IT-Employees

IT-ID	IT-NAME	IT-LOGIN	SALARY
53831	John	john.it	13
53832	Alex	alex.it	18
53666	Jones	jones.it	17
53901	Claire	claire.it	17
53902	Ann	ann.it	11
53903	Mary	mary.it	12
53904	Paul	paul.it	19
53905	Marcus	marcus.it	20
53906	Daphne	daphne.it	19
53907	Ugo	ugo.it	17
53688	Smith	smith.it	15
53650	Simon	simon.it	11
54001	Eric	eric.it	14
54005	Franco	franco.it	12
54010	Gil	gil.it	19
54023	Emil	emil.it	14
54021	Alex	alex2.it	17
54034	Paul	paul2.it	11
54012	Jef	jef.it	13
54045	Ian	ian.it	14
54056	Adam	adam.it	13

Admin-Employees

AD-ID	AD-NAME	AD-LOGIN	SALARY
41831	Peter	peter.ad	14
41832	Carlo	carlo.ad	17
41666	Ben	ben.ad	11
41901	Jones	jones.ad	17
41902	France	france.ad	13
51831	Willy	willy.ad	13
51832	Kevin	kevin.ad	12
51666	Rose	rose.ad	17
51901	Jones	jones2.ad	13
51902	Emma	emma.ad	12
51903	July	july.ad	18
51904	Lory	lory.ad	16
50905	Sarha	sarha.ad	16
50906	Zorro	zorro.ad	19
50907	Smith	smith.ad	17
50688	Zeta	zeta.ad	13
51650	Naomi	naomi.ad	12
51001	Derek	derek.ad	12
51005	David	david.ad	15
51009	Helen	helen.ad	12

Figure 1: An instance of the `IT-Employees` and `Admin-Employees` relations

1 Relational Algebra and SQL Queries

For each of the following Relational Algebra Expressions show an equivalent SQL query.

1. $\sigma_{\texttt{IT-NAME}=\text{``}\texttt{Alex}\text{''}}(\texttt{IT-Employees})$
2. $\Pi_{\texttt{IT-ID},\texttt{IT-NAME}}(\sigma_{\texttt{Salary}=17}(\texttt{IT-Employees}))$.

Given a database containing the two relations in Figure 1 show the result of the following Relational Algebra query (the resulting tuples should be presented in a table headed by the attributes names of the result):

$$\Pi_{\texttt{IT-NAME},\texttt{AD-NAME},\texttt{SALARY}}(\sigma_{\texttt{IT-NAME}=''\texttt{Paul}''}(\texttt{IT-Employees}) \bowtie \texttt{Admin-Employees}).$$

2 Indexes for Selection Queries

Consider the instance of the `Student` relation shown in Figure 2. Assume that each Disk Block can store up to three tuples. Build the following Indexes for the `Student` relation. Show each Index with pointers to Disk Blocks storing the relation. For each Disk Block storing the relation show just `S-ID` and `AGE` attributes.

1. A secondary index with indirect buckets on `AGE`.
2. A primary dense index on `AGE`.

You are now given the following query: $\sigma_{\texttt{AGE}=18}(\texttt{Student})$. Compute the actual number of disk I/O for answering the query in the following cases (assume that the index resides in Main Memory):

3. Using an Index-Based Selection algorithm that uses the secondary index as built in the question 1;
4. Using an Index-Based Selection algorithm that uses the primary index as built in the question 2;
5. Using a One-Pass Selection algorithm without any index.

In the first two cases, compare the obtained actual cost with the formal cost analysis for the *Index-Based Selection Algorithm*. When a discrepancy between the formal cost and the actual cost occurs give a plausible explanation.

Student

S-ID	NAME	LOGIN	AGE	GPA
53831	Madayan	madayan@music	17	1.8
53832	Guldu	guldu@music	16	3.8
53666	Jones	jones@cs	18	3.4
53901	Jones	jones@math	18	3.4
53902	Jones	jones@physics	18	3.4
53903	Jones	jones@genetics	17	3.4
53904	Jones	jones@astro	17	3.4
53905	Jones	jones@chem	18	3.4
53906	Jones	jones@ai	18	3.4
53907	Smith	smith@music	19	3.8
53688	Smith	smith@ec	19	3.2
53650	Smith	smith@math	16	3.8
54001	Smith	smith@cs	16	3.5
54005	Smith	smith@physics	19	3.8
54009	Smith	smith@astro	16	2.2

Figure 2: An instance of the `Student` relation

3 B-Tree Construction

Build the B-Tree implementing a primary index on sequential file on the attribute `AD-ID` of the `Admin-Employees` relation of Figure 1. Assume that each B-Tree node can contain up to 4 keys and 5 pointers (Hint: Start with a single-node B-Tree that will be at the same time both the root and the leaf of the B-Tree. Split such node as a leaf). In particular:

1. Individuate each insertion of a key that generates a node splitting and show the status of the B-Tree before and after the insertion.

2. Show the resulting final B-Tree.

4 Evaluating Query Plans

You are given a relation R with schema $R(A, B, \ldots)$. The relation has 20.000 tuples stored in 1.000 Disk Blocks. An application requires to execute both the following two queries: $\sigma_{A=k_1}(R)$; $\sigma_{B=k_2}(R)$, with k_1, k_2 constant values.

To speed up the query processing you, as DB administrator, are free to build both primary and secondary indexes on top of R.

Specify the indexes that you would build (if any) that minimize the total number of disk I/O to run both queries in each of the cases below:

1. $V(R, A) = 10.000,\ V(R, B) = 20.$

2. $V(R, A) = 10,\ V(R, B) = 20.$

Justify your answer comparing the cost of the different query plans. Do not consider the cost to access indexes when estimating the cost of the different query plans.