Fizično načrtovanje

Koraki fizičnega načrtovanja

Uvod

- Cilj je iz modela **izdelati podatkovno shemo** za **izbrani SUPB**.
- Poznati je potrebno ciljni SUPB (Oracle, MS SQL, mySQL ipd.)
  - Podpora ključem (primarni, tuji ipd.)
  - Obveznost podatkov (NULL in NOT NULL)
  - Domene
  - Omejitve podatkov
  - Sprožilci (triggers) in bazni programi (stored procedures).

Koraki fizičnega načrtovanja (1)

- K3: **Pretvori logični model v jezik za ciljni SUPB**
  - K3.1: **Izdelaj načrt osnovnih relacij**
  - K3.2: **Izdelaj načrt predstavitve izpeljanih atributov**
  - K3.3: **Izdelaj načrt splošnih omejitev**

Koraki fizičnega načrtovanja (2)

- K4: **Izdelaj načrt datotečne organizacije ter indeksov**
  - K4.1: **Analiziraj transakcije**
  - K4.2: **Izberi datotečno organizacijo**
  - K4.3: **Določi indekse**
  - K4.4: **Oceni velikost podatkovne baze**

Koraki fizičnega načrtovanja (3)

- K5: **Izdelaj načrt uporabniški pogledov**
- K6: **Izdelaj načrt varnostnih mehanizmov**
- K7: **Preveri smiselnost uvedbe nadzorovane redundance podatkov (denormalizacija)**

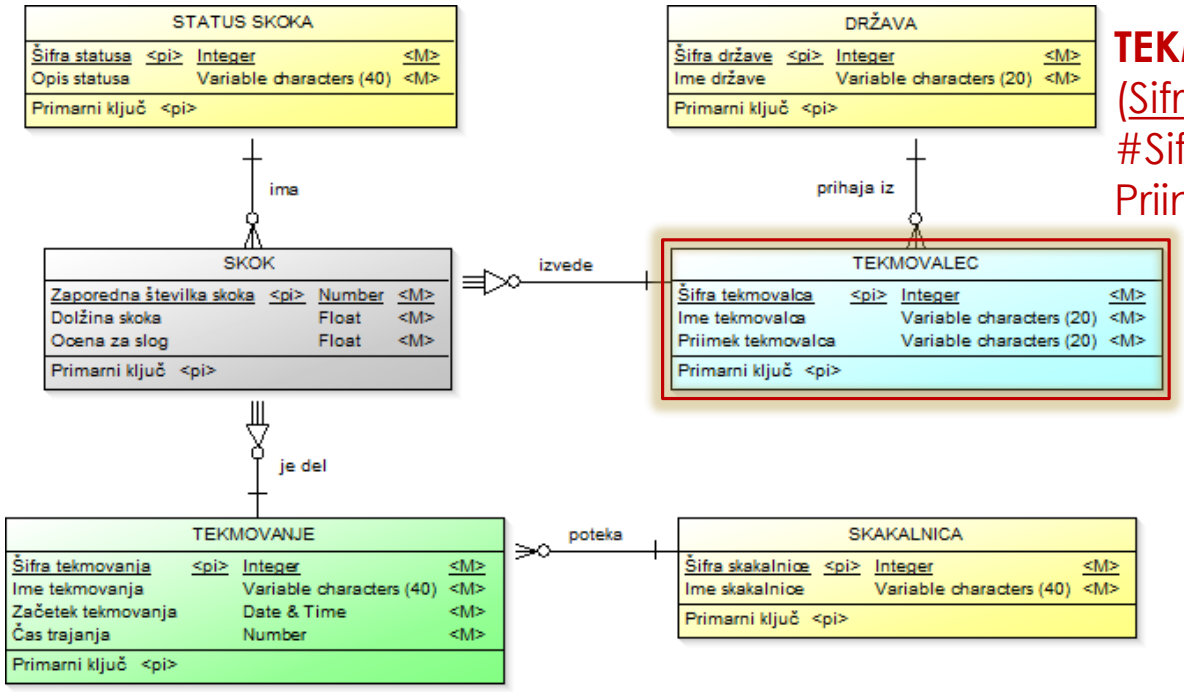
Koraki fizičnega načrtovanja (K3.1) (1)

- K3: **Pretvori logični model v jezik za ciljni SUPB**
  - K3.1: **Izdelaj načrt osnovnih relacij**

- Določiti je potrebno, **kako** bodo osnovne **relacije predstavljene** v ciljnem SUPB.
- Za vsako relacijo opredelimo:
  - **naziv** in **seznam osnovnih atributov**,
  - **primarni ključ** in potencialne tuje ključe ter
  - **omejitve povezav**.

