MTA SZTAKI Közlemények 37/1987 pp- 151-170

A RESTRICTED DESIGN METHODOLOGY TO ALLOW TESTING FOR BCNF IN POLYNOMIAL TIME

F. N. SPRINGSTEEL

University of Missouri at Columbia
Department of Computer Science
Columbia, Missouri, 65211 USA

ABSTRACT

Boyce-Codd Normal Form (BCNF) is a well-known condition on relational database schemas that implies some desirable properties, and is known to prevent some very undesirable "anomalies" from occurring in the use of the database. It is thus important to be able to test a proposed database design for this condition.

This investigation of logical database design methods asks whether there are any conditions on a relational database schema such that, although they are not equivalent to BCNF, do guarantee that this desirable property can be detected in polynomial time (P-time). Conditions equivalent to BCNF are known to be intractable to test, but the conditions we give here are either sufficient for BCNF or else enable its being tested in P-time, and the conditions themselves are likewise testable. They are also desirable for certain design reasons.

While the first set of conditions only logically imply BCNF, they provide a setting (regular entity-relationship diagrams) which helped to suggest the second set of conditions, in which it is easier to test for this normal form than in general databases. Also, we argue, this setting is valuable in its own right, and can lead to a useful, if not universal, database design methodology. This methodology works for many databases expressible by E-R diagrams which have a "database key".

I. INTRODUCTION AND PRELIMINARIES

A. THE PROBLEM AND THE APPROACHES

It is clearly futile to search for conditions that are equivalent to BCNF, because it is an NP-complete problem to test relations for BCNF and conditions very similar to it, in general relational databases [BB79; JF; Osb]. Only very strong conditions seem to imply BCNF, e.g. 4NF,PJNF [LePa], and Berge-acyclicity [JNS83a].

However, it is possible that certain conditions related to "BCNF-ness" are so akin to this normal form that it can be tested in their presence, in tractable time. Indeed we find that this is the case; there are arguably quite reasonable, realistic conditions that one might like to see in the schema design normally, fitting the above description.

For now we shall only describe one set of conditions in intuitive terms : later on we will give their formal descriptions. These conditions model a type of very well designed database: there is one master-file, S, which contains either a key or a determiner of a key for each of the other files. These other files relate either one key to its directly dependent attributes or else relate several equivalent keys, possibly of different entities. We refer to databases of this type as fitting our "master-file" scenario.

In the next part of this first section we review the special concepts needed for our approach, including the definition of "regular" for an entity-relationship diagram (ERD). We assume that the reader is familiar with standard relational database theory, including the concepts of functional dependencies, keys, and schema [Ulm82a].

In Section II we discuss conditions that are sufficient to imply BCNF for relation schemes in the canonical relational schema |R of a regular ERD |E. Certain of these schemes are automatically in BCNF in such |R, viz., those of entity sets, of purely "one-related" relationships, and of binary relationships. Here we give one condition sufficient for BCNF for all other schemes in |R, that the underlying regular ERD also be "loop-free".

In the third section we shall explain the "master-file approach" formally, and show that it enables one to test the BCNF-ness of the particular databases that can be defined to conform with it. This "restrictive design methodology" is then discussed in the last section.

B. ENTITY-RELATIONSHIP PRELIMINARIES

Until recently most authors have used E-R diagrams as concise and intuitive database design indicators, using entities and relationships as a "lingua franca" of data model theory. ERDs act as interfaces between the various conceptual models in the database literature. For example, in the 1982 ACM Symposium on Principles of Database Systems both the opponents [AtPa] and the proponent [Ulm82b] of the "universal relation view" used informal ERD's in their arguments.

We have seen logical analysis of the E-R model reach a new level, that of rigorous and careful treatment of many of the basic assumptions and definitions behind this "jacknife" of the trade [Ch76,80; JNS83a,b,c]. Not only do is this useful to help an understanding of the entity-relation model, but also the newer results have bearing on many issues of database theory. There is now an impetus to study its formalizable aspects with the same care that has been applied to the relational model. As a basic text in database systems [Ulm82a] explains, there is a natural representation of the E-R model in terms of either the relational or network or hierarchical data models.

The conversion of an ERD into a relational model, using one associated relation scheme for each entity or relationship, has become a standard, accepted method [Ulm82a]. We extend this known conversion one step farther, to ask: What do the given quantifying marks on the relation/entity connectors in the E-R diagram imply that we know about functional dependencies (FDs) in its canonical relational schema, of all the associated schemes? We feel that the proper interpretation of these marks, in direct accordance with standard mathematical thinking about many-to-one functions, leads inevitably to the definition of a "regular" ERD and of its "basic" (or fundamental) FDs, as seen below.

Example 0: Consider Figure 0, containing entity sets, EMPL and DEPT, and the relationship set EMPL_DEPT. These entity sets are converted into relation schemes, with the same names, as follows:

EMPL(E#,EN,JC); DEPT(D#,MGR,LOC).

Note that the respective keys are E# and D#, called the primary entity keys, determining the other attributes in their relation schemes, as seen by the quantifying mark 'n' on their connectors, and '1' on the dependent attributes. Hence, in the relationship scheme EMPL_DEPT we need not repeat the dependent attributes of these entities, but only give these keys plus any attributes directly dependent on the relation, here TASK and STart_DATE:

EMPL_DEPT(E#,D#,TASK,ST_DATE).

Note that the attributes EN,JC,MGR,LOC,TASK and ST_DATE are in fact all functionally determined by the key E# of EMPL, which is thus a key of the relation scheme EMPL_DEPT.

