RELAZIONE PROGETTO BASI DI DATI
2022

Riccardo Lambertini | riccardo.lambertini5@studio.unibo.it | 0000934793
Kevin Renda | kevin.renda@studio.unibo.it | 0000915714
Andrea Bianchi | andrea.bianchi26@studio.unibo.it | 0000948164

1. Raccolta/Analisi dei requisiti

Corso di Basi di Dati
CdS Informatica per il Management
TRACCIA di PROGETTO, A.A. 2021/2022

PREMESSA.
Si vuole realizzare la piattaforma **CONFVIRTUAL** per la gestione di conferenze online durante il periodo della pandemia COVID-19. La piattaforma supporta l'organizzazione di conferenze svolte mediante video-conferenze da remoto. In particolare, si consente agli utenti organizzatori la creazione di conferenze con sessioni di presentazioni di articoli/tutorial, e relativi link alle stanze Teams per la partecipazione alle stesse. Gli utenti possono registrarsi alle conferenze, aggiungere i propri dati nel caso di speaker/presenter, interagire con altri utenti mediante servizi di messaggistica interni.

SPECIFICA.
La piattaforma **CONFVIRTUAL** deve gestire i dati delle conferenze a tema informatica/tecnologie digitali. Ogni conferenza dispone di un nome (es. "International Conference on Database Systems"), un acronimo (es. "DB"), un anno di edizione, delle date di svolgimento (uno o più giorni), un'immagine per il logo, uno campo `svolgimento` (campo enum con due possibili valori: "`Attiva`" o "`Completata`"), un campo #`totale_sponsorizzazioni`. Anno di edizione ed acronimo (es. "DB 2021") sono univoci nella piattaforma. La conferenza può disporre di uno o più sponsor; ogni sponsor dispone di un nome, un'immagine per il logo, ed eroga un importo per la sponsorizzazione di ogni convegno in cui appare. Lo stesso sponsor può apparire in più conferenza. Per ogni giorno di svolgimento della conferenza, si vuole tenere traccia del programma giornaliero. Quest'ultimoè strutturato in una o più sessioni. Ogni sessione ha un codice (univoco, a livello globale), un titolo (stringa di 100 caratteri), un campo #`numero_presentazioni`, un'ora di inizio un'ora di fine, ed un link alla stanza Teams per lo svolgimento; ogni sessione è allocata ad un solo programma giornaliero. Invece, è consentito avere sessioni in parallelo (stessa data ed orario inizio/fine sovrapposte). La sessione è composta da una o più presentazioni: ogni presentazione dispone di un codice (univoco), un'ora di inizio, un'ora di fine ed un numero di sequenza all'interno della sessione; non è possibile avere presentazioni che eccedano l'orario di inizio o di fine della propria sessione. Le presentazioni possono appartenere a due tipologie, in maniera esclusiva: presentazioni di articoli o tutorial. Nel primo caso, si vuole tenere traccia del titolo dell'articolo, del numero di pagine e del file PDF, della lista degliautori, e della lista di parole-chiave; ogni parola chiave è una stringa di massimo 20 caratteri, ogni autore dispone di nome e cognome. Inoltre, ogni presentazione di articoli dispone di campo `stato_svolgimento` (campo enum con due possibili valori: "`Coperto`" o "`Non coperto`"). Nel caso di tutorial, si vuole tenere traccia del titolo dell'intervento e dell'abstract (campo testuale di 500 caratteri). La piattaforma consente iscrizioni volontarie da parte degli utenti; ogni utente dispone di

username, password, nome, cognome, data e luogo di nascita. Sono previste tre tipologie di utenti: presenter, speaker ed amministratore. Esistono tuttavia utenti generici che non appartengono ad alcuna delle tre categoria sovra menzionate. Ogni utente può registrarsi alle conferenze presenti nella piattaforma; lo stesso utente può essere registrato ad un numero arbitrario di conferenze, una conferenza può disporre di un numero arbitrario di utenti registrati. Sia i presenter sia gli speaker dispongono di un curriculum vitae(campo testo di 30 caratteri), una foto e un'affiliazione universitaria, con nome dell'università e nome del dipartimento. I presenter possono svolgere SOLO presentazioni di articoli; ogni presentazione di articolo è associata ad uno ed un solo presenter. Un presenter può

presentare più articoli. Un presenter deve essere necessariamente uno degli autori dell'articolo. Gli speaker possono svolgere SOLO presentazioni di tutorial; ogni presentazione di tutorial è associata ad ALMENO uno speaker. Uno speaker può presentare più tutorial. Inoltre è prevista la possibilità per uno speaker (e SOLO a loro) di caricare sulla piattaforma delle risorse aggiuntive per consentire la fruizione del proprio tutorial: una risorsa aggiuntiva consiste in un link Web (es. link a video Youtube), in una breve descrizione di testo e fa riferimento ad un solo tutorial. Un tutorial potrebbe non avere risorse aggiuntive. Gli amministratori (e solo loro) possono creare le conferenze presenti nella piattaforma; ogni conferenza è associata ad ALMENO un amministratore. Un amministratore può creare più conferenze. All'atto della creazione di una nuova conferenza, l'amministratore è anche automaticamente registrato alla stessa (vedi sopra). Gli utenti amministratori (e solo loro) possono inserire una valutazione sulle presentazioni: ogni valutazione dispone di voto (interotra 0 e 10) e note (stringa con massimo 50 caratteri). Ogni sessione del convegno dispone di una chat, con possibilità di invio di messaggi da parte degli utenti (di qualsiasi categoria); ognimessaggio dispone di una data di inserimento ed un testo. Uno stesso utente può inserire più di un messaggio nella stessa chat. La chat consente l'inserimento di messaggi solo nell'orario di inizio della sessione, e si disattiva immediatamente dopo l'orario di fine della stessa. Infine, ogni utente (di qualsiasi categoria) dispone di una propria lista di "presentazioni favorite", ossia un reminder delle presentazioni del convegno che si intende seguire; la lista può includere zero, uno o più elementi.

Si vuole tenere traccia di tutti gli eventi che occorrono nella piattaforma, relativamente all'inserimento di nuovi dati (es. nuovi utenti, nuove conferenze, nuovi messaggi, etc). Tali eventi vanno inseriti, sotto forma di messaggi di testo, all'interno di un log, implementato in un' apposita collezione MongoDB.

Operazioni sui dati[1]:

Operazioni che riguardano tutti gli utenti:

- Autenticazione alla piattaforma
- Registrazione alla piattaforma
- Visualizzazione delle conferenze disponibili
- Registrazione ad una conferenza
- Visualizzazione delle sessioni e delle presentazioni in ogni sessione
- Visualizzazione/inserimento messaggi nella chat di sessione
- Inserimento/visualizzazione lista presentazioni favorite

Operazioni che riguardano SOLO gli utenti AMMINISTRATORE:

- Creazione di una nuova conferenza
- Creazione di una nuova sessione della conferenza
- Inserimento delle presentazioni in una sessione
- Associazione di uno speaker alla presentazione di un tutorial
- Associazione di un presenter alla presentazione di un articolo
- Inserimento/visualizzazione delle valutazioni sulle presentazioni
- Inserimento di uno sponsor/sponsorizzazione

---

[1] La lista contiene le operazioni di base: può essere estesa/modificata a discrezione dello studente.

Operazioni che riguardano SOLO gli utenti PRESENTER:

- Inserimento/modifica del CV/foto ed affiliazione universitaria

Operazioni che riguardano SOLO gli utenti SPEAKER:

- Inserimento/modifica del CV/foto ed affiliazione universitaria
- Inserimento/modifica delle risorse aggiuntive per il proprio tutorial

Statistiche (visibili da tutti gli utenti nella pagina iniziale):

- Visualizzare il numero (totale) di conferenze registrate
- Visualizzare il numero (totale) di conferenze attive
- Visualizzare il numero (totale) di utenti registrati
- Visualizzare la classifica dei presenter/speaker sulla base del voto medio

Popolamento della piattaforma:

A scelta dell'utente, non costituisce oggetto di valutazione. Si può prendere a modello qualche conferenza elencata in: http://www.wikicfp.com/cfp/

Vincoli sull'implementazione:

- Implementare tutte le operazioni sui dati (ove possibile) attraverso **stored procedure**.
- Utilizzare un **trigger** per implementare l'operazione cambio di `stato_svolgimento` di una presentazione di articolo, portandolo da "`Non coperto`" a "`Coperto`" quando si inserisce un presenter valido per quella presentazione.
- Utilizzare un **trigger** per implementare l'operazione di aggiornamento del campo `#numero_presentazioni` ogni qualvolta si aggiunge una nuova presentazione ad una sessione della conferenza.
- Utilizzare un **event** per modificare il campo `svolgimento` di una conferenza**. L'evento setta il campo a** "`Completata`" non appena la data corrente eccede di un giorno l'ultima data di svolgimento di una conferenza.
- Il link alla stanza Teams può essere un campo stringa generico; non si richiede di implementare alcun collegamento con piattaforme esterne.

