Progetto di Basi di Dati
“Studio di Modellazione 3D”

Spazzali Filippo IN0501237

Indice

1 Presentazione Progetto

Uno studio professionale di modellazione 3D gestisce vari progetti per diversi clienti, tenendo traccia di tutti i lavori svolti, i modelli creati e le risorse utilizzate. Lo studio impiega diversi artisti specializzati in varie aree della modellazione 3D.

Lo studio desidera mantenere informazioni dettagliate su ogni cliente, che includono ragione sociale, partita IVA, indirizzo (città, via, civico), email, telefono e, facoltativamente, sito web. I clienti possono essere sia aziende che privati.

Per quanto riguarda i dipendenti dello studio, si tiene traccia del nome, cognome, codice fiscale, data e luogo di nascita, email, telefono e specializzazione (modellazione, texturing, rigging, animazione, illuminazione, rendering).

Ogni progetto commissionato allo studio ha un codice identificativo, un titolo, una data di inizio, una data di consegna prevista, un budget concordato, e appartiene a una categoria (architettura, videogiochi, cinema, pubblicità, ingegneria, formazione). Ogni progetto è assegnato a un cliente e può coinvolgere più dipendenti con ruoli diversi.

I progetti si compongono di modelli 3D, ciascuno con un proprio codice identificativo, nome, data di creazione, tipo (organico, inorganico, ambiente, personaggio), poligoni totali e dimensione in megabyte. Ogni modello può utilizzare diverse risorse esterne come texture, che hanno un proprio identificativo, nome, risoluzione, dimensione in megabyte e formato (JPG, PNG, TIFF, EXR).

Lo studio utilizza vari software di modellazione 3D, di cui si tiene traccia del nome, versione, sviluppatore, costo della licenza e data di scadenza della licenza. Per ogni progetto e modello, si registra quali software sono stati utilizzati.

Inoltre, lo studio offre servizi di formazione attraverso corsi di modellazione 3D solo ed esclusivamente ai propri clienti (ovvero persone che hanno già richiesto un progetto). Ogni corso ha un codice, titolo, descrizione, durata in ore, livello (base, intermedio, avanzato) e numero massimo di partecipanti. I corsi sono tenuti da dipendenti dello studio e possono essere frequentati solo da clienti già esistenti, di cui si mantengono nome, cognome, email, telefono e data di iscrizione. SI specifica che i corsi sono gratuiti.

Azioni

- L'amministrazione vuole ottenere la lista dei progetti completati nell'ultimo mese per la fatturazione

  (2 Volte al Mese)

- Il responsabile tecnico vuole ottenere le statistiche sull'utilizzo dei software per pianificare il rinnovo delle licenze

  (1 Volta ogni 3 Mesi)

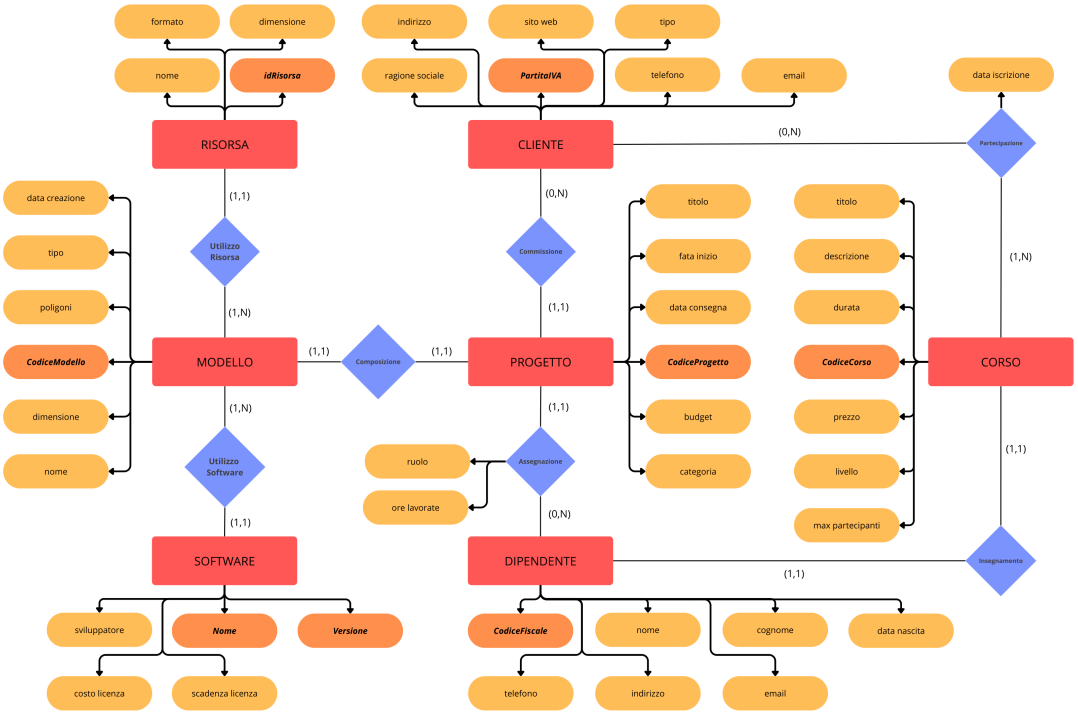
- Il responsabile marketing vuole ottenere i dati dei clienti per inviare promozioni sui corsi di formazione