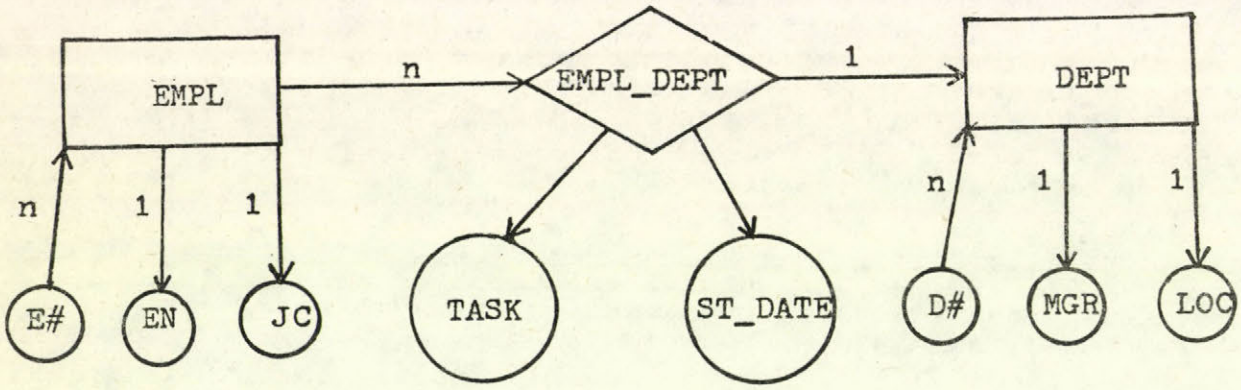

FIGURE 0. A REGULAR ERD

The above dependency analysis can be seen to be correct from the 'n' and '1' (partial) functional indicators on the arrows; i.e., we consistently mark all connectors in an ERD with indicators of the connected entity's functional participation in the affected relationship, whenever this information is known. Thus, by following the arrows, from n- to 1-related attributes, the ERD deducible functional dependencies (FDs) can be written down!

Two cautions are in order: a) to employ this convenience, we disallow any special types of attributes, e.g. those whose existence depends on other entities, which would need special designations; b) the device of "following the arrows" needs interpretation for relationships of three or more entities.

However, for normal entity schemes or for binary relationships as above our convention is clear. Thus, since the keys can be seen by inspection, it follows that the "basic FDs" of the ERD of Figure 0 are these:

E# -> EN, E# -> JC for the schema EMPL;

D# -> MGR, D# -> LOC for the schema DEPT; and

E# -> D#, E# -> TASK, E# -> ST_DATE for schema EMPL_DEPT.

Note that if we did NOT know the n/1 functionality in this last relationship, then we could only assume an arbitrary many-related indicator 'n' on both connectors between EMPL and DEPT, with the consequence being in that case: the only deducible key (from this information) would be (E#,D#). Also, if an k-ary relation has two or more entities (say P1 thru Ps) with 'n' indicators, and

all others marked '1', then the indicated key is P1....Ps. This makes sense if one thinks of the arrows as showing the many-one functionality from the several n-related arguments, Pi, to the other attributes, a unique tuple of values of which is determined by fixing an s-tuple of values for the key.

We shall call all relationship schemas with at least one n-related entity n-RELs: if they have any unmarked entities ('n', by default) we also call them n-RELs. In either case the indicated key would be the union of all the n-related entities' primary keys. In practice an "all n" case is rare. Less rare is the case of all entities being mutually determining keys, i.e. Ei -> Ej, all i and j. When this happens we call the relationship schema 1-REL, because all its Ei should be marked '1'.

RULES FOR THE BASIC FDs GIVEN BY A (REGULAR) E-R DIAGRAM:

For each entity set relation scheme ENT(PK, A1, ...,Ae), where PK is the pre-chosen primary key, we have these 'basic' FD's

RULE 1: F(ENT) = {PK -> Ak : 1<=k<=e.}

For each 1-REL relation scheme, the set of 'basic' FD's is

RULE 2: F(1-REL) = {Ei -> Aj : Ei 1-related, Aj any attribute}.

Finally, for any n-REL relation scheme, where P1,...,Ps are the primary keys of the n-related entities, the 'basic' FD's are

RULE 3: F(n-REL) = {P1...Ps -> Ai : Ai any other attribute}.

The "canonical relational (database) schema" of an ERD is the set of all its associated relation schemes. E.g., for Example O, it can be denoted !R = { EMPL, DEPT; EMPL_DEPT }.

DEFINITION: If an ERD has only relationship schemes of the types 1-REL and/or of the type n-REL, with the basic FDs for each of its entity and relationship schemes as defined above in Rules 1 - 3, then we call the ERD, and its canonical relational schema !R, <u>regular</u>.

NOTE: Since all these rule-given FD's are the only ones clearly visible from the ERD itself, as the database design, and because the ERD is presumed to have clear semantics, the set of all the basic FDs is assumed to be a cover for all (the known) FDs of !R. This assumption is implicit in the definition of "regular" ERD.

Clearly, the ERD of Figure O, with the FDs we have deduced from it as above, is a regular ERD.

DEFINITION: A relation schema R is said to be in Boyce-Codd Normal Form (BCNF) with respect to a set F of FDs if, for any FD X -> A embedded in R, either A is in X (i.e., this FD is trivial) or else X contains a key of R (with respect to the set F). We also say that a relational schema <IR, IF> is in BCNF, ("globally") where IR is a set of relation schemes <Ri, Fi> with FD sets Fi over Ri, if each Ri is in BCNF with respect to all the FDs derivable from IF = Ui Fi. We informally say that a regular ERD IE is in "BCNF" iff its canonical relational schema <IR, IF> is, where IF is the family of all its basic FDs.

