Boyce–Codd Normal Form (BCNF)

- A relation `R(X)` is in *Boyce–Codd Normal Form* if for every non-trivial functional dependency `Y` → `Z` defined on it, `Y` contains a key `K` of `R(X)`. That is, `Y` is a superkey for `R(X)`.
- Example: Person1(*SI#, Name, Address*)
  - ✓The only FD is *SI#* → *Name, Address*
  - ✓Since *SI#* is a key, Person1 is in BCNF
- Anomalies and redundancies, as discussed earlier, do not occur in databases with relations in BCNF.

Non-BCNF Examples

- `Person(SI#,Name,Address,Hobby)`
  - ✓The FD *SI#* → *Name,Address* does *not* satisfy conditions for BCNF since the key is {*SSN,Hobby*}
- `HasAccount(AcctNum,ClientId,OfficeId)`
  - ✓The FD *AcctNum* → *OfficeId* does *not* satisfy BCNF conditions if we assume that keys for `HasAccount` are {*ClientId,OfficeId*} and {*AcctNum,ClientId*}, rather than *AcctNum*.

A Relation not in BCNF

Manager	Project	Branch
Brown	Mars	Chicago
Green	Jupiter	Birmingham
Green	Mars	Birmingham
Hoskins	Saturn	Birmingham
Hoskins	Venus	Birmingham

Assume the following dependencies:

- → Manager → Branch — each manager works in a particular branch;
- → Project,Branch → Manager — each project has several managers, and runs on several branches; however, a project has a unique manager for each branch.

A Problematic Decomposition

- The relation is not in BCNF because the left hand side of the first dependency is not a superkey.
- At the same time, no decomposition of this relation will work: **`Project,Branch`** → **`Manager`** involves all the attributes and thus no decomposition is possible.
- Sometimes BCNF *cannot* be achieved for a particular relation and set of functional dependencies without violating the principles of lossless decomposition and dependency preservation.

Normalization Drawbacks

- By limiting redundancy, normalization helps maintain consistency and saves space.
- *But* performance of querying can suffer because related information that was stored in a single relation is now distributed among several
- Example: A join is required to get the names and grades of all students taking CS343 in 2007F.

```
Student(Id, Name)
Transcript(StudId, CrsCode, Sem, Grade)
```

```
SELECT S.Name, T.Grade
FROM Student S, Transcript T
WHERE S.Id = T.StudId AND
      T.CrsCode = 'CS343' AND T.Sem = '2007F'
```


Denormalization

- Tradeoff: *Judiciously* introduce redundancy to improve performance of certain queries
- Example: Add attribute ***Name*** to **Transcript** → **Transcript'**

```
SELECT  T.Name, T.Grade
FROM    Transcript'  T
WHERE   T.CrsCode = 'CS305'  AND  T.Sem = 'S2002'
```

  - ✓ Join is avoided;
  - ✓ If queries are asked more frequently than Transcript is modified, added redundancy might improve average performance;
  - ✓ But, **Transcript'** is no longer in BCNF since key is **{*StudId*, *CrsCode*, *Sem*}** and ***StudId*** → ***Name***.

BCNF and 3NF

- The Project-Branch-Manager schema is not in BCNF, but it *is* in 3NF.
- In particular, the **Project, Branch** → **Manager** dependency has as its left hand side a key, while **Manager** → **Branch** has a unique attribute for the right hand side, which is part of the {**Project, Branch**} key.
- The 3NF is less restrictive than the BCNF and for this reason does not offer the same guarantees of quality for a relation; it has the advantage however, of *always* being achievable.