  (1 Volta al Mese)

- Il direttore artistico vuole ottenere le informazioni sui modelli più complessi (con più poligoni) per ottimizzare le risorse hardware

  (1 Volta al Mese)

2 Schema Entity-Relationship



3 Dizionario dei Dati

3.1 Dizionario dei Dati - Entità

Entità	Descrizione	Attributi	Identificatore
Cliente	cliente dello studio	partita IVA, ragione sociale, email, telefono, indirizzo, sito web	partita IVA
Dipendente	dipendente dello studio	codice fiscale, nome, cognome, data di nascita, telefono, email, specializzazione	codice fiscale
Progetto	progetto commissionato allo studio	codice progetto, titolo, data inizio, data consegna, budget, categoria	codice progetto
Modello	modello 3D creato per un progetto	codice modello, nome, data creazione, tipo, poligoni, dimensione	codice modello
Risorsa	risorsa utilizzata nei modelli	id risorsa, nome, dimensione, formato	id risorsa
Software	software utilizzato per modellazione	nome, versione, sviluppatore, costo licenza, scadenza licenza	nome, versione
Corso	corso di formazione offerto	codice corso, titolo, descrizione, durata, prezzo, livello, max partecipanti	codice corso

Tabella 1: Dizionario dei Dati - Entità

3.2 Dizionario dei Dati - Relazioni

Relazione	**Descrizione**	**Composizione**	**Attributi**
Commissione	cliente che commissiona un progetto	partita IVA cliente, codice progetto	-
Assegnazione	dipendente assegnato a un progetto	codice fiscale dipendente, codice progetto	ore lavorate, ruolo
Composizione	modello appartenente a un progetto	codice modello, codice progetto	-
Utilizzo Risorsa	risorsa utilizzata in un modello	id risorsa, codice modello	-
Utilizzo Software	software utilizzato per un modello	nome software, versione software, codice modello	-
Insegnamento	dipendente che insegna un corso	codice fiscale dipendente, codice corso	-
Partecipazione	cliente che partecipa a un corso	PartitaIVA, codice corso	data iscrizione

Tabella 2: Dizionario dei Dati - Relazioni

4 Vincoli Non Esprimibili

Dall'analisi del testo del problema emergono i seguenti vincoli non esprimibili:

- I corsi non possono avere più iscritti del numero massimo di partecipanti specificato.
- I progetti devono avere almeno un dipendente assegnato con ruolo di project manager.
- Le texture utilizzate nei modelli devono avere una risoluzione adeguata rispetto alla complessità del modello.

