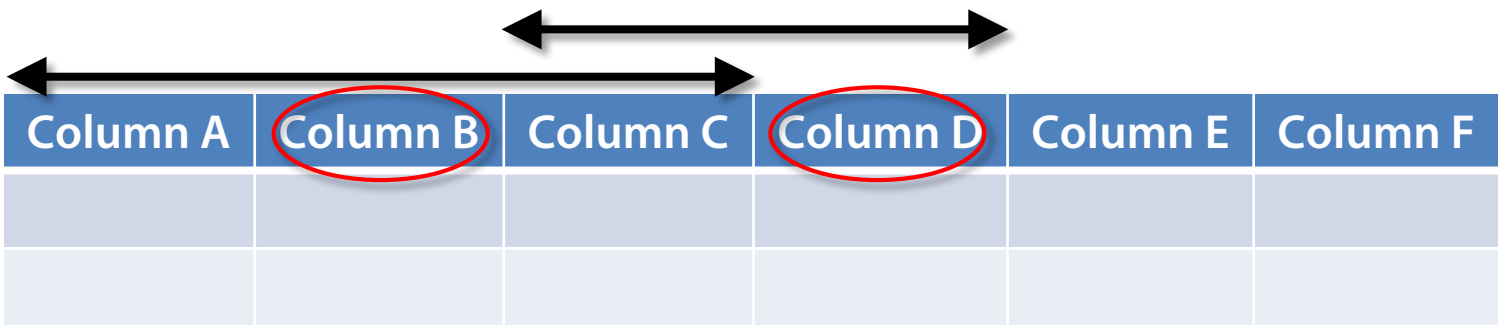
Finding functional dependencies, part 2

- **Look for missing multi-column candidate keys**
  - Check possible composite candidate keys
    - Work systematically, verify all combinations
      - Testing needed?
      - If yes: Test



Finding functional dependencies, part 2

- **Look for missing multi-column candidate keys**
  - Check possible composite candidate keys
    - Work systematically, verify all combinations
      - Testing needed?
      - If yes: Test



Finding functional dependencies, part 2

- **Look for missing multi-column candidate keys**
  - Check possible composite candidate keys
    - Test for combination: like test for single attribute
    - Pattern to be populated:
      - All columns of tested combination identical
      - Other columns irrelevant
      - Satisfy known business rules
    - Impossible to create **valid** example with the required pattern?
      - Tested combination of columns is a candidate key
        - Sure to be minimal

Column A	Column B	Column C	Column D	Column E	Column F
?	?	?	d1	e1	?
?	?	?	d1	e1	?

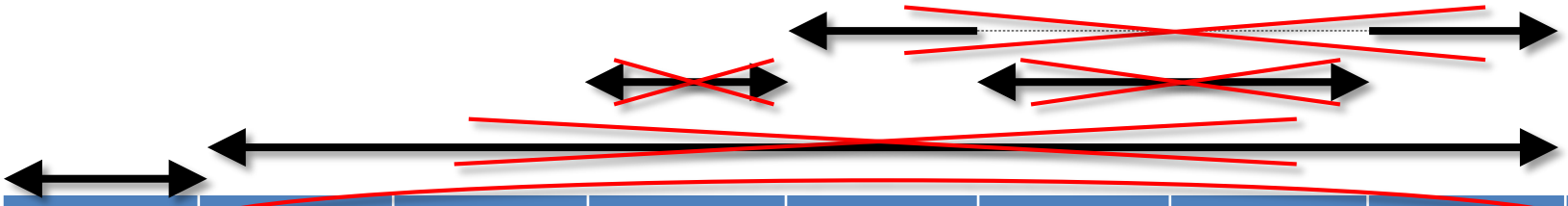
Finding functional dependencies, part 2

- **Look for missing multi-column candidate keys**
  - Check possible composite candidate keys
    - Test larger combinations
      - Skip supersets of smaller candidate keys
      - Skip subsets of larger candidate keys
    - Add keys when found
      - Sure to be minimal
    - Can be a lot of work!!!

Column A	Column B	Column C	Column D	Column E	Column F

Finding functional dependencies, part 2

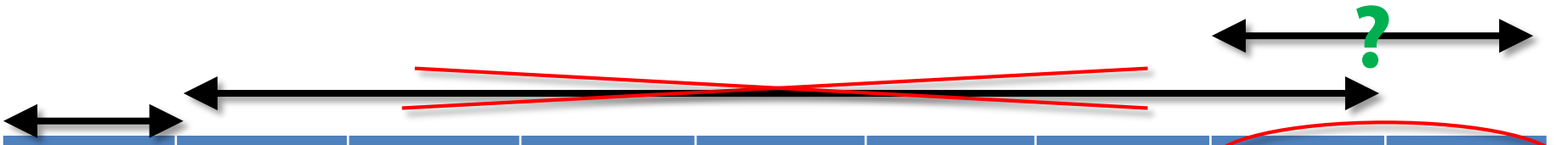
- **Look for missing candidate keys**
  - Alternate procedure
  - Test all suspected non-key columns at once
  - Valid example found or created?
    - Tested combination is not a candidate key
    - Subsets of this combination can't be a candidate key



Column A	Column B	Column C	Column D	Column E	Column F	Column G	Column H	Column I
?	b1	c1	d1	e1	f1	g1	h1	?
?	b1	c1	d1	e1	f1	g1	h1	?

Finding functional dependencies, part 2

- **Look for missing candidate keys**
  - Alternate procedure
  - Test all suspected non-key columns at once
  - Valid example found or created?
    - Tested combination is not a candidate key
    - Subsets of this combination can't be a candidate key
    - Other combinations still have to be tested!



Column A	Column B	Column C	Column D	Column E	Column F	Column G	Column H	Column I
?	?	?	?	?	?	?	h1	I1
?	?	?	?	?	?	?	h1	i1

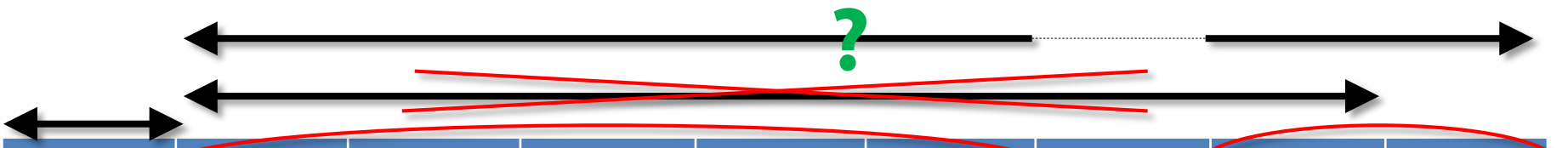
Finding functional dependencies, part 2

- **Look for missing candidate keys**
  - Alternate procedure
  - Test all suspected non-key columns at once
  - Valid example found or created?
    - Tested combination is not a candidate key
    - Subsets of this combination can't be a candidate key
    - Other combinations still have to be tested!

Column A	Column B	Column C	Column D	Column E	Column F	Column G	Column H	Column I
?	?	c1	?	e1	f1	?	?	i1
?	?	c1	?	e1	f1	?	?	i1

Finding functional dependencies, part 2

- **Look for missing candidate keys**
  - Alternate procedure
  - Test all suspected non-key columns at once
  - Valid example found or created?
    - Tested combination is not a candidate key
    - Subsets of this combination can't be a candidate key
    - Other combinations still have to be tested!
      - (Can again be done in bulk)



Column A	Column B	Column C	Column D	Column E	Column F	Column G	Column H	Column I
?	b1	c1	d1	e1	f1	?	h1	i1
?	b1	c1	d1	e1	f1	?	h1	i1

Finding functional dependencies, part 2

- **Look for missing candidate keys**
  - Alternate procedure
  - Test all suspected non-key columns at once
  - Impossible to create valid example with this pattern?
    - Tested combination ***might*** be a candidate key
      - … but probably not minimal!
    - Assume key, then test (using previous procedure) if it's minimal
      - Can be a lot of work!!!

Column A	Column B	Column C	Column D	Column E	Column F	Column G	Column H	Column I
?	b1	c1	d1	e1	f1	g1	h1	?
?	b1	c1	d1	e1	f1	g1	h1	?

Finding functional dependencies, part 2

- **Look for missing candidate keys**
  - "Blend" procedure
  - Test patterns in existing valid examples
    - Two rows with identical value in non-key columns?
      - That combination of columns is **NOT** a candidate key
      - Subsets of that combination are **NOT** candidate keys
  - Follow "normal" procedure
    - Skip attributes and combinations already ruled out
      - Subset of candidate key
      - Superset of candidate key
      - Subset of combination known not to be a candidate key

Demo: verify that attributes are atomic

On October 3, 2012, Katie and Jim played a match in league C. This match ended with 2 frames won by Katie and 1 frame won by Jim.

Competition Score Form

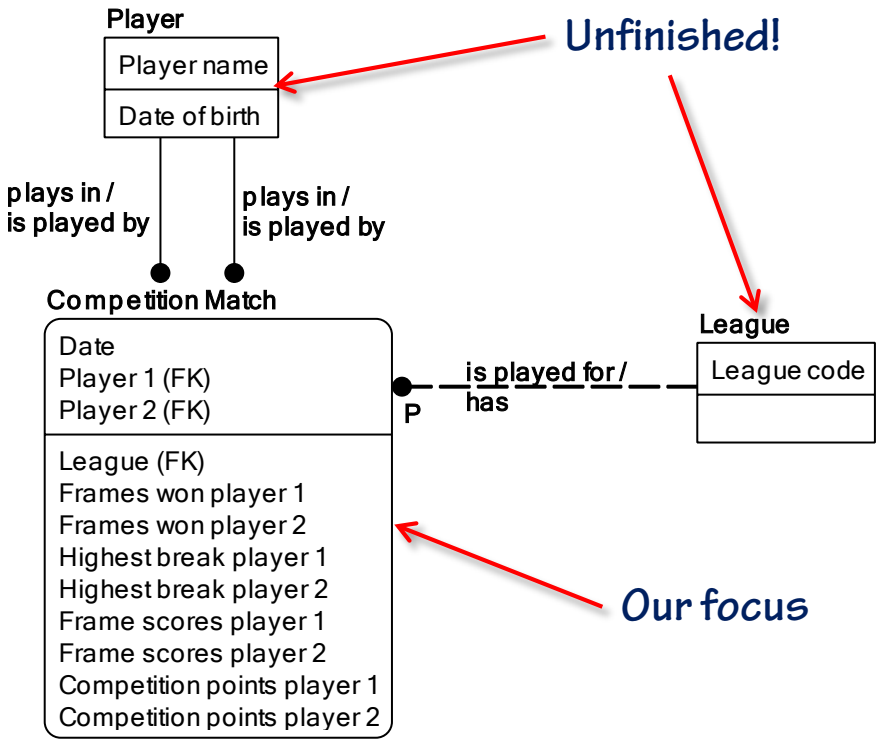
League: C

Date: October 3, 2012

Players	Katie	Jim
Frames won	2	1
HB	40	12
	Frame results	
Frame 1	51	37
Frame 2	30	63
Frame 3	62	18

Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2	First frame		Second frame		Third frame	
3-10-2012	Katie	Jim	C	2	1	40	12	12	10	51	37	30	63	62	18
3-10-2012	Dave	Hugo	C	3	0	27	8	20	0	52	21	40	38	61	25

Demo: verify that attributes are atomic



Competition Score Form

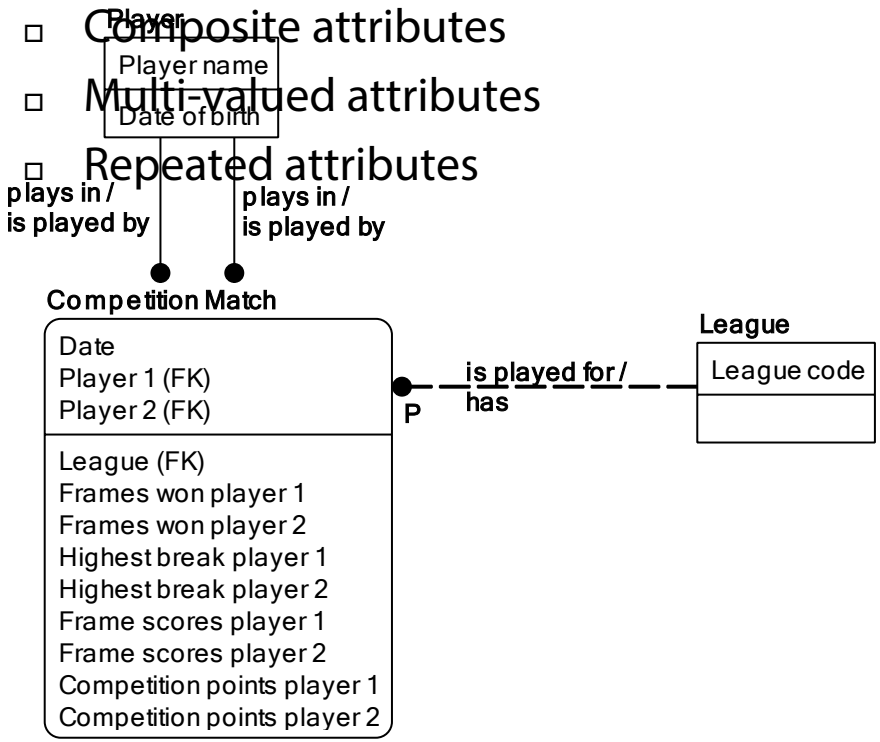
League: *C*

Date: *October 3, 2012*

Players	*Katie*	*Jim*
Frames won	*2*	*1*
HB	*40*	*12*
	Frame results	
Frame 1	*51*	*37*
Frame 2	*30*	*63*
Frame 3	*62*	*18*

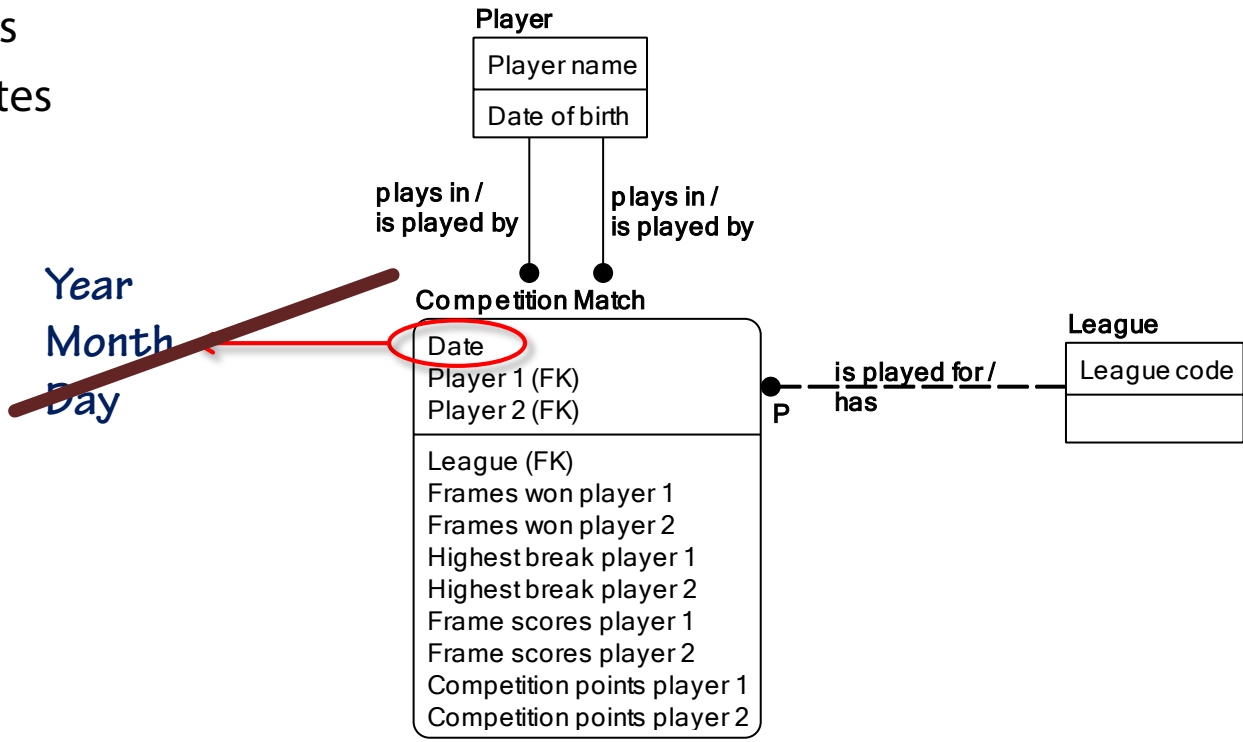
Demo: verify that attributes are atomic

- **Are attributes atomic?**
  - Composite attributes
  - Multi-valued attributes
  - Repeated attributes



Demo: verify that attributes are atomic

- **Are attributes atomic?**
  - Composite attributes
  - Multi-valued attributes
  - Repeated attributes



Demo: verify that attributes are atomic

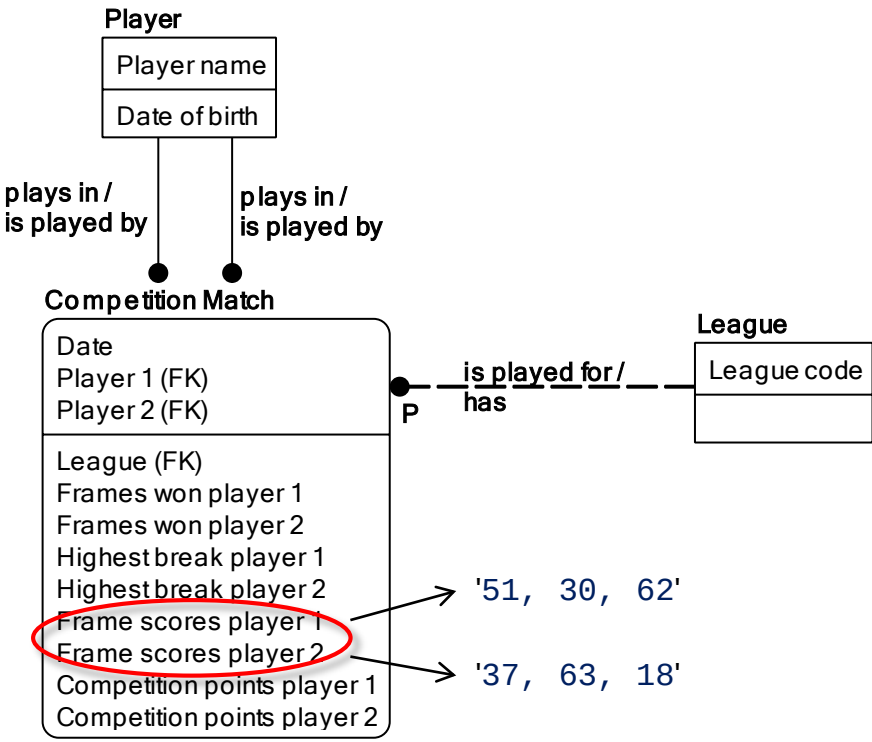
- **Are attributes atomic?**
  - Composite attributes
  - Multi-valued attributes
  - Repeated attributes