Tabelle dei volumi:

- **Inserire pesi e coefficienti per l'analisi dei costi a discrezione di ogni gruppo**
- **Tabella dei volumi:** 10 conferenze, 5 sponsor per conferenza
- Valutare se la seguente **ridondanza**:

  campo `#totale_sponsorizzazioni` **relativo ad ogni conferenza**

  debba essere **tenuta o eliminata**, sulla base delle seguenti operazioni:
  - Inserire una nuova sponsorizzazione per una data conferenza (2 volte/mese, interattiva)
  - Rimuovere una conferenza e tutte le proprie sponsorizzazioni (1 volte/mese, batch)
  - Calcolare e visualizzare il totale delle sponsorizzazioni di una data conferenza (4 volte/mese, interattiva)

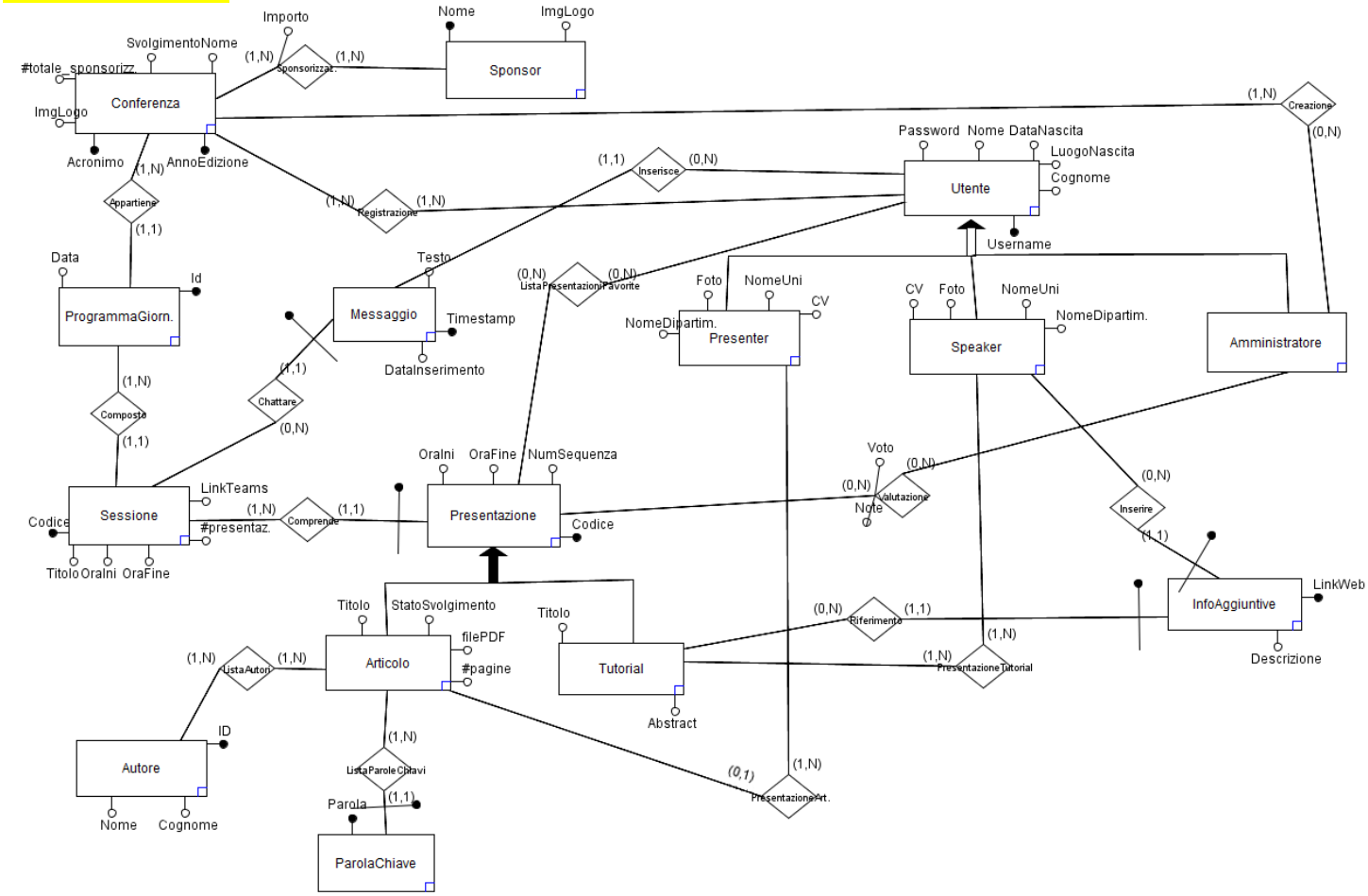
Bonus:

I bonus seguenti sono assegnati SOLO se le componenti back-end ed il database sono stati progettati ed implementati correttamente:

- (**+1pt**) Utilizzo di **librerie CSS** per la realizzazione del front-end Web (es. Bootstrap https://getbootstrap.com)
- (**per la lode**) Implementare un sistema di **clustering** basato su algoritmo di K-Means, attraverso il quale si segmentano gli utenti non amministratori, sulla base della loro età, categoria e numero di registrazioni alle conferenze. Visualizzare -tramite apposita funzionalità nella piattaforma, disponibile SOLO per amministratori- l'elenco degli utenti che appartiene a ciascun cluster. Deve essere possibile configurare il numero di cluster da creare (K) dall'interfaccia della piattaforma.

2. *Progettazione Concettuale*

DIAGRAMMA E-R



DIZIONARIO DELLE ENTITA':

ENTITA'	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORE
Conferenza	Conferenze a tema informatico/tecnologie digitali	Nome, Svolgimento, #totale_sponsorizz. ImgLogo, Acronimo, AnnoEdizione	Acronimo, AnnoEdizione
Sponsor	Sponsor delle conferenze	Nome, ImgLogo	Nome
ProgrammaGiornaliero	Programma del giorno di una conferenza	Id, Data	Id
Sessione	Sessione di un programma giornaliero	Codice, LinkTeams, #presentaz. , OraIni, OraFine, Titolo	Codice
Presentazione	Presentazione di un tema appartenete a una conferenza	Codice, OraIni, OraFine, NumSequenza,	CodiceSessione, Codice
Articolo	Tipo di presentazione	Titolo, StatoSvolgimento, filePDF, #pagine	CodiceSessione, Codice
Tutorial	Tipo di presentazione	Titolo, Abstract	CodiceSessione, Codice
ParolaChiave	Parole importanti di un articolo	Parola	Parola, CodiceSessione, Codice
Autore	Autore/i di uno o più articoli	Id, Nome, Cognome	ID
Messaggio	Messaggio/i scambiati nella chat di una sessione	Testo, DataInserimento, Timestamp	Timestamp, CodiceSessione
Utente	Utente della piattaforma	Password, Nome, Cognome, DataNascita, LuogoNascita, Username	Username
Presenter	Tipo di utente	Foto, NomeUni, CV, NomeDipartimento	Username
Speaker	Tipo di utente	Foto, NomeUni, CV, NomeDipartimento	Username
Amministratore	Tipo di utente		Username
InfoAggiuntive	Informazioni aggiuntive per i tutorial	LinkWeb, Descrizione	LinkWeb, Username, CodiceSessione, Codice

DIZIONARIO DELLE RELAZIONI:

RELAZIONE	DESCRIZIONE	COMPONENTI	ATTRIBUTI
Appartiene	Associa Conferenza con ProgrammaGiornaliera	Conferenza, ProgrammaGiornaliero	
Sponsorizzat.	Associa Conferenza con Sponsor	Conferenza, Sponsor	Importo
Composto	Associa ProgrammazioneGiornaliera con Sessione	ProgrammazioneGiornaliera, Sessione	
Comprende	Associa Sessione con Presentazione	Sessione, Presentazione	
ListaAutori	Associa Autori con Articolo	Autori, Articolo	
ListaParoleChiavi	Associa ParolaChiave con Articolo	ParolaChiave, Articolo	
Chattare	Associa Sessione con Messaggio	Sessione, Messaggio	
PresentazioneArt.	Associa Articolo con Presenter	Articolo, Presenter	
ListaPresentazioniFavorite	Associa Utente con Presentazione	Utente, Presentazione	
Registrazione	Associa Conferenza con Utente	Conferenza, Utente	
Inserisce	Associa Messaggio con Utente	Messaggio, Utente	
Creazione	Associa Conferenza con Amministratore	Conferenza, Amministratore	
Valutazione	Associa Amministratore con Presentazione	Amministratore, Presentazione	Voto, Note
Inserire	Associa Speaker con InfoAggiuntive	Speaker, InfoAggiuntive	
PresentazioneTutorial	Associa Tutorial con Speaker	Tutorial, Speaker	

