Relational Integrity constraints or RULES

Concept of Null or Unknown Value

- Null represent a value for an attribute that is currently unknown or it Is not applicable for this tuple.
- Null is not same as 0 or zero value or a text string.
- It represent absence of value
- At a particular time the following table might not have the value of age (of Rajesh) and Job (of Raja)

Name	Age	Job
Rajesh	-	Clerk
Raja	23	-
Amit	43	Sales

Difference between Null and Not Applicable

- Blank against Amit might indicate that pension is not applicable to this employee as per firm policy.

Name	Age	Job	Pension_Date
Rajesh	-	Clerk	-
Raja	23	-	01/01/1999
Amit	43	Sales	-

Best way to represent it as :

Name	Age	Job	Pension_Date
Rajesh	null	Clerk	null
Raja	23	null	01/01/1999
Amit	43	Sales	Not Applicable

Issues in basic STUDENT table

Basic table suffers from following anomalies:

- There is no check on storing duplicate records
- There is no mandatory column
- There is no check on validity of data

To ensure all these features, there is a need to apply constraints during creation of table.

Relational Integrity constraints/RULES

Relational Integrity constraints is referred to conditions which must be present for a valid relation. These integrity constraints are derived from the rules that the database represents.

There are many types of integrity constraints. Constraints on the Relational database management system is mostly divided into three main categories are:

1. **Domain constraints**
2. **Entity integrity or Key/primary key constraints**
3. **Referential integrity constraints**

Relational Integrity constraints or RULES

Data Integrity

The following categories of data integrity exist with each RDBMS:

- **Entity Integrity:** There are no duplicate rows in a table.
- **Domain Integrity:** Enforces valid entries for a given column by restricting the type, the format, or the range of values.
- **Referential integrity:** Rows cannot be deleted, which are used by other records.
- **User-Defined Integrity:** Enforces some specific business rules that do not fall into entity, domain or referential integrity.

Levels of Constraints

Column level constraints

Constraints which are applied on single column of table are known as column level constraints.

For example:

Roll_Number is Unique.

Table level constraints

Constraints which are applied on more than one column of a table are known as table level constraints.

For example:

Roll_Number and Class combination is unique in primary School database

[In primary school Roll number of students starts with one in each class]

Types of Constraints

- Not Null
- Unique
- Primary key
- Check
- Foreign key
- Default

Check Constraint

- Check constraints allow Oracle to verify the validity of data being entered on a table against a set of constants. These constants act as valid values.
- The Check constraint consists of the keyword CHECK followed by parenthesized conditions.
- Check constraints must be specified as a logical expression that evaluates either to TRUE or FALSE.
- Syntax:
- Columnname datatype (size) [constraint constraintname] CHECK (logical expression)

Example [Check: Column level constraint]

```
CREATE TABLE student
    (
    Roll_Number Number(4) ,  Name Char(15),
    Class Char(10) Check (class in ('BE','ME','MCA')),
    Marks Number(4) Check (marks>=0), DOB Date
    );
```

OR

```
    CREATE TABLE student
    (
    Roll_Number Number(4) ,  Name Char(15),
    Class Char(10) Constraint class_check Check (class in ('BE','ME','MCA')),
    Marks Number(4) Check (marks>=0), DOB Date
    );
```

Example [Check: Table level constraint]

```
CREATE TABLE library
(
Roll_Number Number(4) Primary Key,
Book_Number Number(10) NOT NULL,
DOI Date, DOR Date,
Check (DOR>DOI);
```

OR

```
CREATE TABLE library
(
Roll_Number Number(4) Primary Key,
Book_Number Number(10) NOT NULL,
DOI Date, DOR Date,
Constraint Date_Check Check (DOR>DOI);
```

Example

For example, the following program creates a new table called CUSTOMERS and adds five columns. Here, we add a CHECK with AGE column, so that you cannot have any CUSTOMER who is below 18 years.

```
CREATE TABLE CUSTOMERS(
   ID   INT              NOT NULL,
   NAME VARCHAR (20)     NOT NULL,
   AGE  INT              NOT NULL CHECK (AGE >= 18),
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);;
```

If the CUSTOMERS table has already been created, then to add a CHECK constraint to AGE column, you would write a statement like the one given below.

```
ALTER TABLE CUSTOMERS
  MODIFY AGE INT NOT NULL CHECK (AGE >= 18 );
```

Or

```
ALTER TABLE CUSTOMERS
  ADD CONSTRAINT myCheckConstraint CHECK(AGE >= 18);
```

DROP a CHECK Constraint

```
ALTER TABLE CUSTOMERS
  DROP CONSTRAINT myCheckConstraint;
```

Limitations of Check Constraint

Rno	Name	Class	Marks	DOB
1	Ram	BE	60	12-DEC-1980
2	Rajesh	MCA	70	13-JAN-1989
3	Surinder	ME	78	10-JUN-1980

Check class in ('BE', 'ME', 'MCA')

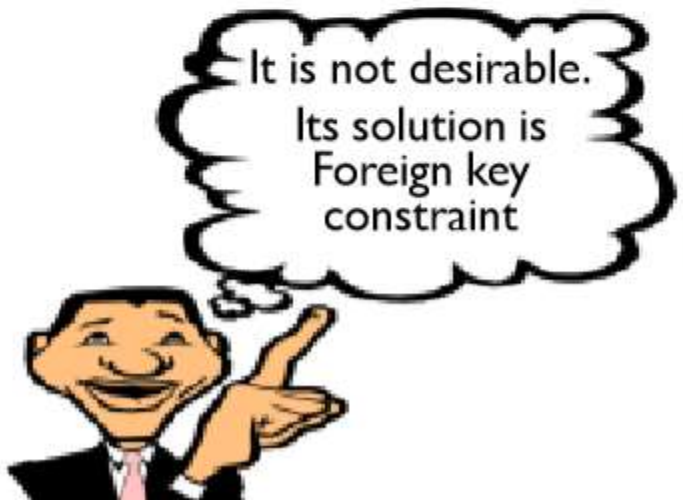
Limitations of Check Constraint

Rno	Name	Class	Marks	DOB
1	Ram	BE	60	12-DEC-1980
2	Rajesh	MCA	70	13-JAN-1989
3	Surinder	ME	78	10-JUN-1980
4	Rahat	MBA	76	15-MAY-1980

Check class in ('BE', 'ME', 'MCA')
It will not be allowed…

Limitations of Check Constraint

Rno	Name	Class	Marks	DOB
1	Ram	BE	60	12-DEC-1980
2	Rajesh	MCA	70	13-JAN-1989
3	Surinder	ME	78	10-JUN-1980
4	Rahat	MBA	76	15-MAY-1980



Check class in ('BE', 'ME', 'MCA')
To allow it we have to add it in check list
by changing schema

Limitations of Check Constraint

- It will not be suitable if check values need to be changed in future.
- Class can be added or removed in future, for check constraint is not desirable.
- Apply check constraint only if its value is not going to change in future.
- For example,
- Marks number(3) check (marks>=0)
- Gender

  Gender char(10) check (gender in

  ('male', 'female', 'transgender'));

NOT NULL Constraint

- By default, a column can hold NULL values.
- If you do not want a column to have a NULL value, then you need to define such a constraint on this column specifying that NULL is now not allowed for that column.
- A NULL is not the same as no data, rather, it represents unknown data.

