Referential Integrity (V)

- Standard behavior of the DBMS when violating a referential integrity constraint: rejection of the action that caused the violation, i.e., the transaction performing the update action is rolled back

- Attempt to update or delete a tuple in the referenced relation *R* that has a matching tuple in the referencing relation *S* depends on the referential action specified using the **on delete** or **on update** sub-clauses of the **foreign key** clause

- Possible referential actions for the **on delete** sub-clause
  - **on delete cascade**: The tuple (including its primary key) of the referenced relation *R* as well as all the matching tuples (including their foreign keys) in a referencing relation *S* are deleted.
  - **on delete set null**: The tuple of the referenced relation *R* is deleted, and the foreign key values in all matching tuples of a referencing relation *S* are set to *null*. This option is only valid if the foreign key attributes in *S* do not have the *not null* constraint specified.

Referential Integrity (VI)

- Possible referential actions for the **on delete** sub-clause (*continued*)
  - **on delete set default**: The tuple of the referenced relation *R* is deleted, and the foreign key values in all matching tuples of a referencing relation *S* are set to their specified default values. This option is only valid if all foreign key attributes in *S* have a default value specified.
  - **on delete no action**: If a tuple of a relation *S* exists that references a tuple of a relation *R*, the deletion of the tuple in *R* is rejected. This is the default setting if the **on delete** rule is omitted.
- Possible referential actions for the **on update** sub-clause
  - **on update cascade**: A change of the value of the primary key attributes in a tuple of the referenced relation *R* is propagated to all tuples in a referencing relation *S* with the same value in their foreign key attributes.
  - **on update set null**: The value of the primary key attributes in the tuple of the referenced relation *R* is changed. The foreign key values in all matching tuples of a referencing relation *S* are set to *null*. This option is only valid if the foreign key attributes in *S* do not have the *not null* constraint specified.

Referential Integrity (VII)

- Possible referential actions for the **on update** sub-clause (*continued*)
  - **on update set default**: The value of the primary key attributes in the tuple of the referenced relation *R* is changed. The foreign key values in all matching tuples of a referencing relation *S* are set to their specified default values. This option is only valid if all foreign key attributes in *S* have a default value specified.
  - **on update no action**: If a tuple of a relation *S* exists that references a tuple of a relation *R*, the update of the primary key attribute values of the tuple in *R* is rejected. This is the default setting if the **on update** rule is omitted.
- Examples
  - **create table** lectures
    (...,
    foreign key(held_by) **references** professors(pers-id)
    on delete cascade
    on update cascade);

Referential Integrity (VIII)

- Examples (*continued*)
  - The deletion of a tuple with the key *pers-id* in *professors* is permitted and has the effect on *lectures* that there all referencing tuples are deleted. This enables the realization of a dependent relationship.
  - A change of a value of the attribute *pers-id* in *professors* is propagated to all matching values of the foreign key attribute *held_by* in *lectures*

  - **create table** lectures
    (...,
    foreign key(held_by) **references** professors(pers-id)
    on delete set null
    on update set null);
    - The value of the attribute *held_by* in *lectures* is set to *null*, if the referenced tuple with the key *pers-id* in *professors* is deleted
    - The value of the attribute *held_by* of all referencing tuples in *lectures* is set to *null*, if the value of the primary key attribute *pers-id* of the referenced tuple in *professors* is changed

General Integrity

- An assertion is a predicate that expresses a condition which is to be always satisfied by a database system
- Domain constraints and referential ICs are special kinds of assertions
- There are also conditions which cannot be expressed with these two kinds like conditions with respect to several relations
- Syntax: **create assertion** <assertion name> **check** <condition>
- Example: There must be at least four professors in order to maintain teaching

  create assertion AlwaysFourProfessors

  check (4 <= (**select count**(*) **from** professors))
- The DBMS tests an assertion for validity first, and if the assertion is valid, future modifications of the database are only allowed if the assertion is not violated
- Complex assertions can lead to an overhead; they should be used with care
- Most DBS have not implemented assertions

