

RIASSUNTO

MODELO RELAZIONALE:

• DEFINITO DA TABELLE

• TABELLE:

• COMPOSTE DA TUPLE

• ATTRIBUTO: COLONNA DI UNA TABELLA

• ISTANZA: TUPA CON GLI VALORI

• DOMINIO: INSIEME DEI VALORI DI UNA COLONNA

• CARDINALITÀ: N° DI TUPLE NELLA TABELLA

• GRADO: N° DI ATTRIBUTI NELLA TABELLA

→ SE RELAZIONE TRA TABELLE AVVIEDE PER VALORE → INSIEME NON ORDINATO (\neq PUNTATORI)

• ASSERZIONI DI VALORI: VALORI SPECIALE → 0, NULL, ?

• VINCOLI ← INTRARELACIONALI: INSIEME DI VALORI → ATTRIBUTO DI 1 RELAZIONE

INTERRELACIONALI: DEF. SU PIÙ RELAZIONI

• VINCOLO D'INTEGRITÀ: PROPRIETÀ CHE DICE CHE ESISTE SODDISFAZIONE DI VARI LE RIGHE DI OGNI

→ NON PIÙ AVERE VALORI NULLI

• CHIAVE PRIMARIA: IDENTIFICAZIONE UNIVOCAMENTE DI UNA TUPLA

PK • CHIAVE: UNIVOCATA MINIMA → NON DENE ESSERE → SE SONO MINIME: SUPER-CHIAVE
UNIVOCA SOTTOCHIAVE UNIVOCATA

• VINCOLO DI DOMINIO: CONDIZIONI SUL VALORE ASSUNTO DA UN ATTRIBUTO (D. 0 < VOTO < 30)

• VINCOLO DI TUPLA: CONDIZIONI SUL VALORE DI UNA TUPLA (es. A + B = C)

• VINCOLO D'INTEGRITÀ REFERENZIALE: TABELLE R & S, S FA RIFERIMENTO A R TRAMITE COLONNA X

→ I VALORI DI X POSSONO ESSERE ESCLUSIVAMENTE VALORI ASSUNTI DALLA PK DI R
↳ FK DI S

ALGEBRA RELAZIONALE

• SELEZIONE (G_p): ESTRAE UN SOTTOINSIEME ORIZZONTALE DI A, PER CUI È VERA P → R = $G_p A$

• PROIEZIONE: (π_L): ESTRAE UN SOTTOINSIEME VERTICALE DI A, CON COSTA DI ATTRIBUTI L → R = $\pi_L A$

• PRODOTTO CARTESIANO (X): GENERA NUOVE COPPIE FORMATE DA TUPLE DI A & B → R = A X B

• JOIN (\bowtie): UNIONE COPPIE DI A & B SEQUENTIUNTE (EGOJE) → R = A \bowtie B

↳ NATURAL JOIN: UNIONE DI SOGLI COPPIE SEGUENTIUNTE IN COMUNE

→ EQUIJOIN: CONDIZIONE: "

JOIN CON P → • THETA-JOIN (\bowtie_p): JOIN UNIENDO SUA CONDIZIONE P → R = A \bowtie_p B

NP = \bowtie_A

• SEMI-JOIN (\bowtie_p): CONDIZIONE TUTTE DI A CHE HANNO CORRISPONDENTI IN P CON B: R = A \bowtie_p B

ELIMINA
DUPLICATI
FATTORE
COMPARA DOPPIE

JOIN CON P →

R = $\pi_A (A \bowtie B)$

- DOPPIIUSCITÀ**
- OUTER-JOIN: JOIN CHE CONSONA LE TUTTE LE TABELLE, CONVERGENDO CON "NULL" $\rightarrow R = A \bowtie_p B$
 - (LEFT, RIGHT, FULL): IN BASE A UNA TABELLA CONSERVANTE
 - UNIONE (\cup): CONSONA TUTTE LE MATE DI $A \& B \rightarrow R = A \cup B$
 - INTERSEZIONE (\cap): .. TUTTE LE TUMI PRESENTI SIA IN A CHE $B \rightarrow R = A \cap B$
 - DIFERENZA ($-$): SELEZIONE ESCLUSIVA DELL'UNICA TUMA DI A (\rightarrow NON PRESENTE IN B) $\rightarrow R = A - B$
 - ANNI-JOIN (\bowtie_p): SELEZIONA TUTTE LE MATE NON LEGATE TRA LORO CON $p \rightarrow R = \bowtie_p$
 - DIVISIONE: VALORI DI A PER CHI VACUA TUMA I VALORI DI $B \rightarrow R = A / B$

