Appunti Documentazione Digitale, aa. 2020/2021 (prof.ssa Damiano - Università di Torino)

Fondamenti di informatica
Università di Torino
38 pag.

DOCUMENTAZIONE DIGITALE

"Capire l'architettura, sia concettuale che software, di un archivio digitale."

ARCHIVI DIGITALI (modelli di archivio e le architetture e i protocolli che si utilizzano per realizzarli)
Gli archivi digitali sono nati alla fine degli anni '90, e si tratta di un argomento recente e in continua evoluzione: il loro cuore è composto dalle **basi di dati**.
Gli archivi digitali sono dei formati digitali: il termine che si usa nel web per parlare di un "oggetto digitale" è **resource** ("risorsa") → es. URL (la "r" sta per "resourse"), sigla che sta per "Uniform Resourse Locator", quindi "indirizzo unico/uniforme per localizzare le risorse". "Resourse" è un termine generico utilizzato per riferirsi ad un documento in formato digitale, che può essere un'immagine, un testo, un modello 3 – D, etc.
Alcune risorse nascono in formato digitale, mentre altre vengono digitalizzate, cioè vengono convertite in formato digitale attraverso un processo di **digitalizzazione** ("digitalization"): quando si digitalizza una risorsa, per permettere la conservazione e lo studio di quel determinato bene, lo si fa seguendo delle procedure standard che assicurano che il formato digitale:
_ sia **di qualità** (che, quindi, abbia le caratteristiche di dimensione, risoluzione, etc. richieste per assolvere al meglio il compito per cui si attua tale digitalizzazione)
_ sia **inter - operabile** (cioè un formato standard reperibile da tutti).
Dunque, in un flusso di lavoro, anche la creazione delle risorse in formato digitale a partire da quelle fisiche deve seguire delle procedure standard.
Il fatto che ogni giorno vengono prodotti, dalle attività quotidiane di tutti, una grande quantità di documenti in formato digitale alimenta il massiccio processo di produzione di contenuti digitali, ed ha portato al fenomeno chiamato "**data deluge**" ("alluvione di dati"): il web ne presenta un'altissima quantità che, pur non essendo eccessiva, è talmente elevata (anche in ambito culturale) di risorse digitale, spesso in una sorta di "osmosi" con i social media, che rende difficile agli utenti di orientarsi.
In questo panorama, diventa sempre più importante riuscire a strutturare i contenuti, per renderli più fruibili e per aiutare le persone a focalizzare la propria attenzione.
_ *Digital libraries*: biblioteche digitali, strutturate secondo una struttura che è maggiormente focalizzata sull' "oggetto libro" (che, quindi, permette di vedere il formato originale sia all'interno di una resa fotografica).
LICENZE → quando si mette a disposizione un contenuto, bisogna mettere bene in evidenza il "consenso" per quanto riguarda l'utilizzo di una determinata risorsa → **digital rights management**: gestione dei diritti nel contesto digitale.
_ *Digital archives*: contengono materiali più eterogenei, con formati diversi. Spesso, un archivio deriva dall'attività di un'istituzione o di una fondazione, ma esistono anche archivi personali, che raccolgono lettere, opere, bozzetti e tutto il materiale che riguardano la carriera di un determinato artista.
_ *Digital repository*: un "deposito" scarsamente strutturato di documenti e materiali digitali di vario genere (può essere anche uno stadio precedente alla creazione di un archivio strutturato).

SCOPI DELL'ARCHIVIAZIONE
1) **CONSERVAZIONE**: conservare a lungo termine un determinato bene / risorsa, cosicché, attraverso il digitale, rimanga traccia di qualcosa che eventualmente può essere deperibile.

Comunque, i formati digitali presentano anch'essi delle problematiche di conservazione: es. i formati digitali sono legati ad una piattaforma che nel tempo si evolve, e ci si può trovare a formati non più apribili in quanto non esistono programmi originali.

2) **FRUIZIONE**: ricerca e accesso alle risorse da parte del pubblico, che può essere un pubblico generico o un pubblico specialistico.

3) **STUDIO** della risorsa.

ICCD (Istituto Centrale per il Catalogo e la Documentazione) → istituto italiano che si occupa di catalogare i beni culturali e documentarli, quindi accompagnarli con immagini, video, etc. che permettono di descrivere il bene. ICCD dà una definizione di catalogazione.

CATALOGAZIONE: attività di registrazione, descrizione e classificazione di tutte le tipologia di beni culturali; si tratta di individuare e conoscere i beni, documentarli in modo opportuno e archiviare le informazioni raccolte secondo precisi criteri. Aree di interesse: archeologico, architettonico, storico – artistico, demo – etno – antropologico.

DISTINZIONE DI DUE DEFINIZIONI DI ARCHIVIO
1) "una raccolta di documenti che si produce come effetto dell'attività di un ente" (**vincolo naturale**);
2) nell'ambito dei beni culturali, "è una raccolta finalizzata a scopi di conservazione e di studio" (**vincolo volontario**).

In un archivio, i documenti / opere (quindi, le risorse) sono descritti in un catalogo, ovvero un insieme di "schede" ("records") che descrivono le risorse in base alle loro proprietà, ovvero le loro caratteristiche, che prendono il nome di megadati (es. autore, titolo, ecc.).

MEGADATI ("dati sui dati")
Sono la descrizione dei dati.
Contengono informazioni diverse:

- Amministrative e gestionali, come proprietà, responsabilità, provenienza, ecc (MAG, ovvero Megadati Amministrativo – Gestionali).
- Relative al contenuto: autore, formato, ecc.

DUBLIN CORE: esempio molto comune di schema di metadati.
[Si chiama così, non perché è legato alla capitale irlandese, ma perché è stato progettato nella città di Dublin, in Ohio, USA, su iniziativa di un gruppo di archivisti.]
Dublin Core è uno schema di metadati che serve per annotare risorse testuali e multimediali. Con l'avvento del web, negli anni '90, gli utenti hanno cominciato a riversare materiali di formati e media diversi: dunque, presto bibliotecari e archivisti hanno espresso l'esigenza di avere un formato generale che fosse *facile da* realizzare e adatto per descrivere risorse di diverso tipo. Dublin Core è, quindi, "nato per il web". Nella sua forma primaria, è lo schema di metadati più semplice (a partire dal quale si possono costruire schemi di metadati più complessi e sofisticati). Di base, quindi, il "core" ("nucleo") di elementi descrittivi ("elements" → 15 in Dublin Core) è intuitivo e si adatta a

qualsiasi tipo di risorsa. E' uno standard, ISO 15836 (ISO = International Standard Organization, che a fronte di un'ingente documentazione che motiva la richiesta di una certa tecnologia di diventare standard, si occupa di vagliare la documentazione e decidere se può diventare uno standard) e stabilisce un insieme di 15 elementi "core" per la descrizione di risorse in ambiti diversi; questi termini sono parte di un insieme di vocaboli specifici per descrivere i metadati delle risorse. Lo scopo esplicito di Dublin Core è permettere il reperimento di risorse e sfruttarle per scopi culturali, educativi, formativi e commerciali: "le organizzazioni, anche piccole, possiedono informazioni che valgono milioni, ma non riescono a farle arrivare alle persone giuste".

Dublin Core descrive il concetto di "risorsa" con la locuzione: "anything that has identity" (definizione "filosofica"). Aldilà di questa definizione, Dublin Core definisce come "risorsa" una qualsiasi entità, fisica o virtuale, che possiede un URI (Uniform Resource Identifier), ovvero un Indirizzo Internet, in cui è collocato il documento digitale o l'identità nel web di qualcosa che esiste nel mondo reale → una risorsa è tipicamente un'informazione, o un servizio, ma "risorsa" è un termine molto generale / generico.
DCMI (Dublin Core Metadata Initiative), ovvero l'organizzazione generale che sviluppa Dublin Core e tutti gli standard collegati, riconosce un insieme di tipologie di risorse ("typelist"): Collezione, Dataset (es. set di dati relativi all'affluenza in un certo museo in un determinato periodo di tempo), Evento, Immagine (ferma o in movimento), una Risorsa Interattiva, un Servizio o Software, Suono, Testo od Oggetto Fisico.

INSIEME DI ELEMENTI DESCRITTIVI ("elements") CHE COMPONGONO I METADATI (DC – MES)

- Title;
- Creator / Author;
- Subject (argomento trattato);
- Description;
- Publisher;
- Contributor (autori secondari);
- Date;
- Type (es. suono, immagine, testo, etc.);
- Format;
- Identifier (identificativo, che permette di identificare in maniera univoca un determinata risorsa: es. indirizzo web);
- Source;
- Language;
- Relation (per descrivere le relazioni che la risorsa può avere con altre);
- Coverage (coordinate spazio – temporali, il dove e il quando, un'opera è stata creata);
- Rights.

Ogni punto è un elemento descrittivo a sé; tutto l'insieme degli elementi di base ("core") di Dublin Core è conosciuto con la sigla DC – MES (Metadata Elements Set).
In una descrizione possono essere presenti anche solamente due elementi, poiché Dublin Core non è prescrittivo, ma fornisce i 15 elementi utilizzabili per descrivere una risorsa, ma vengono utilizzati

solo quelli necessari e, a volte, si possono ripetere. Inoltre, oltre ai descrittori di base, si possono utilizzare delle loro versioni più specifiche, seguendo uno schema qualificato.

EUROPEANA = sistema di collezioni digitali per eccellenza. Esso nasce come portale europeo dei beni culturali, che raccoglie le schede, in un formato unificato basato su Dublin Core, di tantissime opere custodite in musei, collezioni private e archivi (fisici) delle varie nazioni europee, rendendole accessibili attraverso un'unica interfaccia. All'epoca della sua creazione, Europeana era una sorta di infinita collezione di schede, ma si sta evolvendo sempre più verso una struttura basata sul concetto di "collezione": sulla base costituita dalle varie schede, i curatori hanno creato una serie di collezioni tematiche.

SCHEMA DI METADATI: SERIALIZZAZIONE → lo schema di metadati è una nozione astratta, ma il suo contenuto logico (le sue schede di catalogo: "records") deve essere espresso (serializzato) in un formato "machine – readable", in formati tipicamente XML, RDF / XML, JSON.
SERIALIZZAZIONE: prendere quello che è uno schema di metadati, che è astratto, e tradurlo in un formato testuale machine – readable.

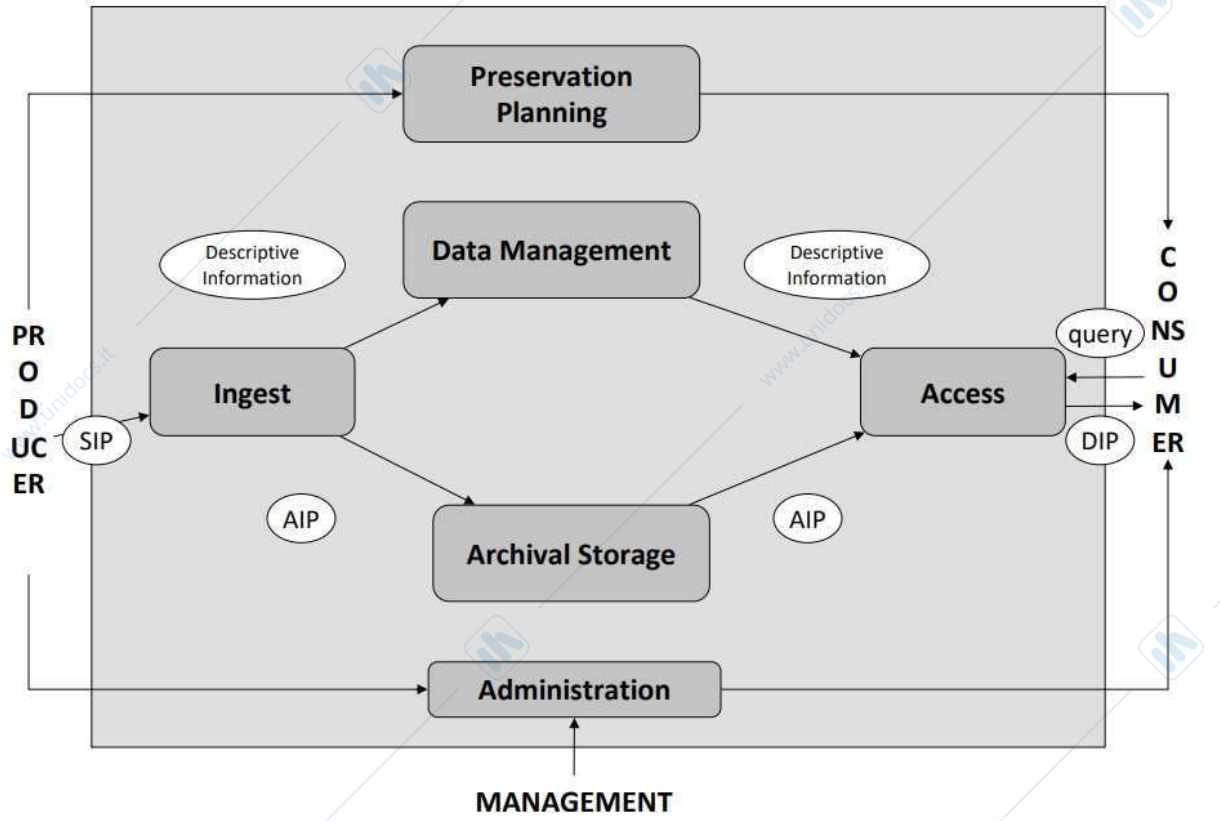
IL MODELLO OAIS (Open Archive Information System)
Il modello OAIS è uno standard ISO dal 2003 (ISO 14721), progettato nel 1999 da un gruppo internazionale (USA, UK e Francia).
Si tratta di un modello astratto e funzionale* di un archivio, nel senso che descrive quelle che sono le componenti essenziali di un archivio e il modo in cui interagiscono (sempre in maniera astratta, cioè senza scendere nei dettagli implementativi, dicendo per esempio il modo in cui una componente deve essere realizzata).
E' uno schema di riferimento che serve per stabilire una terminologia comune per parlare degli archivi. Quindi, non viene realizzato direttamente, ma serve come riferimento per progettare gli archivi.
E' un modello fortemente orientato alla **conservazione e fruizione di dati**.
Il modello identifica una serie di ruoli primari:
1) data producer → produttore dei dati;
2) data management → gestione dei dati;
3) data consumer → fruitore dei dati.
SCHEMA GENERALE: i dati immessi nell'archivio dal produttore vengono resi accessibili al fruitore attraverso l'intervento del gestore dei dati.