NOT NULL Constraint

- Specifies if the column must contain value or might not contain any.
- By default all columns in a table allow nulls.
- NOT NULL specifies that all rows in the table to have value for specified column.
- Syntax:

  Columnname datatype (size) [constraint constraintname] NOT NULL

Example [Not Null: Column level constraint]

```
CREATE TABLE student
(
Roll_Number Number(4) Not Null,
Name Char(15), Class Char(10), Marks Number(4),
DOB Date
);
```

Example

```
CREATE TABLE CUSTOMERS(
   ID   INT                              NOT NULL,
   NAME VARCHAR (20)                    NOT NULL,
   AGE  INT                             NOT NULL,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

If CUSTOMERS table has already been created, then to add a NOT NULL constraint to the SALARY column in

```
ALTER TABLE CUSTOMERS MODIFY SALARY  DECIMAL (18, 2) NOT NULL;
```

UNIQUE Constraint

- **The UNIQUE Constraint prevents two records from having identical values in a column.**

- E.g. In the CUSTOMERS table, for example, you might want to prevent two or more people from having an identical age.

  - Enforce uniqueness of the column or group of columns.
  - Syntax:

    Columnname datatype (size) [constraint constraintname] UNIQUE

Example [Unique: Column level constraint]

```
CREATE TABLE student
(
Roll_Number Number(4) Unique,
Name Char(15), Class Char(10), Marks Number(4),
DOB Date
);
```

OR

```
CREATE TABLE student
(
Roll_Number Number(4) Constraint RNO_Unique Unique,
Name Char(15), Class Char(10), Marks Number(4),
DOB Date
);
```

Example

For example, the following SQL query creates a new table called CUSTOMERS and adds five columns.

Here, the AGE column is set to UNIQUE, so that you cannot have two records with the same age.

```
CREATE TABLE CUSTOMERS(
   ID   INT              NOT NULL,
   NAME VARCHAR (20)     NOT NULL,
   AGE  INT              NOT NULL UNIQUE,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

Adding Constraint

```
ALTER TABLE CUSTOMERS
   ADD CONSTRAINT myUniqueConstraint UNIQUE(AGE, SALARY);
```

DROP a UNIQUE Constraint

To drop a UNIQUE constraint, use the following SQL query.

```
ALTER TABLE CUSTOMERS
   DROP CONSTRAINT myUniqueConstraint;
```

Example [Unique:Table level constraint]

```
CREATE TABLE primary_student
(
Roll_Number Number(4),  Name Char(15), Class Char(10),
Marks Number(4), DOB Date,
Unique(Roll_Number, Class)
);
```

OR

```
CREATE TABLE primary_student
(
Roll_Number Number(4), Name Char(15), Class Char(10), Marks Number(4),        DOB
Date,
Constraint RNO_Class_Unique  Unique(Roll_Number, Class)
);
```

DEFAULT constraint

The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value.

- The default value constraint allows the user to insert the values in the columns where the user do not want to insert the value.
- This is not actually a constraint and used to insert default value if value is missing for this column.
- The datatype of the default value should match the datatype of the column.
- Syntax:
- Columnname datatype (size) default value

Example [Default: Column level constraint]

```
CREATE TABLE student
   (
   Roll_Number Number(4) Primary key,
   Name Char(15),
   Class Char(10) Check (class in ('BE','ME','MCA')),
   Marks Number(4) default 60,
   DOB Date
   );
```

DEFAULT constraint

Example

For example, the following SQL creates a new table called CUSTOMERS and adds five columns.

Here, the **SALARY column is set to 5000.00 by default,** so in case the INSERT INTO statement does not provide a value for this column, then by default this column would be set to 5000.00.

```
CREATE TABLE CUSTOMERS(
   ID   INT              NOT NULL,
   NAME VARCHAR (20)     NOT NULL,
   AGE  INT              NOT NULL,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2) DEFAULT 5000.00,
   PRIMARY KEY (ID)
);
```

if the CUSTOMERS table has already been created, then to add a DEFAULT constraint to the SALARY column, you would write a query like the one which is shown in the code block below.

```
ALTER TABLE CUSTOMERS
MODIFY SALARY  DECIMAL (18, 2) DEFAULT 5000.00;
```

Drop Default Constraint

To drop a DEFAULT constraint, use the following SQL query.

```
ALTER TABLE CUSTOMERS
   ALTER COLUMN SALARY DROP DEFAULT;
```

Entity Integrity OR Key/Primary Key constraints

- **An attribute that can uniquely identify a tuple in a relation is called the key of the table.**
- The value of the attribute for different tuples in the relation has to be unique.
- Entity integrity constraints are rules for primary keys:

- **The primary key cannot have a null value.**
- **If the primary key is a composite key, none of the fields in the key can contain a null value.**

Entity Integrity OR Key/Primary Key constraints

Example:

- In the given table, CustomerID is a key attribute of Customer Table.
- It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =" Google".

