Unitat 4

Llenguatge SQL. DDL. Creació de taules.

SQL

Llenguatge utilitzat per a la definició, manipulació i control de la base de dades.

Es pot dividir en:

- Llenguatge de **definició** de dades (DDL).
- Llenguatge de **manipulació** de dades (DML).
- Llenguatge de **control** de dades (DCL).

Tipus de dades

Els atributs de les taules poden ser de diferents tipus, segons les necessitats:

- Dades numèriques:
  - ID, quantitat, preu...
- Cadenes de caràcters o *strings:*
  - Descripció d'un producte, adreça, nom...
- Dates i hores:
  - Data i hora d'entrega, data de finalització, data d'inici...

Veure: Oracle Built-in Data Types

Dades numèriques: NUMBER

Representa nombres positius i negatius amb un nombre fixat de xifres

`NUMBER(p,s)`

p (*precision*): nombre de dígits significatius (màxim: 38)

s (*scale*): nombre de dígits després del punt decimal

Veure Oracle: Storage of Scale and Precision

Per exemple:

`NUMBER(7)`	Sencer de 7 dígits SENSE DECIMALS.
`NUMBER(10,5)`	Decimal amb 5 dígits abans i 5 després del punt.
`NUMBER`	Oracle s'adaptarà als valors

Dades numèriques: altres

- FLOAT FLOAT(p)

  p: nombre de dígits binaris significatius(per defecte 126 (38 decimal))

- BINARY_FLOAT: *floating point* de 4 bytes.

- BINARY_DOUBLE: *floating point* de 8 bytes.

Cadenes de caràcters

Guarden caràcters alfanumèrics.

Tipus:

- CHAR: cadena de caràcters de mida fixa.

  `CHAR(n)`

- VARCHAR2: cadena de caràcters de mida variable.

  `VARCHAR2(n)`

Dates i temps

Serveixen per representar marques temporals, de diferent precisió.

Tipus:

- DATE: temps amb precisió de segons (com a màxim).
  - Exemples: 2023-12-21, 2024-08-18 09:14:56
- TIMESTAMP (amb variacions WITH [LOCAL] TIME ZONE):
  - Temps amb precisió de fraccions de segons.
  - Exemples: 2023-03-12 14:21:44.333

Tipus no-Oracle (tipus ANSI SQL)

Conversió segons la següent taula:

ANSI SQL Data Type	Oracle Data Type
CHARACTER(n) CHAR(n)	CHAR(n)
CHARACTER VARYING(n) CHAR VARYING(n)	VARCHAR2(n)
NATIONAL CHARACTER(n) NATIONAL CHAR(n) NCHAR(n)	NCHAR(n)
NATIONAL CHARACTER VARYING(n) NATIONAL CHAR VARYING(n) NCHAR VARYING(n)	NVARCHAR2(n)
NUMERIC[(p,s)] DECIMAL[(p,s)]	NUMBER(p,s)
INTEGER INT SMALLINT	NUMBER(38)
FLOAT DOUBLE PRECISION REAL	FLOAT(126) FLOAT(126) FLOAT(63)

Creació de taules: CREATE TABLE

Per crear una taula, s'ha d'utilitzar la instrucció CREATE TABLE amb el nom de la nova taula, i després, entre parèntesi, una llista de les seves columnes (atributs), amb els seus noms i tipus de dades. La sintaxi és la següent:

```
CREATE TABLE nom_taula (
    columna1 tipus_dada,
    columna2 tipus_dada,
    columna3 tipus_dada,
    ...
);
```

```
CREATE TABLE client (
    id number,
    nom varchar2(50),
    llinatge1 varchar2(50),
    llinatge2 varchar2(50),
    email varchar2(100),
    data_registre date
);
```

Creació de taules: clau primària.