5 Tabella dei Volumi

CONCETTO	TIPO	VOLUME
cliente	E	100
dipendente	E	30
progetto	E	200
modello	E	1000
risorsa	E	5000
software	E	15
corso	E	20
commissione	R	200
assegnazione	R	600
composizione	R	1000
utilizzo_risorsa	R	10000
utilizzo_software	R	2000
insegnamento	R	20
partecipazione	R	1000

Tabella 3: Tabella dei Volumi

6 Schema Entity-Relationship Ristrutturato

6.1 Eliminazione generalizzazioni

Per quanto riguarda l'entità cliente, si è scelto di accorpare la generalizzazione azienda/privato nell'entità genitore, aggiungendo l'attributo "tipo" che può assumere i valori "azienda" o "privato". Questo approccio è stato scelto perché da un'analisi delle azioni richieste sulla base di dati risulta che gli accessi a tale entità sono contestuali e non si richiedono frequentemente i record di una sola specializzazione piuttosto che l'altra.

6.2 Analisi ridondanze

La relazione "assegnazione" presenta l'attributo "ore lavorate". Questo attributo, pur essendo molto importante per le statistiche richieste dal direttore dello studio, viene eliminato in quanto campo calcolabile.

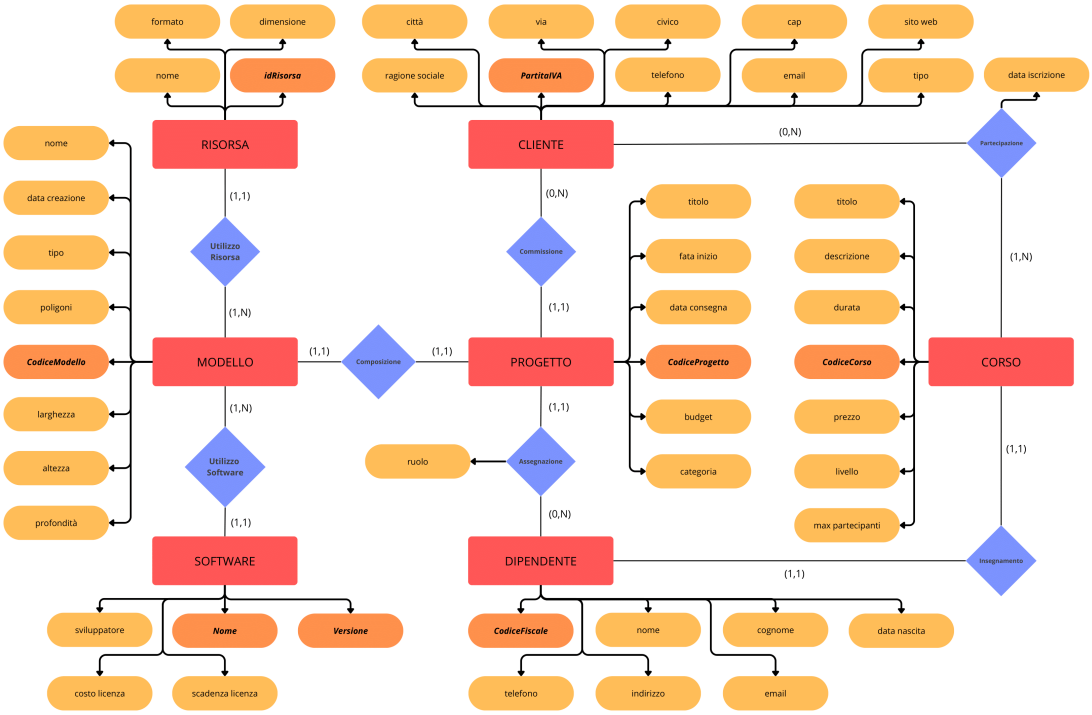
6.3 Eliminazione attributi composti

Vengono inoltre scomposti gli attributi composti in attributi semplici, come "indirizzo" in città, via, civico e CAP, e "dimensione" nella tabella modello in larghezza, altezza e profondità.