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Primary Key Constraint

- It is a combination of both the unique key constraint as well as the not null constraint.
- A primary key is used to identify each row of the table uniquely.
- A primary key may be either consists of a single field (Simple key) or group of fields (Composite Primary Key).
- Oracle enforces all PRIMARY KEY constraints using indexes.
- Syntax:

  Columnname datatype (size) [constraint constraintname] PRIMARY KEY

Example [Primary Key: Column level Constraint]

```
CREATE TABLE student
(
Roll_Number Number(4) Primary Key,
Name Char(15), Class Char(10), Marks Number(4),
DOB Date
);
```

OR

```
CREATE TABLE student
(
Roll_Number Number(4) Constraint RNO_PK Primary Key,
Name Char(15), Class Char(10), Marks Number(4),
DOB Date
);
```

Example [Primary Key: Table level Constraint]

```
CREATE TABLE primary_student
(
Roll_Number Number(4), Name Char(15), Class Char(10),
Marks Number(4), DOB Date,
Primary Key(Roll_Number, Class)
);
```

OR

```
CREATE TABLE primary_student
(
Roll_Number Number(4), Name Char(15), Class Char(10), Marks Number(4), DOB Date,
Constraint RNO_Class_PK Primary Key(Roll_Number, Class)
);
```

Referential integrity constraints

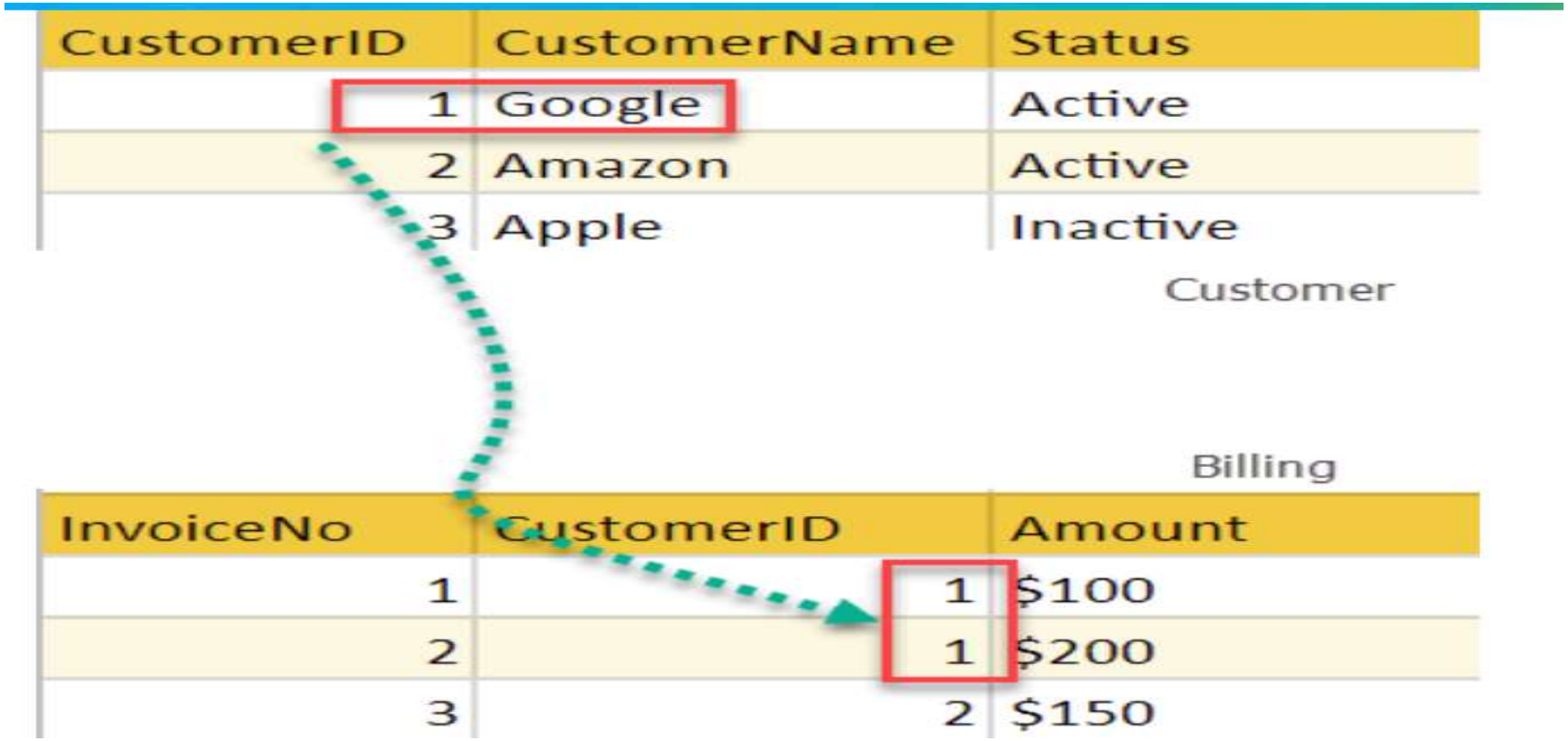
- Referential integrity constraints is base on the **concept of Foreign Keys.**

- A foreign key is an important attribute of a relation **which should be referred to in other relationships.**

- Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

Referential Integrity

- Referential Integrity can help to avoid data input errors, voids data inconsistency and quality problems.
- In a relational database, **referential integrity means that a foreign key value cannot be entered in one table unless it matches an existing primary key in another table.**
- You cannot change a primary key that has matching child table records
  - A child table that has a foreign key for a different record
- Referential integrity also can prevent the deletion of a record has a primary key that matches foreign keys in another table.