LEFT \leftarrow
DOPPIO
RIGHT \rightarrow
DOPPIO
FULL \bowtie
DOPPIO

SQL

DDL : DEFINIZIONE

STANDARD QUERY LANGUAGE

DML : MANAGEMNT

SELECT :

SELECT <COLONNA/FS> \rightarrow NON RIMUOVE
FROM <NAME TAB> \rightarrow DUPLICATI
WHERE <CONDIZIONI>

\rightarrow USO SELECT DISTINCT

• SELECT * : SEL. TUTTE LE COLONNE

• SELECT <COLONNA-X> AS <NUOVA COLONNA> \rightarrow AS: SANA LA NUOVA COLONNA IN UNA NUOVA CESTA,
Dove i valori sono quelli di X

WHERE :

• <> : significa " \neq "
• AND, OR, NOT

• ~ : INDICA UN QUASI-CARATTERE

• RICERCA TESTUALE :

• '%' : INDICA UNA QUALSIASI PARZIALE O STRANIA

WHERE <valore> LIKE '% ... %'

• NOT LIKE: \rightarrow NOT LIKE SE NON VACUA UN VALORE

• WESTONE NULL: ... WHERE <ATTRIBUTO> IS NULL / IS NOT NULL

ORDINAMENTO

SELECT ...

From ...

ORDER BY <ATTRIBUTO> ASC / DESC (SE NON SPECIFICATO È ASC)

\rightarrow POSSO ANCHE DEFINIRE UN ORDINE SECONDO IN CASO DI CHIAVI UNIQUE

JOIN

SI FA SFRUTTANDO IL CARATTERE CONNESSO E CONDIZIONI

SELECT <COLONNA>
FROM T_1, T_2, \dots, T_n
WHERE <CONDIZIONI> \rightarrow N TABELLE \rightarrow AMBENTI M-1 COND. DI JOIN

\rightarrow OPPURE SI PUÒ USARE: FROM <TABLE X> (TIPO JOIN) JOIN <TABLE X> ON <COND. DI JOIN>

FUNZ. AGGREGATE

COUNT: N° LINEE

SUM: \sum VALORI PER UNA COLONNA

AVG: MEDIA VALORI

MAX, MIN

SELECT f(A) (<ATTRIBUTO>)

\rightarrow FROM ...

\rightarrow NUOVA SELECT POSSO METTERE: • SOLO f AGGREGATE.

L) USO GROUP BY PER USCIRE INFORMATI • SOLO ATTRIBUTI

• GROUP BY : \rightarrow DEVE ESSERE GUARDATO UN ATTRIBUTO NELLA SELECT

SONO PER POTER USARE I FILTRI E ATTRIBUTI NELLA SELECT

\rightarrow GLI ATTRIBUTI DELLA SELECT SONO OGNI VOLTA GLI STESSI DELLA GROUP BY O MINORI

\rightarrow ● SELECT C GROUP BY
↳ POSSO SOLO ATTIVARE FUNZIONI

• CONDIZ. SU ATTRIBUTI :

NON USO WHERE, MA USO HAVING :

SELECT ... FROM (...)

HAVING ... FROM (...) >/<= < CONDIZ. >

• INTERROGAZIONI MIGLIORI

ANNO DI SELECT :

SELECT
FROM VEDRE : \rightarrow SE IL TBL. RITORNA E' UN SOLO VERSO,
ANNOLO " - " : (SELECT ...) \rightarrow SI POSSANO Poi RECAPITARE IN UN'UNICA SELECT
...) \rightarrow , SOSTITUENDO IL JOIN

• USO OPERATORE IN (o NOT IN) SE IL RISULTATO RITORNATO E' UNA TABELLA CON UN TUPLE

• COSTRUZIONE DI TUPPA: METTO UNA CONDIZ. MULTIMA NBC WHERE: VEDERE (X_1, X_2) IN (SELECT ...)