Koraki fizičnega načrtovanja (K3.1) (2) (primer)

- **Primer** relacije iz smučarskih skokov



TEKMOVALEC (Sifra_tekmovalca, #Sifra_drzave, Ime, Priimek)

Koraki fizičnega načrtovanja (K3.1) (3) (primer)

- **TEKMOVALEC** (Sifra_tekmovalca, #Sifra_drzave, Ime_tekmovalca, Priimek_tekmovalca)

↓ **Oracle** RDBMS

```
create table TEKMOVALEC  (
   Sifra_tekmovalca          INTEGER          not null,
   Sifra_drzave              INTEGER          not null,
   Ime_tekmovalca            VARCHAR2(20)     not null,
   Priimek_tekmovalca        VARCHAR2(20)     not null,
   constraint PK_TEKMOVALEC primary key (Sifra_tekmovalca)
);
```

Tabela z atributi in primarnim ključem

```
create index prihaja_iz_FK on TEKMOVALEC (Sifra_drzave ASC);
```

Indeks

```
alter table TEKMOVALEC add constraint
FK_TEKMOVAL_PRIHAJA_I_DRZAVA foreign key (Sifra_drzave)
references DRZAVA (Sifra_drzave);
```

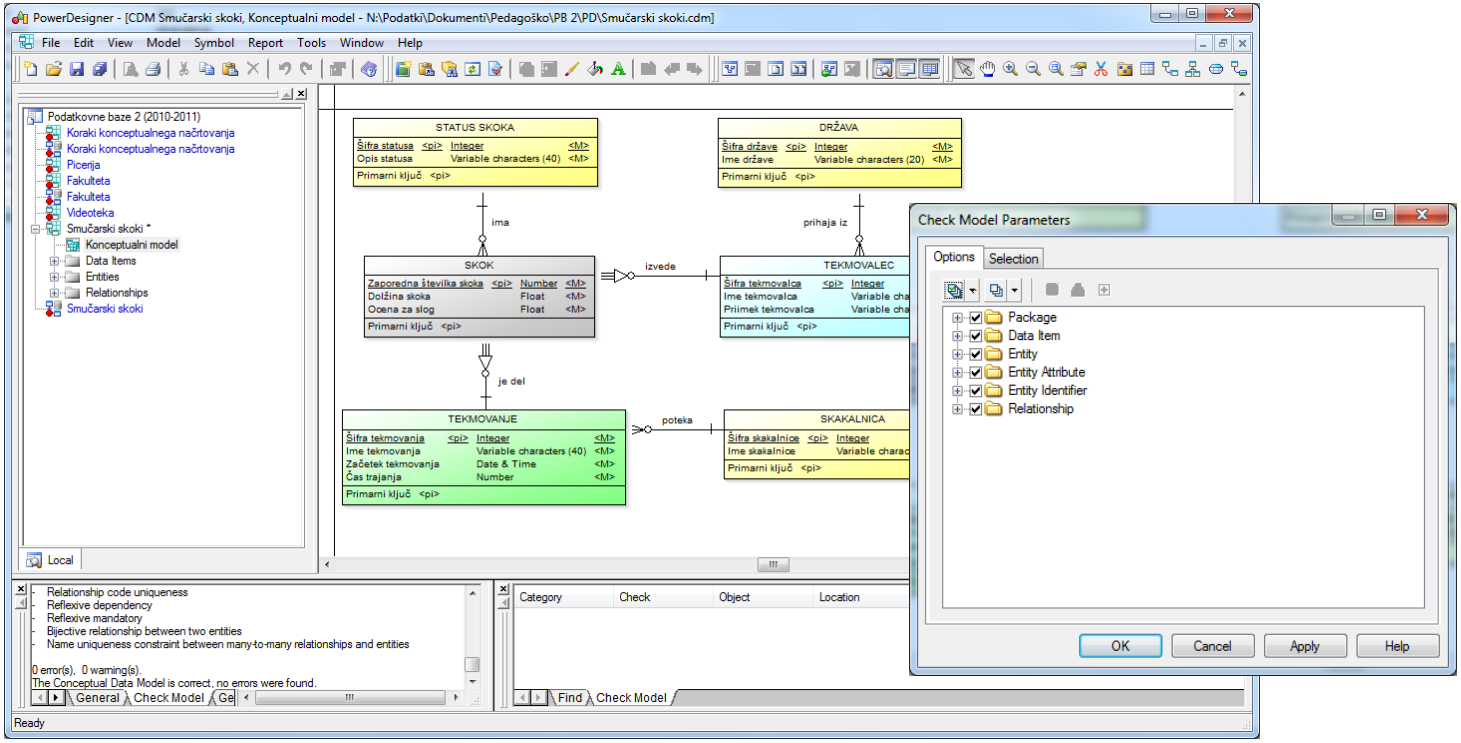
Omejitev tujega ključa

Pretvorba modela v fizičnega in kreiranje SQL skripte

PowerDesigner in SQL Workbench demo

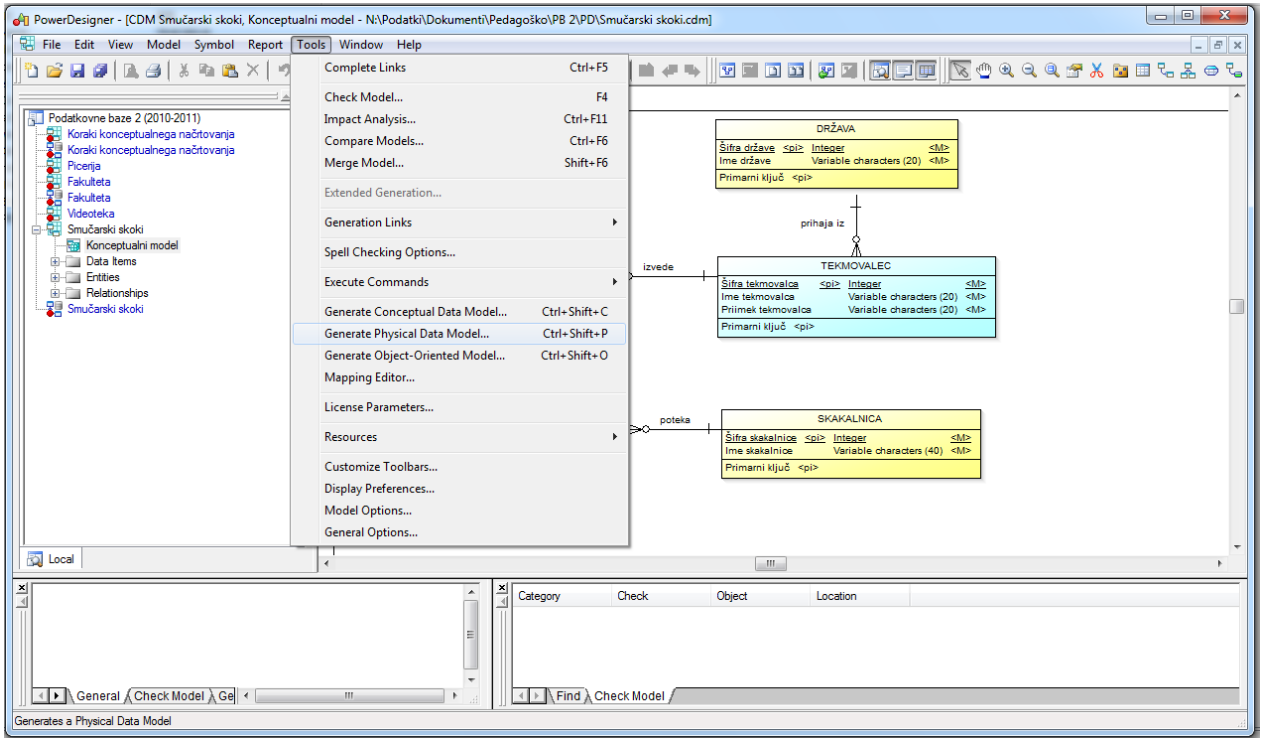
Demo (K3.1) (1)

- Najprej **preverimo** konceptualni model.



