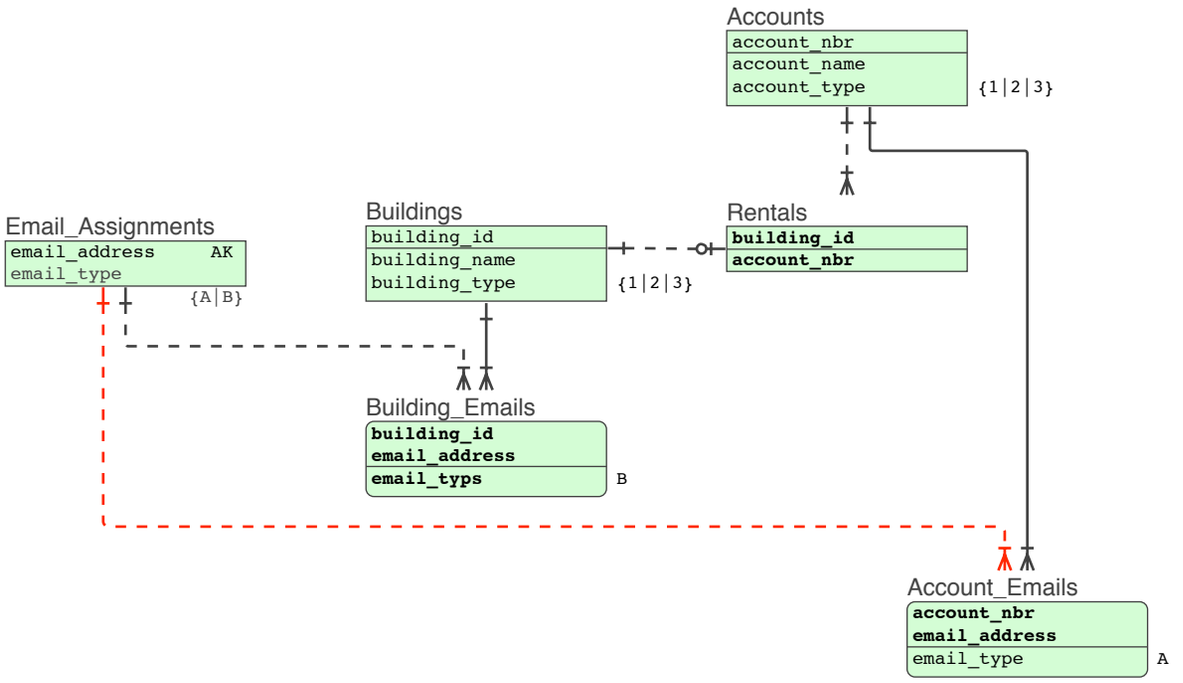
Celko Idiocy (Second Shot)



