Riassunto Finale

Prof. Francesco Gobbi
I.I.S.S. Galileo Galilei, Ostiglia

16 giugno 2025

1. Archivio Digitale

Definizione e Caratteristiche

- Un **archivio** è un insieme organizzato di informazioni, legate da un nesso logico.
- Le informazioni sono rappresentate secondo un formato interpretabile e sono registrate su un supporto (carta, dischi, memorie digitali).
- Le principali caratteristiche di un archivio digitale sono:
  - **Nesso logico**: i dati riguardano uno stesso argomento.
  - **Formato**: strutturazione che permette interpretazione e utilizzo.
  - **Persistenza**: i dati vengono conservati anche dopo la chiusura di un'applicazione.
  - **Facilità di consultazione**: i dati sono organizzati per essere facilmente trovati e usati.
- Gli archivi digitali utilizzano memorie di massa per la conservazione dei dati: la scelta del supporto (hard disk, SSD, cloud, ecc.) influenza la velocità e le modalità di accesso.
- **Record e Campi**: un archivio è composto da record (righe) e ogni record è suddiviso in campi (colonne), ciascuno con un valore specifico.

Vantaggi e Tipologie di Organizzazione

- **Vantaggi**:
  - Conservazione permanente dei dati
  - Accesso veloce
  - Aggiornamento e modifica dei dati facilitati
  - Riduzione dell'uso di carta
- **Organizzazione degli archivi**:
  - **Sequenziale**: i record sono in ordine predefinito. Adatto per elaborazioni su tutti i dati.
  - **Ad accesso diretto**: accesso rapido tramite chiave o funzione hash, utile per ricerche veloci.
  - **A indici**: utilizzo di strutture di indice per associare chiavi agli indirizzi dei record.

2. Database

Definizione e Caratteristiche

- Una **base di dati** (database) è un insieme organizzato di dati archiviati in modo strutturato per un facile accesso, gestione e manipolazione.
- I database superano i limiti degli archivi tradizionali garantendo:
  - **Accesso rapido**
  - **Organizzazione ottimale**
  - **Sicurezza e integrità dei dati**
  - **Supporto decisionale**
- I dati sono organizzati in tabelle (modello relazionale), con campi (colonne) e record (righe).
- Concetti fondamentali:
  - **Entità**: oggetti o concetti distinti (es. Studente, Corso).
  - **Attributi**: proprietà delle entità (es. nome, cognome).
  - **Relazioni**: legami tra entità.
  - **Chiave primaria**: campo che identifica univocamente ogni record.

Vantaggi dei Database

- **Eliminazione della ridondanza**: i dati non sono duplicati in archivi diversi.
- **Facilità di accesso**: interrogazioni semplici anche su grandi quantità di dati.
- **Integrità e consistenza**: regole e vincoli impediscono l'inserimento di dati errati.
- **Indipendenza dei dati**: modifiche alla struttura non influenzano le applicazioni.
- **Multiutenza e sicurezza**: controllo degli accessi e gestione dei permessi.
- **Backup e ripristino**: funzioni integrate per il salvataggio e recupero dei dati.

3. DBMS (DataBase Management System)

Cos'è e Caratteristiche

- Un **DBMS** è un software per la gestione delle basi di dati.
- Principali funzioni:
  - **Organizzare** e strutturare i dati su supporti digitali.
  - **Permettere l'accesso, la manipolazione e l'aggiornamento** dei dati.
  - **Gestire l'integrità e la sicurezza** delle informazioni.
  - **Gestione degli utenti** e dei loro permessi.
  - **Gestione delle transazioni**, garantendo le proprietà ACID:
    - **Atomicità**: tutte le operazioni vengono completate o nessuna.
    - **Consistenza**: lo stato del database resta valido.
    - **Isolamento**: le transazioni contemporanee non interferiscono tra loro.
    - **Durabilità**: i dati confermati sono permanenti.
  - **Backup e ripristino**: protezione da perdite accidentali di dati.
- Esempi di DBMS: MySQL, PostgreSQL, Oracle, SQL Server.

Componenti di un DBMS

- **Motore di archiviazione**: gestisce la memorizzazione fisica.
- **Linguaggio di interrogazione (SQL)**: permette la gestione e l'estrazione dei dati.
- **Sistema di gestione delle transazioni**: assicura le proprietà ACID.
- **Sistema di controllo degli accessi**: regola permessi e sicurezza.
- **Sistema di backup e ripristino**: previene la perdita di dati.
- **Dizionario dei dati**: contiene i metadati (struttura e regole del database).