As examples showing the range of such conditions, we exhibit an (upper conceptual domain of an) ERD which is not in BCNF and a larger diagram which is.

Example 1. In this ERD of an Employee/Department/Project relationship, each EMPL can work in more than one DEPT, and only the combination of DEPT and EMPL determines the Project now being worked on by that employee in that department. Each Project is wholly contained in a single Department. The ERD is NOT in BCNF, because PROJ -> DEPT is a violating FD.

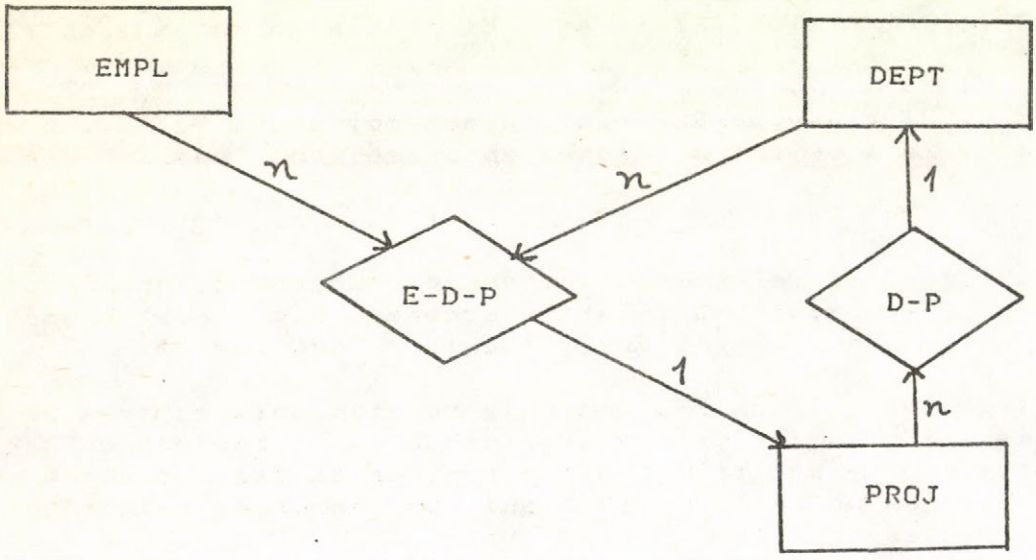

FIGURE 1. Regular ERD not in BCNF

Example 2. This is from a real-world case study of a business enterprise's accounting functions, of a complicated and very inter-related nature [Ulm82b]. (The sample subdiagram shown here represents less than a third of the original!) With applications of this complexity being common, some very realistic databases

can be seen to be in BCNF, as long as they are well structured. In this case the schema is in BCNF because all relationships are in fact binary; cf. Proposition 1 of Section II.

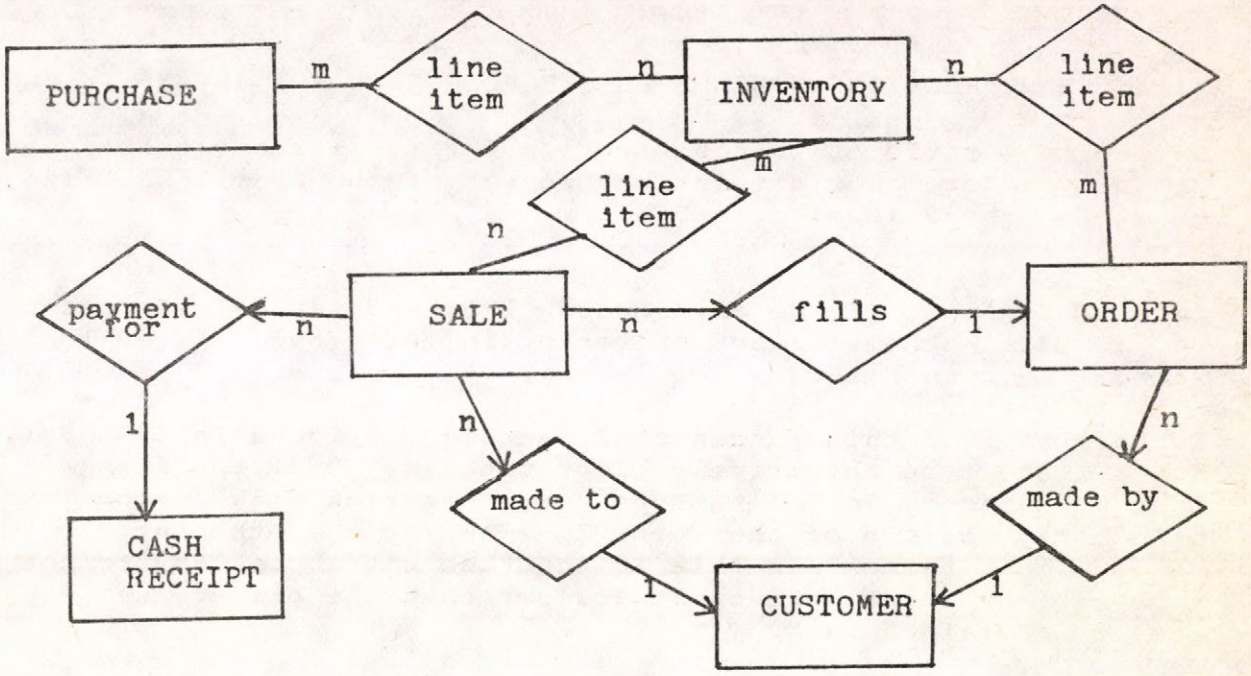

FIGURE 2. Enterprise's schema, in BCNF

II. SUFFICIENT CONDITIONS FOR BCNF

In this section we shall describe some general conditions that are sufficient to imply BCNF. This preface to our Section III discussion of the scenario is needed for two purposes: a) to help orient the reader to terminology used later, and b) to show the power of certain general conditions by exhibiting their desirable consequence when assumed as a "package", namely BCNF.