La clau o claus primàries es marquen amb PRIMARY KEY i posant entre parèntesis l'atribut (o conjunt d'atributs) que en formen part:

```
CREATE TABLE nom_taula (
    columnaPK tipus_dada,
    columna2 tipus_dada,
    columna3 tipus_dada,
    PRIMARY KEY (columnaPK)
);
```

```
CREATE TABLE client (
    DNI varchar2(9),
    nom varchar2(50),
    adreca varchar2(100),
    email varchar2(100),
    PRIMARY KEY (DNI)
);
```

Creació de taules: claus foranes

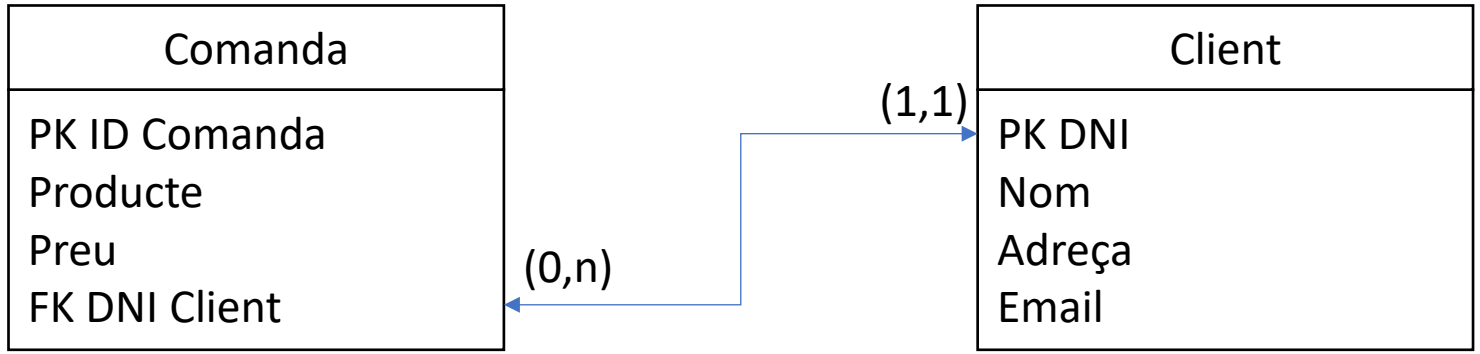
Les claus foranes s'especifiquen afegint FOREIGN KEY i, entre parèntesis, la columna o columnes que la formen.

Seguidament, s'ha d'utilitzar REFERENCES i la taula referenciada amb la seva clau primària entre parèntesis.

```
CREATE TABLE nom_taula (
    columnaPK tipus_dada,
    columna2 tipus_dada,
    columna3 tipus_dada,
    columnaFK tipus_dada,
    PRIMARY KEY (columnaPK),
    FOREIGN KEY (columnaFK)
        REFERENCES nom_taulaFK(columnaPK_taulaFK)
);
```

```
CREATE TABLE comanda (
    id_comanda NUMBER,
    DNI_client varchar2(9),
    producte varchar2(40),
    preu NUMBER,
    PRIMARY KEY (id_comanda),
    FOREIGN KEY (DNI_client) REFERENCES client(DNI)
);
```

Pas de model relacional a codi SQL. Exemple




```
CREATE TABLE comanda (
    id_comanda NUMBER,
    DNI_client varchar2(9),
    producte varchar2(40),
    preu NUMBER,
    PRIMARY KEY (id_comanda),
    FOREIGN KEY (DNI_client) REFERENCES client(DNI)
)
```

```
CREATE TABLE client (
    DNI varchar2(9),
    nom varchar2(50),
    adreca varchar2(100),
    email varchar2(100),
    PRIMARY KEY (DNI)
)
```

Claus foranes: ON DELETE

Quan cream una clau forana, podem indicar a Oracle com s'ha de gestionar l'eliminació del registre vinculat. Ho feim afegint ON DELETE [Opció].

Opcions:

- SET NULL: Si s'elimina el registre pare, la FK del fill quedarà a NULL
- CASCADE: Si s'elimina el registre pare, s'eliminarà TOT el registre fill.
- Per defecte (sense ON DELETE): No permet l'eliminació.

Consideram com a **registre pare** aquell que té la PK i com a **registre fill** el que té la FK.