Vantaggi rispetto ai sistemi file-based

- Eliminazione della ridondanza e inconsistenza dei dati.
- Interrogazioni non predefinite e personalizzabili (es. SQL).
- Controllo della concorrenza tra utenti.
- Maggiore sicurezza e gestione dei permessi a diversi livelli.
- Indipendenza logica e fisica dei dati dalle applicazioni.

4. Schema dei Linguaggi per DBMS

- **DDL (Data Definition Language)**: definizione di tabelle, vincoli, indici.
- **DML (Data Manipulation Language)**: inserimento, modifica, cancellazione dati.
- **QL (Query Language)**: interrogazione e ricerca di dati.

5. Cos'è il Modello Concettuale

La **progettazione concettuale** è la prima fase della progettazione di una base di dati e serve a rappresentare la realtà da modellare in modo astratto e indipendente dalla tecnologia. Si utilizza per mettere in relazione la visione degli utenti e quella dei progettisti, garantendo una rappresentazione chiara e condivisibile anche da persone non tecniche.

Obiettivo: fornire una visione precisa, completa e non ambigua delle informazioni essenziali, tramite schemi e formalismi semplici.

Componenti del Modello E/R (Entity/Relationship)

- **Entità:** Oggetti concreti o astratti della realtà di interesse (ad esempio: Persona, Automobile, Corso). Un'entità è rappresentata come una tabella nel database.
- **Attributi:** Caratteristiche o proprietà che descrivono ciascuna entità o relazione (es: nome, cognome, targa, modello, data di nascita).
- **Relazioni/Associazioni:** Legami che collegano due o più entità (es: uno studente *iscritto* a un corso, un cliente *noleggia* un'auto).

Le Entità

- Ogni entità ha delle **istanze** (i singoli oggetti reali, cioè le righe della tabella).
- Ogni entità deve essere identificabile in modo univoco tramite una **chiave primaria** (PK, Primary Key): uno o più attributi che distinguono un'istanza dalle altre.

Gli Attributi

- Ogni attributo ha un **dominio** (insieme dei valori possibili).
- Gli attributi possono essere obbligatori o facoltativi, e possono avere valore nullo.
- Esistono attributi anche sulle relazioni, quando è necessario caratterizzare il legame tra due entità.

Le Relazioni

- Una relazione associa istanze di due (o più) entità.
- Le relazioni possono avere attributi propri.
- Le relazioni hanno **cardinalità**, cioè specificano il numero minimo e massimo di partecipazioni delle istanze delle entità coinvolte.

Le Cardinalità delle Relazioni

- **Uno a uno (1:1):** ogni istanza di A è collegata a una sola istanza di B e viceversa.
- **Uno a molti (1:N):** ogni istanza di A può essere collegata a più istanze di B, ma ogni istanza di B è collegata a una sola di A.
- **Molti a molti (N:M):** ogni istanza di A può essere collegata a molte istanze di B e viceversa.
- **Zero, uno, molti:** la cardinalità indica anche il minimo e massimo di partecipazione (es: (0,1), (1,N)).

Esempio Lineare di Modello Concettuale (E/R) – Noleggio Auto

Descrizione: Un'azienda di autonoleggio vuole gestire i dati relativi a clienti, veicoli, noleggi e incidenti.
N.B. L'esempio proposto è lineare e non grafico, come viene invece richiesto per il modello concettuale, utilizzando la notazione "a zampa di gallina".

Entità principali:

Cliente (IdCliente, Nome, Cognome, Indirizzo, Telefono, NumeroPatente)

Auto (Targa, Modello, Marca, AnnoImmatricolazione, Categoria)

Noleggio (IdNoleggio, DataInizio, DataFine, ImportoTotale)

Incidente (IdIncidente, Descrizione, Data)

Legenda:

- Gli attributi sottolineati sono chiavi primarie (PK).

Relazioni:

- *Un* **Cliente** può effettuare **molti Noleggi**; ogni **Noleggio** è associato a **un solo Cliente**.
- *Un'* **Auto** può essere noleggiata in **molti Noleggi**; ogni **Noleggio** riguarda **una sola Auto**.
- *Un* **Noleggio** può avere **zero o più Incidenti**; ogni **Incidente** è riferito a un solo **Noleggio**.