*OAIS è funzionale, nel senso che l'architettura dell'archivio in questo modello viene descritta in termini di **funzioni**, che costituiscono il funzionamento dell'archivio e si interfacciano secondo uno schema predefinito (modello funzionale).

COMUNITA' DESIGNATA
Il fruitore viene indicato come "comunità designata alla fruizione" (es. –comunità di studiosi, -pubblico generico, -gruppo di appassionati di un certo dominio di conoscenza / ambito culturale, -comunità locale).

Il modo in cui vengono descritti i dati nell'archivio, e quindi vengono resi fruibili dalla comunità di riferimento, sono concetti collegati → il modo in cui i dati sono descritti è collegato al modo in cui saranno disponibili per la comunità di riferimento.
Lo schema descrittivo non è importante solo per conoscere la risorsa descritta, per identificarla e per poterla studiare, ma i megadati con cui una certa risorsa viene descritta sono importanti per poterla poi reperire all'interno dell'archivio.

FUNZIONI DI BASE / ENTITA' FUNZIONALI NEL MODELLO OAIS
Quattro funzioni di base:
1) Ingest: si riferisce all'immissione di dati all'interno dell'archivio → ha il compito di ricevere i dati dal produttore e predisporli per la memorizzazione;
2) Archival Storage e 3) Data Management: il primo garantisce la persistenza / memorizzazione e l'accessibilità delle risorse nell'archivio, mentre il secondo gestisce l'indicizzazione e la descrizione delle risorse, quindi le schede dei vari documenti caricati nell'archivio, rappresentati all'interno di una base di dati (funzione che si occupa dell'informazione descrittiva).
4) Access: funzione di accesso, ovvero quell'insieme di elementi di interfaccia (maschere di ricerca, collezioni in evidenza, etc.) → funzione che supporta l'accesso ai dati da parte dell'utente.
[Nel momento in cui l'archivio è molto ricco, con molte collezioni e documenti, è molto importante avere una maschera di ricerca, mentre se ci sono pochi elementi potrebbero bastare un insieme di collezioni già preparate, magari presenti nella pagina d'accesso, che contengono in maniera esaustiva tutti quelli che sono gli elementi contenuti nell'archivio.



INFORMATION PACKAGE = "pacchetto informativo" → siccome l'informazione è composta da un dato e dal metadato (quindi, la sua descrizione), è un oggetto complesso.

Ne esiste uno specifico per la "Submission" (SIP → Submissione Information Package), quindi per l'immissione iniziale nell'archivio, e c'è n'è uno specifico per la funzione di "Archival" (AIP → Archival Information Package) e poi ne esiste uno specifico, che può essere diverso da quello iniziale e di quello interno (poiché alcuni elementi vengono o nascosti o tradotti in maniera più leggibile per l'utente finale), che è quello per la "Dissemination" (DIP → Dissemination Information Package). L'elenco delle funzioni dell'archivio non è completo fino a quando non vengono incluse le funzioni di Preservation Planning e Administration, funzioni trasversali rispetto alle quattro funzioni chiave collegate tra di loro, implementate per fronteggiare le problematiche della conservazione di dati (aggiornamenti di formati, dei plug – in, dei server, etc.) → queste due funzioni sono responsabili di operazioni di mantenimento e di amministrazione eseguite affinché l'archivio possa restare al passo con la tecnologia, e garantire la conservazione a lungo termine delle risorse presenti al suo interno.
In ogni passaggio si ha a che vedere con un pacchetto informativo.
Lo standard identifica una serie di pacchetti, o buste virtuali, in cui il dato stesso è costituito dall'informazione descrittiva → il pacchetto informativo completa l'informazione descrittiva del pacchetto stesso, che ne permette l'identificazione e il reperimento dall'interno dell'archivio.

PERCORSO / FLUSSO DI IMMISSIONE DATI
Il Data Producer ha il compito di preparare i dati per l'immissione ed è responsabile della loro qualità, completezza e compatibilità con l'archivio. I dati provenienti dal produttore vengono verificati dalla funzione Ingest.
Una volta inseriti nell'archivio, i dati veri e propri vengono scorporati dalla descrizione e vengono (I) inviati alla funzione di memorizzazione (Storage) e (II) la descrizione viene incorporata alla funzione di gestione dei dati (Data Management).
[E' un modello astratto perché è sia un modello che è stato pensato anche per l'archivio fisico, sia perché le stesse funzioni posso essere realizzate con delle soluzioni più o meno complesse.]
PERCORSO DI RECUPERO DEI DATI
Percorso inverso, che riguarda la fruizione dei dati.
La funzione di accesso (Access) si fa carico delle richieste di accesso (previa verifica dei diritti di accesso) → riceve una query (la parola chiave, i campi completati in una maschera) da parte del fruitore dei dati, verifica che quei dati siano effettivamente disponibili per quella tipologia di utente, l'interrogazione viene raffrontata coi metadati memorizzati nella funzione di Data Management (la funzione di gestione dei dati usa le informazioni descrittive per comporre l'interrogazione), dati e metadati vengono estratti dalla funzione di memorizzazione e i dati vengono inoltrati al fruitore.
ALTRE FUNZIONI
_ PRESERVATION PLANNING → preposta alla definizione delle politiche di conservazione dell'archivio (interventi periodici e straordinari di manutenzione e di aggiornamento software e hardware).
_ ADMINISTRATION (funzione di amministrazione) è preposta alla gestione quotidiana dell'archivio (refresh periodici dell'archivio memorizzato su un supporto digitale).

OPEN ARCHIVES INITIATIVE (OAI)
L'Open Archives Initiative sviluppa e promuove standard / elementi architetturali per il dialogo / scambio (interoperabilità) tra archivi diversi, mirando a facilitare la disseminazione efficiente dei contenuti (quindi la possibilità di far conoscere a tutte le persone interessate le risorse "richieste").

Questa iniziativa è nata nel contesto dei repository (ovvero "archivi un po' meno strutturato"), che ha le sue radici nei movimenti open access, collegati agli archivi istituzionali.
Nel modello degli open archives, si distinguono due concetti:
_ DATA PROVIDERS: che forniscono (espongono) i propri dati attraverso uno specifico protocollo, il PMH (Protocol for Metadata Harvesting → raccolta di dati), permettendo anche a terze parti di accedere a tali dati, più precisamente i metadati.
_ SERVICE PROVIDERS: utilizzano i metadati raccolti da terze parti per fornire ulteriori servizi → usano i metadati raccolti attraverso il processo di harvesting come base per fornire servizi ulteriori.

HARVESTER: sistema che reperisce / raccoglie i dati da un insieme di Data Provider, e li rappresenta in maniera unificata → l'harvester è un'applicazione client che effettua una richiesta OAI – PMH.
L'harvester viene utilizzato dal service provider come mezzo per raccogliere metadati da una repository.

Il protocollo PMH e il modello dei Open Archives contiene un insieme di termini tecnici:
_ RESOURCE = risorsa;
_ ITEM = "oggetto", è un singolo costituente di un repository;
_ RECORD = metadati in un formato specifico (normalmente restituito in formato XML).

Nel protocollo di Metadata Harvesting, l'harvester ha un insieme di comandi a disposizione, tra cui GetRecord, che recupera un singolo record di metadati (una scheda) da un repository → in alcuni casi, può essere effettuato solo tramite autorizzazioni particolari / password, mentre in altri è completamente aperto in alcune istituzioni.
& → è il separatore che separa le singole informazioni che accompagnano la richiesta.

WEB SERVICES → servizi web, un paradigma del funzionamento del web di oggi, che permettono, utilizzando la rete internet, un'applicazione client (un sito web, un app su smartphone, etc.) di riunire le informazioni da più soggetti diversi / sorgenti diverse (usando internet).

DATA MANAGEMENT = *DATABASE*
ARCHIVAL STORAGE = *REPOSITORY SOFTWARE*

SISTEMI PER LA GESTIONE DI BASI DI DATI (DBMS → **Database** Management System, anche chiamato Database Server)
Nel caso dell'architettura software, si ha un Database Server con delle funzioni che servono per gestire le basi di dati.
Nell'ambito delle basi dati / database, si hanno tre distinzioni:
_ dati → stringhe, numeri, date, che descrivono una specifica entità, con delle caratteristiche chiamate proprietà;
_ metadati → schemi generali che accomunano tutte le entità (nel database, i nomi delle entità e delle proprietà e il tipo dei valori delle proprietà sono rappresentate entrambe, ma separatamente).
I metadati delle risorse nel mondo degli archivi, nel database sono i dati veri e propri, che a loro volta devono essere descritti; nel database, i dati veri e propri sono separati dalla loro descrizione.

RELAZIONI TRA I DATI NEL DATABASE

All'interno del database, i dati sono strutturati secondo delle relazioni logiche tra di loro → l'insieme di dati, con le loro relazioni e le "etichette" che li descrivono, forniscono una rappresentazione autodescrittiva (che, quindi, contiene tutti gli elementi che permettono di decifrarne il significato).

Uno dei punti fondamentali del database è che i dati non sono duplicati: utenti diversi possono condividere gli stessi dati e tutti agiscono sulla stessa rappresentazione (cioè non viene generata una scheda differente a seconda di chi ne fa richiesta, ma esiste un'unica funzione dei dati sui database, a cui tutti possono accedere) → es. quando si utilizza un bancomat, si sta interrogando lo stesso database della banca.

I dati sono persistenti, cioè sono memorizzati su supporto permanente (il supporto di memorizzazione è indipendente dal programma che si usa per l'accesso).

IL DBMS

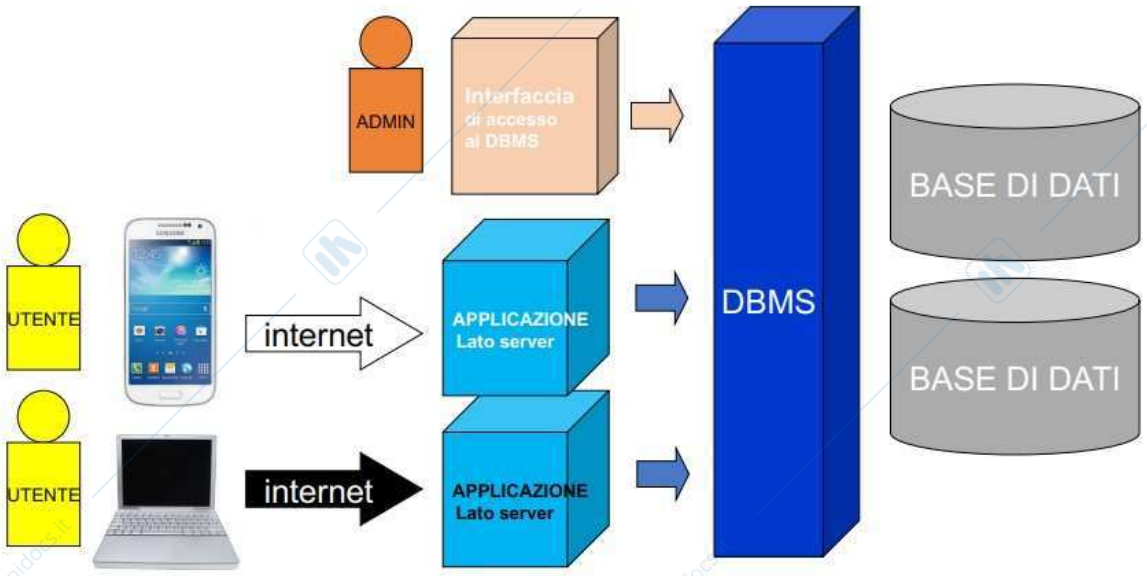
Il DBMS è il sistema per la gestione di basi di dati (DataBase Management System → definito anche DataBase Server) e si interpone tra la base di dati e gli utenti. Ne esistono di vari tipi di largo utilizzo (MySql, Oracle PostgreSQL, Access), tutti simili tra di loro, cioè tutti supportano versioni diverse dello stesso linguaggio, ovvero SQL.

Il database può essere gestito da un'interfaccia (grafica o riga di comando) oppure da un linguaggio di programmazione (es. PHP): gli utenti possono effettuare le operazioni sul database tramite l'interfaccia utente oppure operare su di esso tramite un programma che effettua le operazioni per loro.