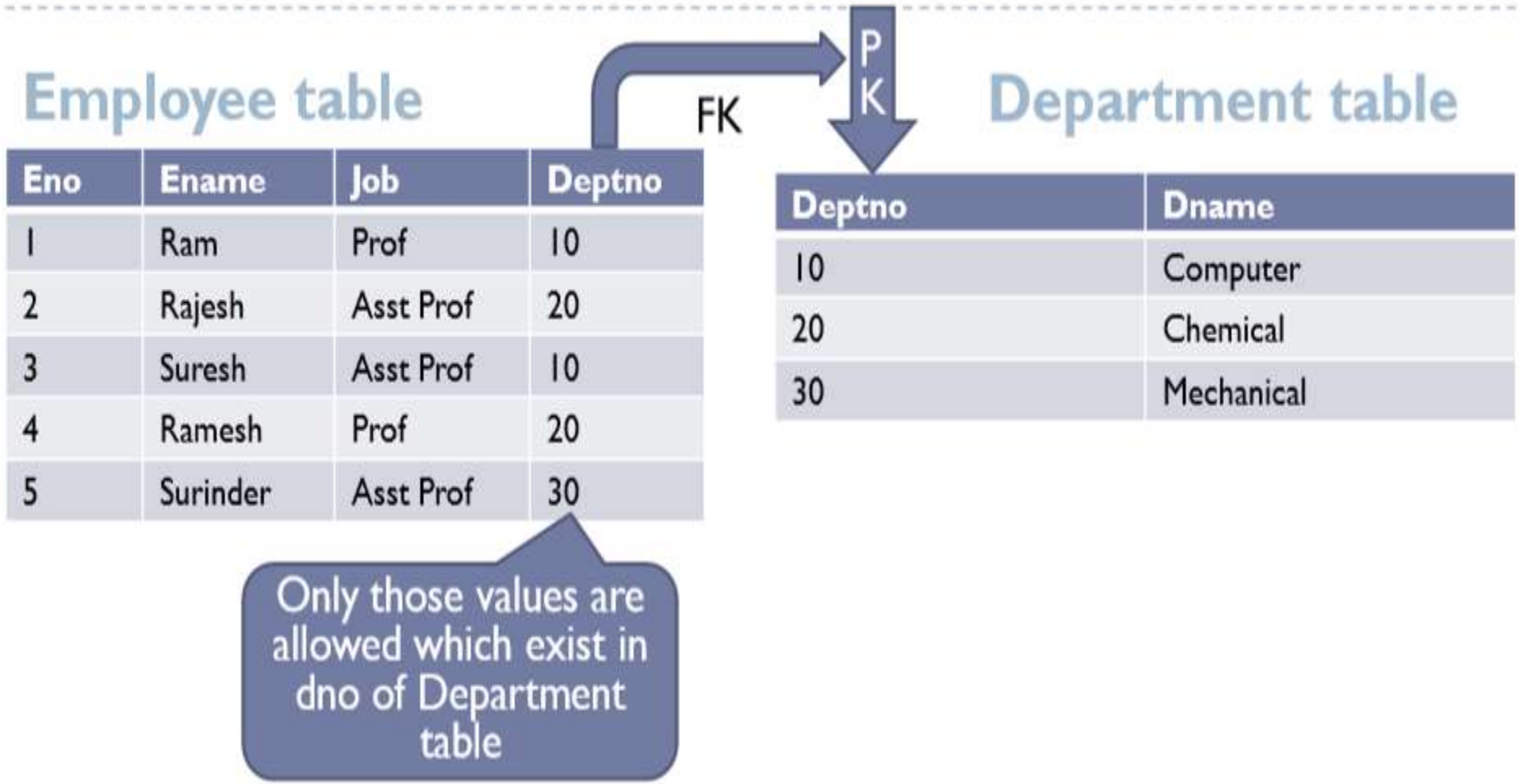


In the above example, we have 2 relations, Customer and Billing.

Tuple for CustomerID =1 is referenced twice in the relation Billing. So we know CustomerName=Google has billing amount $300

Foreign Key Constraint

Concept of Foreign Key: Case Study EMP-DEPT



Eno	Ename	Job	Deptno
1	Ram	Prof	10
2	Rajesh	Asst Prof	20
3	Suresh	Asst Prof	10
4	Ramesh	Prof	20
5	Surinder	Asst Prof	30

Deptno	Dname
10	Computer
20	Chemical
30	Mechanical

Concept of Foreign Key: Case Study EMP-DEPT

Employee table

Eno	Ename	Job	Deptno
1	Ram	Prof	10
2	Rajesh	Asst Prof	20
3	Suresh	Asst Prof	10
4	Ramesh	Prof	20
5	Surinder	Asst Prof	30
6	Rupinder	Prof	40

FK

PK

Department table

Deptno	Dname
10	Computer
20	Chemical
30	Mechanical

No corresponding department exists in University. Not allowed.

Deptno of Employee table refers to other table's PK for its validation, so it is known as FK

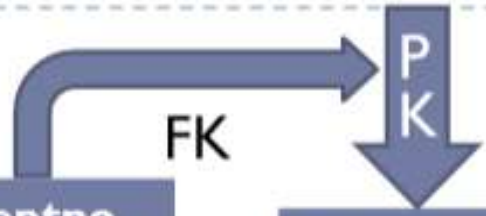
Foreign key Constraint

- Used to enforce referential integrity of data and establishes a relationship among tables.
- A foreign key may be a single column or the combinations of columns, which derive their values, based on the primary key of some other table.
- A table in which foreign key is present is known as child table and to which it refers is known as master or parent table.
- Its Syntax is:

  Columnname datatype (size) references tablename [(columnname)]

  [ON DELETE RESTRICT/ON DELETE CASCADE/ON DELETE SET NULL]

Syntax: Case Study EMP-DEPT



Employee table

Eno	Ename	Job	Deptno
1	Ram	Prof	10
2	Rajesh	Asst Prof	20
3	Suresh	Asst Prof	10
4	Ramesh	Prof	20
5	Surinder	Asst Prof	30

