Foreign Key

A set of attributes FK in relation schema R1 is a **foreign key** of R1 that **references** relation R2 if it satisfies the following rules:

1. The attributes in FK have the same domain(s) as the primary key attributes PK of R2; the attributes FK are said to **reference or refer to** the relation R2.
2. A value of FK in a tuple t1 of the current state r1(R1) either occurs as a value of PK for some tuple t2 in the current state r2(R2) or is NULL. In the former case, we have t1[FK] = t2[PK], and we say that the tuple t1 **references or refers to** the tuple t2.

In this definition, R1 is called the **referencing relation** and R2 is the **referenced relation**. If these two conditions hold, a **referential integrity constraint** from R1 to R2 is said to hold.

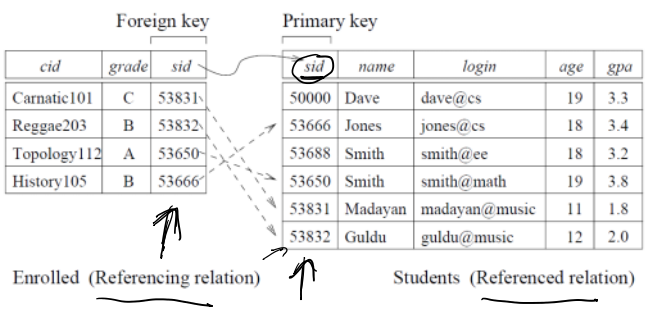
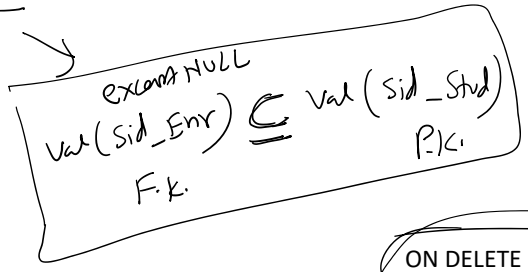
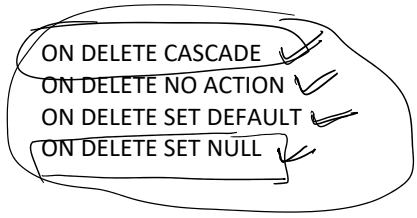


Figure Referential Integrity






```
CREATE TABLE Enrolled
  (sid CHAR(20), cid CHAR(20), grade CHAR(2),
   PRIMARY KEY (sid,cid),
   FOREIGN KEY (sid) REFERENCES Students )
```

```
CREATE TABLE Enrolled ( sid    CHAR(20),
                        cid    CHAR(20),
                        grade  CHAR(10),
                        PRIMARY KEY (sid, cid),
                        FOREIGN KEY (sid) REFERENCES Students
                                ON DELETE CASCADE
                                ON UPDATE NO ACTION )
```

Only students listed in the Students relation should be allowed to enroll for courses.

Enrolled(sid: string, cid: string, grade: string)

The sid field of Enrolled is called a foreign key and refers to Students Relation.

The foreign key in the referencing relation (Enrolled, in our example) must match the primary key of the referenced relation (Students), i.e., it must have the same number of columns and compatible data types, although the column names can be different.

There may well be some students who are not referenced from Enrolled (e.g., the student with sid=50000). However, every sid value that appears in the instance of the Enrolled table appears in the primary key column of a row in the Students table.

If we try to insert the tuple <Art104, A, 55555> into Enrolled , the **integrity constraint** is violated because there is no tuple in Students with the id 55555; the database system should reject such an insertion.

Similarly, if we delete the tuple <53666, Jones, jones@cs, 18, 3.4> from S1, we violate the foreign key constraint because the tuple <53666, History105, B> in E1 contains sid value 53666, the sid of the deleted Students tuple. The DBMS should disallow the deletion or, perhaps, also delete the Enrolled tuple that refers to the deleted Students tuple.

Finally, we note that a foreign key could refer to the **same relation**. For example, we could extend the Students relation with a column called partner and declare this column to be a foreign key referring to Students.