6.4 Scelta Identificatori Primari

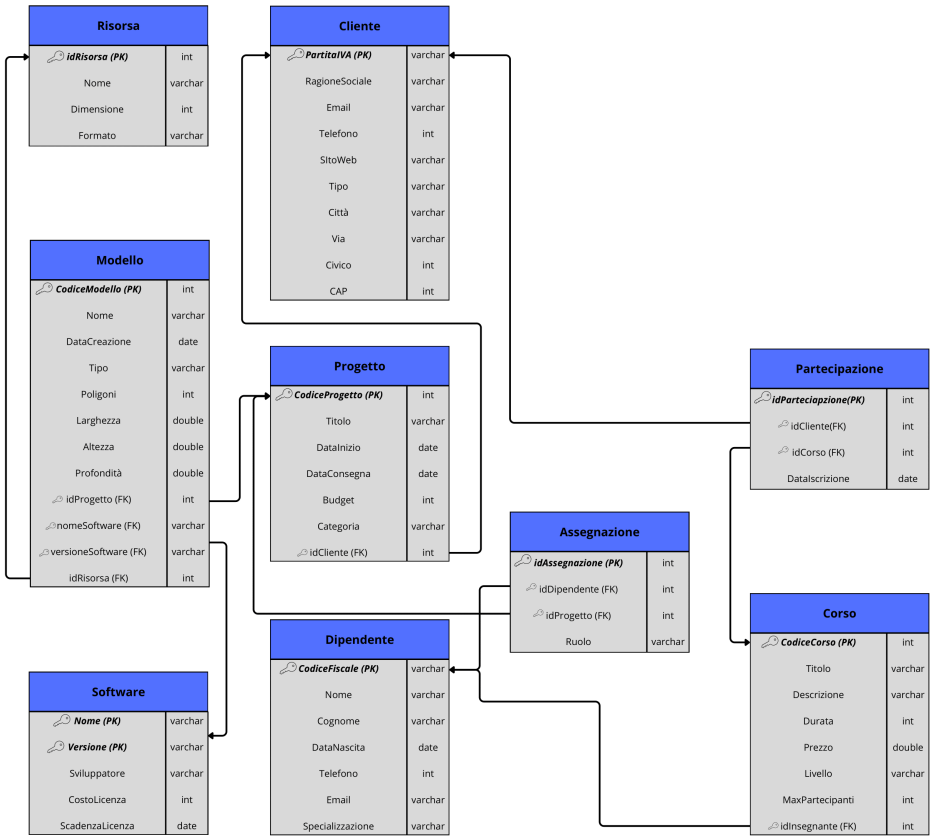
Per ogni entità è stato evidenziato un identificatore primario. Rispettivamente, sono stati scelti:

- Cliente $\longrightarrow$ la partita IVA;
- Dipendente $\longrightarrow$ il codice fiscale;
- Progetto $\longrightarrow$ il codice progetto;
- Modello $\longrightarrow$ il codice modello;
- Risorsa $\longrightarrow$ l'id risorsa;
- Software $\longrightarrow$ la combinazione di nome e versione;
- Corso $\longrightarrow$ il codice corso;



7 Schema Logico

- Cliente (PartitaIVA, RagioneSociale, Email, Telefono, SitoWeb, Tipo, Città, Via, Civico, CAP)
- Dipendente (CodiceFiscale, Nome, Cognome, DataNascita, Telefono, Email, Specializzazione)
- Progetto (CodiceProgetto, Titolo, DataInizio, DataConsegna, Budget, Categoria, IdCliente)
- Modello (CodiceModello, Nome, DataCreazione, Tipo, Poligoni, Larghezza, Altezza, Profondità, IdProgetto, nomeSoftwsre, versioneSoftware, IdRisorsa)
- Risorsa (IdRisorsa, Nome, Dimensione, Formato)
- Software (Nome, Versione, Sviluppatore, CostoLicenza, ScadenzaLicenza)
- Corso (CodiceCorso, Titolo, Descrizione, Durata, Prezzo, Livello, MaxPartecipanti, IdInsegnante)
- Assegnazione (IdAssegnazione, IdDipendente, IdProgetto, Ruolo)
- Partecipazione (IdPartecipazione, IdIscritto, IdCorso, DataIscrizione)



7.1 Normalizzazione

- La base di dati è già in Prima Forma Normale, tutte le colonne sono atomiche.
- La base di dati è già in Seconda Forma Normale, ciascuna colonna dipende dalla primary key.
- La base di dati è già in Terza Forma Normale, ogni attributo dipende solo dalla primary key.