Competition Score Form

League: *C*

Date: *October 3, 2012*

Players	*Katie*	*Jim*
Frames won	*2*	*1*
HB	*40*	*12*
	Frame results	
Frame 1	*51*	*37*
Frame 2	*30*	*63*
Frame 3	*62*	*18*



Demo: verify that attributes are atomic

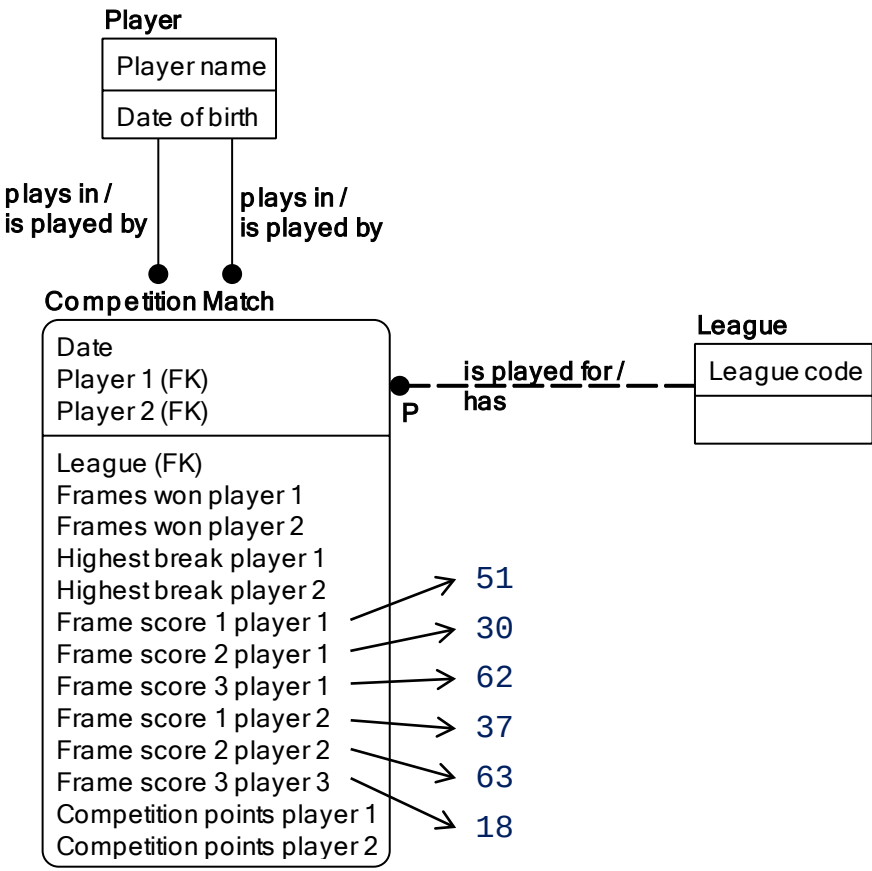
- **Are attributes atomic?**
  - Composite attributes
  - Multi-valued attributes
  - Repeated attributes

Competition Score Form

League: *C*

Date: *October 3, 2012*

Players	*Katie*	*Jim*
Frames won	*2*	*1*
HB	*40*	*12*
	Frame results	
Frame 1	*51*	*37*
Frame 2	*30*	*63*
Frame 3	*62*	*18*



Demo: verify that attributes are atomic

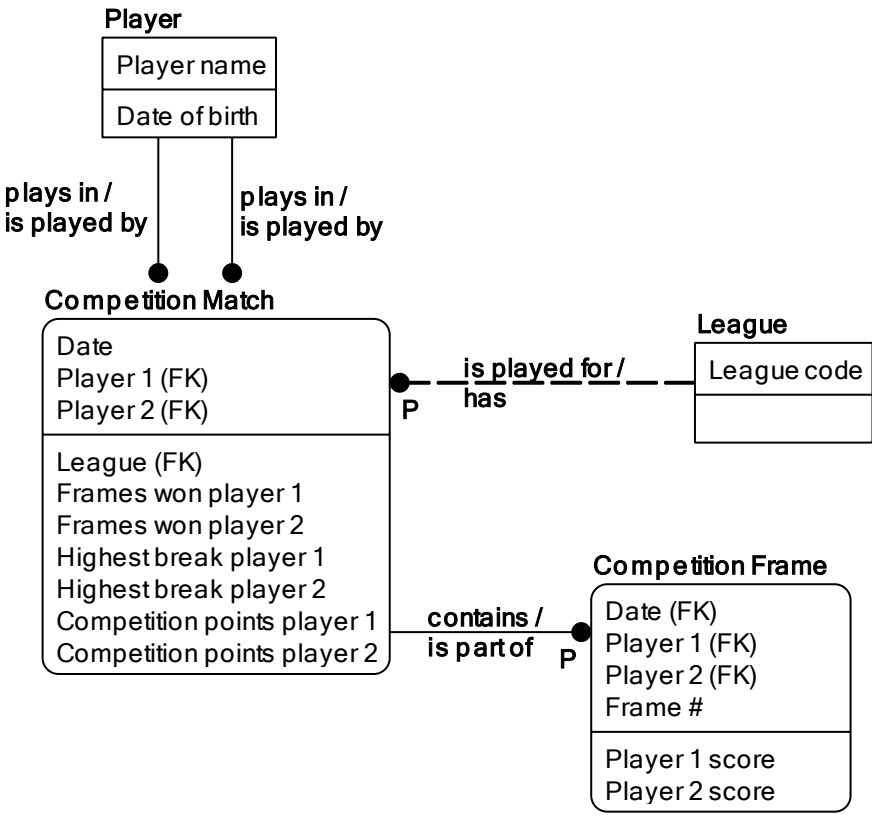
- **Are attributes atomic?**
  - Composite attributes
  - Multi-valued attributes
  - Repeated attributes

Competition Score Form

League: C

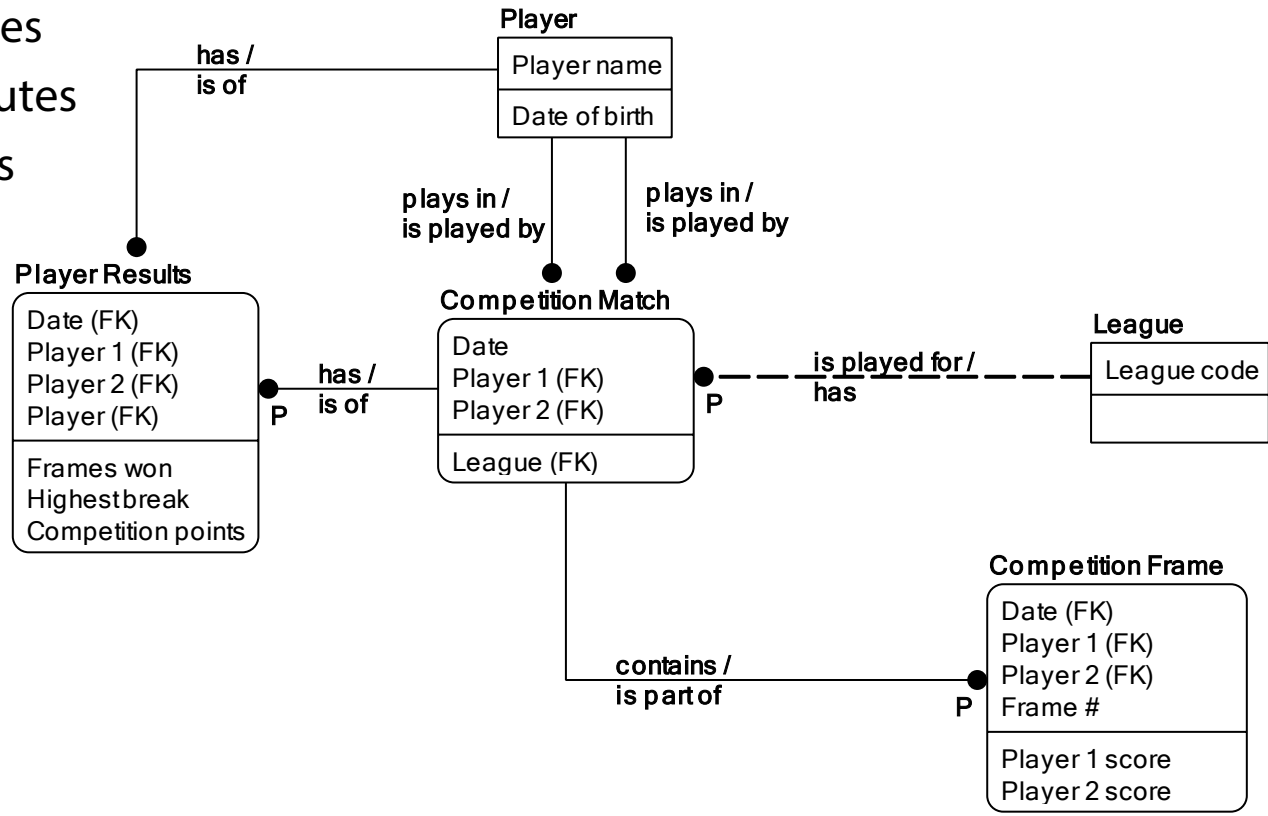
Date: October 3, 2012

Players	Katie	Jim
Frames won	2	1
HB	40	12
	Frame results	
Frame 1	51	37
Frame 2	30	63
Frame 3	62	18



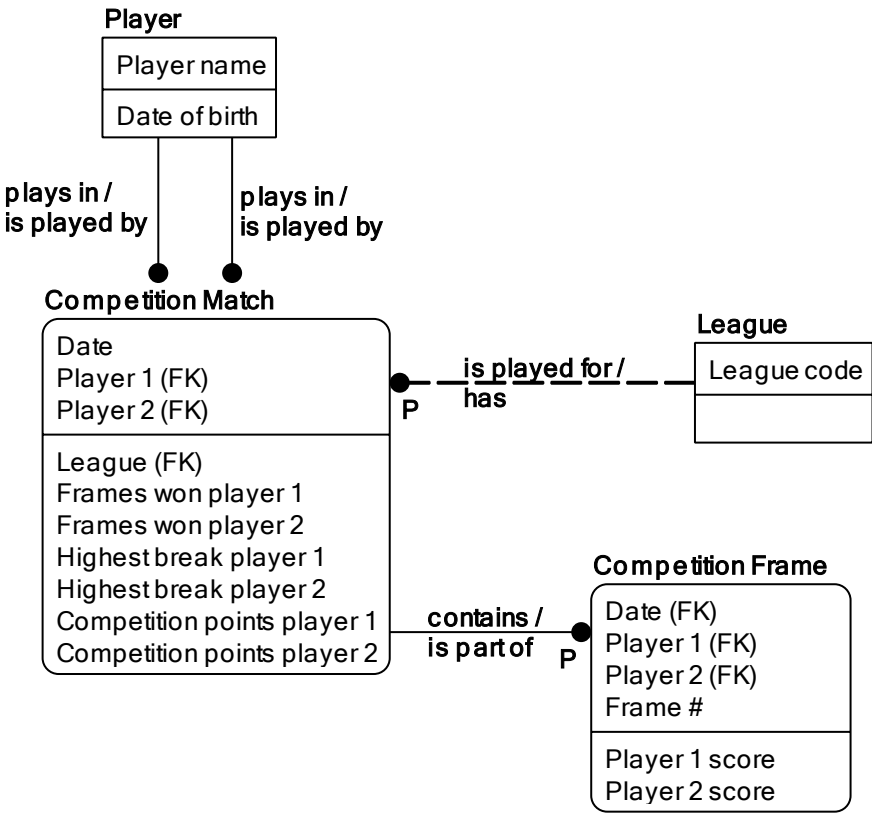
Demo: verify that attributes are atomic

- **Are attributes atomic?**
  - Composite attributes
  - Multi-valued attributes
  - Repeated attributes

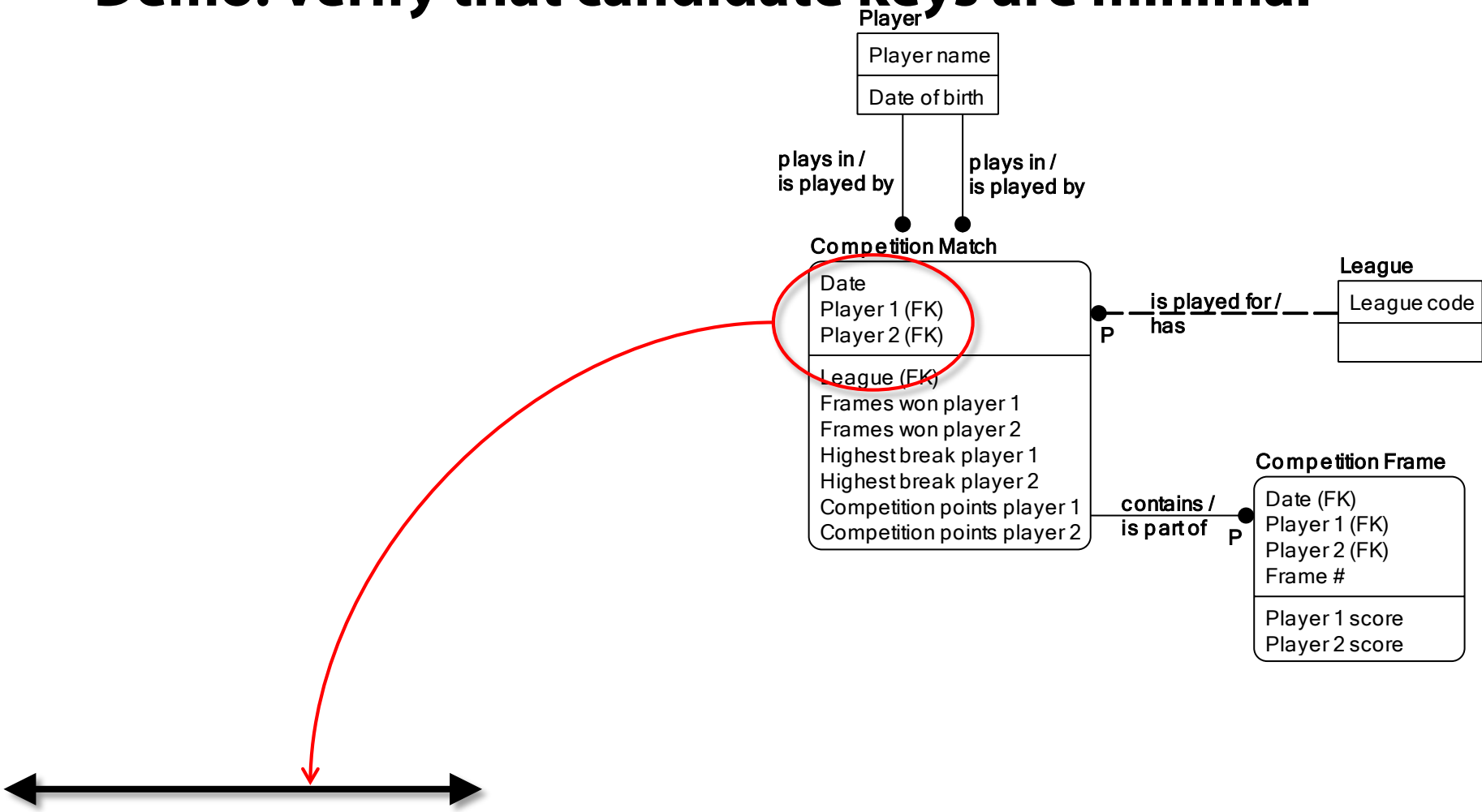


Demo: verify that attributes are atomic

- **Are attributes atomic?**
  - Composite attributes
  - Multi-valued attributes
  - Repeated attributes