• EXIST / NOT EXIST : ANDRO' IN / NOT IN \rightarrow BISOGNA ACCORDARE LA CONDIZ. DI CORRELAZIONE NEL WHERE FINALE
L, VEDERE EXISTS (...)

• OMISIONE (IN SQL)

SPRINTO LA DI SUMMA COUNT() E' IL GROUP BY PER VERIFICARE CHE IL N° NOME SIA

CONMESSO

• UNION e INTERSECT :

(SELECT ...)

UNION / INTERSECT / EXCEPT

(SELECT ...)

\rightarrow CORRISPONDE AD UNA DIFFERENZA

• ISTRUZIONI DI AGGIORNAMENTO

• INSERT : \rightarrow INSERIMENTO DI UNA NUOVA TUBA IN TABELLA

INSERT INTO , nome tabella > (ELENCO COLONNE)
VALUES (ELENCO COSTANTI)

INSERT INTO & nome tabella > (ELENCO COLONNE)

\rightarrow SE NON E' INDICATA LA TABELLA, SI INSERISCE IN TUTTI I VINCOLI
DI INTEGRITA' \rightarrow NESSUN INFORMATIVO SENSO

• DELETE :

\rightarrow DENE RISPETTIVE I VINCOLI DI INTEGRITA'

DELETE FROM & nome tabella >
WHERE & CONDIZIONE >

\rightarrow SI POSSONO FARE

DELETE ANIDDE:

DELETE FROM (1)

... IN .. (2)

①, poi ②

• UPDATE :

UPDATE & nome tabella >
SET

& nome . di Aggiornamento >

WHERE & CONDIZIONE >

\rightarrow DENE RISPECTIVE I VINCOLI DI INTEGRITA'

\rightarrow GUARDARE TUTTI LE ALTRE TABELLE

& nome colonna > = & nuovo attributo >

GESTIONE TABELLE: ~DDL

[]: OPERATORI

• CREAZIONE TABELLE:

CREATE TABLE < nome tabella > (< nome attributo > < dominio > (< valore default >) (vincoli), ...)

DOMINI PRIMITIVI: (CHARACTER, BIT, NUMBER, etc. \dots);
DOMINI DEFINITI DALL'UTENTE

CHAR, INT,

↑
TIPO DI
DATO

• DATE DOMINIO PER LE DATE

• DOMINI ELENCATIVI

- NUMBER/DECIMAL (< precisione >, < scala >) \rightarrow PRECISIONE NUMERICHE
- INTERVAL .. TO .. : INTERVALLO DI TEMPO \rightarrow es. day TO SEC
- TIMESTAMP (< precisione >) (< zone >) : YYYY-MM-DD hh:mm:ss:p

• CREAZIONE DOMINI:

CREATE DOMAIN < nome dominio > AS < tipo dato > [vincoli] [vincoli]

dom. successivo

CHECK ...

• ALTER TABLE:

MODIFICHE DI TABELLE:

ALTER TABLE < nome tabella >

• RESTRICT: VIENE BREVETTA LO NON VIENE VINCOLI
 \rightarrow NON VIENE BREVETTA ALCUNO SENSO

- ADD COLUMN < nome nuova colonna > < dominio >;
- DROP COLUMN < " > [RESTRICT/CASCADE]
- ALTER COLUMN < nome colonna > SET/DROP DEFAULT [< valore >]
- DROP TABLE < nome tabella > [RESTRICT/CASCADE]
- ADD/DROP CONSTRAINT [nomi vincoli]

• CASCADE: VIENE SCATENATA, INIZIA A MUOVERE OGNI
ONE DELL'ALTRA DUE ESSE

• DIZIONARIO DEI DATI: CONTIENE DATI SUL DB \rightarrow MEMORY (es. REGRIST, PROCEDURE, STATEMENT, ecc.)