8 Query SQL per la Realizzazione della Base di Dati

```
CREATE DATABASE IF NOT EXISTS `studiomodellazione3d`;
USE `studiomodellazione3d`;

DROP TABLE IF EXISTS `cliente`;
CREATE TABLE `cliente` (
  `partitaIVA` VARCHAR(15) NOT NULL,
  `ragioneSociale` VARCHAR(100) NOT NULL,
  `email` VARCHAR(100) NOT NULL,
  `telefono` INT(20) NOT NULL,
  `sitoWeb` VARCHAR(100) DEFAULT NULL,
  `tipo` ENUM('azienda','privato') NOT NULL,
  `citta` VARCHAR(50) NOT NULL,
  `via` VARCHAR(100) NOT NULL,
  `civico` INT(10) NOT NULL,
  `cap` INT(5) NOT NULL,

  PRIMARY KEY (`partitaIVA`)
);

DROP TABLE IF EXISTS `dipendente`;
CREATE TABLE `dipendente` (
  `codiceFiscale` VARCHAR(16) NOT NULL,
  `nome` VARCHAR(50) NOT NULL,
  `cognome` VARCHAR(50) NOT NULL,
  `dataNascita` DATE NOT NULL,
  `telefono` INT(20) NOT NULL,
  `email` VARCHAR(100) NOT NULL,
  `specializzazione` ENUM('modellazione','texturing','rigging','animazione','
      illuminazione','rendering') NOT NULL,

  PRIMARY KEY (`codiceFiscale`)
);

DROP TABLE IF EXISTS `progetto`;
CREATE TABLE `progetto` (
  `codiceProgetto` INT(10) NOT NULL,
  `titolo` VARCHAR(100) NOT NULL,
  `dataInizio` DATE NOT NULL,
  `dataConsegna` DATE NOT NULL,
  `budget` INT(10) NOT NULL,
  `idCliente` INT(15) NOT NULL,

  PRIMARY KEY (`codiceProgetto`),
  FOREIGN KEY (`idCliente`)
  REFERENCES `cliente` (`partitaIVA`)
);
```

```
DROP TABLE IF EXISTS `modello`;
CREATE TABLE `modello` (
  `codiceModello` INT(10) NOT NULL,
  `nome` VARCHAR(100) NOT NULL,
  `dataCreazione` DATE NOT NULL,
  `tipo` ENUM('organico','inorganico','ambiente','personaggio') NOT NULL,
  `poligoni` INT NOT NULL,
  `larghezza` DOUBLE NOT NULL,
  `altezza` DOUBLE NOT NULL,
  `profondita` DOUBLE NOT NULL,
  `idProgetto` INT(10) NOT NULL,
  `nomeSoftware` VARCHAR(100) NOT NULL,
  `versioneSoftware` VARCHAR(100) NOT NULL,
  `idRisorsa` INT(10) NOT NULL,

  PRIMARY KEY (`codiceModello`),
  FOREIGN KEY (`idProgetto`)
  REFERENCES `progetto` (`codiceProgetto`),
  FOREIGN KEY (`nomeSoftware`, `versioneSoftware`)
  REFERENCES `software` (`nome`, `versione`),
  FOREIGN KEY (`idRisorsa`)
  REFERENCES `risorsa` (`idRisorsa`)
);

DROP TABLE IF EXISTS `risorsa`;
CREATE TABLE `risorsa` (
  `idRisorsa` INT(10) NOT NULL,
  `nome` VARCHAR(100) NOT NULL,
  `dimensione` INT NOT NULL,
  `formato` ENUM('JPG','PNG','TIFF','EXR') NOT NULL,

  PRIMARY KEY (`idRisorsa`)
);

DROP TABLE IF EXISTS `software`;
CREATE TABLE `software` (
  `nome` VARCHAR(100) NOT NULL,
  `versione` VARCHAR(100) NOT NULL,
  `sviluppatore` VARCHAR(100) NOT NULL,
  `costoLicenza` INT(10) NOT NULL,
  `scadenzaLicenza` DATE NOT NULL,

  PRIMARY KEY (`nome`,`versione`)
);
```

```
DROP TABLE IF EXISTS `corso`;
CREATE TABLE `corso` (
  `codiceCorso` INT(10) NOT NULL,
  `titolo` VARCHAR(100) NOT NULL,
  `descrizione` VARCHAR(100) NOT NULL,
  `durata` INT(10) NOT NULL,
  `prezzo` DOUBLE NOT NULL,
  `livello` ENUM('base','intermedio','avanzato') NOT NULL,
  `maxPartecipanti` INT(10) NOT NULL,
  `idInsegnate` VARCHAR(16) NOT NULL,

  PRIMARY KEY (`codiceCorso`),
  FOREIGN KEY (`idInsegnate`)
  REFERENCES `dipendente` (`codiceFiscale`)
);

DROP TABLE IF EXISTS `assegnazione`;
CREATE TABLE `assegnazione` (
  `idAssegnazione` INT(10) NOT NULL,
  `idDipendente` INT(10) NOT NULL,
  `idProgetto` INT(10) NOT NULL,
  `ruolo` VARCHAR(50) NOT NULL,

  PRIMARY KEY (`idAssegnazione`),
  FOREIGN KEY (`idDipendente`)
  REFERENCES `dipendente` (`codiceFiscale`),
  FOREIGN KEY (`idProgetto`)
  REFERENCES `progetto` (`codiceProgetto`)
);

DROP TABLE IF EXISTS `partecipazione`;
CREATE TABLE `partecipazione` (
  `idPartecipazione` INT(10) NOT NULL,
  `idCliente` INT(10) NOT NULL,
  `idCorso` INT(10) NOT NULL,
  `dataIscrizione` DATE NOT NULL,

  PRIMARY KEY (`idPartecipazione`),
  FOREIGN KEY (`idCliente`)
  REFERENCES `cliente` (`partitaIVA`)
  FOREIGN KEY (`idCorso`)
  REFERENCES `corso` (`codiceCorso`)
);
```