PROPRIETA' DEL DBMS

Il DataBase Server ha una serie di compiti specifici:

- supporto ad accesso contemporaneo da parte di più utenti → DBMS ha il compito di supportare l'accesso allo stesso database da parte di più utenti contemporaneamente (gestire la **concorrenza**, cioè permettere ad ognuno di eseguire le proprie operazioni in maniera sicura, senza avere interferenze tra i vari utenti);
- i dati sono rappresentati in un formato indipendente dalle applicazioni (**formato relazionale** → **riutilizzo** tra le singole applicazioni) → si interagisce col database attraverso il linguaggio SQL, e l'interfaccia d'accesso è disponibile da diversi linguaggi di programmazione (le applicazioni possono essere scritte in Java, PHP, etc.), ma la 'creazione' / 'conservazione' / 'fruizione' dei dati rimane indipendente dalle singole applicazioni;
- aspetto di sicurezza - riservatezza dei dati → fare in modo che, anche se due o più utenti, per effettuare delle operazioni, si connettono allo stesso database, nessuno vede i dati degli altri e nessuno può 'toccarli';
- nessuno può fare delle operazioni che violino l'integrità dei dati: i dati persistono, quindi non scompaiono tra un'operazione e l'altra, e le operazioni che un utente può fare dipendono dalla tipologia di utente a cui appartiene;
- permettere l'ottimizzazione dell'accesso (**tempi di risposta** prevedibili)→ es. indicizzare le tabelle, in modo da dare precedenza a determinate operazioni che vengono fatte più frequentemente.



Tra utente finale (in giallo) e il database (in grigio) si interpongono diversi livelli di programmazione:
_ le app e l'accesso online si basano sulla programmazione lato client (html, javascript, etc.)
_ i comandi dati dagli utenti lato client viaggiano nel web e vengono tradotti lato server in comandi per il DMS tramite le applicazioni lato server (PHP, Node js, etc.).

TRANSAZIONI
Una transazione è una sequenza di operazioni effettuate da un'applicazione su una base di dati (es. scrivere o modificare un dato e poi leggerlo).
Nel momento in cui si ha un errore nella transazione, le operazioni che la compongono possono fallire.
La transazione forma un'unità nella logica dell'elaborazione dei dati.
ESEMPIO: se l'operazione di trasferimento di denaro da un conto ad un altro comporta più operazioni (come modificare il conto di partenza e quello di arrivo), essa non sarà compiuta fino a quando tutte le singole operazioni, nella sequenza prevista, non sono state effettuate.
STATI DI UNA TRANSAZIONE:

1. inizia;
2. legge e / o scrive dei dati all'interno del database;
3. finisce;
4. fase di conferma → nel momento in cui l'operazione è andata a buon fine, quindi tutti i passaggi sono stati eseguiti, c'è il cosiddetto **commit**; se, invece, qualcosa è andato storto, si ha un **abort** (l'operazione non viene eseguita e nessun passaggio sarà eseguito).

ES. nel prelievo con bancomat, il denaro non è erogato se tutte le operazioni sul database non sono state effettuate con successo → se una delle operazioni fallisce, le altre saranno annullate e non ci sarà addebito perché l'intero insieme è unitario.

Ogni transazione deve essere "**ACID**(A)":

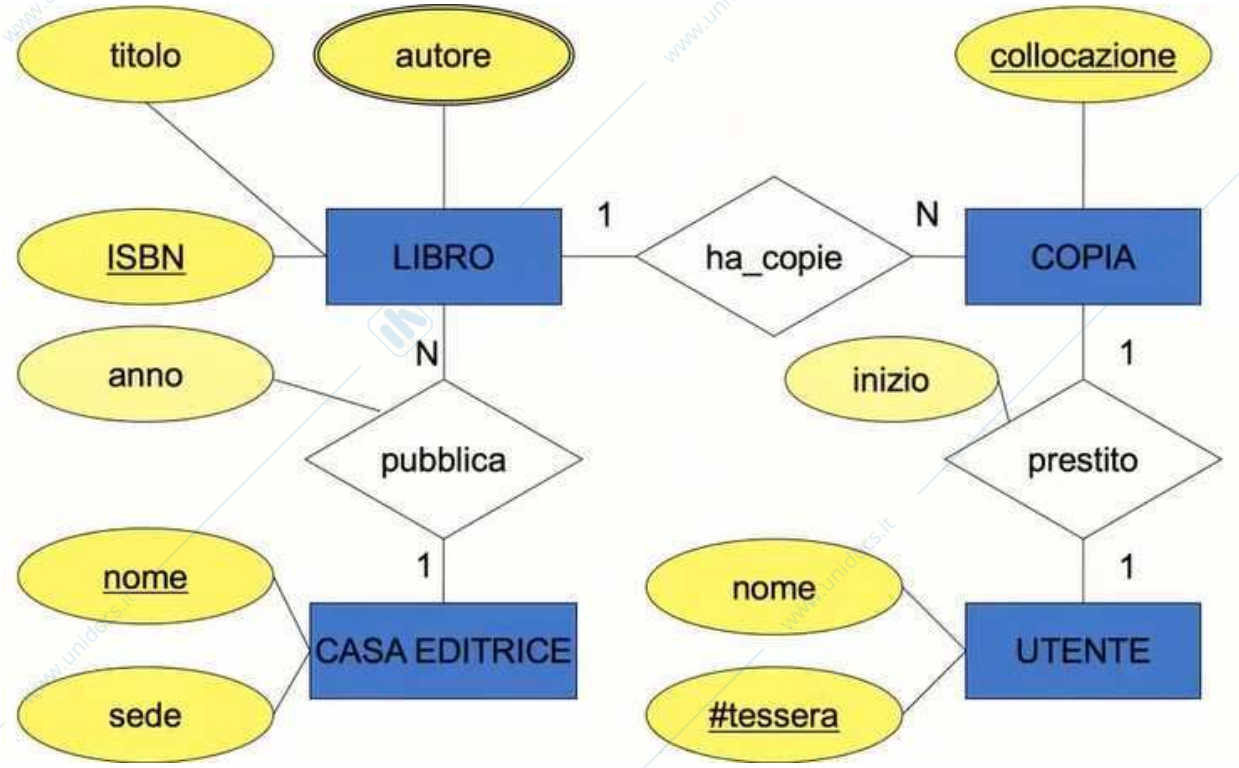
- **A**TOMICITA' = se viene eseguita, la transazione viene eseguita per intero, altrimenti non viene eseguita del tutto;

- **C**ONSISTENZA = non può violare i vincoli di integrità del database → es. non è possibile che un prelievo da un conto modifichi l'associazione tra il numero di conto e la persona a cui appartiene;
- **I**SOLAMENTO = non interferisce con le altre transazioni → es. se avvengono operazioni successive di prelievo e deposito di denaro, ogni operazione inizierà dove finisce la precedente.
- **D**URABILITA' = le modifiche che effettua non vanno perse → es. l'estratto conto si modifica con ogni operazione riuscita in modo permanente.

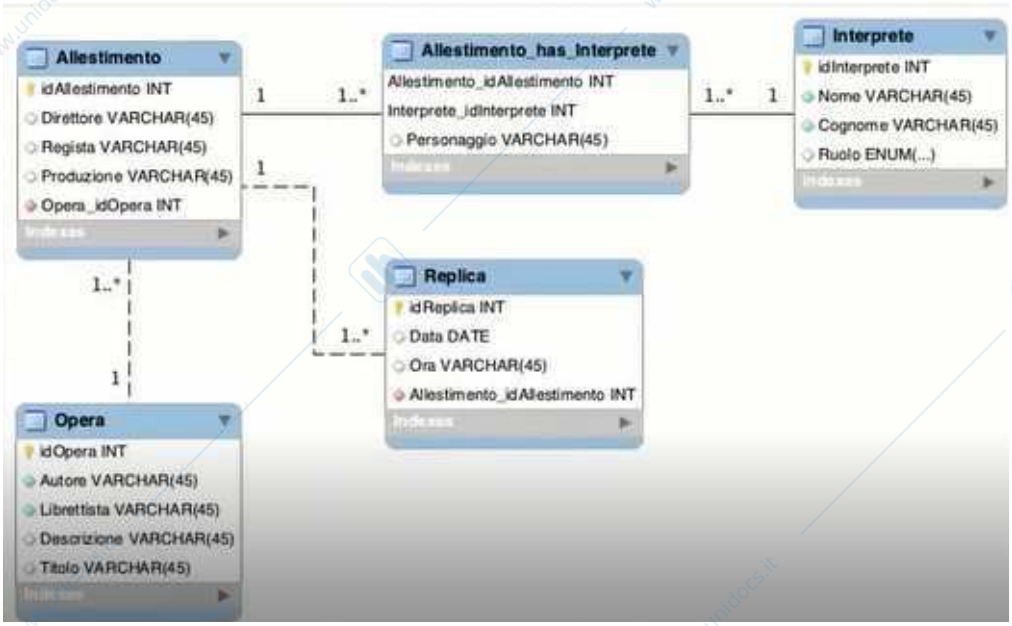
PROGETTAZIONE DI UNA BASE DI DATI

La progettazione di un database si articola in diverse fasi, e ad ogni fase della progettazione corrisponde un modello di rappresentazione dei dati:

1) progettazione **concettuale**: si definisce il **modello concettuale** del database, e lo si fa attraverso un apposito linguaggio grafico → si rappresenta il contenuto del database secondo quello che viene chiamato **diagramma entità – relazioni** (ER);



2) progettazione **logica**: il diagramma grafico si deve trasformare nello **schema relazionale**, che rappresenta le tabelle del database e le relazioni tra di loro;



3) progettazione **fisica**: il database va implementato.

LINGUAGGI PER BASI DI DATI

Quando si va ad operare su un database, nella fase in cui va implementato, si utilizza il **linguaggio SQL**. Questo linguaggio, con tutte le sue varianti proprietarie per gestire il database, è diviso in diversi insiemi di comandi, a seconda del tipo di operazioni si vogliono effettuare:

1. Linguaggio per la **definizione** dei dati (Data Definition Language) = permette di creare la struttura in cui i dati saranno inseriti;
2. linguaggio per la **manipolazione** dei dati (Data Manipulation Language) = permette di inserire dati, modificare dati, interrogare dati, etc.;
3. linguaggio di **memorizzazione** dei dati (Data Storage Language);

MODELLO CONCETTUALE MODELLO ENTITA' – RELAZIONI (ENTITY RELATIONSHIP MODEL → ER)

Modello / diagramma Entity – Relationship, "Entità – Relazioni" (ER).
E' la fase che si basa sull'individuazione, nel dominio dato, di **entità** (concrete o astratte). Le entità sono caratterizzate da un insieme di **proprietà** (attributi), e le entità sono collegate da **relazioni** (associazioni).

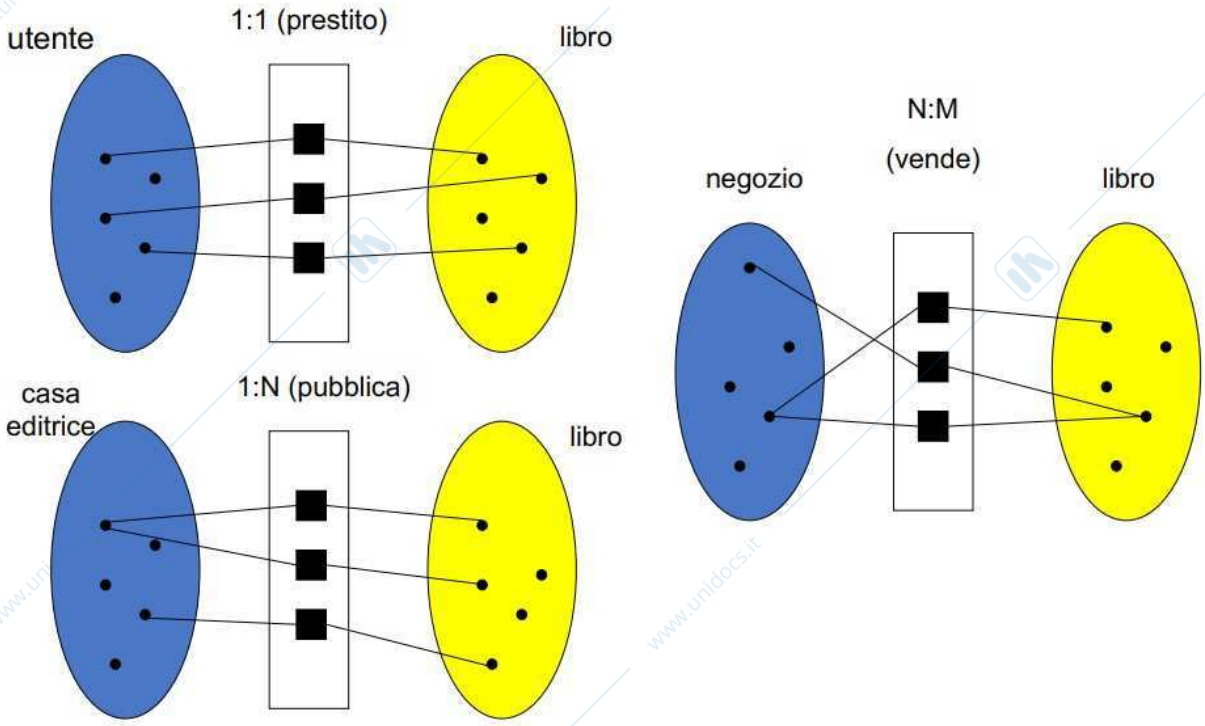
Un tipo di *entità* è caratterizzato da un insieme di attributi: quindi, un insieme di singole entità che condividono certe caratteristiche → tutte hanno gli stessi attributi (o meglio, proprietà), quindi i metadati, ma con valori diversi.
ES. il tipo di entità "libro" è caratterizzato da attributi come "autore", "titolo", "editore", etc.

