Measurement of the goodness of the quality of design can be discussed at 2 levels.

The first is logical or conceptual level - i.e. how users interpret the relational schema and the meaning of their attributes. (Both Base/Virtual R)

The second is the implementational level, how the tuples in the base relation are stored and updated. (Only base R)

Some guidelines to a good relation are discussed as follows

(1) Imparting clear semantics to the attributes in relations i.e the easier it is to explain the semantics of the relation, the better the relational schema design will be
∴ Design a relational schema so that it is easy to explain its meaning and do not combine attributes from multiple entity types and relationship types into a single relation.

(2) The schema design should be such that it minimize the storage space used by the base relation

- employee (Ename, Ssn, Bdate, Address, Dnumber)
  PK, FK
- Department (Dname, Dnumber, Dmgr_ssn)
  PK, FK
- Dept loc (Dnumber, Dlocation)
  F.K, P.K.
- Project (Pname, Pnumber, Plocation, Dnum)
  PK, F.K
- works on (Ssn, Pnumber, Hrs)
  F.K, F.K, P.K

eg. consider if there been a relation -

Emp-Dept (Ename, Ssn, Bdate, Address, Dnumber, Dname, Dmgrssn)
which is a result of natural Join of Employee & Department

The attribute values pertaining to a particular department are repeated for every employee who works for that dept. whereas in department relation, this info appears only once. Only the Department number is repeated in the employee.

Such Relations like Emp-Dept also gives rise to a serious problem i.e. ~~ISER~~ UPDATION Anomalies.

These can be classified into

Emp-Proj (Ssn, Pnumber, hrs, Ename, Pname, Plo)
mixes att of employee, Project, works on

→ Insertion Anomalies

(a) whenever inserting a new tuple, have to ~~fact~~ take care of consistency of data. For eg. Inserting a new employee in emp-dept, attribute values of dept 5 must be entered carefully & correctly so that they are consistent with other values.
Where as if employee table is used do not have to worry about this prob coz we enter only dept no.
More over if the employee havent been assigned a dept, then it will lead to insertion of NULL values, also not a sign of good design

(b) Difficult to insert a new dept that has no employees as yet in the emp-dept relation. Only way to do this is to place NULL values in the attributes of employees. This cause the problem because Ssn is the primary key of empl-dept.

→ Deletion Anomalies If we delete an employee tuple from emp-dept table that also happens to be the last emp working for a particular dept, then the info concerning the dept is also lost from the data base. Same as this prob does not occur if we have two seperate tables employee & department

ification Anomalies

In Emp-dept, if we change the value of one of the attributes of a particular dept, eg manager of dept 5, then we must update the tuples of all employees who work in that dept. Otherwise the database will become inconsistant.

∴ The design of the base relation should be such that no insertion, deletion, modification anomalies are present.

(3) If NULL values are present then the result becomes unpredictable as it can have multiple interpretations

- ① Attribute does not apply to this tuple
- ② " value for this tuple is unknown
- ③ The value is known but absent that is not yet recorded

∴ as far as possible avoid placing attributes in a base relation whose values may ~~be~~ frequently be NULL.

(4) Consider two relations —

- Emp_Locs (Ename, Plocation) — PK
- Emp_Proj1 (Ssn, Pnumber, hrs, Pname, Plocation) — PK

which can be used

Instead of Emp_Proj (Ssn, Pnumber, Hrs, Ename, Pname, Plocation) — Redundancy, Redundancy

This produces a bad schema design coz we cannot recover the info that was originally in Emp_Proj. If we attempt a natural join on Emp_locs & Emp_Proj1 then it produces additional tubles called spurious tuples. coz they represent spurious info that is not valid.

This is because. In this case Plocation is that attribute that relates your 2 tables which is neither a PK or FK

∴ Design relation schemas so that they can be joined with equality condition on the attributes that are PK FK so that gaurntees no spurious tuples generation

Functional Dependency

A functional dependency (FD) is a constraint between two sets of attributes from the database. Suppose that is a relational schema R having n attributes $R(A_1, A_2, \ldots A_n)$ Then a FD denoted by $X \to Y$, between two sets of attributes X & Y that are subset of R specifies a constraint that, for any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$, they also must have $t_1[Y] = t_2[Y]$

This means value of Y component is dependent on or determined by X or value of X component of a tuple uniquely determine the value of Y component

∴ we say is $X \to Y$, then there is a fd from X to Y

or Y is functionally dependent on X.

$A \to BC$, $A \to B$, $A \to C$ ✓
$AB \to C$ $A \to C$, $B \to C$ ✗

NOTE

① if $X \to Y$, that means X is a candidate key for R

② if $X \to Y$ this does not say $Y \to X$ in R.

eg: Teach

Teacher	Course	Text
Smith	DS	Bartram
Smith	DBMS	Martin
Hall	compiler	Hoffman
Brown	DS	Horowitz

✓ Text → Course
✗ Teacher → Course
✗ Course → Text