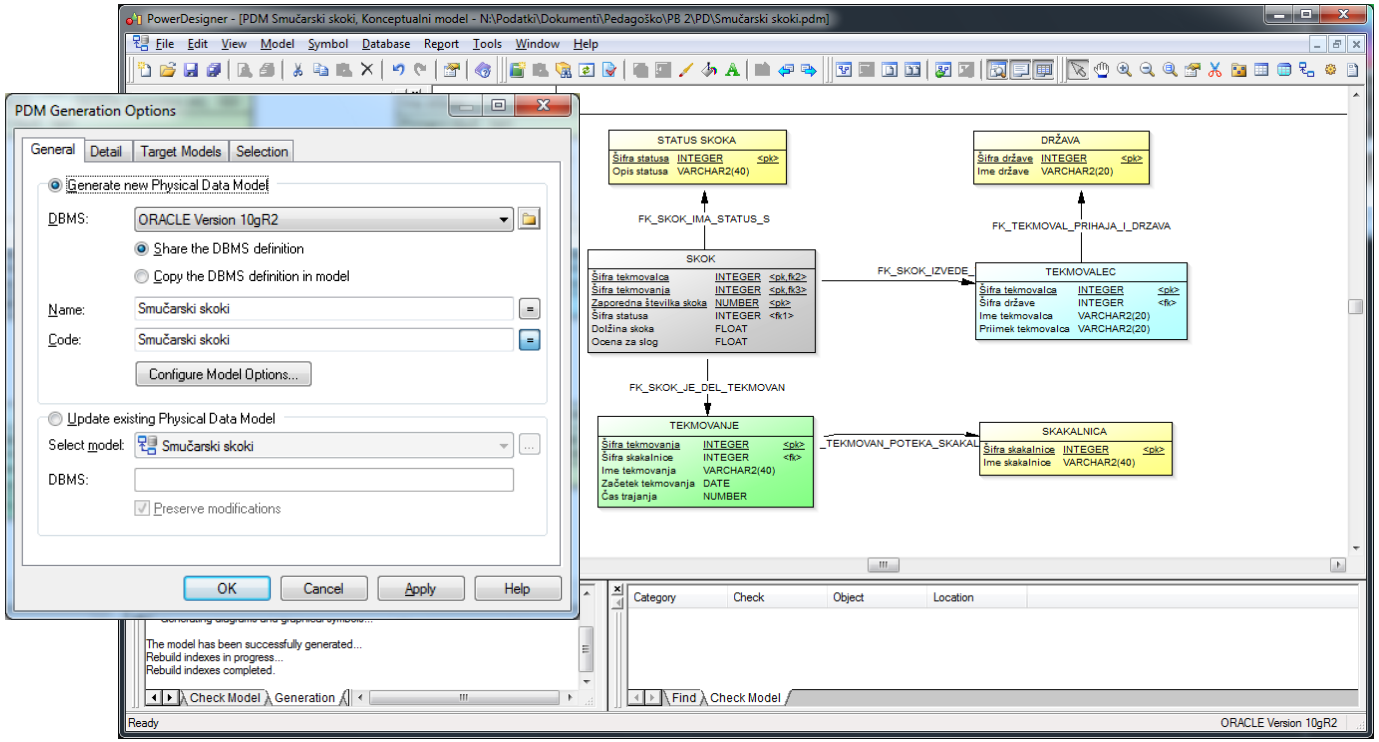
Demo (K3.1) (2)

- Konceptualni model **preslikamo** v fizičnega.



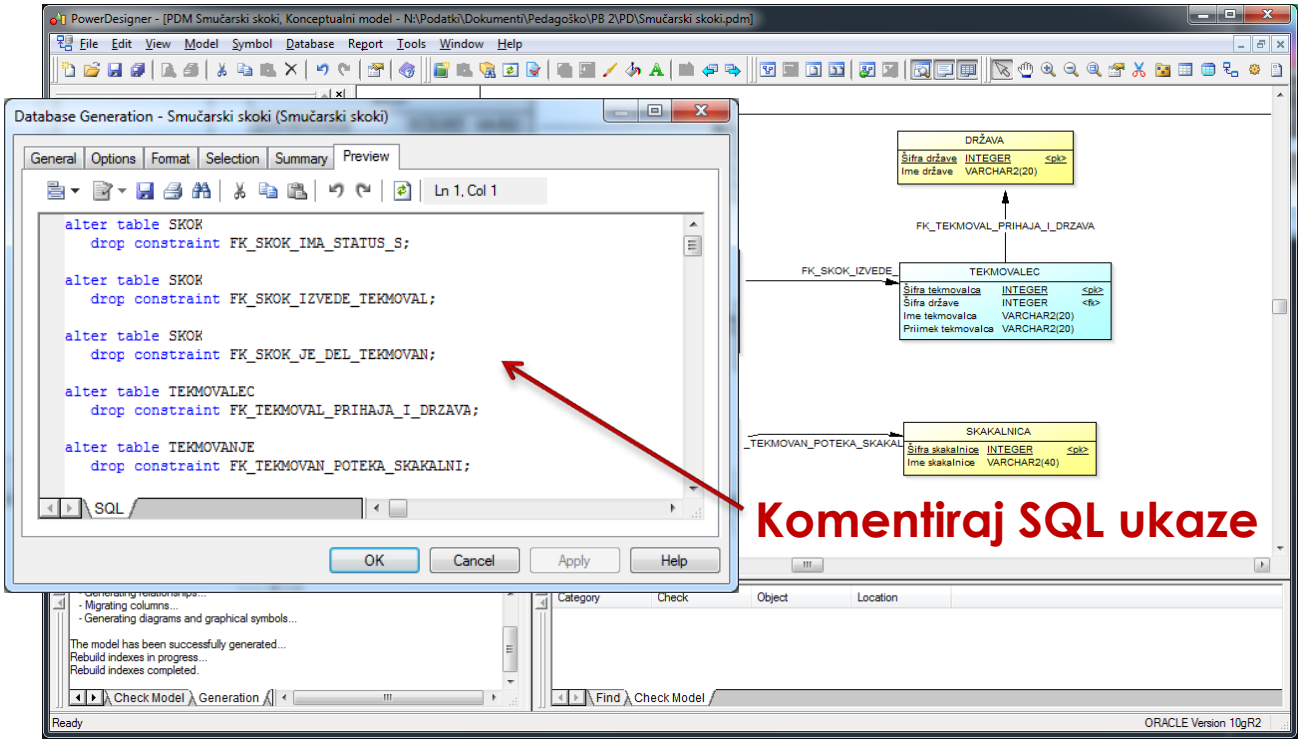
Demo (K3.1) (3)