We assume that the reader is familiar with the principles of database dependencies, as found in [Ulm82a]. We shall adopt the notation and terminology there. Below we shall assume that |E is a regular ERD whose canonical relational scheme can be denoted

|R = (S1, S2, ..., Sn), and that these n relation schemes have their basic sets of FDs, as defined in Section I, denoted F1, F2, ..., Fn. Let the set of all basic FDs of |E be denoted

F(|E) = F1 U F2 U ...U Fn. By the nature of Rules 1-3, each FD K -> A in an Fi must enjoy these properties:

1. be embedded in an Si: KA $\subseteq$ Si;

2. have in K only primary key attributes of entity sets;

3. have a single attribute of Si as its right-hand side A;

4. have a key (or superkey) of Si as its left-hand side K.

A. MANY RELATION SCHEMES FOR REGULAR ERDs ARE ALWAYS IN BCNF

Indeed, the title of this subsection is true for any entity set relation scheme. For a relationship-set of the type 1-REL, its relation scheme is also in BCNF.

THEOREM 1. LET |E be any regular ERD, and let ENT represent the relation scheme of an entity-set in |E. Then ENT is in BCNF with respect to F(|E).

Formal proof of this fundamental fact can be found in [JNS83a]. But it should be intuitively clear that any FD of the form X -> A that is embedded in ENT must be derivable from F(ENT) alone, which only has FDs of the form PK -> Ak, Ak any other attribute of ENT. One reason for this is that the attributes Ak are not found in any other scheme. It follows that the primary key PK must be contained in X.

There is little syntactic distinction, at root, between ENT relation schemes with different, equivalent candidate keys and the relation schemes of type 1-REL. Often, the choice of entity-set or 1-related relationship to represent an "object" like Department, is arbitrary. This suggests that Theorem 1 may be extendable to relation schemes of 1-REL type. We find this to be the case.

THEOREM 2. Let REL be a relationship-set relation scheme, in a regular ERD |E, which has entity-sets E1, ..., Et all 1-related. Then REL is in BCNF with respect to F(|E).

Again, a formal proof can be found in [JNS 83a], based on the simple observation that if X -> A is in any nontrivial FD true in REL, then by regularity it must be derivable from F(REL). Thus, X must contain the primary key of some Ej, $1 <= j <= t$, any of which is a key of REL.

By exhaustive consideration of the cases of keys for any binary relationship R(A, B), we can also conclude the following proposition, which is the basis for the claim that Example 2, despite its complex inter-connections, in in BCNF. In the "worst case", where R is a many-to-many relation, even if some external

connections imply the FD A -> B, we simply conclude that the key A of R must be included on the left-side of any non-trivial FDs derivable from F(R). (Another FD B -> A means B is also a key.)

PROPOSITION 1: If REL is a binary relationship in a regular ERD |E, then REL is in BCNF with respect to F(|E).

B. LOOP-FREE ERDs HAVE ALL RELATION SCHEMAS IN BCNF

The relationship-set relation schema E_D_P (EMPL, DEPT, PROJ) of Example 1 is not in BCNF, because by Rule 3 applied to D_P(DEPT, PROJ), the FD PROJ -> DEPT becomes embedded in the schema E_D_P, where PROJ is NOT a key. We seek a condition to rule out such cases. Note that there is no way to decompose E_D_P into two binary relationships without losing some semantic information.

We conclude that the problem in Example 1 arises from the fact that this ERD contains a loop involving at least two distinct relationship sets. Below we define formally what we mean by "loop" and "loop-free" for an arbitrary regular ERD. Note that, as in this example, it suffices to consider only the entites and relationships, the "upper conceptual domain" of the ERD, rather than the low-level attributes, because the latter do not affect the existence of loops OR of BCNF-violating FDs in the ERD.

DEFINITION: Let the upper conceptual domain U(|E) of an ERD |E be identified as the hypergraph whose nodes are all the entity sets Ei of |E and whose (hyper)edges are all the relationship sets Rj of |E. Each Rj as an edge is the set of distinct entity sets related by Rj. (This notion of possibly non-binary edges generalizes the usual edge notion in graphs [Berg].)

Suppose E and E' denote entity sets in |E. A path from E to E' is a sequence of distinct relationships R1, ..., Rs such that:
E is in R1, E' is in Rs, and Ri $\cap$ Ri+1 $\neq \emptyset$, 1<=i<=s.
The length of the above path is s. A loop based at E is a path from E to E of length at least two. We shall call |E loop-free if there does not exist a loop based at any node in U(|E).

PROPOSITION 2: An ERD |E is loop-free if and only if there is no sequence in U(|E) of the form

(R1, E1, R2, E2, ..., Rs, Es, Rs+1), where

1. E1, E2, ..., Es are distinct entity sets;
2. R1, R2, ..., Rs are distinct relationships;
3. s is at least two, and Rs+1 = R1;
4. Ei is in (and so related by) both Ri and Ri+1, 1<=i<=s.