\rightarrow IN ORACLE: USER_*, ALL_*, DBA_*

\hookrightarrow USER.TABLES, USER.TABSTATISTICS, USER.TAB.COL, STATISTICS

• VINCOLI DI INTEGRITÀ:

3 modi per definire:

\hookrightarrow ACCORDI, ACCORDI DEFINITIVAMENTE NEL DB

- PROCEDURE APPLICATIVE: VERIFICA ANI' INTERNO DEGLI' APPLICATIVI (⚠ NON INTEGRI IN DBIS)
- VINCOLI DI INTEGRITÀ: DEFINITI NEL DBMS, DEFINITI DURANTE LA CREAZIONE O AGGIORNAMENTO TABELL
- TRIGGER: PIÙ SOLO COME SQL, ESEGUITI SE TRIGGERTI \hookrightarrow PIÙ DIFFERENTI CONCETTI PIÙ COMPLESSI

• VINCOLI DI TABELE

DEFINITI DURANTE LA CREAZIONE DI TABELLE E DOMINI \hookrightarrow VINCOLI DEDICATI DEL DDL

\hookrightarrow SONO: PK, AMMISSIONE NULL, UNIVALE, VINCOLI DI TUPLA

• PK: DURANTE LA CREAZIONE TABELLE \rightarrow CREATE TABLE ... (... PRIMARY KEY, ...);

\hookrightarrow SE È UN FUSO DI ATTRIBUTI: CREATE TABLE (...);

\hookrightarrow AL FONDO

PRIMARY KEY (... , ...));

- AMMISSIONI NULL: come per PK \rightarrow CREA UNA TABELLA ... (..., NOT NULL)
- UNICITÀ: (ANALOGO \uparrow) \hookrightarrow UNICO NOT NULL: POSSIBILITÀ DI SCELTA COME CHIAVE COMBINATA
- VINCOLI GENERALI: CREATE TABLE ... (... CHECK (< CONDIZIONE >) ...); \hookrightarrow TRA TABELLE
- VINCOLI DI INTEGRITÀ REFERENZIALE:

• COLONNA REFERENZIANTE: COLONNA CHE FA RIFERIMENTO A UN ALTRO \rightarrow FK

• COLONNA REFERENCIATA: ALTRA COLONNA, A CUI SI FA RIFERIMENTO

CHIAVE ESTERNA

IN: CREATE TABLE ... (...

FOREIGN KEY (< nome colonna >)
REFERENCES (< nome tabella >) (< nome colonna >)

LESSONE VINCOLI:

CONSEGUenze SULLE TABELe REFERENZIALI:

• CASCADE: PROPAGAZIONE DEL' OPERAZIONE

• SET NULL/DEFAULT: I VALORI CHE NON TENGONO UN CORRISPONDENTE VERSO SONO SETTI A NULL/DEFAULT

• NO ACTION: NON SI ESEGUI L'AZIONE INVALIDANTE

\rightarrow LA LESSONE (A SI RECIDE NELLA CREATE TABLE: \rightarrow

POSITION KEY (...)
REFERENCES
[ON UPDATE/DELETE
< scelta di azione >]

AGGIORNAMENTO DI TABELLE:

• INSERT: INSERT INTO < nome tabella > (< seconda colonna >)
VALUES (< elenco valori >)

\rightarrow PUÒ AVERE SOGLIE UN'INTERNAZIONE DI SELECT

• DELETE: DELETE INTO < nome tabella >
WHERE < condizioni >

\rightarrow PUÒ ESSERE UN'INTERNAZIONE

• UPDATE < nome tabella >
SET < nome colonna > = < nuovo valore >
WHERE < condizioni >

\rightarrow PUÒ ESSERE UN'INTERNAZIONE

GESTIONE DELLE TRANSAZIONI:

NECESSITÀ DI SAPER GESTIRE IN UN DB L'AGGIÓ NUMERO DI DUE UTENZE IN CONTEMPORANEA

\rightarrow DUE TRANSAZIONI PUÒ TERMINARE CON SUCCESSO (COMMIT [WORK]) O INSUCCESSO (ROLLBACK [WORK])

• PROPRIETÀ ACID DELLE TRANSAZIONI:

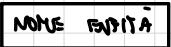
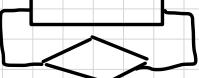
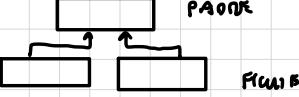
\checkmark ACID. A: ATOMICITÀ \rightarrow LA TRANSAZIONE AVVIENE PER TUTTO O NIENTE, INDIVISIBILE

$\checkmark \rightarrow \checkmark$ C: CONSISTENZA \rightarrow PASSAGGIO DAL DB DA UNO STATO CONCORDO AD UN ALTRO CONCORDO

$T_1 // T_2 // T_m$. I: ISOLAZIONE \rightarrow DUE TRANSAZIONI È ISOLATA, INDIPENDENTE DALLE ALTRE

$T \xrightarrow{\text{trans}}$ D: PERSISTENZA (durabilità) \rightarrow MODIFICAZIONI: PERMANENTI.