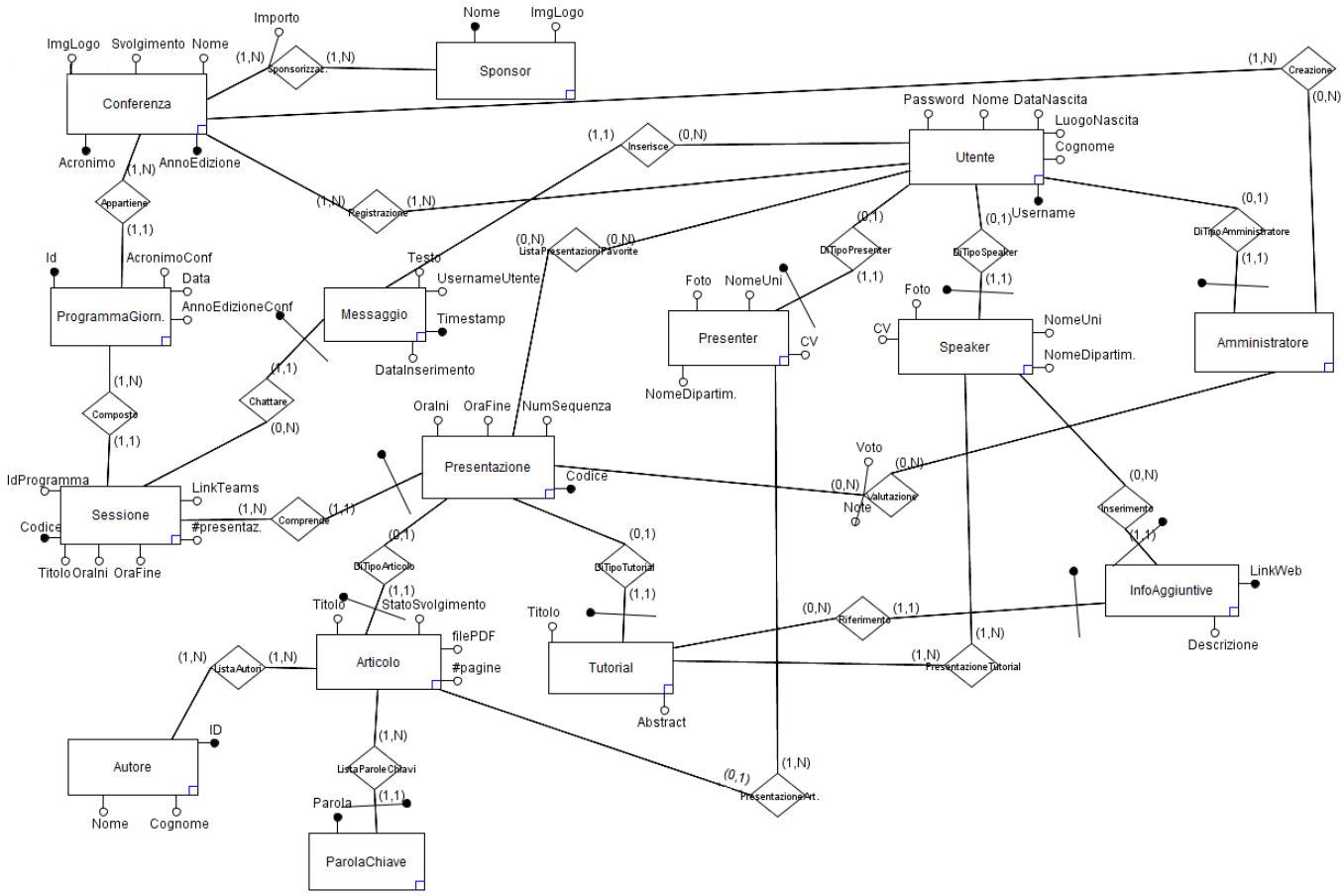
Riferimento	Associa InfoAggiuntive con Tutorial	InfoAggiuntive, Tutorial	

BUSINESS RULES:

1. Il campo "svolgimento" è un *enum* che può avere solo due valori: "*Attiva*" o "*Completata*".
2. Il titolo della sessione è una stringa di massimo 100 caratteri.
3. È consentito avere sessioni in parallelo.
4. Non è possibili avere presentazioni che eccedano l'orario di inizio e di fine della propria sessione.
5. La parola chiave è una stringa di massimo 20 caratteri.
6. Il campo "stato_svolgimento" è un *enum* con valori: "*Coperto*" o "*Non coperto*".
7. L'abstract è un campo testuale di 500 caratteri.
8. Il CV è un campo di testo di 30 caratteri.
9. Un presenter deve essere necessariamente uno degli autori dell'articolo.
10. All'atto della creazione di una nuova conferenza, l'amministratore è anche automaticamente registrato alla stessa.
11. Il voto della valutazione è compreso tra 0 e 10.
12. Le note della valutazione sono una stringa di massimo 50 caratteri.
13. La chat consente l'inserimento di messaggi solo nell'orario di inizio della sessione, e si disattiva immediatamente dopo l'orario di fine della stessa.

3. Progettazione Logica

SCHEMA CONCETTUALE RISTRUTTURATO



ANALISI DELLE RIDONDANZE

Calcolo dei costi per vedere se tenere oppure togliere la ridondanza "#totale_sponsorizzazioni" presente nell'entità CONFERENZA:

Tabella dei volumi: 10 conferenze, 5 sponsor per conferenza.
Dati: Interattiva = 1, Batch = 0.5, alpha = 2.

1) Inserire una nuova sponsorizzazione in una data conferenza (2 volte al mese, interattiva)
Con ridondanza: c(O1) = 2*1*(2*2) = 8
Senza ridondanza: c(O1) = 2*1(2*1) = 4

2) Rimuovere una conferenza e tutte le proprie sponsorizzazioni (1 volta al mese, batch)
Con ridondanza: c(O2) = 1*0.5*(2*(1+5+5)) = 11
Senza ridondanza: c(O2) = 1*0.5*(2*(1+5)) = 6

3) Calcolare e visualizzare il totale delle sponsorizzazioni di una data conferenza (4 volte al mese, interattiva)
Con ridondanza: c(O3) = 4*1*(5) = 20
Senza ridondanza: c(O3) = 4*1*(5) = 20

Risultato con ridondanza -> 39
Risultato senza ridondanza -> 30
Speedup -> 30/39 = 0.77, di conseguenza conviene eliminare la ridondanza "#totale_sponsorizzazioni".

LISTA TABELLE CON VINCOLI DI CHIAVE E DEI VINCOLI INTER-RELAZIONALI

SPONSOR (Nome, ImgLogo)
SPONSORIZZAZIONE (NomeSponsor, AcronimoConferenza, AnnoEdizioneConferenza, Importo)
CONFERENZA (Acronimo, AnnoEdizione, ImgLogo, Nome, Svolgimento)
PROGRAMMA_GIORNALIERO (Id, AcronimoConferenza, AnnoEdizioneConferenza, Data)
SESSIONE (Codice, IdProgramma, LinkTeams, Numpresentazioni, OraFine, OraIni, Titolo)
MESSAGGIO (CodiceSessione, Timestamp, UsernameUtente, Testo, DataInserimento)
PRESENTAZIONE (Codice, CodiceSessione, NumSequenza, OraFine, OraIni)
ARTICOLO (CodicePresentazione, CodiceSessionePresentazione, Numpagine, filePDF, Titolo, StatoSvolgimento, UsernamePresenter)
TUTORIAL (CodicePresentazione, CodiceSessionePresentazione, Titolo, Abstract)
AUTORE (ID, Nome, Cognome)
LISTA_AUTORI (IdAutore, CodiceArticolo, CodiceSessioneArticolo)
PAROLACHIAVE (CodiceArticolo,CodiceSessioneArticolo, Parola)
UTENTE (Username, Password, Nome, Cognome, DataNascita, LuogoNascita)
LISTA_PRESENTAZIONI_FAVORITE (UsernameUtente, CodicePresentazione, CodiceSessionePresentazione)
REGISTRAZIONE (UsernameUtente, AcronimoConferenza, AnnoEdizioneConferenza)
PRESENTER (UsernameUtente, Foto, CV, NomeUni, NomeDipartimento)
SPEAKER (UsernameUtente, Foto, CV, NomeUni, NomeDipartimento)
AMMINISTRATORE (UsernameUtente)
CREAZIONE (UsernameAmministratore, AcronimoConferenza, AnnoEdizioneConferenza)
VALUTAZIONE (UsernameAmministratore, CodiceSessionePresentazione, CodicePresentazione, Voto, Note)
INFO_AGGIUNTIVE (UsernameSpeaker, CodiceTutorial, CodiceSessioneTutorial, LinkWeb, Descrizione)
PRESENTAZIONE_TUTORIAL (UsernameSpeaker, CodiceTutorial, CodiceSessioneTutorial)

Vincoli di integrità referenziale:
SPONSORIZZAZIONE.NomeSponsor -> SPONSOR.Nome
SPONSORIZZAZIONE. AcronimoConferenza -> CONFERENZA.Acronimo
SPONSORIZZAZIONE.AnnoEdizioneConferenza -> CONFERENZA.AnnoEdizione
PROGRAMMA_GIORNALIERO.AcronimoConferenza -> CONFERENZA.Acronimo
PROGRAMMA_GIORNALIERO.AnnoEdizioneConferenza -> CONFERENZA.AnnoEdizione
SESSIONE.IdProgramma -> PROGRAMMA_GIORNALIERO.Id
MESSAGGIO.CodiceSessione -> SESSIONE.Codice
MESSAGGIO.UsernameUtente -> UTENTE.Username
PRESENTAZIONE.CodiceSessione -> SESSIONE.Codice
ARTICOLO.CodicePresentazione -> PRESENTAZIONE.Codice
ARTICOLO.CodiceSessionePresentazione -> PRESENTAZIONE.CodiceSessione
ARTICOLO.UsernamePresenter -> PRESENTER.Username
TUTORIAL.CodicePresentazione -> PRESENTAZIONE.Codice
TUTORIAL.CodiceSessionePresentazione -> PRESENTAZIONE.CodiceSessione
LISTA_AUTORI.IdAutore -> AUTORE.ID
LISTA_AUTORI.CodiceArticolo -> ARTICOLO.CodicePresentazione
LISTA_AUTORI.CodiceSessioneArticolo -> ARTICOLO.CodiceSessione