This Proposition shows that IE is loop-free if and only if its hypergraph is Berge-acyclic [Berg], equivalent here to 1-4, and its proof follows directly from the definitions.

THEOREM 3: Let IE be a regular ERD that is also loop-free. Then every relation scheme S in the canonical relational schema IR of IE is in BCNF with respect to F(IE).

[The formal proof of Theorem 3 is too long to include here, but is based on the intuitive idea that if no loops exist in the ERD, then no "externally implied" FDs can become embedded even in a non-binary, n-REL type relation scheme S. Notice that we already know, by Theorems 1 and 2 and Proposition 1, that all other types of relation schemes in IR are in BCNF with respect to F(IE).]

Example 3: The ERD in Figure 3 is the upper domain of the COMPUCO database (from the manual of a popular DBMS), and it is clearly loop-free. If we assume it is also regular, then we can write down the important, inter-entity FDs simply by following Rules 1-3. However, even before seeing the FDs, we know by Theorem 3 that none of them will violate the BCNF conditions for the relationship TRANSAX, which records daily transactions involving customers, vended products, and sales representatives of a value-added-reseller, where each product contains a certain type of computer. For example, we see that the key transactid of TRANSAX cannot be determined by any combination of the primary entity keys: empid, custid, prodid.

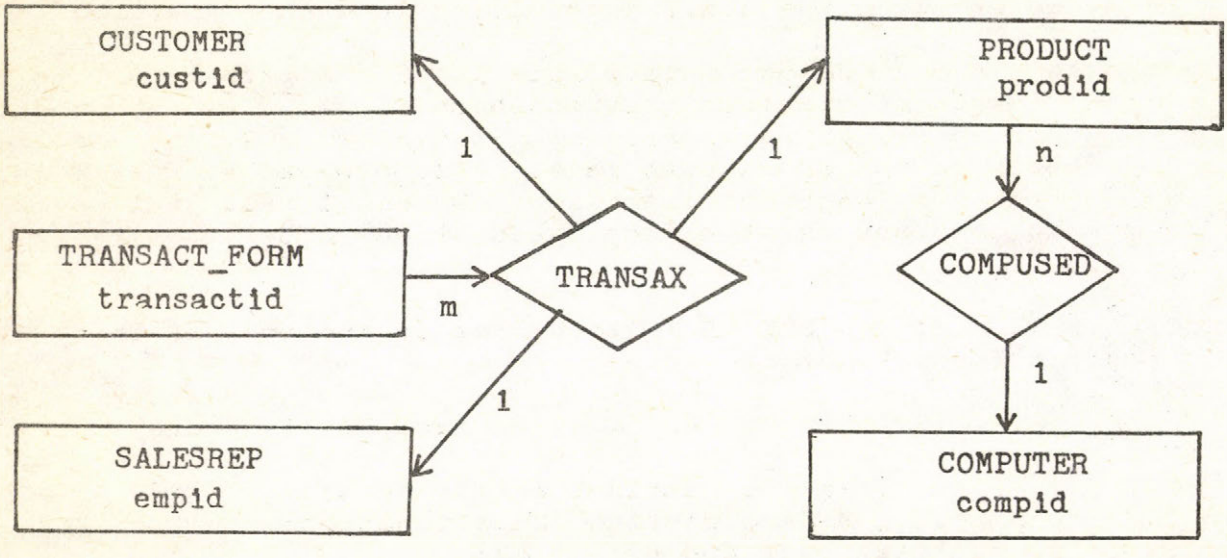

FIGURE 3. The upper ERD of the COMPUCO database.

NOTE: Although Theorem 3 can sometimes be used to conclude that an entire database schema will be in BCNF, even before all the FDs are explicitly seen, the loop-free condition is very much stronger than the BCNF condition. That is, the former is not a necessary condition for the latter, as the following example shows. Also, cf. Example 2, which is in BCNF by Proposition 1, but contains many loops.

Example 4: The entities are Contract, Supplier, Part, and pRice. The parts and their prices are dependent by bid on the contract, but not on the supplier. There is only one supplier for each contract, but a supplier can charge different prices on different contracts. The explicit keys are underlined:

$$\mathrm{IR} = [\ \underline{\mathrm{CS}}\mathrm{PR};\ \underline{\mathrm{CS}};\ \underline{\mathrm{CP}}:\ \underline{\mathrm{CR}};\ \underline{\mathrm{SR}}\ ].$$

Theorem 3 does not apply to this IR, because its easily diagrammed U(IE) has loops, but it is clearly in BCNF.

PROPOSITION 3: There are P-time algorithms to determine whether any ERD IE is: (1) regular, for a given set IF of FDs, or (2) loop-free.

Proof: One can test regularity, given the sets IF and F(IE) of FDs, by testing if $F(\mathrm{IE})^+ = \mathrm{IF}$ by use of the closure-membership algorithm [Ulm82b], in time linear in the product of the sizes of the sets. By Proposition 2, the loop-free test can be done by standard algorithms for cycle-checking in P-time; cf. [Fag81].[]

III. SPECIAL CONDITIONS MAKING BCNF P-TIME TESTABLE

A. THE MASTER-FILE SCENARIO FORMALIZED

Many modest-sized databases have regular ERDs with canonical relational schemas that exemplify the "unique masterfile" scenario, mentioned in the Introduction. For convenience we continue to study their ERD models here via the simplified canonical relational schema which ignores the lower conceptual domains' proper parts: dependent attributes of entity and relationship sets. That is, we consider only the entities - as basic units - and the relationships between them, the "upper conceptual domain", in order to focus on the open question of testing BCNF. We can do so without loss of generality because the BCNF conditions can only be violated in regular ERD's by certain functional dependencies between entities. i.e. between their primary key attributes. Thus, entities are here represented solely by primary keys, the only attributes kept.