Quindi:

- **Cliente** (1) — (N) **Noleggio**: ogni cliente può effettuare più noleggi, ogni noleggio è di un solo cliente.
- **Auto** (1) — (N) **Noleggio**: ogni auto può essere noleggiata più volte, ogni noleggio riguarda una sola auto.
- **Noleggio** (1) — (N) **Incidente**: ogni noleggio può avere più incidenti, ogni incidente è legato a un solo noleggio.

N.B.

- Il modello concettuale non definisce ancora tabelle o database, ma solo le entità, le relazioni e le regole principali del sistema informativo.
- È fondamentale chiarire i concetti di entità forte e debole, chiave primaria, attributo, relazione e cardinalità.
- Il modello E/R è il punto di partenza per il passaggio al modello logico e poi a quello fisico.

6. Il Modello Relazionale

Definizione

- Proposto da **E. F. Codd** nel 1970.
- Modella i dati come **insiemi di tabelle** (relazioni).
- Ogni tabella rappresenta una relazione tra insiemi di oggetti (entità).

Concetto di Relazione

- Una **relazione** è un sottoinsieme del prodotto cartesiano di due o più insiemi.
- Ogni relazione ha:
  - **Attributi** (colonne): proprietà degli oggetti.
  - **Tuple** (righe): istanze della relazione.
  - **Dominio**: insieme dei valori ammessi per ogni attributo.
- Esempio: se $A_1 = \{4, 9, 16\}$ e $A_2 = \{2, 3\}$, il prodotto cartesiano è:

$$A_1 \times A_2 = \{(4,2), (4,3), (9,2), (9,3), (16,2), (16,3)\}$$

Una relazione Q può essere $Q = \{(4,2), (9,3)\}$ dove x è il quadrato di y.

Caratteristiche delle Tabelle Relazionali

- Ogni riga è unica (non ci sono duplicati).
- Esiste una **chiave primaria** che identifica univocamente ogni tupla.
- Gli attributi sono omogenei (stesso tipo di dati).
- L'ordine di righe e colonne non ha significato.
- Tutti i valori sono **atomici** (non scomponibili).

Esempio Tabella

Matricola	Nome	Cognome	Classe
12345	Luca	Rossi	5A
23456	Anna	Bianchi	5B

Chiave Primaria e Chiave Esterna

- **Chiave primaria**: Attributo o insieme di attributi che identifica univocamente una tupla (es. Matricola).
- **Chiave esterna**: Attributo che fa riferimento alla chiave primaria di un'altra tabella, creando collegamenti tra tabelle.

7. Notazione Relazionale

Operazioni fondamentali

- **Selezione** (σ): Estrae le righe che soddisfano una condizione.
  Esempio: $\sigma_{Classe='5A'}(\text{Studenti})$
- **Proiezione** (π): Estrae solo alcune colonne.
  Esempio: $\pi_{Nome,\,Cognome}(\text{Studenti})$
- **Join** (): Combina due tabelle in base a una condizione di uguaglianza (Equi Join) o sugli attributi comuni (Join Naturale).
- **Unione** ($\cup$), **Intersezione** ($\cap$), **Differenza** ($-$): Operazioni insiemistiche sulle relazioni.

Esempi

- **Selezione**:
  $\sigma_{IDDipendente>2}(\text{Dipendenti})$
- **Proiezione**:
  $\pi_{Nome,\,Cognome}(\text{Dipendenti})$
- **Equi Join**:
  Dipendenti $_{IDDipendente=IDDipendente}$ Progetti
- **Join Naturale**:
  Dipendenti Progetti

8. Dal Modello Concettuale al Modello Logico

Regole di Derivazione

- **Ogni entità** del modello E/R diventa una tabella nel modello relazionale.
- **Gli attributi** delle entità diventano attributi della tabella.
- **L'identificatore unico** dell'entità diventa la chiave primaria.
- **Associazioni uno-a-uno**: Un'unica tabella combina gli attributi delle entità collegate.
- **Associazioni uno-a-molti**: La chiave primaria dell'entità "a uno" diventa chiave esterna nella tabella "a molti".
- **Associazioni molti-a-molti**: Si crea una tabella di collegamento con le chiavi primarie delle due entità.