ATTRIBUTI

Gli attributi hanno un nome ed un valore → il valore fa parte di un insieme, denominato **dominio** dell'attributo.
ES. nome dell'attributo: anno (di pubblicazione)
dominio dell'attributo: insieme delle date (anni).
Esistono attributi di tipo diverso:
_ attributi elementari (es. "titolo", "nome", etc., composti quindi da un'unica stringa di caratteri)
_ attributi composti (es. "indirizzo", etc.)
_ attributi a valore singolo / multivalore (es. "titolo" che può essere composto da titolo e sottotitolo").
ATTRIBUTO CHIAVE → un attributo il cui valore è unico per tutte le singole entità di un certo tipo (in modo che non ci siano problemi di riferimento dal mondo rappresentazionale al mondo reale).
ES. non ci sono due persone con lo stesso codice fiscale.
Esistono casi in cui la "chiave" non è data da un singolo attributo, ma da un insieme di attributi (ES. titolo e anno per un libro; per una persona, nome, luogo e data di nascita).
Quindi, l'attributo chiave rappresenta quell'insieme di valori che identificano in maniera univoca un'entità.

Esistono anche attributi obbligatori: alcuni attributi devono necessariamente avere un valore, per altri non è necessario.
Vincolo di integrità → l'attributo chiave deve **per forza** avere un valore.

COME SI RELAZIONANO TRA DI LORO I TIPI DI IDENTITA' ALL'INTERNO DI UN DOMINIO?
Si ha una relazione (o a*ssociazione*) quando si ha un attributo di un certo tipo di entità che si riferisce al valore di un attributo di un altro tipo di entità.
Le relazioni sono caratterizzate dalla loro cardinalità = per *cardinalità* di una relazione o associazione (tra due entità) si intende il numero massimo di singole associazioni a cui può partecipare un'entità (nella progettazione del diagramma ER, si deve specificare per ogni associazione la sua cardinalità). Esistono tre tipologie di cardinalità:
1) **1:1** = es. un libro è prestato a un solo utente per volta;
2) **1:N** = es. una casa editrice pubblica molti libri;
3) **N:M** = es. diverse copie dello stesso libro si trovano in diverse librerie.
La cardinalità delle relazioni riveste un'importanza cardinale per la formazione della base di dati.



Il grado di una relazione è il numero di tipi di entità coinvolti nella relazione.

RUOLI E PROPRIETA' DELLE RELAZIONI

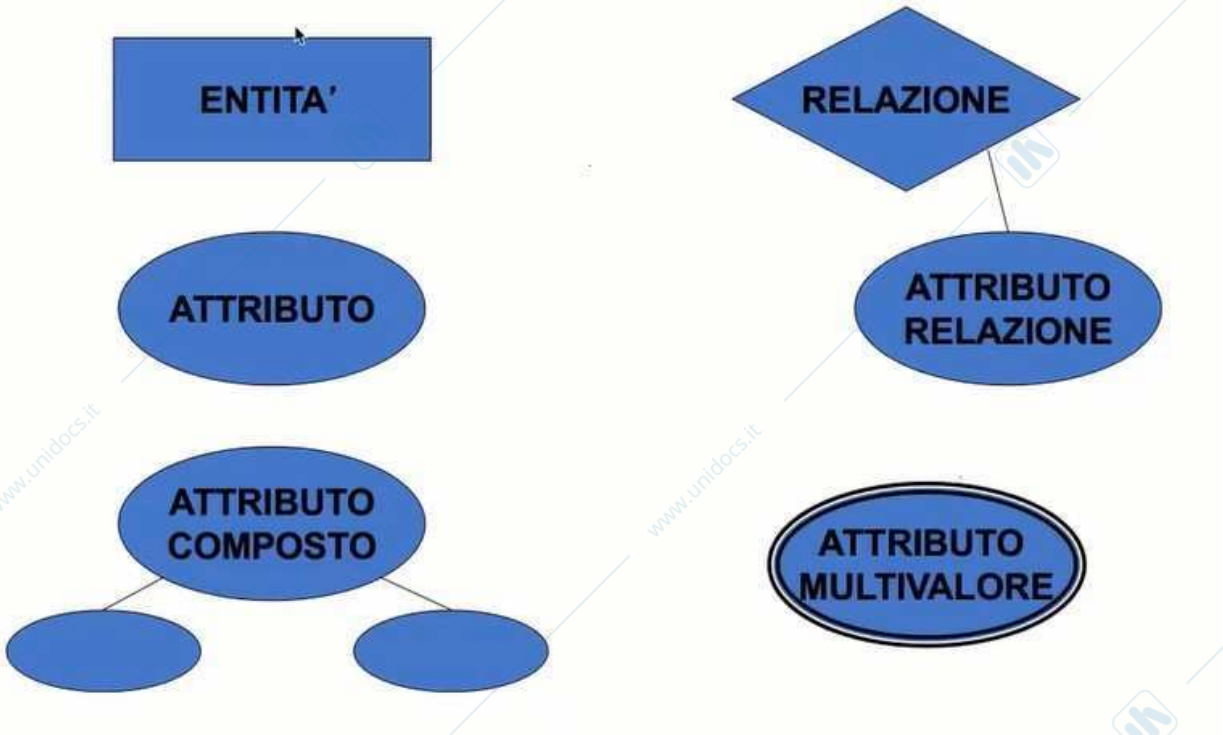
Spesso si usa un particolare termine per riferirsi al ruolo che ha un certo tipo di entità all'interno di relazione o associazione.

Una relazione può essere caratterizzata da proprietà, rappresentate attributi della relazione (ES. l'edizione di un libro può avere un curatore o un anno di edizione).

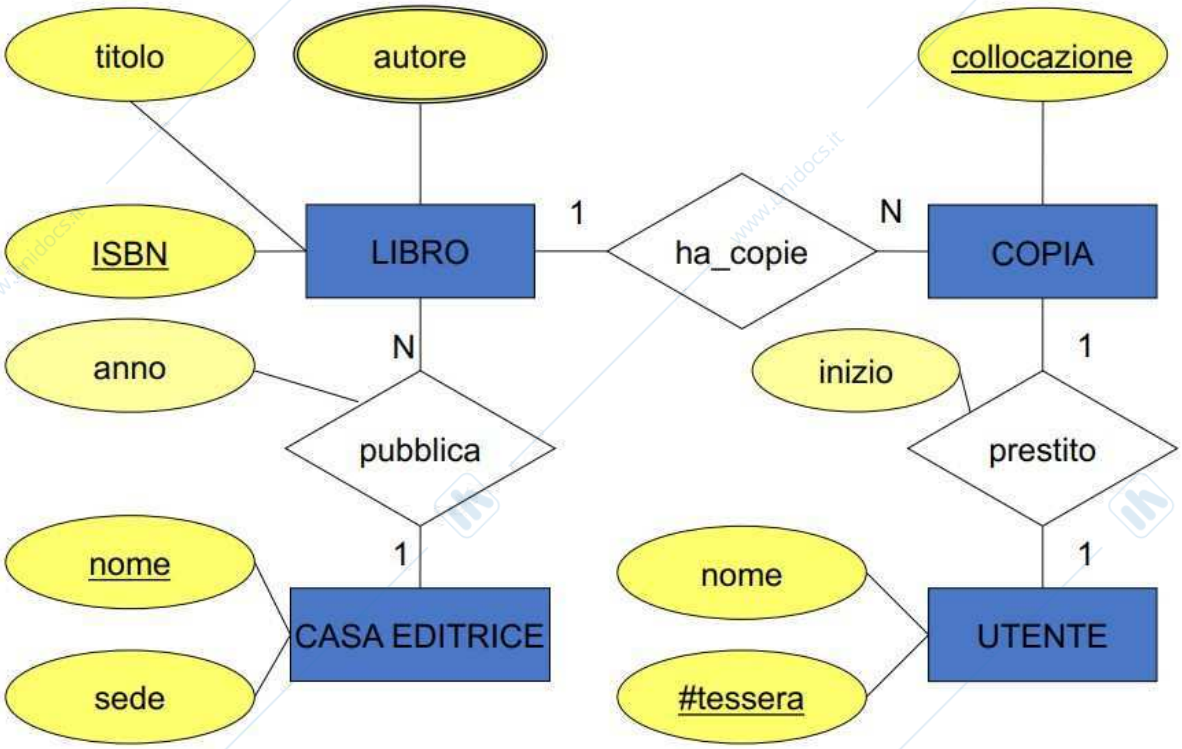
VINCOLI DI PARTECIPAZIONE

Con il vincolo di partecipazione totale, tutte le entità di un certo tipo devono partecipare a una certa relazione. Questo vincolo è anche definito dipendenza di esistenza: un'entità non può esistere se non è in relazione con un'altra entità.

Rappresentazione grafica del modello ER



Esempio di notazione ER



MODELLO RELAZIONALE

Il modello relazionale si interpone tra il modello concettuale e l'effettiva realizzazione e implementazione della base di dati.

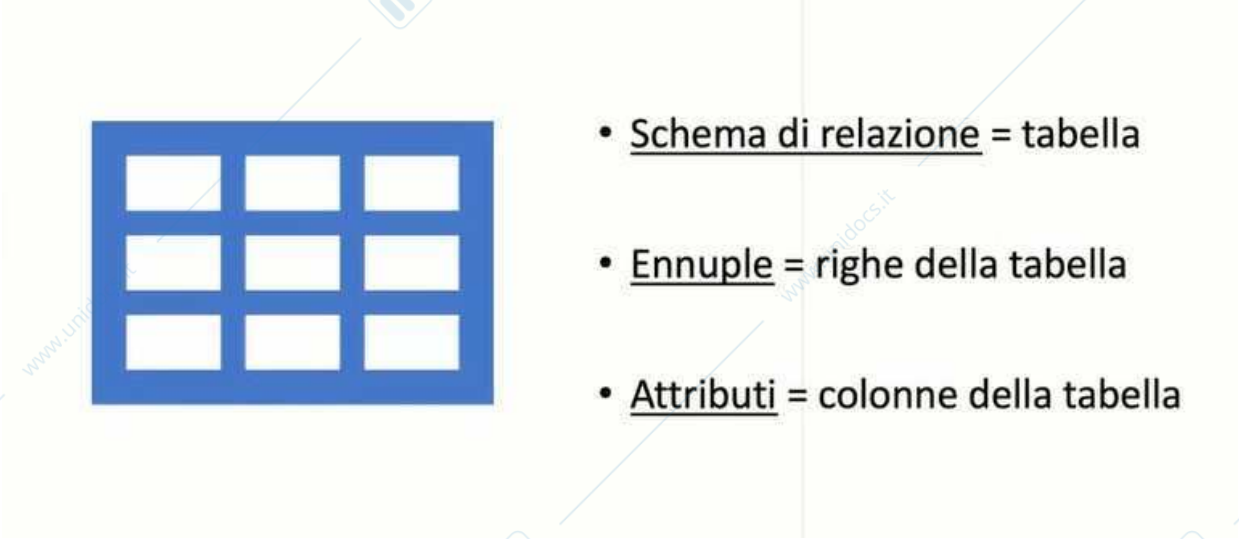
E' un modello formale che descrive la base di dati in maniera non ambigua (quindi in maniera chiara e senza ambiguità), che si tradurre immediatamente in una rappresentazione software (ovvero, gli strumenti utilizzati per la rappresentazione permettono di realizzare in maniera automatica il lavoro → nel momento in cui si è prodotto il modello relazionale, quindi lo schema della base di dati, lo strumento di progettazione utilizzato ha una funzione che permette di creare in maniera automatica la base di dati, quindi senza il bisogno di operazioni manuali).

Si chiama "modello relazionale" perché descrive ogni entità come una relazione tra un insieme di dati → si distinguono:

_ uno schema di relazione, ovvero i metadati che descrivono il tipo di identità, che è dato da un nome e da un insieme di attributi, il cui valore appartiene ad un certo dominio (= tipo di entità a livello ER). Essendo uno schema, non ha al suo interno i dati.

_ una relazione è un insieme di valori (detta anche **ennupla**, cioè un insieme di N valori), che descrive una singola entità secondo le caratteristiche espresse dallo schema → le singole ennuple, quindi le singole relazioni, se vengono prese separatamente senza lo schema che le descrive, potrebbero risultare abbastanza incomprensibili: per capire il significato delle singole righe della tabella (le singole ennuple / le singole relazioni) bisogna fare ricorso allo schema di relazione che le descrive, che attribuisce un significato a quelli che sono i dati veri e propri contenuti nelle singole righe.

DIFFERENZA TRA SCHEMA E STATO: uno schema può avere tanti stati → un insieme di ennuple che possiede le caratteristiche imposte dallo schema è uno stato di una relazione.



CHIAVI

Per capire come collegare tra di loro le tabelle, quindi i singoli schemi di relazioni, bisogna ricorrere al concetto di chiave. Le chiavi sono un insieme di attributi per il quale due ennuple non possono avere la stessa configurazione di valori.

Nei domini reali, quando si va ad analizzare il dominio per decidere quale sarà la chiave primaria, quindi quella che deve avere un valore diverso per ognuna delle righe della tabella (per ogni singola entità), è possibile che ci siano diverse combinazioni di attributi che possono costituire una chiave.