PAROLACHIAVE.CodiceArticolo -> ARTICOLO.CodicePresentazione
PAROLACHIAVE.CodiceSessioneArticolo -> ARTICOLO.CodiceSessionePresentazione
LISTA_PRESENTAZIONI_FAVORITE.UsernameUtente -> UTENTE.Username
LISTA_PRESENTAZIONI_FAVORITE.CodicePresentazione -> PRESENTAZIONE.Codice
LISTA_PRESENTAZIONI_FAVORITE.CodiceSessionePresentazione -> PRESENTAZIONE.CodiceSessione
REGISTRAZIONE.UsernameUtente -> UTENTE.Username
REGISTRAZONE.AcronimoConferenza -> CONFERENZA.Acronimo
REGISTRAZIONE.AnnoEdizioneConferenza -> CONFERENZA.AnnoEdizione
PRESENTER.UsernameUtente -> UTENTE.Username
SPEAKER.UsernameUtente -> UTENTE.Username
AMMINISTRATORE.UsernameUtente -> UTENTE.Username
CREAZIONE.UsernameAmministratore -> AMMINISTRATORE.UsernameUtente
CREAZIONE.AcronimoConferenza -> CONFERENZA.Acronimo
CREAZIONE.AnnoEdizioneConferenza -> CONFERENZA.AnnoEdizione
VALUTAZIONE.UsernameAmministratore -> AMMINSTRATORE.UsernameUtente
VALUTAZIONE.CodicePresentazione -> PRESENTAZIONE.Codice
VALUTAZIONE.CodiceSessionePresentazione -> PRESENTAZIONE.CodiceSessione
INFO_AGGIUNTIVE.UsernameSpeaker -> SPEAKER.UsernameUtente
INFO_AGGIUNTIVE.CodiceTutorial -> TUTORIAL.CodicePresentazione
INFO_AGGIUNTIVE.CodiceSessioneTutorial -> TUTORIAL.CodiceSessionePresentazione
PRESENTAZIONE_TUTORIAL.UsernameSpeaker -> SPEAKER.UsernameUtente
PRESENTAZIONE_TUTORIAL.CodiceTutorial -> TUTORIAL.CodicePresentazione
PRESENTAZIONE_TUTORIAL.CodiceSessioneTutorial -> TUTORIAL.CodiceSessionePresentazione

4. *Descrizione ad alto livello delle funzionalità dell'applicazione Web*

La piattaforma CONFVIRTUAL è un'applicazione web che permette di gestire conferenze online. Per la realizzazione della seguente piattaforma è stata usata la libreria *Bootstrap v4.6* (per la parte grafica) e l'estensione per PHP *mongodb (*per tenere traccia dei LOGS con MongoDB*).* La prima pagina che si aprirà all'avvio dell'applicazione web sarà l'homepage, con un titolo di benvenuto, un bottone di LOGIN, una lista con le informazioni riguardanti il numero totale di conferenze registrate, di quelle attive e gli utenti registrati ed infine una tabella rappresentante la classifica dei Presenter/Speaker in base al voto medio. Cliccando su LOGIN verremo reindirizzati alla pagina di Login, dove potremmo accedere inserendo username e password, in caso avessimo già un account, altrimenti registrarci cliccando il bottone REGISTRATI. A seconda del tipo di account con cui abbiamo eseguito l'accesso ci troveremo di fronte a pagine differenti: quella dell'utente generale, quella dello Speaker, quella del Presenter e quella dell'Amministratore.
Nel primo caso, l'utente può scegliere se tornare all'Homepage, visualizzare le conferenze disponibili (il cui click comporterà il caricamento di un'altra pagina in cui visualizzerà una tabella delle conferenze), registrarsi ad una nuova conferenza (anche qui si aprirà una nuova pagina che mostrerà la tabella delle conferenze alla quale è iscritto e la possibilità di iscriversi alle conferenze disponibili), visualizzare le sessioni e le relative presentazioni (altra pagina in cui sarà possibile visualizzare una tabella con le sessioni e relative presentazioni), visualizzare/inserire nella lista delle conferenze preferite (pagina dove sarà visibile la tabella delle proprie conferenze preferite e la possibilità di aggiungerne altre) ed infine entrare nelle chat (pagina in cui verranno visualizzate tutte le chat in un'unica tabella). Nel caso dello Speaker, esso può effettuare le stesse operazioni dell'utente generale ed in più può:

modificare il proprio CV (pagina in cui può modificare il proprio Curriculum Vitae, con un max di 30 caratteri), modificare la propria foto (pagina che permette di sfogliare e caricare un'immagine presente nel proprio PC), modificare affiliazione universitaria (pagina dove è possibile modificare il nome dell'università e del dipartimento di appartenenza), aggiungere un link e una descrizione delle risorse aggiuntive per il proprio tutorial (pagina dove lo Speaker può visualizzare la tabella dei tutorials e sceglierne uno per inserire il link e la descrizione), visualizzare i tutorials a cui è associato (pagina con tabella dei tutorials dello Speaker). L'accesso come Presenter, invece, permette di effettuare le stesse operazioni dell'utente generale e dello Speaker, ad eccezione dell'inserimento/modifica delle risorse aggiuntive per il proprio tutorial. In conclusione, l'Amministratore può eseguire le stesse operazioni dell'utente generale e può inoltre: creare una nuova conferenza (in questa pagina troviamo la tabella con tutte le conferenze e la tabella delle conferenze a cui è registrato e la possibilità di creare una nuova conferenza inserendo tutti i dati necessari), creare una nuova sessione della conferenza (pagina dove è presente la tabella delle sessioni e una select dove è possibile selezionare la conferenza alla quale vogliamo aggiungere una nuova sessione, si aprirà quindi un form dove è necessario inserire tutti i dati per creare una nuova Sessione per la conferenza scelta), inserire delle presentazioni in una sessione (pagina dove vengono mostrate le sessioni in una tabella e la possibilità di aggiungere una nuova presentazione per una determinata sessione scelta) , associare uno Speaker ad un Tutorial (pagina in cui vengono mostrate le sessioni in una tabella e la possibilità di selezionarne una. Successivamente è possibile selezionare un tutorial da associare ,mediante checkbox, ed infine è possibile selezionare lo Speaker) , associare un Presenter ad un Articolo (pagina dove viene mostrata la tabella delle sessioni con possibilità di selezionarne una, successivamente viene mostrata la tabella degli articoli dove è necessario selezionarne uno per l'associazione ed infine viene chiesto di scegliere quale Presenter associare all'articolo precedentemente selezionato) , inserire/visualizzare delle valutazioni sulle presentazioni (pagina dove viene mostrata la tabella delle valutazioni assegnate dall'amministratore con possibilità di scegliere, tramite form, una nuova presentazione a cui associare un voto e delle note)ed infine inserire uno sponsor/sponsorizzazione (pagina con due form di creazione: il primo permette di creare un nuovo Sponsor con nome e immagine logo, mentre l'altro prevede la possibilità di creare una nuova Sponsorizzazione, selezionando lo Sponsor, inserendo l'importo, l'acronimo della conferenza e l'anno).