- Rezultat je **fizični model** za **Oracle 10gR2**.



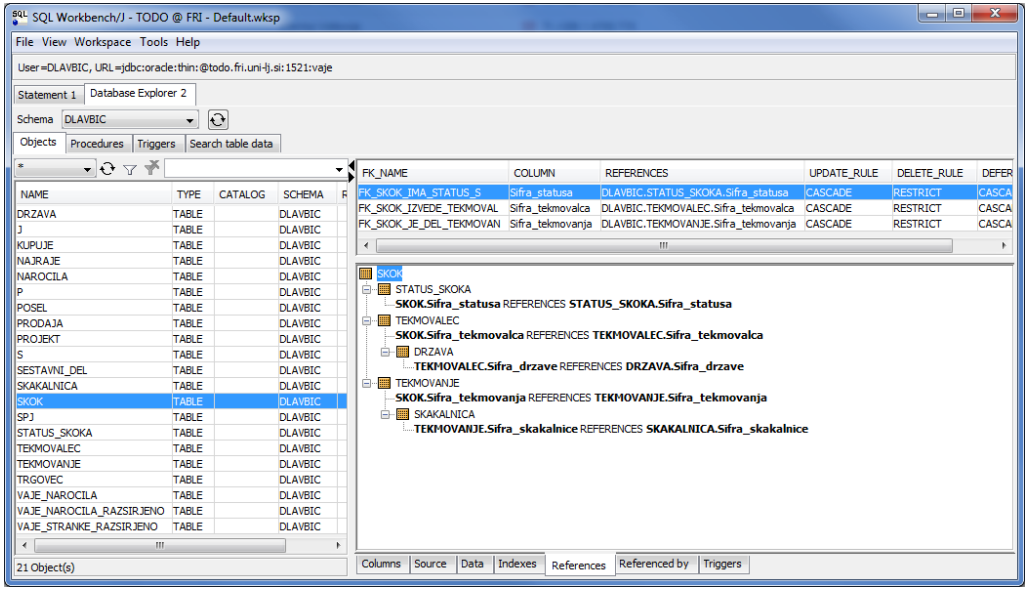
Demo (K3.1) (4)

- Nato kreiramo **SQL skripto**.



Demo (K3.1) (5)

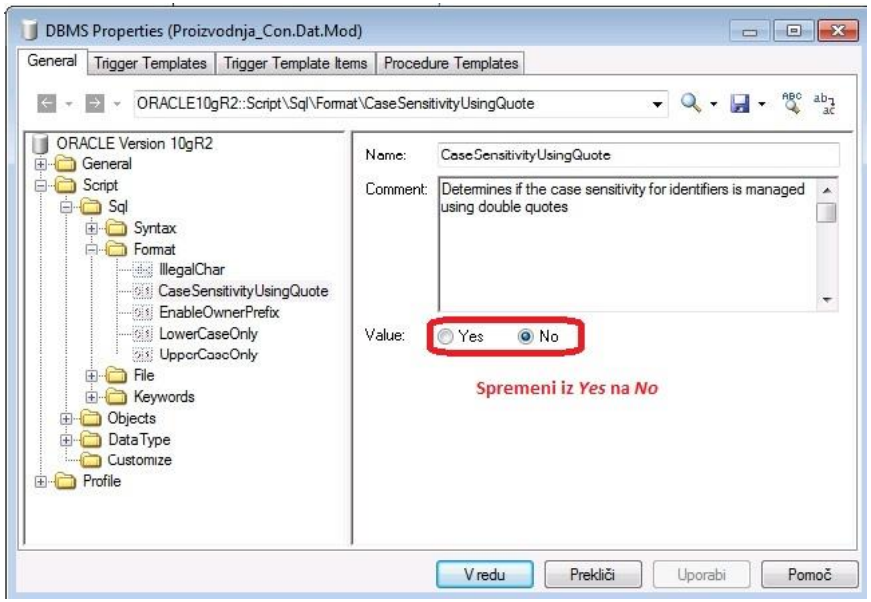
- Pokaži **izvajanje skripte** v SQL Workbench in rezultat v **Database Explorer**.
  - Prikaži atribute, indekse, reference (gnezdenje).