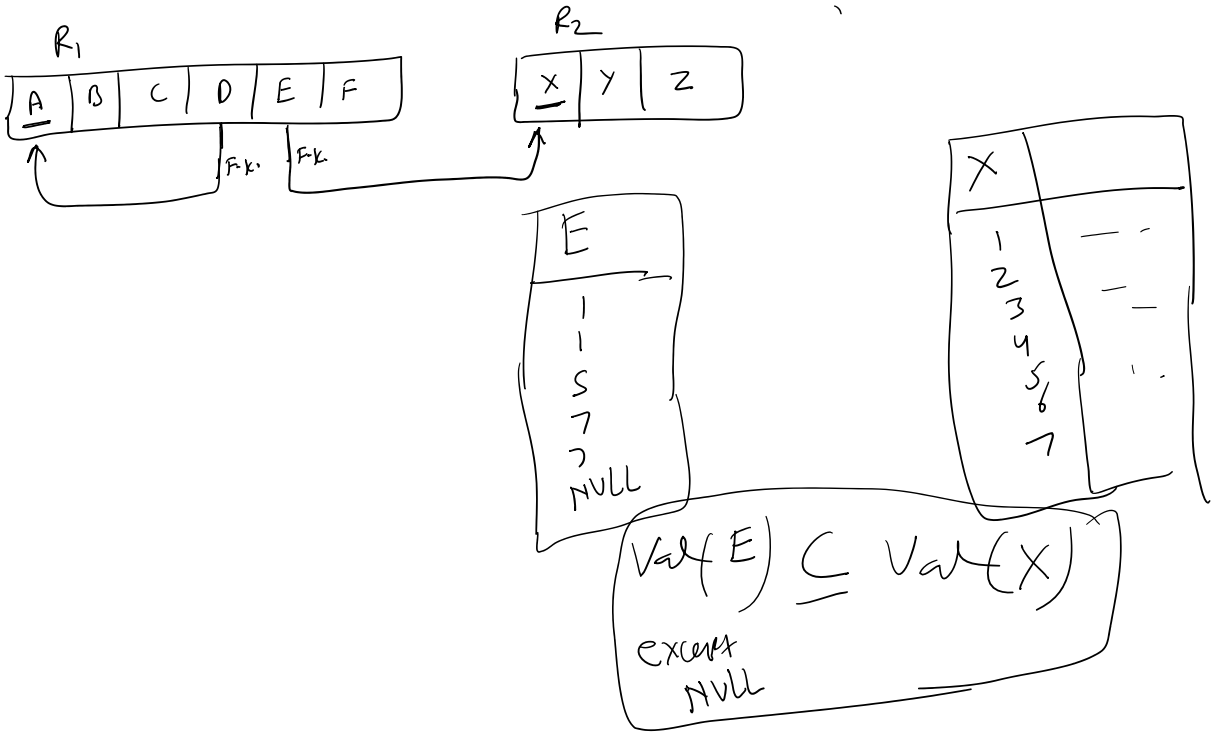
Intuitively, every student could then have a partner, and the partner field contains the partner's sid. The observant reader will no doubt ask, "What if a student does not (yet) have a partner?" This situation is handled in SQL by using a special value called **null**. The use of null in a field of a tuple means that value in that field is either **unknown or not applicable** (e.g., we do not know the partner yet, or there is no partner). The **appearance of null in a foreign key field does not violate the foreign key constraint**. However, null values are not allowed to appear in a primary key field (because the primary key fields are used to identify a tuple uniquely).

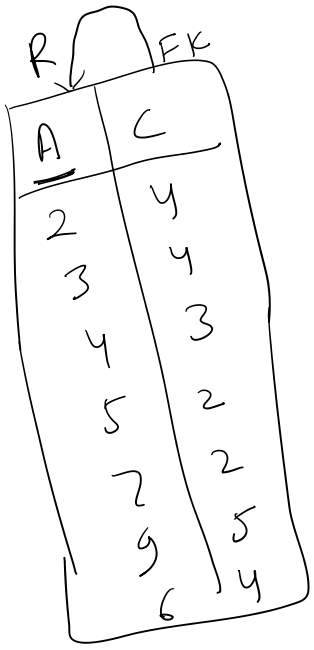
Limits and restrictions

- A foreign key constraint doesn't have to be linked only to a primary key constraint in another table. Foreign keys can also be defined to reference the columns of a UNIQUE constraint in another table.
- When a value other than NULL is entered into the column of a FOREIGN KEY constraint, the value must exist in the referenced column. Otherwise, a foreign key violation error message is returned. To make sure that all values of a composite foreign key constraint are verified, specify NOT NULL on all the participating columns.
- FOREIGN KEY constraints can reference only tables within the same database on the same server. Cross-database referential integrity must be implemented through triggers. For more information, see CREATE TRIGGER.
- FOREIGN KEY constraints can reference another column in the same table, and is referred to as a self-reference.