5. Codice SQL

```
DROP DATABASE IF EXISTS CONFVIRTUAL;
CREATE DATABASE CONFVIRTUAL;
USE CONFVIRTUAL;

SET GLOBAL event_scheduler = ON;

CREATE TABLE UTENTE(
                Username        varchar(30) primary key NOT NULL,
    Password   varchar(30) NOT NULL,
                Nome            varchar(30),
                Cognome         varchar(30),
                LuogoNascita varchar(30),
                DataNascita     Date

) ENGINE = INNODB;

CREATE TABLE PRESENTER(
                UsernameUtente varchar(30) primary key,
                NomeUni          varchar(30),
                NomeDip          varchar(30),
                CV                      varchar(30),
                Foto                MEDIUMBLOB,

    foreign key(UsernameUtente) references UTENTE(Username)
```

```
) ENGINE = INNODB;

CREATE TABLE SPEAKER(
		UsernameUtente varchar(30) primary key,
		NomeUni		varchar(30),
		NomeDip		varchar(30),
		CV			varchar(30),
		Foto		MEDIUMBLOB,

	foreign key(UsernameUtente) references UTENTE(Username)

) ENGINE = INNODB;

CREATE TABLE AMMINISTRATORE(
		UsernameUtente varchar(30) primary key,

	foreign key(UsernameUtente) references UTENTE(Username)

) ENGINE = INNODB;

CREATE TABLE SPONSOR(
		Nome   varchar(30) primary key,
		ImgLogo MEDIUMBLOB

) ENGINE = INNODB;

CREATE TABLE CONFERENZA(
		Acronimo	varchar(30),
		AnnoEdizione YEAR,
		ImgLogo		MEDIUMBLOB,
		Nome		varchar(30) NOT NULL,
		Svolgimento  ENUM("Attiva", "Completata") DEFAULT "Attiva",

	primary key(Acronimo, AnnoEdizione)

) ENGINE = INNODB;

CREATE TABLE SPONSORIZZAZIONE(
		NomeSponsor		varchar(30),
		AcronimoConferenza	varchar(30),
		AnnoEdizioneConferenza YEAR,
		Importo			float NOT NULL,

	primary key(NomeSponsor, AcronimoConferenza, AnnoEdizioneConferenza),

	foreign key(AcronimoConferenza, AnnoEdizioneConferenza) references CONFERENZA(Acronimo,
AnnoEdizione) on delete cascade,
		foreign key(NomeSponsor) references SPONSOR(Nome) on delete cascade

) ENGINE = INNODB;

CREATE TABLE PROGRAMMA_GIORNALIERO(
		Id integer auto_increment,
		AcronimoConferenza varchar(30) references CONFERENZA(Acronimo) on delete cascade,
	AnnoEdizioneConferenza varchar(30) references CONFERENZA(AnnoEdizione) on delete cascade,
		Data			date,

	primary key(Id)
```

```
) ENGINE = INNODB;

CREATE TABLE SESSIONE(
		Codice			varchar(10)   primary key,
		IdProgramma   integer NOT NULL references PROGRAMMA_GIORNALIERO(Id) on delete
cascade,
		LinkTeams		varchar(100),
		NumPresentazioni int DEFAULT 0,
		OraIni			time,
		OraFine			time,
		Titolo			varchar(100)

) ENGINE = INNODB;

CREATE TABLE MESSAGGIO(
		CodiceSessione  varchar(10),
		Timestamp	float,
		UsernameUtente  varchar(30)   NOT NULL,
		Testo		varchar(500),
		DataInserimento date,

	primary key (CodiceSessione, Timestamp),

	foreign key(CodiceSessione) references SESSIONE(Codice) on delete cascade,
	foreign key(UsernameUtente) references UTENTE(Username)

) ENGINE = INNODB;

CREATE TABLE PRESENTAZIONE(
		Codice		varchar(10),
		CodiceSessione varchar(10),
		NumSequenza   int,
	OraIni			time,
		OraFine		time,


	primary key(Codice, CodiceSessione),

	foreign key(CodiceSessione) references SESSIONE(Codice) on delete cascade

) ENGINE = INNODB;

 CREATE TABLE ARTICOLO(
		CodicePresentazione		varchar(10),
		CodiceSessionePresentazione varchar(10),
		Numpagine				int,
		filePDF					MEDIUMBLOB,
		Titolo					varchar(100),
		StatoSvolgimento		ENUM("Coperto", "NonCoperto") DEFAULT "NonCoperto",
		UsernamePresenter		varchar(30),

		primary key(CodicePresentazione, CodiceSessionePresentazione),

	foreign key(CodicePresentazione, CodiceSessionePresentazione) references
PRESENTAZIONE(Codice, CodiceSessione) on delete cascade,
	foreign key(UsernamePresenter) references PRESENTER(UsernameUtente)

) ENGINE = INNODB;
```

```
CREATE TABLE TUTORIAL(
		CodicePresentazione		varchar(10),
		CodiceSessionePresentazione	varchar(10),
		Titolo				varchar(100),
		Abstract			varchar(500),

	primary key(CodicePresentazione, CodiceSessionePresentazione),

	foreign key(CodicePresentazione, CodiceSessionePresentazione) references
PRESENTAZIONE(Codice, CodiceSessione) on delete cascade
) ENGINE = INNODB;

CREATE TABLE AUTORE(
		ID		varchar(30)	primary key,
		Nome	varchar(30) NOT NULL,
		Cognome	varchar(30) NOT NULL
) ENGINE = INNODB;

CREATE TABLE LISTA_AUTORI(
		IdAutore			varchar(30),
		CodiceArticolo			varchar(10),
		CodiceSessioneArticolo	varchar(10),

		primary key(IdAutore, CodiceArticolo, CodiceSessioneArticolo),

		foreign key(CodiceArticolo, CodiceSessioneArticolo) references
ARTICOLO(CodicePresentazione, CodiceSessionePresentazione) on delete cascade,
		foreign key(IdAutore) references AUTORE(ID) on delete cascade

) ENGINE = INNODB;

CREATE TABLE PAROLA_CHIAVE(
		CodiceArticolo	varchar(10) references ARTICOLO(Codice) on delete cascade,
	CodiceSessioneArticolo	varchar(10)  references ARTICOLO(Codice) on delete cascade,
		Parola		varchar(20),
	primary key(CodiceArticolo, CodiceSessioneArticolo, Parola)

) ENGINE = INNODB;

CREATE TABLE LISTA_PRESENTAZIONI_FAVORITE(
		UsernameUtente			varchar(30),
		CodicePresentazione		varchar(10),
		CodiceSessionePresentazione	varchar(10),

	primary key(UsernameUtente, CodicePresentazione, CodiceSessionePresentazione),

	foreign key(UsernameUtente) references UTENTE(Username),
	foreign key(CodicePresentazione, CodiceSessionePresentazione) references
PRESENTAZIONE(Codice, CodiceSessione) on delete cascade

) ENGINE = INNODB;

CREATE TABLE REGISTRAZIONE(
		UsernameUtente			varchar(30),
		AcronimoConferenza	varchar(30),
		AnnoEdizioneConferenza YEAR,

	primary key(UsernameUtente, AcronimoConferenza, AnnoEdizioneConferenza),
```

```
    foreign key(UsernameUtente) references UTENTE(Username),
    foreign key(AcronimoConferenza, AnnoEdizioneConferenza) references CONFERENZA(Acronimo,
AnnoEdizione) on delete cascade

) ENGINE = INNODB;

 CREATE TABLE CREAZIONE_CONFERENZA(
                UsernameAmministratore varchar(30),
                AcronimoConferenza     varchar(30),
                AnnoEdizioneConferenza YEAR,

     primary key(UsernameAmministratore, AcronimoConferenza, AnnoEdizioneConferenza),

                foreign key(UsernameAmministratore) references AMMINISTRATORE(UsernameUtente),
     foreign key(AcronimoConferenza, AnnoEdizioneConferenza) references CONFERENZA(Acronimo,
AnnoEdizione) on delete cascade

 ) ENGINE = INNODB;

CREATE TABLE VALUTAZIONE(
                UsernameAmministratore          varchar(30),
                CodicePresentazione             varchar(10),
                CodiceSessionePresentazione varchar(10),
                Voto                                    int             CHECK(Voto >= 0 and Voto
<= 10),
                Note                                    varchar(50),

    primary key(UsernameAmministratore, CodicePresentazione, CodiceSessionePresentazione),

    foreign key(CodicePresentazione, CodiceSessionePresentazione) references
PRESENTAZIONE(Codice, CodiceSessione) on delete cascade,
                foreign key(UsernameAmministratore) references AMMINISTRATORE(UsernameUtente)

) ENGINE = INNODB;

 CREATE TABLE INFO_AGGIUNTIVE(
                UsernameSpeaker         varchar(30),
                CodiceTutorial                  varchar(10),
                CodiceSessioneTutorial varchar(10),
                LinkWeb                         varchar(100),
                Descrizione             varchar(500),

                primary key(UsernameSpeaker, CodiceTutorial, CodiceSessioneTutorial),

    foreign key(UsernameSpeaker) references SPEAKER(UsernameUtente),
                foreign key(CodiceTutorial, CodiceSessioneTutorial) references
TUTORIAL(CodicePresentazione, CodiceSessionePresentazione) on delete cascade

) ENGINE = INNODB;

CREATE TABLE PRESENTAZIONE_TUTORIAL(
                UsernameSpeaker         varchar(30),
                CodiceTutorial                  varchar(10),
                CodiceSessioneTutorial  varchar(10),

    primary key(UsernameSpeaker, CodiceTutorial, CodiceSessioneTutorial),

                foreign key(UsernameSpeaker) references SPEAKER(UsernameUtente),
```

```