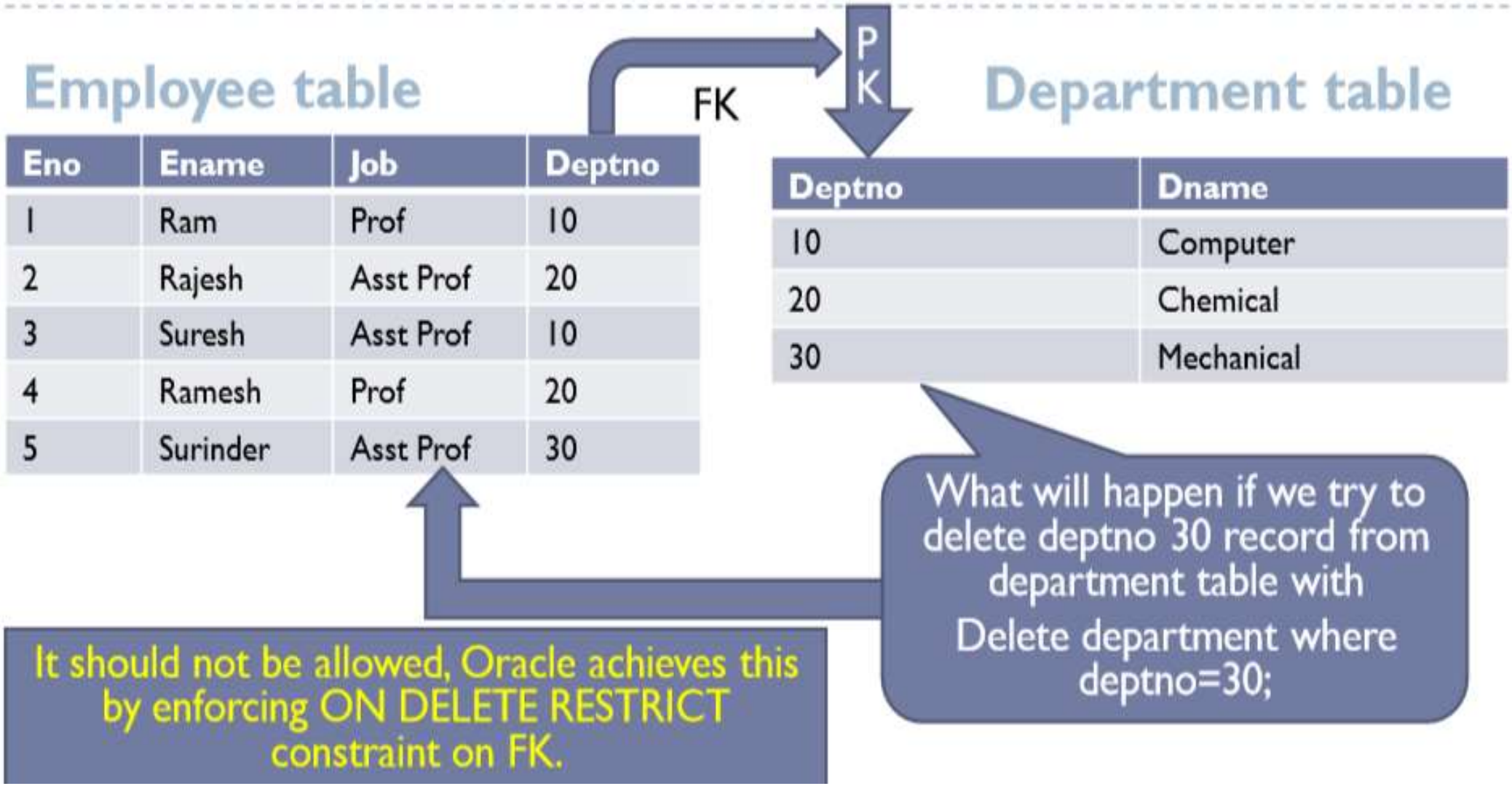
Department table

Deptno	Dname
10	Computer
20	Chemical
30	Mechanical

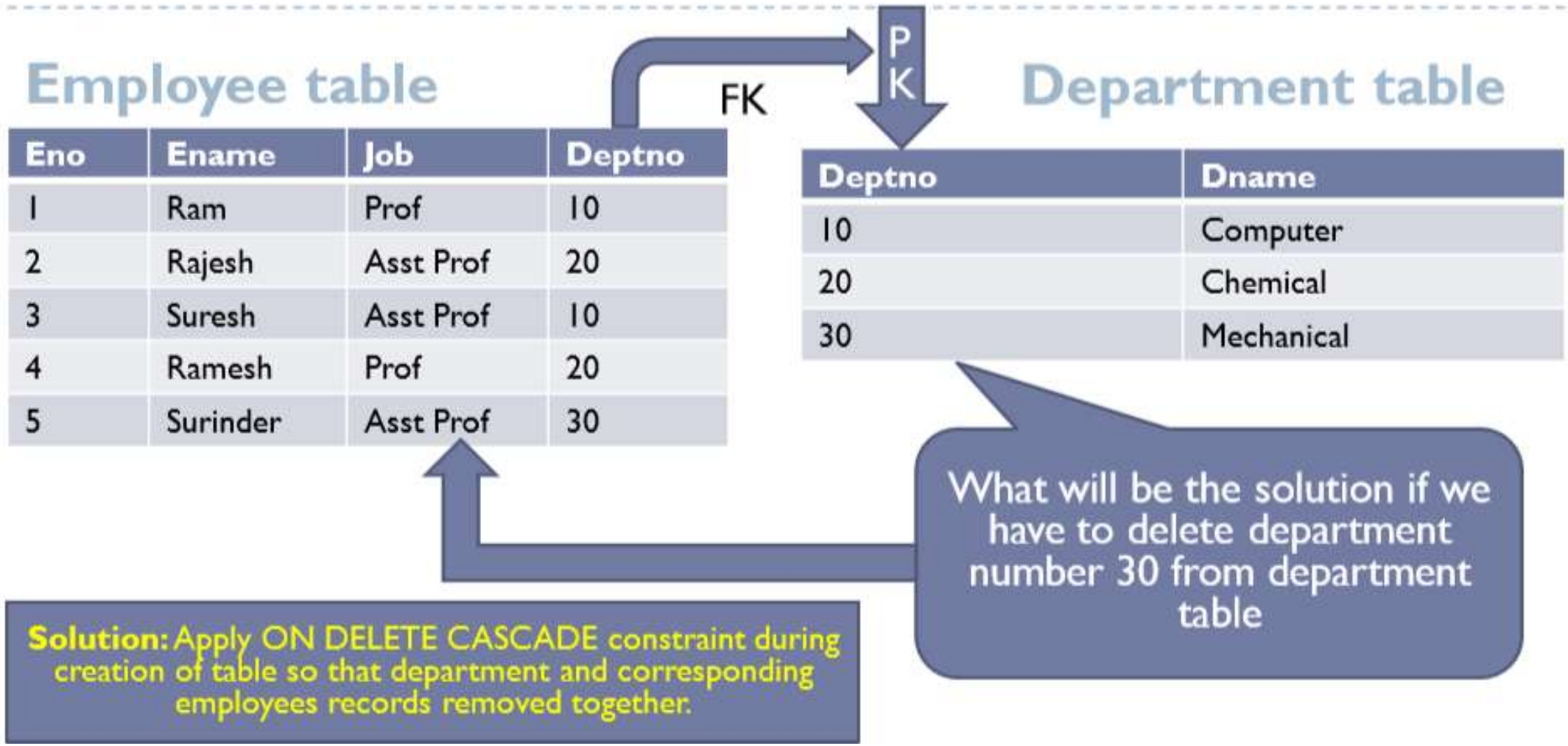
1 Create table department
(deptno number(2) primary key,
dname char(20));

2 Create table employee
(eno number(2) primary key,
ename char(20),job char(10),
deptno number(2) references department(deptno));