Note

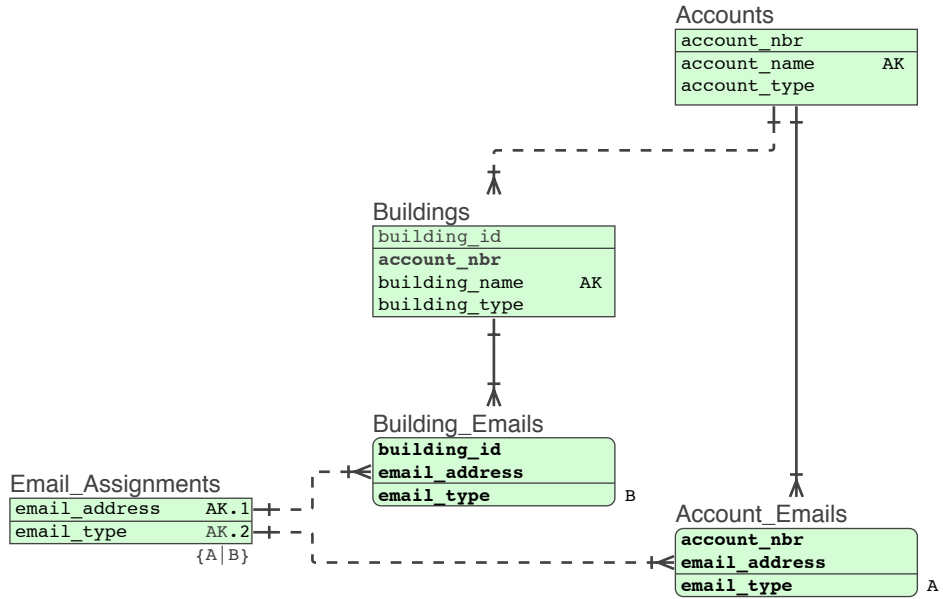
- Ok, he has fixed up the back-to-front issues, but the submission still fails
- Evidently, Celko does not understand SQL or Foreign Keys
  - his DDL (which he demands of others!) fails
  - Account_Emails FK (email_address, email_type) cannot Reference Email_Assignments (email_address): two columns vs one column
  - He has building_nbr and building_id mixed up (error not shown; I have used building_id for sanity)
- The requirement *Account has many Buildings* is not supported
  - Instead, he has a weird form via Rentals, where only Buildings that are Rented, are owned by an Account
  - Therefore un-rented Buildings exist without Accounts, in Disneyland perhaps
- The email_type in Building_Emails and Account_Emails is redundant: we already know what type it is, simply by the fact that it is in the table it is in.
- As usual, he is trying weird functions via indices, without understanding either indices or relationships, which would provide the requirementin a straight-forward manner
  - *"The UNIQUE is how you enforce a 1:M relationship with accounts and buildings"* is hilarious. Er, no, we don't.
  - In SQL we do that with a relationship, a Foreign Key in Building and some index (account_nbr, building_id),
- The index (account_nbr, building_id) or (building_id, account_nbr) is missing. The correct location for that is of course Building, not Rental.
- The PK in Email_Assignments is side-splitting. Listen carefully.
  - Since email_address is unique, email_address plus *anything* will be unique
  - it cannot be made more or less unique by adding email_type and another index
  - Therefore the PK and second index are 100% superfluous
  - Therefore it can be removed
  - However, that requires formal implementation of the Subtype structure
- Last, but not least, the requirement (implicit, ala Account xor Build Email) is that an email can only be used once, in one Building xor one Account
  - That is not supported

Famous Mistakes re Idiot "keys"

- Additionally, Celko makes the usual classic mistakes, which people using Idiot RowIds make, over and over again ... no amount of pointing that out seems to get through to them:
  - The RowId column (eg. building_id) does not uniquely identify a row
  - An index on whatever uniquely identifies a row is required, eg (building_name). That is missing.
  - Therefore building_nbr and its index is 100% redundant, and can be removed
- Likewise for Account

Celko Corrected

1. Let's assume Celko has a capable DBA who understands the tomfoolery he is trying; has lots of patience; makes appropriate corrections; and clarifies the confused mess.
2. He realises that Celko does not understand row uniqueness required for relational tables, and he fixes that as well.

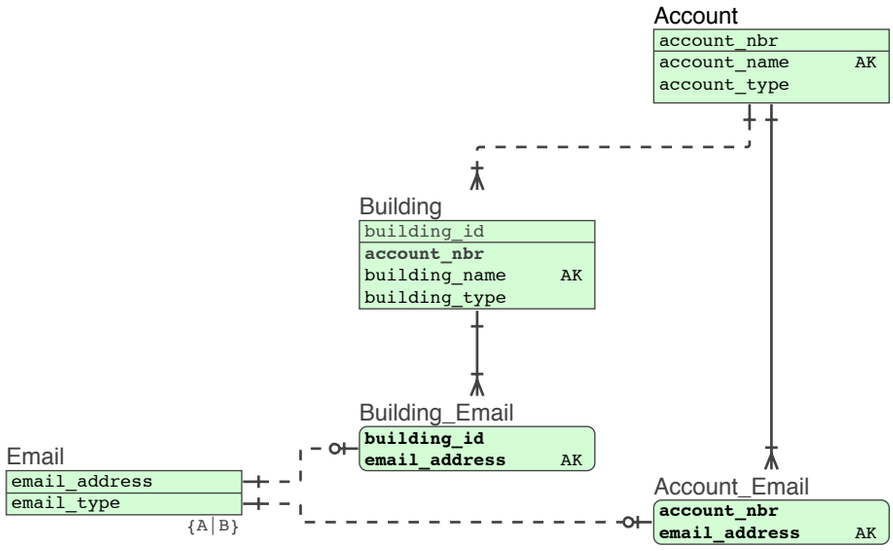


Sub-standard
Negative Performance

- That "works", but with redundancies and extra indices, that the formal structure does not suffer