Esempi pratici

- **Uno a uno**: Persona & Passaporto → Tabella unica con attributi di entrambi.
- **Uno a molti**: Dipartimento & Impiegato
  Tabella Impiegati: ..., ID Dipartimento (chiave esterna)
- **Molti a molti**: Studenti & Corsi
  Tabella StudentiCorsi: ID Studente, ID Corso, Data Iscrizione

N.B.

Nel passaggio dal modello concettuale a quello logico:

- I nomi delle tabelle sono solitamente al plurale.
- È importante scegliere bene i nomi e la posizione degli attributi.

9. Normalizzazione del Modello Logico

Definizione: La normalizzazione delle relazioni è un processo fondamentale nella progettazione delle basi di dati, con l'obiettivo di:

- Ridurre la ridondanza dei dati.
- Evitare anomalie nelle operazioni di inserimento, aggiornamento e cancellazione.
- Garantire una struttura logica ottimale per le tabelle.

Perché è importante normalizzare?

- Duplicazione inutile dei dati e spreco di spazio su disco.
- Anomalie come incoerenze negli aggiornamenti.
- Difficoltà nella gestione e nell'integrità dei dati.

Esempio:
Tabella non normalizzata degli ordini:

IDOrdine	Cliente	Prodotto	Indirizzo
1	Mario Rossi	Laptop	Via Roma, 10
2	Mario Rossi	Smartphone	Via Roma, 10
3	Anna Bianchi	Tablet	Via Milano, 20

Problema: L'indirizzo di Mario Rossi si ripete più volte; aggiornare l'indirizzo richiede modifiche multiple.

10. Il Processo di Normalizzazione

La normalizzazione si basa sull'uso delle **forme normali** (NF) per organizzare correttamente i dati nelle tabelle:

1. **Identificare tutte le dipendenze funzionali e le chiavi candidate.**
2. **Individuare le violazioni delle regole di normalizzazione.**
3. **Scomporre la tabella in relazioni più semplici e prive di ridondanze.**
4. **Ripetere il processo finché tutte le tabelle rispettano le forme normali desiderate.**

Dipendenza funzionale: Un attributo Y dipende funzionalmente da X (si scrive $X \to Y$) se il valore di X determina univocamente quello di Y.

11. Le Tre Forme Normali

1. Prima Forma Normale (1NF)

- Tutti gli attributi devono contenere valori **atomici** (non suddivisibili).
- **Non sono ammessi** gruppi ripetuti di attributi.
- *Esempio:* Una colonna "Telefoni" con più numeri separati da virgola viola la 1NF.

2. Seconda Forma Normale (2NF)

- La tabella deve essere in 1NF.
- Ogni attributo non chiave deve dipendere **dall'intera chiave primaria** (assenza di dipendenze parziali).
- *Esempio:* In una tabella con chiave composta $(IDOrdine, IDProdotto)$, un attributo come "NomeProdotto" che dipende solo da $IDProdotto$ viola la 2NF.

3. Terza Forma Normale (3NF)

- La tabella deve essere in 2NF.
- Non devono esserci **dipendenze transitive** tra attributi non chiave.
- *Esempio:* In "Dipendenti(ID, Nome, IDDipartimento, NomeDipartimento)", "NomeDipartimento" dipende da "IDDipartimento" (che non è chiave primaria): per rispettare la 3NF va separato in un'altra tabella "Dipartimenti".

12. Esempio Semplificato di Normalizzazione

Tabella iniziale (non normalizzata):

ID	Nome	Progetto	Dipartimento
1	Laura Bianchi	Progetto Alpha	Informatica
2	Marco Rossi	Progetto Beta	Fisica
3	Laura Bianchi	Progetto Gamma	Informatica

Dopo la normalizzazione (3NF):

- Dipendenti(ID, Nome)
- Progetti(IDProgetto, NomeProgetto, IDDipartimento)
- Dipartimenti(IDDipartimento, NomeDipartimento)

Vantaggi della Normalizzazione

- Riduzione della ridondanza e dello spazio occupato.
- Maggiore coerenza e integrità dei dati.
- Operazioni più semplici e affidabili (aggiornamenti, cancellazioni, inserimenti).
- Struttura logica ottimale.