9 Query SQL Aggiuntive

9.1 Trigger per controllare che non vengano iscritti più partecipanti del numero massimo per un corso

```
DELIMITER $$
CREATE TRIGGER trg_beforeInsertPartecipazione BEFORE INSERT ON partecipazione
FOR EACH ROW
BEGIN
  DECLARE numIscritti INT;
  DECLARE maxIscritti INT;

  SELECT COUNT(*) INTO numIscritti FROM partecipazione WHERE idCorso = NEW.idCorso;
  SELECT maxPartecipanti INTO maxIscritti FROM corso WHERE codiceCorso = NEW.idCorso;

  IF (numIscritti >= maxIscritti) THEN
    SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Numero massimo di partecipanti
        raggiunto per questo corso';
  END IF;
END $$
DELIMITER ;
```

9.2 Trigger per controllare che un project manager sia sempre assegnato ad ogni progetto

```
DELIMITER $$
CREATE TRIGGER trg_beforeUpdateAssegnazione BEFORE UPDATE ON assegnazione
FOR EACH ROW
BEGIN
  DECLARE pmCount INT;

  IF (OLD.ruolo = 'project manager' AND NEW.ruolo != 'project manager') THEN
    SELECT COUNT(*) INTO pmCount FROM assegnazione
    WHERE idProgetto = NEW.idProgetto AND ruolo = 'project manager' AND idDipendente !=
        NEW.idDipendente;

    IF (pmCount = 0) THEN
      SIGNAL SQLSTATE '45002' SET MESSAGE_TEXT = 'Ogni progetto deve avere almeno un
          project manager';
    END IF;
  END IF;
END $$
DELIMITER ;
```

9.3 Vista per ottenere il carico di lavoro dei dipendenti nei progetti attivi

```
CREATE VIEW v_caricoLavoroDipendenti AS
SELECT
  d.nome,
  d.cognome,
  d.specializzazione,
  p.titolo AS progetto,
  a.ruolo
FROM dipendente d
INNER JOIN assegnazione a ON d.codiceFiscale = a.idDipendente
INNER JOIN progetto p ON a.idProgetto = p.codiceProgetto
WHERE p.dataConsegna >= CURDATE()
ORDER BY d.cognome, d.nome, p.titolo;

SELECT * FROM v_caricoLavoroDipendenti;
```