MODELLO ER

- FANTÀ:  \rightarrow rappresenta classi di oggetti
- RELAZIONE:  \rightarrow relazione logica tra 2 o più entità
- CARDINALITÀ RELAZIONE: \forall entità, più quantità occorrente di una relazione tra entità
 $\rightarrow (\text{min: } 0/1, \text{ max: } 1/N)$
- RELAZIONE TERNAZIA: \exists entità coinvolte
- RELAZIONE RICORSIVA: 
- ATTRIBUTO:  \rightarrow nome attributo \rightarrow possiede una proprietà resa' su una relazione
- \hookrightarrow hanno cardinalità
- IDENTIFICATORE: attributo univoco per un entità:
- GENERALIZZAZIONE: 

genitore:	
composto:	

 \hookrightarrow una relazione ha solo un identificatore
- \hookrightarrow una relazione ha uno o più generalizzatori
- TOTALE: \forall entità di Padre \rightarrow è ad almeno 1 entità figlie (es. uomo-donna)
- PARTZIALE: ALTRIMENTI
- ESCLUSIVA: \forall entità di Padre \rightarrow è al max 1 di una delle entità figlie (es. uomo-donna)
- SOPRAPOSTA: ALTRIMENTI (es. scrittore-venuta)
- DICTIONARIO DEI DATI, VINCOLI DI INTEGRITÀ, REGOLE DI OPERAZIONE DEI DATI
- PROGETTAZIONE LOGICA:
 - \rightarrow RISTRUTTURAZIONE e TASSONIA DEL MODELO ER \rightarrow semplificazione e ottimizzazione
 - ACCOMPAGNAMENTO DELL'ENTITÀ: FIGLIE IN PADRE / PADRE IN FIGLIE ① ATTRIBUTO TIPO
 - PARTEZIONAMENTO DI CONCETTI:
 - ELIMINAZIONE DI ATTRIBUTO COMPOSTI
 - REL. $(0, N)$ \rightarrow TASSA CON PK COMPOSTA
 - ID. ESTERNO; PK COMPOSTA
 - \rightarrow BIDIMENSIONALITÀ, VINCOLI DI INTEGRITÀ REFERENZIALI:
 - \hookrightarrow NOME TABLOWS (\leftarrow nome colonna) REFERENCES NOME TABLOWS (\leftarrow nome colonna)
- NORMALIZZAZIONE \hookrightarrow obiettivo soddisfare: decomposizione SENZA PERDITA E CONSERV. DUE OPISSANTI X DISPERDENDO X DISTRIBUENDO
 - DIPENDENZA FUNZIONALE: DATA UN RELAZIONE $X \rightarrow Y$, A UNA VALEURE t DI $X \rightarrow$ STESSA VALEURE PRE $t(f)$ DI Y
 - BOYCE COX NORMAL FORM (BCNF): SIA $f: X \rightarrow Y$ \rightarrow X CONTIENE UNA CHIAVE DI f
 - \hookrightarrow NON CI SONO ANOMALIE E RISONDANZE \rightarrow CONCETTI INDEPENDENTI SONO IN RELAZIONE DIFFERENTI
 - NORMALIZZAZIONE: SOSTITUISCO UNA RELAZIONE R CON UNA O PIÙ BCNF

GESTIONE DELLE VISTE:

→ SE NECESSARIO

CREATE, VIEW < nome vista > (< attributi vista >) AS < interrogazione SQL >

→ POSSO POCO USARE LA VISTA CREATA PER INTERROGAZIONI SUCCESSIVE

CONSEGUENTI VISTE: DROP VIEW

MODIFICA VISTA: ALTER VIEW < nome vista > (< nuovi attributi >) AS < interrogazione SQL >

• AGGIORNABILITÀ: → PER POI POSSO PROPAGARE SUE TUTTE LE TAB.