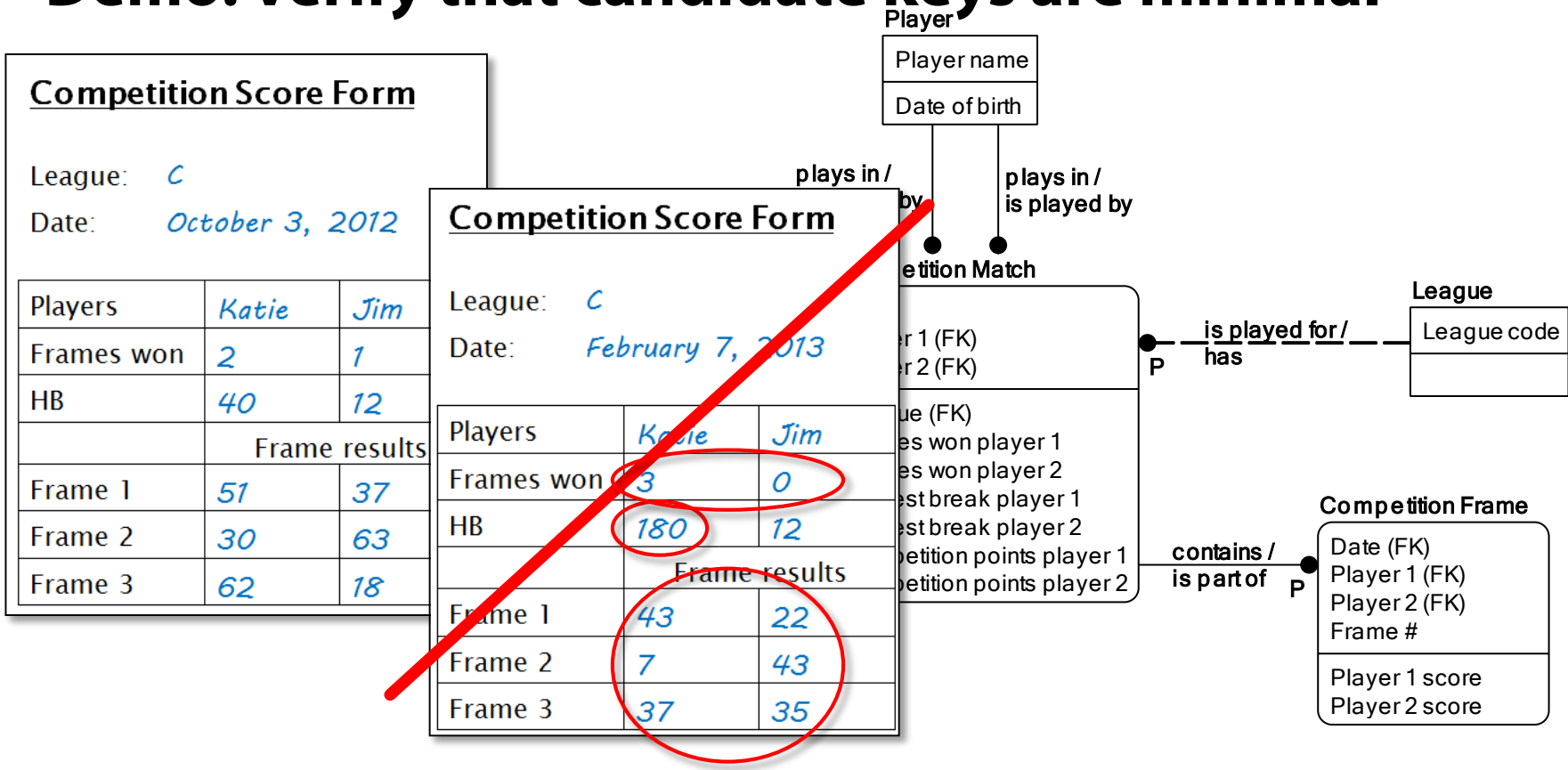


Demo: verify that candidate keys are minimal



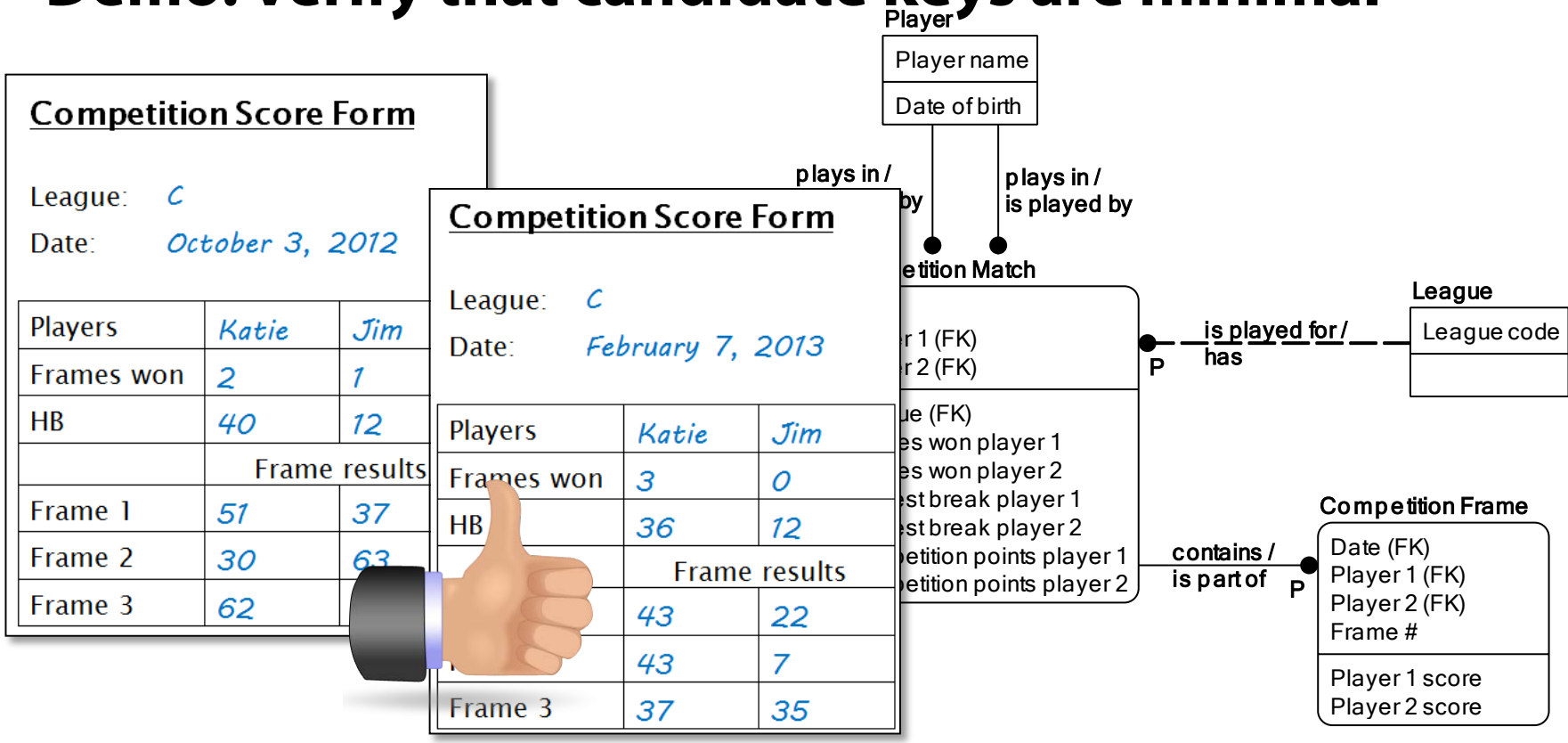
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2

Demo: verify that candidate keys are minimal



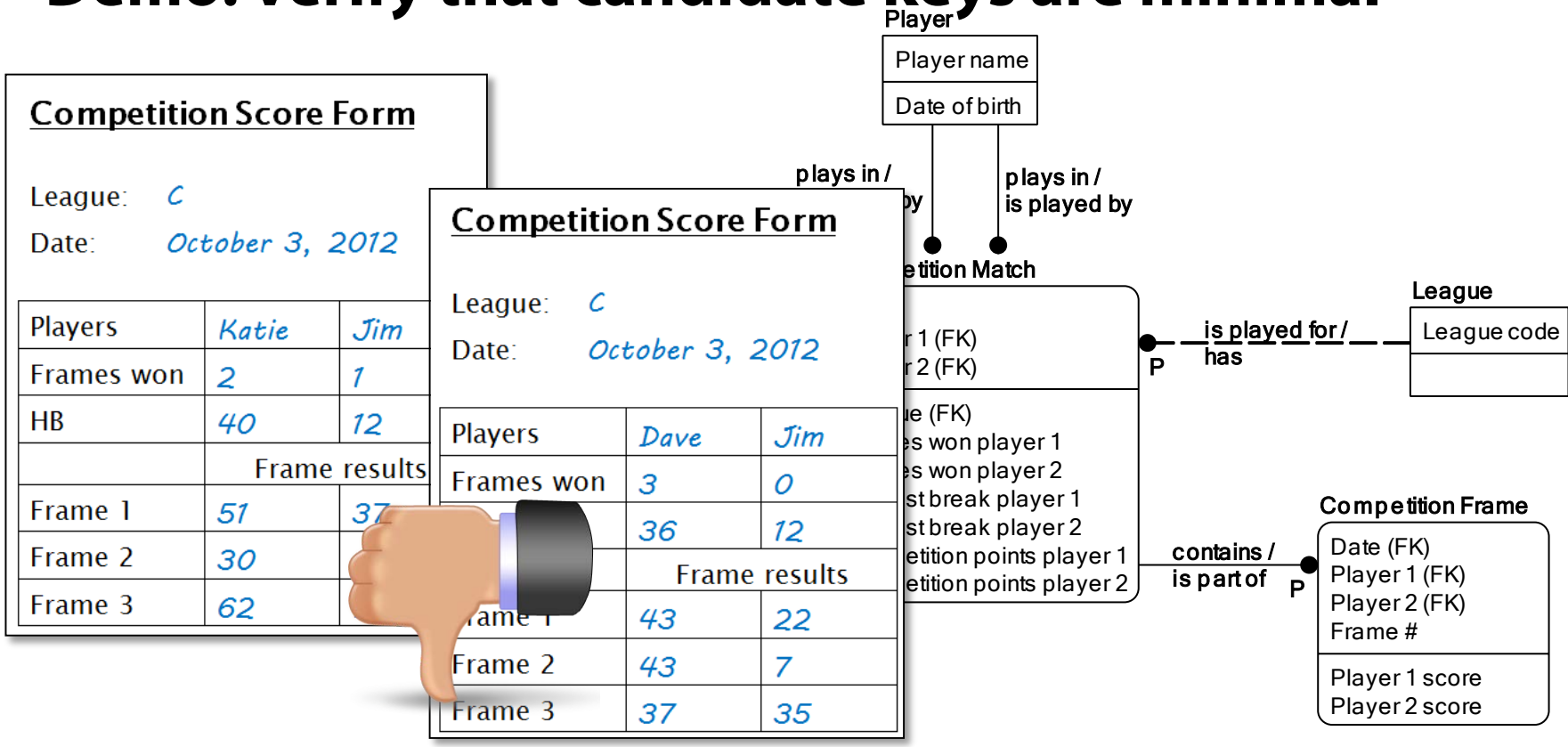
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-07	Katie	Jim	C	3	0	180	12	20	0

Demo: verify that candidate keys are minimal



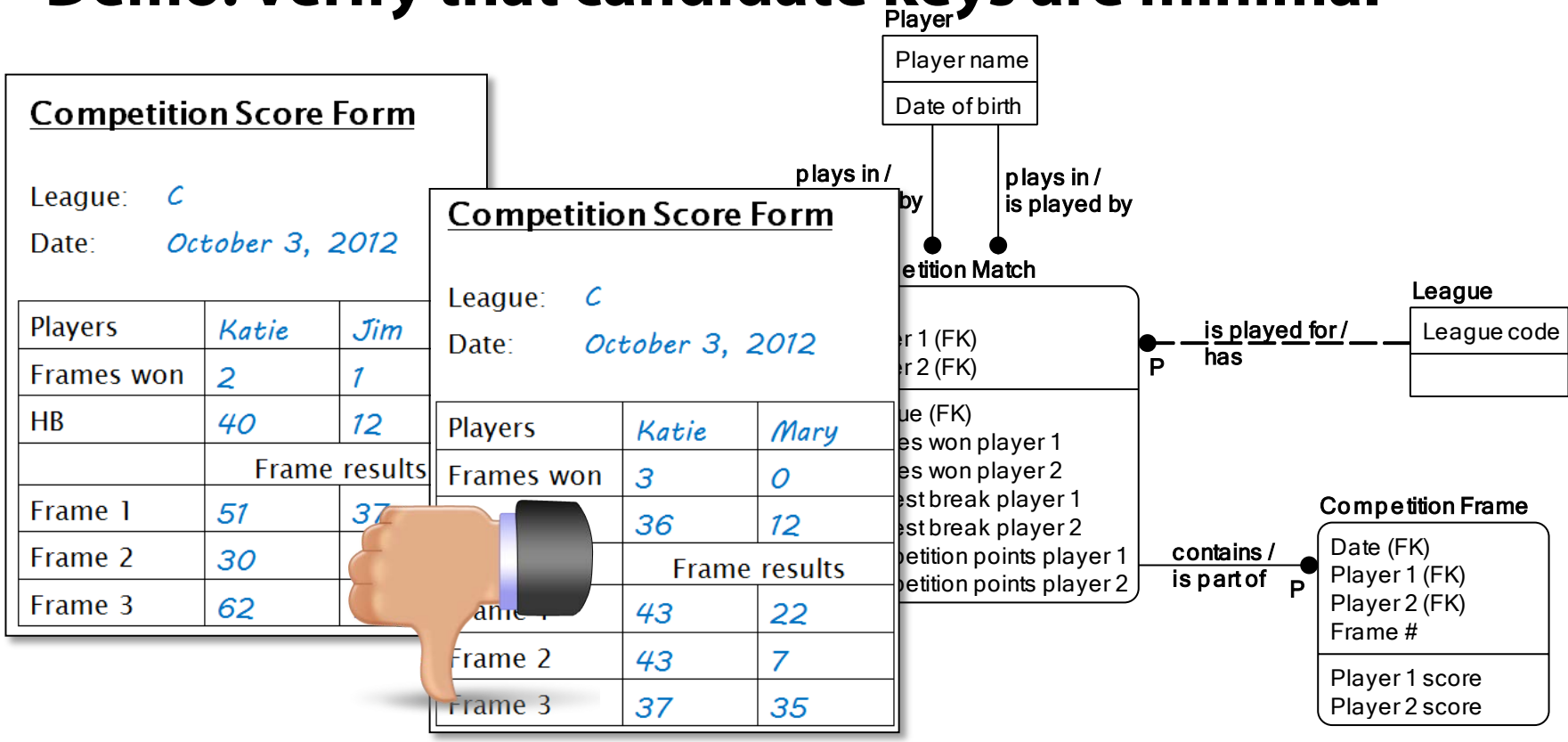
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-07	Katie	Jim	C	3	0	36	12	20	0

Demo: verify that candidate keys are minimal



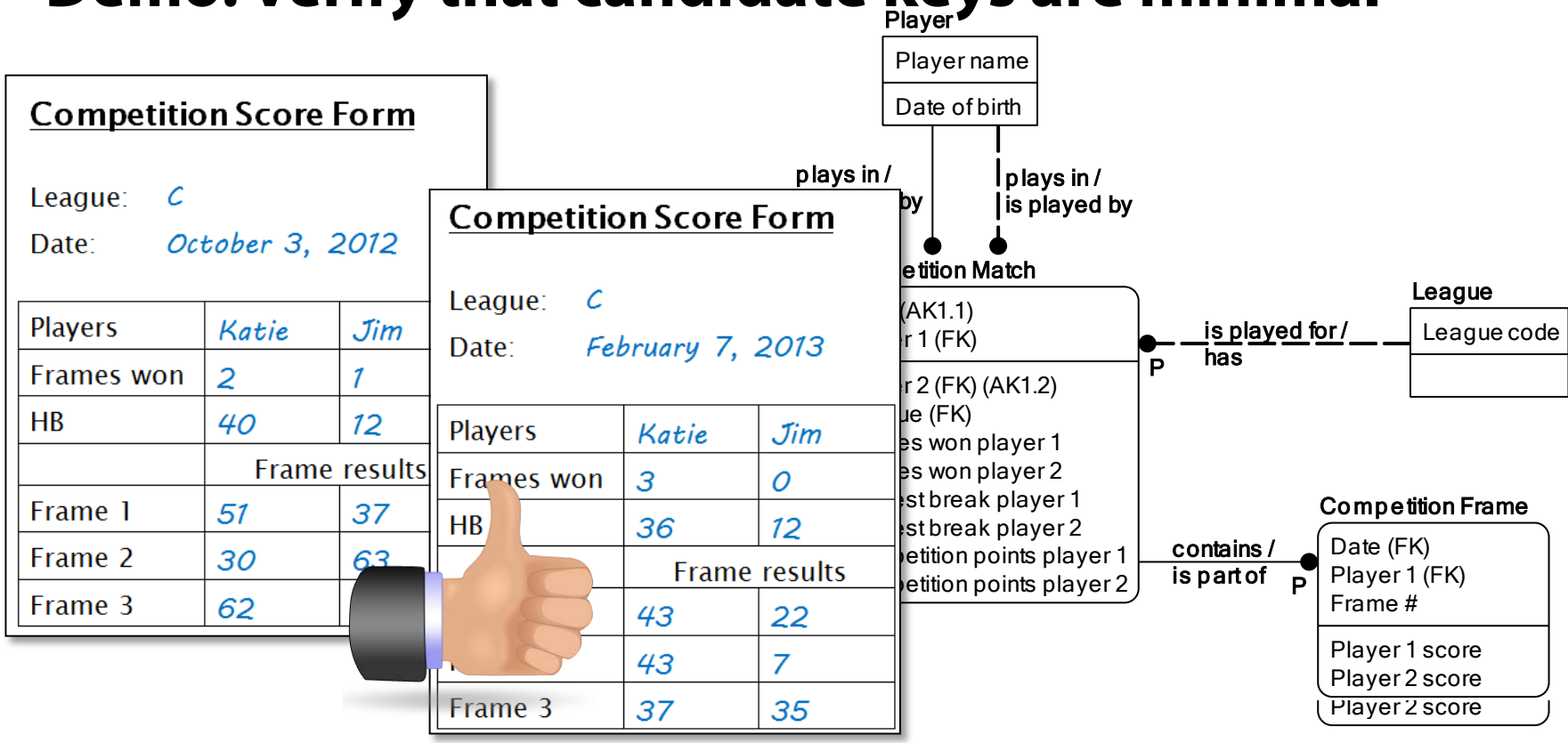
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2012-10-03	Dave	Jim	C	3	0	36	12	20	0

