Normalization Details (Partiality and Transitivity)

Simpler relations can be derived by a series of mathematical steps known as normalization.

Normalization uses the notion of ***dependency***. The idea is to only include things (attributes) that are related to, or dependent on, the primary key, i.e., group attributes that are functionally dependent.

RECALL: an attribute B of relation R is functionally dependent on attribute A if, at every instant of time, each A-value in R is associated with one and only one B-value.

Partial functional dependency occurs if a non-key attribute is dependent on only part of a composite key.

example:

```
STUDENT_COURSE (S#, C#, S_NAME, S_ADDR, C_TITLE, GRADE)
```

Partial dependencies cannot be tolerated because a table that contains such dependencies is subject to data redundancies and, therefore, to update anomalies. The data redundancies are caused by the fact that every row entry requires a duplication of data.

Solution:

```
STUDENT (S#, S_NAME, S_ADDR)
STUDENT_COURSE (S#, C#, GRADE)
COURSE (C#, C_TITLE)
```

Transitive dependency: attribute C is transitively dependent on attribute A if there is an attribute B such that A → B and B → C, giving A → C.

Also called mutual dependency.

example:

```
STUDENT_MAJOR (S#, MAJOR_DEPT, DEPT_HEAD)
```

Assume

S# → MAJOR_DEPT

MAJOR_DEPT → DEPT_HEAD

so...

S# → DEPT_HEAD

DEPT_HEAD is dependent on S# directly, or transitively through

S# → MAJOR_DEPT → DEPT_HEAD

Partial and transitive dependencies cause anomalies and must be eliminated.

Relations can be broken down to eliminate transitivity and partiality.

Elimination of Transitivity:

```
STUDENT_MAJOR (S#, MAJOR_DEPT)
DEPT (MAJOR_DEPT, DEPT_HEAD)
```