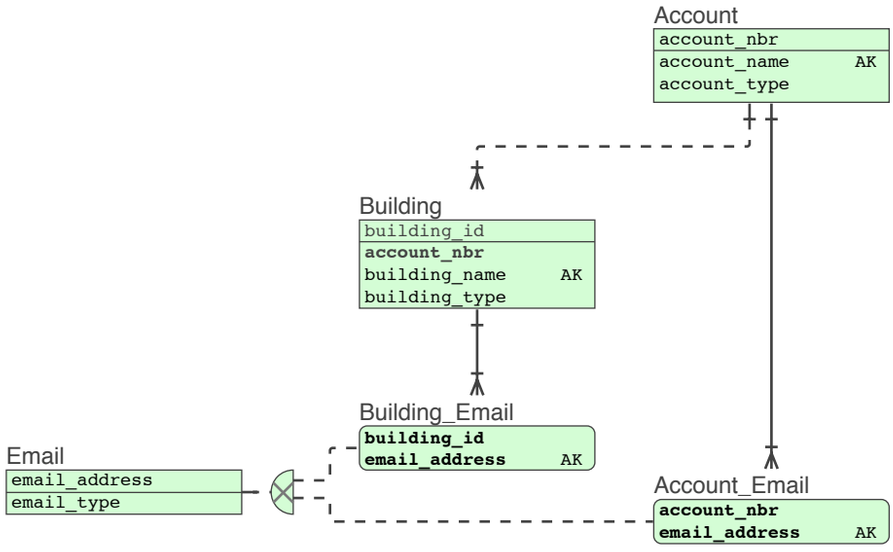
Progression to Standard

3 Next, the DBA realises that the redundant columns and extra indices are intended to support a Subtype structure, which if implemented correctly, eliminates the redundant columns and extra index, so he decides to implement it
(In order to provide the progression, I will show it in incremets)
4 Let's correct the sub-standard naming

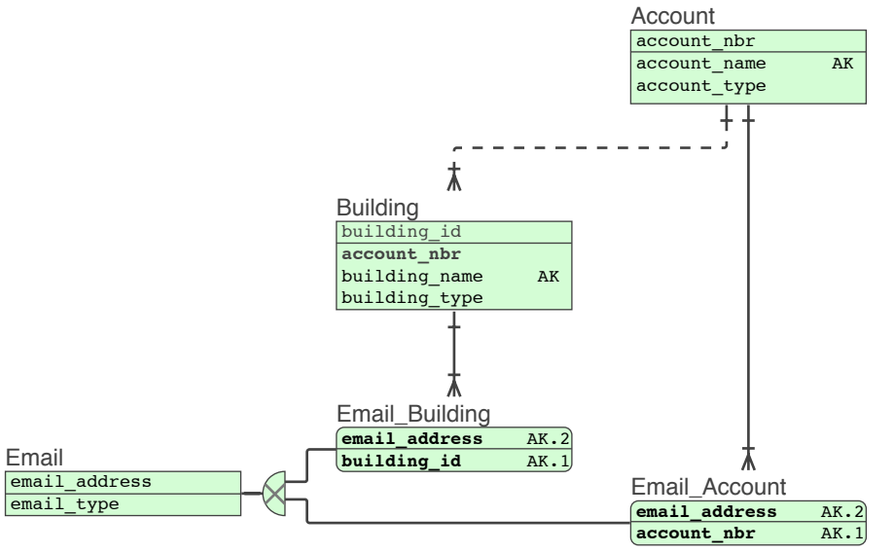


5 The Exclusivity (which [1] has with extra indices, and [3] does not have) is missing, so let's implement it .
- And yes, it is all Declarative Constraints

6 When we do that, the PKs for Building_Email and Account_Email with be reversed, because the Subtype structure now becomes primary, so le't look at the half-way point of that progression. Stated another way, the order of the indices are reversed, with no other changes



7 Now we have the formal structure, which supplies exactly the same referential integrity as the obese sub-standard structure, minus the extra columns and indices.
8 But the tables names do not reflect the primary relationship and subordination, so let's correct that.



9 Now if the Account and Building Idiot "keys" were elevated to Relational Keys, the result would be my submission: **Subtype Basics**