Per quanto riguardano le chiavi candidate, possono esisterne di più, ma tra queste se ne sceglie una come chiave primaria: quella chiave che ha valori unici può diventare una chiave primaria.
La chiave primaria è il primo ingrediente per collegare tra di loro le tabelle.

CHIAVE ESTERNA (FOREIGN KEY, FK)
Una chiave esterna di uno schema di relazione è un attributo che si riferisce alla chiave (primaria) di un'altra relazione.
VINCOLI DI INTEGRITA' (CHE DIPENDONO DALLA RELAZIONE)

- vincolo di integrità di dominio: vincoli che derivano dal dominio (non ci può essere una riga di tabella in cui manca un determinato elemento);
- vincolo di integrità dell'entità: una ennupla non può avere valore nullo per la chiave;
- vincolo di integrità referenziale: una ennupla che è referenziata come chiave esterna da un'altra ennupla non può essere cancellata.

Il contenuto di una base di dati che rispetta i vincoli di integrità è uno stato possibile dello schema della base di dati nel momento in cui:
_ **inserendo** le ennuple nelle tabelle occorre rispettare i vincoli di integrità;
_ lo stesso quando si **cancellano** o **modificano** le ennuple già inserite.

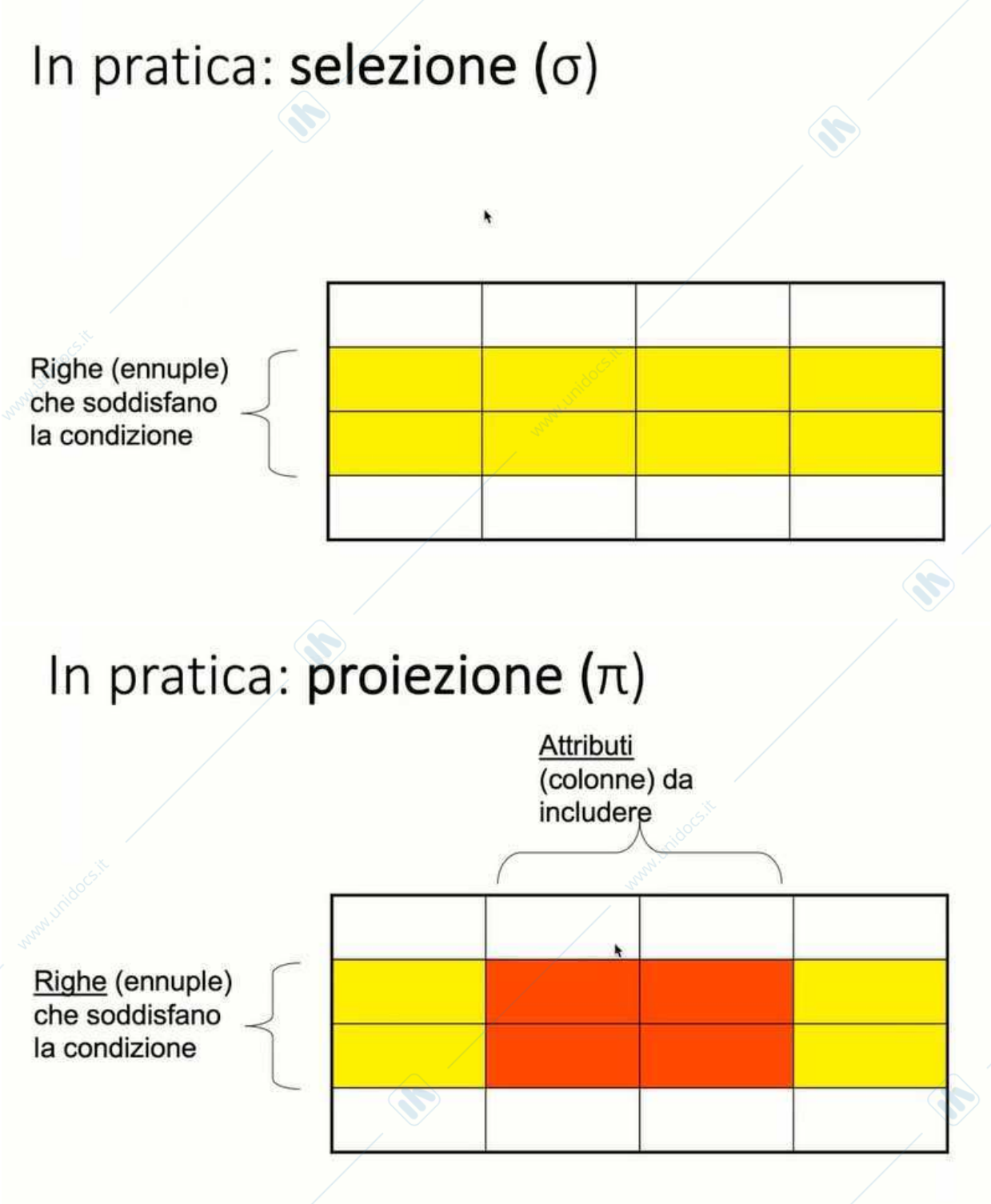
TRADUZIONE DAL MODELLO ER ALLO SCHEMA RELAZIONALE
La traduzione dal modello ER allo schema delle relazioni si basa su un procedimento formale che segue un insieme di regole, in parte incorporate in programmi, come MySQLWorkbench, per la progettazione di basi dati.
I principi sono:

- le entità diventano tabelle;
- le relazioni vengono tradotte attraverso le chiavi esterne;
- gli attributi delle relazioni diventano attributi della tabella (seguono la chiave esterna).

TRADUZIONE DELLA CARDINALITA' DELLE RELAZIONI
A seconda delle cardinalità delle relazioni, la traduzione cambia.
_ **1:1** → in una (a caso) delle relazioni coinvolte si inserisce come chiave esterna **la chiave primaria dell'altra**;
_ **1:N** → si inserisce come chiave esterna nella relazione del lato N **la chiave primaria della relazione lato 1**;
_ **N:M** → si crea una nuova relazione che contenga come chiavi esterne **le chiavi primarie di N e M**.

INTERROGARE IL DATABASE (linguaggio SQL)
Il linguaggio SQL si fonda su quello che è un calcolo formale delle tabelle e del loro contenuto chiamato algebra relazionale → è risultato delle interrogazioni SQL che servono ad estrarre i dati dal database. Il calcolo ci permette di prevedere in anticipo il risultato delle interrogazioni.
ALGEBRA RELAZIONALE
Il modello relazionale si fonda sull'algebra relazionale, che permette di definire formalmente le proprietà delle basi di dati relazionali → permette di fare una sorta di calcolo di quello che è il risultato di una certa interrogazione (**query**) condotta in linguaggio SQL.
QUALI SONO LE OPERAZIONI FONDAMENTALI DELL'ALGEBRA RELAZIONALE?
Sono operazioni che hanno come oggetto la singola tabella:

- selezione: estrarre un sottoinsieme delle ennuple / delle righe della tabella (simbolo: σ → sigma)
- proiezione: estrarre dalla tabella un insieme di colonne (simbolo: π)



Si possono fare anche operazioni insiemistiche, ovvero su più tabelle (si possono vedere le tabelle come un insieme di righe, che hanno una chiave che permette di ordinarle). Esse sono rilevanti, perché i dati sono "sparpagliati" nelle varie tabelle contemporaneamente, e l'algebra relazionale ci mette a disposizione, per lavorare su più tabelle determinate operazioni:

_ unione (di due insiemi, prendendo in considerazione tutti gli elementi che li compongono);
_ intersezione (di due insiemi, prendendo in considerazione solo gli elementi che li accomunano);
_ differenza (sottraendo un insieme da un altro);
_ prodotto cartesiano.

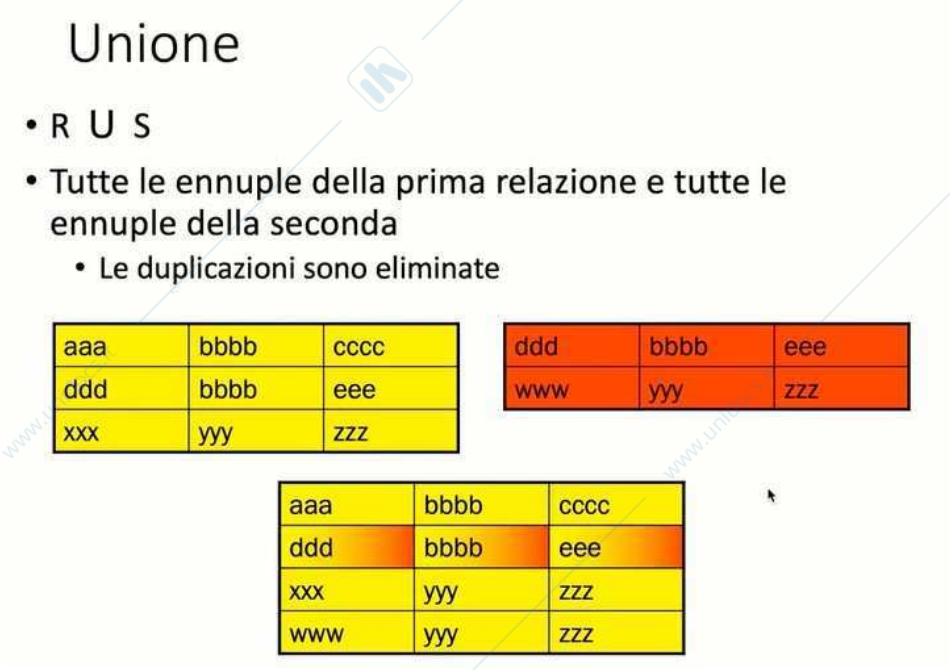
Alla fine di ogni interrogazione si crea una terza tabella, detta tabella dei risultati

Si dice che due tabelle sono compatibili all'unione se possiedono lo stesso schema di base, che quindi devono avere due ennuple dello stesso tipo → queste tabelle sono apparentemente diverse, ma hanno lo stesso grado, cioè contengono un numero uguale di colonne (attributi), e tali attributi possono essere fatti coincidere, a coppie, perché hanno lo stesso dominio, quindi hanno lo stesso tipo di dato associato (ovvero il numero di colonne della tabella).

Dunque, "compatibilità d'unione" significa che si hanno due tabelle che sono perfettamente confrontabili (cambiano le "etichette", come il nome della tabelle e i nomi assegnati agli attributi, quindi alle colonne, ma i dati sono gli stessi).

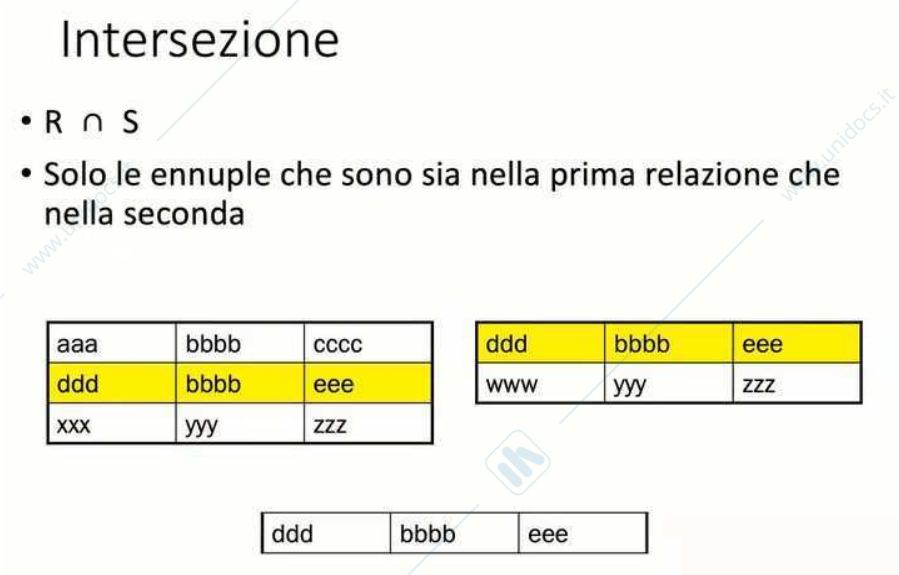
UNIONE

L'unione si rappresenta con il simbolo "U", che nel linguaggio dell'insiemistica rappresenta appunto l'unione, e se sussiste la compatibilità / relazione, consiste nel prendere tutte le righe o ennuple della prima relazione e tutte le righe della seconda (le duplicazioni sono eliminate).



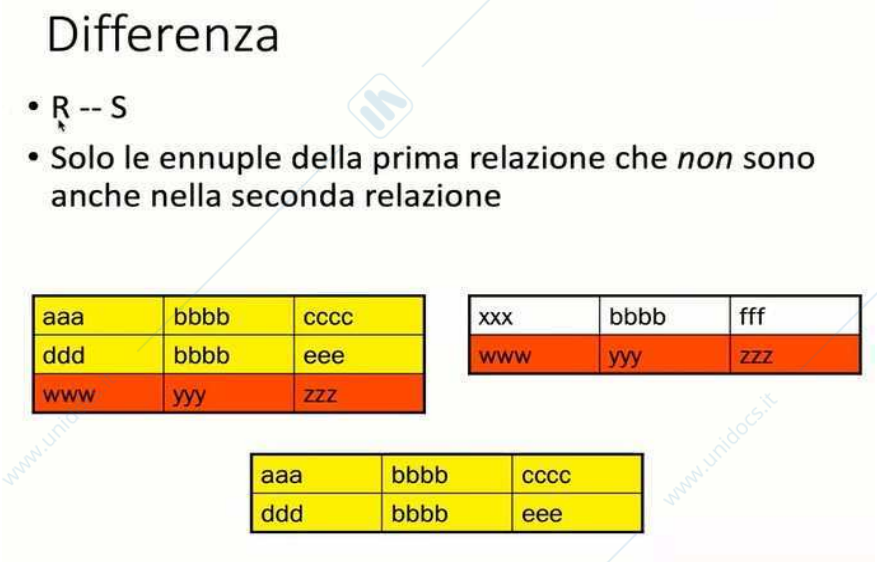
INTERSEZIONE

Per quanto riguarda l'intersezione, si vanno a scegliere solo le righe che sono in comune, ovvero quelle che fanno parte della prima relazione o tabella e della seconda (simbolo ∩)



DIFFERENZA

Con la differenza, vengono incluse nel risultato solo le ennuple della prima relazione che non sono anche nella seconda → dalle righe della tabella si vanno a sottrarre quelle che eventualmente sono anche nella seconda tabella (simbolo --).