Demo: verify that candidate keys are minimal



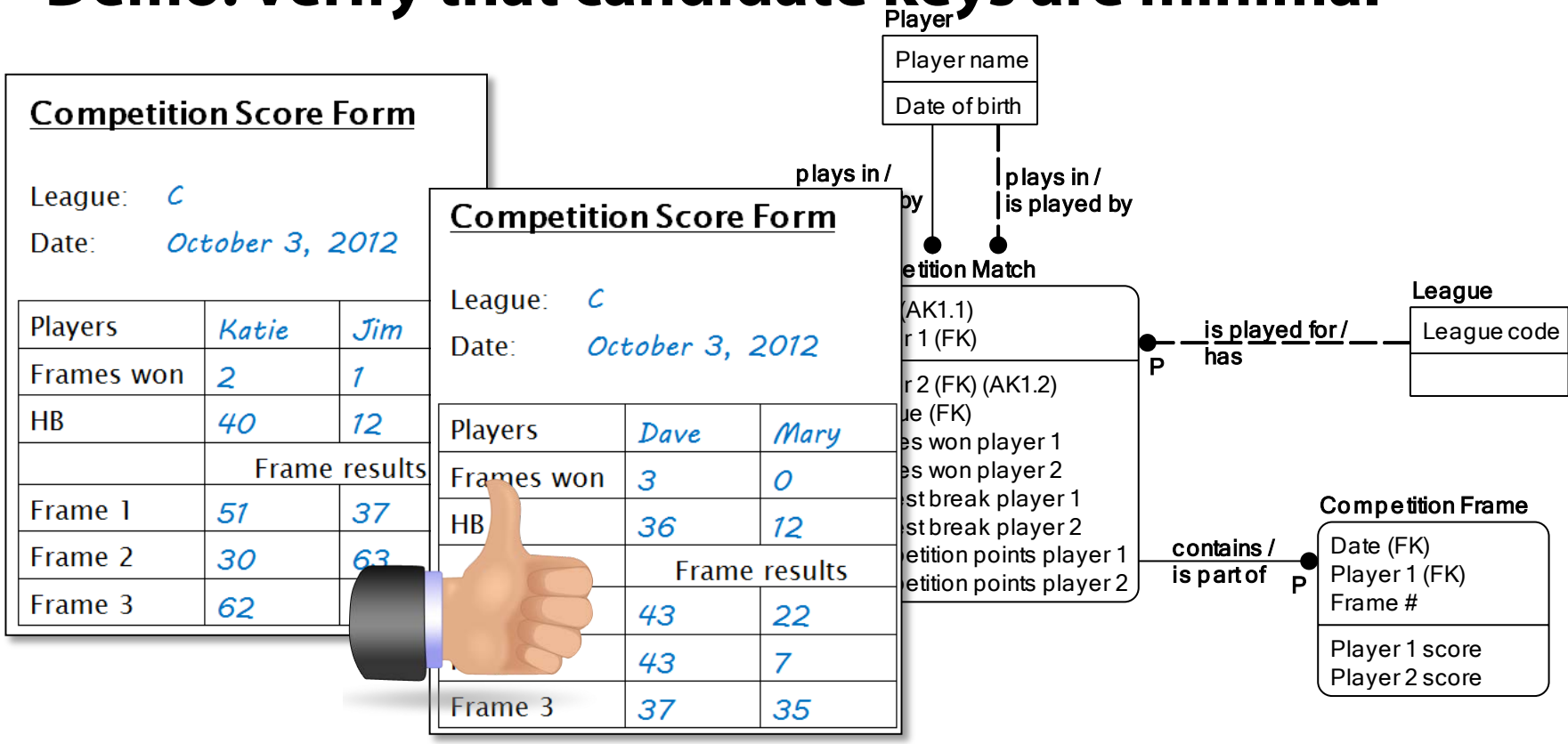
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2012-10-03	Katie	Mary	C	3	0	36	12	20	0

Demo: verify that candidate keys are minimal



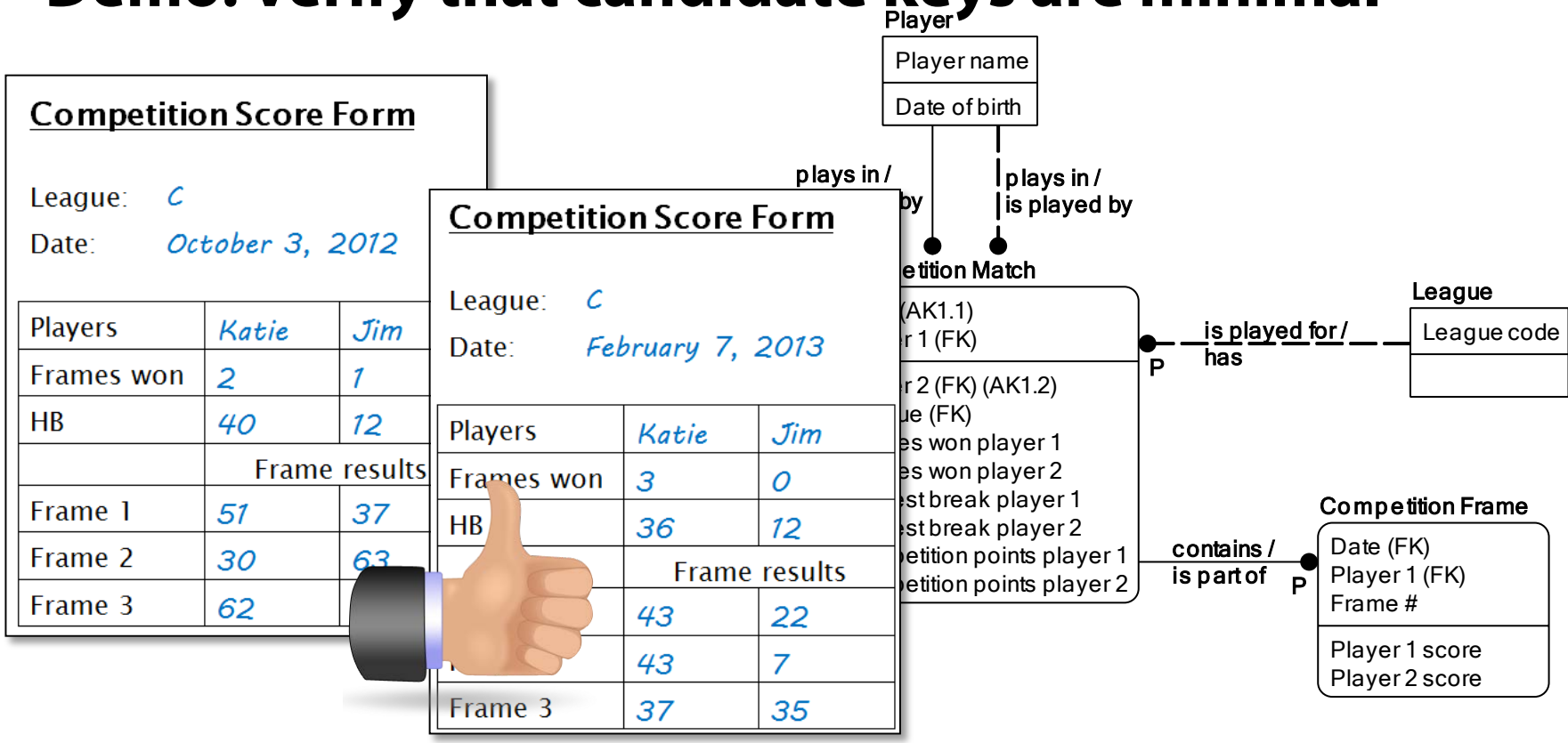
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-07	Katie	Jim	C	3	0	36	12	20	0

Demo: verify that candidate keys are minimal



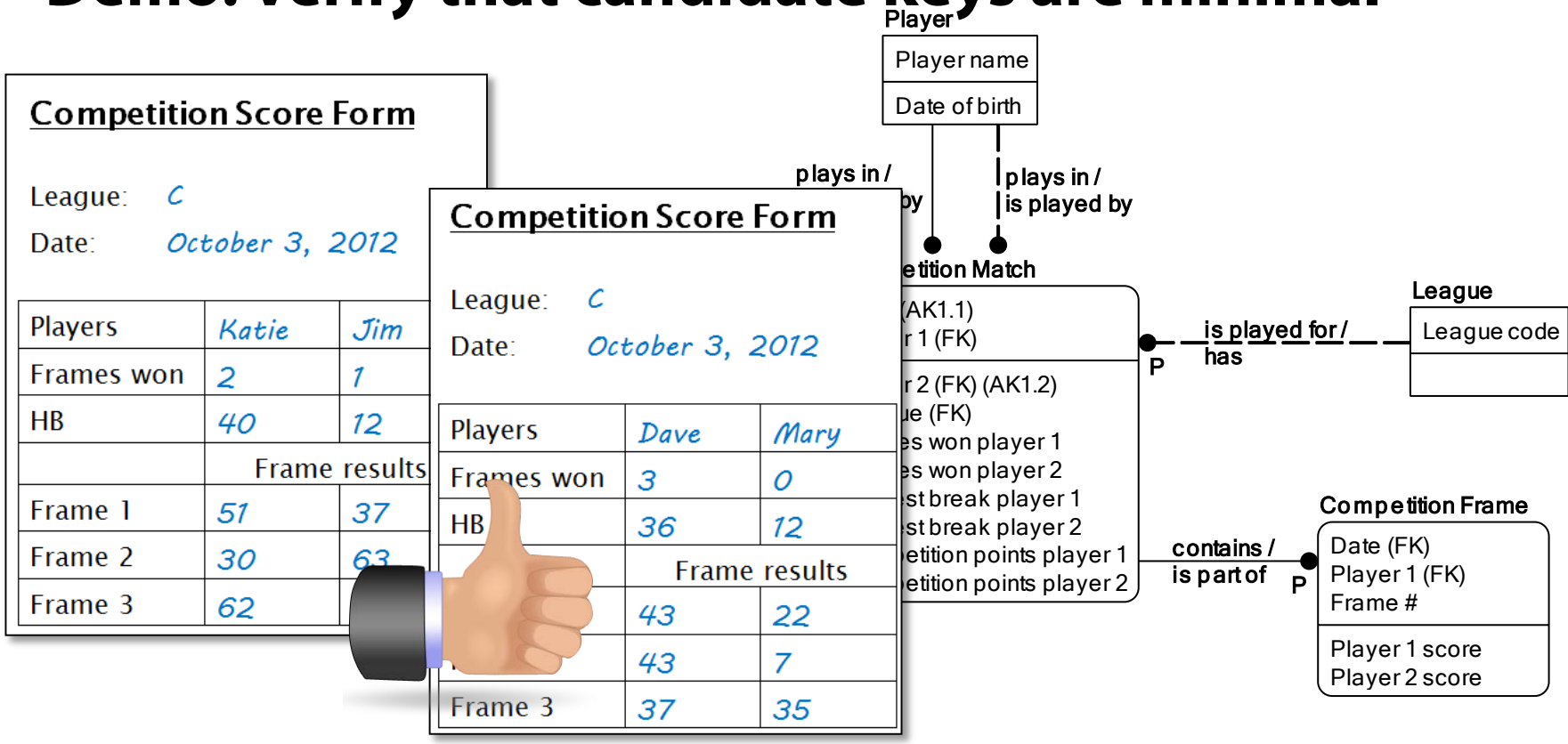
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2012-10-03	Dave	Mary	C	3	0	36	12	20	0

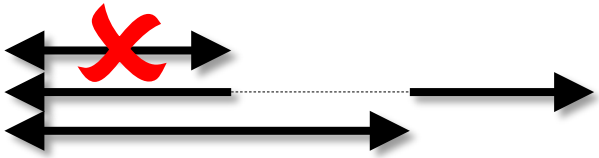
Demo: verify that candidate keys are minimal



Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-07	Katie	Jim	C	3	0	36	12	20	0

Demo: verify that candidate keys are minimal





Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2012-10-03	Dave	Mary	C	3	0	36	12	20	0

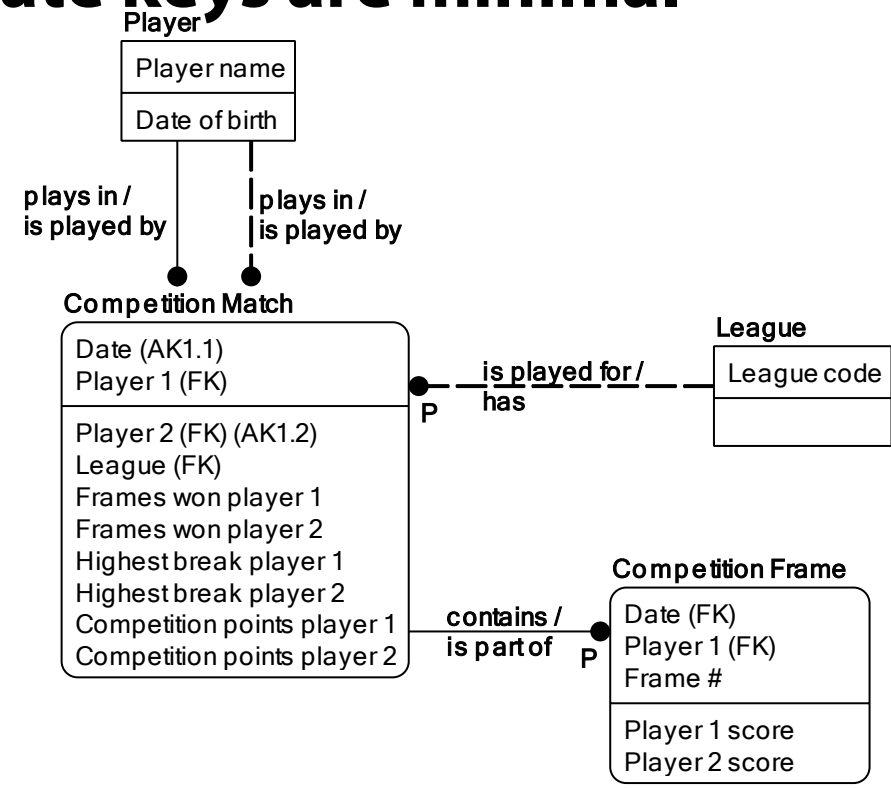
Demo: verify that candidate keys are minimal

Competition Score Form

League: C

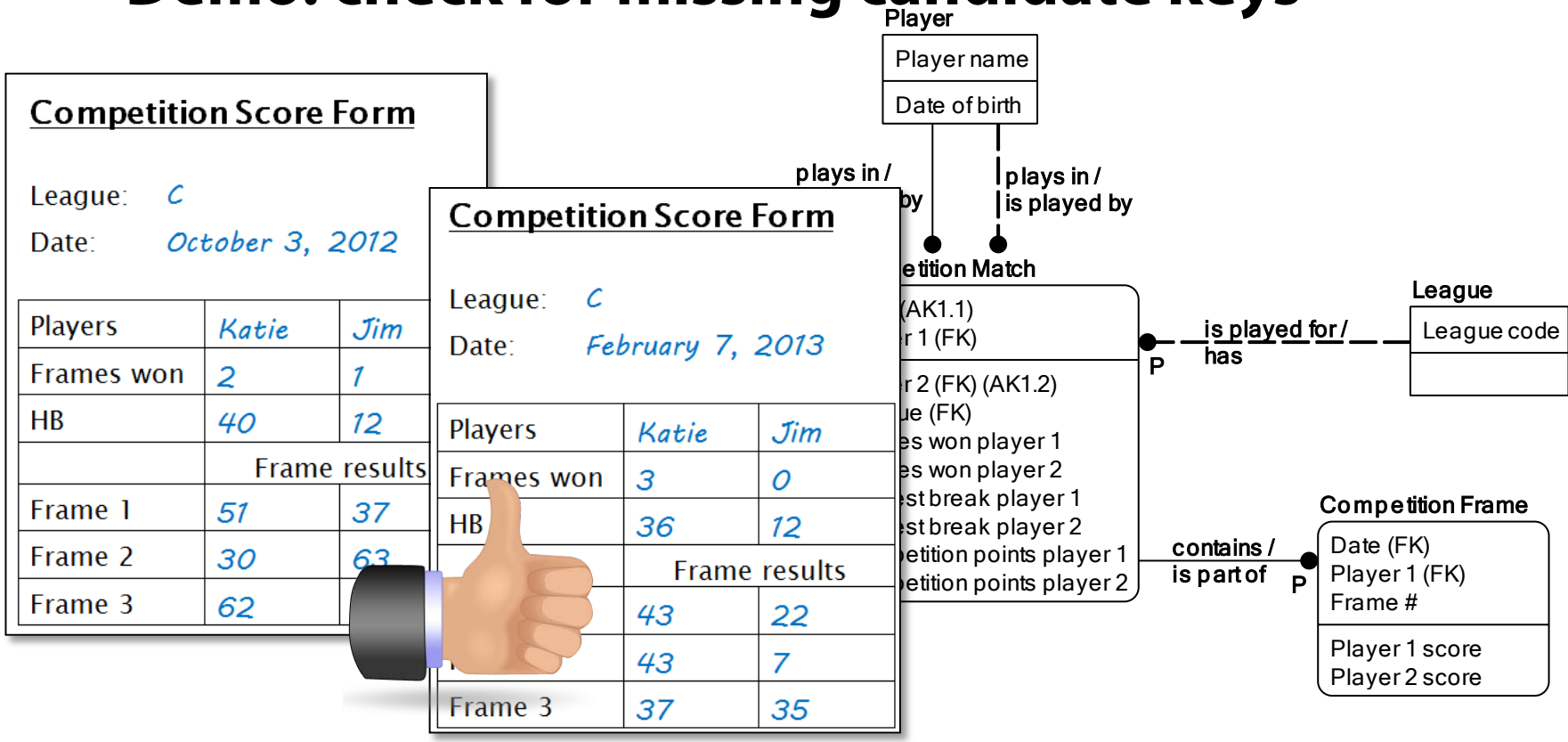
Date: October 3, 2012

Players	Katie	Jim
Frames won	2	1
HB	40	12
	Frame results	
Frame 1	51	37
Frame 2	30	63
Frame 3	62	18