13. Il Modello Fisico

Definizione: Il modello fisico è la trasposizione del modello logico in un database reale su DBMS (come MySQL). Comprende:

- **Tipi di dato** (`INT`, `VARCHAR`, `DATE`...)
- **Vincoli** (`PRIMARY KEY`, `FOREIGN KEY`, `UNIQUE`, `CHECK`)
- **Indici** per ottimizzare le query
- **Gestione della memoria**

Obiettivo: Garantire integrità, performance e coerenza dei dati.
Differenza con il modello logico:

- Il modello logico è indipendente dal DBMS e astratto.
- Il modello fisico è concreto e specifico per il DBMS usato.

14. Creazione di Tabelle (DDL): Attributi chiave ed Esempio

DDL (Data Definition Language): comandi per definire la struttura delle tabelle.
Vincoli principali:

- **PRIMARY KEY**: identifica univocamente ogni riga.
- **FOREIGN KEY**: collega una tabella ad un'altra (integrità referenziale).
- **UNIQUE**: valori unici in una colonna.

- **NOT NULL**: obbliga il campo ad avere un valore.
- **AUTO_INCREMENT**: numerazione automatica.

Esempio concreto (tabella `studenti`):

```
CREATE TABLE studenti (
    id INT PRIMARY KEY AUTO_INCREMENT,   -- Chiave primaria
    nome VARCHAR(50) NOT NULL,
    cognome VARCHAR(50) NOT NULL,
    classe_id INT NOT NULL,
    FOREIGN KEY (classe_id) REFERENCES classi(id) -- Chiave esterna
);
```

15. Inserimento di dati (DML): Attributi chiave ed Esempio

DML (Data Manipulation Language): comandi per inserire, modificare o cancellare dati.

Esempio di inserimento:

```
INSERT INTO studenti (nome, cognome, classe_id)
VALUES ('Mario', 'Rossi', 1);
```

Nota:

- `id` viene assegnato automaticamente grazie a `AUTO_INCREMENT` (chiave primaria).
- `classe_id` deve esistere nella tabella `classi` (chiave esterna).

16. Query sul Database: Attributi e Campi Chiave, Esempi

Query SQL: estraggono, aggregano o modificano dati secondo condizioni specifiche.

a) Selezione dati con JOIN e chiavi

Esempio: Selezionare tutti gli studenti con la loro classe.

```
SELECT s.nome, s.cognome, c.nome AS Classe
FROM studenti AS s, classi AS c
WHERE s.classe_id = c.id;
```

b) Aggregazione e raggruppamento

Contare il numero di studenti per classe:

```
SELECT c.nome AS Classe, COUNT(s.id) AS NumeroStudenti
FROM studenti AS s, classi AS c
WHERE s.classe_id = c.id
GROUP BY c.nome;
```

c) JOIN avanzati

- **LEFT JOIN**: restituisce tutte le righe della tabella sinistra e quelle corrispondenti della destra (le mancanti risultano `NULL`).
- **RIGHT JOIN**: inverso del precedente.
- **FULL JOIN**: MySQL lo simula con `UNION` tra `LEFT JOIN` e `RIGHT JOIN`.

Esempio di LEFT JOIN:

```
SELECT s.nome, i.corso
FROM studenti AS s
LEFT JOIN iscrizioni AS i ON s.id = i.studenteId;
```

d) Funzioni di aggregazione

`COUNT`, `MAX`, `MIN`, `AVG` calcolano rispettivamente quantità, massimo, minimo e media.

```
SELECT COUNT(*) AS totale_esami,
       MAX(voto) AS voto_massimo,
       MIN(voto) AS voto_minimo,
       AVG(voto) AS media_voti
FROM esami;
```

N.B. In questo caso l'utilizzo delle funzioni di aggregazione non è allineato con il `GROUP BY` in quanto vengono considerate tutte le righe della tabella, senza quindi fare raggruppamenti.

e) Query avanzate: filtrare con GROUP BY e HAVING

```
SELECT classe, COUNT(*) AS num_studenti, AVG(voto) AS media_voti
FROM esami
GROUP BY classe
HAVING AVG(voto) >= 25;
```

Solo le classi con media dei voti maggiore o uguale a 25 verranno mostrate.