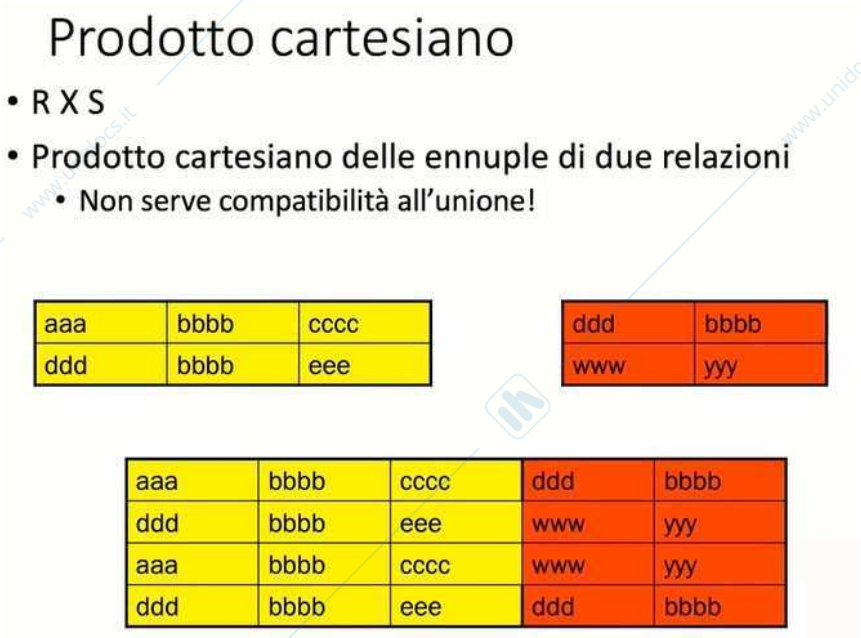


PRODOTTO CARTESIANO

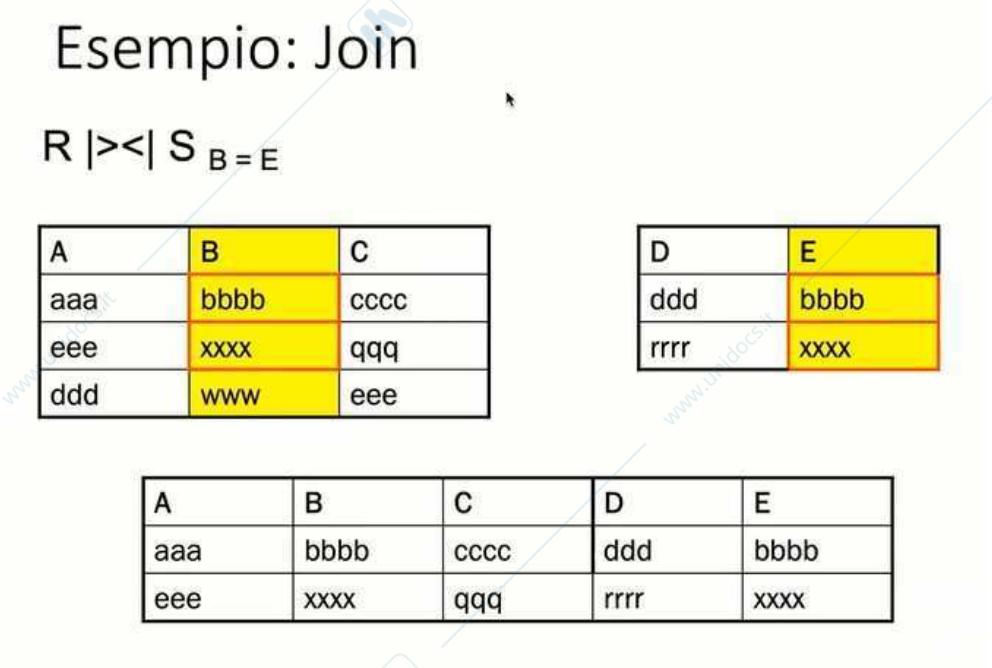
Nel linguaggio SQL, il prodotto cartesiano serve per allineare più tabelle tra di loro.

Il prodotto cartesiano ha senso in pochissime situazioni: per questo tipo di operazione non serve la compatibilità d'unione (anzi, non è praticamente pertinente) → il prodotto cartesiano si fa normalmente per tabelle intrinsecamente diverse tra di loro.

Il prodotto cartesiano va a creare una tabella che sarà composta di tutte le coppie che si possono formare combinando le righe della prima tabella e quelle della seconda tabella (simbolo X).



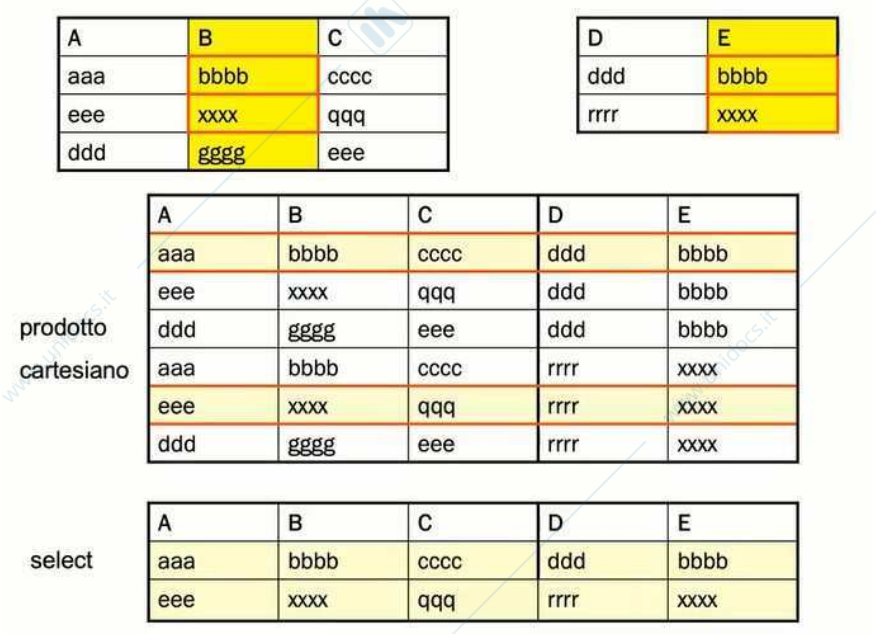
Quasi sempre, il prodotto cartesiano è il primo passo di un'operazione molto utilizzata nelle basi dati relazionali, ovvero il **join** → creare il join (simbolo |><| "tipo farfalla") tra due tabelle equivale a creare tutte le coppie delle righe della prima e della seconda tabella per cui vale una certa relazione, tra un attributo (colonna) della prima e un attributo (colonna della seconda), e deve per forza esistere la relazione.



Quindi, si combinano tutte le righe della prima e della seconda tabella, tenendo solamente quelle per cui vale la relazione di chiave esterna.
Esistono diversi tipi di join:

- equijoin: una relazione che si fonda sull'**uguaglianza** tra le colonne / attributi (lo stesso valore si ritrova sia nella prima che nella seconda tabella);
- join naturale: quando si va ad eliminare la duplicazione, ovvero togliendo una delle due colonne per non creare ridondanza.

L'operazione di join può essere vista come una sequenza di due operazioni:
1) il prodotto cartesiano → in cui si ottiene una tabella "molto più grande" dove ci sono tante righe quante sono le coppie che si possono formare con le ennuple della prima tabella e quella della seconda;
2) la selezione delle ennuple sulla tabella risultante dal prodotto cartesiano per cui vale la condizione di join.



A	B	C
aaa	bbbb	cccc
eee	xxxx	qqq
ddd	gggg	eee

D	E
ddd	bbbb
rrrr	xxxx

prodotto cartesiano

A	B	C	D	E
aaa	bbbb	cccc	ddd	bbbb
eee	xxxx	qqq	ddd	bbbb
ddd	gggg	eee	ddd	bbbb
aaa	bbbb	cccc	rrrr	xxxx
eee	xxxx	qqq	rrrr	xxxx
ddd	gggg	eee	rrrr	xxxx

select

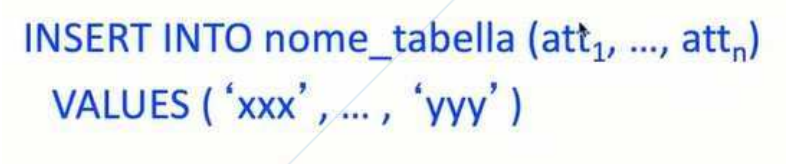
A	B	C	D	E
aaa	bbbb	cccc	ddd	bbbb
eee	xxxx	qqq	rrrr	xxxx

SQL – COME INTERROGARE UNA BASE DI DATI
Il linguaggio SQL contiene delle tipologie di operazioni diverse (Data Manipulation Language):
1) operazioni specifiche per definire le tabelle
- *create table* = creazione di una nuova tabella;
- *drop table* = cancellazione di una tabella;
- *alter table* = modifica una tabella (per es. andando ad inserire una nuova riga).

2) sulle ennuple, oltre ad interrogare i dati, ci sono altre operazioni importanti come
- *insert* = inserisce una nuova riga di valori in corrispondenza degli attributi definiti nello schema;
- *delete* = cancella una o più righe;
- *update* = modifica dei dati.

SQL permette di porre vincoli di chiave esterna sullo schema di una base di dati.

INSERIMENTO DEI DATI
Per inserire le righe (ennuple) in una tabella (relazione) si usa il comando INSERT



Attenzione nell'inserimento: non è detto che il DBSM faccia rispettare i vincoli di integrità.

Esempio

```
INSERT INTO `libri` ( `isbn` , `titolo` , `autore` ,
  `pubblicato_da` , `anno` )
  VALUES ('ISBN 88-7192-22', 'Sistemi di basi di dati',
  'R.E. Elmasri, S. B. Navathe', 'Pearson', '1999 ');
```

MODIFICARE DATI

Per modificare una riga (ennupla) di una tabella (relazione) si usa il comando UPDATE.

```
UPDATE nome_tabella
      SET attr1 = xxx, ..., attn = yyy
      WHERE condizione

UPDATE 'libri' SET 'autore' = 'R.E. Elmasri e S. B.
  Navathe' WHERE 'isbn' = 'ISBN 88-7192-22'
```

OPERAZIONI DI SELEZIONE

Qui entra in gioco l'algebra relazionale.

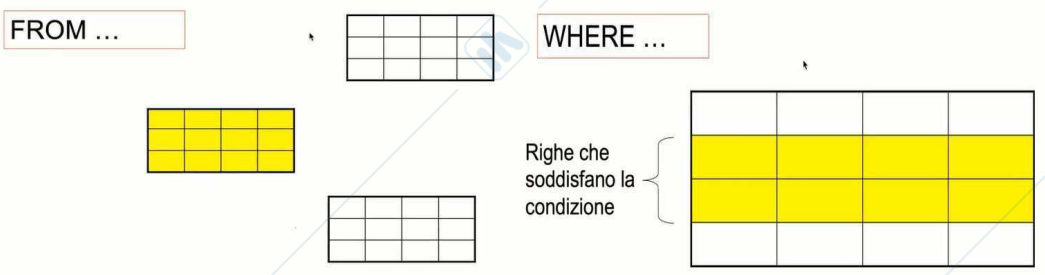
La query SQL è composta da tre clausole:

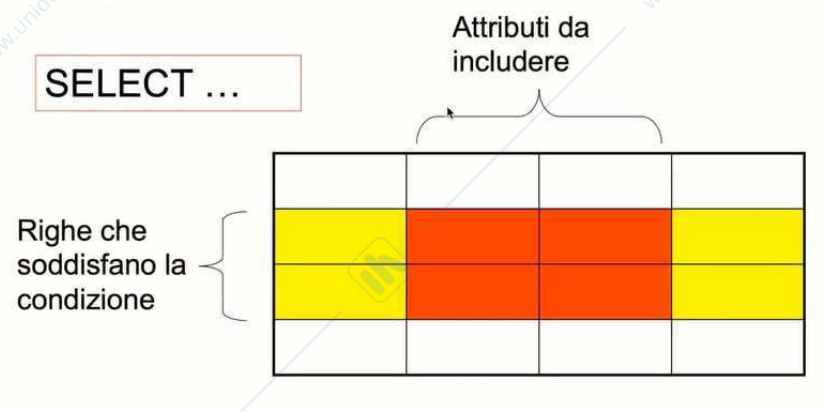
1. *SELECT* = indica quali sono gli attributi da includere;
2. *FROM* = indica la tabella da cui prendere gli attributi;
3. *WHERE* = la condizione da rispettare.