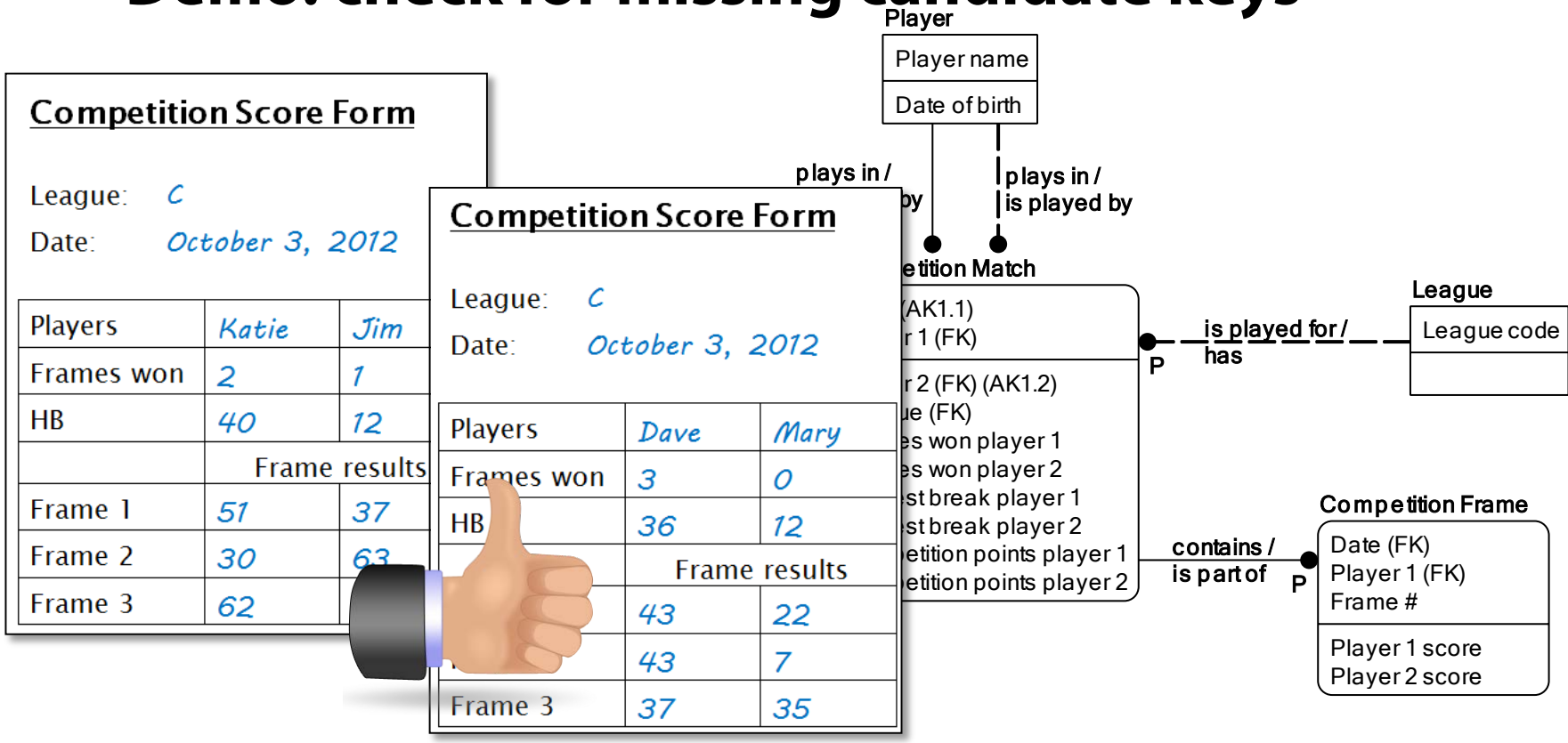
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10

Demo: check for missing candidate keys



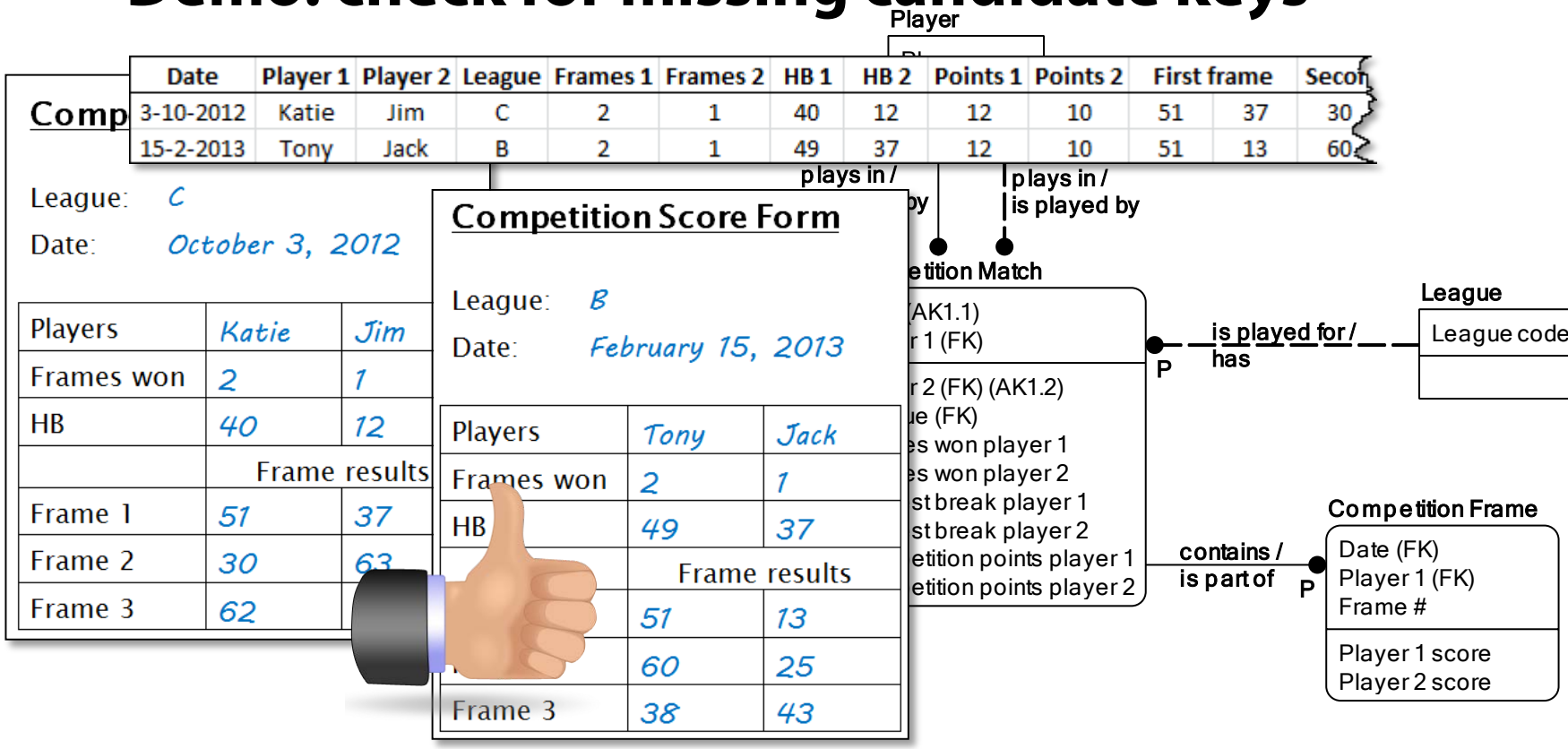
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-07	Katie	Jim	C	3	0	36	12	20	0

Demo: check for missing candidate keys



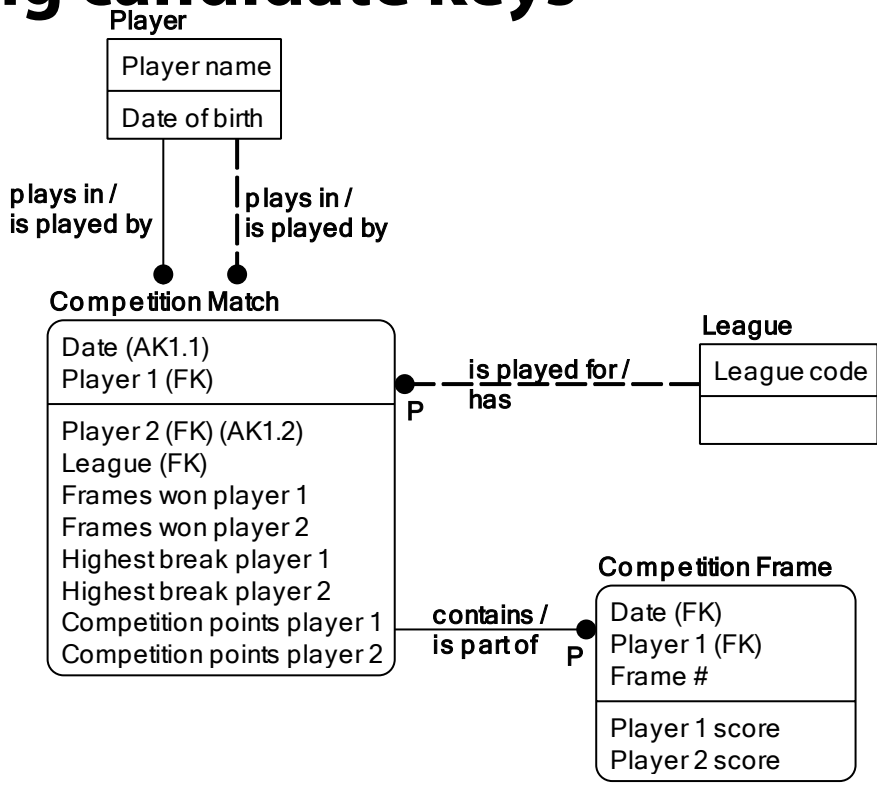
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2012-10-03	Dave	Mary	C	3	0	36	12	20	0

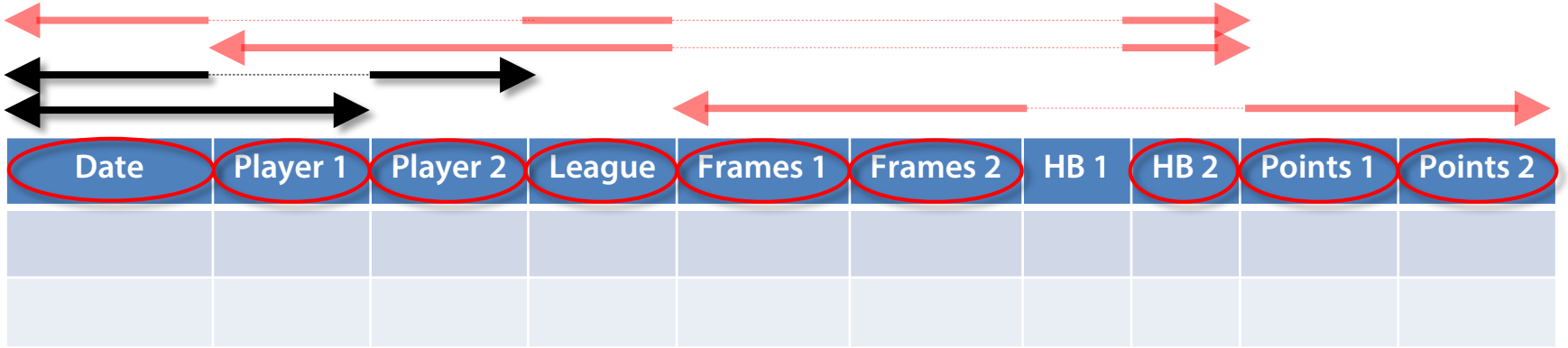
Demo: check for missing candidate keys



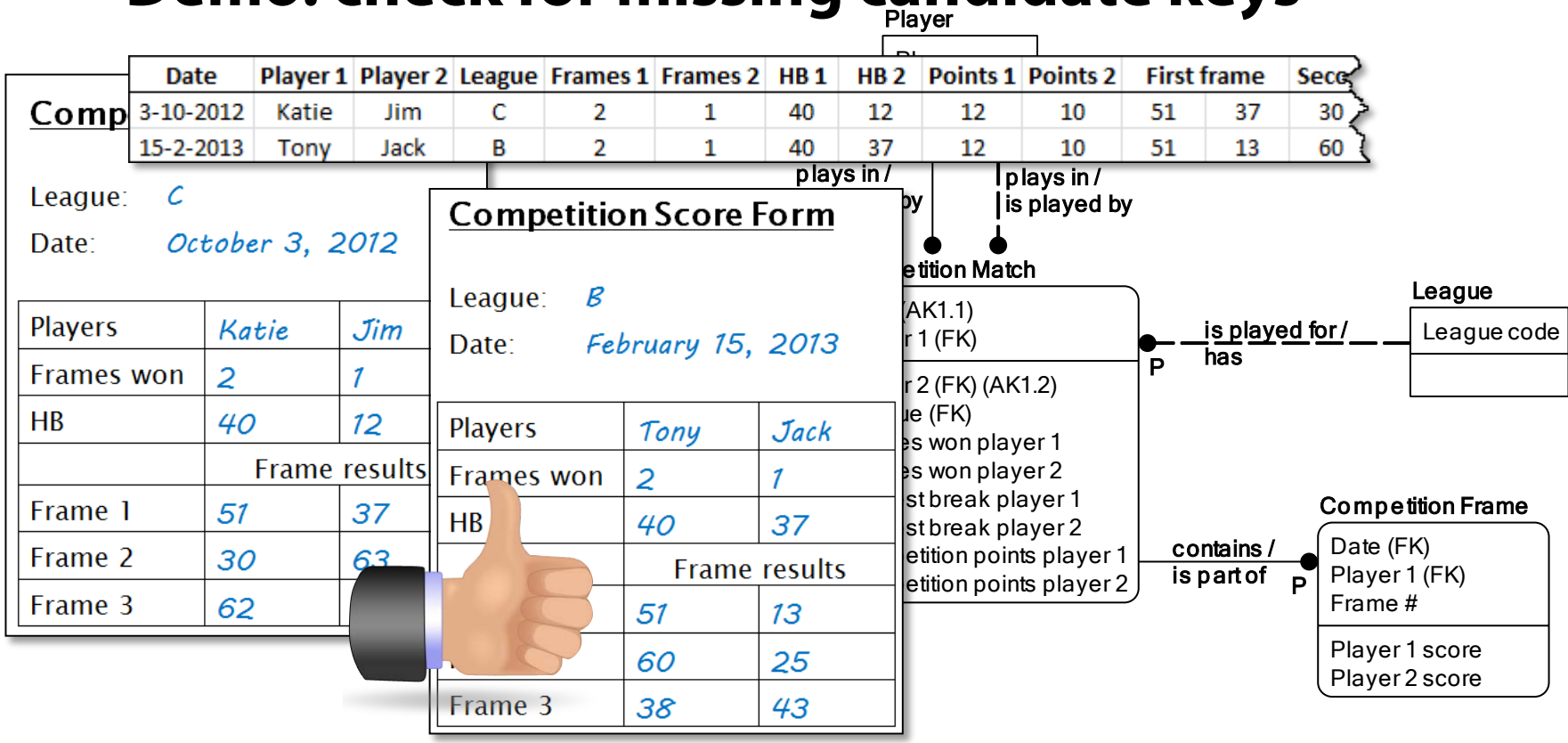
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-15	Tony	Jack	B	2	1	49	37	12	10