17. Altri Esempi Utili (Esercizi tipici)

- Elenco ordini con nome cliente:

```
SELECT o.id_ordine, o.data_ordine, c.nome, c.cognome
FROM Ordine AS o, Cliente AS c
WHERE o.id_cliente = c.id_cliente;
```

- Top 3 clienti per spesa:

```
SELECT c.nome, c.cognome, SUM(p.prezzo * i.quantita) AS spesa_totale
FROM Cliente AS c, Ordine AS o, Ordine_Item AS i, Prodotto AS p
WHERE c.id_cliente = o.id_cliente AND o.id_ordine = i.id_ordine AND i.id_prodotto =
    p.id_prodotto
GROUP BY c.id_cliente
HAVING spesa_totale > 500
ORDER BY spesa_totale DESC
LIMIT 3;
```

- Prodotti mai ordinati:

```
SELECT nome
FROM Prodotto AS p
WHERE NOT EXISTS (
    SELECT 1 FROM Ordine_Item i
    WHERE i.id_prodotto = p.id_prodotto
);
```

18. Esempio Completo: MySQL dalla Creazione alle Query

18.1 Creazione delle Tabelle

Supponiamo di voler gestire una semplice libreria con le tabelle `autori`, `libri` e `prestiti`.

```
CREATE TABLE autori (
  id INT PRIMARY KEY AUTO_INCREMENT,
  nome VARCHAR(50) NOT NULL,
  cognome VARCHAR(50) NOT NULL
);

CREATE TABLE libri (
  id INT PRIMARY KEY AUTO_INCREMENT,
  titolo VARCHAR(100) NOT NULL,
  autore_id INT NOT NULL,
  anno_pubblicazione INT,
  FOREIGN KEY (autore_id) REFERENCES autori(id)
);

CREATE TABLE prestiti (
  id INT PRIMARY KEY AUTO_INCREMENT,
  libro_id INT NOT NULL,
  data_prestito DATE NOT NULL,
  restituito BOOLEAN DEFAULT 0,
  FOREIGN KEY (libro_id) REFERENCES libri(id)
);
```

18.2 Inserimento di Dati di Esempio

```
-- Inserimento autori
INSERT INTO autori (nome, cognome) VALUES
  ('Italo', 'Calvino'),
  ('Umberto', 'Eco');

-- Inserimento libri
INSERT INTO libri (titolo, autore_id, anno_pubblicazione) VALUES
  ('Il barone rampante', 1, 1957),
  ('Se una notte d\'inverno un viaggiatore', 1, 1979),
  ('Il nome della rosa', 2, 1980);

-- Inserimento prestiti
INSERT INTO prestiti (libro_id, data_prestito, restituito) VALUES
  (1, '2025-06-01', 1),
  (3, '2025-06-10', 0);
```

18.3 Query Esemplificative e Output

a) Elenco dei libri con nome e cognome dell'autore

```
SELECT l.titolo, a.nome, a.cognome
FROM libri AS l, autori AS a
WHERE l.autore_id = a.id;
```

Output atteso:

Titolo	Nome	Cognome
Il barone rampante	Italo	Calvino
Se una notte d'inverno un viaggiatore	Italo	Calvino
Il nome della rosa	Umberto	Eco

b) Elenco libri attualmente in prestito (non restituiti), con titolo e data prestito

```sql
SELECT l.titolo, p.data_prestito
FROM libri AS l, prestiti AS p
WHERE l.id = p.libro_id AND p.restituito = 0;
```

Output atteso:

Titolo	Data Prestito
Il nome della rosa	2025-06-10

c) Numero totale di prestiti per ogni libro

```sql
SELECT l.titolo, COUNT(p.id) AS numero_prestiti
FROM libri AS l, prestiti AS p
WHERE l.id = p.libro_id
GROUP BY l.titolo

UNION

SELECT l.titolo, 0 AS numero_prestiti
FROM libri AS l
WHERE l.id NOT IN (SELECT libro_id FROM prestiti);
```

Output atteso:

Titolo	Numero Prestiti
Il barone rampante	1
Se una notte d'inverno un viaggiatore	0
Il nome della rosa	1

N.B. L'UNION in questo caso serve per unire tutti i libri che hanno un numero di prestiti, con quelli che non sono mai stati presi in presisto neanche una volta.

d) Elenco autori che hanno almeno un libro in prestito

```sql
SELECT DISTINCT a.nome, a.cognome
FROM autori AS a, libri AS l, prestiti AS p
WHERE a.id = l.autore_id
  AND l.id = p.libro_id
  AND p.restituito = 0;
```

Output atteso:

Nome	Cognome
Umberto	Eco