Claus foranes: ON DELETE

Exemple:

```
CREATE TABLE expedient(
    id number,
    nom_alumne varchar2(20),
    llinatge_alumne varchar2(50),
    primary key (id),
    foreign key(nom_alumne, llinatge_alumne)
        references alumne(nom, llinatge)
        ON DELETE CASCADE
);
```

Si esborram un alumne, s'esborraran tots els seus expedients

```
CREATE TABLE expedient(
    id number,
    nom_alumne varchar2(20),
    llinatge_alumne varchar2(50),
    primary key (id),
    foreign key(nom_alumne, llinatge_alumne)
        references alumne(nom, llinatge)
        ON DELETE SET NULL
);
```

Si esborram un alumne, tots els seus expedients quedaran amb nom_alumne i llinatge_alumne a NULL

```
CREATE TABLE expedient(
    id number,
    nom_alumne varchar2(20),
    llinatge_alumne varchar2(50),
    primary key (id),
    foreign key(nom_alumne, llinatge_alumne)
        references alumne(nom, llinatge)
);
```

Si esborram un alumne, el SGBD no ens permetrà fer-ho, excepte en el cas que no tengui cap expedient vinculat.

Codis incrementals: GENERATED AS IDENTITY

Si tenim codis que s'han d'incrementar per cada entitat nova, ho hem d'indicar al CREATE TABLE amb GENERATED AS IDENTITY:

```
CREATE TABLE comanda (
    id_comanda NUMBER GENERATED AS IDENTITY,
    DNI_client varchar2(9),
    producte varchar2(40),
    preu NUMBER,
    PRIMARY KEY (id_comanda),
    FOREIGN KEY (DNI_client) REFERENCES client(DNI)
);
```

Codis incrementals: GENERATED AS IDENTITY

Síntaxi: GENERATED (ALWAYS | (BY DEFAULT [ON NULL])) AS IDENTITY

- ALWAYS (opció per defecte): Sempre es genera l'atribut identitat. Si l'usuari intenta inserir un valor predefinit per l'atribut, dona error.
- BY DEFAULT: L'atribut identitat només es genera si l'usuari no el proporciona. Si s'utilitza ON NULL, també es generarà si l'usuari proporciona un valor que sigui o s'avaluï a NULL.

```
CREATE TABLE expedient(
    id number GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    nom_alumne varchar2(20),
    llinatge_alumne varchar2(50),
    foreign key(nom_alumne, llinatge_alumne) references alumne(nom, llinatge)
);
```

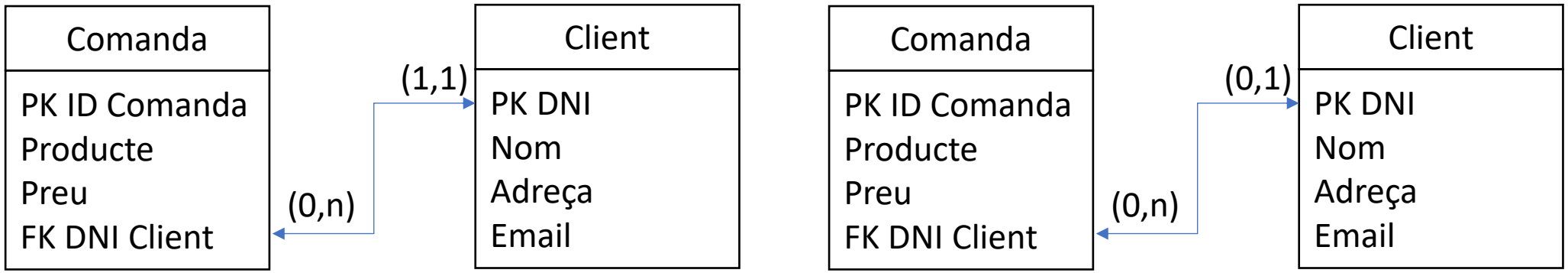
Atributs obligatoris: NOT NULL

Amb la restricció (*constraint*) NOT NULL obligam que no hi pugui haver cap registre sense un valor per l'atribut indicat:

```
CREATE TABLE comanda (
    id_comanda NUMBER GENERATED AS IDENTITY,
    DNI_client varchar2(9) NOT NULL,
    producte varchar2(40),
    preu NUMBER,
    PRIMARY KEY (id_comanda),
    FOREIGN KEY (DNI_client) REFERENCES client(DNI)
);
```

Atributs obligatoris: NOT NULL

Posar NOT NULL o no a una clau forana és molt important, ja que marca la diferència entre la participació total o parcial a la relació.