For our real-world-oriented class of ERDs we can formalize the masterfile scenario as follows, assuming regularity. The class includes all ERDs having canonical relational schemas of form:

(#) $IR = (E1,\ldots,Em: R1,\ldots,Rn: S)$, where:

a. the Ei are all the entity set relation schemes;

b. S is the only relationship scheme which can relate one or more m-related entities, except for the binary relationships;

c. the Ri are the other relationships' schemes, either binary relationships (possibly n-to-1, n-to-n, or 1-to-1) or else "equivalence" relationships where all entities are 1-related: 1-RELs.

d. furthermore, the m-related entities of S, say P1,...,Ps, include a key or a determiner of a key for each entity set Ej, i.e., P1...Ps -> Ej, $1<=j<=m$. Thus, S contains a "database key" [AtPa].

While another type of Ri would be logically consistent, one that has only one n-related entity and two or more 1-related entities, for simplicity we assume this type decomposed into two or more binary relationships, without losing generality. This is a valid simplification when the ERD is regular. For example, the relationship R(ABC) with FD's A -> B and A -> C for A n-related and B,C 1-related. By standard methods, R(ABC) is decomposable into R'(AB) and R"(AC), losslessly, since A = AB ^ AC -> BC [Ulm82a].

B. TRANSFORMATION METHOD: UNIT FDs and KEY EQUIVALENCES

The masterfile scenario (#) desribed above can be subjected to the analysis of a transformational method that may reduce its set IK of key dependencies to a "linear" set (of unit FDs), whose full set of keys can be discovered in P-time. Since the methodic discovery of a new key, even for a single relation scheme, is the NP-complete problem at the source of the intractibility of BCNF [Osb], such a transformation would solve the BCNF problem in this case. To solve it in more general cases seems impossible by this special method, because the needed assumptions are rather closely fitted to our formal description of the given scenario.

The assumed scenario IR models a type of well-designed database: there is one masterfile, S, which contains at least a determiner of a key for each of the other files, the latter files having either one key or else several equivalent keys. Some well-designed, regular ERD's have canonical relational schemata that can be described by the scenario above: Examples 3 and 4 are such cases "in the small", but many others can be imagined, where

one file like the Transact_Form file determines a key of all the other files.

The main contribution of this section of the paper is chiefly the insight that a fairly obscure theorem, published by the Czech researcher J. Pokorny, applies to this subclass of the regular ERD's [Pok]. We use it to help test BCNF: to do so when normal forms are untreated in Pokorny's paper, takes some mathematical manipulation, but the "key" machinery is available in his work!

Application to this special scenario (#) will show the logical power of Pokorny's theorem, but it is impossible within the constraints here even to outline its long proof. Essentially, the result here can be applied to relational schemas other than as depicted in the scenario, as long as their non-binary relation schemes having m-related entities are losslessly joinable into a single scheme, S. The exact conditions under which such joining is possible is worthy of further research but is beyond the scope of this work, which only seeks to clarify the basic approach.

Concerning the masterfile S, the basic Rule 3 FD's of S are in:

$$F(S)=\{ P1...Ps \rightarrow Ej,\ 1<=j<=t\},$$

where the Pi are the m-related and the Ej the 1-related entities of S. [We may assume by renumbering that these entities are the first t in |R.] Of course, P1...Ps is not necessarily a minimal key of S, which is part of our problem. If we knew that P1...Pr, r<s, say were a minimal key then we would retain only P1,.....,Pr as the m-related entities in S and relegate the other Pj's to be among its 1-related entities. Since the minimal left-hand sides of the t FDs in F(S) can be determined easily, by standard algorithms, we shall assume that F(S) is minimal.

Another assumption needed for using Pokorny's Theorem is the non-redundancy of left-hand side attributes in the FDs of F(|R). But this is trivial for the unit FDs in the other schemes, and we have just guaranteed by the minimality of F(S) that we do not have either Pi->Pj or Pj->Pi.

Note that all the fundamental FD's in |F - F(S) are of the form K -> A where K and A are entity-set primary (key) attributes, and K is the key of a relation scheme other than S. FD's in this simple form, with both left and right sides a single attribute, are called "linear" by Pokorny; following common usage, we call them "unit FDs" [BB79].

DEFINITION (ASSOCIATED SET OF UNIT FDs): Let us denote by

$$UFD(|R) = [\ F(|R) - F(S)\ ] \cup \{ Pi \rightarrow Ej: \text{all } i<=s, j<=t\},$$

the "set of unit FDs associated with" F(IR). The set UFD(IR) cannot be expected to be FD-equivalent to F(IR), but, under certain further conditions, it may be "S-key-equivalent", meaning that the sets of keys of S with respect to either F(IR) or UFD(IR) would be identical. This is a desirable state: it would enable testing S for the BCNF-ness! Indeed, Pokorny gives such conditions that are a) easily P-time testable, b) necessary when BCNF does in fact hold, and c) in any case strong enough to make the determination of BCNF possible in P-time.