Triggers (I)

- A trigger
  - is a user-defined procedure that the DBMS executes automatically if a certain condition is fulfilled or as a side effect of a modification of the database
  - is a general and powerful mechanism for maintaining data(base) consistency, monitoring database updates, and updating database statistics
  - cannot only deploy check functions but also computation functions
- A trigger is based on the ECA (Event-Condition-Action) model and has three components:
  - The event(s) to which it reacts
    - These are usually database manipulation operations (insert, delete, update) that are explicitly applied to the database
    - Person who writes the trigger must ensure that all possible events are accounted for
    - Trigger must specify if it should be executed *before* or *after* an event

Triggers (II)

- The condition that determines whether the action should be executed
  - If no condition is specified, the action will be executed once the event occurs
  - If a condition is specified in the *when* clause of the trigger, it is first evaluated, and only if it evaluates to *true*, the action will be executed
- The action(s) to be taken
  - The action is usually a sequence of SQL statements
  - It can also be a database transaction or an external program that will be automatically executed

- Part of the SQL syntax for creating a trigger in Oracle
  - **create trigger** <trigger name>
    {**before** | **after**}
    {**insert** | **delete** | **update** [**of** <column list>]} **on** <table name>
    [**for each row**]
    [**when** <condition>]
    <trigger actions>;

Triggers (III)

- In Oracle the trigger actions are implemented in PL/SQL, which is Oracle's database application programming language
- Example: Trigger preventing that professors can be demoted by a rank

```
create trigger noDemotion
before update on professors
for each row
when (:old.rank is not null)
begin
    if :old.rank = "C3" and :new.rank = "C2" then :new.rank = "C3" end if;
    if :old.rank = "C4" then :new.rank = "C4" end if;
    if :new.rank is null then :new.rank = :old.rank end if;
end
```

- The clause **create trigger** provides a trigger specification
- The time of releasing the trigger body can be **before** or **after** the data manipulation operation that has released the trigger

Triggers (IV)

- Trigger events
  - **update** [**of** <column1, column2, ...>] **on** <relation name>
  - **insert on** <relation name>
  - **delete on** <relation name>

- A trigger can be defined for one or several events. In case of several events, a case distinction can be expressed in the body through the clauses
  - **if updating** [<column1, column2, ...>] **then** ...
  - **if inserting then** ...
  - **if deleting then** ...

- Trigger type
  - A statement-level trigger (default) is released exactly *once* either before or after the respective event.
  - A row-level trigger is indicated by the **for each row** clause and signifies that the trigger is executed *for each changed tuple* in the table

Triggers (V)

- Distinguishing tuples before and after a change in a row-oriented trigger
  - Keyword **:old** refers to a deleted tuple or to a tuple before it was updated
  - Keyword **:new** refers to a newly inserted or newly updated tuple
  - This is the only possibility to address tuples before or after a change
  - Another access to the relation is not possible any more, even if the relation name would be used in the respective block

- Trigger restriction
  - The **when** predicate specifies a condition that must be satisfied to release the execution of the trigger body
  - If a row-oriented trigger is used, the new resp. old tuple of the relation can be addressed by the keywords **:new** resp. **:old**

- Trigger body / trigger actions
  - SQL queries
  - Oracle-specific PL/SQL commands with the aforementioned extensions

Triggers (VI)

- Example: Protocol of the changes of the attribute *salary* of a relation *Persons*

```
create trigger StoreSalary
before update on Persons
for each row
when (:old.salary > 1500)
begin insert into diff values (:old.salary, :new.salary, sysdate) end;
```

- Example: Check at insertion time that a salary increase is inapplicable to persons with a monthly salary greater than $10,000

```
create trigger CheckSalary
before update on Persons
for each row when (:new.salary > 10000)
begin
   :new.salary := :old.salary;     // assignment only possible for before update
end;
```