      foreign key(CodiceTutorial, CodiceSessioneTutorial) references TUTORIAL(CodicePresentazione,
CodiceSessionePresentazione) on delete cascade

) ENGINE = INNODB;

INSERT INTO UTENTE (Username, Password, Nome, Cognome, LuogoNascita, DataNascita)
values ("Aut", "123", "Dario", "Bianchi", "Bologna", "2000-10-10");

INSERT INTO UTENTE (Username, Password, Nome, Cognome, LuogoNascita, DataNascita)
values ("Sp1", "123", "Alice", "Bruna", "Bologna", "2000-10-10");

INSERT INTO UTENTE (Username, Password, Nome, Cognome, LuogoNascita, DataNascita)
values ("Pres1", "123", "Alessanda", "Pasticcio", "Bologna", "2000-10-10");

INSERT INTO UTENTE (Username, Password, Nome, Cognome, LuogoNascita, DataNascita)
values ("CarloAm", "123", "Carlo", "Rossi", "Bologna", "2000-10-10");

INSERT INTO PRESENTER (UsernameUtente, NomeUni, NomeDip, CV, Foto)
values ("Pres1", "Unibo", "InfoMan", "CV1", "Foto1");

INSERT INTO AMMINISTRATORE (UsernameUtente)
values ("CarloAm");

INSERT INTO SPEAKER (UsernameUtente, NomeUni, NomeDip, CV, Foto)
values ("Sp1", "Unibo", "Informatica", "CV1", "foto1");

#Lista stored procedure
/**********************************************************************************************
**********************/
#Stored procedure --> crea Conferenza (Controllo che l'annoEdizione sia maggiore della data corrente)
start transaction;
delimiter |
CREATE PROCEDURE CreaConferenza(Acronimo varchar(30),AnnoEdizione YEAR,ImgLogo
MEDIUMBLOB,Nome varchar(30))
        BEGIN
   IF(AnnoEdizione >= YEAR(NOW()) AND AnnoEdizione < 2155 ) THEN
                        INSERT INTO CONFERENZA SET Acronimo = Acronimo, AnnoEdizione =
AnnoEdizione, ImgLogo = ImgLogo, Nome = Nome;
        END IF;
        END;
| delimiter ;
commit;

#Stored procedure
start transaction;
delimiter |
CREATE PROCEDURE AssociaAmministratore(UsernameAmministratore varchar(30),
AcronimoConferenza varchar(30), AnnoEdizioneConferenza YEAR)
        BEGIN
                        INSERT INTO registrazione SET UsernameUtente = UsernameAmministratore,
AcronimoConferenza = AcronimoConferenza, AnnoEdizioneConferenza = AnnoEdizioneConferenza;
        END;
| delimiter ;
commit;

#Stored procedure --> associa admin a conferenza creata
start transaction;
delimiter |
CREATE PROCEDURE AssociaCreazioneConf(UsernameAmministratore varchar(30),
```

```
AcronimoConferenza varchar(30), AnnoEdizioneConferenza YEAR)
	BEGIN
			INSERT INTO CREAZIONE_CONFERENZA SET UsernameAmministratore =
UsernameAmministratore, AcronimoConferenza = AcronimoConferenza, AnnoEdizioneConferenza =
AnnoEdizioneConferenza;
	END;
| delimiter ;
commit;

#Stored procedure --> crea programma_giornaliero
start transaction;
 delimiter |
 CREATE PROCEDURE CreaProgrammaGiornaliero(AcronimoConferenza varchar(30),
AnnoEdizioneConferenza varchar(30), Data date)
	BEGIN
   INSERT INTO PROGRAMMA_GIORNALIERO SET AcronimoConferenza = AcronimoConferenza,
AnnoEdizioneConferenza = AnnoEdizioneConferenza, Data = Data;
   COMMIT;
   END
| delimiter ;

#Stored procedure --> crea Sessione
start transaction;
delimiter |
CREATE PROCEDURE CreaSessione(IN Codice varchar(10), IN IdProgramma integer,  IN LinkTeams
varchar(100),IN NumPresentazioni int(11), IN OraIni time, IN OraFine time, IN Titolo varchar(100))
	begin
		IF(OraIni < OraFine &&(SELECT count(PROGRAMMA_GIORNALIERO.Id)
                                                                FROM PROGRAMMA_GIORNALIERO
                                                               WHERE PROGRAMMA_GIORNALIERO.Id =
IdProgramma) > 0)
		THEN
			INSERT INTO SESSIONE
			SET Codice = Codice, IdProgramma = IdProgramma, LinkTeams =
LinkTeams,NumPresentazioni=NumPresentazioni, OraIni = OraIni, OraFine = OraFine, Titolo = Titolo;
			COMMIT;
		ELSE ROLLBACK;
		END IF;
	END;
| delimiter ;

#Stored procedure --> crea Presentazione
start transaction;
delimiter |
CREATE PROCEDURE CreaPresentazione(Codice varchar(10), CodiceSessione varchar(10),
NumSequenza int, NewOraIni time, NewOraFine time )
	begin
		IF((SELECT count(SESSIONE.Codice)
			 FROM SESSIONE
			WHERE (SESSIONE.Codice = CodiceSessione)
			 AND (NewOraFine <= SESSIONE.OraFine)
			 AND (NewOraIni >= SESSIONE.OraIni)
	 AND NewOraIni < NewOraFine)) > 0
		THEN
			INSERT INTO PRESENTAZIONE
	SET Codice = Codice, CodiceSessione = CodiceSessione, NumSequenza = NumSequenza, OraFine =
NewOraFine, OraIni = NewOraIni;
			COMMIT;
		ELSE ROLLBACK;
```

```
                END IF;
        END;
| delimiter ;

 # Stored procedure --> associa speaker - tutorial
 start transaction;
delimiter |
CREATE PROCEDURE AssociaSpeaker(UsernameSpeaker varchar(30), CodiceTutorial varchar(10),
CodiceSessioneTutorial varchar(10))
        BEGIN
                if(SELECT count(SPEAKER.UsernameUtente) FROM SPEAKER WHERE
SPEAKER.UsernameUtente = UsernameSpeaker) > 0 THEN
      INSERT INTO PRESENTAZIONE_TUTORIAL
      SET UsernameSpeaker = UsernameSpeaker, CodiceTutorial = CodiceTutorial, CodiceSessioneTutorial
= CodiceSessioneTutorial;
      COMMIT;
      end if;
   END
 | delimiter ;

/**********************************************************************************************************
**********************/
 #Stored procedure --> Associa un presenter alla presentazione di un articolo
 start transaction;
 delimiter |
 CREATE PROCEDURE AssociaPresenter(CodicePresentazione varchar(10),CodiceSessionePresentazione
varchar(10),UsernamePresenter varchar(30))
 BEGIN
 if((SELECT count(ARTICOLO.CodicePresentazione) FROM ARTICOLO WHERE
 ((ARTICOLO.CodicePresentazione=CodicePresentazione) and
(ARTICOLO.CodiceSessionePresentazione=CodiceSessionePresentazione)))>0 AND
 (SELECT count(PRESENTER.UsernameUtente) FROM PRESENTER WHERE
(PRESENTER.UsernameUtente=UsernamePresenter))>0 ) THEN
 UPDATE ARTICOLO
 SET
 UsernamePresenter=UsernamePresenter
 WHERE
 CodicePresentazione=CodicePresentazione AND
CodiceSessionePresentazione=CodiceSessionePresentazione;
 COMMIT;
 end if;
 END
 |delimiter;

 # Stored procedure --> crea Utente, utile per la registrazione di un nuovo utente
 start transaction;
 delimiter |
 CREATE PROCEDURE CreaUtente(Username varchar(30), Password varchar(30), Nome varchar(30),
Cognome varchar(30), LuogoNascita varchar(30), DataNascita Date)
        BEGIN
   INSERT INTO UTENTE SET  Username = Username, Password = Password, Nome = Nome, Cognome =
Cognome, LuogoNascita = LuogoNascita, DataNascita = DataNascita;
   COMMIT;
   END
| delimiter ;

# Stored procedure --> crea Speaker
 start transaction;
 delimiter |
```

```
 CREATE PROCEDURE CreaSpeaker(UsernameUtente varchar(30), NomeUni varchar(30), NomeDip
varchar(30), CV varchar(30), Foto MEDIUMBLOB)
        BEGIN
   if(SELECT count(UTENTE.Username) FROM UTENTE WHERE UTENTE.Username = UsernameUtente)
> 0 THEN
   INSERT INTO SPEAKER SET  UsernameUtente = UsernameUtente, NomeUni = NomeUni, NomeDip =
NomeDip, CV = CV, Foto = Foto;
   COMMIT;
   end if;
   END
| delimiter ;

# Stored procedure --> crea Presenter
 start transaction;
 delimiter |
 CREATE PROCEDURE CreaPresenter(UsernameUtente varchar(30), NomeUni varchar(30), NomeDip
varchar(30), CV varchar(30), Foto MEDIUMBLOB)
        BEGIN
   if(SELECT count(UTENTE.Username) FROM UTENTE WHERE UTENTE.Username = UsernameUtente)
> 0 THEN
   INSERT INTO PRESENTER SET  UsernameUtente = UsernameUtente, NomeUni = NomeUni, NomeDip =
NomeDip, CV = CV, Foto = Foto;
   COMMIT;
   end if;
   END
| delimiter ;

# Stored procedure --> modifica CV da parte dello speaker
 start transaction;
 delimiter |
 CREATE PROCEDURE ModificaCVSpeaker(UsernameUtente varchar(30), CV varchar(30))
        BEGIN
   UPDATE SPEAKER
   SET CV = CV
   WHERE (SPEAKER.UsernameUtente = UsernameUtente);
   COMMIT;
   END
| delimiter ;

# Stored procedure --> modifica Foto da parte dello speaker
 start transaction;
 delimiter |
 CREATE PROCEDURE ModificaFotoSpeaker(UsernameUtente varchar(30), Foto MEDIUMBLOB)
        BEGIN
   UPDATE SPEAKER
   SET Foto = Foto
   WHERE (SPEAKER.UsernameUtente = UsernameUtente);
   COMMIT;
   END
| delimiter ;

# Stored procedure --> modifica affiliazione universitaria da parte dello speaker
 start transaction;
 delimiter |
 CREATE PROCEDURE ModificaAffiliazioneSpeaker(UsernameUtente varchar(30), NomeUni varchar(30),