NOT NULL

Atributs amb valors irrepetibles: UNIQUE

Amb UNIQUE podem forçar que un atribut (o diversos) que no sigui clau primària NO es pugui repetir. Les claus primàries ja són UNIQUE.

Exemple:

```
CREATE TABLE treballador(
    id number primary key,
    dni varchar2(9) unique,
    nom varchar2(50),
    primer_llinatge varchar2(100),
    segon_llinatge varchar2(100),
    unique(nom, primer_llinatge, segon_llinatge)
);
```

Comprovacions valors atributs: CHECK

Es poden restringir els valors vàlids d'un atribut determinat amb CHECK.

Exemple:

```
CREATE TABLE alumne(
    nom varchar2(20) CHECK (LENGTH(nom) > 2),
    llinatge varchar2(50),
    CHECK(SUBSTR(llinatge,1,1) != 'A'),
    PRIMARY KEY (nom, llinatge)
);
```

Exercici

Crea les taules SQL del model relacional de la Biblioteca.

Noves restriccions:

- Títol i any han de ser únics (en conjunt) a Llibre.
- Nom, cognoms i nacionalitat han de ser únics (en conjunt) a Autor.
- L'any d'un llibre no pot ser superior a 2023.
- Els exemplars d'un llibre han de ser superiors a 0.
- Posau els NOT NULL a les FK que calguin segons les cardinalitats.

Alteració de taules: ALTER TABLE

ALTER TABLE permet alterar taules ja existents:

- Afegir columnes noves.

  ALTER TABLE nom_taula ADD nom_columna tipus [constraint];

  - Es pot passar una llista de columnes entre parèntesis:

    ALTER TABLE nom_taula ADD (nom1 tip1 [cons1], nom2 tip2 [cons2], ...);

- Eliminar columnes existents.

  ALTER TABLE nom_taula DROP COLUMN nom_columna;

Alteració de taules: ALTER TABLE

- Modificar el tipus d'una columna.

  ALTER TABLE nom_taula MODIFY nom_columna tipus [constraint];

- Renombrar columnes.

  ALTER TABLE nom_taula RENAME COLUMN nom_columna TO nou_nom;

- Renombrar taules.

  ALTER TABLE nom_taula RENAME TO nou_nom;

Alteració de taules: ALTER TABLE

Exemples:

```
ALTER TABLE comanda ADD data DATE;

ALTER TABLE comanda MODIFY producte varchar2(100);

ALTER TABLE comanda DROP COLUMN preu;
```

Eliminació de taules: DROP TABLE

Les taules s'eliminen amb DROP TABLE i el nom de la taula.

```
DROP TABLE nom_taula;
```

```
DROP TABLE comanda;
DROP TABLE client;
```

IMPORTANT!!

Si la PK d'una taula és la FK d'una altra taula, abans s'ha d'eliminar aquesta segona taula. Quan s'elimina una taula, no hi pot haver informació que hi faci referència, excepte si s'ha usat ON DELETE.

Esborrar dades d'una taula: TRUNCATE TABLE

Les dades d'una taula (tots els seus registres) s'eliminen amb TRUNCATE TABLE i el nom de la taula:

```
TRUNCATE TABLE nom_taula;
```

```
TRUNCATE TABLE expedient;
TRUNCATE TABLE alumne;
```

Tot i que pugui semblar una operació DML, es considera DDL pel seu funcionament intern i perquè no es pot desfer (*rollback*).

S'apliquen les restriccions de les FK (com a DROP TABLE).