SÌ: 1 riga esattamente corrisponde a 1 riga della vista → CORRISP. 1:1 tra VIEW e TAB.
NO: CONTIENE DISTINCT, IF ANONYMOUS, JOIN (1->N o N->N), NO PK TABELLA ORIGINALE

↳ SI PUÒ CARRIARE L'INTERROGAZIONE DAI DATI ALLA AGGIORNAMENTO

* PER LE VISTE AGGIORNABILI, SI PUÒ INSERIRE: WITH [LOCAL/CASCADE] CHECK OPTION → CONTROLLO RIGOROSO VINCOLI

→ POSSONO SOGGETTI AGLI VINCOLI, SENZA OPERAZIONI NON PERMESSA

↳ LOCAL: VERIFICA AGGIORNAMENTO SOLO SU VISTA EST. CASCATA: → SU TUTTE LE VISTE CONNESSE

• AGGIORNAMENTO TUPLE: UPDATE < nome vista > SET < operazione >

CONTROCCIO DELL' ACCESSO:

• LE RISORSE DI UN DB SONO PROTETTE TRAMITE DEI PRIVILEGI DI ACCESSO

< ESERCIZIO PRIVILEGI > : INSERT, DELETE, UPDATE, SELECT, REFERENCES, USAGE → PERMESSO DI UTILIZZARE UNA RISORSA

• CONCESSIONI PRIVILEGI: GRANT < ESERCIZIO PRIVILEGI > ON < nome risorsa > TO < ESERCIZIO UTENTE > [WITH GRANT OPTION]

↳ ALL PRIVILEGES PER DARE IL PRIVILEGIO A TUTTI

PERMESSO AD OGNI UTENTE

• REVOCARE PRIVILEGI: REVOKE < ESERCIZIO PRIVILEGI > ON < nome utente > FROM < ESERCIZIO UTENTE > [RESTRICT|CASCADE]

↳ NON POSSETTE REVOCARE SE HA GIA' PRIVILEGI IN ALTRI

• REVOKE: INSIEME DI PRIVILEGI

→ ATTRIBUISCHE AD UN UTENTE

• DEFINIZIONE RUOLO: CREATE ROLE < nome ruolo >

• MODIFICA RUOLO: SET ROLE < nome ruolo >

• APPLICAZIONI WEB, HTML e PHP:

• ARCHITETTURA / CENTRALIZZATA : APP. SONO RIPIEGATI IN UN NODO CENTRALE, Q. CON UN ALTRO ENGINERINGS POSSONO RIFERIRSI
 • DISTRIBUITA : " " " Sono distribuiti TRA I NODI
 ↗ SPORTE APP. DI UN SISTEMA UNICO
 ↗ MIGLIORE PORTABILITÀ E INTEROPERABILITÀ

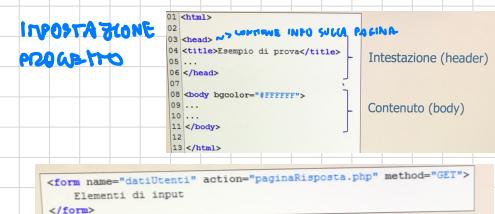
- CLIENT: APP. CHE UTILIZZA I SERVIZI { COMUNICO TRAMITE PROTOCOLLI
- SERVER: APP. CHE FORNISCE SERVIZI
- ACTOR: ASSUME ENTRATE I DATI

• ESEZIONE SQL / COMPLIES & LD: LA QUERY È INVIA A UN SERVER DAL KEEPER

• COMPILE & STORE: LA QUERY È INVIA AL SERVER, RIFIGURATO, DAL SERVER → UNA OVALE QUERY RIPRINTATA

• HTML:

- TAG E ATTRIBUTO
- PAGINE WEB STATICHE, NON DINAMICHE
- URLIZZO DEI TAG C FORMI PER L'INVIO DATI A UN SERVER



- METODO GET: I DATI INVIATI SONO PASSATI IN CHIAVE NELL'URL
- METODO POST: 2 FASI → PRIMA CONTATRA IL SERVER, Poi INVIO DATI → GLI INPUT INVIATI NON SONO VISIBLE
- <input>; CREA FORMULIARIO GLI INPUT, es. BOTTONI (type='button'), TESTO (type='text'), ecc.
- TABLES:
 - ROW & COLUMN:

trovata la pagina :

```

<select name="...>
  <option> ... </option>
  <option> ... </option>
  ...
  </select>
  
```

```

<table border="1">
  <tr>
    <td>prima cella</td>
    <td>seconda cella</td>
  </tr>
  <tr>
    <td>terza cella</td>
    <td>quarta cella</td>
  </tr>
</table>
  