NomeDip varchar(30))
        BEGIN
   UPDATE SPEAKER
   SET NomeUni = NomeUni, NomeDip = NomeDip
```

```
   WHERE (SPEAKER.UsernameUtente = UsernameUtente);
   COMMIT;
   END
| delimiter ;

# Stored procedure --> modifica CV da parte del presenter
 start transaction;
 delimiter |
 CREATE PROCEDURE ModificaCVPresenter(UsernameUtente varchar(30), CV varchar(30))
          BEGIN
   UPDATE PRESENTER
   SET CV = CV
   WHERE (PRESENTER.UsernameUtente = UsernameUtente);
   COMMIT;
   END
| delimiter ;

# Stored procedure --> modifica Foto da parte del presenter
 start transaction;
 delimiter |
 CREATE PROCEDURE ModificaFotoPresenter(UsernameUtente varchar(30), Foto MEDIUMBLOB)
          BEGIN
   UPDATE PRESENTER
   SET Foto = Foto
   WHERE (PRESENTER.UsernameUtente = UsernameUtente);
   COMMIT;
   END
| delimiter ;

# Stored procedure --> modifica affiliazione universitaria da parte del presenter
 start transaction;
 delimiter |
 CREATE PROCEDURE ModificaAffiliazionePresenter(UsernameUtente varchar(30), NomeUni
varchar(30), NomeDip varchar(30))
          BEGIN
   UPDATE PRESENTER
   SET NomeUni = NomeUni, NomeDip = NomeDip
   WHERE (PRESENTER.UsernameUtente = UsernameUtente);
   COMMIT;
   END
| delimiter ;

# Stored procedure --> creazione di un tutorial
 start transaction;
 delimiter |
 CREATE PROCEDURE CreaTutorial(CodicePresentazione varchar(10), CodiceSessionePresentazione
varchar(10), Titolo varchar(100), Abstract varchar(500))
          BEGIN
                INSERT INTO TUTORIAL
                SET CodicePresentazione = CodicePresentazione, CodiceSessionePresentazione =
CodiceSessionePresentazione, Titolo = Titolo, Abstract = Abstract;
                COMMIT;
   END
| delimiter ;

# Stored procedure --> crea info aggiuntive
 start transaction;
 delimiter |
 CREATE PROCEDURE CreaInfoAggiuntive(UsernameSpeaker varchar(30), CodiceTutorial varchar(10),
```

```
CodiceSessioneTutorial varchar(10), LinkWeb varchar(100), Descrizione varchar(500))
        BEGIN
                INSERT INTO INFO_AGGIUNTIVE
                SET UsernameSpeaker = UsernameSpeaker, CodiceTutorial = CodiceTutorial,
CodiceSessioneTutorial = CodiceSessioneTutorial, LinkWeb = LinkWeb, Descrizione = Descrizione;
                COMMIT;
   END
| delimiter ;

# Stored procedure --> inserisci o modifica il link in info_aggiuntive
 start transaction;
 delimiter |
 CREATE PROCEDURE ModificaLinkInfoAggiuntive(UsernameSpeaker varchar(30), CodiceTutorial
varchar(10), CodiceSessioneTutorial varchar(10), LinkWeb varchar(100))
        BEGIN
   UPDATE INFO_AGGIUNTIVE
   SET LinkWeb = LinkWeb
   WHERE (INFO_AGGIUNTIVE.UsernameSpeaker = UsernameSpeaker) AND
(INFO_AGGIUNTIVE.CodiceTutorial = CodiceTutorial) AND (INFO_AGGIUNTIVE.CodiceSessioneTutorial =
CodiceSessioneTutorial);
   COMMIT;
   END
| delimiter ;

# Stored procedure --> inserisci o modifica la descrizione in info_aggiuntive
 start transaction;
 delimiter |
 CREATE PROCEDURE ModificaDescrizioneInfoAggiuntive(UsernameSpeaker varchar(30),
CodiceTutorial varchar(10), CodiceSessioneTutorial varchar(10), Descrizione varchar(500))
        BEGIN
   UPDATE INFO_AGGIUNTIVE
   SET Descrizione = Descrizione
   WHERE (INFO_AGGIUNTIVE.UsernameSpeaker = UsernameSpeaker) AND
(INFO_AGGIUNTIVE.CodiceTutorial = CodiceTutorial) AND (INFO_AGGIUNTIVE.CodiceSessioneTutorial =
CodiceSessioneTutorial);
   COMMIT;
   END
| delimiter ;

# Stored procedure --> registrazione a una conferenza
 start transaction;
 delimiter |
 CREATE PROCEDURE RegistrazioneConferenza(UsernameUtente varchar(30), AcronimoConferenza
varchar(30), AnnoEdizioneConferenza YEAR)
        BEGIN
   INSERT INTO REGISTRAZIONE
   SET UsernameUtente = UsernameUtente, AcronimoConferenza = AcronimoConferenza,
AnnoEdizioneConferenza = AnnoEdizioneConferenza;
   COMMIT;
   END
| delimiter ;
#Store Procedure --> Crea Articolo
start transaction;
delimiter |
CREATE PROCEDURE CreaArticolo(CodicePresentazione varchar(10), CodiceSessionePresentazione
varchar(10), Numpagine int(11),
                                filePDF MEDIUMBLOB, Titolo varchar(100))
 BEGIN
        if(SELECT count(CodicePresentazione)
```

```
        FROM PRESENTAZIONE
                WHERE ((CodicePresentazione = CodicePresentazione)
            and (CodiceSessionePresentazione = CodiceSessionePresentazione)) > 0)
        then
                        INSERT INTO ARTICOLO
                        SET CodicePresentazione = CodicePresentazione,
                        CodiceSessionePresentazione = CodiceSessionePresentazione,
                        Numpagine = Numpagine,
                        filePDF = filePDF,
                        Titolo = Titolo,
                        StatoSvolgimento = "NonCoperto";
                        COMMIT;
        end if;
 END
| delimiter;

# Stored procedure --> inserimento lista presentazioni favorite
 start transaction;
 delimiter |
 CREATE PROCEDURE InserisciPresentazionePreferitaInLista(UsernameUtente varchar(30),
CodicePresentazione varchar(10), CodiceSessionePresentazione varchar(10))
        BEGIN
   INSERT INTO LISTA_PRESENTAZIONI_FAVORITE
   SET UsernameUtente = UsernameUtente, CodicePresentazione = CodicePresentazione,
CodiceSessionePresentazione = CodiceSessionePresentazione;
   COMMIT;
   END
| delimiter ;

# Stored procedure --> inserimento sponsor
 start transaction;
 delimiter |
 CREATE PROCEDURE InserisciSponsor(Nome varchar(30), ImgLogo MEDIUMBLOB)
        BEGIN
   INSERT INTO SPONSOR
   SET Nome = Nome, ImgLogo = ImgLogo;
   COMMIT;
   END
| delimiter ;

# Stored procedure --> inserimento sponsorizzazione
 start transaction;
 delimiter |
 CREATE PROCEDURE InserisciSponsorizzazione(NomeSponsor varchar(30), AcronimoConferenza
varchar(30), AnnoEdizioneConferenza YEAR, Importo float)
        BEGIN
   INSERT INTO SPONSORIZZAZIONE
   SET NomeSponsor = NomeSponsor, AcronimoConferenza = AcronimoConferenza,
AnnoEdizioneConferenza = AnnoEdizioneConferenza, Importo = Importo;
   COMMIT;
   END
| delimiter ;

# Stored procedure --> inserisci autore
 start transaction;
 delimiter |
 CREATE PROCEDURE InserisciAutore(ID varchar(30), Nome varchar(30), Cognome varchar(30),
CodiceArticolo varchar(10), CodiceSessioneArticolo varchar(10))
        BEGIN
```

```
                INSERT INTO AUTORE
                SET ID = ID, Nome = Nome, Cognome = Cognome;
                INSERT INTO LISTA_AUTORI
                SET IdAutore = ID, CodiceArticolo = CodiceArticolo, CodiceSessioneArticolo =
CodiceSessioneArticolo;
                COMMIT;
   END
| delimiter ;


# Stored procedure --> inserisci autore nella lista
 start transaction;
 delimiter |
 CREATE PROCEDURE InserisciListaAutori(IdAutore varchar(30), CodiceArticolo varchar(10),
CodiceSessioneArticolo varchar(10))
        BEGIN
   INSERT INTO LISTA_AUTORI
   SET IdAutore = IdAutore, CodiceArticolo = CodiceArticolo, CodiceSessioneArticolo =
CodiceSessioneArticolo;
   COMMIT;
   END
| delimiter ;

# Stored procedure --> inserimento admin
 start transaction;
 delimiter |
 CREATE PROCEDURE InserisciAmministratore(UsernameUtente varchar(30))
        BEGIN
   INSERT INTO AMMINISTRATORE
   SET UsernameUtente = UsernameUtente;
   COMMIT;
   END
| delimiter ;

# Stored procedure --> inserimento messaggio
 start transaction;
 delimiter |
 CREATE PROCEDURE InserisciMessaggio(CodiceSessione varchar(10),Timestamp float,
UsernameUtente varchar(30), Testo varchar(500), DataInserimento date)
        BEGIN
   INSERT INTO MESSAGGIO
   SET CodiceSessione = CodiceSessione, Timestamp = Timestamp, UsernameUtente = UsernameUtente,
Testo = Testo, DataInserimento = DataInserimento;
   COMMIT;
   END
| delimiter ;

# Stored procedure --> inserimento valutazione
 start transaction;
 delimiter |
 CREATE PROCEDURE InserisciValutazione(UsernameAmministratore varchar(30), CodicePresentazione
varchar(10), CodiceSessionePresentazione varchar(10), Voto int, Note varchar(50))
        BEGIN
   INSERT INTO VALUTAZIONE
   SET UsernameAmministratore = UsernameAmministratore, CodicePresentazione =
CodicePresentazione, CodiceSessionePresentazione = CodiceSessionePresentazione, Voto = Voto, Note =