9.4 Stored Procedure per ottenere i progetti completati nell'ultimo mese

```
DELIMITER $$
CREATE PROCEDURE sp_getProgettiCompletatiUltimoMese()
BEGIN
  SELECT
    p.codiceProgetto,
    p.titolo,
    p.dataInizio,
    p.dataConsegna,
    p.budget,
    c.ragioneSociale AS cliente,
    c.partitaIVA
  FROM progetto p
  INNER JOIN cliente c ON p.clienteID = c.partitaIVA
  WHERE p.dataConsegna BETWEEN DATE_SUB(CURDATE(), INTERVAL 1 MONTH) AND CURDATE()
  ORDER BY p.dataConsegna DESC;
END $$
DELIMITER ;

CALL sp_getProgettiCompletatiUltimoMese();
```

9.5 Vista per ottenere statistiche sull'utilizzo dei software

```
CREATE VIEW v_statisticheUtilizzoSoftware AS
SELECT
  s.nome,
  s.versione,
  s.sviluppatore,
  s.scadenzaLicenza,
  COUNT(DISTINCT us.idModello) AS numeroModelli,
  COUNT(DISTINCT m.idProgetto) AS numeroProgetti
FROM software s
LEFT JOIN utilizzoSoftware us ON s.nome = us.softwareNome AND s.versione = us.
    softwareVersione
LEFT JOIN modello m ON us.idModello = m.codiceModello
GROUP BY s.nome, s.versione
ORDER BY numeroModelli DESC;

SELECT * FROM v_statisticheUtilizzoSoftware;
```

9.6 Vista per ottenere i dati dei clienti per promozioni

```
CREATE VIEW v_datiClientiPerPromozioni AS
SELECT
  c.partitaIVA,
  c.ragioneSociale,
  c.email,
  c.telefono,
  COUNT(DISTINCT p.codiceProgetto) AS numeroProgetti,
  MAX(p.dataConsegna) AS ultimoProgetto
FROM cliente c
LEFT JOIN progetto p ON c.partitaIVA = p.clienteID
GROUP BY c.partitaIVA
ORDER BY numeroProgetti DESC;

SELECT * FROM v_datiClientiPerPromozioni;
```

9.7 Vista per ottenere i modelli più complessi

```
CREATE VIEW v_modelliPiuComplessi AS
SELECT
  m.codiceModello,
  m.nome,
  m.tipo,
  m.poligoni,
  m.dimensione,
  p.titolo AS progetto,
  GROUP_CONCAT(DISTINCT CONCAT(s.nome, ' ', s.versione)) AS software
FROM modello m
INNER JOIN progetto p ON m.idProgetto = p.codiceProgetto
LEFT JOIN utilizzoSoftware us ON m.codiceModello = us.idModello
LEFT JOIN software s ON us.softwareNome = s.nome AND us.softwareVersione = s.versione
GROUP BY m.codiceModello
ORDER BY m.poligoni DESC
LIMIT 20;

SELECT * FROM v_modelliPiuComplessi;

CALL sp_getModelliPiuComplessi();
```

9.8 Stored Procedure per ottenere una lista di email di tutti i clienti, per scopi promozionali

```
DELIMITER $$
CREATE PROCEDURE sp_listaEmail(OUT listaemail VARCHAR(10000))
BEGIN
  DECLARE finito INTEGER DEFAULT 0;
  DECLARE var_email VARCHAR(100) DEFAULT "";
  DECLARE email_cursor CURSOR FOR SELECT email FROM cliente;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET finito = 1;

  OPEN email_cursor;

  WHILE (finito = 0) DO
    FETCH email_cursor INTO var_email;
    IF finito = 0 THEN
      SET listaemail = CONCAT(var_email, "; ", listaemail);
    END IF;
  END WHILE;

  CLOSE email_cursor;
END $$
DELIMITER ;

SET @lista_emails = '';
CALL sp_listaEmail(@lista_emails);
```