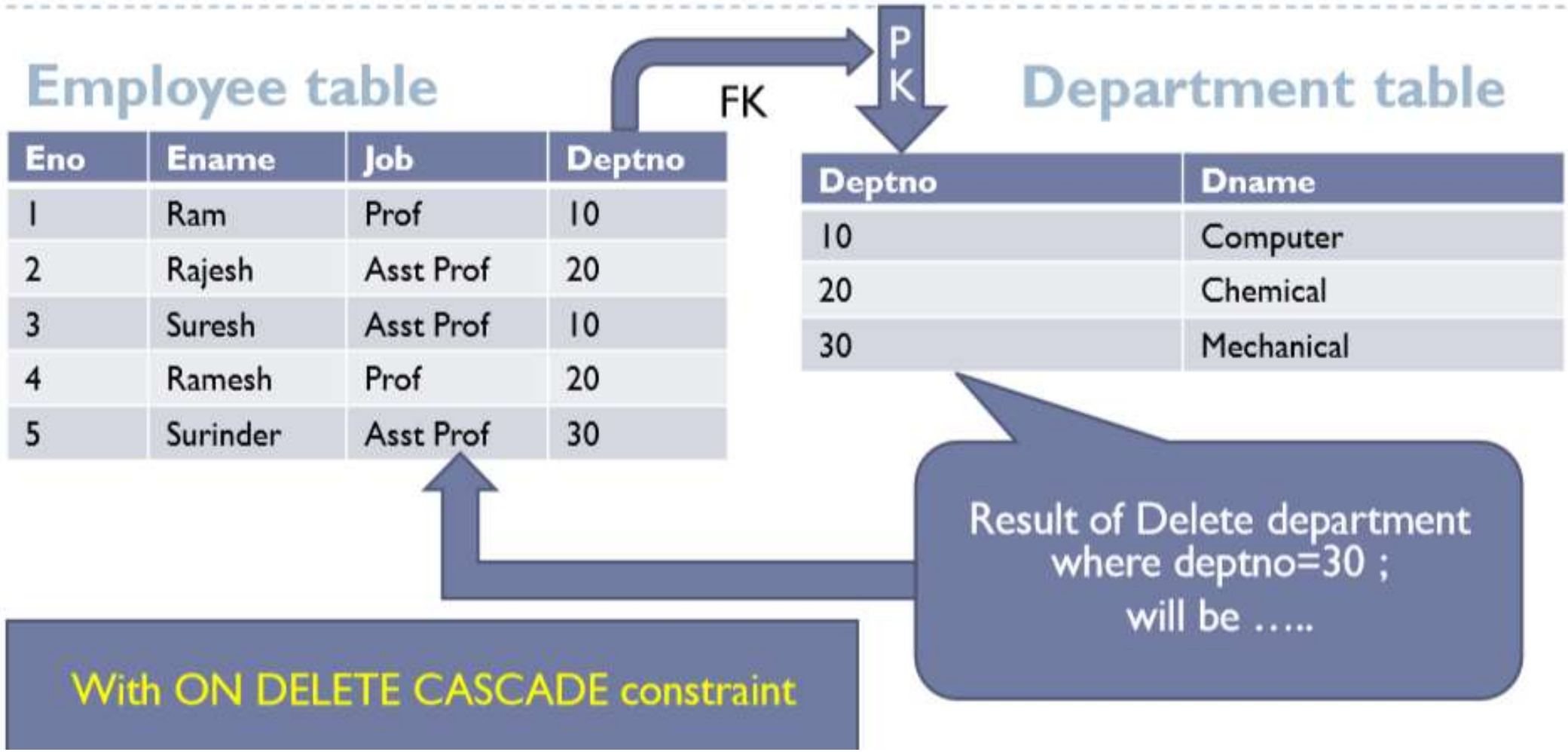
Concept of Foreign Key: Case Study EMP-DEPT



Concept of Foreign Key: Case Study EMP-DEPT



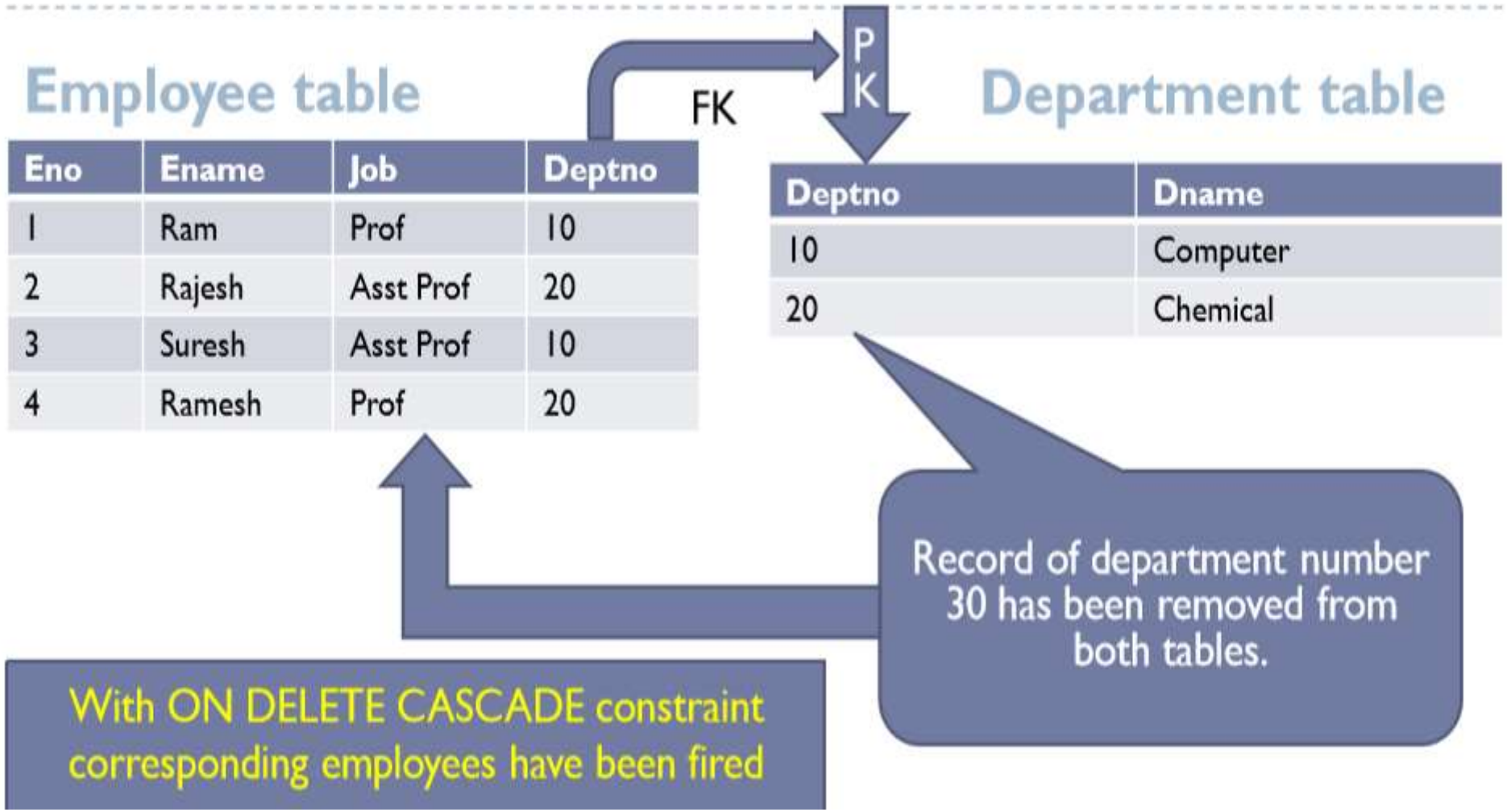
Concept of Foreign Key: Case Study EMP-DEPT