From <https://docs.microsoft.com/en-us/sql/relational-databases/tables/create-foreign-key-relationships?view=sql-server-ver15>

Foreign key- A set of attributes that references primary/alternate key of the same relation or some other relation.



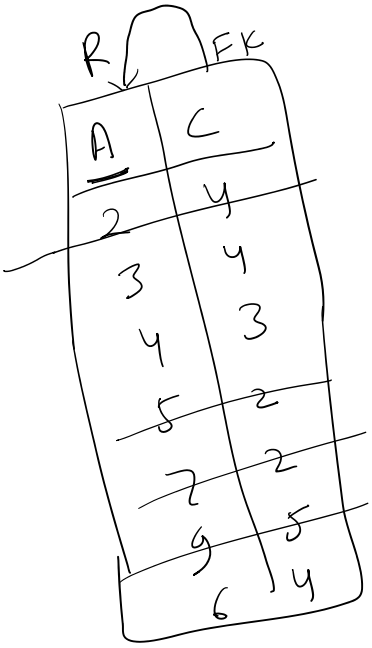


PK: A

FK: C

ON Del CAS

Del (2,4)



$Val(C) \subseteq Val(A)$

Functional Dependencies

Let R be a relation schema and let X and Y be nonempty sets of attributes in R. We say that an instance r of R satisfies the FD $X \rightarrow Y$ if the following holds for every pair of tuples t_1 and t_2 in r:

If $t1.X = t2.X$, then $t1.Y = t2.Y$.

This means that the values of the Y component of a tuple in r depend on, or are *determined by*, the values of the X component; alternatively, the values of the X component of a tuple uniquely (or **functionally**) *determine* the values of the Y component. We also say that there is a functional dependency from X to Y, or that Y is **functionally dependent** on X. The abbreviation for functional dependency is **FD** or **f.d.** The set of attributes X is called the **left-hand side** of the FD, and Y is called the **right-hand side**.

Thus, X functionally determines Y in a relation schema R if, and only if, whenever two tuples of $r(R)$ agree on their X-value, they must necessarily agree on their Y-value. Note the following:

- If a constraint on R states that there cannot be more than one tuple with a given X-value in any relation instance $r(R)$—that is, X is a **candidate key** of R—this implies that $X \rightarrow Y$ for any subset of attributes Y of R (because the key constraint implies that no two tuples in any legal state $r(R)$ will have the same value of X). If X is a candidate key of R, then $X \rightarrow R$.
- If $X \rightarrow Y$ in R, this does not say whether or not $Y \rightarrow X$ in R.

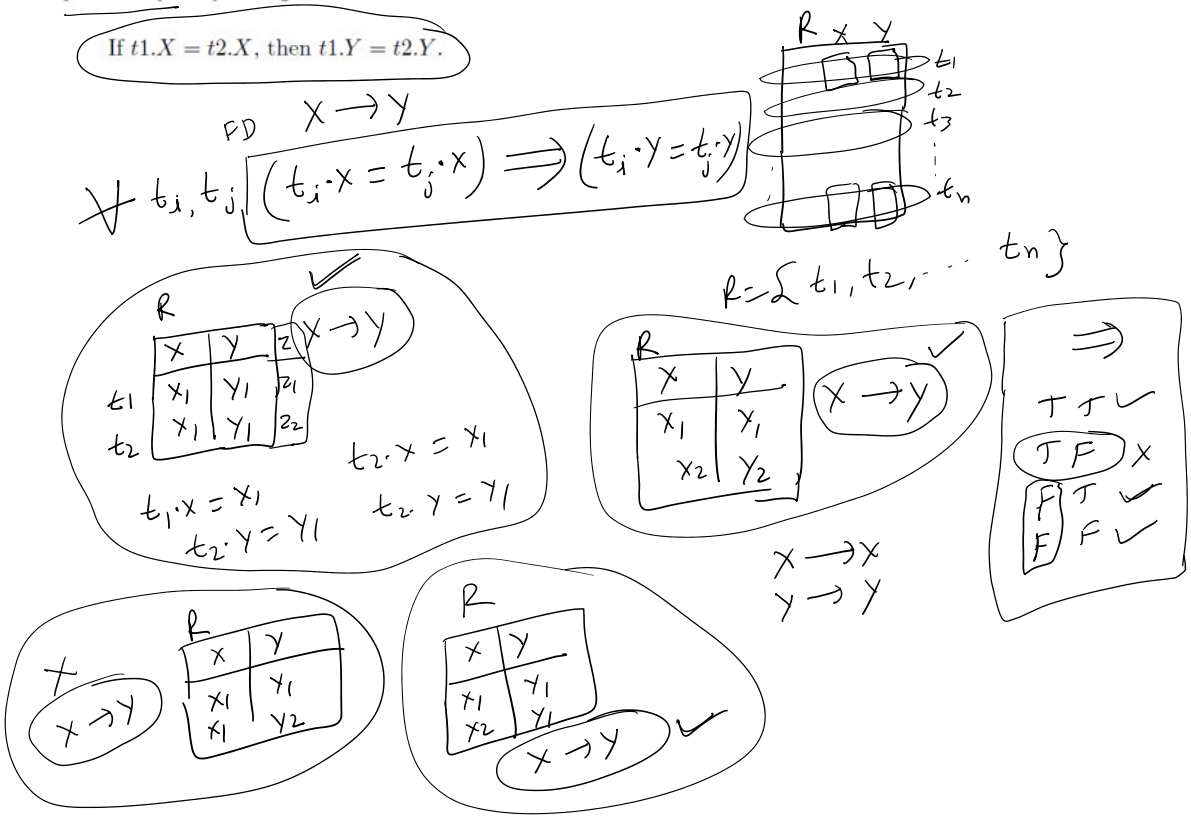
K is a superkey for relation schema R if and only if $K \rightarrow R$