Demo: check for missing candidate keys



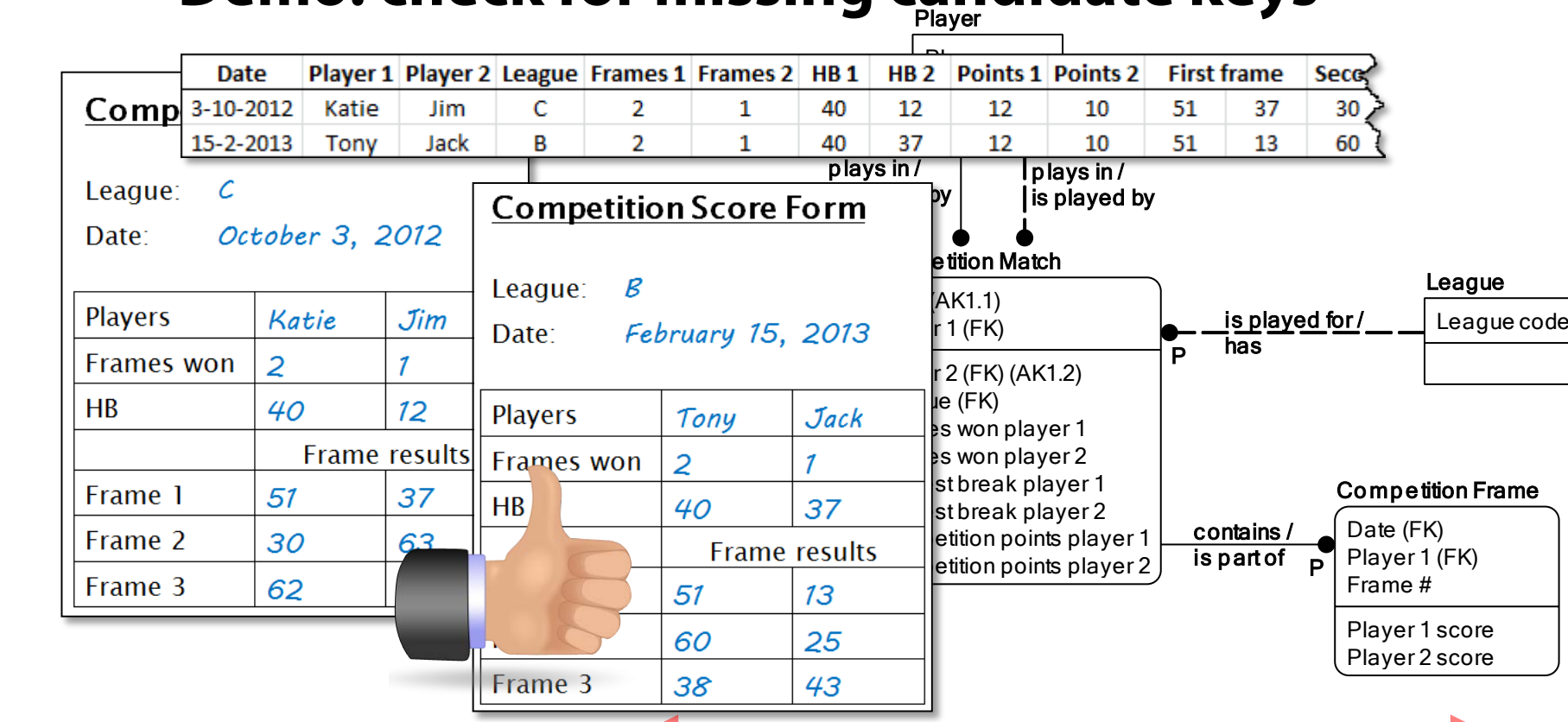


Demo: check for missing candidate keys



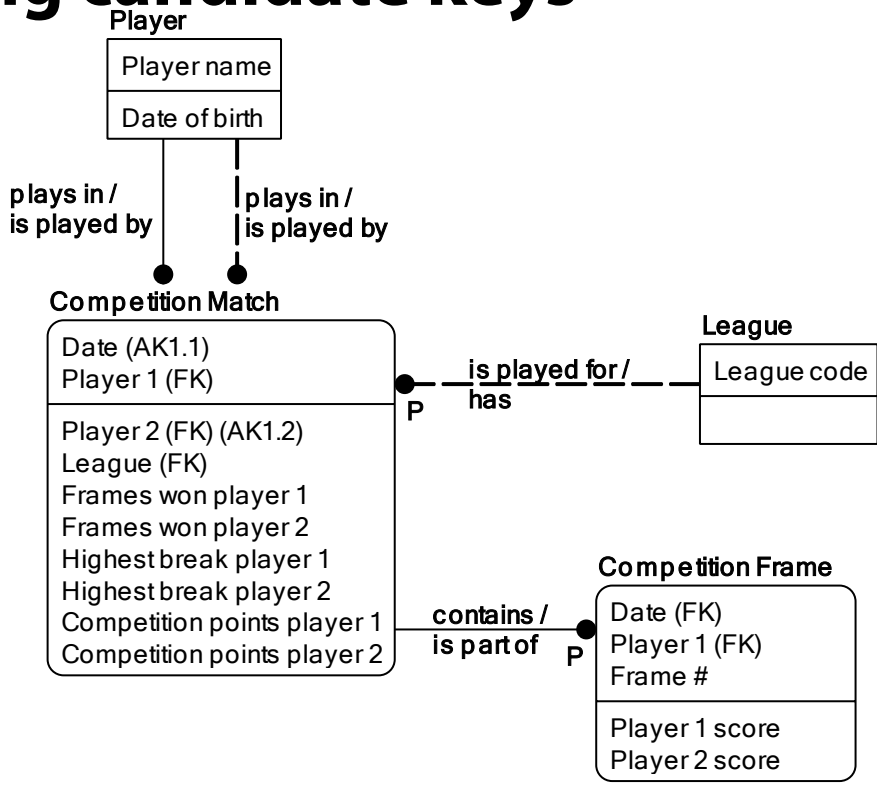
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-15	Tony	Jack	B	2	1	40	37	12	10

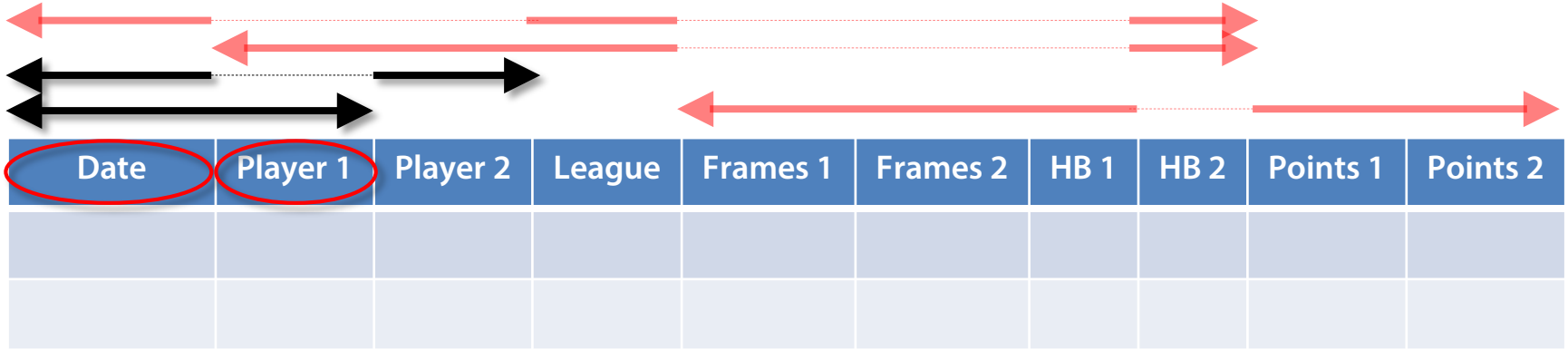
Demo: check for missing candidate keys



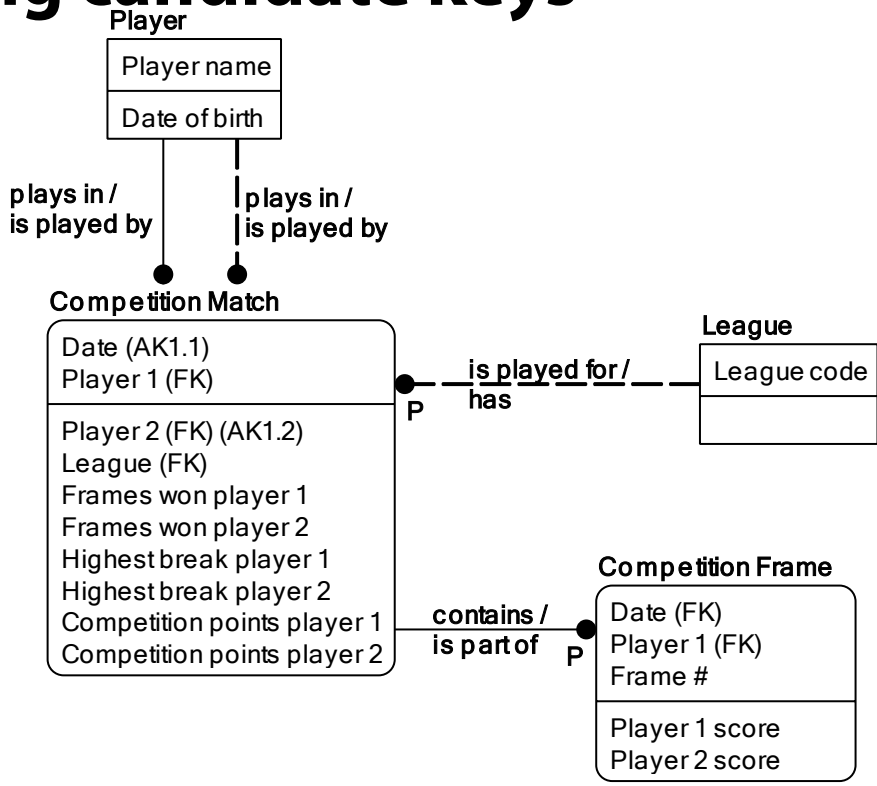
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-15	Tony	Jack	B	2	1	40	37	12	10