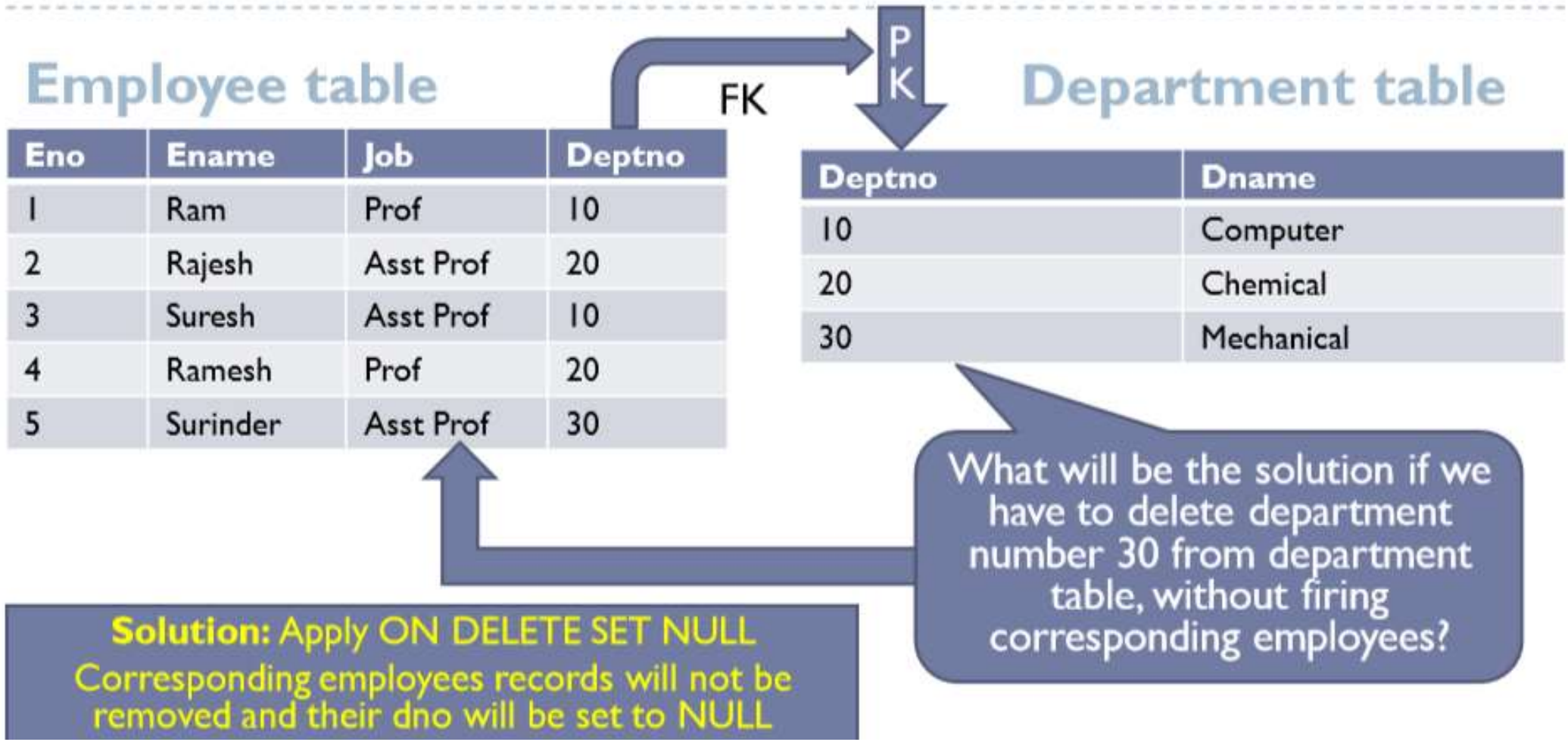
Eno	Ename	Job	Deptno
1	Ram	Prof	10
2	Rajesh	Asst Prof	20
3	Suresh	Asst Prof	10
4	Ramesh	Prof	20
5	Surinder	Asst Prof	30

Deptno	Dname
10	Computer
20	Chemical
30	Mechanical

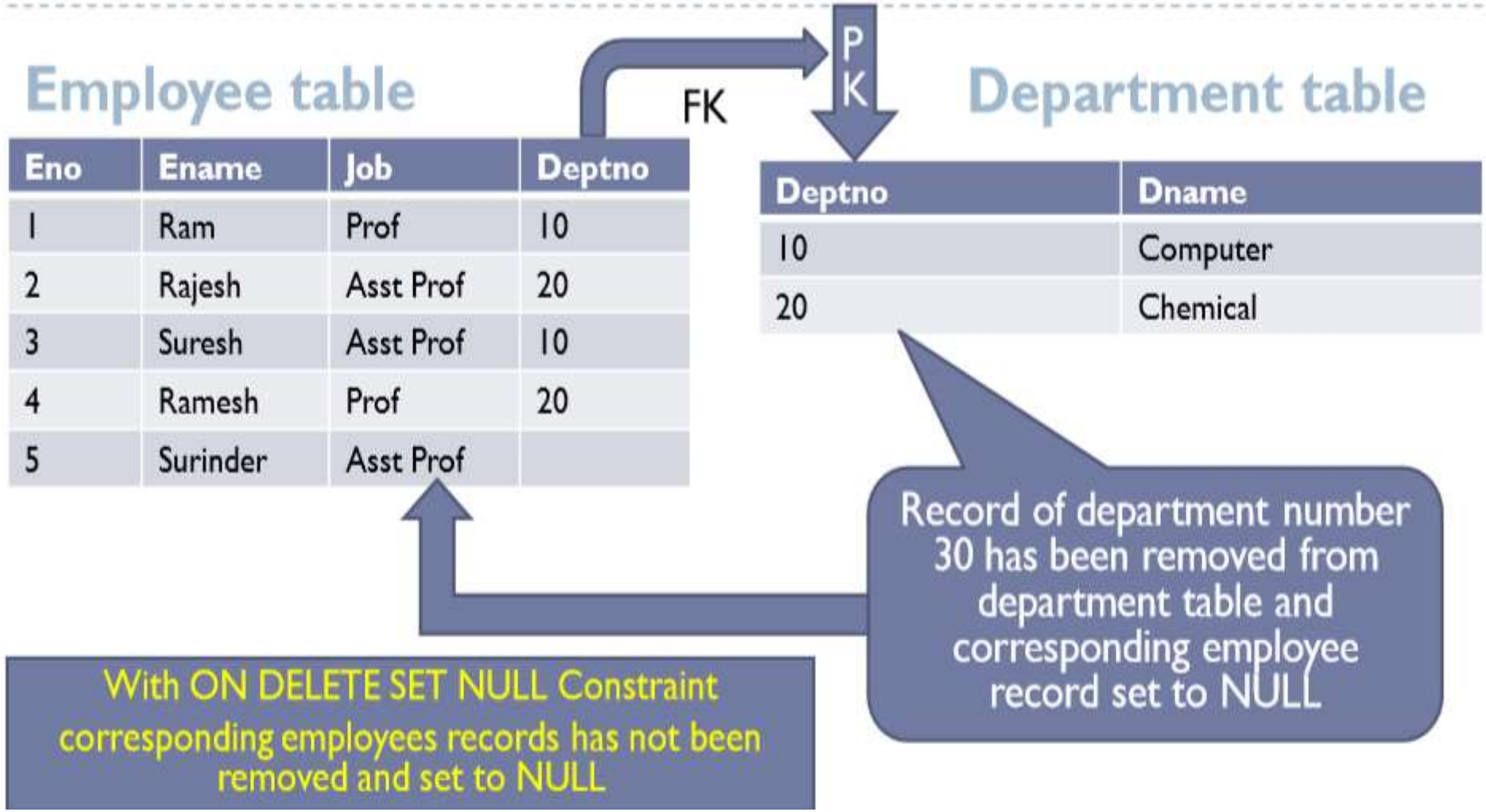
Concept of Foreign Key: Case Study EMP-DEPT