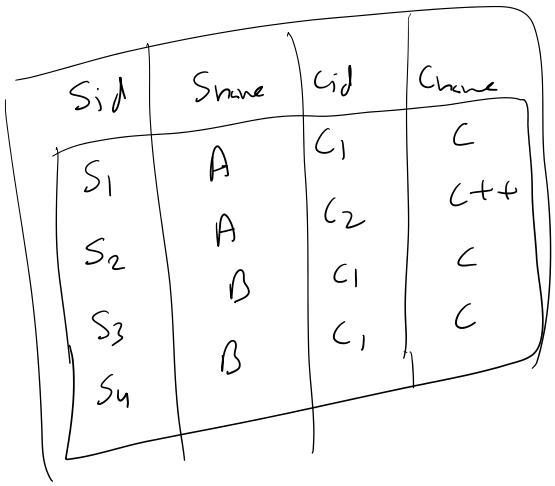
K is a candidate key for R if and only if

- $K \rightarrow R$, and
- for no $\alpha \subset K$, $\alpha \rightarrow R$

Let R be a relation schema and let X and Y be nonempty sets of attributes in R. We say that an instance r of R satisfies the FD $X \rightarrow Y$ if the following holds for every pair of tuples t_1 and t_2 in r:

If $t1.X = t2.X$, then $t1.Y = t2.Y$.





Cid → Cname

Sid → Sname

Sid → Cid

Sid → Cname

R(A B C D)

FD: (Any key) → (any attribute R ABCD)

PK
or
CK
or
SK
or AK

Types of Functional Dependency

1. Trivial Functional Dependency

A functional dependency X → Y is trivial if X ⊇ Y (Y is a subset of X); otherwise, it is nontrivial.

$X \supseteq Y$ $X \rightarrow Y$ F.D.

Sid Sname → Sname

Sname → Sname

2. Non trivial Functional Dependency

A functional dependency X → Y is nontrivial if Y is not a subset of X.

When X intersection Y is NULL, then X → Y is called as complete non trivial

$X \cap Y = \phi$ $X \rightarrow Y$

Question. Identify all complete non trivial functional dependencies (FD) for the given relation

R

A	B	C
1	4	2
2	5	2
3	4	2
3	5	2

✗ $A \rightarrow B$ ✓ $B \rightarrow C$ $C \rightarrow A$ ✗

✓ $A \rightarrow C$ ✗ $B \rightarrow A$ $C \rightarrow B$ ✗

✗ $A \rightarrow BC$ ✗ $B \rightarrow AC$ $C \rightarrow AB$ ✗

R (ABC)

$AB \rightarrow C$ ✓

$BC \rightarrow A$ ✗

$AC \rightarrow B$ ✗

$A \rightarrow C$

$B \rightarrow C$

$AB \rightarrow C$

R

A	B	C
1	2	3
4	5	6