```

prima cella	seconda cella
terza cella	quarta cella

→ A NOME I TAG <p>
 VERRÀ APPLICATO LO
 STILE DEFINITO NEL CSS

→ CREO UNA CLASSE

```

<html>
  <head>
    <link href="mystyle.css" rel="stylesheet type="text/css" />
  </head>
  <body>
    <p> Stile Esterno. </p>
  </body>
</html>
  
```

• CSS:

MODIFICARE IL DESIGN DI UNA PAGINA HTML

- IN-LINE: CSS DEFINITO ALL'INTERNO DELLO stesso TAG: <.. style=" [css] " .. >
- INTERNA: CSS ALL'INTERNO UNA PAGINA HTML → DEFINITO IN HEADER: <style> ... </style>
- ESTERNA: CSS IN UN DOCUMENTO ESTERNO
- CLASSES: .NAME.CLASS { [ATTRIBUTI]} → VERRÀ APPLICATA A TUTTI I TAG CON ATTRIBUTO CLASS= 'NAME.CLASS'
- ID: IDENTIFICA UN SOLO ELEMENTO
- HIERARQUIA: ELEMENTO → CLASSE → ID → IN-LINE

più specifico

• PHP:

IN HTML ALL'INTERNO DEI TAG. < ? php ... ? >

• VARIABILI: \$VAR

• isset(): CONTROLLO CHE LA VARIABILE SIA DEFINITA → RETURN TRUE/FALSE

• is_int(), is_float(), is_bool(), is_string(), is_numeric()

• ARRETI: SEPARA E ASSOCIA (es. \$var["chiave"])

↳ POSSONO ESSERE ASSOCIAZIONI

- CONCATENAZIONE DI STRUTTURE ; USO IL PUNTO $\rightarrow \$x . \y
- STAMPA : echo "Hello world \$a" \rightarrow STAMPA "Hello world \$a" se $\$a=5$
- FOR ANIDATO : foreach ($\$a$ as $\$value$) \uparrow VETTORE
- SCOPE : DEFINISCE DOVE LE VARIABILI SONO VISIBILI : LOCAL / GLOBALE SCOPE
- VARIABILI SUPERGLOBALI : $\$_GET$ contiene tutte le var passate col metodo GET
 $\$_POST$ POST } SOLO ARRAY ASSOCIAZIONI
- die(), exit() TERMINA L'ESECUZIONE DELLO SCRIPT
- $\$_REQUEST$: ARRAY ASSOCIAZIONE CONTENENTE TUTTE LE VAR IN GET, POST & COOKIES
 $\$var = \$_REQUEST["name"]$ \hookrightarrow DEFINITO COME ATTRIBUTO: name = "NOME"

• SQL PER LE APPLICAZIONI

Mysql per PHP:

- mysqli_connect() : PERMETTE DI CONNETTORSI AL DB
- mysqli_connect_errno() : RESTORNA IL CODICE DI ERRORE
- mysqli_connect_error() : UNA CATEGORIA D'ERRORE (STRINNA)
- die() : ARRESTA IL SCRIPT E STAMPA MESSAGGIO
- mysqli_close() : CHIUSA LA CONNESSIONE AL DB

```
// Connessione a MySQL tramite mysqli_connect()
$ccon = mysqli_connect('localhost', 'root', 'mysql', 'db_name');
// Aggiungere la password se necessaria
// Connessione
if (mysqli_connect_errno())
{
    die('Failed to connect to MySQL: ' . mysqli_connect_error());
}

// Chiusura della connessione
mysqli_close($ccon);
```

ESECUZIONE IMMEDIATA: \sim LA QUERY NON È MIGRATA IN RAM

- SCRIBO LA QUERY IN UNA VAR
- USO mysqli_query(\$con, \$sql) PER ESEGUIRE LA QUERY
 \downarrow CONNESSIONE \downarrow ESECUSIONE SQL
 \hookrightarrow RESTORNARE IL RISULTATO

```
/* QUERY SQL */
$sql = "SELECT autore.cognome, opera.nome,
           FROM autore, opera
           WHERE autore.coda = opera.autore";
$result = mysqli_query($con, $sql);

if (!$result)
    die("Query error: " . mysqli_error($con));
```