Concept of Foreign Key: Case Study EMP-DEPT



Concept of Foreign Key: Case Study EMP-DEPT



Creating Master and Child tables with ON DELETE RESTRICT (By default)

- Parent Table
- SQL>CREATE TABLE department
  (DEPTNO NUMBER(2) PRIMARY KEY,
  DNAME CHAR(10));
- Child table
- SQL>CREATE TABLE employee(
  empno NUMBER(4) PRIMARY KEY,
  ename CHAR(20), job CHAR(10),
  deptno NUMBER(2) REFERENCES department(deptno)
  ON DELETE RESTRICT);

Creating Master and Child tables with ON DELETE CASCADE

- Parent Table
- SQL>CREATE TABLE department
  (DEPTNO NUMBER(2) PRIMARY KEY,
  DNAME CHAR(10));
- Child table
- SQL>CREATE TABLE employee(
  empno NUMBER(4) PRIMARY KEY,
  ename CHAR(20), job CHAR(10),
  deptno NUMBER(2) REFERENCES department(deptno)
  ON DELETE CASCADE);

Creating Master and Child tables with ON DELETE SET NULL

- Parent Table
- SQL>CREATE TABLE department
  (DNO NUMBER(2) PRIMARY KEY,
  DNAME CHAR(10));
- Child table
- SQL>CREATE TABLE employee(
  empno NUMBER(4) PRIMARY KEY,
  ename CHAR(20), job CHAR(10),
  dno NUMBER(2) REFERENCES department(deptno)
  ON DELETE SET NULL);

USER_CONSTRAINTS and USER_CONS_COLUMNS tables to list constraints details

- To see the name of the constraints on a table you can query USER_CONSTRINTS table (maintains internally by Oracle).

To query it:

- SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME='NAME_OF_TABLE';

For example:

- SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME='STUDENT';
- SELECT * FROM USER_CONS_COLUMNS WHERE TABLE_NAME='STUDENT';

Viewing Constraints

Query the USER_CONSTRAINTS table to view all constraint definitions and names.

```
SELECT    constraint_name, constraint_type,
          search_condition
FROM      user_constraints
WHERE     table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	C	SEARCH_CONDITION
EMP_LAST_NAME_NN	C	"LAST_NAME" IS NOT NULL
EMP_EMAIL_NN	C	"EMAIL" IS NOT NULL
EMP_HIRE_DATE_NN	C	"HIRE_DATE" IS NOT NULL
EMP_JOB_NN	C	"JOB_ID" IS NOT NULL
EMP_SALARY_MIN	C	salary > 0
EMP_EMAIL_UK	U	

...

Viewing the Columns Associated with Constraints

View the columns associated with the constraint names in the `USER_CONS_COLUMNS` view.

```
SELECT    constraint_name, column_name
FROM      user_cons_columns
WHERE     table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	COLUMN_NAME
EMP_DEPT_FK	DEPARTMENT_ID
EMP_EMAIL_NN	EMAIL
EMP_EMAIL_UK	EMAIL
EMP_EMP_ID_PK	EMPLOYEE_ID
EMP_HIRE_DATE_NN	HIRE_DATE
EMP_JOB_FK	JOB_ID
EMP_JOB_NN	JOB_ID

...

Adding a Constraint Syntax

Use the `ALTER TABLE` statement to:

- Add or drop a constraint, but not modify its structure
- Enable or disable constraints
- Add a `NOT NULL` constraint by using the `MODIFY` clause

```
ALTER TABLE  table
ADD [CONSTRAINT constraint] type (column);
```

Adding a Constraint

Add a `FOREIGN KEY` constraint to the `EMPLOYEES` table indicating that a manager must already exist as a valid employee in the `EMPLOYEES` table.

```
ALTER TABLE      employees
ADD CONSTRAINT   emp_manager_fk
  FOREIGN KEY(manager_id)
  REFERENCES employees(employee_id);
Table altered.
```

Cascading Constraints

- The `CASCADE CONSTRAINTS` clause is used along with the `DROP COLUMN` clause.
- The `CASCADE CONSTRAINTS` clause drops all referential integrity constraints that refer to the primary and unique keys defined on the dropped columns.
- The `CASCADE CONSTRAINTS` clause also drops all multicolumn constraints defined on the dropped columns.

Disabling Constraints

- Execute the `DISABLE` clause of the `ALTER TABLE` statement to deactivate an integrity constraint.
- Apply the `CASCADE` option to disable dependent integrity constraints.

```
ALTER TABLE       employees
DISABLE CONSTRAINT  emp_emp_id_pk CASCADE;
Table altered.
```

Enabling Constraints

- Activate an integrity constraint currently disabled in the table definition by using the ENABLE clause.

```
ALTER TABLE          employees
ENABLE CONSTRAINT    emp_emp_id_pk;
Table altered.
```

- A UNIQUE or PRIMARY KEY index is automatically created if you enable a UNIQUE key or PRIMARY KEY constraint.