Note;
   COMMIT;
   END
```

```
| delimiter ;

# Stored procedure --> inserimento parola chiave
 start transaction;
 delimiter |
 CREATE PROCEDURE InserisciParolaChiave(CodiceArticolo varchar(10), CodiceSessioneArticolo
varchar(10), Parola varchar(20))
        BEGIN
   INSERT INTO PAROLA_CHIAVE
   SET CodiceArticolo = CodiceArticolo, CodiceSessioneArticolo = CodiceSessioneArticolo, Parola =
Parola;
   COMMIT;
   END
| delimiter ;

# Stored procedure --> elimina conferenza
 start transaction;
 delimiter |
 CREATE PROCEDURE EliminaConferenza(Acronimo varchar(30), AnnoEdizione YEAR)
        BEGIN
   DELETE FROM CONFERENZA
        WHERE (CONFERENZA.Acronimo = Acronimo) AND (CONFERENZA.AnnoEdizione =
AnnoEdizione);
   COMMIT;
   END
| delimiter ;

# Stored procedure --> elimina sessione
 start transaction;
 delimiter |
 CREATE PROCEDURE EliminaSessione(Codice varchar(10))
        BEGIN
   DELETE FROM SESSIONE
        WHERE (SESSIONE.Codice = Codice);
   COMMIT;
   END
| delimiter ;

# Stored procedure --> elimina presentazione
 start transaction;
 delimiter |
 CREATE PROCEDURE EliminaPresentazione(Codice varchar(10), CodiceSessione varchar(10))
        BEGIN
   DELETE FROM PRESENTAZIONE
        WHERE (PRESENTAZIONE.Codice = Codice) AND (PRESENTAZIONE.CodiceSessione =
CodiceSessione);
   COMMIT;
   END
| delimiter ;

# Stored procedure --> elimina info aggiuntive
 start transaction;
 delimiter |
 CREATE PROCEDURE EliminaInfoAggiuntive(UsernameSpeaker varchar(30), CodiceTutorial
varchar(10), CodiceSessioneTutorial varchar(10))
        BEGIN
   DELETE FROM INFO_AGGIUNTIVE
        WHERE (INFO_AGGIUNTIVE.UsernameSpeaker = UsernameSpeaker) AND
(INFO_AGGIUNTIVE.CodiceTutorial = CodiceTutorial) AND (INFO_AGGIUNTIVE.CodiceSessioneTutorial =
```

```
CodiceSessioneTutorial);
   COMMIT;
   END
| delimiter ;

# Stored procedure
 start transaction;
 delimiter |
 CREATE PROCEDURE CreaNuovoArticolo(Codice varchar(10), CodiceSessione varchar(10),
NumSequenza varchar(10),
                                        OraIni time, OraFine time, Numpagine int(11) ,filePDF
MEDIUMBLOB, Titolo varchar(100))
          BEGIN
                    SET @Codice = Codice, @CodiceSessione = CodiceSessione, @NumSequenza =
NumSequenza, @OraIni = OraIni, @OraFine = OraFine,
                              @Numpagine = Numpagine, @filePDF = filePDF, @Titolo = Titolo;

                    CALL CreaPresentazione(@Codice, @CodiceSessione, @NumSequenza, @OraIni,
@OraFine);
      CALL CreaArticolo(@Codice, @CodiceSessione, @Numpagine, @filePDF, @Titolo);

   COMMIT;
   END
| delimiter ;

# Stored procedure
 start transaction;
 delimiter |
 CREATE PROCEDURE CreaNuovoTutorial(Codice varchar(10), CodiceSessione varchar(10),
NumSequenza varchar(10),
                                        OraIni time, OraFine time, Titolo varchar(100), Abstract
varchar(500))
          BEGIN
                    SET @Codice = Codice, @CodiceSessione = CodiceSessione, @NumSequenza =
NumSequenza, @OraIni = OraIni,
                              @OraFine = OraFine, @Titolo = Titolo, @Abstract = Abstract;

                    CALL CreaPresentazione(@Codice, @CodiceSessione, @NumSequenza, @OraIni,
@OraFine);
      CALL CreaTutorial(@Codice, @CodiceSessione, @Titolo, @Abstract);

   COMMIT;
   END
| delimiter ;

# Stored procedure
 start transaction;
 delimiter |
CREATE PROCEDURE CreaEAssociaPresenter(UsernamePresenter varchar(30), CodiceArticolo
varchar(10),
                                   CodiceSessioneArticolo varchar(10))
          BEGIN
                    SET @UsernamePresenter = UsernamePresenter, @Codice = CodiceArticolo,
@CodiceSessione = CodiceSessioneArticolo;

                    IF(SELECT count(UsernameUtente)
                              FROM PRESENTER
                              WHERE (UsernameUtente = @UsernamePresenter) = 0)
                    THEN
```

```
                        CALL CreaUtente(@UsernamePresenter, "password", null, null, null, null);
                        CALL CreaPresenter(@UsernamePresenter, null, null, null, null);
        COMMIT;
                ELSE ROLLBACK;
                END IF;

    CALL AssociaPresenter(@Codice, @CodiceSessione, @UsernamePresenter);

  COMMIT;
  END
| delimiter ;

/*************************************************************************************************************
************************/



 #Lista dei trigger
/*************************************************************************************************************
************************/
#Trigger --> Aggiorna il numero di presentazioni dentro la tabella SESSIONE
delimiter |
CREATE TRIGGER AggiornaNumeroPresentazioni
                AFTER INSERT ON PRESENTAZIONE
 FOR EACH ROW
                BEGIN
                        UPDATE SESSIONE
                          SET SESSIONE.NumPresentazioni =  SESSIONE.NumPresentazioni + 1
                         WHERE SESSIONE.Codice = NEW.CodiceSessione;
                 END;
| delimiter ;

#DROP TRIGGER IF EXISTS CambiaStatoSvolgimento;
# trigger : setta stato svolgimento a "Coperto" quando viene associato un Presenter ad un Articolo
delimiter |
CREATE TRIGGER CambiaStatoSvolgimento BEFORE UPDATE ON ARTICOLO
 FOR EACH ROW
 BEGIN
                        SET NEW.StatoSvolgimento = "Coperto";
 END
| delimiter ;

#Lista delle view
/*************************************************************************************************************
************************/

#View | Visualizza i presenter/speaker sulla base del voto medio
delimiter |
CREATE VIEW presenter_speaker_votomedio(Username,VotoMed,Tipo) AS
          SELECT Utente.Username, AVG(VALUTAZIONE.Voto) AS VotoMed,
   CASE WHEN Utente.Username=Speaker.UsernameUtente THEN "Speaker"
   ELSE "Presenter" END as TipoUtente
   FROM Utente,Speaker,Presenter,Valutazione,presentazione_tutorial,Articolo
   WHERE ((Utente.Username=Speaker.UsernameUtente) AND (SPEAKER.UsernameUtente =
PRESENTAZIONE_TUTORIAL.UsernameSpeaker) AND
                (PRESENTAZIONE_TUTORIAL.CodiceTutorial = VALUTAZIONE.CodicePresentazione)
AND
       (PRESENTAZIONE_TUTORIAL.CodiceSessioneTutorial =
VALUTAZIONE.CodiceSessionePresentazione)) # Caso in cui è uno Speaker
```

```
	OR
	((Utente.Username=Presenter.UsernameUtente) AND (PRESENTER.UsernameUtente =
ARTICOLO.UsernamePresenter) AND
			(ARTICOLO.CodicePresentazione = VALUTAZIONE.CodicePresentazione) AND
	(ARTICOLO.CodiceSessionePresentazione = VALUTAZIONE.CodiceSessionePresentazione)) # Caso
in cui è uno Presenter
	group by (Username)
			ORDER BY VotoMed DESC;

| delimiter ;


#View  | Mostra Username e tipo Utente (Escluso utente generale)
delimiter |
CREATE VIEW username_tipoutente(Username,Tipo) AS
	SELECT Username,
  CASE WHEN (Utente.Username=Amministratore.UsernameUtente) THEN 'Amministratore'
			WHEN (Utente.Username=Presenter.UsernameUtente) THEN 'Presenter'
    ELSE 'Speaker' END AS TipoUtente
  FROM Utente,Amministratore,Speaker,Presenter
  WHERE (Utente.Username=Amministratore.UsernameUtente) OR
(Utente.Username=Presenter.UsernameUtente)
			OR (Utente.Username=Speaker.UsernameUtente)
	group by (Username);
| delimiter ;

/*****************************************************************************************************
************************/


# evento 1: setta svolgimento della conferenza a "Completata" dopo la scadenza
CREATE VIEW dataMax(AnnoEdizioneConferenza,AcronimoConferenza,DataMassima) AS (
	SELECT AnnoEdizioneConferenza, AcronimoConferenza, MAX(Data) AS DataMassima
	FROM PROGRAMMA_GIORNALIERO, CONFERENZA
	WHERE (PROGRAMMA_GIORNALIERO.AcronimoConferenza = CONFERENZA.Acronimo)
	AND (PROGRAMMA_GIORNALIERO.AnnoEdizioneConferenza = CONFERENZA.AnnoEdizione)
	GROUP BY PROGRAMMA_GIORNALIERO.AcronimoConferenza,
PROGRAMMA_GIORNALIERO.AnnoEdizioneConferenza
);

delimiter |
CREATE EVENT ModificaSvolgimento
ON SCHEDULE EVERY 24 HOUR
DO
	UPDATE CONFERENZA, dataMax
	  SET CONFERENZA.Svolgimento = "Completata"
	 WHERE(CONFERENZA.Acronimo = dataMax.AcronimoConferenza)
		AND (CONFERENZA.AnnoEdizione = dataMax.AnnoEdizioneConferenza)
		AND (now() > dataMax.DataMassima);
| delimiter ;
```