To employ Pokorny's theorem in our setting, we define the directed graph

G(IR) = <(E1,...,Em); UFD(IR) >, where each unit FD

Ei->Ej in UFD(IR) is considered as an edge joining node Ei to node Ej. Thus G(IR) is the directed graph with nodes all the entity sets in IR and edges the unit FD's associated with F(IR). Since IR can be assumed to be a connected ERD (via the database key), it is easy to see that the strongly connected components of G(IR), hereafter called components, determine what sets of attributes are the keys of S under the FD constraints in UFD(IR). By one other assumption, Pokorny's theorem shows us how to determine that the same sets are also the keys of S under the FD's in all of F(IR)! We shall explain that other assumption, which is easy to check in our scenario case.

Recall that any key of S is a key/determiner of any entity-set Ei. By a source component in G(IR) we shall mean, as in [Pok], a component whose entities (all equivalent under unit fd's) have in-degree zero: there are no unit FDs from other components of which they are the right-hand side. Clearly, any key of S must consist of attributes Pi that are in separate source components. All source components are easily computable in (small) polynomial time, using Tarjan's algorithm [AHU].

Let the source components of G(IR) be denoted as S1,...,Sk.

DEFINITION: Let f: A1.....An -> E be any FD in F(IR). Let C be any non-source component in G(IR). We say that f is <u>hierarchial for C</u> if: (a) E is in C, and (b) none of the Aj are in C, for 1<=j<=n.

THEOREM 4: For the defined UFD(IR), the masterfile S of IR with the assumptions above, and F(IR) the set of basic FDs of IR having the minimality assumptions made above:

the sets of keys of S with respect to either F(IR) or UFD(IR) are identical, if and only if:

(*) for each non-source component C in G(IR), there is at least one FD f in F(IR) that is hierarchial for C.

NOTE: This is a restatement and application of Pokorny's Theorem 2 to our scenario. When this theorem is applicable, it will be tractable to check the BCNF conditions for S, because finding keys with respect to a set of unit FDs is solvble in P-time [Pok].

Suppose that the condition (*) of Theorem 4 is true. Then, in particular for any non-source component C holding an Ej of S: there is at least one FD in F(S) of the form P1...Ps -> Ej such that Ej is in C and no Pi is in C, i.e. Ej->Pi is NOT in F(!R). Indeed, this is case, because, as we next show, for our special scenario, this desirable condition (*) is in fact true!

LEMMA: For relational schema !R of the special form (#), with the minimality of F(!R) assumed above, the condition (*) of Theorem 4 is always true.

PROOF: In the acenario case (#), recall, the minimal left-sides of basic FDs in F(S) contain Pi that can be found in source components of G(!R). Now, each non-source component C lies at the end of an incoming edge (a unit FD) from some other component B. Let E in C be the right-side of such a unit FD. By the nature of the two kinds of unit FDs in G(!R), associated with the basic FDs in F(!R), there are two main cases:

(1) B is a source component, and the unit FD can be taken to be P -> E. Then either this is a basic FD of F(!R) or else, when E is in S and P is some Pi, a minimal FD P1...Ps -> E is in F(!R). In either subcase, none of the left side P's is in C because C is non-source;

(2) B is a non-source component distinct from C, and the connecting unit FD in G(!R) is say D -> E, where D is in B. But this implies that D -> E is a basic FD in F(!R), and that D is not in C because distinct components are disjoint.

Thus, in either main case for C an arbitrary non-source component, there is an FD in F(!R) which is hierarchial for C.[]

COROLLARY. For a relational schema of a regular ERD, in the form
(#) !R = (E1,...,Em; R1,...,Rn; S), as described above, the property of being in BCNF can be decided algorithmically in polynomial time.

PROOF: Recall that all schemes in !R, except possibly S, are already in BCNF, by Theorems 1 and 2. Because Theorem 4's condition (*) is true for !R, it can be applied to verify whether S is in BCNF wrt F(!R) by finding the keys of S with respect to the unit FD's in UFD(!R). (Pokorny also showed that the key-finding problem for a set of unit FD's is solvable in P-time). Then, for any FD X -> A in the scope of S, one will know whether X contains a key of S wrt F(!R), and so whether S is in BCNF. Since the basic FDs of F(!R) form a cover of all FDs, the whole process can be done in time polynomial in the size of F(!R). []

IV. CONCLUDING REMARKS

We have seen two possible approaches to the problem of whether a relational database schema $\mathcal{R}$ is in BCNF, where we consider $\mathcal{R}$ the canonical schema of a regular ERD $\mathcal{E}$. In the first approach, that of finding sufficient conditions to imply BCNF, we saw incidentally that many relation schemes in $\mathcal{R}$ are in fact in BCNF with respect to $F(\mathcal{E})$ due to the regularity of $\mathcal{E}$: namely, the schemes of entity sets (Theorem 1), of any binary relationships (Proposition 1), and of any purely 1-related relationship sets (Theorem 2). To ensure that every relation scheme in $\mathcal{R}$ is in BCNF, we defined the loop-free condition for an ERD; this condition, in addition to regularity, obtains the desired result (Theorem 3). There are P-time algorithms to determine the regularity or loop-freeness of arbitrary ERDs (Proposition 3).

However, loop-freeness is equivalent to Berge acyclicity of $\mathcal{R}$, which is the strongest notion of acyclicity usually studied for relational schema [Fag81]. It is also a fairly restrictive condition: an ERD with two different relationships that share at least two entity sets (as in Example 1) violates it. Fagin's results on Berge acyclicity show it eliminates ambiguity in navigational paths to answer queries, and so it is nonetheless a desirable property, both from this aspect and from the fact that the implied BCNF condition prevents certain of the classical update and insertion anomalies [LePa].