1) SELECT e algebra relazionale

Il comando SELECT effettua una selezione delle righe → (rappresenta in realtà l'ultimo passaggio)

- di una tabella (FROM)
- in base a una condizione (WHERE)
- proietta il risultato in base agli attributi indicati in SELECT.





La selezione si effettua attraverso gli operatori di confronto:

- = → utile per lavorare sia coi numeri che con le stringhe;
- > → il confronto avviene con un numero e non con una stringa;

CERCARE LE STRINGHE

Per cercare tutte le righe in cui un certo campo contiene una certa stringa si usa `nome_campo LIKE “%stringa%”`

Il % indica una qualsiasi sequenza di caratteri.

```
SELECT *
  FROM `libri`
  WHERE autore LIKE "%Coe%"
```

L’operatore AND serve per cercare le righe che soddisfano due o più condizioni contemporaneamente, come anche OR e NOT.

E’ possibile recuperare dati da più tabelle contemporaneamente basandosi sulle relazioni tra di esse (operazione JOIN)

```
SELECT Autore, Titolo, Nome_Editore
  FROM Libri, Case_Editrici
  WHERE Editore = Nome_Editore
```

Nell’esempio, si confrontano le due tabelle sulla base della relazione tra la colonna `Editore` di `Libri` e `Nome_Editore` di `Case_Editrici`.

TIPI DI JOIN

- INNER JOIN estrae solo le righe che hanno un valore per la condizione data;
- LEFT / RIGHT JOIN stabilisce quale tabella governa il JOIN relativamente alla presenza di valori.

ORDER BY

La clausola ORDER BY, aggiunta a una SELECT, permette di orinare il risultato in base a un attributo.

```
SELECT Autore, Titolo, Data
  FROM Libri
  WHERE Autore = "..."
  ORDER BY Data
```

- Ordina il risultato in base alla data

ELEMENTI DI PROGETTAZIONE EER

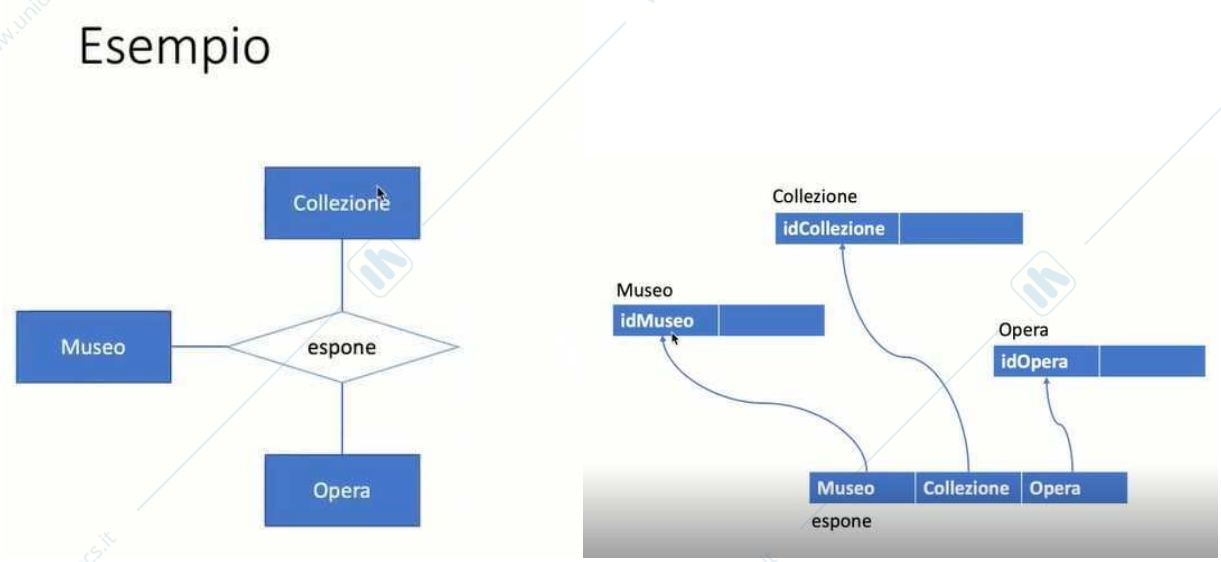
Nella traduzione del diagramma ER in uno schema relazionale, gli attributi composti vengono scomposti nei loro componenti semplici e rappresentati come attributi semplici (es. l'attributo indirizzo, composto da via, numero civico e CAP, sarà rappresentato dai tre attributi semplici via, numero_civico, CAP, diventando singole colonne del database).

Per quanto riguarda gli attributi multivalore (es. l'attributo autore può essere più di uno), si crea una nuova relazione (quindi una tabella), che avrà come chiave esterna la chiave primaria della relazione di partenza, che conterrà tante righe quanti sono gli attributi possibili.

ASSOCIAZIONI TERNARIE

Le associazioni ternarie coinvolgono più entità, e si possono rappresentare in vari modi (es. la relazione tra museo, opera e collezione).

In generale, si crea una nuova relazione (tabella) per rappresentare l'associazione che contenga tante chiavi esterne quante sono le entità collegate



ALTRO MODO IN CUI SI PUO' RAPPRESENTARE

TIPI DI ENTITA' GENERALI E SPECIFICI

In alcuni casi, durante la modellazione, ci si rende conto che alcuni tipi di entità sono casi particolari di un solo tipo di entità.

E' possibile creare una sola tabella, che conterrà

- gli attributi (colonne) comuni a tutti i tipi specifici;
- gli attributi specifici di ogni singolo tipo (che potranno avere valori nulli).

In alternativa, si possono creare più tabelle:

- una tabella per rappresentare il tipo di entità generale, che conterrà solo gli attributi in comune;
- tabelle specifiche che contengano solo gli attributi propri di ogni tipo specifico;
- la chiave primaria è condivisa tra le tabelle.

RELAZIONE UNARIA / RICORSIVA → che non coinvolge due entità, ma coinvolge una relazione con sé stessa.

PROCESSO DI NORMALIZZAZIONE

La normalizzazione è quella procedura standard utilizzata per verificare le basi di dati dopo che sono state progettate. Si basa su una sequenza di passi, a ciascuno dei quali corrisponde una forma normale. Esistono cinque forme normali, ma solo le prime tre sono di uso comune.

1) PRIMA FORMA NORMALE: è necessario che i tipi di entità rappresentati all'interno dello schema abbiano solo attributi con valori atomici → si eliminano gli attributi multi – valore e gli attributi composti: nel primo caso, essi vengono rappresentati creando una tabella separata con una relazione referenziale (chiave esterna verso la tabella di partenza), mentre nel secondo caso, gli attributi composti vengono scomposti in attributi semplici.

2-3) SECONDA E TERZA FORMA NORMALE: si basano sulla nozione di dipendenza funzionale → si dice che la colonna *x* dipende funzionalmente alla colonna *y* quando i valori di *x* dipendono dai valori di *y*.

_ Seconda forma normale → perché una tabella sia nella seconda forma normale, tutti gli attributi che non fanno parte della chiave primaria dipendono (funzionalmente) dalla chiave primaria.

- Tabella Studenti: se mettiamo in una sola tabella Corso di laurea e Dipartimento avremmo Corso di Laurea che dipende da Dipartimento e non dall'id utente (la chiave primaria)
- Meglio creare nuova tabella!

Esempio

Utente

Tessera	Nome	Recapito	Corso di laurea	Dipartimento
0	Rossi	…	dams	Studium
1	Bianchi	…	lettere	Studium
2	Verdi	…	chimica	Scienze

Utente

Tessera	Nome	Recapito	Corso di laurea
0	Rossi	…	dams
1	Bianchi	…	lettere
2	Verdi	…	chimica

Corsi

Corso di laurea	Dipartimento
dams	Studium
chimica	Scienze

_ Terza forma normale → una base di dati è in terza forma normale se in ogni tabella non esistono dipendenze tra le colonne che non sono basate sulla chiave primaria.

- Esempio: immaginiamo di inserire nella tabella libri una colonna "sede": tale colonna dipenderebbe dalla chiave di un'altra tabella, casa editrice

Terza forma normale: esempio

Libri

Titolo	Autore	isbn	editore	luogo
Pinco	Pallino	234	Tizio	Roma
…	…	…	Caio	Bari
…	…	…	Tizio	Roma

Editore

Nome_ed	…
Tizio	…
Caio	…

Libri

Titolo	Autore	isbn	editore	~~luogo~~
Pinco	Pallino	234	Tizio	~~Roma~~
…	…	…	Caio	~~Bari~~
…	…	…	Tizio	~~Roma~~

Editore

Nome_ed	luogo
Tizio	Roma
Caio	Bari

XML

XML (Extensible Markup Language) è uno degli standard del web, emanato dal Web Consortium (W3C).

La terminologia di XML si è evoluta nell'arco di quasi vent'anni, e consiste anche in una famiglia di tecnologie che fanno riferimento a XML.
XML deriva da un precedente linguaggio standard, chiamato SGML (ISO 8879), creato a partire dagli anni '50/'60 per rappresentare documenti in formato elettronico, in ambito di archivi soprattutto aziendali, e si tratta dell'acronimo della dicitura estesa Standard Generalized Mark-up Language.
XML è un meta – linguaggio, un linguaggio di mark – up: ci sono dei tag, che contendono testi o altri tag → la particolarità di XML è che non ha tag propri ma, in quanto meta – linguaggio, è una tecnologia che permette di definire nuovi linguaggi, che saranno comunque linguaggi di mark – up (come HTML o SVG).
XML viene descritto come un formato semplice, molto flessibile e derivato da SGML, che aveva la particolarità di essere un sistema di rappresentazione in forma elettronica dei testi piuttosto complessa, con una manualistica molto estesa che richiedeva uno studio intenso per impadronirsene. SGML fu inventato per creare uno standard internazionale per la definizione di metodi "device – independent" e "system - independent" per rappresentare testi in forma elettronica.
Originalmente progettato per andare incontro alle sfide della pubblicazione elettronica su larga scala, ora XML sta giocando un ruolo sempre più importante nello scambio di varietà di dati sul web e altrove (es. uno dei principali linguaggi basati su XML è SVG, un formato di grafica, che quindi si allontana molto dall'idea originale di testo digitale).
Di recente, nello scambio di dati tra le varie applicazioni nel web, XML ha lasciato il passo ad uno standard più leggero chiamato JSON.
W3C descrive il linguaggio XML come un formato basato su testo (quindi, non esistono caratteri che non siano stampabili e visualizzabili) per rappresentare un'informazione strutturata (documenti, dataset, etc.); è stato per molto tempo un formato privilegiato per scambiare informazioni strutturate tra programmi, tra persone e tra persone e programmi, poiché la sua particolarità è quella di essere un formato facilmente leggibile.
DIFFERENZE CON HTML
XML è più "rigoroso", cioè gli strumenti deputati per gestire e processare documenti XML sono meno tolleranti del web browser, quindi:

- tutti gli elementi devono essere chiusi o marcati con i vuoti;
- i valori degli attributi devono essere sempre racchiusi tra le virgolette doppie (`es. happiness type = "joy"`/)
- non esistono elementi predefiniti (infatti, XML non possiede tag propri);
- in HTML esistono dei " built – in character entities" (es. &eacute, per la è), mentre in XML esistono solo cinque "built – in character entities" → &lt, &gt, &amp, &quot, &apos.

Inoltre, rispetto ad HTML, che si preoccupa del layout e della formattazione (quindi è fatto per marcare la struttura ipertestuale, come identificare tutti gli elementi logici del documento, come tabelle, liste, etc), XML si riferisce più precisamente al contenuto e la sua strutturazione.
SCOPI DEL DESIGN DI XML
XML è stato progettato per rendere facile la marcatura di documenti in formato digitale, con in mente alcune priorità:

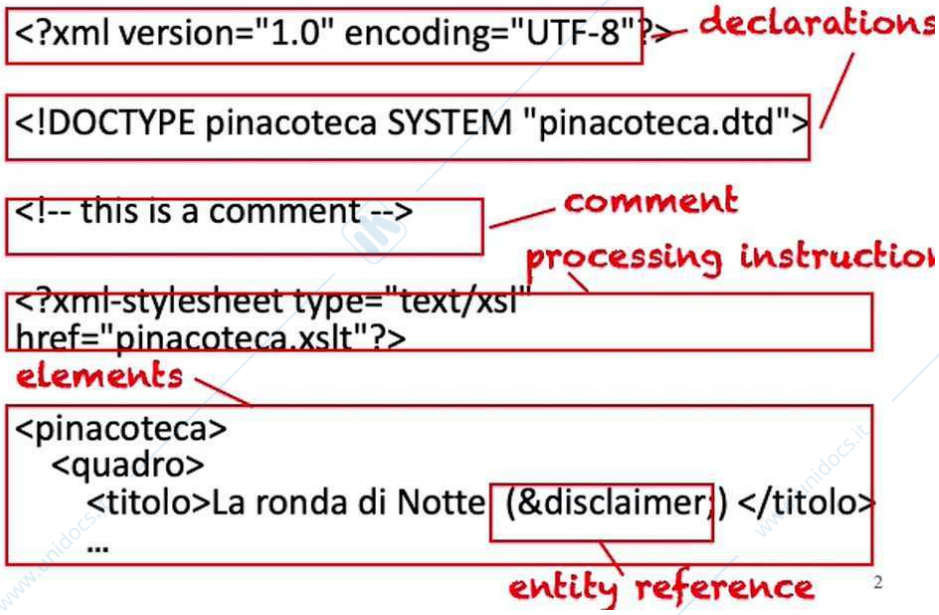
- XML deve essere utilizzabile direttamente su Internet;
- XML deve supportare un'ampia varietà di applicazioni (per questo non ha tag propri, ma essi possono essere inventati di volta in volta a seconda della necessità del momento);

- XML deve essere compatibile con SGML;
- XML deve essere facile scrivere programmi che processano documenti XML;
- XML deve avere un numero di caratteristiche opzionali ridotto al minimo, per non rendere complesso il suo utilizzo / processamento informatico;
- XML deve essere "human - legible" e chiaro;

Quando viene caricato un documento XML da solo all'interno di un browser non si produce nessun risultato, ma necessita di una forma di trattamento automatico tramite un linguaggio di programmazione, oppure tramite un foglio di stile particolare, chiamato XSLT → il trattamento automatico con XSLT è stato creato appositamente per XML, e serve a manipolare la struttura del documento.