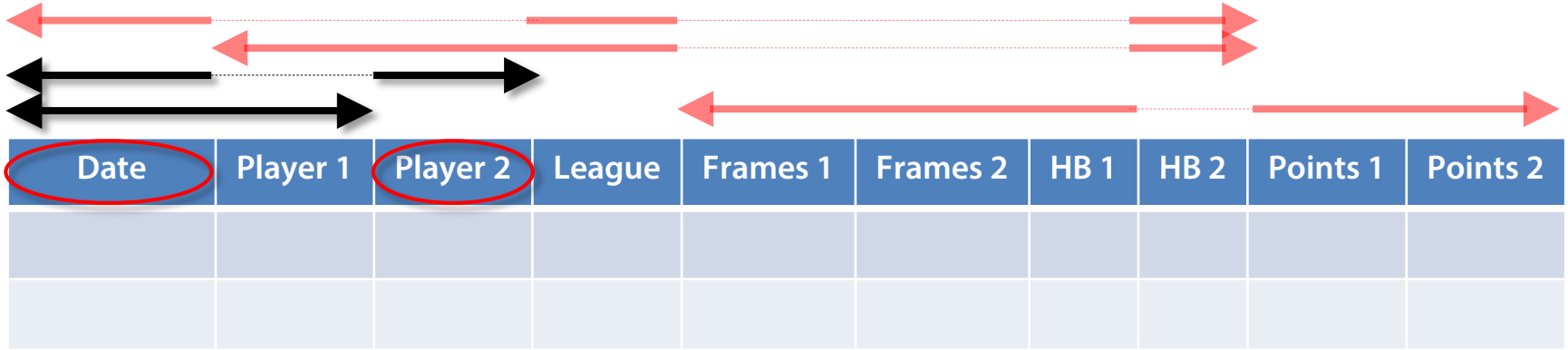
Demo: check for missing candidate keys



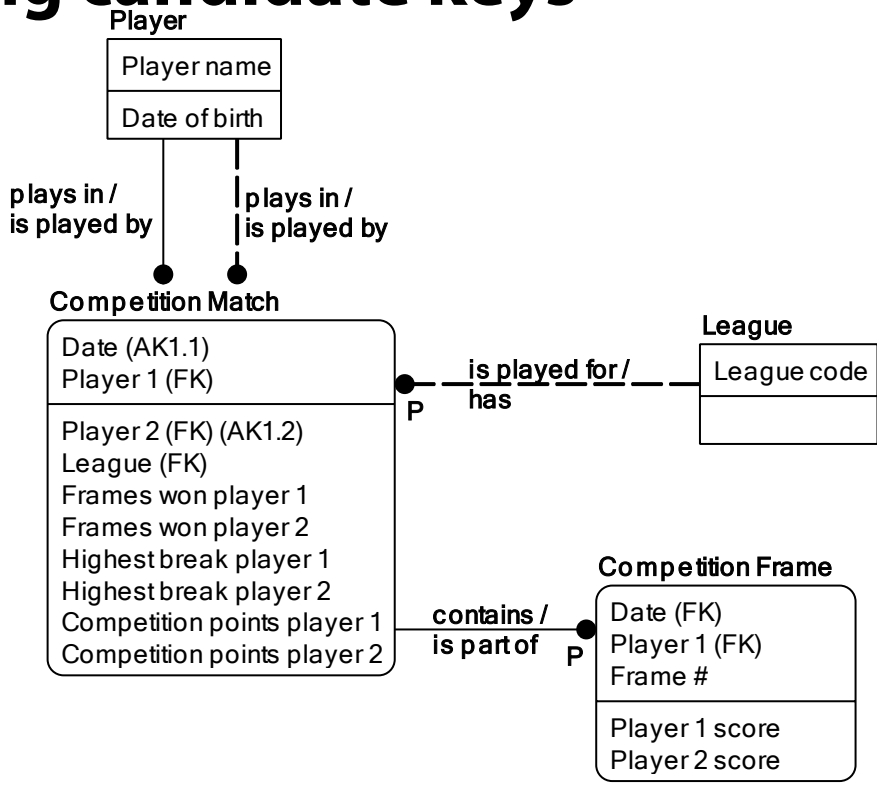


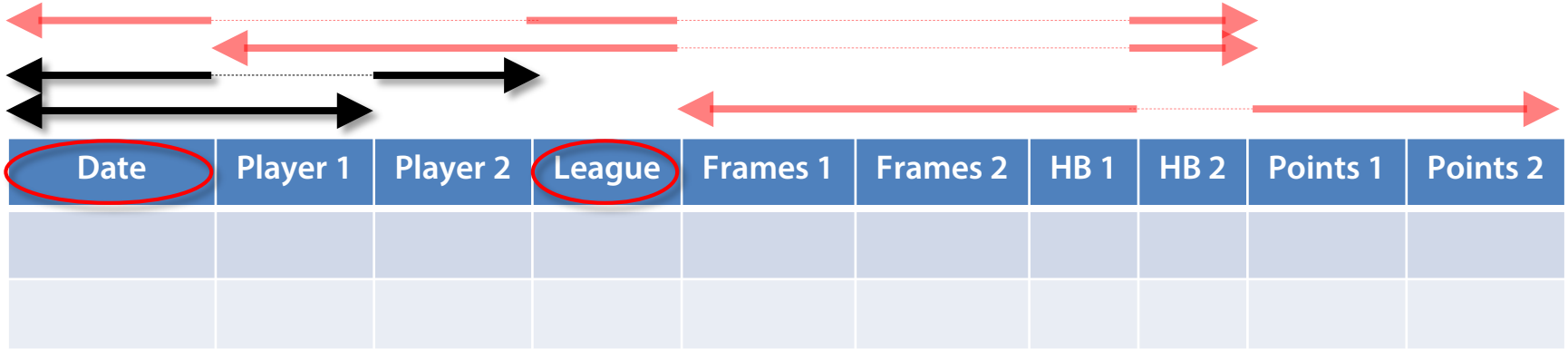
Demo: check for missing candidate keys



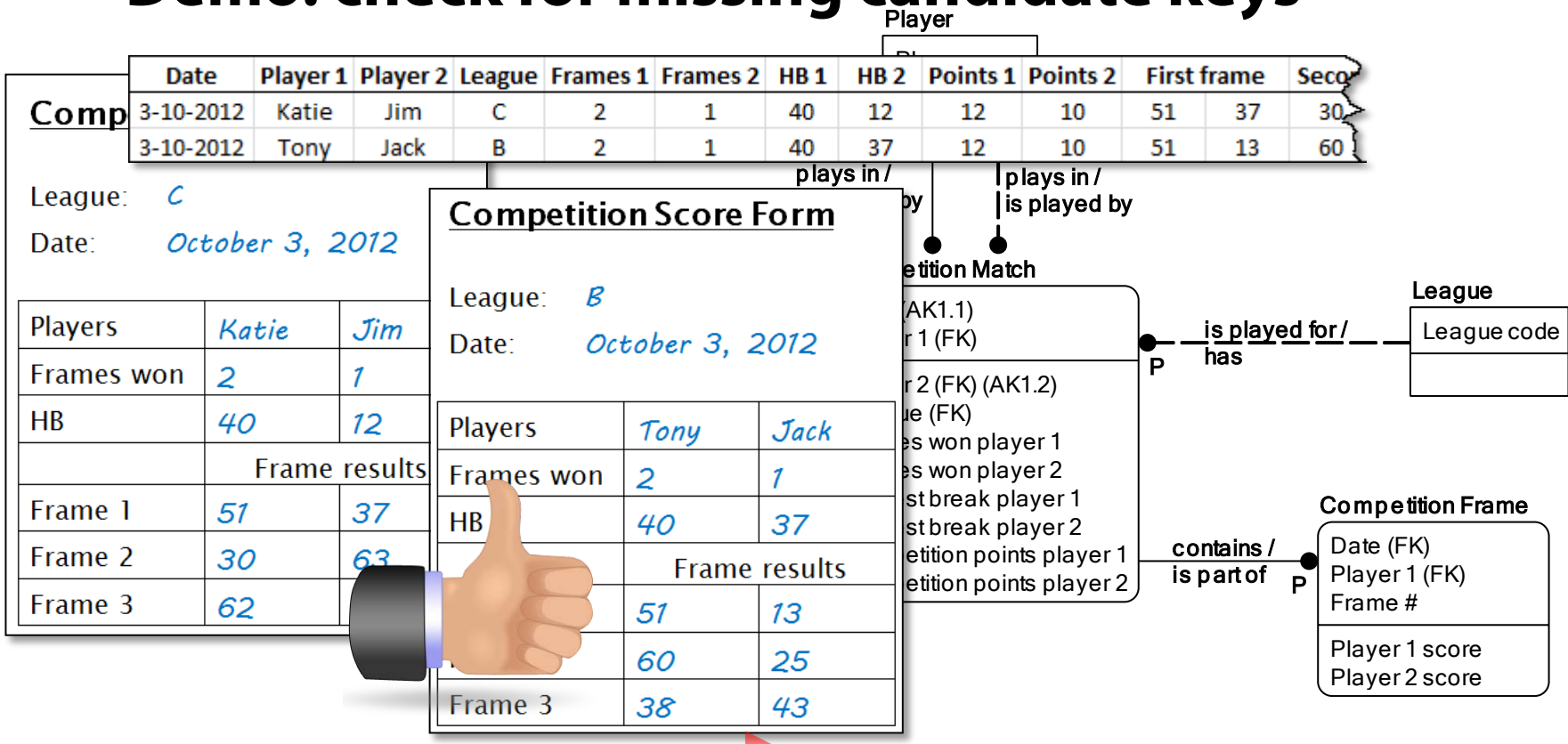


Demo: check for missing candidate keys



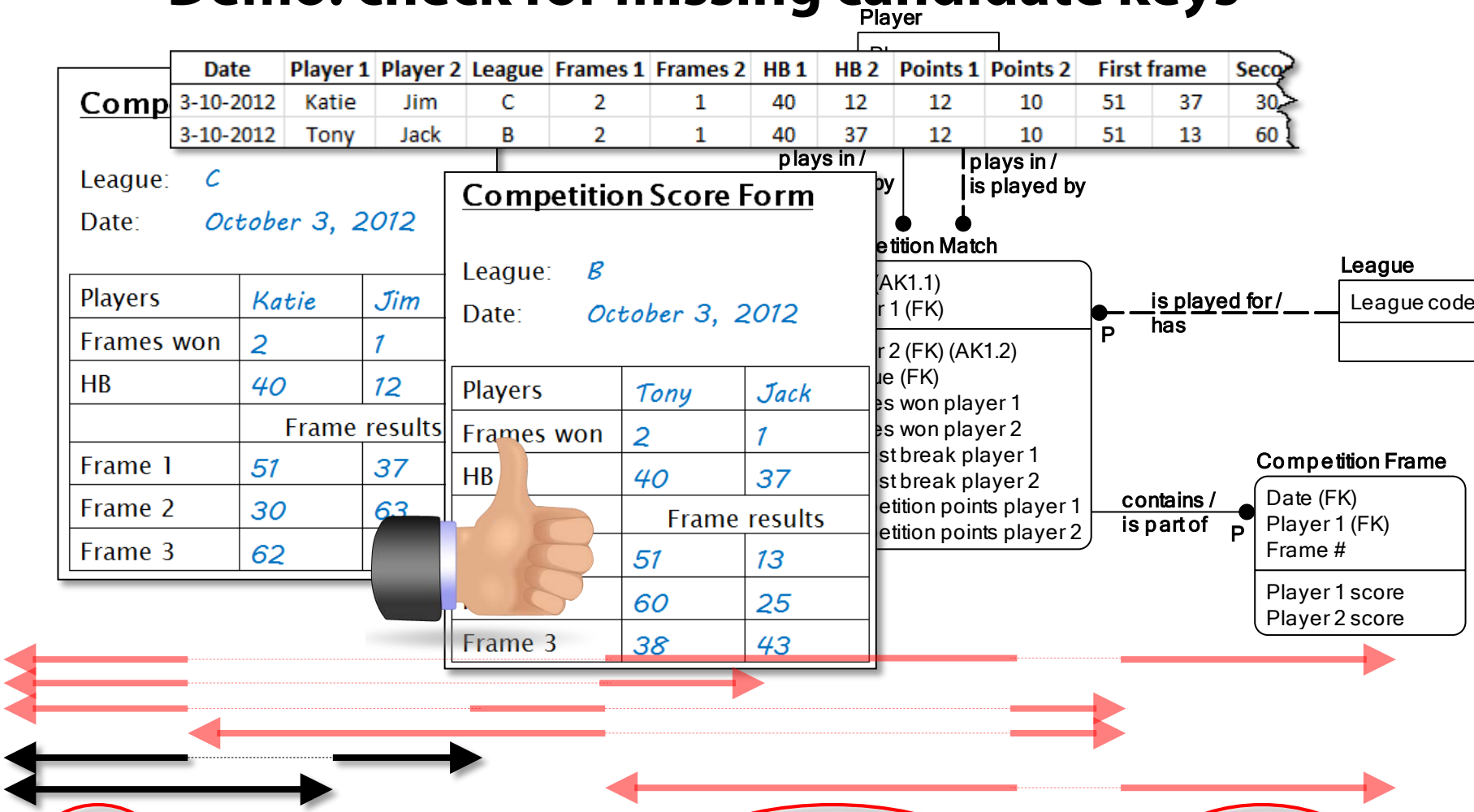


Demo: check for missing candidate keys



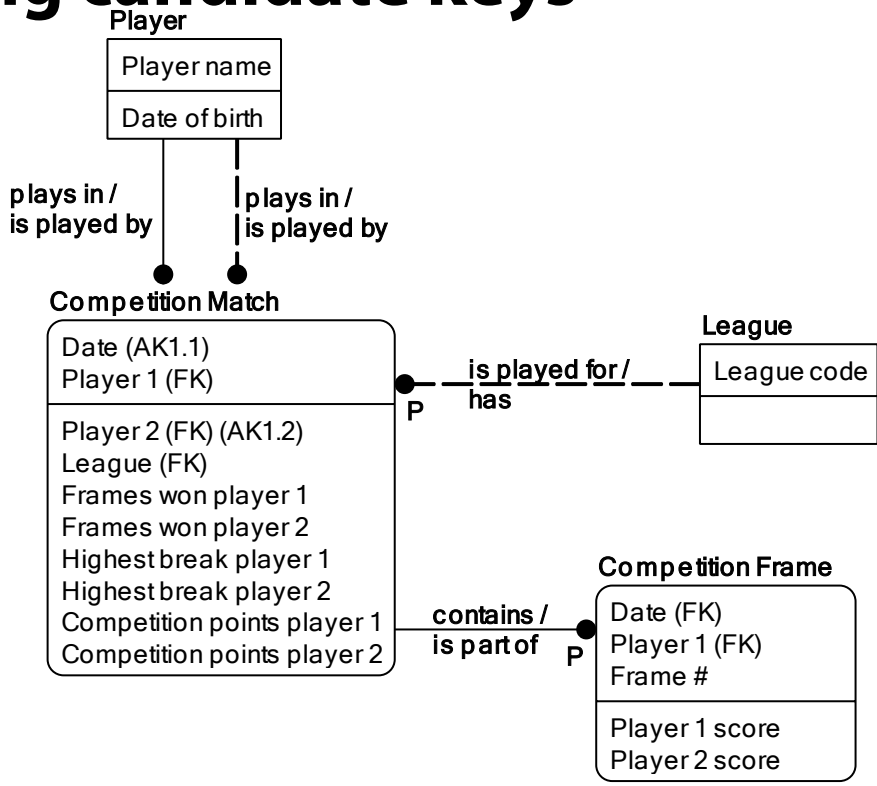
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2012-10-03	Tony	Jack	B	2	1	40	37	12	10

Demo: check for missing candidate keys



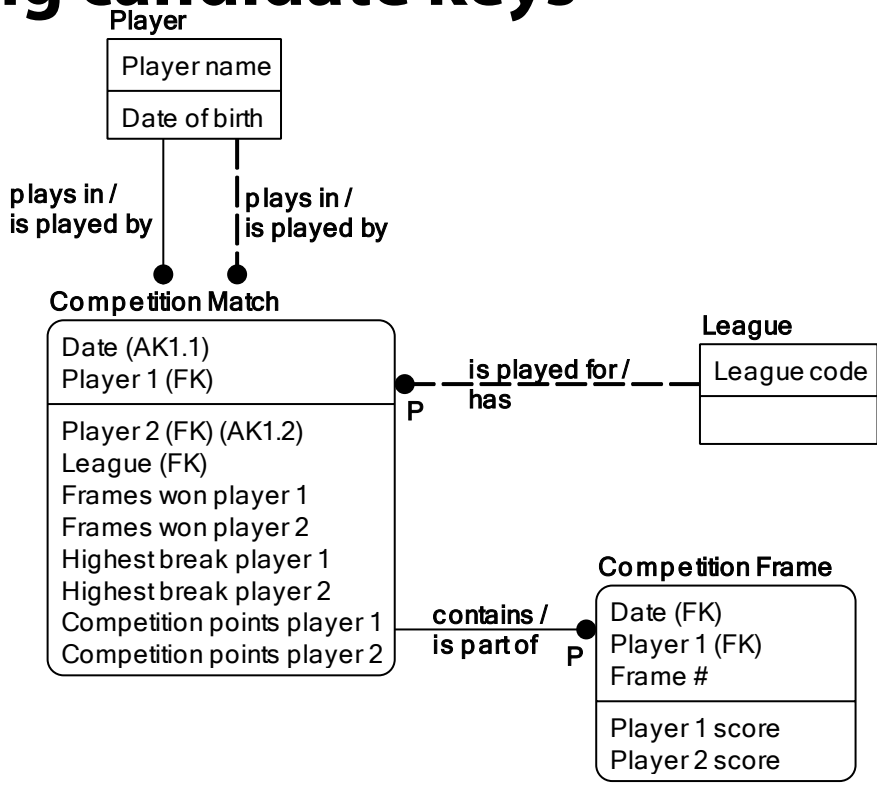
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2012-10-03	Tony	Jack	B	2	1	40	37	12	10

Demo: check for missing candidate keys



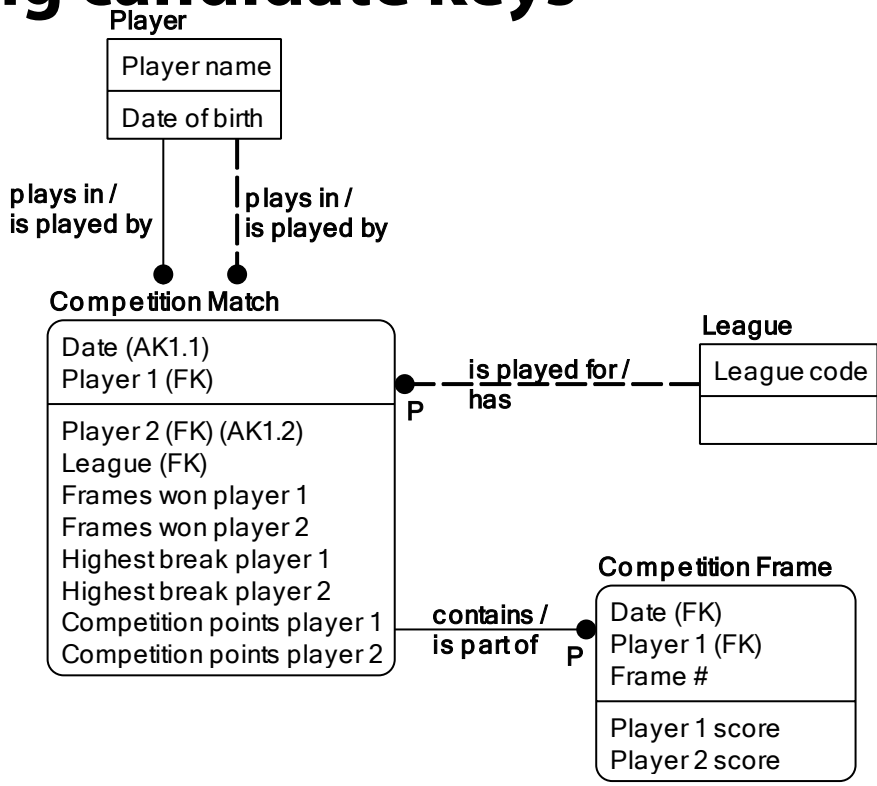
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2

Demo: check for missing candidate keys



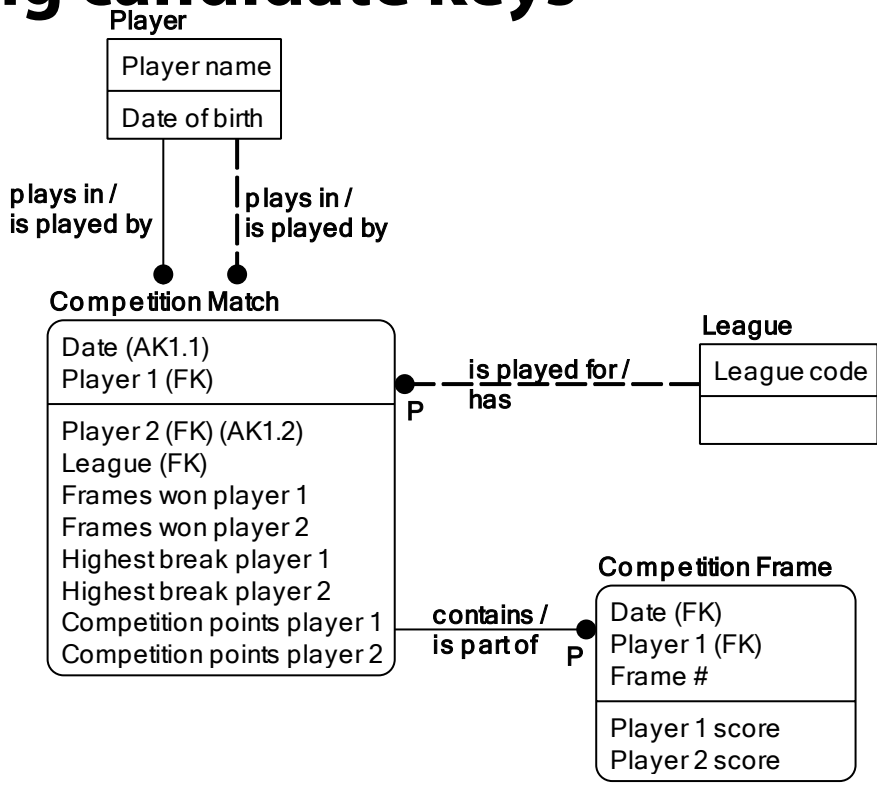
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2

Demo: check for missing candidate keys



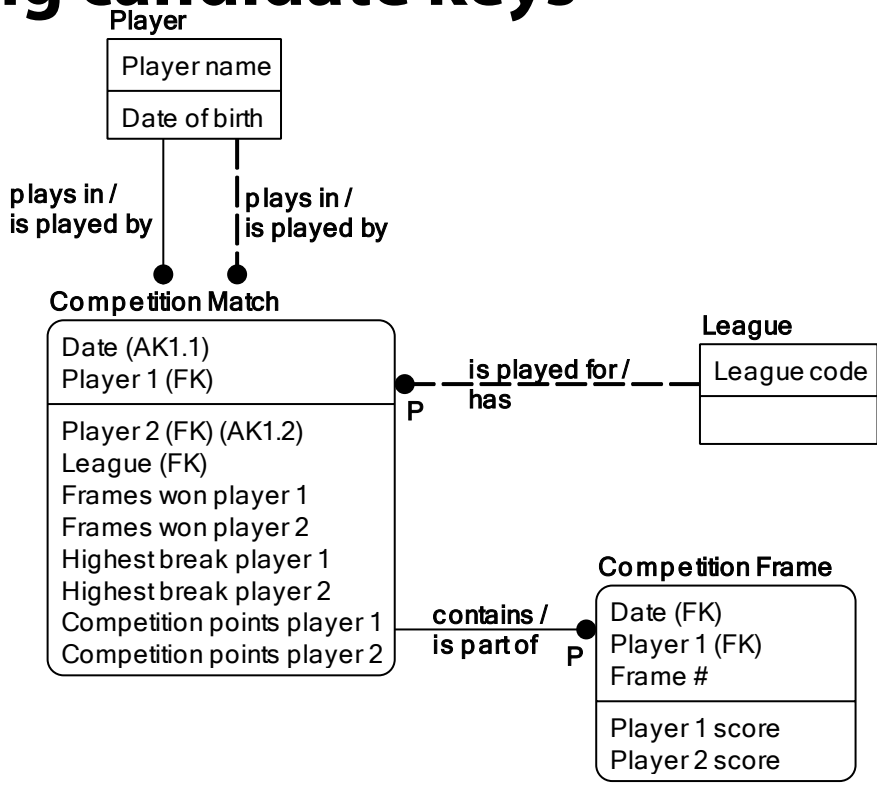
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2

Demo: check for missing candidate keys



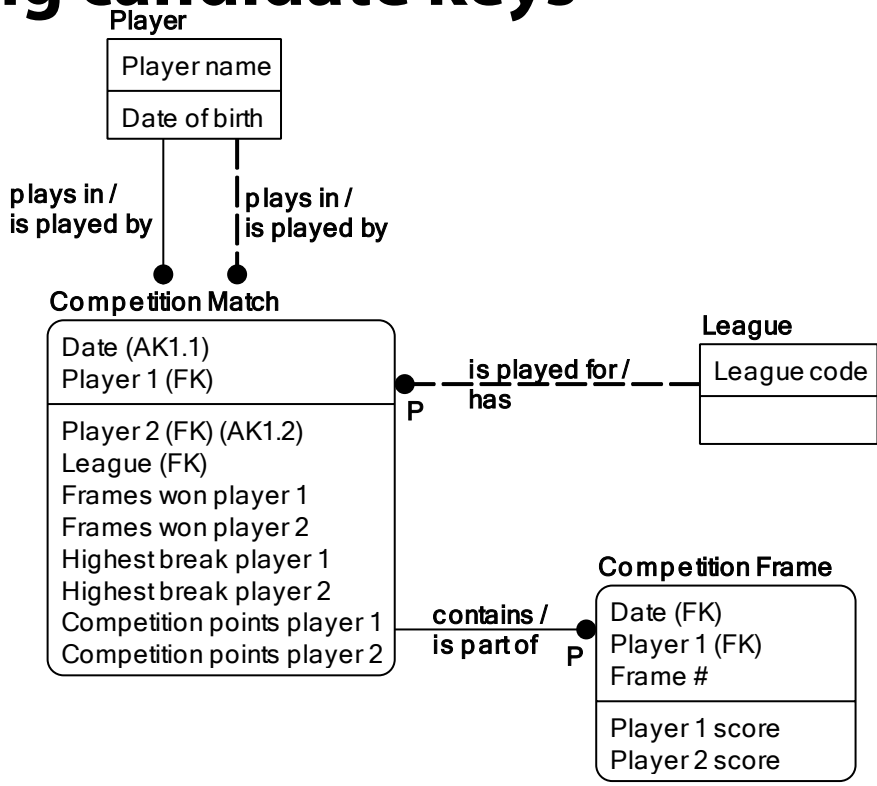
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2

Demo: check for missing candidate keys



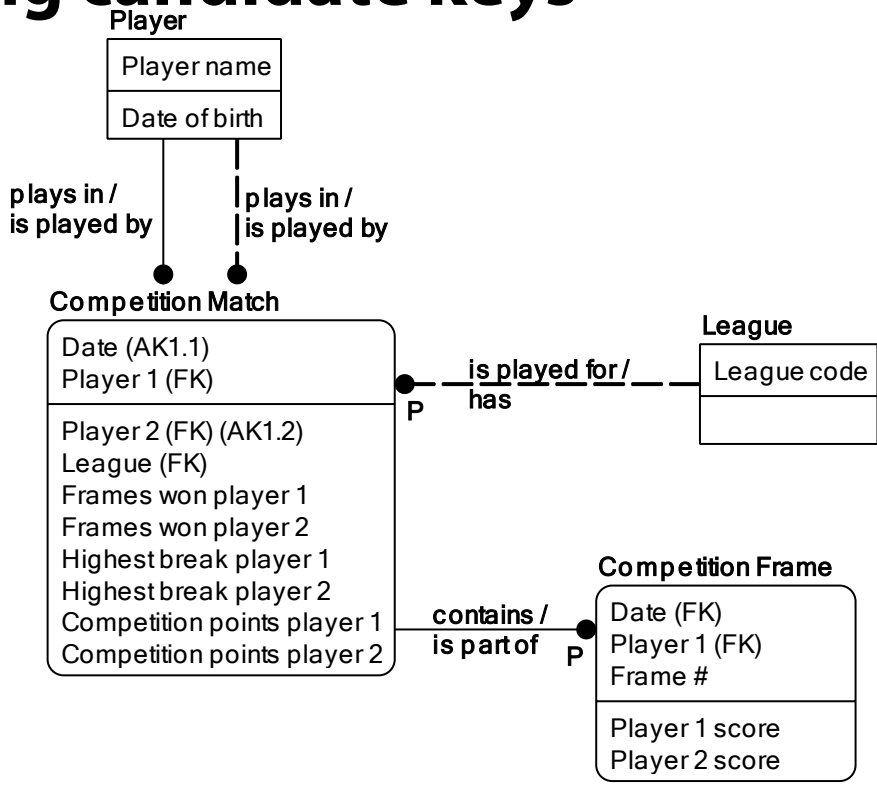
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2

Demo: check for missing candidate keys



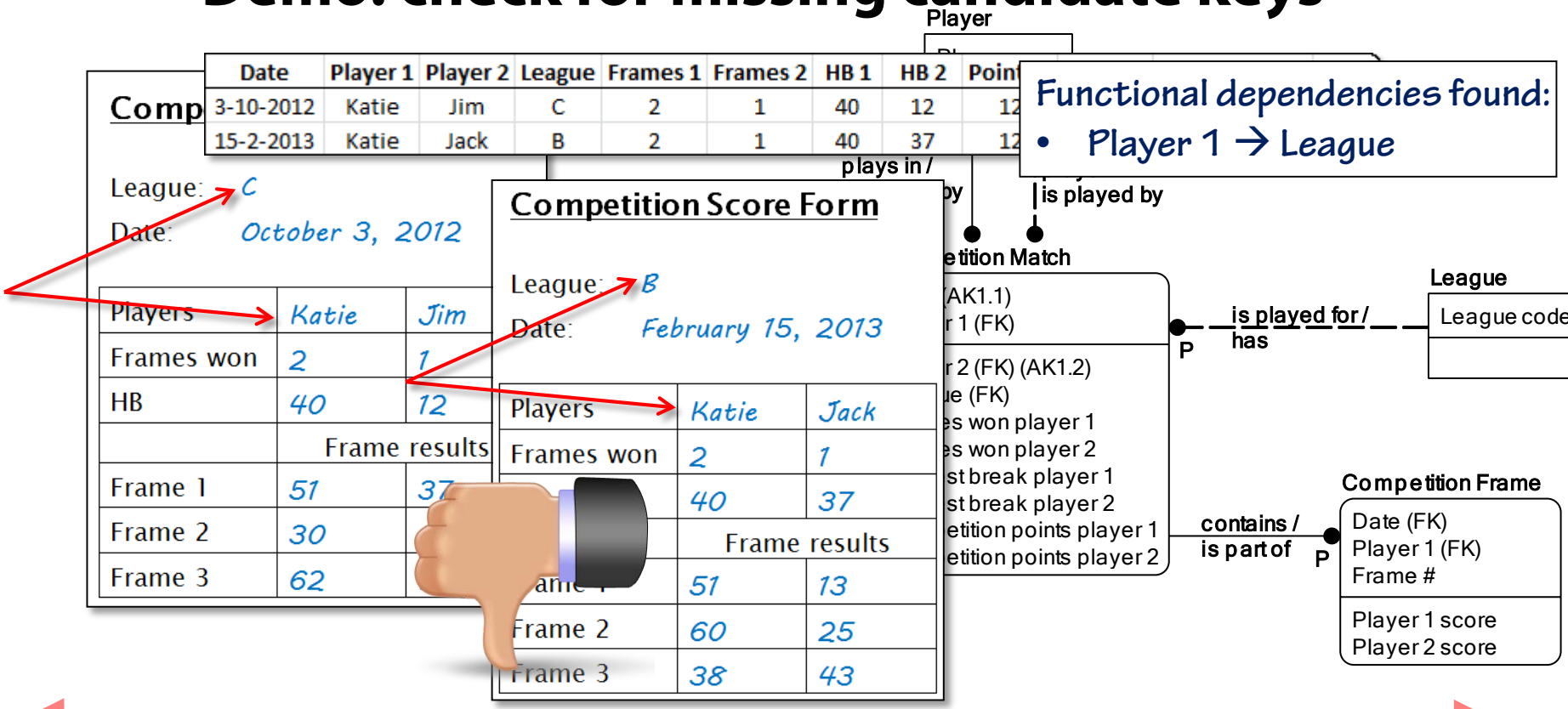
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2

Demo: check for missing candidate keys



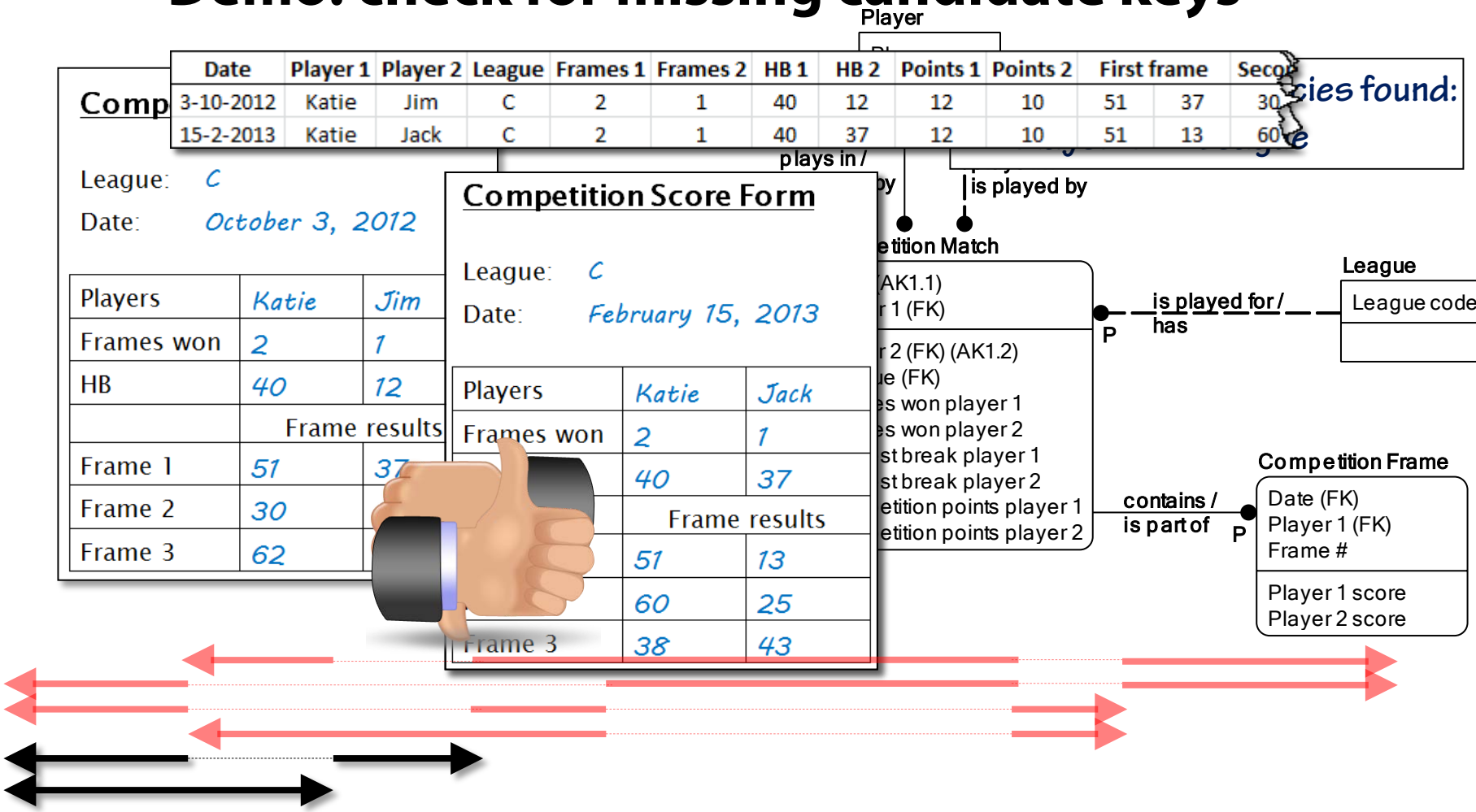
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2

Demo: check for missing candidate keys



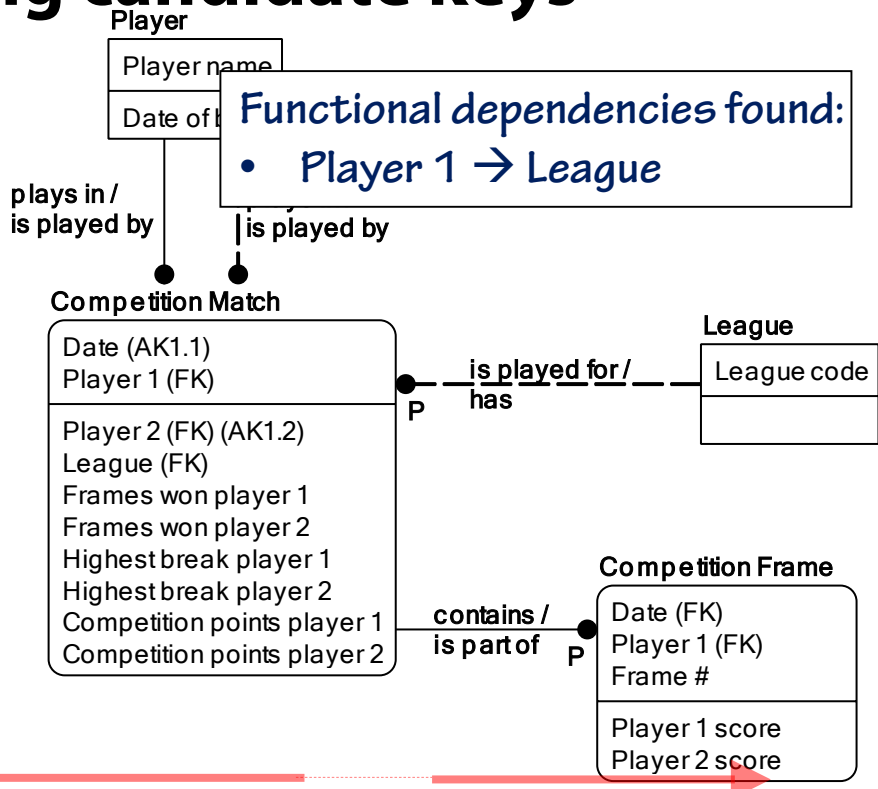
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-15	Katie	Jack	B	2	1	40	37	12	10

Demo: check for missing candidate keys



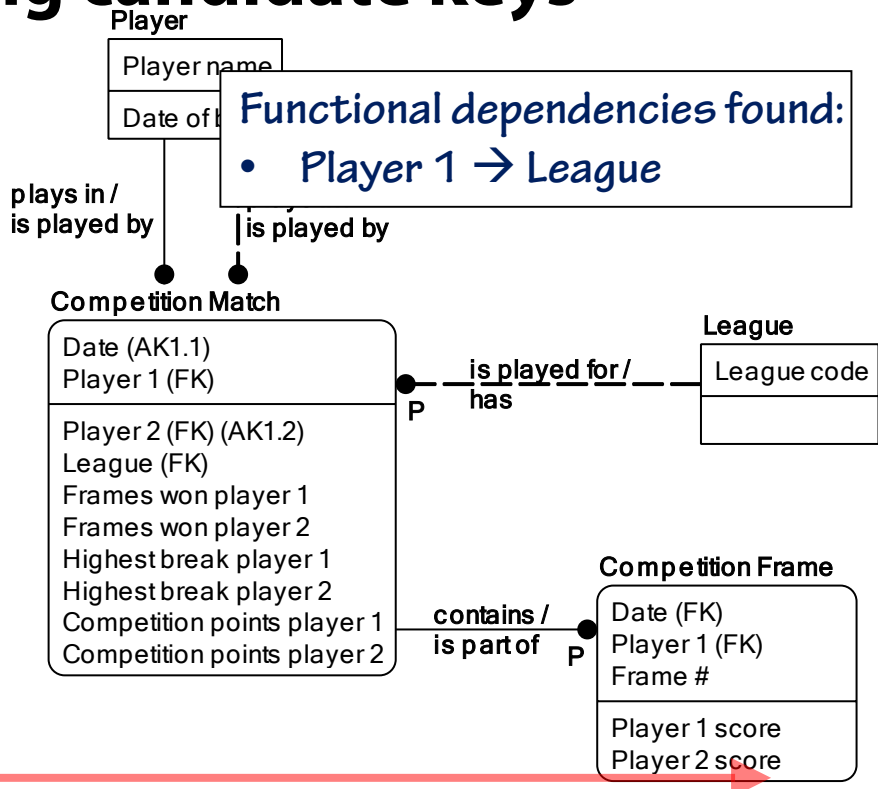
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-15	Katie	Jack	C	2	1	40	37	12	10

Demo: check for missing candidate keys



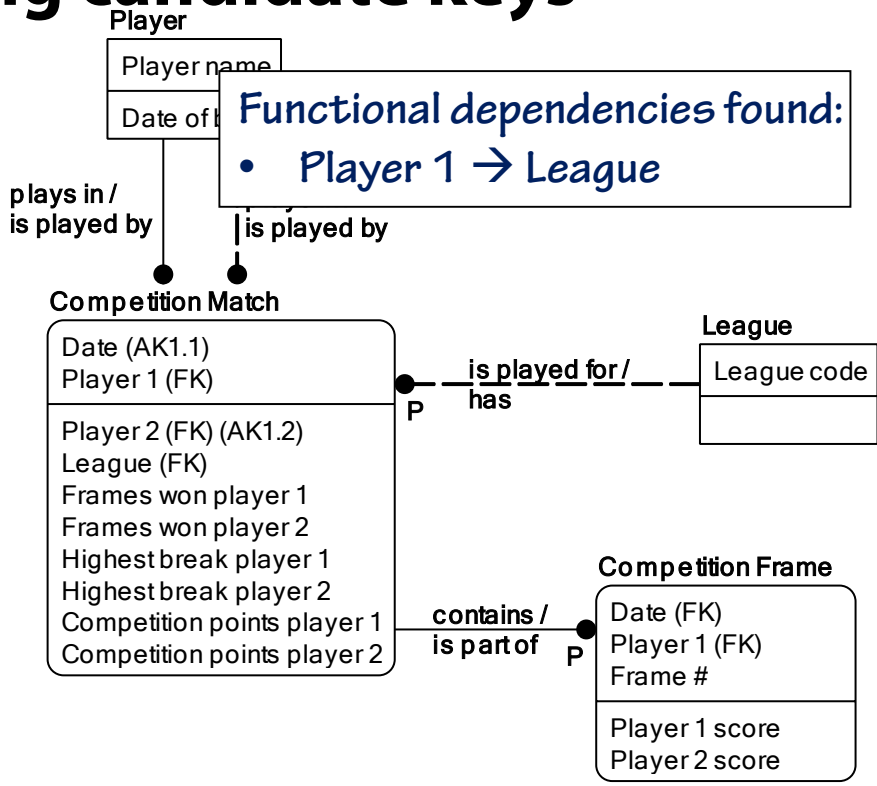
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-15	Katie	Jim	C	2	1	40	37	12	10

Demo: check for missing candidate keys



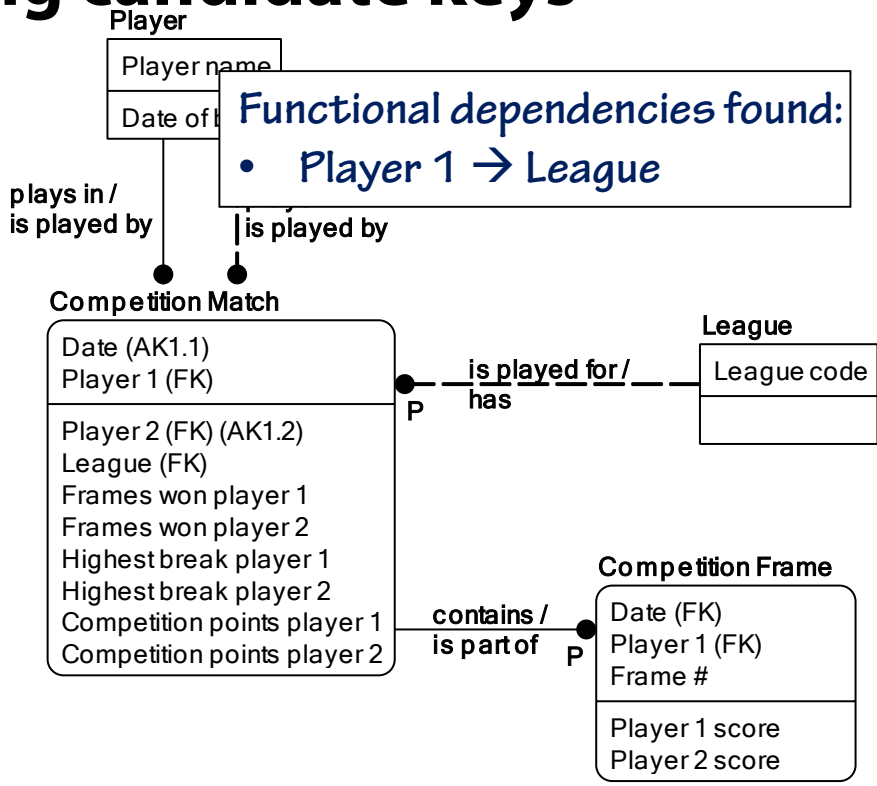
Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2013-02-15	Katie	Jim	C	2	1	40	12	12	10

Demo: check for missing candidate keys



Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2
2012-10-03	Katie	Jim	C	2	1	40	12	12	10
2012-10-03	Dave	Mary	C	2	1	40	12	12	10

Demo: check for missing candidate keys



Date	Player 1	Player 2	League	Frames 1	Frames 2	HB 1	HB 2	Points 1	Points 2

Summary

- **Problems**
  - Redundancy
  - Modification anomalies
  - Fixed by normalization
- **Functional dependencies**
- **First Normal Form**
  - No composite attributes
  - No repeating groups
    - In an attribute or by repeating the attribute
  - Candidate key