Demo (K3.1) (5)

- Kako se znebimo dvojnih narekovajev:

Iz logičega modela ustvari fizični model. Ko imaš narejen fizični model najdi v meniju "Database" -> "Edit Current DBMS". Pod zavihkom "General" pojdi na : "Script \ Sql \ Format \ CaseSensitivityUsingQuote" in nastavi value iz Yes na No.



Koraki fizičnega načrtovanja (K3.2)

- K3: **Pretvori logični model v jezik za ciljni SUPB**
  - K3.2: **Izdelaj načrt predstavitve izpeljanih atributov**

- Vrednosti izpeljanih atributov **se izračunajo** iz vrednostih obstoječih atributov.
  - Izpeljani atributi se po navadi ne nahajajo v logičnem modelu, so pa dokumentirani v podatkovnem slovarju.
  - **Problem je posodabljanje** → vrednosti posodobiti vsakič, ko se spremenijo vred. odvisnih atributov.

Poizvedbe s shranjenim in izračunanim atributom

SQL Workbench demo

Demo (K3.2) (1)

- Pokaži tabelo **VAJE_NAROCILA**, ki vsebuje 450.000 zapisov.

```
SELECT  *
FROM    VAJE_NAROCILA
WHERE   ROWNUM <= 50;
```

```
SELECT  COUNT(*)
FROM    VAJE_NAROCILA;
```

```
DESC    VAJE_NAROCILA;
```

- Pokaži iste stvari (**atribute** in **vrednosti**) ter **velikost tabele** s pomočjo **Database Explorer** in zavihka **Data**.

Demo (K3.2) (2)

- V tabelo VAJE_NAROCILA želimo dodati izpeljan atribut **CENA_SKUPAJ**.
  - Formula za izračun je enostavna: **CENA_SKUPAJ = CENA_NA_ENOTO × KOLICINA × (1 - POPUST)**

Demo (K3.2) (3)

- Kreirajmo novo tabelo **MOJA_NAROCILA_1**, ki bo kopija tabele **VAJE_NAROCILA** in vsebuje izpeljan atribut **CENA_SKUPAJ**.

```
DESC    VAJE_NAROCILA;
```

```
CREATE  TABLE   MOJA_NAROCILA_1   AS
SELECT  ID_NAROCILA,  ID_IZDELKA, CENA_NA_ENOTO, KOLICINA, POPUST,
        CENA_NA_ENOTO*KOLICINA*(1-POPUST) AS CENA_SKUPAJ
FROM    VAJE_NAROCILA;
```

```
SELECT  *
FROM    MOJA_NAROCILA_1
WHERE   ROWNUM <= 50;
```

Demo (K3.2) (4)

- Drugi način je da najprej kopiramo tabelo VAJE_NAROCILA v **MOJA_NAROCILA_2** in naknadno spremenimo definicijo tabele.

```
CREATE TABLE MOJA_NAROCILA_2 AS
SELECT *
FROM VAJE_NAROCILA;
```

```
SELECT *
FROM MOJA_NAROCILA_2
WHERE ROWNUM <= 50;
```

```
ALTER TABLE MOJA_NAROCILA_2
ADD (CENA_SKUPAJ number);
```

```
SELECT *
FROM MOJA_NAROCILA_2
WHERE ROWNUM <= 50;
```

```
UPDATE MOJA_NAROCILA_2
SET CENA_SKUPAJ =
CENA_NA_ENOTO * KOLICINA *
(1-POPUST);
```

```
SELECT *
FROM MOJA_NAROCILA_2
WHERE ROWNUM <= 50;
```

Demo (K3.2) (5)

- Na dva načina smo kreirali tabelo (MOJA_NAROCILA_1 in MOJA_NAROCILA_2), ki vsebuje **shranjen izpeljan atribut**.
- Primerjajmo s poizvedbo kakšne so performance pri shranjenem izpeljanem atributu in pri **izračunanem izpeljanem atributu**.

Demo (K3.2) (6)

- 1. način: **shranjen** izpeljan atribut

```
SELECT  MAX(CENA_SKUPAJ) AS NAJVECJA_SKUPNA_CENA
FROM    MOJA_NAROCILA_1;
```

0,06 s

(3x hitreje)

- 2. način: **izračunan** izpeljan atribut

```
SELECT  MAX(CENA_NA_ENOTO*KOLICINA*(1-POPUST)) AS
        NAJVECJA_SKUPNA_CENA
FROM    VAJE_NAROCILA;
```

0,19 s

Koraki fizičnega načrtovanja (K3.3) (1)

- K3: **Pretvori logični model v jezik za ciljni SUPB**
  - K3.3: **Izdelaj načrt splošnih omejitev**

- SUPB Oracle ne omogoča opredelitve splošnih omejitev s pomočjo SQL stavka.
- Lahko pa opredelimo prožilec **(trigger)** na ravni tabele.

Koraki fizičnega načrtovanja (K3.3) (2)

- Kot primer bomo dodali prožilec za izračun vrednosti atributa **CENA_SKUPAJ** na tabelo **MOJA_NAROCILA_1**.

Kreiranje prožilca na tabelo

SQL Workbench demo

Demo (K3.3) (1)

- Najprej **dodajmo** v tabelo MOJA_NAROCILA_1 **nov zapis**, kjer lahko opazimo, da se CENA_SKUPAJ ne izračuna samodejno.

```
INSERT    INTO   MOJA_NAROCILA_1
          (ID_NAROCILA, ID_IZDELKA, CENA_NA_ENOTO, KOLICINA, POPUST)
VALUES    (1, 1, 500, 20, 0);
```

```
SELECT    *
FROM      MOJA_NAROCILA_1
WHERE     ID_IZDELKA = 1;
```

Demo (K3.3) (2)

- Nato kreirajmo **prožilec**, ki **izračuna vrednost atributa CENA_SKUPAJ**.

```
CREATE OR REPLACE TRIGGER izracunajSkupnoCeno
    BEFORE INSERT OR UPDATE ON MOJA_NAROCILA_1
    FOR EACH ROW
    BEGIN
        :new.CENA_SKUPAJ := :new.CENA_NA_ENOTO*:new.KOLICINA*
        (1-:new.POPUST);
    END;
/
```

- Pokaži še ravno kreirani prožilec v **Database Explorer**.

Demo (K3.3) (3)

- In ponovno **dodajmo** v tabelo MOJA_NAROCILA_1 **nov zapis**, kjer opazimo, da se **CENA_SKUPAJ** sedaj **izračuna samodejno**.

```
INSERT    INTO   MOJA_NAROCILA_1
                 (ID_NAROCILA, ID_IZDELKA, CENA_NA_ENOTO, KOLICINA, POPUST)
VALUES    (1, 1, 300, 15, 0);
```

```
SELECT    *
FROM      MOJA_NAROCILA_1
WHERE     ID_IZDELKA = 1;
```

Koraki fizičnega načrtovanja (K4.1) (1)

- K4: **Izdelaj načrt datotečne organizacije ter indeksov**
  - K4.1: **Analiziraj transakcije**

- Potrebno je **razumeti namen transakcij** in **analizirati najpomembnejše**.
  - Pravilo 80/20
  - Uporaba različnih metod:
    - Matrika „transakcija/relacija“
    - Diagram uporabe transakcij

Koraki fizičnega načrtovanja (K4.1) (2)

- Podrobna analiza transakcij:
  - **Identifikacija relacij** in **atributov**, do katerih transakcija dostopa in **tip dostopa** (INSERT, UPDATE, DELETE, QUERY).
  - Identifikacija **atributov**, ki nastopijo **v predikatih** (WHERE del SQL stavka)
    - Preverjanje **ujemanja** (npr. LIKE '%FRI%')
    - **Iskanje po območju** (npr. BETWEEN 100 AND 200)
    - **Točno ujemanje** (npr. popust = 0,1)
  - Ti atributi so **kandidati za indekse**!

Koraki fizičnega načrtovanja (K4.1) (3)

- Podrobna analiza transakcij:
  - Identifikacija atributov, ki imajo v poizvedbi povezovalno vlogo med dvema ali več relacijami.

```
SELECT   *
FROM     OSEBA, POSTA
WHERE    OSEBA.Post_st = POSTA.Post_st;
```

  - Ti atributi so **kandidati za indekse**!

Koraki fizičnega načrtovanja (K4.1) (4)

- Podrobna analiza transakcij:
  - Opredeliti **pričakovane frekvence** izvajanja transakcij,
  - Opredelitev zahtevanih **časovnih performanc** transakcije (npr. transakcija naj se izvede v 2 s).
  - **Atributi**, ki nastopajo **v več predikatih** imajo **prioriteto pri izboru indeksov**!

Koraki fizičnega načrtovanja (K4.2) (1)

- K4: **Izdelaj načrt datotečne organizacije ter indeksov**
  - K4.1: **Izberi datotečno organizacijo**

- Datotečna organizacija mora biti učinkovita za vse osnovne relacije.
  - Nekatere datotečne organizacije so bolj primerne za polnjenje PB in so za poizvedovanje neprimerne.

Koraki fizičnega načrtovanja (K4.2) (2)

- Večina SUPB-jev nam daje zelo malo možnosti pri izbiri datotečne organizacije in SUPB-ji jih tudi različno podpirajo.
- Najbolj pogoste implementacije so:
  - **Kopica** (Heap),
  - **Hash**,
  - **ISAM** (Metoda indeksiranega zaporednega dostopa),
  - **B+ drevo**
  - **Gruča** (Cluster)

Koraki fizičnega načrtovanja (K4.3) (1)

- K4: **Izdelaj načrt datotečne organizacije ter indeksov**
  - K4.3: **Določi indekse**

- Potrebno je ugotoviti ali lahko z dodatnimi indeksi povečamo učinkovitost sistema.
  - Dve skrajnosti: ne indeksiramo nič oz. indeksiramo po vseh atributih.
  - Kaj so prednosti in slabosti?

Koraki fizičnega načrtovanja (K4.3) (2)

- Osnovni pojmi:
  - **Primarni indeks** je indeks po poljih, ki vsebujejo primarni ključ.
  - **Sekundarni indeks** je indeks, ki ne temelji na poljih, ki bi vsebovala primarni ključ.
  - **Indeks gruče** je indeks po poljih, po katerih je urejena tudi podatkovna datoteka
- Za vsako datoteko velja, da lahko ima:
  - **Primarni indeks** ali **indeks gruče** in
  - **več sekundarnih indeksov**.

Koraki fizičnega načrtovanja (K4.3) (3)

- Če ugotovimo, da bi indeksi izboljšali performance imamo na voljo **2 pristopa**:
  - Za hranjenje relacij uporabimo **neurejeno datoteko** in kreiramo **več sekundarnih indeksov**.
  - Zapise uredimo tako, da opredelimo **primarni indeks** ali **indeks gruče**.

Koraki fizičnega načrtovanja (K4.3) (4)

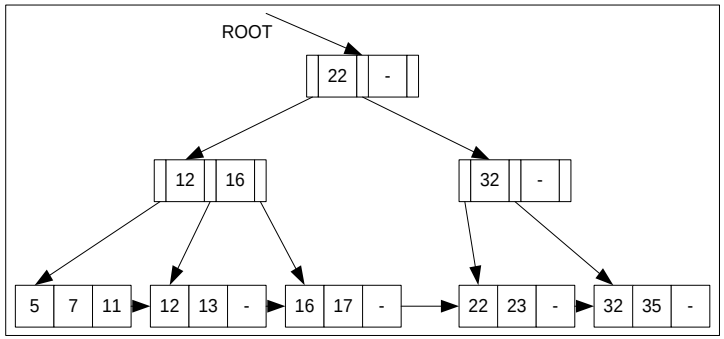
- Smernice za izbiro sekundarnih indeksov:
  - Če datoteka ni urejena po primarnem ključu, kreiraj **indeks na osnovni primarnega ključa**.
  - Če je tuj ključ pogosto v uporabi, dodaj **sekundarni indeks na tuj ključ**.
  - **Sekundarni indeks** na atributih v pogojih za **selekcijo**, **stik** in **sortiranje** (ORDER BY, GROUP BY, UNION, DISTINCT itd.)
  - **Majhnih relacij ne indeksiramo**.

Koraki fizičnega načrtovanja (K4.3) (5)

- Kdaj ne indeksiramo?
  - **Majhnih relacij** ne indeksiramo.
  - **Atributov**, ki se **pogosto spreminjajo**, ne indeksiramo.
  - Atributov v relacijah, kjer se pogosto izvajajo **poizvedbe**, ki v rezultatu **vključujejo večino zapisov**, ne indeksiramo.
  - **Atributi**, ki vsebujejo **daljše nize**, niso primerni kandidati za indeksiranje.

Koraki fizičnega načrtovanja (K4.3) (6)

- SUPB Oracle pozna indeksiranje z:
  - **B+ drevesi** in



  - **bitne indekse**.

Ime	Priimek	Pozicija
David	Ford	Manager
John	White	Supervisor
Tim	Becker	Manager

Manager	Supervisor
1	0
0	1
1	0