Un tipo di applicazione che ha conferito una certa fortuna al linguaggio XML è stato il suo utilizzo nel servizi web, ovvero quei servizi attraverso cui un'applicazione che "gira" su un server può essere interrogata da varie applicazioni client (i client mandano richieste, la risposta viene fornita in formato machine – readable, tipicamente XML o JSON).

ANATOMIA DI UN DOCUMENTO XML



Ogni elemento ha un nome, ovvero il nome del tag, chiamato anche identificatore generico, e può avere una serie di attributi (gli attributi sono facoltativi).

`<element-name…> … </element-name…>`

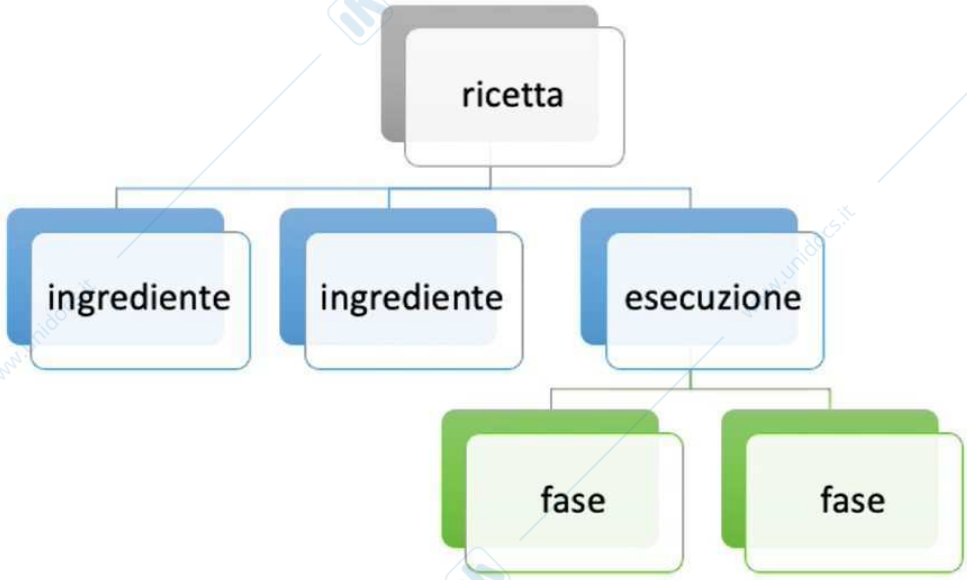
Per quanto riguarda gli attributi, ognuno ha nome insieme ad un valore.

`<element-name attribute-name=value…> … </element-name attribute-name=value>`

Esempio

```
<ricetta nome="Ciambella marmorizzata" perquanti="8" tempo="1 ora">
        <ingrediente>500 grammi farina</ingrediente>
        <ingrediente>3 uova</ingrediente>
        <ingrediente>250 grammi zucchero</ingrediente>
        <ingrediente>250 grammi burro</ingrediente>
        <ingrediente>100 grammi latte</ingrediente>
        <ingrediente>1 bustina lievito vanigliato</ingrediente>
        <ingrediente>50 grammi cacao in polvere</ingrediente>
        <esecuzione>
            <fase>Montare il burro con lo zucchero.</fase>
            <fase>Aggiungere le uova una per una, continuando a montare.</fase>
            <fase>Aggiungere la farina setacciata insieme al lievito e il latte.</fase>
            <fase>Mettere a cucchiaiate i due terzi dell'impasto in uno stampo a ciambella imburrato
                    e infarinato.</fase>
            <fase>Aggiungere il cacao in polvere all'impasto restante.</fase>
            <fase>Versare l'impasto al cacao nello stampo, mescolandolo grossolanamente all'impasto
bianco.</fase>
            <fase>Cuocere in forno a 180 gradi per 60 minuti circa.</fase>
        </esecuzione>
</ricetta>
```

Rappresentazione gerarchica del documento



ENTITA'

Esiste una serie di caratteri (character entities) che non si possono mettere all'interno di un documento XML così come sono, cioè sono caratteri che compaiono come parte del markup, e per essere utilizzate all'interno dei documenti, essi devono essere scritti in modo diversi:

- < = <
- & = &
- > = >

- " = “
- ' = ‘

Per quanto riguarda la leggibilità del documento XML, essa è dovuta dalla sua struttura che contiene sia il markup che i dati stessi → il markup, che funge da “intestazione”, è costituito dai tag insieme ai loro attributi, costituendo una struttura gerarchica (gli elementi XML sono “annidati” gli uni dentro gli altri); i dati rappresentano il testo e il riferimento alle cinque entità predefinite e, inoltre, i dati sono anche contenuti all’interno degli attributi (sezioni CDATA, ovvero “Character Data”).

PARSER XML

Il parser è in grado di capire la grammatica del linguaggio definito dall’utente.

Il parser XML svolge due operazioni:

1) valuta se il documento è “well-formed” (“ben formato”), cioè che segue la sintassi di XML;

2) valuta se il documento è valido rispetto al DTD, ovvero lo specifico linguaggio utilizzato.

(I) *REGOLE DEI DOCUMENTI BEN FORMATI*

- Tutti i tag sono chiusi
- I tag sono annidati senza incrociarsi
- C’ è solo un elemento radice
- Tutti i valori sono tra virgolette
- Non ci sono attributi con stesso nome per stesso elemento
- Non ci sono commenti nei tag
- Non ci sono < o & nei dati o nei valori

Se ci sono degli errori di sintassi nell’XML (es. tag non chiusi, tag incrociati, etc.), questo da origine ad un errore sintattico, facendo sì che il documento non sia ben formato, senza quindi passare alla seconda fase.

CONVALIDA → valutazione della correttezza sintattica del documento e la sua “well-formedness”.

(II) *CREARE UN LINGUAGGIO XML*

Per definire un linguaggio basato su XML bisogna definire la sintassi (grammatica) di un linguaggio, definendo un insieme di tag, gli attributi dei tag e le possibili combinazioni tra di loro.

Esistono due metodi:

1) DTD (Document Type Declaration);

2) XML SCHEMA (metodo più recente).

DTD

DTD è la specifica della grammatica di un linguaggio basato su XML. Ogni documento ha una sua DTD, la cui conformità ad un documento è contenuta in un file esterno (`<!DOCTYPE esempio SYSTEM “prova.dtd”>`), oppure può essere definita nel documento stesso (`!DOCTYPE esempio [`… *definizione DTD* …`]`).

Esempio

```
<!ELEMENT ricetta (ingrediente+,esecuzione)>

<!ELEMENT esecuzione (fase+)>

<!ELEMENT ingrediente (#PCDATA)>

<!ELEMENT fase (#PCDATA | commento | nota )*>

<!ELEMENT commento (#PCDATA)>

<!ELEMENT nota (#PCDATA)>

<!ATTLIST ricetta    nome CDATA #REQUIRED
                     tempo CDATA #IMPLIED
                     perquanti CDATA #IMPLIED>
```

DICHIARARE UN ELEMENTO

```
<!ELEMENT      nome_elemento
               (modello del contenuto)>
```

- Testo (caratteri) PCDATA
- Un elemento
  - Uno o nessuno ?
  - Nessuno o più *
  - Uno o più +
- Una sequenza di elementi (e1, e2)
- Elementi alternativi (e1 | e2)
- Nessun elemento EMPTY

SEQUENZE

```
<!ELEMENT capitolo (titolo, testo_capitolo)>
```

significa:

un elemento capitolo contiene un solo elemento titolo seguito da un solo elemento testo_capitolo → l'ordine con cui appaiono gli elementi nel modello di contento, separati dalla virgola, è vincolante.

valido

```
<capitolo>
  <titolo>
      Capitolo I
  </titolo>
  <testo_capitolo>
      bla bla bla…
  </testo_capitolo>
</capitolo>
```

NUMERO DI ELEMENTI

```
<!ELEMENT biblioteca(libro+)>
```

significa:

l'elemento biblioteca contiene uno o più elementi libro.

ALTERNATIVE

```
<!ELEMENT titolo_capitolo (titolo|numero)>
```

significa

l'elemento titolo_capitolo contiene un titolo o un numero.

BLOCCHI

Per specificare che un elemento contiene un certo elemento o una sequenza di due elementi si usa

```
<!ELEMENT e1 (e2|(e2,e3)>
```

L'elemento e1 contiene l'elemento e2 oppure la sequenza di e2 e e3.

ESEMPIO

```
<!ELEMENT capitolo((titolo|numero),corpo)>
```

Tipi di valore

Caratteri	CDATA
Numeri	NMTOKEN
Identificativo unico	ID
Alternative	(v1 | v2 | v3)
Entità	**<!Entity …>** **entity declaration**

Attenzione:

- CDATA indica una stringa come tipo di dato di un attributo
- #PCDATA indica il testo contenuto in un elemento

VALORI ALTERNATIVI

```
<!ATTLIST prefazione valutazione(pessimo|sufficiente|buono|
ottimo)#REQUIRED>
```

significa

l'elemento prefazione ha un attributo valutazione obbligatorio che può avere uno dei primi valori specificati.

valido

<prefazione valutazione="ottimo">
<prefazione valutazione="buono">

Non valido

<prefazione valutazione="mediocre">
<prefazione>

COLLEGARE UN DOCUMENTO XML A UNA DTD

DTD può essere esterna o interna al documento.

- DTD esterna → `<!DOCTYPE radice SYSTEM URI>` dopo la dichiarazione XML;
- DTD interna → `<!DOCTYPE radice [dtd]>`

LIMITI

Le DTD hanno alcuni limiti intrinseci:

- non permettono di distinguere tipi di dati diversi (nel documento ci sono solo stringhe di caratteri ed elementi);
- non sono scritte in un XML (le DTD sono scritte in un linguaggio specifico).

SCHEMI XML

La definizione di linguaggi basati su XML tramite "Schemi XML" ha soppiantato l'uso delle DTD (come un foglio di stile XSL è un documento xml, anche uno schema XML è un documento xml → è più espressivo / verboso, cioè permette di creare una specifica più articolata e precisa.

XML SCHEMA è uno standard che ha soppiantato l'uso delle DTD.

In quanto documento XML, uno schema XML comincia con una dichiarazione del tipo di linguaggio:

```
<?xml versione='1.0'?>
```

Tutti i tag hanno il prefisso xsd (XML Schema Definition), che permette di individuare il linguaggio di riferimento.

COLLEGARE UN DOCUMENTO AD UNO SCHEMA

Ogni schema XML deve essere collegato col documento attraverso l'attributo `schemaLocation`, che si colloca nel primo tag del primo elemento del documento.

CONTENT MODEL DTD E SCHEMI XML

La sintassi degli schemi XML trova corrispondenza negli elementi del linguaggio delle DTD.

- `!ELEMENT` diventa `<xs:element…> … </xs:element>;`
- `!ATTLIST` diventa `<xs:attribute…> … </xs:attribute>.`

All'interno dello schema XML si distinguono quelli che sono i tipi di dato semplici (stringhe, data e numero, numerici) e tipi di dato complessi, cioè elementi che contengono altri elementi (sequenze, choice → "alternative", che nel linguaggio DTD sono le barre verticali.

TIPI DI DATO SEMPLICI

```
<!ELEMENT titolo (#PCDATA)>
```

diventa

```
<xs:element name="titolo" type="xs:string"/>
```

- L'elemento titolo viene riferito in altre parti dello schema tramite l'attributo ref.
  - Il valore di ref è il nome (name) dell'elemento a cui ci si riferisce

```
<xs:element ref="titolo"/>
```

Il linguaggio XML Schema contiene molti tipi di dato:

- string
- int
- decimal
- anyUri
- gYear
- dayTime
- …

```
<xs:element name="anno" type="xs:gYear"/>

<anno>2001</anno>
```

ATTRIBUTI IN XML SCHEMA

```
<!ATTLIST copertina immagine CDATA #IMPLIED>
```

diventa

```
<xs:attribute  name="immagine"
               type="xs:string"
               use="optional"/>
```

RESTRIZIONI

E' possibile porre restrizioni sul numero di occorrenze di un elemento (come si fa anche nelle DTD con gli operatori *,+,?).

Nello schema XML, invece di usare i simboli, vengono utilizzati degli attributi `mixOccurs` o `maxOccurs`: il primo indica il minimo numero di occorrenze, mentre il secondo indica il massimo.

Per le valutazioni, invece

```
<xs:attributeGroup name="prefazione">
  <xs:attribute name="valutazione">
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="pessimo"/>
        <xs:enumeration value="sufficiente"/>
        <xs:enumeration value="buono"/>
        <xs:enumeration value="ottimo"/>
      </xs:restriction>
```