ESECUZIONE PREPARATA :

AVVIENE IN 2 FASI:

- USO mysqli_prepare(\$con, "struttura SQL con '?' per segnali")
 \hookrightarrow return uno mysqli_stmt,

```
// preparazione della query
$stmt = mysqli_prepare($con, "INSERT INTO Forniture VALUES (?, ?, ?)");
// inserimento dei valori
$stmt->bind_param("sss", $NomeF, $Tipologia, $Qta);
// Passo i valori da inserire
```

• SCRIBO LA QUERY CON mysqli_stmt_execute(\$stmt)

- USO mysqli_fetch_row() : RESTORNA UN ARRAY CORRESPONDENTE ALLA RIGA DESIATA RISULTATI

```
while ($row = mysqli_fetch_row($result)) {
    echo "<tr>";
    echo "<td>$row[0]</td><td>$row[1]</td>";
    echo "</tr>";
}

while ($row = mysqli_fetch_row($result)) {
    foreach ($row as $cell) {
        echo "<t><td>$cell</td></t>";
    }
    echo "</tr>\n";
}
```

- my-sqli-fetch-assoc() : INSERISCE, MA È UN ARRAY ASSOCIAZIONE

• my-sqli-num-fields(), my-sqli-num-rows() \downarrow CONTROLLA IL NUMERO DI RIGHE. ABILITA/DEABILITA AUTO COMMIT.

- TRANSAZIONI: bool my-sqli-autocommit(mysqli \$link, bool mode), SE SI SETTA A TRUE

NomeF	NSoci
Andrea	2
Gabriele	2

```
bool mysqli_commit (mysqli$link)
bool mysqli_rollback (mysqli$link)
```

• GESTIONE DEGLI INDICI:

INDICI: struttura fisica accessoria, per rendere più efficiente l'esecuzione di query

↳ CONTIENE RIFERIMENTI AI VALORI DESIDERATI

↳ EFFICIENTI IN TERMINI DI TEMPO, NON DI SPAZIO

→ ricerche tramite ALBERI o TABELLE o HASH o STRUTTURE SEQUenzIALI

• CREATE INDEX < nome indice > ON < nome tabella > (< elenco attributi >)

↳ I Dati vengono memorizzati ordinati

• DROP INDEX < nome indice >

• UN INDICE SI DEFINISCE DURANTE LA PROGETTAZIONE FISICA

↳ IN BASE AGLI UTILIZZI, SI SCELGONO GLI INDICI DA CREARE → SE SI VOLGONO MIGRARE: TUNING

ALCUNO O TOLGO
↘ INDICI

• TRIGGER:

DB ATTIVI: ESSICANO IN AUTOMATO LE ECA CON LE OVATIVE
→ EVENTO, CONDIZIONE, AZIONE

• MODE: BEFORE/AFTER → TRIGGER ESEGUITO PRIMA/DOPPO LO STATEMENT DEL TRIGGER

• EVENT: INSERT/DELETE/UPDATE [OF < column name >]

• TARGET TABLE: quando viene inserito su chi eseguirlo EVENT
↳ GRANULARITÀ

• FOR EACH ROW/STATEMENT: trigger eseguito A TUTTI OPERAZIONI 1 Sola Volta per tutti le righe

• ACCESSO AI 2 STATEMENT ROW: OLD.COLUMN NAME & NEW.COLUMN NAME → SOLO PER FOR EACH ROW

• WHEN: CONDIZIONE PER CHI ESEGUE IL TRIGGER

↳ SOLO SE FOR EACH ROW → lavoro a livello di TUPA

• PL/SQL BLOCK: ACTION TO EXECUTE

→ PUÒ SVOLGERE OPIÙ DI UNA ESECUZIONE DI TRIGGER A CASO → POSSIBILE SETTARE UN LIMIT (LIMIT > 32 TRIGGERS)

• MUTATING TABLE: TABELLA GENERATA DAL TRIGGER, ACCESSIBILE SOLO DAL TRIGGER STESSO → FOR EACH STATEMENT

• SQL BLOCK:

DECLARE

< VAR. DA dichiarare >

BEGIN

< BODY DEL CODICE > ↳ SCRIVERE VARIABILI: ... INTO N

 < IF() THEN ... END IF >

END;

• TRIGGER DESIGN:

