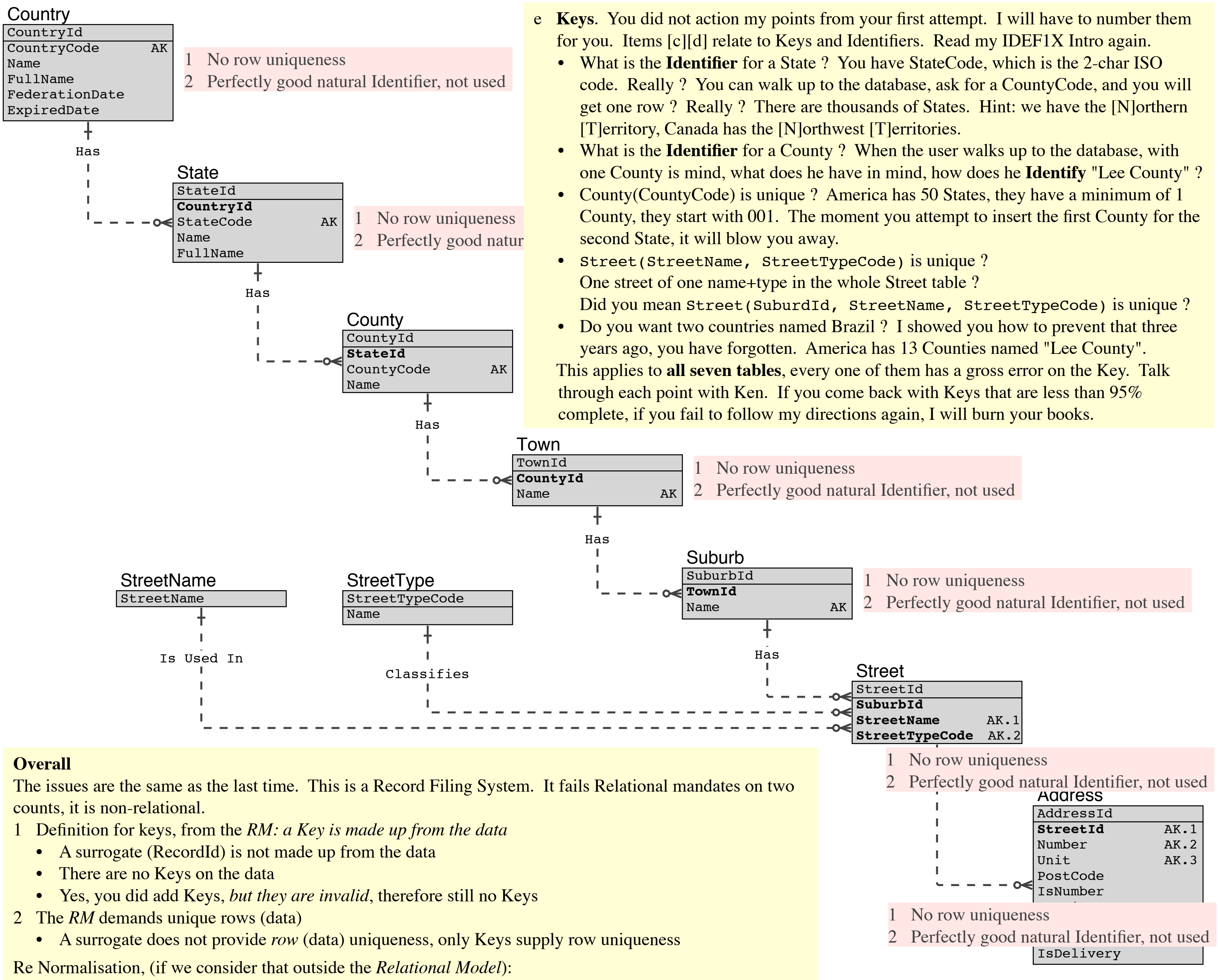
Relational Database 101

Address C

Status

There has been no [C] submission yet.
Previous submissions [B][A] are available on the following pages.

Address B



1 No row uniqueness
2 Perfectly good natural Identifier, not used

1 No row uniqueness
2 Perfectly good natur

e **Keys**. You did not action my points from your first attempt. I will have to number them for you. Items [c][d] relate to Keys and Identifiers. Read my IDEF1X Intro again.

- What is the **Identifier** for a State ? You have StateCode, which is the 2-char ISO code. Really ? You can walk up to the database, ask for a CountyCode, and you will get one row ? Really ? There are thousands of States. Hint: we have the [N]orthern [T]erritory, Canada has the [N]orthwest [T]erritories.
- What is the **Identifier** for a County ? When the user walks up to the database, with one County is mind, what does he have in mind, how does he **Identify** "Lee County" ?
- County(CountyCode) is unique ? America has 50 States, they have a minimum of 1 County, they start with 001. The moment you attempt to insert the first County for the second State, it will blow you away.
- `Street(StreetName, StreetTypeCode)` is unique ?
  One street of one name+type in the whole Street table ?
  Did you mean `Street(SuburdId, StreetName, StreetTypeCode)` is unique ?
- Do you want two countries named Brazil ? I showed you how to prevent that three years ago, you have forgotten. America has 13 Counties named "Lee County".

This applies to **all seven tables**, every one of them has a gross error on the Key. Talk through each point with Ken. If you come back with Keys that are less than 95% complete, if you fail to follow my directions again, I will burn your books.

1 No row uniqueness
2 Perfectly good natural Identifier, not used

1 No row uniqueness
2 Perfectly good natural Identifier, not used

1 No row uniqueness
2 Perfectly good natural Identifier, not used

1 No row uniqueness
2 Perfectly good natural Identifier, not used

Overall
The issues are the same as the last time. This is a Record Filing System. It fails Relational mandates on two counts, it is non-relational.

1 Definition for keys, from the *RM: a Key is made up from the data*
   - A surrogate (RecordId) is not made up from the data
   - There are no Keys on the data
   - Yes, you did add Keys, *but they are invalid*, therefore still no Keys
2 The *RM* demands unique rows (data)
   - A surrogate does not provide *row* (data) uniqueness, only Keys supply row uniqueness

Re Normalisation, (if we consider that outside the *Relational Model*):

3 It breaks Third Normal Form, because in each case, there is no Key for the data to be Functionally Dependent upon (it is dependent, yes, but on a pork sausage, outside the data, not on a Key).

This RFS is nowhere near ready for Normalisation, let alone Relational Normalisation.

```
    Address is uniquely Identified by ( AddressId )
    Address is uniquely Identified by ( StreetId, Number, Unit )

Country is I
    Country
    Country
    Country

County is In
    County
    County
    County h

State is Inc
    State is
    State is
    State ha

Street is In
    Street
    Street
    Street h

StreetName i
    StreetNa
    StreetNa

StreetType i
    StreetTy
    StreetTy

Suburb is In
    Suburb is uniquely Identified by ( SuburbId )
    Suburb is uniquely Identified by ( Name )
    Suburb has 0-n Streets

Town is Independent
    Town is uniquely Identified by ( TownId )
    Town is uniquely Identified by ( Name )
    Town has 0-n Suburbs
```

I hate to repeat, but nothing has changed.

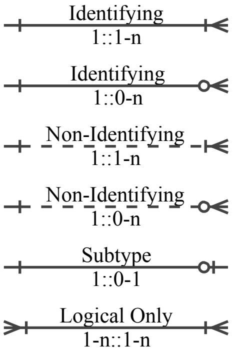
a These are idiotic Business Rules, they merely declare the Records in your Record Filing System. Of course, every File in a RFS is independent, but not so in a Relational Database. No better than last time, and last time, they were bad, bad, bad.

b I have told you one hundred times, if you start the design process by sticking RecordId on every box, you cripple yourself and the modelling exercise. Forget the rectangles with an ID. Start with circles, the way I showed all of you on the course. IDEF1X is a *Methodology*, not merely a notation. Follow it.

c Go and discuss this with the business, and determine the real **Identifiers**, the real Business Rules. Eg.
`State is Dependent on Country`
It is crazy to imagine a State that doesn't belong to a Country !

d Find out what the data is, what it means, how it relates to all other data in this cluster. Progress the circles on this basis. I would find those incomplete-but-correct circles more acceptable than finished rectangles that are broken.

IEEE Relation Notation



IDEF1X Notation

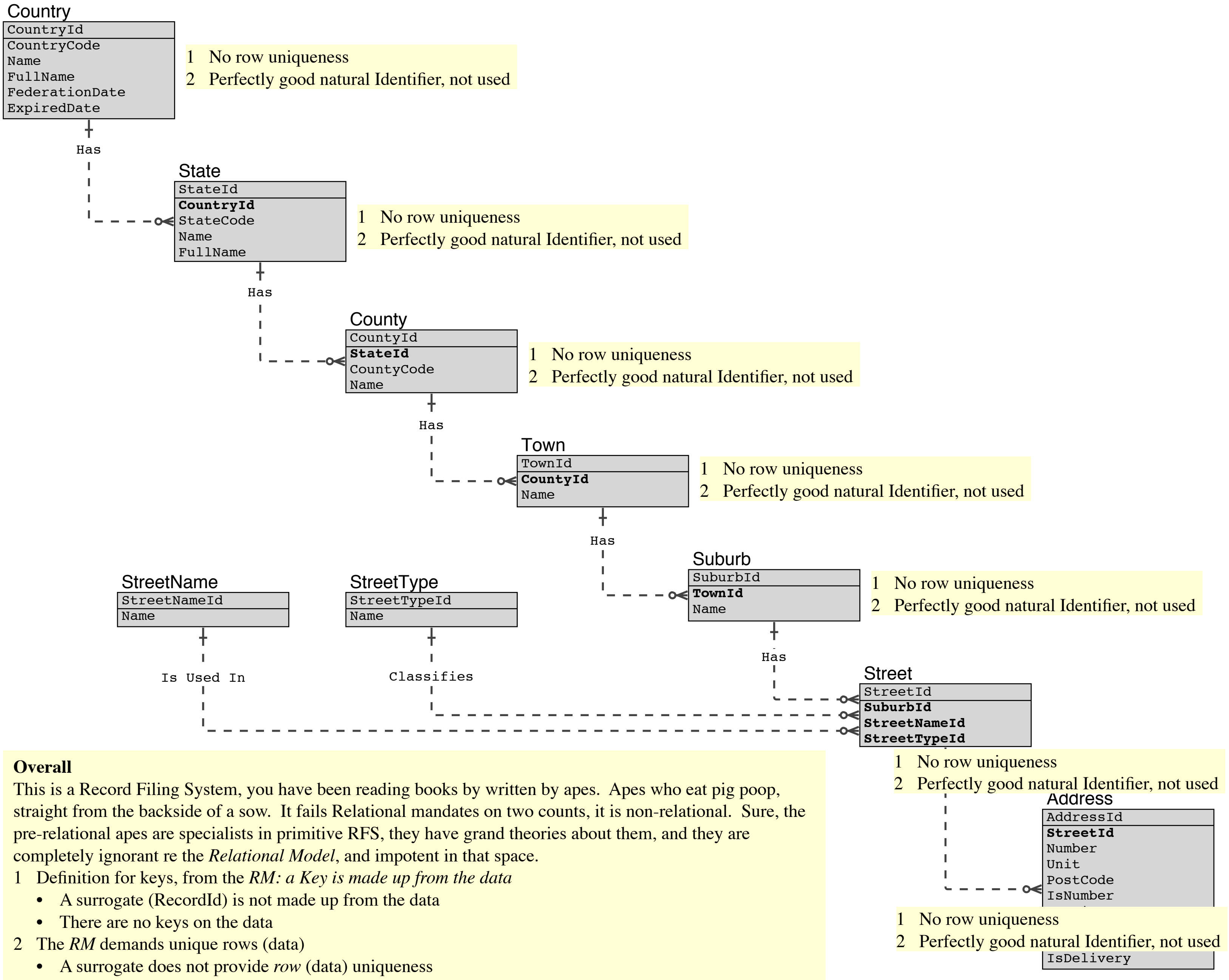
IDEF1X Method & Notation

Item	Definition	Display
Key		
Primary Key:	Unique	Above line
Alternate Key:	Other Unique key	AK
Inversion Entry:	Non-Unique index	IE
Entity		
Independent:	Exists without a parent	Square
Dependent:	Dependent on a parent	Round
Relation		
Identifying:	Parent PK forms child PK	Solid line
Non-Identifying:		Broken line
Parent PK is *always* a migrated FK in child		Bold

Erection: Developer

Relational Database 101

Address A



1 No row uniqueness
2 Perfectly good natural Identifier, not used

1 No row uniqueness
2 Perfectly good natural Identifier, not used

1 No row uniqueness
2 Perfectly good natural Identifier, not used

1 No row uniqueness
2 Perfectly good natural Identifier, not used

1 No row uniqueness
2 Perfectly good natural Identifier, not used

1 No row uniqueness
2 Perfectly good natural Identifier, not used

1 No row uniqueness
2 Perfectly good natural Identifier, not used

Overall

This is a Record Filing System, you have been reading books by written by apes. Apes who eat pig poop, straight from the backside of a sow. It fails Relational mandates on two counts, it is non-relational. Sure, the pre-relational apes are specialists in primitive RFS, they have grand theories about them, and they are completely ignorant re the *Relational Model*, and impotent in that space.

1 Definition for keys, from the *RM: a Key is made up from the data*
  - A surrogate (RecordId) is not made up from the data
  - There are no keys on the data

2 The *RM* demands unique rows (data)
  - A surrogate does not provide *row* (data) uniqueness

Re Normalisation, (if we consider that outside the *Relational Model*):

3 It breaks Third Normal Form, because in each case, there is no Key for the data to be Functionally Dependent upon (it is dependent, yes, but on a pork sausage, outside the data, not on a Key).

This RFS is nowhere near ready for Normalisation, let alone Relational Normalisation. Stop visiting the barn

Business Rule

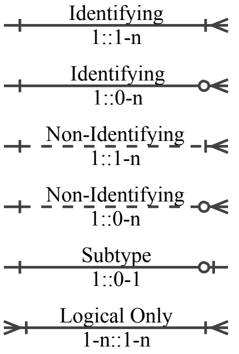
To assist those who don't know how to read an IDEF1X data model (R Brown 1985, after E F Codd), all Business Rules (predicates) implemented in the model are given in text form:

```
Address is Independent
    Address is uniquely Identified by ( AddressId )
Country is Independent
    Country is uniquely Identified by ( CountryId )
    Country has 0-n States
County is Independent

Stat

Stre

Stre

Stre

Suburb is Independent
    Suburb is uniquely Identified by ( SuburbId )
    Suburb has 0-n Streets
Town is Independent
    Town is uniquely Identified by ( TownId )
    Town has 0-n Suburbs
```

- These are idiotic Business Rules, they merely declare the Records in your Record Filing System. Of course, every File in a RFS is independent, but not so in a Relational Database.
- I have told you one hundred times, if you start the design process by sticking RecordId on every box, you cripple yourself and the modelling exercise.

  IDEF1X is a *Methodology*, not merely a notation. Follow it.
- Go and discuss with the business, and determine the real Identifiers, the real Business Rules.
- Find out what the data is, what it means, how it relates to all other data in this cluster.

IEEE Relation Notation



IDEF1X Notation

IDEF1X Method & Notation

Item	Definition	Display
Key		
Primary Key:	Unique	Above line
Alternate Key:	Other Unique key	AK
Inversion Entry:	Non-Unique index	IE
Entity		
Independent:	Exists without a parent	Square
Dependent:	Dependent on a parent	Round
Relation		
Identifying:	Parent PK forms child PK	Solid line
Non-Identifying:		Broken line
Parent PK is *always* a migrated FK in child		Bold

Erection: Developer