For our second approach to the BCNF problem, we posited an ERD-based design scenario suggested by several examples that makes it possible to test for the condition. To see that this test can be done in the scenario setting, it was helpful to use the "key-equivalence via unit FDs" result of Pokorny (Theorem 4). In its corollary, we have given a method that determines truth or falsity of BCNF, under quite feasible conditions. These conditions include the non-existence of pairs of redundant attributes P1....Ps in the primary key of S, which holds a database key, implying all other attributes. To reduce the seeming "restriction" of this design requirement, note that the DBA can avoid redundancy by combining equivalent attributes into one entity, with separate fields for the attributes.

Overall, we have seen that when the Section III relational model $\mathcal{R}$ of the scenario is used as a guide, a specialized database design methodology is obtained that has the desired property: its specific cases, various relational schemata, can be tested for BCNF in polynomial time. The scenario design model also has other pleasing properties, e.g, that its most complex relation scheme S acts as a "masterfile" for the database, and that its own minimal keys are easy to find. While this model is clearly not universally applicable, in many cases (e.g., where a database key can be given) it may turn out to be useful, at least as a guide.

#####

REFERENCES

[AHU] Aho, A., Hopcroft, J. and Ullman, J. "The Design and Analysis of Computer Algorithms", Addison-Wesley, Reading, 1976

[AtPa] Atzeni, P. and Parker, D.S. Assumptions in relational database theory. Proc. 1st ACM Conf. Principles DB Systems, 1982

[BB79] Beeri, C. and Bernstein, P. Computational problems related to the design of normal form relational databases. ACM Trans. Database Sys., 4(1), Mar. 1979

[Berg] Berge, C. "Graphs and Hypergraphs", North-Holland, Amsterdam, 1976.

[Ch76] Chen, P. The entity-relationship model: towards a unified view of data. ACM Trans. Database Sys., 1(1), Mar. 1976

[Ch80] Chen, P. (ed.) "Entity-Relationship Approach to Systems Analysis and Design", North-Holland, Amsterdam, 1980

[Codd] Codd, E.F. Recent investigations in relational database systems, Proc. 1974 IFIP Cong., North-Holland, 1974

[Fag81] Fagin, R. Types of acyclicity for hypergraphs and relational databases, Technical Report RJ 3330, IBM San Jose, 1981

[JF] Jou, J. and Fischer, P. The complexity of recognizing third-normal form, Info. Proc. Letters 14 (4), June 1982

[JNS83a] Jajodia,S., Ng,P. and Springsteel, F. Entity-relationship diagrams which are in BCNF, Intl. J. Comp. Info. Sci. 12 (4), 1983

[JNS83b] Jajodia, Ng, and Springsteel. Problems of equivalence for entity-relationship diagrams, IEEE Trans. Software Engr. 9, 1983

[JNS83c] Jajodia, Ng, and Springsteel. On universal and representative instances for inconsistent databases. Proc. 3rd Entity-Relationship Conf., North-Holland, 1983

[LePa] LeDoux, C. and Parker, D.S. Reflections on Boyce-Codd Normal Form, Proc. 8th Intl. Conf. Very Large Data Bases, 1982

[Lien] Lien, E. On equivalence of database models, J.ACM 29(2), 1982

[Osb] Osborn, S. Testing for existence of a covering Boyce-Codd Normal Form, Information Processing Letters 8 (1), 1979

[Pok] Pokorny, J. Key-equivalence of functional dependency systems, Proc. 10th Symp. Math. Foundations Computer Science. Lecture Notes in Comp. Sci. #118, Springer-Verlag (1981)

[Ulm82a] Ullman, J. "Principles of Database Systems", Second Edtn., Computer Science Press, Rockville, Maryland, 1982

[Ulm82] Ullman, J. The U(niversal) R(elation) strikes back! Proc. 1st ACM Conf. Principles Database Systems, 1982

Prof. F.N. Springsteel

University of Missouri at Columbia
Department of Computer Science
Columbia, Missouri 65211 USA
Telephone: (314) 882-4480, 882-3842

A restricted design methodology to allow testing for BCNF in polynomial time

F.N. Springsteel

Summary

In a relational database it is important to be able to test a proposed database design for the Boyce-Codd Normal Form (BCNF) condition. In the paper conditions are given that are either sufficient for BCNF or show that BCNF cannot be tested in polynomial time. There are two sets of conditions. The first one logically imply BCNF and it provide a setting which helps to suggest the second one in which it is easier to test for BCNF. The proposed methodology works for many databases expressible by E-R diagrams which have a "database key".

Egy korlátozott tervezési metodika BCNA tesztelésére polinomiális időben

F.N. Springsteel

Összefoglaló

A relációs adatbázis modell esetén igen fontos kitesztelni a javasolt adatbázis tervet, vajon teljesiti-e a Boyce-Codd-féle Normál Alak /BCNA/ feltételt. A cikkben olyan feltételek vannak megadva, amelyek vagy elégségesek a BCNA-hoz, vagy megmutatják, hogy a BCNA-t nem lehet polinom idő alatt kitesztelni. Kétfajta feltételrendszer van. Az elsőnek logikai következménye a BCNA és egyben megmutatja hogyan néz ki a második, amelyben a BCNA-t már könnyebben lehet tesztelni. A javasolt metodika különösen alkalmas azokra az adatbázisokra, amelyeket "adatbázis kulcs"-al rendelkező E-R diagramm segitségével lehet leirni.